Cryptografie

Cryptografie

Beveiliging van de digitale maatschappij

Gerard Tel

Universiteit Utrecht

Instituut voor Informatica en Informatiekunde
Universiteit Utrecht

Trefw: cryptografie, informatiebeveiliging, digitale handtekening, elektronisch geld

© Copyright 2006 Gerard Tel.

Alle rechten voorbehouden. Niets uit deze uitgave mag worden verveelvoudigd, opgeslagen in een geautomatiseerd gegevensbestand, of openbaar gemaakt, in enige vorm of op enige wijze, hetzij elektronisch, mechanisch, door fotokopieën, opnamen, of enige andere manier, zonder voorafgaande toestemming van de auteur.
Voorzover het maken van kopieën uit deze uitgave is toegestaan op grond van artikel 16B Auteurswet 1912 j^{o} het Besluit van 20 juni 1974, St.b. 351, zoals gewijzigd bij Besluit van 23 augustus 1985, St.b. 471 en artikel 17 Auteurswet 1912, dient men de daarvoor wettelijk verschuldigde vergoedingen te voldoen aan de Stichting Reprorecht. Voor het overnemen van gedeelte(n) uit deze uitgave in bloemlezingen, readers en andere compilatie- of andere werken (artikel 16 Auteurswet 1912), in welke vorm dan ook, dient men zich tot de auteur te wenden.

Ondanks alle aan de samenstelling van dit boek bestede zorg kan noch de redactie, noch de auteur, noch de uitgever aansprakelijkheid aanvaarden voor schade die het gevolg is van enige fout in deze uitgave.

Inhoudsopgave

Inhoudsopgave **v**

Voorwoord **viii**

1 Inleiding tot symmetrische encryptie **1**
1.1 Symmetrische en asymmetrische cryptografie 1
1.2 Eenvoudige voorbeelden en aanvallen 2
1.3 Informatietheoretische veiligheid 7
1.4 Kritieke lengte en computationele veiligheid 9
Samenvatting en conclusies 12
Opgaven bij hoofdstuk 1 13

2 Data Encryption Standard **15**
2.1 De werking van DES 15
2.2 DES breken met brutekracht 22
2.3 Differentiële cryptanalyse 25
2.4 Opkomst en ondergang van een standaard 28
Samenvatting en conclusies 30
Opgaven bij hoofdstuk 2 31

3 Advanced Encryption Standard (*Marries van de Hoef*) **33**
3.1 Ontstaan 34
3.2 Werking van AES 35
3.3 Gebruik 43
3.4 Kraken 44
Samenvatting en conclusies 46
Opgaven bij hoofdstuk 3 46

4 Symmetrische encryptie: vervolg **48**
4.1 Hellmans cubic-rootaanval 48
4.2 Productsystemen 51
4.3 Stroomversleuteling 55
4.4 Sleutelbeheer 59
4.5 AES: Advanced Encryption Standard 65
Samenvatting en conclusies 66
Opgaven bij hoofdstuk 4 67

5 Getallen: eigenschappen **70**
5.1 De gehele getallen . . . 71
5.2 Invoerblindering . . . 74
5.3 Modulaire aritmetiek . . . 76
5.4 Multiplicatieve groep: priemmodulus . . . 81
5.5 Modulaire ring: samengestelde modulus . . . 86
Samenvatting en conclusies . . . 88
Opgaven bij hoofdstuk 5 . . . 89

6 Getallen en public-keycryptografie **93**
6.1 Berekenbare functies in modulaire groepen . . . 94
6.2 Niet-berekenbare functies in modulaire groepen . . . 101
6.3 Public-keycryptosystemen . . . 105
Samenvatting en conclusies . . . 116
Opgaven bij hoofdstuk 6 . . . 117

7 Digitale handtekeningen **120**
7.1 Eisen en toepassingen . . . 120
7.2 Handtekeningschema's . . . 125
7.3 Hashfuncties . . . 128
7.4 Blinde handtekeningen . . . 134
Samenvatting en conclusies . . . 137
Opgaven bij hoofdstuk 7 . . . 138

8 Identificatie **140**
8.1 Definities en aanvallen, voorbeelden . . . 140
8.2 Authenticatie op basis van cryptografie . . . 145
8.3 Zero-knowledge proof voor identificatie . . . 148
8.4 Meer over zero-knowledge proofs . . . 155
8.5 Biometrie . . . 159
Samenvatting en conclusies . . . 162
Opgaven bij hoofdstuk 8 . . . 163

9 Verdeelschema's en groepscryptografie **165**
9.1 Inleiding: schema's en gebruik . . . 166
9.2 Een additief vetoschema . . . 174
9.3 Een drempelschema: Shamirs polynoomschema . . . 183
Samenvatting en conclusies . . . 189
Opgaven bij hoofdstuk 9 . . . 190

10 Toepassingen **193**
10.1 Kerberos . . . 193
10.2 Universal Mobile Telecommunication System (UMTS) . . . 200
10.3 Secure Socket Layer . . . 211
Samenvatting en conclusies . . . 214
Opgaven bij hoofdstuk 10 . . . 214

11 Regulering van cryptografie **216**
11.1 Totstandkoming van wetgeving . . . 216
11.2 Verkoop en export van cryptografie . . . 218
11.3 Regels rond geheimhouding . . . 219
11.4 Verdere wetgeving . . . 228
Samenvatting en conclusies . . . 230
Opgaven bij hoofdstuk 11 . . . 230

12 Smartcards **232**
12.1 Inleiding . . . 233
12.2 Aanvallen op smartcards . . . 236
12.3 Openen van smartcards . . . 240
Samenvatting en conclusies . . . 243
Opgaven bij hoofdstuk 12 . . . 243

13 Elektronisch geld **244**
13.1 Geld en eisen op elektronisch geld . . . 244
13.2 E-muntsystemen: Ecash . . . 253
13.3 Registersystemen . . . 260
Samenvatting en conclusies . . . 262
Opgaven bij hoofdstuk 13 . . . 264

14 Secure computing **266**
14.1 Inleiding . . . 267
14.2 Homomorfe encryptie . . . 272
14.3 Geheime verkiezingen . . . 276
14.4 Voorbij functie-evaluatie . . . 282
Samenvatting en conclusies . . . 288
Opgaven bij hoofdstuk 14 . . . 288

15 De discrete logaritme **290**
15.1 Groepen en het logaritmeprobleem . . . 292
15.2 Groepen voor cryptografie . . . 299
15.3 Groepsalgoritmen . . . 304
15.4 De indexcalculus . . . 313
Samenvatting en conclusies . . . 317
Opgaven bij hoofdstuk 15 . . . 317

A Kansrekening **319**
A.1 Experimenten en verwachting . . . 319
A.2 Bernoullireeksen . . . 321
A.3 Botsingen en de verjaardagsstelling . . . 323

B Verantwoording van afbeeldingen **325**

Bibliografie **327**

Index **335**

Voorwoord

De cryptografie staat de laatste jaren sterk in de belangstelling, hetgeen zich uit in een groot aanbod van colleges, cursussen en leerboeken. Dit boek verschilt in een aantal aspecten van het gangbare aanbod, en daarom meen ik dat het tussen het bestaande materiaal bestaansrecht heeft.

Veel boeken zijn erg wiskundig, behandelen getaltheorie en coderingen erg diepgaand, maar besteden dan weinig aandacht aan de berekeningsaspecten. Andere boeken behandelen juist wel berekeningen en protocollen, maar laten de wiskundige achtergronden onbesproken. Dit boek probeert een middenweg tussen een wiskundig en een informaticaboek te zoeken. In kaders, die los staan van de hoofdtekst, wordt steeds een link gelegd naar een toepassing of voorbeeld: soms historisch, soms modern, soms uit de praktijk, soms grappig bedoeld. Ik hoop hierdoor een breed lezerspubliek aan te spreken.

De cryptografie en dit boek. De gangbare definities of omschrijvingen van cryptografie zijn te eng om de tegenwoordige veelzijdige toepassingen van deze tak van wetenschap te dekken. Daarom doe ik een nieuwe poging het werkterrein van dit boek af te bakenen:

Definitie 0.1 *De cryptografie is de kunst van beveiliging met behulp van informatie.*

In deze omschrijving wordt *informatie* aangehaald als *middel* ter beveiliging en niet als beveiligings*object*, al kan uiteraard ook informatie zelf cryptografisch worden beveiligd. Geheimhouding van informatie is de oudste toepassing van cryptografie en ook bij het publiek het best bekend. Daarnaast vindt de cryptografie toepassing bij toegangsbeveiliging (tot computersystemen zowel als gebouwen), digitale handtekeningen, elektronisch geld, en noem maar op.

Wat staat er in dit boek. Dit boek is opgebouwd uit drie delen, die wel afhankelijk van elkaar zijn, maar niet elke lezer in gelijke mate of op dezelfde tijd zullen aanspreken.

Deel I: Geheimhouding. Het eerste deel, de hoofdstukken 1 tot 6, behandelt geheimhouding: de basis van cryptografische algoritmen, zoals gebruikt voor de bescherming van informatie tegen uitlekken en afluisteren. Er wordt aandacht besteed aan symmetrische algoritmen (zoals DES en encryptie in gsm-telefoons), aanvallen op dit soort cryptografie en toepassingen ervan, en public-keycryptografie.

Hoofdstuk 1 introduceert het versleutelen en ontsleutelen van informatie met een (gemeenschappelijke) sleutel. Informatietheorie wordt gebruikt om uit de verhouding tussen de sleutelgrootte en de hoeveelheid data grenzen op de veiligheid af te leiden. Hoofdstuk 2 behandelt het DES-algoritme, dat 25 jaar lang de standaard voor symmetrische encryptie is geweest.

Verdere technieken voor symmetrische encryptie komen in hoofdstuk 4 aan de orde. Hoofdstuk 5 is een excursie buiten de eigenlijke cryptografie: het behandelt de getaltheorie die aan de moderne cryptografische technieken ten grondslag ligt. Voor wiskundigen is veel hiervan bekend, terwijl de lezer met ICT-achtergrond er weinig zal tegenkomen over computers, bits en berekeningen. Er is naar gestreefd het materiaal zo duidelijk mogelijk te presenteren en te voorzien van voorbeelden die aantonen waarom deze theorie voor de cryptografie belangrijk is. Men kan ervoor kiezen dit hoofdstuk (of delen ervan) bij eerste lezing over te slaan en het als referentie te gebruiken wanneer de verdere tekst daar aanleiding toe geeft. In hoofdstuk 6 worden de wortels van de cryptografie blootgelegd. Onder andere wordt hier de vraag behandeld of het wel mogelijk is om informatie te gebruiken voor beveiliging. In dit hoofdstuk komen de wiskunde van getallen en de theorie van berekeningen bij elkaar.

Een inleidend college in de cryptografie zal over het algemeen dit deel geheel omvatten, mogelijk met uitzondering van enkele delen van hoofdstuk 5.

Deel II: Cryptografische protocollen. De hoofdstukken 7 tot 10 laten zien dat de cryptografie kan worden gebruikt voor zaken die verder gaan dan geheimhouding. Bijvoorbeeld het beschermen van (op zich bekende) informatie tegen oneigenlijk gebruik, maar ook toegangscontrole, digitaal geld en anonimiteit.

Hoofdstuk 7 gaat over de digitale handtekening, hashfuncties en blindering. Hoofdstuk 8 behandelt identificatie, waarbij ook *zero-knowledge proofs* en biometrie aan de orde komen. In hoofdstuk 9 wordt het verdelen van informatie behandeld: hierbij krijgen verschillende partijen elk een aandeel in bijvoorbeeld een cryptografische sleutel, zodat ze die alleen samen kunnen gebruiken. In elk van deze drie hoofdstukken speelt de asymmetrische cryptografie de belangrijkste rol, maar ook met symmetrische cryptografie kunnen grootschalige systemen met hoge veiligheid worden gebouwd. De protocollen die rond symmetrische cryptogafie worden opgebouwd lenen zich echter minder goed voor een algemene behandeling, en daarom worden er in hoofdstuk 10 enkele specifieke systemen behandeld: Kerberos, UMTS en SSL.

Voor een inleidend college in de cryptografie kan uit deze hoofdstukken een keuze worden gemaakt, afhankelijk van interesse van de docent of de studenten.

Deel III: Vervolgstudie. De hoofdstukken 11 tot 15 behandelen uiteenlopende zaken, eigenlijk te divers om in een college te behandelen. Dit deel is bedoeld voor de geïnteresseerde lezer of als inleiding tot een verdere studie van het onderwerp. Het materiaal is dan ook in minder detail uitgewerkt dan in de eerdere hoofdstukken; delen van bewijzen en het leggen van verbanden worden meer aan de lezer overgelaten.

De lezers met bedrijfskundige of financiële interesse kunnen de hoofdstukken 11 (over regelgeving rond cryptografie) en 13 (over elektronisch geld) bestuderen. Lezers met een ICT-achtergrond zullen hoofdstuk 12 over de techniek van smartcards kunnen waarderen. Wie hoofdstuk 9 (over verdeelschema's) mooi vond, zal ook van hoofdstuk 14 (over multipartyberekeningen) kunnen genieten. De wiskundigen ten slotte worden in hoofdstuk 15 verwend met een verhandeling over groepentheorie en het berekenen van logaritmen.

Voelt u zich door een van deze hoofdstukken niet aangesproken? Geen nood, zet het boek in de boekenkast en raadpleeg het als u later toch in een van deze onderwerpen belangstelling krijgt.

Voor wie is dit boek bedoeld. Dit boek kan worden gebruikt voor colleges in de cryptografie aan universitaire opleidingen informatica en wiskunde. Qua niveau zal de stof daar

passen in het tweede of derde leerjaar (de bachelorfase). Ook in het HBO zal dit boek nuttig kunnen zijn voor gespecialiseerde opleidingen.

Dankbetuiging. Ik wil iedereen bedanken die aan de totstandkoming van dit boek heeft meegewerkt. De studenten van het college Cryptografie in het najaar van 2001 (met name Robbert Brak, Bart Groot en Alain Böhm) zijn erg behulpzaam geweest in het opsporen van typefouten. Marries van der Hoef heeft een heel hoofdstuk geschreven (Hfst. 3). De volgende personen hebben een of meer hoofdstukken becommentarieerd. Dr. Marjan van den Akker (Hfst. 13 en bijlage A), Drs. Jan Brands (Hfst. 1 tot 5), Dr. Hans Bodlaender (Hfst. 1 tot 9 en 14), Mr. dr. Jaap Dijkstra (Hfst. 11), Dr. Rogier van Eijk (Hfst. 11), Dr. Jaap-Henk Hoepman (Hfst. 6, 9, 10 en 13), Dr. Han Hoogeveen (Hfst. 6), Dr. Bert-Jaap Koops (Hfst. 11), Drs. Sieuwert van Otterloo (Hfst. 15), Dr. Berry Schoenmakers (Hfst. 14).

Bij de herziene versie van Oktober 2006. De eerste versie van het boek wordt sinds het voorjaar van 2006 niet meer geleverd door de uitgever. Daarom is, voor het college *Cryptografie* in het najaar van 2006 en later, en voor het college *Security* vanaf 2012, deze uitgave gedrukt. Ten opzichte van de eerste uitgave zijn enkele typefouten verbeterd en een paar inhoudelijke aanvullingen gemaakt.

Gerard Tel,
Utrecht, maart 2012.

Hoofdstuk 1

Inleiding tot symmetrische encryptie

Cryptografie staat de laatste jaren sterk in de belangstelling en gezien de vele toepassingen is dat niet verwonderlijk. De opkomst van internet en zelfs internetwinkelen maken een goede beveiliging noodzakelijk, en in het dagelijks betalingsverkeer heeft de electronische portemonnee (chipknip) naast het oudere pinnen zijn intrede gedaan. In alle gevallen speelt cryptografie de hoofdrol bij de beveiliging.

Tegenwoordig is de cryptografie een sterk wiskundig terrein, waar vooral getaltheorie een grote rol speelt; de moderne technieken zijn namelijk allemaal op het manipuleren van getallen gebaseerd. De klassieke technieken, de zogenaamde *symmetrische* algoritmen, maken hiervan geen gebruik en worden in dit boek het eerst behandeld.

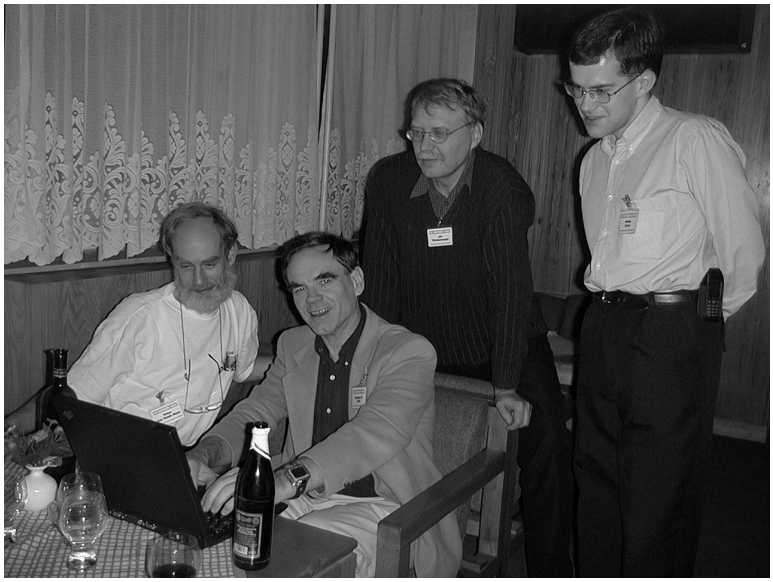
De bescherming van informatie berust op de geheimhouding van een doorgaans kleinere hoeveelheid informatie, de *sleutel* of *key*. Een cryptografisch algoritme zal zorgen voor een goede vermenging van de informatie en de sleutel, maar is zelf (meestal) niet geheim. Een belangrijk onderwerp in dit hoofdstuk is daarom een analyse van de mate van bescherming die met een sleutel van een zekere lengte mogelijk is.

1.1 Symmetrische en asymmetrische cryptografie

Een eerste toepassing van de cryptografie is het *versleutelen* (ook *vercijferen*, Engels: *encrypt*) van informatie die vertrouwelijk moet blijven. Alice en Bob willen elkaar iets meedelen maar zij vrezen dat hun opponent Oscar hun communicatie beluistert. Ze gaan het afluisteren voor Oscar zinloos maken door elke boodschap voor het versturen te veranderen door toepassing van een functie E (voor *encryptie*). In plaats van de boodschap X zendt Alice dus $Y = E(X)$; de oorspronkelijke boodschap X heet de *klare tekst* (*plaintext*), de versleutelde boodschap Y de *cijfertekst* (*ciphertext*). Natuurlijk moet Bob wel, maar Oscar niet over de inverse functie D (voor *decryptie*) beschikken om de boodschap te kunnen *ontsleutelen* (of *ontcijferen*, Engels: *decrypt*).

De encryptie- en decryptieprogramma's (of machines) zijn altijd parametriseerbaar: dat betekent dat je ze kunt 'instellen' op een element van een klasse van functies. De instelling noemen we de *sleutel* of *key*, weergegeven als k en de functies zijn dan E_k en D_k. Alice en Bob

Kader 1.1: Informatiebeveiliging in laptops



Bij het beveiligen van informatie wordt meestal als eerste gedacht aan informatie die via een netwerk wordt verzonden. Bestanden op een vaste schijf of diskette zijn echter ook verre van veilig, zelfs als de computer niet of alleen via een *firewall* aan het net is gekoppeld. Diskettes met gevoelige informatie kunnen gemakkelijk kwijtraken (bij diefstal van auto of tas bijvoorbeeld), en jaarlijks wisselen duizenden laptops onbedoeld van eigenaar. Niet alleen is dan de hardware verloren, maar de informatie in de laptop valt ook in andere handen en dit is in vele gevallen een nog grotere schadepost. Zelfs justitie kwam hierdoor enkele malen in opspraak. Het kan daarom verstandig zijn, alle vertrouwelijke informatie in versleutelde vorm op de disks op te slaan. Programma's als PGP bieden deze functionaliteit.

moeten nu de sleutels zo kiezen, dat de ingestelde functies elkaar opheffen. We onderscheiden nu twee typen algoritmen, afhankelijk van hoe die omkeerbaarheid wordt bereikt.

In een *symmetrisch* algoritme bestaat de encryptie uit een reeks *omkeerbare* stappen, zoals het optellen van een bepaald getal, het verwisselen van bepaalde bits, of het puntsgewijs XOR-en met een bitrij. Dit betekent dat kennis van het versleutelproces voldoende informatie inhoudt om de operaties in omgekeerde volgorde terug te rekenen en het origineel te herstellen. Daarom is bij deze systemen de encryptiesleutel en decryptiesleutel gelijk (al wordt hij soms anders toegepast).

Bij *asymmetrische* algoritmen maken we gebruik van operaties die niet omkeerbaar zijn: denk bijvoorbeeld aan een reductie (deling met rest) modulo een groot getal n. Je kunt van een getal x gemakkelijk berekenen wat de rest is bij deling van x^2 door n, maar aan die rest kun je niet meer zien wat x^2 (of x) was en daarmee is deze stap niet direct omkeerbaar. Natuurlijk bestaat er wel een berekening die het origineel terug kan vinden, anders was het cryptosysteem onbruikbaar. Maar hoe de berekening van D verloopt, is aan de encryptiesleutel niet te zien. Omdat er twee verschillende sleutels worden gebruikt, noemen we deze klasse van algoritmen asymmetrisch, en omdat de encryptiesleutel niet geheimgehouden hoeft te worden (Oscar kan er immers toch niets mee beginnen) spreken we ook wel van *public-keycryptografie*.

1.2 Eenvoudige voorbeelden en aanvallen

Omdat we voor getallen allerlei bewerkingen beschikbaar hebben, is het soms gemakkelijk de klare tekst op te vatten als een reeks getallen. We rekenen dan modulo m, vaak de grootte

van het alfabet. In de volgende voorbeelden gaan we uit van een alfabet van 32 tekens, en rekenen derhalve in $\mathbb{Z}_{32}$ (met de getallen 0 tot en met 31).

De elementaire eenheid van informatie is een *bit*, waarvan de waarde alleen 0 of 1 kan zijn. Om meer verschillende waarden te representeren worden meer bits gebruikt, met drie bits kunnen bijvoorbeeld al de acht waarden 000, 001, 010, 011, 100, 101, 110 en 111 worden weergegeven. De vroegere telexmachines werkten met 5-bits tekens en gebruikten een alfabet met 32 tekens. De 0 is spatie, 1 tot en met 26 zijn a tot en met z, 27 en 28 zijn punt en komma, 29 en 30 zetten de speciale tekens aan en uit. De laatste code, 31, is een *no-op*: deze code drukt geen enkel teken af en werd gebruikt voor het corrigeren van fouten. Had men namelijk een verkeerd teken in een papierstrook geponst, dan kon men dit corrigeren door enkele plaatsen terug te gaan, op de foute plaatsen vijf gaatjes te ponsen, en daarna verder te gaan.

Men laat de geschiedenis van de cryptografie meestal beginnen bij de Romeinse keizer Caesar, die van alle geheime berichten die hij verzond de letters over drie plaatsen in het alfabet liet verschuiven. Dus werd, in dit *Caesariaans systeem*, een A een D, een B een E et cetera. Deze conventie was bij de ontvanger bekend en kon gemakkelijk ongedaan worden gemaakt. Dit Caesariaanse systeem gebruikte dus slechts één encryptiefunctie en één bijbehorende decryptiefunctie en voldeed daarom niet aan de eigenschap van parametriseerbaarheid.

Het nadeel hiervan is gemakkelijk in te zien: de *methode* als geheel moet geheim blijven, want als de encryptie-decryptiemethode eenmaal is uitgelekt dan is die vanaf dat moment waardeloos. Het is daarom beter om niet een *geheime methode* te gebruiken, maar een *klasse* van functies waaruit een *geheime sleutel* een bepaald element specificeert.

1.2.1 Drie primitieve systemen

Het *schuifsysteem* (*shift system*, ook nog wel als Caesariaans aangeduid) is een generalisatie van Caesars methode doordat de verschuiving door het alfabet over een variabel aantal posities wordt gedaan. Het gebruikt een sleutel $k \in \mathbb{Z}_{32}$ en verschuift alle tekens k posities op in het alfabet, dus:

$$E_k(x) = x + k.$$

Een voorbeeld: met $k = 3$ zal de tekst **APEKOP** worden versleuteld tot *DSHNRS* en uiteraard zal Oscar de subtiliteit van deze mededeling van Alice ontgaan. Bob schuift na ontvangst de letters natuurlijk gewoon weer terug:

$$D_k(y) = y - k.$$

In de encryptie- en decryptiefunctie is te zien dat de zender en de ontvanger *dezelfde* geheime informatie k gebruiken, alleen op een *andere manier* (optellen, respectievelijk aftrekken). Anders gezegd, de encryptiefunctie en de decryptiefunctie zijn *niet gelijk*, maar *kennis van de encryptiefunctie* is wel gelijk aan *kennis van de decryptiefunctie*. Cryptografische algoritmen die hieraan voldoen noemen we symmetrische algoritmen.

In het *affiene systeem* maken we het nog iets mooier door de optelling te vervangen door een algemene lineaire transformatie modulo m (eerder was m 32, maar het kan ook een ander getal zijn). De sleutel is een paar (a, b), waarbij a een multiplicatieve inverse heeft modulo m (dat wil zeggen, er bestaat een a' waarvoor $a \cdot a' = 1$ modulo m).

$$E_{(a,b)}(x) = a \cdot x + b, \qquad \text{en} \qquad D_{(a,b)}(y) = a'(y - b).$$

Een voorbeeld: de sleutel (11, 8) voldoet aan de eis, want 11 heeft inverse 3, immers $3.11 = 1$ (we rekenen immers in $\mathbb{Z}_{32}$, dus modulo 32). Met deze sleutel is

$$E(\mathbf{A}) = E(1) = 11 \cdot 1 + 8 = 19 = S,$$

en

$$D(S) = D(19) = 3 \cdot (19 - 8) = 1 = \mathbf{A}.$$

Het schuif- en affiene systeem zijn speciale gevallen van het toepassen van een willekeurige permutatie op de letters van het alfabet. In het *substitutiesysteem* is de sleutel een willekeurige permutatie $\pi \in S_m$, en

$$E_\pi(x) = \pi(x), \qquad \text{en} \qquad D_\pi(y) = \pi^{-1}(y).$$

1.2.2 Drie primitieve aanvallen

Helaas zal Oscar zich niet neerleggen bij de ondoorgrondelijkheid van de verzonden mededelingen en proberen er, ondanks de versleuteling, wijs uit te worden. Een poging om een cryptografische beveiliging te doorbreken wordt steeds een *aanval* genoemd. Hier is het doel van de aanval, versleutelde berichten te kunnen lezen. Hoewel men zou kunnen denken dat Alice en Bob een extra veiligheid zouden genieten door ook de werking van hun systeem voor Oscar geheim te houden, gaan we hier nooit van uit:

Aanname 1.1 (Kerckhoffs' principe) *Oscar kent het cryptosysteem dat door Alice en Bob wordt gebruikt; alleen de sleutel is hem onbekend.*

Motivaties voor deze aanname zijn:

1. Een sleutel is gemakkelijker te vernieuwen dan een methode. Bij het onverhoopt uitlekken van de sleutel kan daarom snel een nieuwe worden gekozen, maar bij het onverhoopt uitlekken van de methode is het niet gemakkelijk een nieuwe te kiezen. Je moet dan weer een echt nieuw idee krijgen en software en apparatuur vervangen.
2. Bij veel vormen van communicatie tussen veel partijen wordt gebruikgemaakt van gestandaardiseerde software in browsers en email-programma's. In deze setting is het natuurlijk volstrekt onmogelijk om de standaarden geheim te houden.
3. Zeker een veelgebruikt algoritme is op de lange duur toch niet geheim te houden, zelfs niet als het alleen in hardware wordt aangeboden; zie kader 1.2. De kennis over het algoritme is sowieso bij de ontwerpers en bouwers van het systeem bekend, terwijl je als gebruiker niet wilt dat de ontwerpers je berichten kunnen lezen.
4. De veiligheid van communicatie berust op deze manier op de geheimhouding van zo weinig mogelijk informatie. En hoe minder informatie je geheim hoeft te houden, des te gemakkelijker dit is.
5. Als de methode bekend is, kunnen alle geschoolde cryptografen hun mening vormen over de veiligheid ervan, en deze kwestie publiekelijk bespreken.

Omdat de sleutel k nu de enige informatie is waarvan we de geheimhouding aannemen, gaan we straks analyseren welke mate van bescherming de sleutel biedt.

We kijken eerst naar de situatie waarin Oscar slechts een stukje cijfertekst heeft opgevangen: *DSHNRS*, maar over de sleutel en verdere teksten niets weet. We spreken dan van een *cijfertekstaanval* (*Ciphertext-Only Attack* (COA)).

In het geval van het schuifsysteem zal Oscar de letters van de tekst steeds een positie terugschuiven door het alfabet en vindt hij in het gegeven voorbeeld achtereenvolgens:

DSHNRS
CRGMQR
BQFLPQ
APEKOP

In dit geval is de 'geheime' boodschap na drie stappen al gevonden. Algemeen vindt Oscar natuurlijk een lijst van 32 (aantal mogelijke verschuivingen) van teksten, waaruit hij die tekst kiest die er als een echte boodschap uitziet.

Oscar probeert in feite alle mogelijke verschuivingen uit. Een aanval waarbij alle mogelijke sleutels worden uitgeprobeerd noemen we een *brutekrachtaanval* (*Brute Force Attack* (BFA)).

Bij het affiene systeem zijn er meer sleutels mogelijk, maar in ons geval ook niet meer dan 512; voor a zijn namelijk alleen oneven getallen mogelijk (dit wordt uitgelegd in hoofdstuk 5). Met computers ligt dit natuurlijk gemakkelijk binnen het bereik van een BFA maar we zullen hier een andere aanpak illustreren. Een *klaretekstaanval* (*Known Plaintext Attack* (KPA)) is een aanval waarbij de tegenstander beschikt over enkele bij elkaar passende paren (x, y) waarvoor $E(x) = y$. Behalve de cijfertekst is dus ook de klare tekst bij de aanvaller bekend, en hij kan nu proberen hieruit de sleutel te vinden of de klare tekst die bij volgende stukken cijfertekst hoort.

Oscar kan tegen het affiene systeem een aanval lanceren op basis van twee zulke paren. Uit

$$\begin{aligned} y_1 &= a \cdot x_1 + b \\ y_2 &= a \cdot x_2 + b \end{aligned}$$

volgt namelijk $(y_1 - y_2) = a(x_1 - x_2)$ en dus (mits $x_1 - x_2$ een inverse heeft):

$$\begin{aligned} a &= (y_1 - y_2) \cdot (x_1 - x_2)^{-1} \\ b &= y_1 - a \cdot x_1 \end{aligned}$$

Deze aanval is ook uit te voeren zonder klare tekst, namelijk door er een element van statistische analyse aan toe te voegen. Niet alle symbolen van het alfabet worden even vaak gebruikt: de spatie komt het vaakst voor (15 tot 25 procent), daarna zijn de **E**, **A** en **T** erg populair. Oscar kan in een cijfertekst van enkele woorden zoeken welk symbool (zeg y_1) het vaakst voorkomt en welk (zeg y_2) daarna. Hij probeert dan de aanval met x_1 een spatie en x_2 een **E**, en vindt vrij snel een sleutel waarmee de hele tekst leesbaar wordt.

In het substitutiesysteem is het aantal mogelijke sleutels, 32! ofwel ongeveer 2.63×10^{35}, veel te groot om ze allemaal uit te proberen. Een brutekrachtaanval is hier dus niet mogelijk, toch is een cijfertekstaanval al snel succesvol, weer door statistische analyse. Het vaakst voorkomende symbool wordt op de spatie afgebeeld (et cetera).

1.2.3 Drie minder elementaire systemen

Omdat het breken van de behandelde codes zo eenvoudig is, wilde men natuurlijk systemen maken die nog iets ingewikkelder in elkaar zaten, maar het substitutiesysteem is het meest algemene dat met de afzonderlijke letters is uit te voeren. Daarom ging men, al eeuwen geleden, over op zogenaamde *blokversleuteling* (*block cipher*), waarin een reeks $x_1 x_2 \ldots x_b$ van b symbolen wordt gecombineerd en in één keer versleuteld.

Het Vigenère-systeem uit de zestiende eeuw wisselde b verschillende verschuivingen af: de sleutel was dus een rijtje $(b, k_1, \ldots, k_b)$ en

$$E(x_1 x_2 \ldots x_b) = (x_1 + k_1 \; x_2 + k_2 \; \ldots \; x_b + k_b),$$

Kader 1.2: Versleuteling bij mobiele telefonie

Geen infrastructuur is zo open voor afluisteren als de vrije ether; radiosignalen verspreiden zich onafhankelijk van de bedoelde bestemming en kunnen door iedereen in het bestreken gebied met een ontvanger worden beluisterd. In sommige landen (Duitsland, de Verenigde Staten) genieten bellers een *wettelijke* bescherming tegen het beluisteren van gesprekken, maar echte gangsters (en de politie!) trekken zich hiervan weinig aan. In Nederland is het beluisteren van alle ethercommunicatie toegestaan; het kwetsbaarst zijn gebruikers van analoge draadloze telefoons in de 31MHz- en 46MHz-band. De modernere DECT-telefoons werken digitaal en eenvoudige scanners (foto) zijn niet met een DA-omzetter uitgerust.
Door de scanner aan een pc te koppelen kan echter ook digitale communicatie worden afgetapt. De DECT-standaard ondersteunt versleuteling, maar of dit ook daadwerkelijk is gebruikt verschilt per fabrikant.
Het Europese gsm-systeem voor mobiele telefonie is niet alleen volledig digitaal, maar voorziet ook in encryptie tussen de mobiele telefoon en het basisstation. Het gebruikte algoritme, A5/1, is lang door de ontwerpers 'onder de pet' gehouden, maar de werking werd achterhaald via reverse engineering van gsm-telefoons; zie sectie 4.3. Bij het ontwerp van de volgende generatie mobiele communicatie, het UMTS netwerk, is besloten te werken volgens Kerckhoffs' principe: het ontwerp is geheel openbaar en de veiligheid berust alleen op geheimhouding van de sleutels.

het blok $(x_{b+1} \dots x_{2b})$ gaat eender. Als variant op dit systeem kun je ook b affiene of zelfs substitutieversleutelingen gebruiken; we krijgen dan een *polyalfabetisch* systeem.

Een *transpositiesysteem* verandert de letters niet, maar plaatst ze door elkaar. De sleutel is een permutatie over de indices van 1 tot b, en

$$E(x_1 x_2 \dots x_b) = (x_{\pi(1)} x_{\pi(2)} \dots x_{\pi(b)}).$$

Polyalfabetische substitutie en transpositie kunnen naar hartelust worden gecombineerd. Een generalisatie van het affiene systeem is het *Hill-systeem*, met als sleutel een (inverteerbare) matrix K over $\mathbb{Z}_m$, en $E_K(x_1 x_2 \dots x_b) = K \cdot \vec{x}$.

Alle genoemde systemen hebben een bruikbare periode gehad, maar die is reeds lang geleden. De Vigenère-systemen zijn als de shift- en substitutiemethoden te breken zodra de blokgrootte b bekend is, en die is er met een statistisch filter gemakkelijk uit te zeven. Het systeem van Hill is gemakkelijk te breken met een bekende plaintext van b^2 tekens: de coefficiënten van K zijn dan uit een lineair stelsel oplosbaar.

Een nog gecompliceerdere vorm van versleuteling is de stroomversleuteling, waarbij met een bepaald proces een zeer lange reeks sleutels wordt gegenereerd. Een mooi voorbeeld waren

de *rotormachines* die in de Tweede Wereldoorlog werden gebruikt. Elke rotor genereerde in elke stand een andere substitutie, en omdat de rotoren met verschillende relatieve snelheid ronddraaiden werd een groot aantal verschillende substituties gegenereerd voordat de machine weer in haar begintoestand was.

De tijd van het letter-voor-letter versleutelen van een tekst was rond 1960 voorbij. Een leesbaar en anecdotisch boek over die periode is dat van Bauer [Bau97].

1.3 Informatietheoretische veiligheid

Rond 1948 werd door Shannon een theorie over informatie ontwikkeld, die toepassing vond in communicatie, compressie en cryptografie. We zullen nu enkele delen van die theorie in vereenvoudigde vorm behandelen: het begrip entropie wordt niet statistisch benaderd.

1.3.1 Perfecte verhulling

We nemen aan dat Oscar beschikt over niets dan de cijfertekst Y (en dat hij weet welk systeem is gebruikt); hij zoekt een oplossing X die voldoet aan de relatie

$$\exists k : Y = E_k(X).$$

Een ideaal cryptosysteem onthult in dit geval niets over de klare tekst, de oplossingsruimte van deze vergelijking bestaat dan uit de verzameling van alle strings (symboolrijen). Als dit voor alle Y geldt, is X goed beschermd, ongeacht de hoeveelheid rekenwerk die Oscar voor de aanval kan gebruiken. We spreken dan van *perfecte veiligheid*.

Definitie 1.2 *Een cryptosysteem is* perfect verhullend *als*

$$\forall Y : \forall X : \exists k : E_k(X) = Y.$$

Perfecte verhulling wordt ook wel *informatietheoretische* verhulling genoemd omdat de cijfertekst geen informatie over de klare tekst bevat.

De *lengte* van de tekst zal doorgaans bij versleuteling onveranderd blijven, die wordt dus voor een afluisteraar bekend verondersteld en daarom moet er voor de kwantificatie over X worden gelezen: van dezelfde lengte als Y. We accepteren dat de lengte van X aan Oscar wordt onthuld.

Omdat encryptie en decryptie elkaars inverse zijn, moet dus gelden:

$$\forall Y : \forall X : \exists k : D_k(Y) = X.$$

Corollarium 1.3 *Perfecte verhulling impliceert dat er ten minste zo veel sleutels als klare teksten bestaan.*

Bewijs. Om een vaste Y op alle mogelijke X af te beelden zijn ten minste zo veel verschillende sleutels nodig. □

Als er bijvoorbeeld een boodschap van 100.000 byte moet worden versleuteld (hiervan zijn er 2^{800000}), moet er ook een sleutel van 100.000 byte worden gebruikt (want van kortere sleutels zijn er minder). De situatie is te vergelijken met het op slot zetten van fietsen. Om niet met teveel ijzer op de nek rond te lopen, sluiten we de fiets af met sloten waarvan we slechts de sleutels rond hoeven te dragen. De fiets is pas veilig als de sleutelbos even zwaar is als de fiets zelf.

Kader 1.3: Militaire telexkamer uit de Koude Oorlog

Bovenaan staat een rij standaard telexmachines met een vijf-bitscodering van tekens, elk met een directe lijn verbonden met een tegenstation (aangegeven op het bordje erboven).
Per telex is er (middelste rij) een versleutelmachine, de Ecolex IV, die de telexsignalen combineert met een one-time pad. Dit pad is op de twee linkermachines te zien als doorlopende papierstrook, die uit een verzegelde metalen cassette komt. Men ging ervan uit dat De Rus de inhoud niet kon achterhalen, zeker niet ongemerkt: daartoe diende de verzegeling, die natuurlijk door het bedienend personeel steeds nauwgezet moest worden gecontroleerd!
De overige machines werken met een 'benadering' van het one-time pad. De kasten onderaan werken als een soort random-numbergenerator die een eindeloze reeks bits genereren uit een klein strookje papier (aan beide partijen bekend). De uitvoer van de generator wordt als pad gebruikt. De bedoeling hiervan is om te besparen op de hoeveelheid sleutelmateriaal, maar het systeem is natuurlijk niet langer perfect verhullend! Een ander voorbeeld van een dergelijk algoritme is A5/1 voor gsm-telefonie (sectie 4.3).
Je mag een one-time pad nooit een tweede maal voor een volgend bericht gebruiken. Een afluisteraar vindt dan een belangrijke correlatie tussen de eerste en de tweede cijfertekst waaruit de klare teksten zijn af te leiden. Daarom, en om gesnuffel door Russen in onze papierbakken tegen te gaan, waren de Ecolexen van speciale mesjes voorzien die de papierstroken vernietigden.

1.3.2 Het one-time pad

Dit klinkt onbruikbaar, maar toch is er werkelijk een systeem toegepast dat perfect veilig is: het *one-time pad*.

De sleutel bij het one-time pad is een bitrij $b_1 \dots b_n$, even lang als de te coderen tekst. Om de bits $x_1 \dots x_n$ te coderen, berekent Alice y_i als $x_i \oplus b_i$. Bob vindt x_i terug door y_i opnieuw te XOR-en met de (hem bekende) b_i.

Met Oscar staat het echter beroerd: hij kent $y_1 \dots y_n$, maar deze ciphertext kan de codering zijn van *elke* boodschap x', namelijk als de sleutel $b' = y \oplus x'$ was gebruikt. Anders gezegd, als Oscar alle mogelijke sleutels zou gaan proberen, dan zou hij ook alle zinvolle teksten van n bits een keer als uitkomst vinden en hij heeft dan geen mogelijkheid om uit te maken welke de juiste is. Het one-time pad is dus perfect veilig omdat de ciphertext geen informatie over de plaintext bevat. Tot enkele decennia geleden werd het systeem door het Nederlandse leger gebruikt; zie kader 1.3.

Het one-time pad is in 1918 uitgevonden en stond sindsdien bekend als een 'onbreekbaar'

systeem, alleen wist niemand hoe je zoiets kon bewijzen totdat Shannon rond 1948 met zijn entropietheorie en de definitie van perfecte veiligheid kwam.

1.4 Kritieke lengte en computationele veiligheid

Het zal duidelijk zijn dat een cryptosysteem waarin de sleutels zo groot zijn als de boodschappen, niet erg bruikbaar is in veel moderne toepassingen. Wij willen dus veel data beschermen met een relatief kleine sleutel, en uit corollarium 1.3 blijkt dat het systeem dan niet perfect verhullend is: enige informatie over de klare tekst zal worden onthuld. Wij zullen nu gaan beredeneren, dat deze onthulling *totaal* is, dat wil zeggen, dat er bij een gegeven cijfertekst $Y = E_{k_0}(X_0)$ *slechts één klare tekst* is die op Y wordt afgebeeld:

$$\neg\exists X,\, k: \quad (X,\, k) \neq (X_0,\, k_0) \;\wedge\; Y = E_k(X).$$

De oplossing van Oscars vergelijking is derhalve uniek; belangrijke aanvullende randvoorwaarde is wel dat Oscar zich gaat beperken tot 'zinvolle' boodschappen.

In sectie 1.4.1 bekijken we alleen de uniciteit van de oplossing en zullen we ons niet bekommeren om de rekentijd die met het vinden ervan gemoeid is. Dat aspect wordt besproken in sectie 1.4.2.

1.4.1 Berekening van kritieke lengte

Neem aan dat Oscar een boodschap $Y = E_{k_0}(X_0)$ heeft onderschept, dat klare en cijfertekst lengte n bits hebben, en dat de sleutel lengte b heeft. Er zijn dus $N = 2^n$ mogelijke strings die als klare tekst of cijfertekst in aanmerking komen en $K = 2^b$ mogelijke sleutels; doorgaans is b flink kleiner dan n. Dat is prettig voor Oscar, want het impliceert dat er, na onderschepping van een cijfertekst Y_0, nog maar 2^b mogelijkheden voor de klare tekst overblijven, namelijk de 2^b waarden $D_k(Y_0)$, waar k elke mogelijke sleutel kan zijn.

Nu hebben wij bij de aanval op het schuifsysteem gezien dat Oscar gebruik kan maken van kennis over de taal waarin Alice en Bob communiceren: Oscar verwierp de tekst *BQFLPQ*, maar accepteerde **APEKOP**. Het onderscheid tussen onzinnige en zinnige boodschappen wordt geformaliseerd in de aanname dat Alice en Bob communiceren in een *taal*, een verzameling L van strings: $L \subseteq \{0,1\}^*$. Nu is het bij veel talen (natuurlijke, maar ook formele) zo dat het aantal strings van lengte n ongeveer groeit als $2^{\alpha \cdot n}$ voor zekere α; zie opgaven 1.6 tot 1.9.

Definitie 1.4 *Een taal L heeft* entropie α *als het aantal strings (zinnen) van L van lengte n ongeveer $2^{\alpha \cdot n}$ bedraagt. De* redundantie *van L is $\rho = 1 - \alpha$.*

De entropie van natuurlijke talen is erg klein: van het Engels is bekend dat deze ver onder de 25 procent ligt. Het bestuderen van entropie is ook bij compressie erg belangrijk. Van een taal met entropie α kun je namelijk zeggen dat strings in die taal voor een fractie α betekenis dragen, en hoe kleiner α is, des te betere compressie is mogelijk. Compressie verhoogt de entropie, en wel evenredig: heeft dus Engelse tekst (taal: 'ASCII-Engels') een entropie van bijvoorbeeld 20 procent, en zou men door compressie de omvang van de teksten met een factor 3 reduceren, dan heeft de resulterende taal ('GeZIPt Engels') een entropie van 60 procent. De taal van alle strings heeft entropie 1 en is dus niet te comprimeren.

Kerckhoffs' principe zegt hier dat Oscar niet alleen de functies E en D kent, maar dat hij ook weet in welke taal Alice en Bob communiceren. Oscar zoekt dan bij de onderschepte Y een oplossing $X \in L$ van de relatie

$$\exists k : Y = E_k(X).$$

Hoe groot is de oplossingsruimte van deze vergelijking? Uiteraard bevat hij de daadwerkelijke boodschap X_0 en sleutel k_0. Een sleutel $k \neq k_0$ waarvoor $D_k(Y) \in L$ noemen we een *bastaardsleutel* (*spurious key*). De verdere berekening gaat uit van een aanname, namelijk dat het versleutelen en vervolgens ontsleutelen met *verschillende* sleutels werkt als een soort randomgeneratie in de strings van lengte n, dat wil zeggen, we rekenen verder alsof voor $k \neq k_0$, $D_k(Y)$ een random string van n bits is.

Lemma 1.5 *Het verwachte aantal bastaardsleutels is kleiner dan* $K/2^{(\rho \cdot n)}$.

Bewijs. De kans dat een willekeurig getrokken bitrij in L zit is $\frac{2^{\alpha \cdot n}}{2^n}$; immers er zijn precies 2^n bitrijen van lengte n en ongeveer $2^{\alpha \cdot n}$ zinnen in L. Elke $k \neq k_0$ heeft dus een kans van $\frac{2^{\alpha \cdot n}}{2^n}$ dat $D_k(Y) \in L$. Omdat het aantal sleutels ongelijk k_0 precies $K-1$ is (net iets minder dan K dus!), verwachten we dat dit minder dan $K \cdot \frac{2^{\alpha \cdot n}}{2^n} = K/2^{(\rho \cdot n)}$ keer voorkomt (stelling A.4). □

Bij langere berichten zijn er dus minder bastaardsleutels.

Definitie 1.6 *De* kritieke lengte *(unicity distance) van een cryptosysteem is de berichtlengte waarbij het verwachte aantal bastaardsleutels daalt onder 1.*

Stelling 1.7 *Bij een cryptosysteem met sleutels van b-bits, waarmee een taal met entropie* α *wordt versleuteld, bedraagt de kritieke lengte*

$$n_0 = \frac{b}{1-\alpha} \qquad \left(= \frac{b}{\rho} \right).$$

Bewijs. Voor $n > \frac{b}{\rho}$ is het verwachte aantal bastaardsleutels volgens lemma 1.5 kleiner dan $2^b/2^{\rho.(b/\rho)} = 1$. □

Wat is de betekenis van de kritieke lengte? De waarde van n_0 hangt alleen af van de combinatie van cryptosysteem (alleen de sleutellengte) en gebruikte taal, dus verschilt niet per bericht. Een cijfertekst die langer is dan n_0 heeft een unieke klare tekst als oplossing: er is (waarschijnlijk) maar één tekst die met enige sleutel erop kan worden afgebeeld. Het kan dan nog wel moeilijk zijn om die klare tekst te vinden, maar er is geen sprake van te moeten kiezen tussen meerdere klare teksten, zoals dat bij perfecte verhulling wel het geval is.

In het diepergaande werk van Shannon wordt geen strikte scheiding tussen taal en nontaal verondersteld, maar wordt verondersteld dat elke string een bepaalde kans heeft om als klare tekst voor te komen. De entropie gaat dan van 1 afwijken doordat sommige strings waarschijnlijker zijn dan andere. We verwijzen naar de bespreking door Stinson [Sti95, Hf. 2].

1.4.2 Veiligheid is computationeel

De kritieke lengte van cryptosystemen is doorgaans verrassend laag; in ieder geval zo laag, dat berichten in de praktijk altijd langer zijn dan de kritieke lengte. Het substitutiesysteem op een alfabet van 32 tekens heeft bijvoorbeeld $32! \approx 2^{118}$ sleutels, en een Engelse zin van

118 bits/75% $\approx$ 160 bits overschrijdt de kritieke lengte! Het veelgebruikte DES (zie hoofdstuk 2) gebruikt sleutels van slechts 56-bits. Ook met een zeer gunstige waarde van de parameters blijft de kritieke lengte laag; neem als voorbeeld een taal met entropie 95 procent en versleuteling met sleutellengte 512-bits: n_0 is dan 1280 bytes, ruim minder dan de lengte van een gemiddelde e-mail.

Zo kunnen we concluderen dat bij alle in de praktijk gebruikte cryptosystemen de inhoud van de klare tekst vastligt door kennis van de cijfertekst. De systemen bieden dus *geen* informatietheoretische bescherming: bij vrijwel alle gecodeerde boodschappen Y is de oplossing X van $\exists k : E_k(X) = Y$ uniek. De veiligheid van het cryptosysteem berust daarom altijd hierop, dat het vinden van deze unieke oplossing onhaalbaar veel werk vereist; veiligheid is niet *informatietheoretisch*, maar *computationeel* van aard.

Definitie 1.8 *Een cryptosysteem is* computationeel verhullend *als het vinden van de klare tekst uit de cijfertekst (zonder kennis van de sleutel) een onuitvoerbaar grote hoeveelheid werk vereist.*

De klare tekst X kan altijd worden gezocht met een brutekrachtaanval, waarbij alle sleutels worden geprobeerd met de test $D_k(Y) \in L$. Voor computationele veiligheid is vereist dat dit in ieder geval onhaalbaar is, en daarnaast dat er geen andere, efficiënte oplosmethode bestaat.

1. **Sleutellengte.** Het aantal bits in de sleutel moet zo hoog zijn dat een brutekrachtaanval niet uitvoerbaar is. Bij de huidige (jaar 2002) stand van de techniek betekent dit dat b ten minste 80 à 120 moet bedragen.
2. **Algoritmisch sterk.** Het algoritme moet zo in elkaar zitten dat er niet een of ander trucje bestaat waarmee de unieke oplossing van $X = D_k(Y)$ met minder dan 2^b rekenstappen te vinden is.

Het DES-algoritme heeft een te kleine sleutellengte; een brutekracht DES-kraker wordt in sectie 2.2.2 beschreven. Het substitutiesysteem is *niet* algoritmisch sterk; door het een voor een raden van letters kan de gebruikte permutatie veel sneller worden gevonden dan door het uitproberen van alle bestaande permutaties.

1.4.3 Compressie voor versleuteling

Het is in cryptografische systemen gebruikelijk de data te comprimeren alvorens de versleuteling toe te passen; dit wordt bijvoorbeeld gedaan in Pretty Good Privacy (PGP) en in gsm-telefoons.

Soms wordt beargumenteerd dat compressie de veiligheid van de encryptie verder vergroot, en enkele argumenten hiervoor worden hieronder besproken. In een goed ontwerp mag echter de bijdrage van compressie aan de veiligheid van encryptie *niet doorslaggevend zijn* voor de veiligheid van encryptie!

In geen enkel geval mag de kwaliteit van de encryptie afhankelijk zijn van bewerkingen die de data ervoor of erna ondergaat; een soortgelijke redenering komt men tegen bij A5/1 voor gsm-telefonie (sectie 4.3.2). *Separation of Concern* in een systeemontwerp dicteert dat cryptografie zorgt voor de veiligheid, en compressie voor het verkleinen van de hoeveelheid data, en niets anders. De compressie gebruikt altijd eigenschappen van de data die verdwijnen door encryptie; het is daarom volstrekt zinloos om compressie op versleutelde data toe te passen.

Het bericht en de kritieke lengte. Door compressie neemt de entropie van het bericht toe en de lengte ervan af! Voor de vergelijking van berichtlengte met de kritieke lengte snijdt het compressiemes dus aan twee kanten.

Hoe hard het snijdt kun je ook uitrekenen (opgave 1.12), maar hiervan hebben we in feite weinig te verwachten; de kritieke lengte blijft klein in verhouding tot de meeste communicatie.

Rekenwerk voor brutekrachtaanval. Bij de aanval moet Oscar met elke sleutel een stuk tekst ontsleutelen en testen. Wanneer compressie wordt toegepast, neemt het werk dat Oscar voor elke sleutel moet doen flink toe, omdat hij elke ontsleutelde tekst moet decomprimeren voordat hij de test op inclusie in L uitvoert.

Bij een brutekrachtaanval zal Oscar natuurlijk niet met elke sleutel de *gehele boodschap* decoderen, maar slechts een beginstuk met een lengte van ongeveer n_0 (of nog minder). Vaak is voor decompressie een string van minstens enkele honderden bytes nodig, wat Oscar dwingt om met elke sleutel een flink stuk van de boodschap te decoderen, veel meer dan de kritieke lengte.

Deze redenering is niet bij alle vormen van compressie geldig. Compressie voor encryptie kan ook juist risico's opleveren. Standaard compressie-methoden voorzien het gecomprimeerde bestand van een standaard header (met de softwareversie, datum etc.). Daardoor wordt het begin van het bericht juist erg voorspelbaar en levert compressie juist de aanvaller een stukje known plaintext.

Bruikbare entropie: machinaal herkenbare taal. Als een brutekrachtaanval automatisch wordt uitgevoerd, zal ook de test of de ontsleutelde string in de taal zit, automatisch moeten worden uitgevoerd. Dat is jammer voor Oscar, want er is nog geen computerprogramma uitgevonden dat de taal *'ASCII-Engels'* of *'JPEG van gevechtsvliegtuig'* kan herkennen.

Daarom is het Oscar altijd onmogelijk de redundantie geheel te benutten: in plaats van een exacte taaldefinitie gebruikt hij een grotere verzameling die machineherkenbaar is. Voor 'ASCII-Engels' is dit bijvoorbeeld de verzameling ASCII-bestanden (elke achtste bit van een byte is 0) en voor 'JPEG van gevechtsvliegtuigen' is dit bijvoorbeeld de verzameling van JPEG-bestanden waarin meer dan 40 procent van de pixels blauw is.

Maar daarvan hebben Alice en Bob weer voordeel: elke algoritmisch herkenbare eigenschap van strings in de taal kan namelijk worden gebruikt om de bestanden te comprimeren tot kortere bestanden, die juist deze eigenschap niet meer hebben. Inderdaad kun je ASCII-bestanden (als boven gekarakteriseerd) heel gemakkelijk met een achtste comprimeren door elke achtste bit te verwijderen. Zelfs als de entropie door een dergelijke maatregel niet kwantitatief significant wordt verlaagd, scoren Alice en Bob toch belangrijk in hun strijd tegen Oscar omdat zij juist het stukje redundantie elimineren dat Oscar kon exploiteren.

Risico's van compressie: BREACH. Gluck e.a. [GHP13] demonstreerden een risico dat optreedt wanneer een aanvaller een hem onbekende tekst kan combineren met een gekozen tekst, en de lengte van het gecomprimeerde en versleutelde resultaat kan zien. De effectiviteit van de compressie (met `zip`) hangt af van hoeveel de gekozen tekst lijkt op de verborgen tekst. Door de gekozen tekst te variëren, kan de aanvallen letter voor letter het verborgen deel achterhalen.

Samenvatting en conclusies

Bij symmetrische cryptografie gebruiken zender en ontvanger dezelfde sleutel voor het versleutelen, respectievelijk ontcijferen, van de boodschap. Het versleutelproces bestaat namelijk uit omkeerbare stappen, en kennis van het versleutelproces (de sleutel) is voldoende om de oorspronkelijke boodschap uit de versleutelde tekst af te leiden.

Een aanval op het systeem is een poging van een buitenstaander de berichten of de sleutel te achterhalen; steeds wordt hierbij aangenomen dat de aanvaller de details van de werking van het systeem kent, maar alleen de sleutel niet. Bij een cijfertekstaanval beschikt de aanvaller alleen over de versleutelde berichten. Bij een klaretekstaanval beschikt hij over enige corresponderende versleutelde en niet-versleutelde berichten. Bij een gekozen-cijfertekstaanval kan hij van enkele door hem opgestelde berichten de ontsleutelde versie vernemen. Een aanval die alle sleutels systematisch probeert is een brutekrachtaanval.

Een cryptografisch algoritme is perfect veilig wanneer, gegeven de versleutelde tekst, alle teksten nog mogelijk zijn als klaretekstbericht. Deze eigenschap is aantrekkelijk, maar vereist dat de gebruikte sleutel zo groot is als de tekst. Is de sleutel kleiner, dan kan de aanvaller altijd informatie over de boodschap verkrijgen door te zoeken naar een sleutel die de verkregen cijfertekst op een leesbare tekst afbeeldt. Bij verrassend korte teksten (vanaf de kritieke lengte, die exact te berekenen is) is de gevonden sleutel uniek. Compressie van de boodschap (voor versleuteling) maakt de kritieke lengte groter, en het kraken van de cryptografie moeilijker.

Opgaven bij hoofdstuk 1

Opgaven 1.1 en 1.2 vereisen bekendheid met modulair rekenen. De lezer die hiermee niet vertrouwd is kan er mee wachten tot na hoofdstuk 5.

Opgave 1.1 *Hoeveel 2×2 matrices over $\mathbb{Z}_{32}$ bestaan er en hoeveel hiervan zijn inverteerbaar? Wat is het belang van de inverteerbare matrices?*

Opgave 1.2 *De KPA op het affiene systeem veronderstelde twee bekende letters x_1 en x_2 met* inverteerbaar *verschil $x_1 - x_2$. Hoe groot is de kans om in een bekende tekst van k willekeurige letters* geen enkel paar *aan te treffen met inverteerbaar verschil $x_i - x_j$?*

Opgave 1.3 *Bedenk een methode om van een stuk cijfertekst in het Vigenère-systeem de gebruikte blokgrootte te bepalen. (***Hint:** *Gebruik de onevenwichtige frequentieverdeling van tekens in natuurlijke taal.)*

Opgave 1.4 Perfecte en Computationele veiligheid. *(Uit het tentamen van januari 1997.)*
Geef de definitie van perfecte *en* computationele *veiligheid van een cryptosysteem.*

Zijn de volgende technieken perfect of computationeel veilig:
(1) ElGamal Vercijfering; (2) One-time pad; (3) Triple-DES; (4) Input-blindering zoals gebruikt bij blinded signatures.

Opgave 1.5 *Bewijs dat het affiene systeem perfect veilig is als je er slechts één letter mee versleutelt.*

In opgaven 1.6 tot 1.9 is het nodig te berekenen hoeveel strings een gegeven taal heeft van zekere lengte.

Opgave 1.6 *In een ASCII-bestand worden er van de 256 mogelijke waarden van een byte slechts 128 gebruikt. Bereken de entropie en de redundantie van ASCII-bestanden.*

Opgave 1.7 Shannon's informatietheorie. *(Uit het tentamen van oktober 1996.)*
Wat is de redundantie en entropie *van een taal, en hoe hangt dit begrip samen met de sleutellengte en de kritieke lengte (*unity distance*) van een cryptosysteem?*

Opgave 1.8 *Bereken de entropie van de taal van binaire strings waarin geen twee 1-en achter elkaar voorkomen.*

Opgave 1.9 *Bereken de entropie van de taal der welgevormde haakjesparen.*

Opgave 1.10 *Alice en Bob gebruiken een cryptosysteem met 512-bitssleutels en wisselen boodschappen uit met een entropie van 20 procent. Bereken de kritieke lengte. Hoe groot wordt de kritieke lengte als compressie met ratio 1:3 wordt toegepast?*

Opgave 1.11 Unicity Distance. *Wat is de betekenis van de zogenaamde* Unicity distance*? Alice en Bob communiceren in een taal met een entropie van 25%, gebruiken een symmetrisch cryptosysteem met een blokgrootte van 80 bits en een sleutellengte van 120 bits. Welk van deze gegevens is overbodig voor het berekenen van de Unicity distance?*

Hoeveel bedraagt in dit geval de Unicity Distance?

Welke maatregelen kunnen Alice en Bob nemen om de U.D. te vergroten? Hoe belangrijk is dat?

Opgave 1.12 *Als compressie de lengte van bestanden halveert, wat gebeurt er dan met de entropie en de redundantie? Druk α' en ρ' uit in α en ρ. Druk de kritieke lengte (van boodschappen vòòr compressie) uit in b, ρ en de compressieratio R.*

Opgave 1.13 *Hoe lang moet Oscar rekenen aan een BFA tegen het DES-algoritme (dit maakt gebruik van 56-bitssleutels)?*
Neem eerst aan dat Oscar over een pc beschikt die een miljoen sleutels per seconde kan uitproberen. Heeft Oscar veel aan een supercomputer die een miljard sleutels per seconde aan kan?

Hoofdstuk 2

Data Encryption Standard

In dit hoofdstuk duiken we diep in de theorie achter symmetrische cryptografie met een studie van de Data Encryption Standard (DES). De rol van DES op het cryptografische toneel is uitgespeeld, zoals we in sectie 2.2 zullen zien, maar de ideeën die aan het algoritme ten grondslag liggen zijn ijzersterk en komen terug in alle succesvolle opvolgers. Bovendien is het DES algoritme het uitvoerigst bestudeerde cryptografische algoritme dat er bestaat. Heel veel van wat bekend is over het ontwerp, het gebruik en het breken van symmetrische cryptografie is ontdekt op basis van DES. Een studie van symmetrische cryptografie kan daarom, ook aan het begin van de 21^{ste} eeuw, niet compleet zijn zonder een verdieping in DES.

2.1 De werking van DES

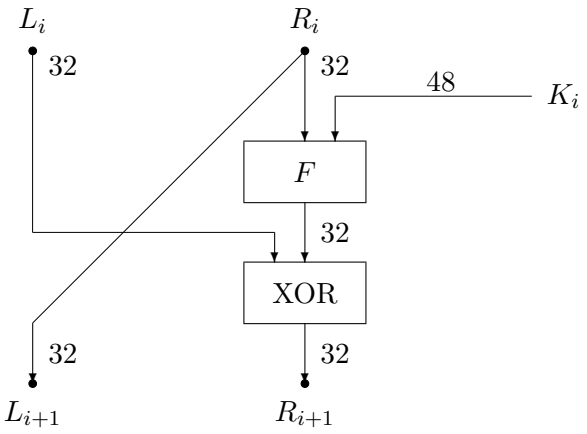
Het Data-Encryption-Standardalgoritme is ontworpen door IBM en in gebruik genomen in 1974 (een uitgebreide beschouwing van de historische ontwikkeling van DES volgt in sectie 2.4). Bij het ontwerp heeft men sterk gelet op een efficiënte implementeerbaarheid in hardware: daarom maakt het algoritme hoofdzakelijk gebruik van bitoperaties zoals het permuteren van bits en het puntsgewijs XOR-en met een bitrij.

$\oplus$	0	1
0	0	1
1	1	0

De XOR-functie beeldt twee bits af op 0 als ze gelijk zijn en op 1 als ze ongelijk zijn, als in deze tabel. De XOR-operatie is erg geschikt voor encryptie van een waarde x, omdat XOR-en met een waarde a (die hier als 'sleutel' werkt) als inverse dezelfde operatie heeft: er geldt $(x \oplus a) \oplus a = x$. De functie kan van twee bits worden uitgebreid naar rijen bits van gelijke lengte, waarbij dan steeds de bits op overeenkomstige posities met elkaar worden gecombineerd. Ook dan geldt weer: als Y verkregen wordt als $Y = X \oplus A$, dan wordt X weer uit Y verkregen door $X = Y \oplus A$.

Het algoritme werkt op blokken data (plaintext en ciphertext) van 64 bits en gebruikt een sleutel van 56 bits. We kunnen het algoritme dus beschouwen als een substitutiesysteem op het alfabet van 2^{64} bitrijen van lengte 64, maar terwijl er $2^{64}! \approx 10^{3.474 \times 10^{20}}$ van zulke substituties bestaan, worden er slechts $2^{56} = 7.21 \times 10^{16}$ van gebruikt.

Voor de beschrijving van de werking van het DES-algoritme is geput uit Schneier [Sch96] en Stinson [Sti95].



Figuur 2.1: Datastroom in DES-iteratie (Feistel-netwerk)

2.1.1 De DES-iteratie

De data wordt in DES behandeld in zestien achtereenvolgende ronden. Elke ronde versleutelt *de helft* van de data door deze te XOR-en met een bitrij van gelijke lengte. Deze bitrij wordt afgeleid van een deel (48) van de sleutelbits en de andere helft van de data; zie figuur 2.1. Wanneer een deel van de data al met de key wordt verweven als voorbereiding op de eigenlijke encryptie (het XOR-en met de andere helft), spreekt men van een *autocipher*. De rechterhelft van de data (R_i) wordt met de iteratiesleutel K_i gecombineerd in de functie F, hetgeen resulteert in een bitrij. De linkerhelft van de data wordt hiermee puntsgewijs geXORd en doorgegeven als rechterhelft R_{i+1}, terwijl de nieuwe linkerhelft L_{i+1} gelijk is aan de oude rechterhelft. Ten overvloede is in de figuur ook de breedte van elke datastroom aangegeven.

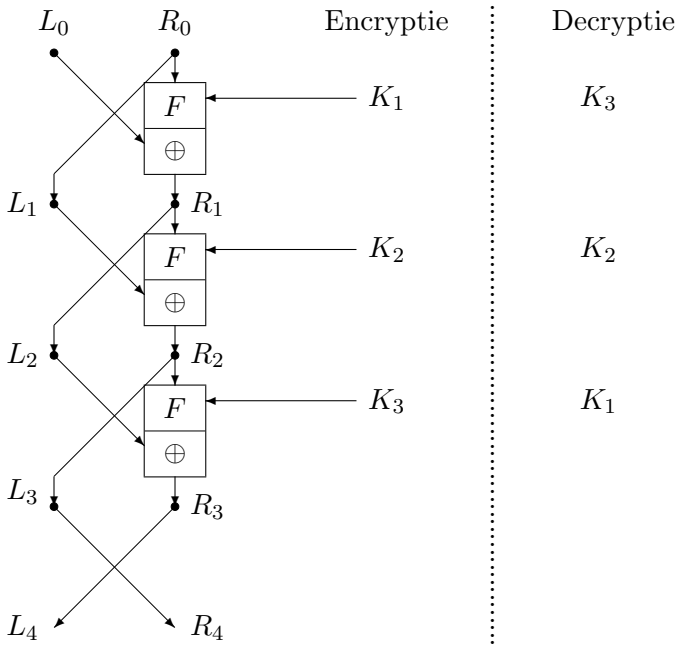
Je kunt deze iteratie, die bekend staat als een *Feistel-netwerk*, ook in formulevorm weergeven:

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus F(R_i,\ K_i) \end{aligned}$$

De functie F die in DES wordt gebruikt, wordt in sectie 2.1.3 verder besproken. Nu moet voor het decrypten de bewerking natuurlijk ongedaan gemaakt kunnen worden, maar dat gaat erg makkelijk bij een Feistel-netwerk. De datahelft die als invoer voor F is gebruikt, werd immers onveranderd doorgegeven en is daarom bij decryptie weer beschikbaar om de te XOR-en string opnieuw te berekenen. Aan de formules kunnen we zien dat de iteratie *zijn eigen inverse* is, mits de twee datahelften in omgekeerde volgorde worden aangeboden. We berekenen L' en R' door in de formules L_{i+1} en R_{i+1} in te voeren op de plaats van R_i en L_i en vinden:

$$\begin{aligned} L' &= L_{i+1} && && = R_i \\ R' &= R_{i+1} \oplus F(L_{i+1},\ K_i) && = L_i \oplus F(R_i,\ K_i) \oplus F(R_i,\ K_i) && = L_i \end{aligned}$$

Merk op dat niet alleen de twee datahelften weer precies worden teruggevonden, maar ook is hun volgorde alvast verwisseld voor het inverteren van de volgende iteratie. Figuur 2.2 laat zien hoe drie iteraties elkaar opvolgen. De iteraties worden gevolgd door het omwisselen van de twee datahelften. Het complete proces wordt geïnverteerd door de uitvoer (L_4 en R_4)



Figuur 2.2: Feistel-netwerk met drie iteraties en inverse

weer bovenin in te voeren, maar met de iteratiesleutels in omgekeerde volgorde, met andere woorden, het proces wordt opnieuw doorlopen maar nu met de laatst gebruikte iteratiesleutel als eerste enzovoort.

Een Feistel-netwerk kan ook de uitkomst van de functie op een andere manier met L_i combineren, bijvoorbeeld door een modulaire optelling. In dat geval wordt natuurlijk bij decryptie een aftrekking gebruikt. Feistel-netwerken zijn ook in latere algoritmen veel gebruikt, bijvoorbeeld in Kasumi (sectie 10.2.3).

2.1.2 De sleutelgeneratie

De sleutel wordt doorgaans in een DES-module (chip of software) ingevoerd als een reeks van 64 bits, maar elke achtste bit is een pariteitsbit die de juistheid van de zeven bits ervoor controleert en die doet in het algoritme verder niet mee. De netto sleutellengte is dus 56 bits.

Zoals al is opgemerkt bestaat in elke iteratie de sleutel uit een selectie van 48 van de 56 sleutelbits. Je kunt daarom de hele sleutelfabriek beschrijven als een tabel van zestien regels, elk met de 48 indices van de bits die in een iteratie worden gebruikt. Encryptie gebruikt de tabel van boven naar beneden, decryptie van beneden naar boven.

Om implementatie in hardware eenvoudiger te houden is er een zekere regelmaat in de sleutelgeneratie gehouden. De bits van de sleutel (genummerd 1 tot 64 inclusief pariteitsbits) worden verdeeld over twee 28-bitsbrede schuifregisters. Men neemt links de eerste bit van elke byte in aflopende volgorde, de tweede, enzovoort, en rechts de zevende bit, de zesde bit enzovoort, en aan het eind de vier vierde bits van de eerste vier bytes. Anders gezegd worden de twee schuifregisters geladen met de volgende bits (sleutelinvoerselectie):

Links	57	49	41	33	25	17	9	1	58	50	42	34	26	18	10	2
	59	51	43	35	27	19	11	3	60	52	44	36				
Rechts	63	55	47	39	31	23	15	7	62	54	46	38	30	22	14	6
	61	53	45	37	29	21	13	5	28	20	12	4				

Vervolgens wordt in elke ronde een circulaire verschuiving naar links toegepast over een of twee bits volgens deze tabel (sleutelrotatie):

Ronde	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Verschuiving	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Na de verschuiving worden deze 48 bits geselecteerd (sleutelcompressie, L van het linkerregister en R van het rechterregister):

Bits 1–12	L14	L17	L11	L24	L1	L5	L3	L28	L15	L6	L21	L10
Bits 13–24	L23	L19	L12	L4	L26	L8	L16	L7	L27	L20	L13	L2
Bits 25–36	R13	R24	R3	R9	R19	R27	R2	R12	R23	R17	R5	R20
Bits 37–48	R16	R21	R11	R28	R6	R25	R18	R14	R22	R8	R1	R4

Hoe worden de iteratiesleutels in omgekeerde volgorde gegenereerd voor decryptie? Merk op dat in de zestien ronden de totale verschuiving van de sleutelbits precies 28 posities bedraagt, met andere woorden, de zestiende iteratiesleutel bestaat uit de sleutelcompressie losgelaten op de sleutelinvoerselectie. De voorgaande iteratiesleutels worden verkregen door beide schuifregisters over een of twee posities *naar rechts* te schuiven, waarbij het sleutelrotatiepatroon natuurlijk van achteren naar voren wordt gelezen. Merk ten slotte op dat, behoudens de 1-rotatie in ronde 1, dit patroon palindromisch is, zodat in elke ronde over evenveel posities wordt verschoven als bij encryptie.

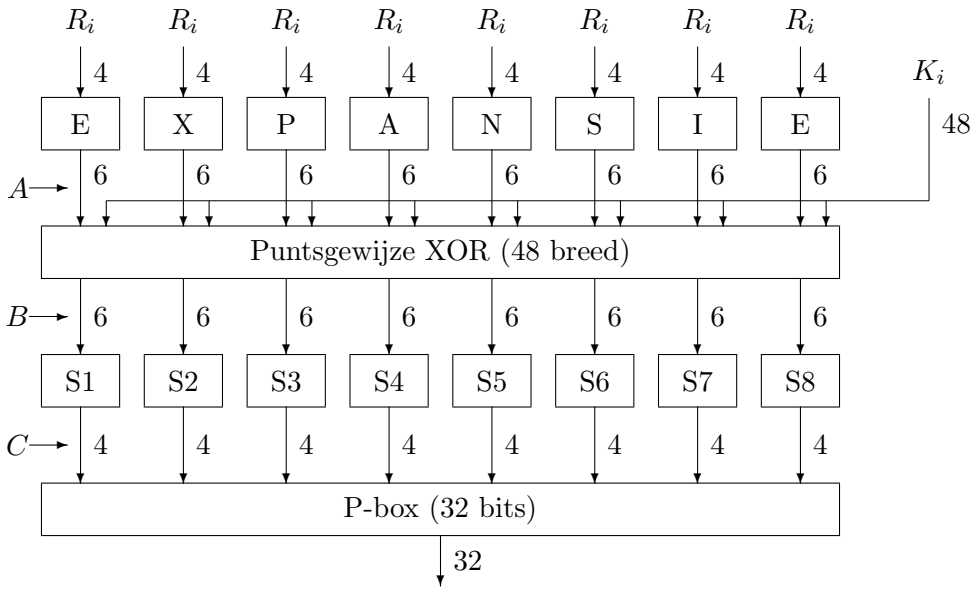
Het enige verschil tussen encryptie en decryptie zit dus in de sleutelrotatie en wel:

1. Bij decryptie wordt in ronde 1 geen rotatie toegepast.
2. Bij encryptie zijn de rotaties naar links, bij decryptie naar rechts.

2.1.3 De F-functie en S-boxen

Het cryptografische hart van DES ligt in de berekening van de functie F, of, nog specifieker, in de daarin gebruikte S-boxen. Figuur 2.3 geeft een overzicht van deze module. De F-berekening begint met de datahelft R_i en de iteratiesleutel K_{i+1} in acht blokjes te verdelen en puntsgewijs te XOR-en. Omdat de acht sleuteldelen elk zes bits hebben en de acht datadelen maar vier bits, krijgt elk datablokje er een bit van het linkerbuurblokje en een van het rechterbuurblokje bij (expansiefunctie):

Blokje	Heeft bits				Wordt bits nrs. ...					
1	1	2	3	4	32	1	2	3	4	5
2	5	6	7	8	4	5	6	7	8	9
3	9	10	11	12	8	9	10	11	12	13
4	13	14	15	16	12	13	14	15	16	17
5	17	18	19	20	16	17	18	19	20	21
6	21	22	23	24	20	21	22	23	24	25
7	25	26	27	28	24	25	26	27	28	29
8	29	30	31	32	28	29	30	31	32	1



Figuur 2.3: Gebruik van S-boxen en P-box in DES

Elk van de acht blokken van zes bits wordt nu gebruikt voor de aansturing van een zogenaamde *S-box* (S voor substitutie), die niets anders doet dan een bitrijtje van lengte 4 uit een tabel van 64 (= 2^6) stuks plukken. De acht S-boxen zijn verschillend (maar wel in iedere iteratie gelijk) en alle stringetjes zijn te vinden in tabel 2.4. Per box zijn de zestien waarden van de invoerbits 2 tot en met 5 horizontaal uitgezet en de eerste en zesde bit verticaal. Op deze manier weergegeven is elke regel van de tabel een permutatie van de zestien strings van vier bits. Je kunt een S-box dus zien als een *substitutie* losgelaten op de verzameling bitrijtjes van lengte 4, waarbij de gebruikte substitutiepermutatie wordt gekozen door het eerste en zesde invoerbit. Dit zijn precies de twee bits die bij de expansiefunctie van R_i zijn 'overgenomen' van de buurblokjes.

De 32 bits die uit de acht S-boxen komen worden gepermuteerd door ze op te leveren in deze volgorde (P-box):

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Het doel van deze permutatie is ervoor te zorgen dat elke S-box in de volgende ronde door meerdere S-boxen in deze ronde wordt beïnvloed.

In formulevorm kunnen we de bepaling van F zo uitdrukken:

R_i		invoer	32 bits
A	=	$Exp(R_i)$	48 bits
B	=	$A \oplus K_{i+1}$	8×6 bits
C	=	$Sbx(B)$	8×4 bits
F	=	$Pbx(C)$	32 bits

S1, reageert op $c_1c_2c_3c_4c_5c_6$

	Waarde van $c_2c_3c_4c_5$															
c_1c_6	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

S2, reageert op $c_7c_8c_9c_{10}c_{11}c_{12}$

	Waarde van $c_8c_9c_{10}c_{11}$															
c_7c_{12}	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1111	0001	1000	1110	0110	1011	0011	0100	1001	0111	0010	1101	1100	0000	0101	1010
01	0011	1101	0100	0111	1111	0010	1000	1110	1100	0000	0001	1010	0110	1001	1011	0101
10	0000	1110	0111	1011	1010	0100	1101	0001	0101	1000	1100	0110	1001	0011	0010	1111
11	1101	1000	1010	0001	0011	1111	0100	0010	1011	0110	0111	1100	0000	0101	1110	1001

S3, reageert op $c_{13}c_{14}c_{15}c_{16}c_{17}c_{18}$

	Waarde van $c_{14}c_{15}c_{16}c_{17}$															
$c_{13}c_{18}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1010	0000	1001	1110	0110	0011	1111	0101	0001	1101	1100	0111	1011	0100	0010	1000
01	1101	0111	0000	1001	0011	0100	0110	1010	0010	1000	0101	1110	1100	1011	1111	0001
10	1101	0110	0100	1001	1000	1111	0011	0000	1011	0001	0010	1100	0101	1010	1110	0111
11	0001	1010	1101	0000	0110	1001	1000	0111	0100	1111	1110	0011	1011	0101	0010	1100

S4, reageert op $c_{19}c_{20}c_{21}c_{22}c_{23}c_{24}$

	Waarde van $c_{20}c_{21}c_{22}c_{23}$															
$c_{19}c_{24}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0111	1011	1110	0011	0000	0110	1001	1010	0001	0010	1000	0101	1011	1100	0100	1111
01	1101	1000	1011	0101	0110	1111	0000	0011	0100	0111	0010	1100	0001	1010	1110	1001
10	1010	0110	1001	0000	1100	1011	0111	1101	1111	0001	0011	1110	0101	0010	1000	0100
11	0011	1111	0000	0110	1010	0001	1101	1000	1001	0100	0101	1011	1100	0111	0010	1110

S5, reageert op $c_{25}c_{26}c_{27}c_{28}c_{29}c_{30}$

	Waarde van $c_{26}c_{27}c_{28}c_{29}$															
$c_{25}c_{30}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

S6, reageert op $c_{31}c_{32}c_{33}c_{34}c_{35}c_{36}$

	Waarde van $c_{32}c_{33}c_{34}c_{35}$															
$c_{31}c_{36}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1100	0001	1010	1111	1001	0010	0110	1000	0000	1101	0011	1000	1110	0111	0101	1011
01	1010	1111	0100	0010	0111	1100	1001	0101	0110	0001	1101	1110	0000	1011	0011	1000
10	1001	1110	1111	0101	0010	1000	1100	0011	0111	0000	0100	1010	0001	1101	1011	0110
11	0100	0011	0010	1100	1001	0101	1111	1010	1011	1110	0001	0111	0110	0000	1000	1101

S7, reageert op $c_{37}c_{38}c_{39}c_{40}c_{41}c_{42}$

	Waarde van $c_{38}c_{39}c_{40}c_{41}$															
$c_{37}c_{42}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0100	1011	0010	1110	1111	0000	1000	1101	0011	1100	1001	0111	0101	1010	0110	0001
01	1101	0000	1011	0111	0100	1001	0001	1010	1110	0011	0101	1100	0010	1111	1000	0110
10	0001	0100	1011	1101	1100	0011	0111	1110	1010	1111	0110	1000	0000	0101	1001	0010
11	0110	1011	1101	1000	0001	0100	1010	0111	1001	0101	0000	1111	1110	0010	0011	1100

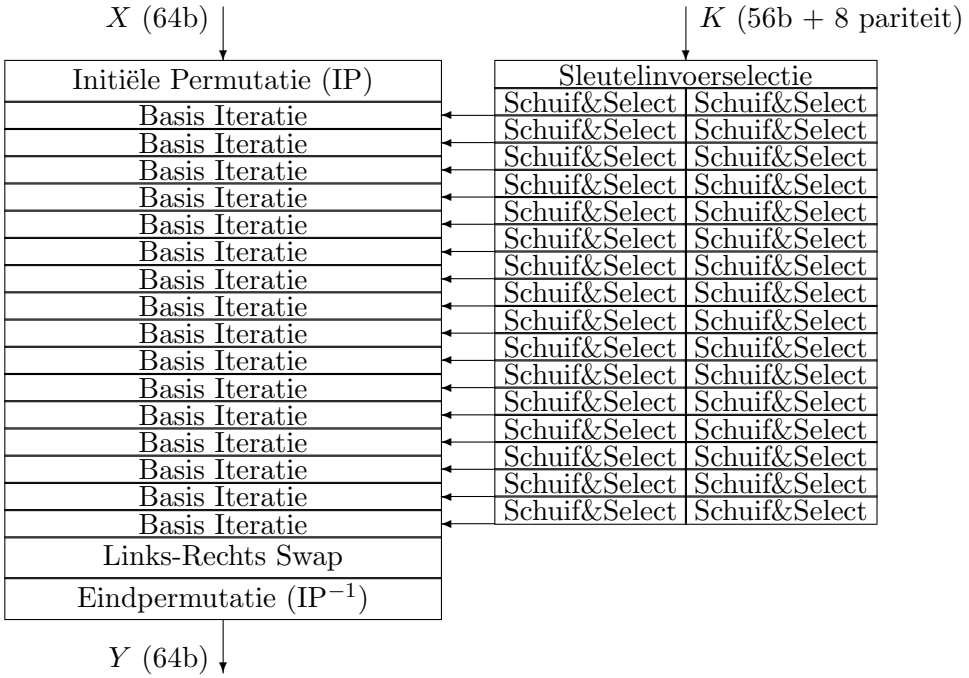
S8, reageert op $c_{43}c_{44}c_{45}c_{46}c_{47}c_{48}$

	Waarde van $c_{44}c_{45}c_{46}c_{47}$															
$c_{43}c_{48}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1101	0010	1000	0100	0110	1111	1011	0001	1010	1001	0011	1110	0101	0000	1100	0111
01	0001	1111	1101	1000	1010	0011	0111	0100	1100	0101	0110	1011	0000	1110	1001	0010
10	0111	1011	0100	0001	1001	1100	1110	0010	0000	0110	1010	1101	1111	0011	0101	1000
11	0010	0001	1110	0111	0100	1010	1000	1101	1111	1100	1001	0000	0011	0101	0110	1011

Tabel 2.4: De S-boxen S1 tot S8.

2.1.4 Overzicht van het algoritme

Figuur 2.5 laat zien hoe DES de data in negentien achtereenvolgende ronden onherkenbaar verandert. De ronden 2 tot en met 17 zijn de reeds uitgebreid besproken DES-iteratie en ronde 18 is het verwisselen van de linker- en rechterdatahelft die ervoor zorgt dat het decrypten goed gaat. De ronden 1 en 19 zijn een permutatie van de databits. Men vond het namelijk niet ingewikkeld genoeg om de bits rechtstreeks in de eerste iteratie in te voeren en daarom wordt dat gedaan in deze volgorde (initiële permutatie):



Figuur 2.5: Het complete DES-algoritme

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

De eindpermutatie is precies de inverse van de initiële permutatie, en daarom hebben de ronden 1 en 19 geen enkele invloed op het decrypten, noch voegen zij iets toe aan de veiligheid van het algoritme. Het enige doel is eventuele concentraties enen en nullen in X direct goed te spreiden over de invoer van de volgende ronden.

Men is natuurlijk vrij te bepalen hoe men de encryptie- en decryptiefuncties uit wil rekenen. Dit kan bijvoorbeeld in software (Tanenbaum [Tan90, p. 602] geeft een compleet programma), maar uitvoering in hardware is veel efficiënter omdat het algoritme hiervoor speciaal is ontworpen. Permutaties van bits, die in het algoritme heel veel voorkomen, kosten in hardware namelijk helemaal niets: zij worden gerealiseerd door verbindingen op chips gekruist in plaats van recht te leggen. Ook de S-boxen hebben een zeer efficiënte hardware-implementatie.

Omdat de berekening in elke ronde hetzelfde is, zijn in principe twee hardware-ontwerpen denkbaar. Men kan de hardware eenmaal uitvoeren en de data en sleutel hier zestienmaal doorvoeren. Als alternatief kan men de hardware zestienvoudig uitvoeren en de data en sleutel steeds laten opschuiven naar de volgende iteratie. Dezelfde hardware kan dan als pipeline worden gebruikt om zeer snel achtereenvolgende versleutelingen uit te voeren (met dezelfde of met verschillende sleutels).

2.2 DES breken met brutekracht

Het DES algoritme heeft vanaf zijn introductie onder zware kritiek gestaan, onder andere vanwege de korte sleutels: al in 1974 werd beseft dat een algoritme met 56-bitssleutels met brute kracht te breken is. Het is alleen wonderlijk dat het een kwart eeuw, tot 1998, heeft geduurd, voordat een kant-en-klare machine aan het publiek werd gepresenteerd.

2.2.1 Benodigde rekenkracht

Met de inzichten uit dit en het vorige hoofdstuk kunnen we al een schatting maken van de benodigde rekenkracht. Als eerste kunnen we gaan inzien dat de kritieke lengte erg laag is. Verzenden we Engelse tekst in ASCII, met een entropie van rond de 25 procent, dan is de kritieke lengte slechts 56b/75% ofwel zo'n tachtig bits. Zelfs met optimistischer aannamen over de entropie blijft de kritieke lengte ver onder de lengte van zelfs maar het kortste e-mailberichtje. Een cijfertekstaanval is dus al op de allerkortste berichtjes mogelijk, en vereist per sleutel het decrypten van slecht enkele DES-blokken (van 64 bits).

In opgave 1.13 keken we al naar de benodigde tijd. Met een miljoen sleutels per seconde schiet het niet op, en met een miljard sleutels per seconde eigenlijk ook nog niet echt. Hoe mobiliseer je voldoende rekenkracht om de klus in bijvoorbeeld een week te klaren? Al in 1977 werden schattingen gemaakt van de kosten die zouden zijn gemoeid met het bouwen van een speciale machine voor dit doel.

Er zijn wel projecten uitgevoerd waarin ongebruikte rekencapaciteit van pc's over de hele wereld werd ingezet voor één cryptografische aanval. De decryptie wordt dan altijd in software uitgevoerd, en met enkele honderdduizenden decrypties per seconde per pc is de uitvoerbaarheid van het idee voor DES twijfelachtig.

Quisquater en Girault suggereerden in 1991 een soort gespecialiseerde supercomputer, opgebouwd door elke Chinees een radio te geven met een speciale DES-chip erin. De aan te vallen tekst wordt per radio uitgezonden, en onmiddellijk starten een miljard processors elk met een miljoen tests per seconde, zodat na ongeveer een minuut alle sleutels zijn uitgeprobeerd. Het lijkt me niet iets voor de Chinese regering om bij de ontrafeling van vijandige staatsgeheimen burgers in te schakelen, en ook worden de organisatorische problemen wat licht ingeschat, maar de boodschap is duidelijk. Het is mogelijk voldoende hardware bijeen te krijgen voor een brutekrachtaanval op DES.

2.2.2 De EFF DES-kraker

We weten niet of en sinds wanneer er DES-krakers in het geheim zijn gebouwd, maar een publieke demonstratie vond voor het eerst plaats in 1998. Het was de Electronic Frontier Foundation (EFF, zie kader 2.6) een doorn in het oog dat de Amerikaanse overheid DES bleef propageren als cryptografische standaard en ontkende over de middelen te beschikken om DES-gecodeerde boodschappen met brute kracht te ontcijferen.

Het EFF-ontwerp [Ele98] lift mee met de goede implementeerbaarheid van DES in hardware. Al jaren leveren diverse chipfabrikanten gigabitimplementaties, dat wil zeggen, chips die (tegen een kostprijs van vijf euro) zo'n vijftig miljoen versleutelingen per seconde uitvoeren. De machine is een mooi voorbeeld van 'meer van hetzelfde'. Bedenk dat het uitproberen van verschillende sleutels onafhankelijk verloopt en een brutekrachtaanval is daarom zeer goed parallelliseerbaar.

Het ontwerp van de machine begint met een zoekeenheid (*search unit*) die 2.500.000 sleutels per seconde test. De speciale *Deep-Crack*-chip combineert 24 zoekeenheden en kan dus zestig

Kader 2.6: De Electronic Frontier Foundation

"Technology is not just a value-neutral set of tools – it embodies the norms, perspectives, and biases of those who create it." – EFF website.
De opmars van automatisering en netwerken kan de belangen van individuen schaden, zeker als de sturing van het proces van wetgeving en onderzoek geheel wordt overgelaten aan de partijen waar kennis en macht is geconcentreerd: overheid en bedrijven. In 1990 richtten Mitchell Kapor en John Barlow een burgerrechtenorganisatie op: de Electronic Frontier Foundation (EFF), met als doel de positie van individuen in het veranderende ICT-landschap te verdedigen.
Dit wordt onder andere gedaan door bewustmaking, bijvoorbeeld met cursussen, door proefprocessen en door projecten die het ongelijk van de genoemde grote spelers kunnen aantonen; hieronder valt de DES-kraker. De EFF strijdt tegen censuur op het internet, bevecht het recht op anonieme communicatie, strijdt voor vrijwaring van providers tegen vervolging op grond van de aangeboden informatie, en bevordert de opbouw van een (anoniem te gebruiken) communicatie-infrastructuur in Kosovo.
Niet iedereen is even blij met het werk van de EFF. Uitgeverijen beweren dat de EFF steun geeft aan softwarepiraterij door kopiëerblokkades te omzeilen en vrije uitwisseling van informatie te bevorderen. Wetshandhavers vinden de EFF op hun weg als zij gebruik van cryptografie aan banden willen leggen om te kunnen blijven afluisteren. Pornobestrijders claimen dat de EFF met het bevechten van vrije meningsuiting op het internet ook de weg opent voor de verspreiding van (kinder-) porno en racistisch materiaal.
De website van de EFF is `www.eff.org`.

miljoen sleutels per seconde verwerken. Een board bevat 64 Deep-Crackchips en heeft dus een capaciteit van 3,84 miljard sleutels per seconde. Een chassis bevat twaalf boards en haalt dus ruim 46 miljard sleutels per seconde. De machine bevatte oorspronkelijk twee chasses en kon ruim 92 miljard sleutels per seconde proberen, waarmee de totale sleutelverzameling wordt doorzocht in 782 duizend seconden, ongeveer negen dagen. De gemiddelde zoektijd is de helft, vier en een halve dag. Figuur 2.7 toont de hoofdontwerper, Paul Kocher, voor een uitgebreidere machine met zes chasses.

Om het ontwerp van de chips simpel te houden, decodeert de search unit slechts twee DES-blokken met elke sleutel, en test of de uitkomst voldoet aan een opgegeven patroon. Is dit het geval, dan wordt de sleutel doorgegeven aan de pc die de gehele machine beheert. De pc decodeert met de ontvangen sleutel meer blokken data om eventuele bastaardsleutels te verwerpen. De search unit telt 1 bij de sleutel op en vervolgt zijn werk.

Figuur 2.7: De ontwerper met de machine.

Voor de test op de uitkomst zijn twee mogelijkheden. De machine kan worden gebruikt voor een klaretekstaanval: dan zijn de eerste twee blokken van de klare tekst bekend en de search unit test op gelijkheid. De kans op een bastaardsleutel is in deze opzet nihil, want een foute sleutel geeft slechts met kans $\frac{1}{2^{128}}$ het goede resultaat. De besturende pc heeft het in deze opzet dus niet druk met het verwerken van bastaarsleutels. De tweede opzet is een cijfertekstaanval, waarbij ASCII als plaintext wordt verondersteld: de search unit test op een 0-bit in elke achtste positie. Het gaat hier om zestien (van de 128) posities, we verwachten dus dat een op de 2^{16} sleutels een bastaardsleutel is en de pc krijgt er $\frac{92.16\times10^9}{2^{16}} = 1.41 \times 10^6$ per seconde ter controle.

De constructie van de machine is zo simpel, dat het ondenkbaar wordt geacht dat er niet eerder zulke machines in het geheim zijn gebouwd. Het project werd binnen een jaar voltooid op een budget van ruim tweehonderdduizend dollar. De Amerikaanse overheid doet de prestatie van de cracker af als niet relevant: een klaretekstaanval, danwel cijfertekstaanval met ASCII-berichten zou niet representatief zijn voor de problemen waarmee een aanvaller bij een 'meer realistische' aanval te maken zou krijgen. De machine is echter gebouwd om te bewijzen dat de rekenkracht voor het uitproberen van 2^{56} sleutels bijeen gebracht kan worden. Een 'meer realistische' toepassing van DES vereist natuurlijk een ingewikkelder test in de search unit, maar het zou erg onverstandig zijn om op de onhaalbaarheid van zo'n test te vertrouwen.

2.3 Differentiële cryptanalyse

De brutekrachtaanval uit de vorige sectie benadert de DES-functie als een black box; zijn er efficiëntere aanvallen mogelijk door gebruik te maken van de interne structuur van het algoritme? Het werk voor een brutekrachtaanval is te halveren met een eenvoudige relatie die wordt uitgewerkt in opgave 2.5.

In deze sectie beschrijven we zo'n aanval op een versleuteling die niet zestien, maar drie ronden gebruikt. De eerste beschrijving is als een *gekozen-klaretekstaanval* of *Chosen Plaintext Attack*. Het uitvoeren van zo'n aanval vereist dat de aanvaller een aantal *zelfgekozen* strings X kan aanbieden aan de encryptiefunctie en de beschikking krijgt over de bijbehorende encryptie Y. De aanval verschilt dus van de klaretekstaanval (die soms ter onderscheid ook wel *gekende-klaretekstaanval* wordt genoemd) omdat de aanvaller de klare tekst die bij een bepaalde cijfertekst hoort niet alleen kent, maar die ook zelf kan kiezen.

Notatie van de aanval. Het principe van de differentiële cryptanalyse werd door Biham en Shamir [BS90] beschreven en toegepast op DES. De analyse heet differentieel omdat steeds wordt gekeken naar het verschil (XOR) tussen overeenkomstige bits in de versleutelprocedure bij het versleutelen van twee invoeren. De invoer- en uitvoerpermutatie en het verwisselen van datahelften, zoals toegepast in ronden 1, 18 en 19 van DES, zijn voor deze aanval irrelevant en worden hier weggelaten. We stellen ons daarom de werking van het algoritme voor als in figuur 2.2 en ook de notatie is overeenkomstig: de twee invoeren zijn $X = (L, R)$ en $X' = (L', R')$, en verdere waarden in figuur 2.2 noemen we L_1, L'_1, etc. De aanvaller kan X en X' bepalen en Y en Y' verkrijgen.

De aanval poogt de 48 sleutelbits van de laatste ronde te verkrijgen door een analyse van de invoer en uitvoer van de laatste S-boxen. We benoemen daarom die invoer en uitvoer en de verschillen als volgt (*Exp*, *Sbx* en *Pbx* staan voor de expansie, S-boxen en P-box):

$$\begin{array}{llllll} A &= Exp(R_2) & A' &= Exp(R'_2) & A^{\oplus} &= A \oplus A' \\ B &= A \oplus K_3 & B' &= A' \oplus K_3 & B^{\oplus} &= B \oplus B' \\ C &= Sbx(B) & C' &= Sbx(B') & C^{\oplus} &= C \oplus C' \\ F &= Pbx(C) & F' &= Pbx(C') & F^{\oplus} &= F \oplus F' \end{array}$$

Merk op dat in beide versleutelingen *dezelfde sleutel* K_3 wordt gebruikt. We schrijven ook $X^{\oplus}$ voor $X \oplus X'$, en $R_2^{\oplus}$ voor $R_2 \oplus R'_2$ et cetera.

In- en uitvoer-XORs van de S-boxen. Eerste doel bij de aanval is het verkrijgen van de in- en uitvoer-XOR van de laatste S-box, dit zijn de strings $B^{\oplus}$ en $C^{\oplus}$; zij worden uit de bekende waarden X, Y et cetera afgeleid. Het gemakkelijkst vindt de aanvaller de waarde $B^{\oplus}$.

Propositie 2.1 $B^{\oplus} = Exp(L_3^{\oplus})$.

Bewijs. We gebruiken hier dat het nemen van XOR commuteert met het permuteren en expanderen van bits (we spreken van *lineaire* operaties); bovendien valt het XOR-en met de ons onbekende sleutel K_3 geheel weg!

$$\begin{array}{lll} B^{\oplus} &= B \oplus B' & \text{Definitie} \\ &= (A \oplus K_3) \oplus (A' \oplus K_3) & \text{Berekening van } B \text{ in DES} \\ &= A \oplus A' & \text{Tweemaal XOR met } K_3 \text{ valt weg} \\ &= Exp(R_2) \oplus Exp(R'_2) & \text{Berekening van } A \text{ in DES} \\ &= Exp(R_2^{\oplus}) & \text{XOR commuteert met expansie} \\ &= Exp(L_3^{\oplus}) & L_3 \text{ is } R_2 \text{ bij DES} \end{array}$$

Merk op dat $L_3^{\oplus}$ beschikbaar is als de linkerhelft van $Y^{\oplus}$. □

Het is lastiger om vat te krijgen op de uitvoer van de S-boxen, maar dit kan door de invoerparen zo te kiezen dat $R_0^{\oplus} = 0$ (ofwel $R_0 = R_0'$).

Propositie 2.2 *Als* $R_0^{\oplus} = 0$ *dan is* $C^{\oplus} = Pbx^{-1}(R_3^{\oplus} \oplus L_0^{\oplus})$.

Bewijs. Met $R_0 = R_0'$ is ook $F(R_0, K_1) = F(R_0', K_1)$, met andere woorden, L_0 en L_0' worden in de eerste ronde met dezelfde (maar ons onbekende) waarde F_1 geXORd, waarmee

$$L_2^{\oplus} = R_1^{\oplus} = (L_0 \oplus F_1) \oplus (L_0' \oplus F_1) = L_0^{\oplus}.$$

Nu is $C^{\oplus} = Pbx^{-1}[F(R_2, K_3) \oplus F(R_2', K_3)]$, en

$$F(R_2, K_3) = R_3 \oplus L_2 \qquad \text{en} \qquad F(R_2', K_3) = R_3' \oplus L_2'$$

(door het terugrekenen van de manier waarop R_3 en R_3' worden bepaald) dus

$$\begin{array}{lll} C^{\oplus} & = Pbx^{-1}[(R_3 \oplus L_2) \oplus (R_3' \oplus L_2')] & \text{Zie hierboven} \\ & = Pbx^{-1}[(R_3 \oplus R_3') \oplus (L_2 \oplus L_2')] & \text{Hergroepeer termen} \\ & = Pbx^{-1}[R_3^{\oplus} \oplus L_2^{\oplus}] & \text{Definitie} \\ & = Pbx^{-1}[R_3^{\oplus} \oplus L_0^{\oplus}] & \text{Omdat } L_2^{\oplus} = L_0^{\oplus} \end{array}$$

Merk op dat $R_3^{\oplus}$ en $L_0^{\oplus}$ bekend zijn als de rechterhelft van $Y^{\oplus}$ en de linkerhelft van $X^{\oplus}$. □

De aanvaller kent invoer en uitvoer van de encryptie en kan dus $B^{\oplus}$ en $C^{\oplus}$ bepalen voor elk paar waarbij $R_0^{\oplus} = 0$.

Het achterhalen van de sleutelbits. Hoe de 48 sleutelbits van de derde iteratiesleutel uit de in- en uitvoer-XORs van de S-boxen worden achterhaald gaan we illustreren aan de hand van S-box S1. De S-box S1 heeft zes invoerbits en bij bekende $B^{\oplus}$ zijn er daarom 64 mogelijkheden voor de daadwerkelijke combinatie B, B' die in de box is gegaan. Definieer $\Delta(B^{\oplus})$ als de verzameling mogelijke combinaties met invoer-XOR $B^{\oplus}$, dat wil zeggen,

$$\Delta(B^{\oplus}) = \{(B, B') | B \oplus B' = B^{\oplus}\}.$$

Voor elk van de 64 paren in $\Delta(B^{\oplus})$ bepalen we de uitvoer-XOR die erbij hoort, dat wil zeggen, voor elk van de 64 waarden van B berekenen we

$$C^{\oplus} = S_1(B) \oplus S_1(B \oplus B^{\oplus}).$$

De uitvoer-XOR is vier bits breed, er zijn dus maar zestien uitkomsten. Deze staan opgesomd voor de voorbeeld string $B^{\oplus} = 110100$ in tabel 2.8. De sleutelbits die bovenaan worden genoemd zijn de sleutelbits die in S1 worden gebruikt.

Er zijn veel tabellen nodig, want voor elk van de acht S-boxen en elk van de 64 $B^{\oplus}$'s wordt de tabel opgezet. De tabellen worden als volgt gebruikt. Nadat de bijbehorende waarden van $B^{\oplus}$ en $C^{\oplus}$ zijn bepaald, geeft de tabel een aantal (typisch tussen zes en zestien) mogelijkheden $IN(S_1, B^{\oplus})[C^{\oplus}]$ voor de invoer van S1. Het rijtje sleutelbits (51, 27, 10, 36, 25, 58) gebruikt voor die S-box is dan $y \oplus Exp(L_3)$ voor een van de strings y uit de tabel.

Elk gebruikte (X, Y) paar wordt dus gebruikt om het aantal mogelijke sleutels terug te brengen, en doorgaans zijn ongeveer drie paren nodig om de laatste iteratiesleutel uniek te bepalen. Daarmee zijn 48 van de 56 sleutelbits bekend, en een vervolg is nodig om uit de 256 overgebleven mogelijke sleutels de juiste te kiezen.

S-Box	S1
Sleutelbits	51, 27, 10, 36, 25, 58
$B^{\oplus}$	110100

Tabel $IN(S_1, 110100)$:		
$C^{\oplus}$	komt voor bij $B = \ldots$	aantal
0000		0
0001	000011, 001111, 011110, 011111, 101010, 101011, 110111, 111011	8
0010	000100, 000101, 001110, 010001, 010010, 010100, 011010, 011011, 100000, 100101, 010110, 101110, 101111, 110000, 110001, 111010	16
0011	000001, 000010, 010101, 100001, 110101, 110110	6
0100	010011, 100111	2
0101		0
0110		0
0111	000000, 001000, 001101, 010111, 011000, 011101, 100011, 101001, 101100, 110100, 111001, 111100	12
1000	001001, 001100, 011001, 101101, 111000, 111101	6
1001		0
1010		0
1011		0
1100		0
1101	000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010	8
1110		0
1111	000111, 001010, 001011, 110011 111110, 111111	6

Tabel 2.8: Een van de 512 IN-tabellen: $IN(S_1, 110100)$.

Klaretekstaanval en meer DES-iteraties. We hebben de aanval beschreven als een gekozen-klaretekstaanval, waarbij Oscar paren X, X' kon encrypten met $R_0^{\oplus} = 0$ en de resulterende Y, Y' vernemen. De berekening is natuurlijk ook uit te voeren met een willekeurige reeks bekende (X, Y) paren, maar dan zijn ze niet allemaal bruikbaar. Omdat er 2^{32} mogelijkheden voor R zijn, vinden we volgens de verjaardagsstelling (stelling A.6) in een tekst met lengte ongeveer $\sqrt{2^{32}}$ twee symbolen X en X' met $R = R'$, dus $R^{\oplus} = 0$. Als we in een willekeurige reeks wachten op *drie* van zulke bruikbare paren (genoeg om de sleutel te vinden als bovenbeschreven), dan moeten we (verwacht) over ongeveer 2^{17} bekende invoer/uitvoerparen beschikken. De conclusie is, dat een DES-algoritme met slechts drie iteraties kwetsbaar is tegen een differentiële aanval op een gekozen klare tekst van drie blokken, of een klaretekstaanval van 2^{17} gekende blokken. Dit is weergegeven in tabel 2.9.

Het principe van de differentiële aanval is ook uitvoerbaar voor versleuteling met meer DES-iteraties. Met het groeien van het aantal wordt de aanval gecompliceerder, omdat de input-XOR van de laatste S-boxen niet meer met zekerheid is te bepalen (vergelijk propositie 2.1) maar alleen met een bepaalde kans, en ook weer wanneer de juiste invoer-XOR $X^{\oplus}$ wordt gebruikt. Begrijpelijk neemt ook het aantal benodigde paren (X, X') toe. De lengte van een willekeurige reeks waarin voldoende paren voorkomen groeit (zie weer tabel 2.9) tot 2^{58} bij zestien iteraties... en dit is meer dan het verwachte aantal decrypties dat wordt gedaan in een brutekrachtaanval!

DES-iteraties	Gekozen klare teksten	Gekende klare teksten
3	3	2^{17}
8	2^{14}	2^{44}
12	2^{31}	2^{47}
14	2^{39}	2^{51}
15		2^{52}
16	2^{47}	2^{58}

Tabel 2.9: Kwetsbaarheid van DES afhankelijk van iteraties.

Een differentiële aanval tegen het DES-algoritme op basis van een gekende klare tekst vereist evenveel werk als een brutekrachtaanval.

Biham en Shamir hebben aangetoond dat de kwetsbaarheid van het algoritme tegen deze aanval zeer sterk afhangt van de frequentieverdeling in de laatste kolom van tabel 2.8. Worden de S-boxen random gekozen, dan is de verwachte bescherming uiterst gering, en het veranderen van een enkele waarde in de S-boxen (ten opzichte van tabel 2.4) kan de bescherming dramatisch doen dalen. Dit toont aan dat bescherming tegen differentiële aanvallen een rol heeft gespeeld bij het ontwerp, en dat de NSA niet alleen op de hoogte was van de principes achter deze aanval, maar ook van hoe de bescherming ertegen is op te bouwen.

2.4 Opkomst en ondergang van een standaard

Het is in de voorafgaande secties duidelijk geworden dat DES, ondanks zijn algoritmische sterkte, aan het eind van zijn levensduur is gekomen door de beperkte sleutellengte. Deels wordt dit probleem overkomen door drievoudige toepassing als *'triple DES'* (3DES); hierop gaan we in sectie 4.2.3 uitgebreider in. We proberen hier een overzicht te geven van de meest turbulente gebeurtenissen uit het leven van een encryptiealgoritme.

1970 De cryptografie wordt nog niet op wetenschappelijk niveau beoefend. IBM beschikt over een systeem *Lucifer*.

1972 In de bancaire wereld groeit de belangstelling voor beveiligde datacommunicatie en ontstaat behoefte aan een gestandaardiseerd, veilig encryptiealgoritme. Het Amerikaanse National Bureau of Standards (NBS) doet een oproep om algoritmen aan te melden die kunnen worden ontwikkeld tot een nationale standaard (DES = Data Encryption Standard).

1974 Bij IBM worden de principes achter Lucifer verder verbeterd, en het resultaat wordt als voorstel voor DES bij NBS ingediend. Het NBS laat het ontwerp kritisch bekijken door de National Security Agency (NSA) (zie kader 2.10). De NSA wijzigt het ontwerp: de datablokgrootte wordt teruggebracht van 128 naar 64 bits en de sleutelgrootte van 112 naar 56 bits. Er gaan geruchten dat ook de S-boxtabellen op last van de NSA zijn gewijzigd.

1975 Het DES-algoritme als in dit hoofdstuk beschreven wordt gepubliceerd en aangekondigd als officiële standaard hetgeen leidt tot verbazing, verontwaardiging en enthousiasme van drie zijden.

1. De NSA zag knarsetandend dat het door haar goedgekeurde ontwerp openbaar werd gemaakt. Bij het beoordelen en veranderen van het algoritme was de organisatie ervan uitgegaan dat het algoritme slechts als 'black box' (bijvoorbeeld in de vorm van chips) beschikbaar zou worden gemaakt. Medewerking aan het DES-project zou zijn geweigerd als men van tevoren had geweten dat het DES-algoritme openbaar zou worden.
2. De wetenschappelijke wereld kreeg met de publicatie van DES iets in handen wat door de NSA was goedgekeurd en dit gaf een grote stimulans aan cryptografisch onderzoek. Waarom moest DES zestien iteraties hebben? Wat was het geheim van de S-boxen?
3. Burgerrechtenorganisaties zagen een grote bedreiging in het gebruik van een algoritme waarvan het ontwerp door de NSA was beïnvloed. Men vreesde dat door het terugdringen van de sleutellengte het uitvoeren van een brutekrachtaanval binnen het bereik van juist de NSA was gekomen. Men vreesde ook dat achter het wijzigen van de S-boxen een truc zat waardoor de NSA in staat was DES-communicatie snel te kraken.

1976 DES wordt met ingang van 1977 tot standaard uitgeroepen, en de bruikbaarheid zal elke vijf jaar worden herzien; er zou dus hercertificering moeten plaatsvinden in 1982, 1987, 1992 et cetera.

1980 Rond deze tijd komt het gebruik van DES echt op gang met de ontwikkeling van standaarden en applicaties voor bancair verkeer; veel overigens op basis van drievoudige toepassing (3DES). De controverse rond DES laat de banken koud, want zij beseffen dat de NSA, zo deze al in staat zou zijn DES-communicatie te ontcijferen, andere prioriteiten heeft dan het ontregelen van het elektronisch betalingsverkeer.

1983 De eerste hercertificering vindt verlaat, maar zonder veel weerstand plaats.

1987 De tweede hercertificering biedt meer spektakel. De NSA zegt dat DES niet lang meer veilig zal zijn en biedt een alternatief aan: een nieuw, door NSA ontwikkeld algoritme dat geheim zal blijven maar aan het publiek zal worden aangeboden in de vorm van gesloten chips. Na de controverse van de beginperiode kwam juist het vertrouwen in DES op gang: men besefte dat eventuele achterdeurtjes in het ontwerp nu toch wel zouden zijn uitgelekt, en er waren na bijna vijftien jaar nog steeds geen geslaagde kraken op DES geweest. Daarom was er grote weerstand tegen een nieuw, geheim algoritme en DES werd opnieuw voor vijf jaar gecertificeerd.

Ondertussen werken diverse groepen aan de ontwikkeling van mogelijke vervangers voor DES, vrijwel allemaal op dezelfde leest geschoeid: het in achtereenvolgende ronden modificeren van data, vaak volgens het Feistel-principe of een variant daarop. We noemen hier IDEA (International Data Encryption Algorithm), een systeem met 128-bitssleutels en gebruikmakend van 16-bits aritmetische operaties om software-implementaties efficiënt te maken.

1990 Differentiële cryptanalyse wordt op DES toegepast, maar blijkt geen verbetering ten opzichte van brute force (zie sectie 2.3). De alternatieven waaraan wordt gewerkt, blijken allemaal kwetsbaar tegen een differentiële aanval en verdwijnen geruisloos van het toneel.

Kader 2.10: NSA? 'No Such Agency'

De NSA (National Security Agency) werd in 1952 opgericht met als doel de Amerikaanse belangen op het gebied van informatie-uitwisseling te beschermen. Dat houdt onder meer in: veel cryptografisch onderzoek en grote geheimhouding, vandaar de bovenvermelde bijnaam van dit buro. Inmiddels heeft de NSA een homepage: `www.nsa.gov`.
De organisatie staat bekend als Amerika's grootste werkgever van wiskundigen en de wereldwijd grootste afnemer van computerapparatuur. Binnen de muren van NSA is een enorme hoeveelheid cryptografische kennis aanwezig, maar onderzoekers 'van buiten' kunnen slechts raden hoever men op de buitenwereld vooruit is. Tekenend is dat de industriële en academische wereld vijftien jaar hebben gespeurd naar principes en geheimen achter DES, die bij de NSA al twintig jaar lang bekend bleken.
De NSA dient geen algemene, maar *Amerikaanse* belangen; zonder Amerikaans staatsburgerschap hoeft men niet te solliciteren. Mensen vermoorden doet de NSA niet (volgens de website), maar het tegenhouden van cryptografische publicaties (Diffie en Landau [DL98, p. 62]), het binnendringen in Europese regeringscomputers en het op grote schaal afluisteren van gsm-communicatie behoort tot de gangbare praktijk.

De NSA verklaart nu, dat het principe van de differentiële aanval al voor 1970 bij de NSA bekend was. Het DES-ontwerpteam had deze vitale kennis herontdekt, maar in het belang van de nationale veiligheid werd hen verzocht hierover te zwijgen. De S-boxen zijn (door IBM en NSA) zo aangepast dat de bescherming tegen deze aanval maximaal is, en het aantal ronden is precies zo gekozen dat de differentiële aanval met gekende klare tekst even complex is als een brutekrachtaanval. Uiteraard kon men deze overwegingen in 1974 niet onthullen zonder de veiligheid van de dan bestaande algoritmen te schaden.

1992 De derde hercertificering wordt een pijnlijk verhaal. De kleren van Keizer DES zijn erg dun geworden maar er is eenvoudigweg *geen alternatief*. De ontwerpers van IDEA claimen hun ontwerp aan te passen om bescherming tegen differentiële aanvallen te geven, maar het vertrouwen in het nieuwe algoritme is nog niet voldoende. In de loop van de jaren negentig wint het gebruik van IDEA terrein, bijvoorbeeld in PGP. DES wordt opnieuw voor vijf jaar gecertificeerd maar men besluit tegelijkertijd een programma op te zetten dat moet leiden tot een nieuwe standaard.

1997 Het Amerikaanse National Institute of Standards and Technology (NIST, voortzetter van NBS) publiceert een lijst van eisen waaraan het nieuwe standaard algoritme wordt geëvalueerd, en in 1998 worden vijftien voorstellen ingediend als kandidaat voor de Advanced Encryption Standard (AES); zie sectie 4.5.

1998 De Electronic Frontier Foundation sluit het tijdperk van DES als encryptiealgoritme definitief af met de demonstratie van de werkende DES-kraker, volgens de methode die al rond 1975 was voorspeld.

Rond de eeuwwisseling werd DES in enkele vorm nog sporadisch toegepast, maar 3DES (sectie 4.2.3) is nog steeds het meest gebruikte cryptografische algoritme.

Samenvatting en conclusies

Data Encryption Standard (DES) is het meest invloedrijke symmetrische algoritme. Hoewel het gebruik ervan zal afnemen, is het voor studie van de cryptografie interessant omdat DES heel veel principes van algoritmen (en aanvallen erop) in zich draagt.

DES manipuleert steeds een deel van de data door puntsgewijze XOR met een bitrij, die afhangt van de sleutel en de rest van de data. Dit gebeurt in zestien achtereenvolgende ronden, de data wordt in blokken van 64 bits bewerkt en de sleutel heeft 56 bits.

Met DES voor ogen hebben onderzoekers zeer ingenieuze technieken ontwikkeld om cryptografische algoritmen aan te vallen. Zo is er bijvoorbeeld de differentiële cryptanalyse, waarin steeds twee datablokken worden bewerkt, en gekeken wordt naar welke tussenresultaten verschillen bij de twee operaties. Hoewel de techniek voor DES werd ontwikkeld, is DES er niet mee te kraken omdat het algoritme er erg goed tegen is beschermd.

Een veel banalere aanval is succesvol tegen DES: het aantal mogelijke sleutels is zo klein, dat het haalbaar is ze allemaal uit te proberen. Een speciale machine hiervoor werd in 1998 publiek gedemonstreerd, maar het is zeer waarschijnlijk dat er eerder al in het geheim zulke projecten zijn geweest.

DES wordt nog steeds gebruikt, maar de data wordt dan driemaal achtereen versleuteld, met verschillende sleutels. Een brutekrachtaanval is dan onmogelijk, met de huidige techniek (of de komende jaren).

Opgaven bij hoofdstuk 2

Opgave 2.1 DES Chronologie. *(Uit het tentamen van oktober 1996.)*
Deze jaartallen spelen een rol in de geschiedenis van de Data Encryption Standard: 1974, 1977, 1990, 1997. Wat gebeurde er in die jaren met DES?

Opgave 2.2 *Cryptografiedocente Alice stuurt wekelijks aan onderwijsassistent Bob een email (ASCII-bestand) met regels van de volgende vorm:*

```
ccccccc aaa bbb
```

Hierin is `ccccccc` *een zevencijferig studentnummer,* `aaa` *(*`AAN` *of* `AFW`*) geeft aan of de student het college heeft bezocht en* `bbb` *(*`WEL` *of* `NEE`*) of de student op de werkgroep extra moet worden geholpen. Alice en Bob gebruiken DES en hun key is niet uitgelekt. Hoe kan student Oscar na een paar weken toch van hun plannen op de hoogte blijken te zijn?*

Opgave 2.3 *Er bestaan paren DES-sleutels k_1 en k_2 met de eigenschap dat* encryptie *met k_1 hetzelfde is als* decryptie *met k_2. Geef een voorbeeld van zo'n paar.*

Opgave 2.4 *Bestaat er voor DES een* self-inverting key*, dat wil zeggen, een sleutel k waarvoor geldt dat $E_k = D_k$?*

Opgave 2.5 *Laat zien dat voor DES-encryptie geldt $E_{\overline{k}}(\overline{X}) = \overline{E_k(X)}$. (Waar $\overline{s}$ staat voor de bitsgewijze inverse van s.)*
Laat zien hoe je een brutekrachtaanval met vijftig procent kunt versnellen als je over de encrypties Y en Y' van bekende strings X en $X' = \overline{X}$ beschikt.

Opgave 2.6 *Er bestaan vier DES-sleutels met de eigenschap dat er uit de sleutelfabriek in elke ronde* exact dezelfde *iteratiesleutel rolt. Welke zijn dat?*

Opgave 2.7 DES Ronden. *(Uit het tentamen van januari 1997.)*
Beschrijf de werking van een ronde van DES en leg uit, waarom deze operatie zijn eigen inverse is. Hoeveel ronden zijn er en waarom is dit aantal gekozen?

Opgave 2.8 *Hoe organiseer je een gekozen-klaretekstaanval? Op een geldautomaat, een telefoon, een e-mailsysteem, een PrimaCryptionkaart (kader 4.1)?*

Opgave 2.9 *Bereken het aantal MIPS-jaar nodig voor een brutekrachtaanval zoals uitgevoerd in de DES-kraker. Hoeveel MIPS-jaar zou nodig zijn voor een aanval op een systeem met 112-bitssleutels? (Reken voor een DES-decryptie één instructie.)*

Opgave 2.10 *Overtuig je ervan dat tabel 2.8 de volgende eigenschappen heeft, en verklaar ze:*

- *Alle getallen in de rechterkolom zijn even.*
- *De getallen in de rechterkolom sommeren tot 64.*
- *String b in de middenkolom staat met $b \oplus 110100$ in hetzelfde vakje.*
- *Elke 6-bitsstring komt eenmaal voor in de middelste kolom.*
- *Er komen nullen voor in de rechterkolom.*

Hoofdstuk 3

Advanced Encryption Standard

Door *Marries van de Hoef*

De Advanced Encryption Standard is een algoritme om data te versleutelen, en hierin sinds 2001 de opvolger van DES (hoofdstuk 2). Dit is nuttig omdat vertrouwelijke informatie geheim kan blijven, ook wanneer deze mogelijk onderschept wordt. Aan de versleutelde tekst mag de oorspronkelijke tekst dus niet te zien zijn. De oorspronkelijke tekst mag alleen terug te halen zijn met de sleutel.

De sleutel die gebruikt wordt bij het versleutelen is bij AES gelijk aan de sleutel die nodig is voor het ontsleutelen, dit maakt AES een symmetrisch encryptie algoritme. Er zijn ook asymmetrische encryptie algoritmes, bijvoorbeeld RSA (hoofdstuk 6). Een symmetrisch algoritme kan veel sneller uitgevoerd worden dan een asymmetrisch algoritme, maar heeft als nadeel dat de sleutel bij beide partijen beschikbaar moet zijn. Bijvoorbeeld bij het maken van een verbinding over het internet is het niet praktisch als er eerst een sleutel uitgewisseld moet worden via een onafhankelijk, veilig kanaal. Vaak wordt bij beveiligde communicatie over het internet een asymmetrisch encryptie algoritme gebruikt om een sleutel te versturen voor een symmetrisch encryptie algoritme zoals AES. Beveiligde internet verbindingen die bijvoorbeeld bij internetbankieren gebruikt worden, zijn op deze manier afhankelijk van onder andere AES. Het is dus van groot belang dat de versleuteling van AES niet gekraakt kan worden.

In dit hoofdstuk wordt uitgelegd hoe AES werkt en waarom het niet makkelijk te kraken is. Eerst wordt in sectie 3.1 verteld waarom AES nodig was en hoe het tot stand is gekomen. Deze achtergrondinformatie maakt het duidelijker waarom sommige keuzes in het algoritme zijn gemaakt. In sectie 3.2 wordt de algoritmische werking uitgelegd. Ook wordt uitgelegd waarom het algoritme juist op deze manier in elkaar steekt. Vervolgens wordt in sectie 3.3 besproken hoe AES in een applicatie gebruikt moet worden, en hoe het al daadwerkelijk gebruikt wordt. Daarna wordt in sectie 3.4 bekeken welke soorten aanvallen er op AES gedaan zijn. Tot slot wordt er een blik op de toekomst geworpen om een conclusie te trekken over het verdere gebruik van AES.

3.1 Ontstaan

3.1.1 Tekortkomingen van DES

AES is niet de eerste standaard op het gebied van symmetrische encryptie. Zijn voorganger is DES, Data Encryption Standard (hoofdstuk 2). Deze is in de jaren zeventig voortgekomen uit een groeiende behoefte naar een standaard op het gebied van encryptie. Voordat DES geïntroduceerd werd, heeft het Amerikaanse National Security Agency een aantal details van het algoritme gewijzigd. De precieze wijzigingen zijn nooit bekend gemaakt, maar het is zeker dat de sleutellengte gehalveerd werd tot 56 bits [Joh98]. Dat de Amerikaanse overheid het algoritme heeft gewijzigd heeft veel argwaan gewekt, want misschien zou er wel een achterdeur in het algoritme gemaakt zijn. De achterdeur is nooit gevonden, en er zijn ook geen andere serieuze zwakheden van DES aan het licht gekomen [Cop94]. Door de open verspreiding van DES is het de standaard geworden, en vele jaren gebruikt.

Begin jaren negentig kwam DES in de problemen. Door de toenemende kracht van computers kwam het kraken van DES binnen bereik. De reden hiervoor is dat de sleutel maar 56 bits lang is, waardoor er 2^{56} sleutels mogelijk zijn. De snelheid waarmee deze allemaal uitgeprobeerd konden worden nam toe, en zo zou de goede sleutel in een redelijke tijd gevonden kunnen worden. Deze aanval wordt een brute kracht aanval genoemd (in het Engels bekend als brute force attack). Om zo'n aanval te ontmoedigen, werd een variant met 168 bits sleutellengte bedacht: Triple DES. Dit houdt in dat het DES algoritme met drie (onafhankelijke) sleutels herhaald uitgevoerd moet worden. Het lijkt misschien logischer om DES maar twee keer te herhalen, maar deze aanpak biedt geen grote verbetering boven DES (hoofdstuk 4. Door deze herhaling wordt namelijk een nieuwe soort aanval mogelijk, de meet-in-the-middle aanval van Diffie en Hellman [DH77b]. Hiervoor is een aantal bekende paren versleutelde en ontsleutelde data nodig. Bij de aanval versleutelen we de onversleutelde data door één DES-stap met alle mogelijke sleutels uit te voeren, en we ontsleutelen de versleutelde data door één omgekeerde DES-stap met alle mogelijke sleutels uit te voeren. De tussenresultaten die bij deze versleuteling en ontsleuteling ontstaan kunnen we filteren, omdat we weten dat het tussenresultaat bij de versleuteling en ontsleuteling hetzelfde moet zijn. Zo houden we een stuk minder mogelijke sleutels over en door het proces een aantal keer te herhalen met andere bekende paren van versleutelde en onversleutelde data kunnen we de gebruikte sleutels achterhalen. Dit is niet significant langzamer dan een brute kracht aanval op DES.

Bij Triple DES kan deze aanval ook uitgevoerd worden, maar op twee van de drie DES versleutelingen. Triple DES is daarom maar twee keer zo veilig als DES, terwijl er drie keer zo veel berekend moet worden. De bescherming die Triple DES biedt zou volgens het National Institute of Standards and Technology (NIST) voldoende zijn tot 2030 [BBB+07].

3.1.2 AES

Het nadeel van Triple DES is dat dit algoritme drie keer zo langzaam is als DES [SASR01]. Daarom ging het NIST opzoek naar een nieuw algoritme, en deed een openbare oproep voor kandidaten voor de opvolger [BM97]. Er volgden maar liefst vijftien reacties, waarvan de vijf beste algoritmen overbleven [NBD+]. Van de vijf finalisten is uiteindelijk het Rijndael algoritme gekozen als winnaar. Dit algoritme is ontwikkeld door de twee Belgen Joan Daemen en Vincent Rijmen. De redenen dat er voor het Rijndael algoritme is gekozen [NBB+]:

- Het algoritme kan zeer snel worden uitgevoerd in software en hardware, en vereist weinig geheugen. Het algoritme kan dus in allerlei omgevingen gebruikt worden, waaronder op chipkaarten.

- Ten opzichte van de andere kandidaten is Rijndael makkelijk te beschermen tegen side-channel aanvallen. Zie voor meer informatie over deze aanvallen sectie 3.4.4 en hoofdstuk 12.
- Op instructie-niveau is Rijndael goed te parallelliseren, waardoor het algoritme voordeel kan halen uit multikernprocessoren.

Ten opzichte van Triple DES is het AES algoritme zelfs 5 keer zo snel [SL00]!

De openbare oproep van het NIST heeft dus een goed algoritme opgeleverd, en kan zeker een succes genoemd worden. Bij de daadwerkelijke omdoping van Rijndael naar AES, zijn de toegestane blok en sleutel groottes gelimiteerd. De eisen die het NIST had gesteld bij de oproep waren dat het algoritme een blokgrootte van 128 bits en sleutel groottes van 128, 196 en 256 bits moet ondersteunen. Daarbij vermeldden ze dat als het gekozen algoritme meer blokgroottes ondersteunde, ze zouden overwegen deze extra blokgroottes in AES op te nemen [BM97]. Rijndael ondersteunde meer blokgroottes, maar de ondersteuning hiervoor is toch uit AES geschrapt [NIS01]. De reden hiervoor is dat de overige opties niet in het selectieproces voor AES zijn bekeken [DR02]. Dit is merkwaardig omdat er was gezegd dat het overwogen zou worden, terwijl dit in het selectieproces niet is meegenomen.

3.2 Werking van AES

AES versleutelt invoerdata per blok van 128 bits. Op een blok worden herhaaldelijk dezelfde bewerkingen gedaan. Elke herhaling noemen we een ronde. Het enige dat tussen de rondes verschilt, is de rondesleutel. Deze rondesleutel wordt afgeleid van de hoofdsleutel. Het aantal rondes hangt af van de lengte van de hoofdsleutel. Voor sleutels van 128, 196 en 256 bits worden respectievelijk 10, 12 en 14 rondes uitgevoerd. Alle bewerkingen die gedaan worden zijn omkeerbaar. Voor de ontsleuteling moet dus het algoritme van achter naar voor worden doorgelopen, met de omgekeerde bewerkingen.

3.2.1 Wiskundige achtergrond

De berekeningen worden op byte-niveau uitgevoerd, en hierbij moet rekening gehouden worden met de maximale grootte van een byte. Door een optelling of vermenigvuldiging mag het getal niet groter worden dan 255, de maximale waarde van een byte. Een eenvoudige manier om dit te doen, is om alle berekeningen modulo 256 uit te voeren. Dit is geen goed idee voor een cryptografisch algoritme, omdat de berekening niet inverteerbaar is. Stel dat bij de versleuteling $133 \cdot 2 = 266$ berekend wordt, dan wordt dit getal door de modulus van 256 veranderd naar het getal 10. Bij het ontsleutelen draaien we de berekening om, en we moeten dan 10 delen door 2. Nu komen we in problemen, want hoe weten we of $\frac{10}{2}$ gelijk is aan 5 of aan 133 ($= \frac{(256+10)}{2}$)?

Multiplicatieve inverse. Dit kunnen we kloppend maken door de operatie delen te berekenen als vermenigvuldigen met de *multiplicatieve inverse* (hoofdstuk 5), want dan is er een unieke uitkomst. De multiplicatieve inverse van getal a is de waarde waarvoor geldt dat als je a ermee vermenigvuldigt en dan modulo 256 doet, dat je dan het getal 1 krijgt. De multiplicatieve inverse van bijvoorbeeld 5 is 205, want $5 \cdot 205 \equiv_{256} 1$ (hierbij betekent $\equiv_{256}$ dat de berekening modulo 256 gedaan moet worden). Dus om bijvoorbeeld $\frac{10}{5}$ te berekenen

doen we $10 \cdot 205 \equiv_{256} 2$. Door gebruik te maken van deze multiplicatieve inverse is er een unieke uitkomst. Maar de multiplicatieve inverse bestaat niet voor alle getallen als we modulo 256 rekenen, dit is namelijk alleen zo als we modulo een priemgetal rekenen. We zouden ervoor kunnen kiezen om modulo 251 te rekenen (het hoogste priemgetal onder 256), maar dat brengt andere vervelende problemen met zich mee omdat de invoer uit 8 bits bestaat en dus 256 waarden kan hebben.

Eindig lichaam. In de wiskunde ligt een oplossing, we moeten dan gaan rekenen in het *eindig lichaam*. In AES wordt dit toegepast door een binair getal voor te stellen als een polynoom, waarbij de bits de coëfficiënten zijn. Bijvoorbeeld het getal 10111 wordt voorgesteld als het polynoom $1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$, vereenvoudigd wordt dit $x^4 + x^2 + x^1 + 1$. Omdat een byte acht bits heeft, zijn de polynomen altijd van graad 7. Als de graad bij een berekening zoals een vermenigvuldiging toch hoger wordt dan 7, kunnen we rekenen modulo een ander polynoom. Dit polynoom moet dan van graad 8 zijn zodat het resulterende polynoom van graad 7 wordt. Het polynoom moet ook *irreducibel* zijn, dat betekent dat het niet gevormd kan worden door twee andere polynomen van een lagere graad te vermenigvuldigen. Het is dan eigenlijk een priem-polynoom.

In dit eindige lichaam gaan wiskundige operaties niet zoals we gewend zijn. Het optellen is gelijk aan een xor (*exclusieve disjunctie*) operatie en omdat xor isomorf[1] is, is aftrekken ook gelijk aan een xor operatie. Vermenigvuldiging wordt gedaan door de polynomen met elkaar te vermenigvuldigen, modulo het irreducibele polynoom. Meer details en uitleg over de overige operaties in het eindige lichaam in AES kunnen gevonden worden in *The Design of Rijndael* [DR02, sec. 2.1]. Operaties zoals machtsverheffen en logaritmen kunnen voorberekend worden in tabellen zodat deze in het algoritme zeer snel gedaan kunnen worden[2]. Dit is mogelijk omdat we per byte rekenen, en een byte heeft maar 256 mogelijke getallen.

3.2.2 Encryptie

De invoer staat in een matrix van vier bij vier bytes, zoals weergegeven in Figuur 3.1. Deze matrix wordt per kolom van boven naar beneden gevuld. In elke cel staat precies 1 byte, en in de voorbeelden zullen we deze byte weergeven als 2 hexadecimale tekens. We gebruiken deze matrix om de meeste bewerkingen in het algoritme uit te leggen.

Er worden vier bewerkingen achtereenvolgens op de data uitgevoerd. Dat zijn substitutie, rijrotatie, kolomvermenigvuldiging en sleuteltoevoeging. Deze bewerkingen worden elke ronde uitgevoerd, behalve in de laatste ronde. Daarin wordt de kolomvermenigvuldiging overgeslagen, omdat deze het algoritme niet sterker maakt. Hij kan zonder kennis van de sleutel uit de laatste ronde gefilterd worden [DR02, sec. 10.2.3]. Vóór de eerste ronde wordt nog een sleuteltoevoeging gedaan met de hoofdsleutel, zodat de eerste ronde niet teruggerekend kan worden zonder kennis van de sleutel.

In de volgende paragrafen volgt een korte uitleg van de bewerkingen, voor meer informatie over de bewerkingen en de uitwerkingen verwijs ik je door naar het boek *The Design of Rijndael* [DR02] of de officiële standaard [NIS01].

Substitutie. Elke byte in de matrix wordt vervangen met een andere byte, zoals weergegeven in figuur 3.2. Welke byte dat is wordt opgezocht in een tabel, de S-box. Deze S-box is altijd hetzelfde, en de waarden erin zijn zo gekozen dat het verband niet lineair is. Als er niet

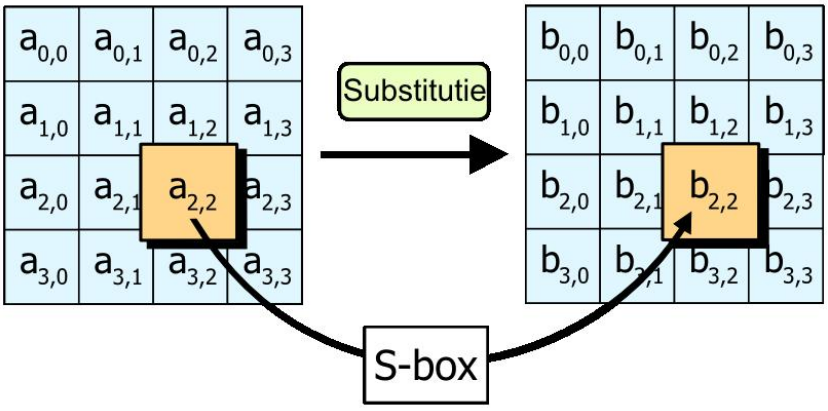
[1] Een operatie is isomorf als de inverse van de operatie gelijk is aan de operatie zelf.
[2] Zie voor een praktische implementatie: http://www.samiam.org/galois.html

00	04	08	0c
01	05	09	0d
02	06	0a	0e
03	07	0b	0f

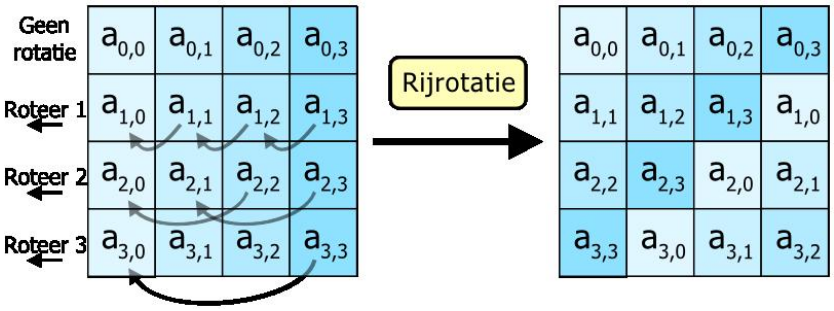
Figuur 3.1: AES BEREKEN-MATRIX

genoeg geheugen is voor deze tabel, kunnen de waarden ook opnieuw worden uitgerekend. Deze berekening vindt plaats in het eindige lichaam. De *multiplicatieve inverse* van de invoerbyte wordt genomen. Vervolgens wordt de binaire representatie van deze byte voorvermenigvuldigd met een matrix. Daarna wordt de xor operatie gedaan op het resultaat en de constante waarde 99 in binaire vorm. Door de matrixvermenigvuldiging wordt de algebraïsche representatie ingewikkelder en aanvallen op de S-box moeilijker.

Rijrotatie. De rijen worden geroteerd met een wisselend aantal posities, zoals weergegeven in Figuur 3.3. De bytes in de bovenste rij worden nul plaatsen verschoven. De bytes in de tweede rij worden een plaats naar links verschoven, hierbij komt de meest linker byte helemaal rechts te staan. De bytes in de derde rij worden twee plaatsen naar links verschoven waarbij de twee linker bytes roteren zodat ze rechts komen te staan. De bytes in de vierde rij worden



Figuur 3.2: AES SUBSTITUTIE

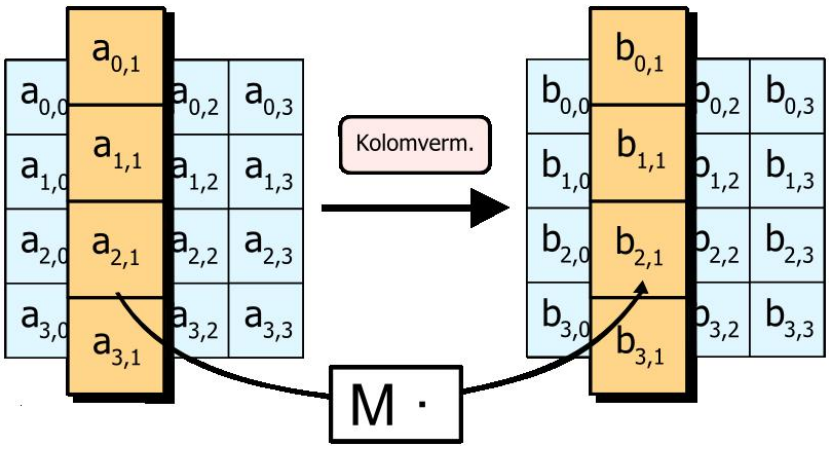


Figuur 3.3: AES Rij-rotatie

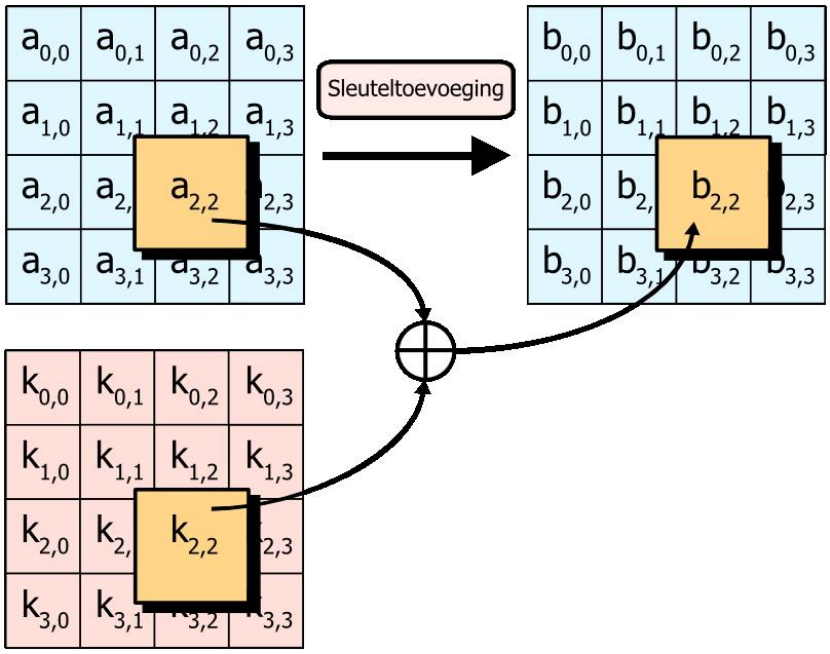
drie plaatsen naar links geroteerd of één plaats naar rechts, dit levert hetzelfde resultaat op. Merk hierbij op dat de rijen worden verschoven terwijl de bytes juist per kolom in de matrix worden gezet, zoals beschreven aan het begin van Paragraaf 3.2.2. Omdat de rijen geroteerd worden terwijl de bytes er per *kolom* ingezet worden, is de bewerking op byte-niveau minder eenvoudig dan het door figuur 3.3 doet voorkomen.

Kolomvermenigvuldiging. Elke kolom wordt voorvermenigvuldigd met een matrix, zoals weergegeven in Figuur 3.4. Dit is een constante matrix, en hij is inverteerbaar zodat bij ontsleuteling de inverse matrix gebruikt kan worden. De matrix is een *Maximum Distance Separable* matrix, afgekort een MDS-matrix. Dat betekent dat de matrix voor perfecte diffusie zorgt, de noodzaak hiervoor wordt besproken in Paragraaf 3.2.4.

Het optellen en de vermenigvuldigingen in matrixvermenigvuldiging worden gedaan volgens de speciale manier van het eindige lichaam, beschreven in Paragraaf 3.2.1. De matrix bestaat bij versleuteling alleen uit de getallen 1, 2 en 3. Hierdoor kan het vermenigvuldigen eenvoudig uitgevoerd worden. De inverse matrix die bij ontsleuteling gebruikt wordt, komt ook mooi uit en kan ook eenvoudig gebruikt worden.



Figuur 3.4: AES Kolomvermenigvuldiging



Figuur 3.5: AES Sleuteltoevoeging

Sleuteltoevoeging. Als laatste stap komt de sleutel aan bod. Vanwege deze stap heb je de sleutel nodig om de boodschap te ontsleutelen. We gebruiken elke ronde een andere sleutel. Deze rondesleutels worden afgeleid van de hoofdsleutel. Zie Paragraaf 3.2.3 voor de generatie van de rondesleutels. De huidige rondesleutel wordt met de xor-operatie op de data toegepast, zoals weergegeven in Figuur 3.5.

3.2.3 Sleutelgeneratie

De sleuteltoevoeging die nog vóór de eerste ronde gedaan wordt, wordt met de hoofdsleutel gedaan. Het is alsof de rondesleutel van ronde 0 gelijk is aan de hoofdsleutel (in het geval van een 128 bits sleutel). De volgende rondesleutel wordt gegenereerd uit de vorige rondesleutel, de rondesleutel voor ronde 1 wordt dus uit de rondesleutel van ronde 0 gegenereerd, en dat was de hoofdsleutel.

De sleutelgeneratie kan worden gezien alsof er een stroom aan sleutelbits gegenereerd wordt uit de hoofdsleutel. De rondesleutel is altijd 128 bits, dus elke 128 bits in die stroom vormen een rondesleutel. Bij de varianten met een 192 of 256 bits hoofdsleutel zitten er al 192 of 256 bits aan het begin van de stroom, dat is namelijk de hoofdsleutel. De eerste 128 bits daarvan worden gebruikt voor rondesleutel 0, en de rest wordt gebruikt voor de volgende rondesleutel. De sleutelbits die hierna gegenereerd worden in de stroom, worden per 4 bytes (32 bits) gemaakt. Deze 4 bytes hangen af van de vorige 4 bytes, en nog van 4 andere bytes eerder in de stroom. Hoe ver deze eerder in de stroom liggen, is gelijk aan de lengte van de hoofdsleutel.

Voorbeeld: Bij een 192 bits hoofdsleutel zijn de eerste 24 bytes gelijk aan de hoofdsleutel. We willen bytes 25-28 genereren. Dit gaat aan de hand van de vorige 4 bytes, dat zijn bytes 20-24. En aan de hand van de 4 bytes die eerder in de stroom liggen. Omdat de

hoofdsleutel 24 bytes lang is, liggen die bytes 24 plaatsen eerder. Dat zijn dus de bytes 1-4.

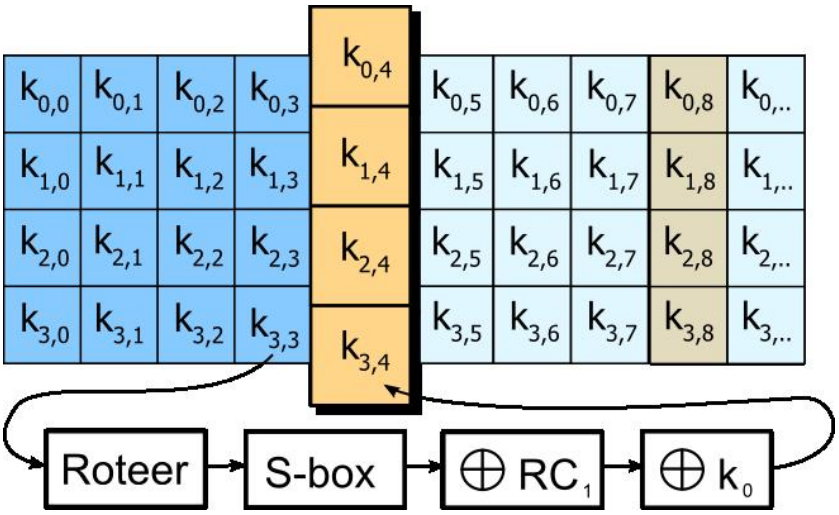
Het is belangrijk dat alle sleutelbits in de rondesleutels verwerkt worden.

Om 4 nieuwe bytes te genereren, kopiëren we de vorige 4 bytes en doen we de xor operatie erop met de 4 bytes eerder in de stroom. Dit doen we echter niet voor alle bytes in de sleutelstroom. Sommige bytes worden anders uitgerekend. Dat is om ervoor te zorgen dat een simpele hoofdsleutel, zoals één waarbij alle bits 0 zijn, toch rondesleutels oplevert die er willekeurig uitzien. Wanneer dit gebeurt is afhankelijk van de lengte van de hoofdsleutel. Als de hoofdsleutel 16 bytes (128 bits) lang is, gebeurt dit elke 16 bytes. Als de hoofdsleutel 24 bytes (192 bits) lang is, gebeurt dit elke 24 bytes. En als de hoofdsleutel 32 bytes (256 bits) lang is, gebeurt dit elke 32 bytes. Deze alternatieve berekening gaat als volgt:

1. Kopieer de vorige 4 bytes.
2. Roteer alle bytes 1 plaats naar links. De eerste byte komt dus op positie 4, en de andere bytes schuiven 1 plaats naar links.
3. Voer de substitutiestap uit op de vier bytes, zoals beschreven in Paragraaf 3.2.2.
4. Alleen op de eerste byte moet nu de xor operatie uitgevoerd worden met de *ronde constante*. De ronde constanten zijn altijd hetzelfde en kunnen dus van tevoren berekend worden. Deze constanten worden berekend door de formule: $RC(i) = x^{i-1}$ waarbij i het rondenummer is. De x komt van de polynoomrepresentatie van het eindige lichaam en vertegenwoordigt $1 \cdot x^1 + 0 \cdot x^0$, en dus het binaire getal 10, wat gelijk is aan het decimale getal 2. Er staat dus eigenlijk: $RC(i) = 2^{i-1}$, met als kanttekening dat het wel volgens de wiskunde in het eindige lichaam berekend wordt. Het rondenummer i wordt *niet* bij elke gemaakte rondesleutel opgehoogd. Hij wordt opgehoogd wanneer deze stap wordt uitgevoerd en dat hangt af van de lengte van de hoofdsleutel.
5. Zoals bij de gewone berekening, wordt nog de xor operatie uitgevoerd met de 4 bytes eerder in de stroom.

Om de sleutelgeneratie te verduidelijken is een voorbeeld gevisualiseerd in Figuur 3.6. Dit voorbeeld werkt met een 128 bits hoofdsleutel, het eenvoudigste geval. De bytes zijn per kolom gerangschikt, en er wordt naar rechts een stroom gegenereerd. De eerste 4 donkerblauwe kolommen worden gevuld met de hoofdsleutel en dit is tevens de rondesleutel voor ronde 0. Om de volgende 4 bytes te genereren (in kolom 4), kopiëren we eerst de vorige 4 bytes (in kolom 3). Deze roteren we één plaats naar links, in het figuur is dat één plaats naar boven. Byte $k_{1,4}$ komt zo bovenaan te staan en byte $k_{0,4}$ onderaan. Vervolgens vervangen we elke byte door een andere byte die we opzoeken in de S-box. Dan voeren we de xor operatie uit op de bovenste byte, samen met de ronde constante van ronde 1. Deze ronde constante is $RC(1) = 2^{1-1} = 2^0 = 1$. Omdat er bij de berekening geen getallen boven de 255 voorkomen, kunnen we de uitkomst eenvoudig bepalen. Als er wel getallen boven de 255 voorkomen, moeten we modulo het irreducibele polynoom werken zoals beschreven in Paragraaf 3.2.1. Als laatste moeten we op de 4 bytes nog de xor operatie uitvoeren met 4 bytes eerder in de stroom. Omdat we werken met een 128 bits (16 bytes) hoofdsleutel moeten we 16 bytes eerder kijken, dat zijn dus 4 kolommen eerder. We moeten de xor operatie dus uitvoeren met kolom 0.

De volgende drie kolommen worden op de eenvoudige manier berekend. Deze manier is gelijk aan de stappen 1 en 5 van de alternatieve berekening. Omdat we met een 128 bits (16



Figuur 3.6: AES Sleutelgeneratie

bytes) hoofdsleutel werken, wordt de alternatieve berekening elke 16 bytes gedaan. Dit wordt dus gedaan in de kolommen 4, 8, 12, etc.

Er is nog een uitzondering bij een hoofdsleutel van 256 bits. De alternatieve stap wordt daar elke 32 bytes uitgevoerd, dat is dus elke 2 rondesleutels. Op de totale 14 rondes zou deze stap dan maar 7 keer uitgevoerd worden. Om toch meer bewerkingen op de sleutel te doen, is ervoor gekozen om tussentijds nog een gereduceerde alternatieve bewerking te doen. Deze bestaat uit de stappen 1, 3 en 5 zoals die hierboven zijn beschreven. Deze gereduceerde alternatieve berekening wordt ook elke 32 bytes uitgevoerd, maar met een verschuiving van 16 bytes waardoor deze precies afwisselend met een volledige alternatieve berekening wordt gedaan. Bij een 256 bits hoofdsleutel wordt er dus elke 16 bytes een alternatieve berekening gedaan, en deze alternatieve berekening is afwisselend een volledige en een gereduceerde alternatieve berekening.

De figuren in dit hoofdstuk zijn overgenomen van WikiPedia[3].

3.2.4 Onderbouwing

Diffusie en Confusie. Een cryptografisch algoritme dat bestand is tegen onder andere statistische aanvallen heeft de eigenschappen *diffusie* en *confusie* [Sha46]. Diffusie wil zeggen dat er een sterke vermenging bestaat tussen de invoer tekst en de versleutelde tekst. Deze vermenging moet zo intens zijn dat als er één bit in de invoer verandert, dit effect heeft op de gehele versleutelde tekst. Dit effect moet bovendien volstrekt willekeurig lijken. Als er één bit in de invoer verandert, is bij elke bit in de versleutelde tekst de kans 50% dat hij verandert. Er is dan geen statistisch verband te ontdekken tussen de invoer en de versleutelde tekst.

Confusie lijkt op de diffusie-eigenschap, maar gaat over het verband tussen de *hoofdsleutel* en de versleutelde tekst. Daar moet een sterke vermenging bestaan waardoor elke bit van de hoofdsleutel effect heeft op elke bit in de versleutelde tekst. Als er één bit in de hoofdsleutel verandert, is bij elke bit in de versleutelde tekst de kans 50% dat hij verandert. Er is dan geen statistisch verband te vinden tussen de hoofdsleutel en de versleutelde tekst.

[3]Zie `http://en.wikipedia.org/wiki/Advanced_Encryption_Standard`.

Een perfecte diffusie en confusie is niet altijd haalbaar, maar dat is niet nodig. Als het de perfectie nadert en als dus de kansen die hierboven genoemd zijn bij de 50% in de buurt komen, zijn er zeer veel paren van bekende invoer en versleutelde tekst nodig om de statistische verbanden te kunnen gebruiken. Als het meer werk is om deze statistische verbanden te vinden dan om alle sleutels uit te proberen met een brute kracht aanval, dan heeft het toch geen zin om deze statistische verbanden te zoeken. Het is immers sneller om alle sleutels uit te proberen.

In AES is diffusie verwerkt door de stappen: substitutie, kolomvermenigvuldiging en rijrotatie. De S-box die in de substitutiestap wordt gebruikt is zó gekozen dat hij voor optimale diffusie zorgt binnen één byte. Als er in een byte één bit wordt veranderd, dan wordt de byte door de S-box in een andere byte veranderd. Deze andere byte heeft totaal geen verband met de invoer-byte, dat wil zeggen dat ongeveer de helft van de bits is veranderd. Om te zorgen dat het effect van een veranderende bit niet alleen binnen één byte effect heeft, is er de kolomvermenigvuldigingstap. Door de matrixvermenigvuldiging in die stap wordt het effect van een veranderde bit doorgevoerd naar de hele kolom, elke byte heeft namelijk invloed op elke andere byte in die berekening. Nu is het effect van één veranderende bit al zichtbaar in de hele kolom, maar nog niet in de andere kolommen. Dit wordt gedaan door de rijrotatiestap. De veranderde bytes komen zo in andere kolommen terecht, waarna ze met een kolomvermenigvuldiging de rest van de bytes in die kolom beïnvloeden. Nu zijn alle bytes beïnvloed door één veranderende invoer-bit. Omdat deze stappen een aantal keer herhaald worden, wordt het diffusie-effect steeds groter.

De confusie is op eenzelfde manier in AES verwerkt, maar deze vindt voornamelijk in de sleutelgeneratie plaats. Als elke bit van de hoofdsleutel effect heeft op elke bit van (latere) rondesleutels, dan beïnvloedt elke bit in de hoofdsleutel ook de gehele versleutelde tekst. Want de rondesleutels worden door de xor-operatie met de data wordt gecombineerd. De confusie in de sleutelgeneratie vindt plaats door de substitutie en rotatie. De substitutie heeft hetzelfde effect als bij diffusie. De rotatie verspreidt dit effect over de kolom. Omdat elke volgende kolom gegenereerd wordt door de vorige kolom te nemen en (na eventuele bewerkingen) de xor operatie erop uit te voeren met een eerdere kolom, wordt de diffusie altijd over de kolommen verspreid. De enige bewerking in de sleutelgeneratie die niet bijdraagt aan de confusie is de toevoeging van de ronde constante. Deze bewerking is er alleen om de symmetrie tegen te gaan.

De confusie gaat minder snel dan de diffusie. In twee tot drie rondes is er al volledige diffusie, maar het duurt minstens 4 ronden voor er volledige confusie is [DR02].

The Wide Trail Design Strategy. De makers van AES hebben een algemene methode bedacht om een robuust cryptografisch algoritme te maken. Hierbij is rekening gehouden met differentiële en lineaire cryptoanalyse, de meest populaire vormen van cryptoanalyse ten tijde van de ontwikkeling van AES. De door Joan Daemen en Vincent Rijmen ontwikkelde methode heet *The Wide Trail Design Strategy* [DR01] en is als basis gebruikt voor AES. De methode bestaat uit twee stappen:

1. Een lokale niet-lineaire transformatie. In AES is dit de substitutiestap.
2. Een niet-lokale lineaire transformatie. IN AES zijn dit de rijrotatie en kolomvermenigvuldiging stappen.

Samen zorgen deze stappen voor optimale diffusie, en het kan snel uitgevoerd worden.

Eenvoud. AES heeft een van de eenvoudigste algebraïsche structuren onder de blok encryptie algoritmes [FS03]. Dit is met opzet door de makers gedaan. Zo konden ze eenvoudiger

bewijzen dat sommige typen aanvallen onmogelijk uitgevoerd kunnen worden. Een ander belangrijk argument is dat een eenvoudige structuur ervoor zorgt dat meer mensen het algoritme gaan bestuderen, en dat zwakheden zo sneller gevonden worden. Maar omdat deze zwakheden tot nu toe nog niet gevonden zijn terwijl het algoritme wél veel is bestudeerd, geeft dit vertrouwen in de veiligheid van AES. Een ander voordeel is dat het makkelijker is om AES correct te gebruiken. Als een algoritme erg ingewikkeld is dan zijn de effecten van kleine veranderingen niet goed te overzien en is het waarschijnlijker dat daardoor zwakheden in de implementatie ontstaan.

Snelheid. Zoals bij de oproep voor kandidaten voor AES genoemd werd [BM97], is het erg belangrijk dat het algoritme snel uit te voeren is in zowel software als hardware. Bij de ontwikkeling van AES is daar vanaf het begin rekening mee gehouden. De bewerkingen zijn zo gekozen dat de implementatie snel is, zelfs ondanks alle ingewikkelde wiskunde. Zo is de matrix die in de kolomvermenigvuldigingstap gebruikt wordt zó gekozen dat de bewerkingen in het eindige lichaam zeer snel uit te voeren zijn. En de inverse van die matrix komt goed uit zodat de berekeningen bij ontsleuteling ook snel uit te voeren zijn. Daarnaast is AES zeer toegankelijk voor chipkaarten, ook wel smart cards genoemd. Deze hebben een 8-bit processor. AES kan hier goed op geïmplementeerd worden omdat elke bewerking op een byte uitgevoerd wordt. Als AES op 32 bits waarden zou werken, zou een implementatie op een chipkaart erg veel gekunstel vergen en erg traag worden. In *The Design of Rijndael* [DR02] worden de snelheden van een aantal implementaties genoemd. Een 8-bit processor die op 4 MHz loopt kan 6750 bytes per seconde versleutelen. Op een Pentium III processor op 800 MHz kunnen ongeveer 50 Megabytes per seconde worden versleuteld. Dat is veel meer dan een gewone harde schijf weg kan schrijven!

3.3 Gebruik

3.3.1 Gebruiksmodus

Als AES in een applicatie gebruikt wordt, wil dat niet zeggen dat de informatie daadwerkelijk veilig is. Het is erg van belang welke gebruiksmodus gebruikt wordt. Een gebruiksmodus is de manier waarop AES wordt ingezet. De eenvoudigste manier is door de data op te splitsen in blokken van 128 bits en ieder blok apart te versleutelen met de sleutel. Het probleem hierbij is echter dat als er meerdere blokken met dezelfde inhoud zijn, de versleutelde blokken ook hetzelfde zijn. Op deze manier is er toch verband te zien in de versleutelde data, terwijl dat juist niet de bedoeling is. Een betere manier van versleuteling is door eerst de xor operatie uit te voeren op de invoer samen met het vorige versleutelde blok, voordat de data in het AES-algoritme gaat. Op deze manier hangt elk blok van het vorige blok af, en kan je geen verbanden meer tussen blokken onderling vinden. Omdat er bij het eerste blok geen vorig versleuteld blok is, wordt daar een *initialisatie vector* gebruikt. Daarvoor wordt vaak willekeurige data gekozen zodat als je twee keer hetzelfde bestand versleutelt, de uitkomst toch verschillend is. Het nadeel van de beschreven methode is dat de versleuteling van blokken niet parallel uitgevoerd kan worden. Er zijn nog veel andere gebruiksmodi met ieder hun eigen voor en nadelen. Het is dus erg belangrijk om een goede modus te kiezen. Het National Institute of Standards and Technology (NIST) heeft een aantal gebruiksmodi vastgesteld waarmee AES gebruikt mag worden [Dwo01].

3.3.2 Adoptie

AES is in 2001 officieel een standaard geworden. In juni 2003 kondigde de overheid van de Verenigde Staten dat ze de standaard gingen gebruiken ter bescherming van hun geheime data [CNS03]:

> The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use.

Dit betekent dat er vertrouwen is in de standaard door niet de minste partij, namelijk de Verenigde Staten. Verder wordt AES op vele andere gebieden gebruikt, zoals in het draadloze encryptie protocol WPA2 en het beveiligde ip protocol IPsec. Ook in archiveringssoftware zoals WinZip, en schijfencryptiesoftware zoals BitLocker wordt AES gebruikt. De nieuwste Intel processoren, en toekomstige AMD processoren ondersteunen zelfs speciale instructies om het AES versleutelproces te versnellen [Gue08].

De Advanced Encryption Standard wordt dus gebruikt in veel belangrijke nieuwe omgevingen, en met de introductie van speciale instructies in processoren wordt dit proces verder versneld. De introductie van de nieuwe standaard is dus een succes. Er zijn nog steeds een hoop oude systemen die Triple DES gebruiken. Er is nog geen noodzaak om deze systemen naar AES om te bouwen want voorlopig is Triple DES nog veilig genoeg [BBB+07]. Maar in nieuwe systemen zal AES gebruikt worden omdat deze veiliger en sneller is.

3.4 Kraken

In de cryptografie wordt een algoritme als gekraakt beschouwd als er een manier is om sneller achter de originele data te komen dan door alle sleutels uit te proberen. Het uitproberen van alle sleutels wordt een brute kracht aanval genoemd. Deze is afhankelijk van het aantal mogelijke sleutels. Bij de 128 bits variant van AES zijn er 2^{128} mogelijke sleutels. Dit zijn er zo erg veel dat dit niet haalbaar is voor het eind van deze eeuw [DRB02]. De vraag is of er ook een kortere route gevonden kan worden. Er zijn veel verschillende manieren om deze te zoeken, en zo het algoritme te breken. De belangrijkste zijn differentiële en lineaire cryptoanalyse. Zoals ook in Paragraaf 3.2.4 is uitgelegd, is AES ontworpen met differentiële en lineaire cryptoanalyse in het achterhoofd. De makers hebben zelfs bewezen dat deze aanvallen onmogelijk zijn op AES [DR02]. Er zijn sinds die tijd echter een hoop nieuwe soorten aanvallen ontdekt, is AES daar ook tegen bestand?

3.4.1 Integrale cryptoanalyse

In het AES algoritme wordt veelvuldig gebruik gemaakt van de xor operatie. Een eigenschap van de xor operatie is dat $a \oplus b \oplus a = b$, en deze eigenschap wordt bij integrale cryptoanalyse gebruikt. Het idee achter deze aanvalsmethode is om verschillende invoeren te construeren zodat, als de xor operatie op alle invoeren wordt uitgevoerd, het resultaat gelijk is aan nul. Als op de versleutelde versies van deze invoeren de xor operatie wordt uitgevoerd, dan is de invoerdata als het ware eruit gefilterd. Wat overblijft is de door elkaar geklutste sleutel.

Deze aanval was al voor de constructie van het AES algoritme bekend. Uit een AES versie met 4 ronden kan hiermee de sleutel herleid worden, en door de twee opvolgende rondes te gokken kan zo een versie van AES met zes ronden gekraakt worden. Het heeft geen zin om de overige ronden te gokken, want dan zijn er zo veel mogelijkheden dat het sneller is om een brute kracht aanval te doen. AES maakt dus gebruik van meer dan zes ronden, zodat deze aanval niet meer mogelijk is [DR02].

3.4.2 Algebraïsche cryptoanalyse

Lange tijd werd er gedacht dat deze nieuwe vorm van cryptoanalyse AES zou breken. De meeste hoop was gevestigd op de *XSL-attack*. Het idee hierachter is om misbruik te maken van de relatief eenvoudige structuur van AES. De bewerkingen die AES uitvoert worden in een stelsel van vergelijkingen genoteerd, waarin de sleutel de onbekende factor is. Door deze vergelijkingen op te lossen, zou de sleutel er zo uit rollen. De werkelijke uitvoering hiervan bleek praktisch echter niet haalbaar te zijn. Het oplossen van een stelsel vergelijkingen zoals deze is NP-moeilijk, en vormt een grotere uitdaging dan simpelweg alle mogelijke sleutels uit te proberen.[DRB02]

3.4.3 Nieuwe aanvallen

In 2009 is de aandacht gevestigd op de sleutelgeneratie van AES. In mei is een kraak gevonden van AES-256 waarbij één op de 2^{35} sleutels gekraakt kan worden in een tijdscomplexiteit van 2^{120}, daarbij zijn er ook 2^{120} versleutelde teksten nodig van gekozen invoerdata [BKN09]. Dit betekent dat AES theoretisch is gekraakt. De aanval is echter extreem onpraktisch en er wordt verder gezocht naar praktische aanvallen, vooral naar aanvallen die op elke sleutel werken. Er zijn een paar aanvallen die een deel van de rondes van AES kunnen kraken. Zo kunnen bij de 256 bits variant al 10 van de 14 ronden gekraakt worden [BDK$^+$09], en bij de 128 bits variant al 8 van de 10 [GP09]. Het lijkt door deze aanvallen alsof AES bijna op een praktische manier te kraken is, maar dat valt wel mee. De latere rondes zijn namelijk exponentieel moeilijker te kraken. Bovendien zal de aanval nooit écht praktisch zijn, omdat er voor de recente typen aanvallen erg veel bekende versleutelde teksten nodig zijn van bekende invoerdata. De recente aanvallen geven wel aan dat het AES algoritme wankelt, en dat er bepaalde zwakheden in zitten.

3.4.4 Side-channel aanvallen

In plaats van het aanvallen van AES zelf, kunnen we ook specifieke implementaties van AES aanvallen. Dit wordt een *side-channel aanval* genoemd. We richten ons hierbij niet op de gebruiksmodus, maar op andere factoren die informatie geven over het systeem. Als je bij de hardware van het encryptiesysteem kunt komen, kun je bijvoorbeeld meten hoe lang er over een AES encryptie wordt gedaan. Uiteindelijk kan je zo de sleutel afleiden. Een andere benadering is door te meten hoeveel energie de hardware verbruikt bij encryptie. Je kunt je voorstellen dat het veranderen van 3 bits meer energie kost dan het veranderen van 4 bits, en zo zou je ook de sleutel kunnen afleiden. Een andere aanpak is door doelbewust de hardware in de fout te laten gaan door bijvoorbeeld oververhitting. Zo kan er een verkeerde berekening gedaan worden waardoor er onbedoeld kritieke informatie naar buiten lekt. Hiermee is al een succesvolle aanval gedaan [SMR09]. Deze aanvallen zijn natuurlijk allerminst praktisch. Je moet immers bij het versleutelen aanwezig zijn. Maar je zou een situatie kunnen bedenken

waarin het mogelijk is. Als de encryptie bijvoorbeeld in een chipkaart wordt gedaan, zou dit wel mogelijk zijn.

Tegen de simpelste vormen van deze aanvallen is AES te wapenen [DR02]. Als je ervoor zorgt dat de encryptie altijd evenveel tijd in beslag neemt door no-operation commando's op bepaalde plaatsen toe te voegen, dan is de aanval waarbij de tijd wordt gemeten kansloos. Het algoritme is moeilijker te wapenen tegen het meten van energieverbruik. Een mogelijkheid hier is om hardwarematig componenten toe te voegen die het energieverbruik zo veel mogelijk spreiden over de tijd.

In plaats van de hardware kunnen we ook side-channel aanvallen op software implementaties uitvoeren. In de implementatie van WinZip wordt de data met AES versleuteld maar de data die WinZip zelf eraan toevoegt, de *metadata*, wordt niet versleuteld. Deze metadata geeft echter zoveel hints dat hiermee de sleutel binnen redelijke tijd achterhaald kan worden [Koh04].

Samenvatting en conclusies

We hebben gezien dat AES een belangrijk encryptie algoritme is. Het heeft een zware selectieprocedure moeten doorstaan, en alleen daardoor al verdient het al een hoop vertrouwen. Over het ontwerp zelf is goed nagedacht door de makers. Alle bestaande aanvallen konden AES niet kraken en daardoor is men op zoek gegaan naar nieuwe aanvallen. In 2009 zijn een hoop nieuwe soorten aanvallen bedacht waarmee AES werd geprobeerd te kraken. Er kan dus gesteld worden dat AES de ontwikkeling van de cryptografie heeft gestimuleerd.

De recente aanvallen op AES lijken erg onheilspellend voor de toekomst van AES. Het lijkt echter onwaarschijnlijk dat het daadwerkelijk tot een praktische kraak zal komen. We hebben gezien dat AES met een 256 bits hoofdsleutel al gekraakt is, maar deze aanval is extreem onpraktisch. De aanvallen zullen misschien iets praktischer worden. Het ligt voor de hand dat, áls er een grote zwakte in AES zou liggen, deze al tijdens de zware selectieprocedure gevonden zou zijn. Dit is echter niet gebeurd dus is het onwaarschijnlijk dat er een grote zwakte in AES bestaat. Desondanks doen de nieuwe aanvallen het vertrouwen in AES wankelen, want het betekent dat er kennelijk toch een kleine zwakheid in AES zit.

Meer reden tot zorgen zijn de side-channel aanvallen. Deze relatief nieuwe aanvallen kunnen specifieke implementaties van AES al kraken. Omdat deze soort aanvallen nog zo jong is, wordt er bij de implementatie niet altijd aan gedacht. Vooral als AES meer toegepast gaat worden op chipkaarten is het reden tot zorgen. Een chipkaart is namelijk erg toegankelijk om side-channel aanvallen op uit te voeren omdat je bijna direct bij de chip kan komen. Mijn verwachting is dat er de komende jaren meer aandacht besteed zal worden aan het verdedigen van het AES algoritme tegen dit soort aanvallen.

Opgaven bij hoofdstuk 3

Opgave 3.1 *Zie* `http://www.wired.com/threatlevel/2012/03/ff_nsadatacenter/all/1`. *In Amerika wordt gebouwd aan een reuzencomputer waarvan de rekencapaciteit zich laat uitdrukken in Petaflops, Exaflops en het nieuw uit te vinden woord Yottaflops.*

(a) Als de rekenkracht van de computers elke anderhalf jaar verdubbelt, wanneer kan de Yottaflop-machine dan klaar zijn?

(b) Hoe lang rekent de Yottaflop-machine aan een brutekrachtaanval op 128-bits AES (neem aan dat een decryptie een enkele machine-instructie kost!)?
(c) Wanneer zal de machine klaar zijn die 256-bits AES in een jaar kan aanvallen?

Hoofdstuk 4

Symmetrische encryptie: vervolg

In dit hoofdstuk bekijken we verdere toepassingen van en aanvallen op symmetrische encryptie. De beschreven methoden, zoals meervoudige encryptie, en aanvallen, zoals Hellmans *cubic-rootaanval*, zijn algemeen toepasbaar, maar om de bespreking niet te abstract te maken, gaan we bij de beschrijving vaak uit van toepassing op DES.

Ook behandelen we de versleuteling die in gsm-telefonie wordt toegepast, inclusief de manier waarop de sleutels worden gekozen, en de sleutelbepaling in de chipknip. Ten slotte wordt een beschrijving gegeven van Rijndael, de nieuwe standaard voor symmetrische encryptie.

4.1 Hellmans cubic-rootaanval

De aanval die in deze sectie wordt beschreven, vermindert de hoeveelheid werk voor een brutekrachtaanval niet, maar het grootste deel van het werk wordt gedaan voordat de aanval daadwerkelijk wordt uitgevoerd, als *precomputation* dus. De aanval werd beschreven door Hellman [Hel80].

Om het belang hiervan in te zien, moet men zich realiseren dat veel cryptografische sleutels slechts kort worden gebruikt, en dat ze ook slechts kort voor een aanvaller interessant zijn. Denk als voorbeeld aan een geldautomaat: bij het begin van een transactie wordt een (bijvoorbeeld: DES-) sleutel door bank en automaat gekozen en de communicatie begint. Voor criminelen is het interessant deze sleutel te pakken te krijgen *terwijl de transactie nog gaande is*, omdat in dat geval de communicatie kan worden beïnvloed. Een aanval die de sleutel na vijftien dagen, of vijftien minuten oplevert is onbruikbaar omdat de transactie al is afgelopen en de sleutel weggegooid.

Zoals gezegd is het met Hellmans aanval mogelijk het meeste werk te doen als precomputation: aan het opbouwen van de tabellen kan desnoods weken worden gerekend (de precomputation), zodra dan wat data is afgetapt, kan de sleutel in enige minuten of seconden worden gevonden.

Bij het aanmaken van de tabellen wordt gerekend met een van tevoren gekozen klare tekst x; deze hoeft aan geen enkele speciale eigenschap te voldoen. Wel moet aanvaller Oscar na het berekenen van de tabellen zorgen dat Alice en Bob deze boodschap gaan versleutelen; we spreken van een *gekozen-klaretekstaanval* of *Chosen Plaintext Attack*. Oscar kan ook de x kiezen waarvan hij weet of vermoedt dat Alice die ooit aan Bob zal zenden.

4.1.1 Encryptie als one-way functie

De encryptie beeldt een sleutel en klare tekst af op een cijfertekst; een wiskundige stelt dit voor als functie van twee parameters:

$$E : P \times K \longrightarrow C.$$

Omdat Alice en Bob een vaste sleutel k gebruiken, is de tweede parameter voor hen niet interessant, ze zien encryptie als manier om klare tekst naar cijfertekst te vertalen en terug:

$$E_k : P \longrightarrow C.$$

Oscar vindt de sleutel natuurlijk *wel* interessant! Hij beschouwt echter slechts één vaste klare tekst x en ziet daarom encryptie als een functie die de (hem onbekende) sleutel afbeeldt op een cijfertekst:

$$E_x : K \longrightarrow C.$$

Deze functie E_x is voor Oscar te beschouwen als een *one-wayfunctie*, waarmee we bedoelen dat Oscar wel de functie kan berekenen (voor elke k) maar niet de inverse. Immers, gegeven $y = E_x(k)$ kan Oscar k niet gemakkelijk vinden. We kunnen Hellmans cubic-rootaanval zien als een algemene methode om het terugrekenen van one-way functies grotendeels als voorberekening (precomputation) uit te kunnen voeren, dit wil zeggen dat het meeste rekenwerk wordt gedaan voordat y bekend is. Zij verder N de grootte van de verzameling waaruit de gezochte k moet komen ($N = 2^{56}$ in het geval van DES).

We bekijken eerst twee extreme strategieën voor het vinden van k. Methode 1 is *tijdsintensief*: na ontvangst van y probeert Oscar alle N mogelijkheden tot de juiste k is gevonden. Dit is natuurlijk niets anders dan de ons al bekende brutekrachtaanval die ruwweg N tijd kost. Methode 2 is *geheugenintensief*: *voordat* de aanval begint rekent Oscar voor alle N waarden van k het beeld $E_x(k)$ uit en slaat die op in een tabel. Na ontvangst van y raadpleegt Oscar zijn tabel en vindt (in $O(1)$ tijd!) de bijbehorende sleutel.

Dat is snel! Helaas voor Oscar is de aanval niet uitvoerbaar omdat er een onrealistisch grote hoeveelheid schijfruimte voor nodig is. Het opslaan van de N beelden (elk acht byte) bij DES neemt ongeveer 5×10^{17} byte. Laat Oscar disks van 50GB kopen voor 200 euro per stuk met een stroomverbruik van 10W. Hij heeft tien miljoen disks nodig, die twee miljard euro kosten en 100MW electriciteit verbruiken.

Voor beide methoden is het ook mogelijk een kleinere hoeveelheid tijd of geheugen te gebruiken, maar de slagingskans van de aanval wordt dan kleiner. Als een aanval S verschillende sleutels bekijkt is de slagingskans S/N. Als de aanval S *willekeurig gekozen* sleutels bekijkt is de slagingskans iets kleiner dan S/N omdat mogelijk dezelfde sleutel meerdere keren wordt bekeken.

4.1.2 Organisatie van een enkele poging

De kern van Hellmans aanval is een methode om S sleutels te bekijken, niet in tijd S zoals in methode 1 of met geheugen S zoals in methode 2, maar met een tijdgeheugen-*product* S. Zij M de hoeveelheid geheugen die we willen inzetten voor een poging en T de hoeveelheid tijd (na ontvangst van y).

Voor het inverteren van een one-way functie f wordt een tabel gebruikt waarin zeer veel waarden van f zijn opgeslagen in weinig geheugen. Om dit te bereiken wordt f geïtereerd, dus herhaald toegepast op een willekeurige beginwaarde. In een tabel worden de beginwaarde en de eindwaarde opgeslagen, dus een X met daarbij $X' = f^T(X)$. Of een gegeven y onder

de tussenliggende waarden zit, is te controleren door de functie f op y te itereren en te kijken of X' wordt gevonden; deze zoekmethode kost natuurlijk T tijd. De effectiviteit van de zoekactie wordt vergroot door voor vele waarden van X (namelijk, zoveel als men geheugen wil investeren) een paar $(X,\ f^T(X))$ op te slaan.

Aanmaken van de tabel: reductiefunctie. Voor het gebruik van de tabel moet de doelfunctie E_x, zoals boven uitgelegd, worden geïtereerd en bij DES levert dit het bezwaar dat argument en waarde niet even lang zijn (namelijk 56 respectievelijk 64 bits). We gebruiken $\mathbb{B}$ voor de verzameling van bitwaarden en zeggen dan dat het argument (sleutel) in $\mathbb{B}^{56}$ zit en de waarde (cijfertekst) in $\mathbb{B}^{64}$. Dit verschil wordt overbrugd door het toepassen van een *reductiefunctie* R. De reductiefunctie is een functie van $\mathbb{B}^{64}$ naar $\mathbb{B}^{56}$, het makkelijkst is simpelweg 56 van de 64 bits in een of andere volgorde te selecteren, en er eventueel een aantal te inverteren. De combinatie $f = (R \circ E_x)$ is een functie van $\mathbb{B}^{56}$ naar $\mathbb{B}^{56}$.

De tabel wordt aangemaakt door te beginnen met M random strings $X_{10}, \ldots, X_{M0}$ uit $\mathbb{B}^{56}$. Op elk van deze strings wordt de gecombineerde functie f geïtereerd, en wel T maal, dus van X_{i0} wordt een keten afgeleid:

$$X_{i0},\ X_{i1} = f(X_{i0}),\ X_{i2} = f(X_{i1}), \ldots,\ , X_{iT}.$$

De strings X_{ij} voor $j < T$, waarop f is toegepast, noemen we de basis van de tabel en deze bestaat uit $M \cdot T$ strings (die echter niet allemaal verschillend hoeven te zijn). Van elke keten wordt het laatste element X_{iT} opgeslagen, samen met de beginwaarde X_{i0}. De tabel beslaat dus M records van elk twee bitstrings.

Het zoeken in de tabel. Zij nu gegeven een $y = E_x(k)$, dus de encryptie van x met een onbekende sleutel k; Oscar past de reductiefunctie R toe en vindt $y' = R(y) = f(k)$. Vervolgens kan in $O(T)$ tijd worden bepaald of k in de basis van de tabel zit, namelijk door f te itereren op y' en steeds het resultaat te zoeken in de tabel. Als $k = X_{ij}$ (met $j < T$), dan is $y' = X_{i,j+1}$, en $f^{T-j-1}(y') = X_{iT}$, dus na $T - j - 1$ iteraties van f wordt het resultaat in de tabel gevonden als X_{iT}. Een kandidaat voor k wordt dan gevonden als $f^j(X_{i0})$.

Merk op dat wel geldt

$$k = X_{ij} \Rightarrow f^{T-j-1}(y') = X_{iT},$$

maar niet de omgekeerde implicatie. Het is dus mogelijk dat de gevonden waarde $f^j(X_{i0})$ *niet* de gezochte sleutel is en daarom moet de gevonden waarde nog met een stuk van de cijfertekst worden getest.

Analyse. Schrijf S voor $M \cdot T$; het opzetten van de tabel kost $O(S)$ werk, het opslaan kost $O(M)$ geheugen, en een zoekactie kost $O(T)$ tijd. Nu mag S niet te groot zijn omdat er dan veel dubbele waarden in de tabel voorkomen waardoor het aantal geteste sleutels te sterk onder S daalt. Hellman bewees dat, wanneer $M \cdot T^2 \leq N$, het verwachte aantal voorkomende sleutels ongeveer $0.8S$ is, en de slagingskans van de aanval ongeveer $0.8S/N$.

4.1.3 Parallelle pogingen

De slagingskans wordt vergroot door een aantal onafhankelijke pogingen op te zetten: omdat de reductiefunctie een willekeurige afbeelding van 64-bits- naar 56-bitsstrings is, kan hiermee worden gevariëerd. Oscar kan A verschillende reductiefuncties kiezen en A onafhankelijke

tabellen opzetten. De slagingskans van de totale aanval is dan (volgens stelling A.3)

$$1 - (1 - \frac{0.8M \cdot T}{N})^A$$

Voor een groot getal x is $(1 - \frac{1}{x})^x \approx \frac{1}{e}$, en daarom is dit ongeveer $1 - \frac{1}{e^a}$ met $a = \frac{0.8A \cdot M \cdot T}{N}$.

Een typische keuze, waaraan de methode zijn naam ontleent, is om M, T en A ongeveer $N^{1/3}$ te kiezen. Het voorwerk (precomputation) is dan $O(N)$ werk, de totale geheugenruimte $A.M$ is $O(N^{2/3})$ en de rekentijd voor een aanval (na ontvangst van y) is $A \cdot T = O(N^{2/3})$. De slagingskans is $1 - \frac{1}{e^{0.8}} \approx 0.55$.

In het geval van DES zijn M, T en A alle ongeveer 10^6. Een tabel bevat een miljoen records van zestien byte elk, beslaat dus 16MB en past in het werkgeheugen van een pc. De tabellen samen beslaan 16000GB; ze zullen na het opzetten worden opgeslagen op een array van optische disks. Het doorzoeken van een tabel itereert f een miljoen maal, dit kan gemakkelijk binnen een seconde gebeuren. Met een cluster van duizend pc's is de aanval in een kwartier voltooid. Op speciale hardware, zoals de EFF DES-kraker (sectie 2.2.2) is het een kwestie van seconden.

4.2 Productsystemen

In deze sectie bespreken we de mogelijkheden, de veiligheid van een cryptografisch algoritme te vergroten door toepassing van *meervoudige encryptie*. Een boodschap kan meermalen met hetzelfde algoritme worden vercijferd, of achtereenvolgens met verschillende systemen. Het eerste heeft als doel de sleutellengte te vergroten, het tweede kan worden gebruikt als Alice en Bob hun vertrouwen niet op een enkel systeem willen stellen. Wanneer meerdere keren achter elkaar wordt versleuteld spreken we van *productversleuteling*.

Stellen wij ons weer een cryptosysteem voor als een met een sleutel geparametriseerde klasse functies: $S = \{E_k : k \in K\}$. We bekijken eerst de situatie waarin verschillende systemen $S_1 = \{E_{1,k} : k \in K_1\}$ en $S_2 = \{E_{2,k} : k \in K_2\}$ worden gebruikt. De complete sleutel bestaat uit een paar (k_1, k_2) (waar $k_1 \in K_1$ en $k_2 \in K_2$) en de encryptiefunctie E_k is $(E_{2,k_2} \circ E_{1,k_1})$. De sleutellengte (en dus ook de kritieke lengte) is de som van de sleutellengten (respectievelijk kritieke lengten) van de twee afzonderlijke systemen. Zoals uit de beschouwing in hoofdstuk 1 blijkt, zal de kritieke lengte doorgaans nog steeds onpraktisch klein zijn. Een toename van het werk van een brutekrachtaanval is er evenmin zoals we zullen zien in sectie 4.2.2. Deze vorm van gecombineerde encryptie wordt gebruikt wanneer men op geen van de twee systemen afzonderlijk wil vertrouwen. Op zijn best is het een zwak alternatief voor het beschikken over een enkel systeem van voldoende betrouwbaarheid.

Meervoudige encryptie met hetzelfde algoritme wordt vaak toegepast en het enige doel is van een op zich betrouwbaar geacht cryptosysteem de sleutellengte te vergroten.

4.2.1 Valkuilen: idempotentie

Het kiezen van meerdere sleutels leidt er niet automatisch toe dat de klasse van encryptiefuncties ook groter wordt. Laten we aannemen dat Alice en Bob twee verschuivingen (zie sectie 1.2.1) gebruiken, eerst over k_1 en dan over k_2 posities. Helaas voor Alice en Bob is deze gecombineerde bewerking dezelfde functie als het slechts eenmaal verschuiven over $k_1 + k_2$ posities, kortom, de verkregen klasse functies is dezelfde als bij een enkele verschuiving:

$$\{E_{k_2} \circ E_{k_1} : k_1, k_2 \in K\} = \{E_k : k \in K\}.$$

Heeft een cryptosysteem deze eigenschap, dan noemen we het *idempotent*.

Je kunt eenvoudig nagaan dat niet alleen het schuifsysteem, maar ook de andere systemen uit hoofdstuk 1 (affiene, substitutie-, en permutatie- en Hill- bij gelijke blokgrootte) idempotent zijn. Voor deze systemen heeft meervoudige toepassing dus geen zin.

We kunnen het DES-algoritme (en andere Feistel-gebaseerde systemen) als productsysteem beschouwen, omdat de data in achtereenvolgende ronden wordt bewerkt. Van de DES-iteratie is echter bekend dat deze *niet* idempotent is, en dat het itereren leidt tot een echt grotere klasse van functies [Wer92]. Ook is bekend dat het DES-algoritme als geheel niet idempotent is en dat tweevoudige toepassing een klasse van bijna $2^{2 \cdot 56}$ functies oplevert [CW92].

4.2.2 Meet-in-the-middleaanval

Men zou verwachten dat het breken van een productsysteem met brute kracht flink langer duurt door de toegenomen sleutellengte: de aanvaller moet immers alle combinaties van twee sleutels uitproberen. Diffie en Hellman [DH77a] lieten zien dat dit *niet* het geval is, en daardoor, dat productversleuteling een bedenkelijk alternatief is voor het hebben van een goed cryptosysteem met voldoende lange sleutels.

Is Y het resultaat van een productversleuteling van X, ofwel $Y = E_{k_2}(E_{k_1}(X))$, dan is er een tussenresultaat $Z = E_{k_1}(X) = D_{k_2}(Y)$. De *meet-in-the-middleaanval* gebruikt deze tussenwaarde Z om mogelijkheden voor k_1 en k_2 te elimineren.

Notaties. De meet-in-the-middleaanval is een klaretekstaanval: de aanvaller moet beschikken over enkele paren (X_1, Y_1), (X_2, Y_2) et cetera van bijbehorende klare- en cijfertekstblokken. Voor elk paar is Y berekend als $E_{k_2}(E_{k_1}(X))$.

Oscar legt twee lijsten L_1 en L_2 aan, waarin alle mogelijkheden voor k_1 en k_2 zijn opgenomen. Initiëel is $L_1 = K_1$ en $L_2 = K_2$, de lengte van de lijsten is dus $|K_1|$, respectievelijk $|K_2|$. De lijsten bevatten initieel de gezochte sleutels k_1 en k_2, en deze eigenschap blijft behouden.

Sleutel-eliminatie. De eliminatie gebruikt een paar (X, Y) door X te versleutelen met elke sleutel uit lijst L_1 en Y te ontsleutelen met elke sleutel uit de lijst L_2: vorm lijsten

$$M_1 = \{E_k(X) : k \in L_1\} \qquad \text{en} \qquad M_2 = \{D_k(Y) : k \in L_2\}.$$

Omdat de tussenwaarde Z voldoet aan $Z = E_{k_1}(X)$ en $Z = D_{k_2}(Y)$ komt Z in beide lijsten voor: $E_{k_1}(X) \in M_2$ en $D_{k_2}(Y) \in M_1$. De eliminatieronde bestaat hieruit, dat uit L_1 alle k worden verwijderd waarvoor $E_k(X) \notin M_2$ en uit L_2 alle k worden verwijderd waarvoor $D_k(Y) \notin M_1$.

De juiste sleutels worden niet verwijderd, en er blijven mogelijk vele onjuiste sleutels staan in L_1 en L_2.

Analyse voor 2DES. De rekentijd voor een ronde van deze aanval is $O(|L_1| + |L_2|)$, voor de eerste ronde dus $|K_1| + |K_2|$, wat heel gunstig is in vergelijking met $|K_1| \times |K_2|$ voor een brutekrachtaanval.

De eerste ronde kost $2 \cdot 2^{56}$ DES-berekeningen. Omdat lijst M_2 'slechts' 2^{56} van de 2^{64} strings uit $\mathbb{B}^{64}$ bevat, heeft een $k \neq k_1$ in L_1 een kans van slechts ongeveer $\frac{2^{56}}{2^{64}}$ dat $E_k(X_1)$ voorkomt in M_2. We verwachten daarom na één ronde nog een $\frac{1}{2^8}$ fractie van de sleutels, dus 2^{48} stuks, over te hebben in lijst L_1. Vanwege de symmetrie in lijst L_2 ook.

De tweede ronde begint met twee lijsten van 2^{48} sleutels. De lijsten M_1 en M_2 bevatten dus 2^{48} van de 2^{64} strings, een sleutel $k \neq k_1$ in L_1 heeft dus een kans van slechts $\frac{1}{2^{16}}$ dat

$E_k(X_2)$ voorkomt in M_2. Na de tweede ronde verwachten we daarom nog maar $2^{48} \cdot \frac{1}{2^{16}} = 2^{32}$ sleutels in elke lijst.

Dat lijkt nog veel, maar de derde ronde maakt het werk af: M_1 en M_2 zijn nog maar 2^{32} lang, een onjuiste sleutel in L_1 of L_2 heeft dus nog maar een kans van $\frac{2^{32}}{2^{64}} = \frac{1}{2^{32}}$ om door te komen. Naar verwachting bevatten L_1 en L_2 na de derde ronde dus nog maar één onjuiste sleutel elk.

We zien dat het rekenwerk zo snel van ronde tot ronde afneemt, dat het werk van de totale aanval geheel door de eerste ronde wordt gedomineerd. De eliminatieronden zijn vrij effectief zodra de lengte van de lijsten korter is dan het aantal mogelijke klare- of cijferteksten: dan is er voor onjuiste sleutels een mooie kans om af te vallen. Bij DES is dit vanaf het begin het geval omdat er 2^{56} sleutels zijn en 2^{64} strings. Is de sleutellengte gelijk aan of zelfs groter dan de blokgrootte, dan is de eliminatie vooral in het begin veel minder effectief.

De aanval kan nog worden verbeterd door van de overgebleven sleutelkandidaten bij te houden bij welke sleutel in de andere lijst ze horen. In rekentijd maakt dit weinig uit omdat het werk volledig door de eerste ronde wordt gedomineerd, maar er zijn dan maar twee klaretekst-cijfertekstparen nodig.

Wat moeten we aan met 2DES? Het rekenwerk voor een meet-in-the-middleaanval op dubbel-DES is van dezelfde orde als een brutekrachtaanval op (enkele) DES, dus lijkt voor de DES-kraker goed te doen. Het is echter de vraag of dat ontwerp goed uit te breiden is om de nodige disktoegang ook met dezelfde snelheid af te werken.

Vanwege het geheugengebruik is de aanval waarschijnlijk in het geheel niet uitvoerbaar. De beschreven aanval slaat tweemaal 2^{56} strings van acht byte op en zal dus moeten beginnen met een miljardeninvestering in disks en elektriciteitscentrales. Door de lijsten L_1 en L_2 beurtelings te behandelen is wel een halvering van het geheugengebruik mogelijk (zie ook opgave 4.13).

Nu gebruikt niemand graag een systeem waarop een aanval *waarschijnlijk* niet uitvoerbaar is. Men wil meer zekerheid en daarom wordt 2DES niet gebruikt.

4.2.3 Triple-DES (3DES)

De huidige cryptografische praktijk wordt gedomineerd door het twijfelachtige adagio dat voldoende *kwantiteit* een acceptabel substituut is voor gedegen *kwaliteit*. Wij hebben gezien dat DES als algoritmisch zeer sterk mag worden beschouwd (sectie 2.3) maar door ingrijpen van NSA bij het ontwerp een te korte sleutel heeft om een brutekrachtaanval uit te sluiten (sectie 2.2.2). Tweevoudige toepassing verdubbelt de sleutellengte tot een solide 112 bits, maar is kwetsbaar tegen een meet-in-the-middleaanval, met een rekentijd vergelijkbaar met een aanval op de enkele DES.

De volgende stap is drievoudige toepassing. Bij triple-DES, of 3DES, worden drie DES-sleutels afgesproken, samen 168 bits, en wordt driemaal achtereenvolgens versleuteld met de drie onafhankelijke sleutels. Men kan eenvoudig narekenen dat de behandelde aanvallen en trucs op de sleutelomvang volstrekt stuklopen. Een meet-in-the-middleaanval moet aan een kant de 2^{56} DES-sleutels zetten en aan de andere kant de 2^{112} combinaties van twee sleutels, en heeft dus een rekentijd van orde 2^{112}.

Er zijn geen aanvallen bekend waarvan toepassing op 3DES haalbaar is of binnenkort haalbaar zal zijn. Alle cryptografische experts zijn het erover eens dat 3DES nu en de komende jaren als absoluut veilig kan worden beschouwd: er is geen enkele reden om aan te nemen dat 3DES de komende jaren zal kunnen worden gebroken.

Kader 4.1: PrimaCryption producten met 3DES.

Het Nederlandse bedrijf PrimaCryption levert cryptografische modules in de vorm van smartcards, die in een speciale lezer op de pc of laptop worden gestoken. Opslag van sleutels in smartcards wordt tegenwoordig vaak toegepast omdat men zo'n kaartje gemakkelijk bij zich kan dragen, de gebruiker hoeft dus geen sleutel te onthouden. Het ongemerkt uitlekken van de sleutel is vrijwel onmogelijk omdat de kaart niet kan worden gekopieerd.
PrimaCryptionkaarten die voor e-mailverkeer zijn bedoeld (producten *Ambassador* en *President*) worden geleverd in setjes (van twee of twaalf) met gelijke sleutel. E-mail kan worden uitgewisseld met bezitters van een kaart uit dezelfde set. Ook is er een 'open' versie (*Diplomat*) waarmee met iedere Diplomat-bezitter kan worden gecommuniceerd na het overeenkomen van een wachtwoord. Een kaart voor versleuteling van bestanden op een pc of laptop (product *Fort Knox*) wordt geleverd als enkele kaart met één reserve.
Het gebruikte cryptografische algoritme is 3DES.
Bron: `www.primacryption.com`.

De banken gebruiken 3DES voor de beveiliging van vrijwel alle transacties, via betaalautomaten en overboekingen, inclusief voor de elektronische portemonnee chipknip. Ook e-mailbeveiligingssoftware zoals Pretty Good Privacy (PGP, zie kader 6.7) en PrimaCryption (kader 4.1) werken met 3DES.

Om technische reden wordt bij encryptie de tweede sleutel in *decryptie*richting gebruikt, de encryptie en decryptie met sleutel (k_1, k_2, k_3) luiden dus

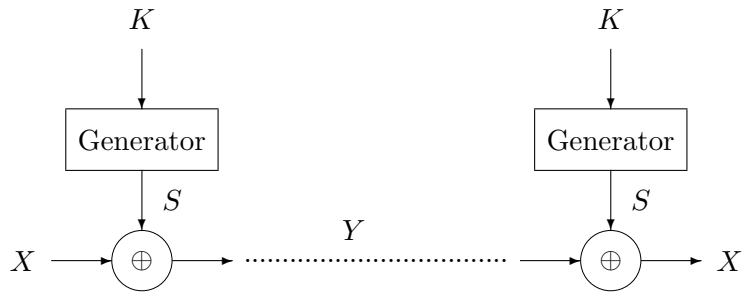
$$Y = E_{k_3}(D_{k_2}(E_{k_1}(X))) \qquad \text{resp.} \qquad X = D_{K_1}(E_{k_2}(D_{k_3}(Y))).$$

Voor de cryptografische bescherming maakt dit weinig uit omdat encryptie en decryptie met DES niet principieel verschillen. Het voordeel is dat op eenvoudige wijze compatibiliteit met DES-gebruikers kan worden verkregen: door $k_2 = k_1$ te kiezen is de encryptie compatibel met enkele toepassing van DES met sleutel k_3.

Ook ziet men wel 3DES toegepast met sleutels van 112 bits: de deelsleutels k_1 en k_3 worden dan gelijk gekozen. De sleutel wordt dan kleiner, maar toch moet bij een meet-in-the-middleaanval een van de lijsten 2^{112} sleutels bevatten. Merkle en Hellman [MH81] beschreven een aanval op deze techniek, die echter veel gekozen-klaretekstparen nodig heeft en daarom practisch slecht uitvoerbaar is.

4.2.4 DES-modes

Men kan een lang bestand versleutelen door het te verdelen in blokken van 64 bits, die dan elk afzonderlijk met DES of 3DES worden versleuteld. Schrijft men een lang bestand X als $X = X_1X_2X_3 \ldots X_n$, dan geldt bij deze toepassing simpelweg $Y_i = E(X_i)$. Deze manier van encryptie heet *electronic codebook mode* en heeft het nadeel dat gelijkluidende blokken in de plaintext ook in de ciphertext steeds gelijkluidend zijn. Opgave 2.2 is op deze zwakke plek gebaseerd. Niet alleen kan het begrijpen van een bericht erdoor mogelijk worden, ook kan een tegenstander blokken uit een bericht vervangen door andere waarvan hij de betekenis heeft geraden.



Figuur 4.2: Principe van Stroomversleuteling

Er zijn diverse strategieën om deze zwakte te ondervangen. Steeds wordt bij het versleutelen van een blok informatie gecombineerd met informatie uit de vorige blokken. Bij deze methoden moet men erop letten, dat de gebruikte informatie ook bij de ontvanger op tijd bekend is. Bij de veelgebruikte *cipher block chaining* wordt elk blok geXORd met de vorige ciphertext: $Y_i = E(X_i \oplus Y_{i-1})$. De encryptie van blok i hangt zo af van alle voorgaande blokken.

Door dit schakelen wordt het voor een tegenstander moeilijker individuele blokken te interpreteren of te manipuleren, maar de communicatie is ook kwetsbaarder tegen transmissiefouten. Het verminken van een enkel blok tijdens transmissie leidt tot het verlies van meerdere datablokken, soms zelfs tot alle latere blokken.

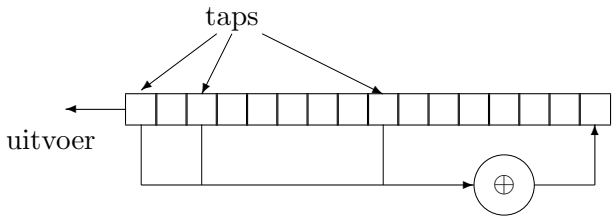
4.3 Stroomversleuteling

Bij bloksystemen zoals DES en alle andere Feistel-systemen wordt de data zelf gecombineerd met sleutelmateriaal, zie sectie 2.1.1. Een geheel andere aanpak is juist om de data in zijn geheel eenmaal te XOR-en met een bitrij die onafhankelijk van de data is berekend; zie figuur 4.2.

De sleutel K wordt gebruikt om een generatormodule aan te sturen die een lange rij bits genereert. De module is deterministisch en wordt door de zender en de ontvanger met dezelfde sleutel aangestuurd, zodat dezelfde bitrij S wordt verkregen. Het principe doet denken aan het one-time pad (sectie 1.3.1), maar een belangrijk verschil is dat het pad S niet willekeurig is, maar uit de veel kortere sleutel K wordt afgeleid.

4.3.1 Het A5-algoritme voor gsm-telefonie

Als voorbeeld van een stroomsysteem bekijken we het A5 algoritme dat wordt gebruikt in gsm-telefoons [BSW00]. Communicatie vindt in twee richtingen tegelijk plaats en de generator levert bits om beide richtingen te versleutelen. Het analoge signaal wordt gedigitaliseerd en gecomprimeerd, en er wordt stuurinformatie aan toegevoegd, waarna de datastroom in frames van 114 bits wordt geknipt. Een frame staat voor 4,6 milliseconde van de conversatie (die dus inclusief stuurinformatie zo'n 24 kilobit per seconde vraagt). Omdat er bij gsm-communicatie zeer vaak blokken door transmissiefouten worden verminkt kan een chaining mode niet worden gebruikt, en dus moeten de frames onafhankelijk van elkaar worden versleuteld.



Figuur 4.3: Principe van een schuifregister

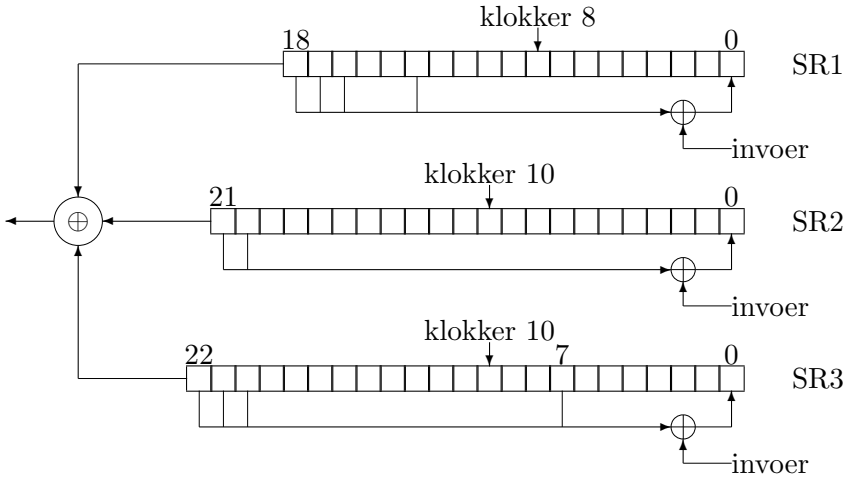
Het A5-algoritme gebruikt een 64-bitssleutel, en het 22-bitsserienummer i van de frames, die samen door een eenvoudige hardwaremodule worden afgebeeld op een 228-bitsstring S_i. Elke communicatierichting gebruikt de helft van deze bits. Er zijn twee varianten: A5/1 wordt in de Europese telefonie gebruikt en A5/2 in de exporttoestellen; we beschrijven hier A5/1.

Linear Feedback Shift Registers. Een eenvoudig middel om lange bitrijen te genereren is een schuifregister of *Linear Feedback Shift Register* (LFSR), zie figuur 4.3. Het register bestaat uit een bitrij, die in elke stap een positie naar links kan worden verschoven. De meest linkse bit is dan de uitvoer, terwijl aan de rechterkant een combinatie van een paar bits op vaste posities (de *taps*) wordt ingevoerd.

Een schuifregister met n bits heeft natuurlijk precies 2^n toestanden. Een ervan is bijzonder, namelijk die waarin alle bits gelijk aan nul zijn: de volgende uitvoerbit is een nul, en de volgende toestand is dezelfde, zodat het register de oneindige rij $000000\ldots$ zal genereren. Door de keuze van de taps kan een register zo worden ontworpen, dat de uitvoer repeteert met *maximale lengte*, dit wil zeggen dat alle 2^n-1 overige toestanden achtereenvolgens voorkomen, en de uitvoer periodiek is met periodelengte 2^n-1.

De registerstructuur van A5/1. Figuur 4.4 toont de drie schuifregisters van A5/1, met lengte 19, 22 en 23 bits, en hun taps; de in totaal 64 bits geven 2^{64} toestanden. (Sluiten we alle toestanden uit waarbij één van de registers geheel op nul staat, dan blijven $(2^{19}-1)(2^{22}-1)(2^{23}-1)$ combinaties over, maar dit is nagenoeg gelijk aan 2^{64}.) De string S_i voor het versleutelen van frame i wordt als volgt berekend.

1. De schuifregisters worden op 0 gezet en er worden 64 slagen gedraaid, waarbij steeds een bit van de sleutel wordt meegeXORd voor het berekenen van de rechter bit.
2. Vervolgens worden er 22 slagen gedraaid waarbij bits van het framenummer worden meegeXORd.
3. Er worden honderd slagen gedraaid waarbij de uitvoer wordt genegeerd. In deze fase worden de registers gestuurd op basis van een klokbit, een speciale tap op elk register. Een register wordt alleen geschoven als de waarde van zijn klokker overeenkomt met de meerderheid van de drie klokkers. Dit is de enige manier waarop de drie registers elkaar beïnvloeden. In elke slag worden twee of drie registers verschoven.
4. Er worden 228 slagen gedraaid, weer met gebruik van de klokkers, waarbij de 228 bits uitvoer worden gebruikt als S_i.



Figuur 4.4: Schuifregisters van A5/1

In totaal duurt de procedure 414 slagen van de registers, en door implementatie in hardware is de benodigde rekenkracht verwaarloosbaar.

4.3.2 Aanval op A5

Door Biryukov e.a. [BSW00] werd een known plaintext attack op A5/1 beschreven; de behandeling beperkt zich hier tot het noemen van de belangrijkste zwakheden die worden uitgebuit. Dit begint met de sleutellengte die, met 64 bits, net als bij DES weer iets aan de kleine kant is. De standaardtrucs als Hellmans aanval zijn daardoor weliswaar niet direct haalbaar, maar zijn, zoals door Biryukov opgemerkt, slechts "marginaal onhaalbaar".

Zwakke plekken van A5/1. De aanval buit de volgende zwakke punten van het algoritme uit; de meeste zijn wellicht door de ontwerpers niet voorzien.

1. Het A5/1-algoritme is heel snel op een pc uit te voeren. Dit komt omdat je van elk van de drie schuifregisters een circulaire bitrij kunt maken die alle achtereenvolgende toestanden van dat register weergeeft. Omdat de klokker van elk register op enige afstand van de andere taps ligt, kun je een aantal toestanden in één keer vooruit rekenen.
2. Het A5/1-algoritme is redelijk snel terug te rekenen. Elke registertoestand heeft precies één opvolger, maar van nul tot vier voorgangers. Omdat nu het gemiddelde aantal voorgangers precies één is, kun je redelijk goed de registertoestand terugrekenen met backtracking zonder dat de recursieboom erg breed wordt.
3. De waarschijnlijkheidsverdeling van toestanden na fase 3 is zeer scheef. De kans op zekere toestand σ is namelijk ongeveer evenredig met het aantal toestanden in de beginfase dat na invoering van het framenummer en de honderd loze slagen op σ wordt afgebeeld.

Ontwerp van de aanval. Voor de aanval is een voorberekening nodig die enkele maanden kan nemen op een netwerk van pc's: Op disk wordt een flink aantal, zeg A, toestanden

Kader 4.5: Het A5/1-algoritme en Kerckhoffs' principe

Hoe kan het favoriete luiermerk van de familie Doorsnee uitlekken naar justitie en maffia?
De miljoenen Europese gsm-toestellen zijn tegen afluisteren via de ether beveiligd met het A5/1-algoritme; een zwakkere versie A5/2 wordt gebruikt voor export naar landen waar het gebruik van sterke encryptie door de overheid aan banden is gelegd.
De werking van het algoritme werd door de ontwerpers niet gepubliceerd, maar was natuurlijk bij de hardwarefabrikanten wel bekend. In de loop der jaren lekte er een en ander over uit, en in 1999 slaagden Briceno, Goldberg en Wagner [BGW99] erin, het complete algoritme te achterhalen door reverse engineering van een gsm-telefoon. Geheimhouding kan blijkbaar niet worden gebruikt als middel om de veiligheid van een veelgebruikt algoritme te vergroten.
Na het uitlekken van het algoritme werden ook aanvallen beschreven. Door openheid over het algoritme vooraf zouden wellicht deze aanvallen ook eerder bekend zijn geweest, waardoor invoering van het algoritme achterwege had kunnen blijven. De gang van zaken rond het A5/1-algoritme voor gsm-telefonie steunt daarom Kerckhoffs' principe: cryptografische bescherming moet alleen berusten op geheimhouding van de sleutel, niet op geheimhouding van het algoritme.

van de schuifregisters opgeslagen, geïndexeerd met de output die vanuit die toestand wordt gegenereerd.

Vervolgens wordt een stuk uitvoer van het A5/1-algoritme van lengte B bekeken, en er wordt gezocht naar output die op de disks voorkomt. Normaal wordt de kans op een treffer voldoende groot wanneer het product $A.B$ zo groot wordt als het aantal toestanden, dus 2^{64} (stelling A.7). Door de scheve kansverdeling van toestanden valt de kans bij een juiste keuze van opgeslagen toestanden ongeveer honderd maal gunstiger uit.

Zodra een treffer is gevonden, kan de toestand van de schuifregisters uit de tabel worden gelezen en de sleutel worden teruggerekend.

Bij het bepalen van de tabelgrootte gingen de ontwerpers uit van de grootste verkrijgbare harddisks, namelijk van 73GB; met twee stuks wordt een beschikbare tabelgrootte van bijna 150GB gerealiseerd.

Lessen uit deze aanval. De beschreven aanval is waarschijnlijk in een daadwerkelijk gsm-netwerk niet uitvoerbaar; de ciphertext Y is gemakkelijk te verkrijgen, maar het is in het gsm-systeem erg moeilijk om de plaintext X te achterhalen, zelfs als het geluidssignaal beschikbaar is. Het geluid wordt immers gedigitaliseerd, gecomprimeerd, en gecombineerd met allerlei voor de aanvaller onbekende controle-informatie (zoals meldingen over het al of niet ontvangen van een frame van de tegenpartij). Om een compleet frame van A5/1-uitvoer te bepalen, moet deze informatie voor beide richtingen van de conversatie bepaald worden. Barkan en Biham [BB06] beschreven manieren om delen van de plaintext te achterhalen.

De onuitvoerbaarheid van de aanval betekent natuurlijk niet dat A5/1 acceptabel is. Geheimhouding van de conversatie wordt namelijk niet langer verzekerd door de kwaliteit van de gebruikte cryptografie, maar door eigenschappen van andere delen van het systeem, die helemaal niet op het bieden van cryptografische bescherming zijn ontworpen.

De aanval kan ook worden uitgevoerd op alleen de eerste of tweede helft van elk frame, dus met de plaintext van slechts één richting van communicatie. Oscar kan een boodschap op Bobs voicemail inspreken en wachten tot Bob die beluistert: de bekende (eigen) boodschap levert Oscar de sleutel, waarna hij de overige voicemailberichten (van Alice) mee kan luisteren. Voor Bobs volgende conversatie wordt de sleutel niet meer gebruikt.

De A5/1-geschiedenis bevat een paar fouten gemaakt door de ontwerpers. Dat een sleutelomvang van 64 bits onvoldoende is, had men moeten weten. Men heeft in afwijking van Kerckhoffs' principe het algoritme geheim willen houden, wat uiteindelijk niet is gelukt, maar wat wel een tijdige doorlichting van het algoritme heeft tegengehouden.

Opvolger: Kasumi. De volgende generatie van mobiele telefoons, het UMTS-systeem, zal worden beveiligd met een nieuw algoritme: Kasumi. Kasumi is een symmetrisch algoritme dat werkt met sleutels van 128 bits op datablokken met een grootte van 64 bits. Het volgt het Feistel-principe en heeft acht ronden. De S-boxen van Kasumi zijn speciaal ontworpen voor een zeer hoge bescherming tegen differentiële en lineaire aanvallen. Het ontwerp is beschreven in open literatuur.

Een behandeling van Kasumi en andere beveiligingsmechanismen van UMTS volgt in sectie 10.2.

4.4 Sleutelbeheer

Zoals gezegd is het voor toepassing van symmetrische encryptie nodig dat de zendende en ontvangende partij over dezelfde sleutel beschikken, zonder dat de tegenpartij er eveneens over beschikt. Het probleem om dit voor elkaar te krijgen noemen we *sleutelbeheer* of *key management* en is het voornaamste struikelblok bij toepassing van symmetrische cryptografie.

Wanneer een gezelschap van, zeg k personen, met elkaar wil communiceren zijn er twee uitersten voor wat betreft de onderlinge privacy. Ten eerste kan men binnen deze club van een enkele sleutel gebruik maken; er hoeft dan maar één sleutel te worden aangemaakt en bewaard, maar het nadeel is dat communicatie tussen twee personen door alle anderen ook kan worden gelezen. Het andere uiterste is voor elk tweetal een andere sleutel te gebruiken; elke gebruiker bezit dan $k-1$ sleutels, en het totale aantal is $\frac{k(k-1)}{2}$. Er is betere onderlinge privacy, nadeel is wel dat rondzendbrieven voor elke deelnemer apart moeten worden vercijferd en verstuurd.

We zullen nu enkele populaire methoden voor het maken en verspreiden van sleutels bekijken.

Kader 4.6: Een Telewerkprotocol

ITsec is een Haarlems bedrijf dat sinds 1995 diensten en producten levert op het gebied van informatiebeveiliging en electronisch afrekenen.
In 1999 ontwierp De Boer [Boe99] een protocol dat het telewerken zo veilig mogelijk kan ondersteunen. De telewerkende werknemer beschikt thuis over een pc met smartcardlezer en krijgt een smartcard met een sleutel. De smartcard kan door de werknemer worden ontsloten met een toegangscode, waarna deze zich via het internet identificeert tegenover de bedrijfsserver. Tijdens deze identificatie wordt een sessiesleutel bepaald met het Diffie-Hellman-protocol met dit verschil, dat de getallen (x en y) per sessie opnieuw worden gekozen. De sessiesleutel wordt gebruikt om al het verkeer met 3DES te versleutelen, en wordt elke vijftien minuten vernieuwd. De 3DES-encryptie vindt in de pc plaats, maar de geheime sleutel die voor identificatie nodig is verlaat de smartcard nooit.

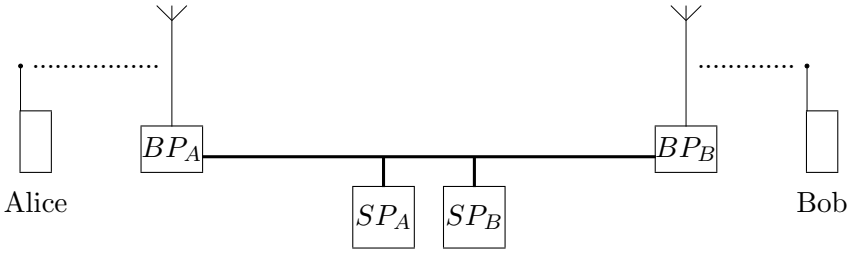
4.4.1 Het Diffie-Hellman-protocol

Een van de oudste methoden, maar nog steeds veel toegepast, is dat van Diffie en Hellman [DH76]. Deze methode is gebaseerd op het onderscheid tussen publieke en private informatie van een partij, en is getaltheoretisch van aard; door deze twee eigenschappen lopen we met de behandeling van het algoritme vast vooruit op hoe de meeste moderne cryptografische algoritmen werken. Getallen voldoen aan allerlei rekenregeltjes, waardoor je op verschillende manieren iets kunt berekenen en toch hetzelfde resultaat verkrijgen.

Het systeem maakt gebruik van een algemeen en publiek bekend getal g en alle berekeningen worden gedaan modulo een groot priemgetal p. Elke gebruiker kiest een *geheim getal* x, en berekent het *publieke getal* $y = g^x$; deze y wordt gepubliceerd, bijvoorbeeld op de homepage. Wil Alice met Bob communiceren, dan leest zij y_B (Bobs publieke getal) en exponentieert dat met haar eigen geheim, dus berekent $K = y_B^{x_A}$; Bob leest het getal y_A van Alice en berekent $y_A^{x_B}$. De twee uitkomsten zijn gelijk, en zo hebben Alice en Bob zonder directe onderlinge communicatie een sleutel afgesproken.

Kan Oscar uit de hem bekende informatie (y_A en y_B) de sleutel afleiden? Dit komt neer op het vinden van een getal $k = g^{x_A \cdot x_B}$ wanneer g^{x_A} en g^{x_B} bekend zijn. In hoofdstuk 5 zien we dat het berekenen van logaritmen in eindige groepen (dwz., renenen modulo p) niet efficiënt mogelijk is. Het vinden van k staat bekend als het *Diffie-Hellman-completeringsprobleem* en het is waarschijnlijk moeilijk (zie sectie 6.2.1).

Of het berekenen van een logaritme werkelijk niet uitvoerbaar is, hangt uiteraard af van



Figuur 4.7: Mobiele en vaste stations in gsm-netwerk

de grootte van de gebruikte getallen. Het Amerikaanse NIST heeft in 2010 de aanbeveling gedaan, geen getallen meer te gebruiken met een lengte van 1024 bits, maar voor alle (nieuwe) keys tenminste 2048b te gebruiken. Adrian e.a. [ABD+15] rapporteren dat zij met eenvoudige middelen logaritmen konden berekenen voor 512b moduli, dat universitaire teams zich kunnen inspannen om tot 768b te komen, en dat de NSA zeer waarschijnlijk al vanaf ongeveer 2013, 1024b moduli kan aanvallen. Ook zagen zij, dat de NIST aanbeveling op grote schaal wordt genegeerd, zodat 1024b moduli nog zeer veel worden gebruikt.

4.4.2 Beveiligde verzending

De sleutels kunnen op een centrale plaats worden gekozen en via een beschermd kanaal naar Alice en Bob worden overgebracht. PrimaCryption (kader 4.1) en ITsecs protocol (kader 4.6) gebruiken hiervoor smartcards, waarbij het vrijwel onmogelijk is de sleutels onbevoegd en/of ongemerkt uit de cards te lezen. In de tijd van geponste papierstroken werden de rollen papier in verzegelde cassettes vervoerd (sectie 1.3.1).

Een wat omslachtiger werkwijze is, de sleutel door een partij te laten bepalen en dan met een ander cryptografisch algoritme te versleutelen voor de andere partij. Dit andere algoritme is dan van het *public-keytype*, waarbij het sleutelbeheer een veel eenvoudiger probleem is. Dit is precies wat er in PGP (Pretty Good Privacy, zie kader 6.7) gebeurt. De boodschap wordt met een symmetrisch algoritme versleuteld, en de gebruikte sleutel wordt met public key versleuteld en meegezonden. De werkwijze combineert het gemakkelijke sleutelbeheer van public-keysystemen met de encryptiesnelheid van symmetrische algoritmen.

4.4.3 Encryptie en sleutelkeuze bij gsm

Bij een gsm-conversatie onderhoudt elk mobiel station een radioverbinding met een nabijgelegen *basispost*; zie figuur 4.7. De basisposten communiceren met elkaar en met de administratieve centra van de serviceproviders via een vast netwerk van glasvezelkabels (en incidenteel straalverbindingen).

De encryptie met het A5-algoritme, zoals beschreven in sectie 4.3.1, wordt alleen op het radiogedeelte toegepast, dus tussen Alice en haar basispost BP_A. Er is dus geen sprake van eind-eindencryptie tussen Alice en Bob, waarvan zij zelf het sleutelbeheer in eigen hand houden. Dat zou met A5 niet kunnen, omdat, zoals we zagen, de controleinformatie wordt meeversleuteld, en de basispost moet hier toegang toe hebben. Er bestaan wel telefoniesystemen die eind-eindencryptie bieden, maar gsm behoort daar niet toe. Men heeft ervoor gekozen

de beveiliging te beperken tot afluisteren via de ether, want het aftappen van gesprekken via de vaste infrastructuur (door de providers en overheid) moest mogelijk blijven ten behoeve van justitiële opsporing.

Voor de toepassing van A5 is nodig, dat Alice' mobieltje en de basispost BP_A over dezelfde sleutel K beschikken; het is hierbij mogelijk dat BP_A *niet* tot de infrastructuur van Alice' provider SP_A behoort. De sleutel van Alice wordt berekend op basis van een geheime string Y_A die aanwezig is in de SIM van de handset en die bekend is bij de serviceprovider SP_A die de SIM heeft uitgegeven. Bij het begin van de conversatie kiest SP_A een random string X en past op X en Y_A een functie A8 toe: $K = A8(X, Y_A)$. In een bericht over het vaste netwerk stuurt SP_A aan BP_A zowel X als K. De basispost gebruikt K en stuurt X door de ether naar Alice, die Y_A uit de SIM leest en K berekent als $K = A8(X, Y_A)$.

De functie A8 is niet gestandaardiseerd en verschilt per provider; zoals boven geschetst, wordt deze alleen door Alice en SP_A toegepast om K te bepalen, de basispost hoeft hetzelfde A8-algoritme dus niet te kennen. Een indringer die het bericht met K van het vaste netwerk leest, krijgt daarmee de sleutel K voor die conversatie in handen, maar niet het geheim Y_A waarmee de volgende conversaties van Alice kunnen worden afgeluisterd. Ook de basispost verkrijgt geen informatie die de sleutel van andere conversaties van Alice kan onthullen. In lange conversaties kan de sleutel worden veranderd door hernieuwde interactie met SP_A en keuze van X; ook als Alice zich tijdens de conversatie naar een andere basispost beweegt gebeurt dat.

4.4.4 Het masterkeysysteem

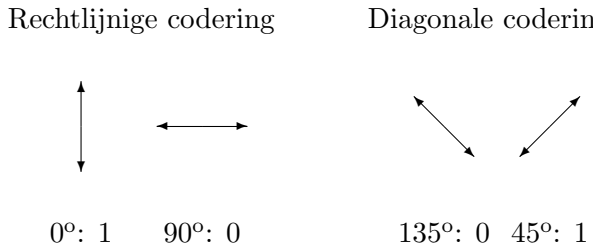
Net als bij gsm-telefonie is er bij de elektronische betaalmiddelen zoals de chipknip sprake van een groot aantal mobiele stations, die bij een beperkter aantal vaste stations (hier de betaalpunten) terecht kunnen. In beide gevallen moeten het vaste en mobiele station een encryptiesleutel delen.

Bij het gsm-systeem is echter de communicatie tussen de basispost en de serviceprovider (die de geheime string Y_A bewaart) essentieel. De laagdrempelige inzet van de chipknip vereist dat de betaalpunten kunnen opereren zonder verbinding met de bank. Het is niet mogelijk, elk betaalpunt te voorzien van een geheime sleutel voor elke betaalpas: het kost te veel geheugen, geeft een te groot risico van misbruik, en is onvoldoende flexibel bij de uitgifte van nieuwe passen of het intrekken van oude.

Om de betaalpunten zonder ruggespraak met de bank aan een geheime sleutel van de betaalpas te laten komen wordt gebruik gemaakt van een masterkey M. Deze sleutel is heel belangrijk omdat hij voor de toegang tot alle passen wordt gebruikt, en M wordt dan ook streng geheimgehouden. Elke bankpas is voorzien van een uniek en publiek serienummer x; bij het aanmaken wordt de pas ook voorzien van een sleutel, berekend uit x en de masterkey, namelijk door x met de masterkey te encrypten: $K_x = E_M(x)$. De pas bevat dan x en K_x, maar niet de masterkey M; je kunt dus het sleutelbeheer-systeem niet openbreken door een bankpas te ontleden.

De pas meldt zich bij een betaalpunt met vermelding van het nummer x; betaalpunten bevatten M *wel*, zodat het betaalpunt K_x kan uitrekenen. De communicatie tussen de partijen vermeldt alleen het serienummer x, dus alleen een afluisteraar die eveneens over M beschikt kan hieruit de waarde K_x berekenen. Is deze waarde eenmaal uitgelekt, dan is de pas ook verder waardeloos omdat bij elke transactie dezelfde sleutel K_x wordt gebruikt.

De partijen gebruiken de sleutel vervolgens om zich van de 'echtheid' van de tegenpartij te overtuigen. Encryptie van een random string wordt toegepast om te zien dat de ander eveneens over dezelfde waarde van K_x beschikt. De betaalautomaat concludeert dat de aangeboden pas



Figuur 4.8: Rechtlijnige en diagonale codering

naast x ook K_x kent, met kennis van M is gemaakt en 'dus' echt is. De pas concludeert dat de machine uit x de waarde K_x kan berekenen, M bevat en 'dus' een echte betaalautomaat is.

Het gebruikte cryptografische algoritme is, zoals in vrijwel alle bancaire systemen, 3DES.

4.4.5 Kwantumcryptografie

Het afluisteren door Oscar van de communicatie tussen Alice en Bob moet in fysische zin worden vergeleken met een meting aan het verzonden signaal. Uit de kwantummechanica, de natuurkunde van de elementaire deeltjes, is bekend dat het niet mogelijk is een signaal te meten zonder het te verstoren. Dit principe kan worden gebruikt om te communiceren met fotonen (lichtkwanta), waarbij een poging om de communicatie te beluisteren direct door de ontvanger als een verandering in signaal wordt gemerkt.

Lichtkwanta en bemeting. De kleinste eenheid van licht (of andere straling) is het *foton*, dat men zich voor moet stellen als een klein trillend elektrisch veld. De richting waarin het veld trilt heet de *polarisatie* van het foton.

Een foton wordt alleen gemeten wanneer de stand van de detector overeenkomt met de polarisatierichting van het foton, waardoor communicatie kan plaatsvinden door de polarisatie van achtereenvolgende fotonen. In de *rechtlijnige codering* gebruikt Bob een zuiver verticale detector, en Alice codeert een één als een verticaal gepolariseerd, en een nul als een horizontaal gepolariseerd foton. Een verticaal foton geeft in Bobs detector een reactie (die als één wordt geïnterpreteerd) en een horizontaal foton geeft geen reactie (dus een nul).

Een foton met een schuine (dus niet zuiver horizontale of verticale) polarisatie geeft in Bobs detector niet, zoals men misschien zou verwachten, een wat zwakkere reactie. De kwantummechanica sluit dit uit en leert dat het foton óf wel, óf niet een reactie zal geven. Hierbij is de kans op een reactie groter naarmate de polarisatie dichter bij de verticale ligt. Wanneer de polarisatie van een foton exact diagonaal is (een hoek van 45° met de verticaal maakt), is de kans op een reactie exact een half.

Dit betekent dat de informatie in een foton wordt verstoord wanneer men niet precies weet hoe de polarisatie gemeten moet worden, waardoor pogingen tot afluisteren door Bob en Alice worden bemerkt. Naast de rechtlijnige codering wordt een diagonale codering gebruikt, waarbij Bobs detector over 45° is gedraaid; zie figuur 4.8. Bij het versturen van een bit maakt Alice een keuze tussen rechtlijnige of diagonale codering. Wanneer de ontvanger de detector op de gebruikte codering instelt, wordt de juiste waarde van de bit ontvangen. Wordt de detector echter op de andere codering ingesteld, dan maakt het binnenkomende foton altijd een hoek

van 45° met de detector, en de reactie van de detector is volledig random. Bij onjuiste meting gaat de informatie in het foton dus verloren.

Sleuteluitwisseling met kwantumcryptografie. Wanneer Alice en Bob een sleutel willen overeenkomen, stuurt Alice een reeks van random bits over een kwantumkanaal, waarbij ze voor elke bit een random keuze maakt tussen rechtlijnige en diagonale codering. Bob probeert de reeks te ontvangen en maakt ook voor elke bit een random keuze tussen rechtlijnige en diagonale codering. Na afloop van deze fase beschikt Bob over de juiste waarde van die bits, waarvoor Alice en hij dezelfde codering kozen; de andere bits zijn veranderd in random bits.

Bob vertelt Alice (via een onbeschermd kanaal) welke reeks coderingen hij heeft gebruikt, en Alice vertelt Bob welke van zijn metingen de juiste codering gebruikten. Alice en Bob verwijderen uit hun bitrij alle waarden waarvoor de meting van Bob niet de juiste codering gebruikte. Als er geen poging is gedaan hun kwantumcommunicatie af te luisteren zijn de twee bitrijen gelijk; deze gelijkheid kan worden gecontroleerd, bijvoorbeeld door een checksum over deze reeks uit te wisselen. De reeks succesvol verstuurde bits kan als sleutel worden gebruikt.

Oscar kan de kwantumcommunicatie niet ongemerkt afluisteren. Hij weet namelijk niet welke codering voor iedere bit is toegepast, en een verkeerde codering vernietigt de informatie in het foton. Het gevolg is dat voor de bits waar Bob de juiste codering gebruikte maar Oscar niet, de waarde (met kans een half) is veranderd.

Verdere mogelijkheden van kwantumcryptografie. Communicatie met polaritatie van fotonen is geen science fiction: met kwantumcommunicatie zijn door IBM succesvolle proeven gedaan. Kwantumcryptografie biedt meer mogelijkheden dan sleuteluitwisseling, zo zijn er bijvoorbeeld ook kwantumprotocollen voor blinde overdracht (sectie 14.4.1). De noodzaak van speciale kwantumapparatuur en -kanalen betekent echter dat het gebruik van kwantumcryptografie beperkt zal blijven tot zeer speciale (bijvoorbeeld militaire en satelliet-) toepassingen. In dit boek zal kwantumcryptografie verder niet worden besproken.

4.4.6 Opslag en gebruik

Na het aanmaken en verspreiden van een sleutel moet deze vaak gedurende lange tijd worden gebruikt en bewaard, liefst zonder gevaar van uitlekken. Het is mogelijk een cryptografische sleutel simpelweg te onthouden, net als een password, maar dit is tegenwoordig niet meer afdoende. De lengte van een te onthouden password komt ongeveer overeen met een enkele DES-sleutel, waarvan we weten dat die onvoldoende bescherming biedt.

Dan kan een sleutel natuurlijk worden opgeschreven, in een computer worden opgeslagen of op een magneetstrip of streepjescode worden gezet zodat hij automatisch is in te lezen. Nadeel van deze vormen van opslag is dat een tegenstander de sleutel ongemerkt te pakken kan krijgen als hij het medium ongestoord kan inspecteren.

Het opslaan van de sleutel in een smartcard (hoofdstuk 12) biedt meer mogelijkheden. De smartcard is namelijk niet te kopiëren, en ook niet ongemerkt te openen of uit te lezen, waarmee al een belangrijk voordeel ten opzichte van briefjes of magneetkaarten is gescoord: *alleen* met gebruik van de echte kaart kan worden versleuteld. Een verder voordeel is de mogelijkheid de kaart te beveiligen met een pincode of uitgebreidere identificatieprocedure. De kaart moet dan aan de pc worden aangeboden, en vraagt om een code (die de rechtmatige gebruiker via de pc kan invoeren) voordat de sleutel gebruikt kan worden.

Vervolgens kan hetzij de sleutel naar de pc worden overgebracht, danwel de data naar de kaart worden gestuurd voor encryptie of decryptie. Het voordeel van de eerste mogelijkheid

(toegepast bij PrimaCryption, kader 4.1) is dat de snelheid van versleuteling niet wordt geremd door de trage pc-kaartinterface. Het nadeel is dat de ontwerper van het systeem onvoldoende in de hand heeft wat er in de pc met de sleutel gebeurt: de pc is, in tegenstelling tot de smartcard, *geen* 'veilig platform'. Een pc kan worden herprogrammeerd met kwaadwillende software en de sleutel is dan in de pc niet veilig. In het ITsec telewerkprotocol (kader 4.6) blijft de geheime sleutel altijd op de kaart, maar de sessiesleutel (die per kwartier verandert) is in de pc aanwezig.

4.5 AES: Advanced Encryption Standard

Symmetrische encryptie heeft zich, zowel in de gedaante van stroomversleuteling als van Feistelnetwerken, bewezen als levensvatbare (efficiënte én veilige) techniek voor de bescherming van data. Bij het ontwerp van op grote schaal gebruikte algoritmen zijn echter fouten gemaakt waardoor deze niet meer acceptabel zijn. DES en A5 hebben te korte sleutels, A5/1 heeft daarnaast een zwak ontwerp, dat door geheimhouding niet kan worden beschermd. Dit leidt ertoe, dat diverse partijen uitwijken naar allerlei andere algoritmen, zoals 3DES en IDEA, en dit komt uiteindelijk de interoperabiliteit en de veiligheid niet ten goede.

In 1997 is daarom een programma gelanceerd om tot een algemeen aanvaarde opvolger van DES te komen. Het nieuwe algoritme zou AES (Advanced Encryption Standard) heten en net als in 1974 was er een open inschrijving van voorgestelde algoritmen. Er zijn in 1998 maar liefst vijftien algoritmen ingediend, waaruit in het najaar van 2000 een keuze voor de nieuwe standaard werd gemaakt. De nieuw gekozen cryptografische standaard is het algoritme Rijndael, ontworpen door Joan Daemen (Proton World Int.) en Vincent Rijmen (Katholieke Universiteit Leuven) [DR99].

4.5.1 Rijndael: korte beschrijving

Rijndael is, net als DES, IDEA en vele andere cryptografische algoritmen, een versleuteling die blokken data van steeds dezelfde lengte bewerkt in een aantal opeenvolgende ronden. Elke ronde wordt gestuurd door een *rondesleutel* die is afgeleid van de hoofdsleutel. Rijndaels proces voor sleutelgeneratie is wat ingewikkelder dan dat voor DES, de encryptie zelf is overzichtelijker omdat er geen S-boxen van onduidelijke herkomst worden gebruikt. Zowel de blokgrootte als de sleutellengte zijn flexibel, namelijk 128b, 192b of 256b, onafhankelijk van elkaar te kiezen.

De versleutelronde volgt niet het pure Feistel-principe, waarbij steeds de helft van de data onveranderd blijft in een ronde; Rijndael verandert in elke ronde alle bits. Het algoritme is bytegeoriënteerd en ziet de data als een matrix van vier rijen met elk vier, zes of acht bytes (afhankelijk van de blokgrootte). De versleuteling in elke ronde vindt in vier layers (fasen) plaats.

1. **Bytesubstitutie.** Een vaste substitutie wordt toegepast op elke individuele byte; dat wil zeggen, er vindt een permutatie van de 256 mogelijke waarden van een byte plaats. Implementatie kan een tabel van lengte 256 gebruiken, maar de gebruikte substitutie is ook te schrijven als een lineaire functie op getallen in $GF(2^8)$. De substitutie is dan $y = a \cdot x^{-1} + b$ (met constanten a en b) en de inverse is natuurlijk $x = a \cdot (y - b)^{-1}$. De beschrijving van $GF(2^8)$ valt buiten het doel van dit hoofdstuk, maar zie sectie 15.2.3.

2. **Rijrotatie.** De bytes worden per rij geroteerd over een aantal posities dat per rij verschilt en afhangt van de blokgrootte.

3. **Kolomvermenigvuldiging.** Per kolom wordt een vaste transformatie uitgevoerd; deze implementeren als tabel is niet mogelijk aangezien de tabel ruim vier miljard 32-bits-entries zou moeten bevatten. De kolommen (vier bytes) worden geïnterpreteerd als polynomen met coëfficiënten in $GF(2^8)$ en vermenigvuldigd met een vast polynoom.

4. **Sleuteltoevoeging.** Alle data wordt puntsgewijs geXORd met de rondesleutel (die dezelfde lengte heeft als de data).

Er komen in dit algoritme elementen voor die ook bij DES te vinden zijn, zoals de afwisseling van substitutie en permutatie, alleen in dit modernere algoritme vinden deze op byteniveau plaats. De DES-sleutelgeneratie verplaatst alleen maar bits zodat elke iteratiekeybit overeenkomt met een bepaalde hoofdsleutelbit; de sleutelgeneratie is *lineair*. In Rijndael wordt de hoofdsleutel 'verlengd' door sleutelbytes te XOREn met andere sleutelbytes, soms na toepassing van de S-box op een van de operanden. (N.B.: er is natuurlijk, vanuit het oogpunt van veiligheid, niet echt sprake van verlenging van de sleutel omdat de toegevoegde bits berekend worden uit de hoofdsleutel.) Dit proces wordt voortgezet tot er genoeg sleutelbytes zijn voor alle iteraties.

Het aantal ronden hangt af van de blok- en sleutelgrootten en is 10, 12 of 14.

4.5.2 Waarom Rijndael?

In 1998 zijn vijftien voorstellen voor cryptografische algoritmen ingediend en deze werden geëvalueerd aan de hand van een aantal vooraf vastgestelde criteria. Uiteraard was de veiligheid van het algoritme hierbij nummer één, maar ook snelheid, flexibiliteit in blok- en sleutellengte, en verschil tussen encryptie- en decryptietijd telden mee. In de laatste selectieronde bleven vijf kandidaten over, waarvan het NIST schrijft dat ze allemaal aan de geformuleerde eisen voldoen.

Bovenop de genoemde eisen werd van Rijndael aangetoond dat het enkele aanvullende goede eigenschappen heeft.

1. Rijndael is niet alleen computationeel efficiënt, maar is ook heel goed te implementeren met weinig geheugenruimte (tabellen) wat het voor bijvoorbeeld smartcards geschikt maakt.

2. Er is goede bescherming tegen *stroomaanvallen* en *tijdaanvallen*. Hiermee worden technieken bedoeld die van een hardwaremodule de verstreken tijd en eventueel de opgenomen stroom opmeten tijdens de encryptie (sectie 12.2). Is de uitvoering van bepaalde instructies op een of andere manier afhankelijk van een sleutelbit, dan kan uit de opgenomen stroom of verstreken tijd informatie over de sleutel worden afgeleid. Bij Rijndael echter wordt sleutelmateriaal alleen voor een XOR gebruikt, waarmee dit soort aanvallen weinig kansrijk zijn.

3. Het algoritme heeft mogelijkheden voor intern parallellisme.

Samenvatting en conclusies

De toepassingen en aanvallen die met symmetrische cryptografie mogelijk zijn, zijn divers en een greep eruit is in dit hoofdstuk behandeld.

Het uitproberen van alle sleutels in een brutekrachtaanval kost doorgaans te veel tijd tijdens de aanval. Hellmans aanval biedt de mogelijkheid om dit werk vrijwel geheel als voorberekening te doen, dus voordat de feitelijke aanval begint. Hiertoe worden tabellen gemaakt met informatie over vele sleutels en wordt er van een slimme zoekmethode gebruik gemaakt. Hoe meer geheugenruimte beschikbaar is, des te sneller kan de aanval uiteindelijk verlopen, en de tabellen zijn voor meerdere aanvallen (op hetzelfde systeem) bruikbaar.

Vervolgens is meervoudige encryptie bekeken; hierbij wordt data meermalen achtereen met hetzelfde algoritme versleuteld. De effectieve sleutellengte wordt hiermee vergroot. Omdat er tegen meervoudige encryptie een speciale aanval mogelijk is, de meet-in-the-middleaanval, is meervoudige encryptie minder efficiënt dan men zou denken. Drievoudige encryptie verdubbelt de effectieve sleutellengte, en verdrievoudiging van de effectieve sleutellengte treedt pas op bij vijfvoudige encryptie.

De werking van het cryptografische algoritme van gsm-telefoons, A5/1, en een effectieve klaretekstaanval op dit algoritme zijn uitgelegd. Voor de aanval moet de aanvaller beschikken over enkele seconden van de onversleutelde conversatie en dit is in de praktijk een zo groot probleem dat de aanval (voorzover bekend) nog niet in praktijk is gebracht. Het bestaan van de aanval wordt wel als een ernstige zwakte beschouwd.

Een moeilijke kwestie bij symmetrische encryptie is het sleutelbeheer: hoe kan men ervoor zorgen dat de communicerende partijen over een gemeenschappelijke sleutel beschikken, zonder dat een aanvaller die te weten komt. In diverse systemen worden sleutels met 'hoofdsleutels' afgeleid uit serienummers. Een ander bekend algoritme is het Diffie-Hellman-protocol.

Opgaven bij hoofdstuk 4

Opgave 4.1 *Hoe kunnen Alice en Bob zich verdedigen tegen een gekozen-klaretekstaanval?*

Opgave 4.2 *Hoeveel reductiefuncties van $\mathbb{B}^{64}$ naar $\mathbb{B}^{56}$ zijn er die bestaan uit een keuze van verschillende bits gevolgd door een inversie van nul of meer bits?*
Hoeveel zijn er van $\mathbb{B}^n$ naar $\mathbb{B}^m$?

Opgave 4.3 *Hoe kun je het geheugengebruik van Hellmans aanval halveren door X_{i0} niet op te hoeven slaan?*

Opgave 4.4 *Schrijf pseudocode voor Hellmans cubic-rootaanval.*

Opgave 4.5 *Bij Hellmans aanval lijkt het geheugen een groter probleem te zijn dan de tijd. Pas de parameters zo aan, dat minder geheugen nodig is en becommentarieer de haalbaarheid van de aanval.*

Opgave 4.6 *Bereken de haalbaarheid van Hellmans aanval bij de oorspronkelijke sleutelgrootte van DES: 112 bits.*

Opgave 4.7 Triple en Quadruple DES. *(Uit het tentamen van maart 1999.)*
Een zekere bank is niet tevreden met het veiligheidsniveau van Triple DES (3DES), omdat het kwetsbaar is voor een Known Plaintext Attack met een complexiteit van (slechts) ongeveer 2^{112} rekenstappen. Men besluit over te gaan op Quadruple DES (4DES), waarin het DES algoritme viervoudig wordt toegepast.

(a) Hoe lang is een sleutel in het 4DES systeem?

(b) Beschrijf (kort!) een Known Plaintext Attack tegen 4DES die ook maar ongeveer 2^{112} stappen kost. Hoeveel bekende (X, Y) paren zijn nodig voor de aanval?
(c) Hoevaak moet je DES itereren opdat een Known Plaintext Attack tenminste 2^{168} rekenstappen kost, en hoe groot is dan de sleutel?

Opgave 4.8 *Bewijs de idempotentie van substitutie- en affiene systemen.*

Opgave 4.9 De Meet-in-the Middle aanval. *(Uit het tentamen van januari 1999.)*
Na 1 ronde van de Meet-in-the-Middle aanval op 2DES heeft Oscar nog twee lijsten: de ene met ongeveer 2^{48} mogelijkheden voor k_1 en de andere met circa 2^{48} mogelijkheden voor k_2.
(a) Wat zijn k_1 en k_2?
(b) In ronde 2 gebruikt Oscar een Plaintext/Ciphertext paar (X, Y). Wat doet Oscar in deze ronde?
(c) Leg uit waarom er na ronde 2 lijsten van lengte ongeveer 2^{32} over zijn.
(d) Hoeveel ronden zijn er nodig?

Opgave 4.10 *Bij de analyse van de meet-in-the-middleaanval is geen rekening gehouden met de mogelijkheid van duplicaten in de lijst M_1 of M_2. Duplicaten zorgen ervoor dat het aantal daadwerkelijk voorkomende strings kleiner is dan de lengte van de lijst, waardoor de eliminatie nog iets sneller verloopt dan is berekend. Kun je de analyse aanpassen om hiermee rekening te houden?*

Opgave 4.11 *Een jonge cryptanalist komt op het idee om een meet-in-the-middleaanval te draaien met minder geheugen: twee tabellen L_1 en L_2 van 2^{48} sleutels slechts. Voor de eerste ronde worden de lijsten niet opgeslagen maar met een loopje geënumereerd. De lijsten M_1 en M_2 worden niet opgeslagen maar door een zoekroutine vervangen: de test $z \in M_i$ is geïmplementeerd met een loop over L_i. Leg uit waarom de jongeman voorlopig niet op promotie hoeft te rekenen.*

Opgave 4.12 *Analyseer een meet-in-the-middleaanval op 3DES, waarbij de aanvaller een tabel gebruikt met één DES-sleutel aan de ene kant en twee DES-sleutels aan de andere kant.*

Opgave 4.13 *Wat is het geheugen- en tijdgebruik van een meet-in-the-middleaanval waarbij eerst L_1 wordt uitgedund, waarna M_1 vanuit de verkorte L_1 wordt berekend?*

Opgave 4.14 *Na hoeveel seconden in een gsm-conversatie wordt een framenummer opnieuw gebruikt?*

Opgave 4.15 *Bekijk een Linear Feedback Shift Register waarbij, zoals gebruikelijk, een van de taps op de meest linkerpositie zit. Bewijs dat de nultoestand (alleen maar 0-en in het register) niet kan voorkomen als opvolger van een niet-nul toestand (minstens een 1).*

Opgave 4.16 *Beschouw een linear feedback shift register L_1 van vier bits en taps op posities 3 en 2. Noem e_i de toestand waarin alleen op positie i een 1 staat, dus $e_0 = 0001$, $e_1 = 0010$, etc., en $S(b)$ die bitstroom die ontstaat bij een start in toestand b.*
Bereken de eerste 15 bits van $S(e_3)$ en $S(e_0)$.
Bereken het begin van $S(1001)$ en constateer dat $S(e_3 \oplus e_0) = S(e_3) \oplus S(e_0)$. Bewijs dat $S(b) = \sum b_i \cdot S(e_i)$.

Opgave 4.17 *Als je een LFSR bekijkt, geven de eerste uitvoerbits direct de begintoestand, maar bij meerdere registers is dat niet zo. Beschouw L_1 met 4 bits en taps op 3 en 2, en L_2 met 5 bits en taps op 4 en 2. Noem e_i voor $i < 4$ de toestand met een 1 op positie i in L_1, en voor $i >= 4$ met een 1 in positie $i-4$ in L_2. De functie S geeft de uitvoer bij een begintoestand.*
Bereken $S(e_1)$ en $S(e_6)$. Kun je de bits van de begintoestand aflezen uit de stroom?
Beredeneer dat geldt: $S(b) = \sum b_i \cdot S(e_i)$.
Veronderstel dat een beginstuk van $S(b)$ bekend is; laat zien hoe je hieruit b, en daarmee ook de rest van $S(b)$, kunt berekenen.

Opgave 4.18 *Bewijs dat Alice en Bob dezelfde sleutel vinden als ze het Diffie-Hellman-protocol gebruiken.*

Hoofdstuk 5

Getallen: eigenschappen

De moderne cryptografische technieken die we in het vervolg van dit boek gaan behandelen zijn vrijwel allemaal gebaseerd op het manipuleren van (grote) getallen. Het dagelijks leven in een hoog ontwikkelde maatschappij is zonder getallen ondenkbaar. Getallen zijn een onmisbaar hulpmiddel voor het uitdrukken van kwantificeerbare eigenschappen van de wereld om ons heen (zie kader 5.1) zoals prijzen, aantallen en afmetingen. Daarnaast worden ze gebruikt ter identificatie, zoals in huisnummers, buslijnen en telefoonnummers.

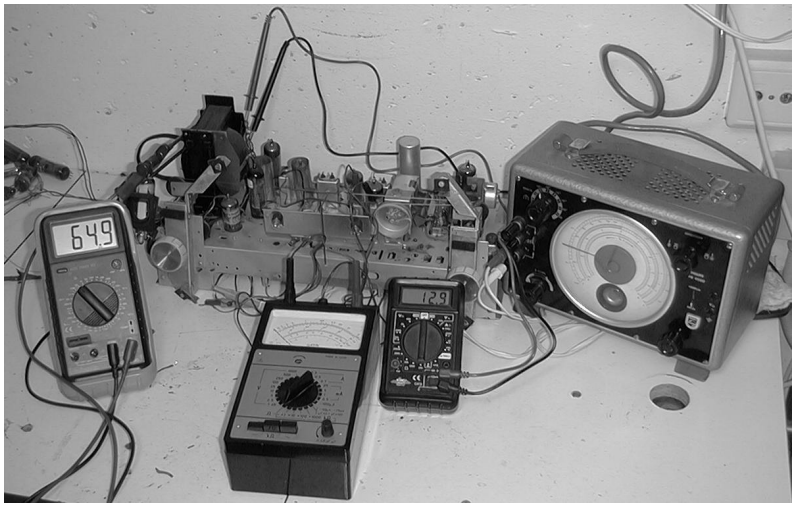
Het belang van getallen in de cryptografie hangt samen met hun zeer precies gedefinieerde manipuleerbaarheid door elementaire operaties als optellen, exponentiatie, worteltrekken et cetera. Niet alleen zijn deze operaties (net als de manipulaties in cryptografische algoritmen als DES) deterministisch en daardoor reproduceerbaar, maar ook voldoen getallen aan *rekenregels* waarmee kan worden afgeleid dat *verschillend uitgevoerde* manipulaties *dezelfde* uitkomst geven. Hiervan hebben we al een voorbeeld gezien, namelijk sleuteluitwisseling volgens het Diffie-Hellman-protocol in sectie 4.4.1.

Een volgende belangrijke kwestie, die rechtstreeks te maken heeft met de bescherming van data, betreft de uitvoerbaarheid van berekeningen. In het genoemde voorbeeld is het niet alleen van belang dat Alice en Bob na afloop over dezelfde sleutel beschikken, maar ook dat een buitenstaander die sleutel niet kan afleiden uit de onderschepte communicatie. Hiervoor is het nodig aan te nemen, dat bepaalde operaties op getallen *niet* effectief kunnen worden uitgevoerd; deze ingewikkelde materie snijden we pas in hoofdstuk 6 aan.

In dit hoofdstuk staan de eigenschappen van de gebruikte getallen zelf centraal, en wordt alleen naar concrete berekeningen ermee gekeken als dat voor de lijn van het betoog nodig is (bijvoorbeeld in het geval van het uitgebreide Euclides-algoritme). De complexiteit van diverse berekeningen wordt in hoofdstuk 6 bekeken, en ook zullen we daar zien hoe de getalsystemen in de cryptografie kunnen worden ingezet.

Het materiaal in sectie 5.1 gaat over de gehele getallen (verzameling $\mathbb{Z}$), deelbaarheid, priemgetallen en grootste gemene delers en zal de meeste lezers grotendeels bekend voorkomen. Sectie 5.2 is wat minder droog van toonzetting dan de omringende secties maar introduceert wel een paar zeer fundamentele ideeën uit de cryptografie, en maakt onder andere duidelijk dat de reële of gehele getallen niet altijd adequaat zijn. In de cryptografie wordt meestal *modulair* gerekend, dat wil zeggen, met resten die overblijven bij deling door een vast getal, de *modulus* m; de verzameling van deze getallen noemen we $\mathbb{Z}_m$ en de eigenschappen ervan zijn zeer sterk afhankelijk van m! De verzameling $\mathbb{Z}_m$ wordt geïntroduceerd in sectie 5.3, en in de volgende twee secties gaan we verder in op twee gevallen die in het bijzonder veel worden gebruikt,

Kader 5.1: Meten is weten?



Het meten van spanningen en stromen geeft een duidelijk beeld van wat er mis is met deze kapotte radio. Techniek en wetenschap floreren door de mogelijkheid eigenschappen van de wereld om ons heen te kwantificeren en de getallen te gebruiken om deze wereld naar ons ideaal te manipuleren.
Het ideaalbeeld is hier in getallen uit te drukken (inclusief de toegestane afwijking), de metingen hebben een bekende nauwkeurigheid, afwijkingen hebben een verklaarbare oorzaak en exact voorspelbare gevolgen. Het determinisme van de technische systemen maakt de heerschappij van Koning Getal in de techniek tot een goedkoop, maar onvermijdelijk en enorm succes. Meten is weten, weten is beheersen, beheersen wordt manipuleren, en kennis is macht.
Het succes van de techniek wekt afgunst in andere gebieden van menselijke activiteit en zo kon Het Getal zijn macht uitbreiden, bijvoorbeeld naar de geneeskunde. Van de patiënt worden diverse parameters bepaald (bloeddruk, cholesterol, hartslag) en klachten worden in verband gebracht met afwijkingen van de gemiddelde waarden. Kritiek op de afgenomen aandacht voor de zieke als persoon wordt weerlegd met cijfers over het succes van de moderne geneeskunde, uitgedrukt in gestegen genezingspercentages en medische efficiency.
Zie ook kader 5.13.

namelijk waar m een priemgetal is (sectie 5.4) danwel het product van twee priemgetallen (sectie 5.5).

De presentatie wordt met diverse rekenvoorbeelden geïllustreerd. Een voorbeeld kan geen bewijs vervangen, maar kan wel het begrip vergroten van wat er eigenlijk bewezen is. Daarom wordt de lezer aangemoedigd om bij de diverse uitspraken meer voorbeelden uit te werken.

5.1 De gehele getallen

De verzameling getallen die begint met

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \ldots$$

staat bekend als de *natuurlijke getallen* en wordt aangeduid met $\mathbb{N}$. (Formeel is natuurlijk een meer precieze definitie nodig, zoiets als "$\mathbb{N}$ is de kleinste verzameling die 0 bevat en gesloten is onder ophoging met 1", maar dat laten we hier achterwege.) De geleerden zijn het erover eens dat er aan deze rij getallen nooit een einde zal komen, daar hoef je dus niet bang voor te zijn.

Voor wie toch denkt aan meer getallen behoefte te hebben, kunnen we de rij ook naar links oneindig ver aanvullen met negatieve varianten:

$$\ldots, -9, -8, -7, -6, -5, -4, -3, -2, -1 \ .$$

De verzameling met de genoemde negatieve en natuurlijke getallen noemen we de *gehele getallen* en wordt aangeduid met $\mathbb{Z}$.

Volgens technici en ingenieurs bestaan er nog getallen die tussen de gehele getallen inzitten. Mocht dit inderdaad al het geval zijn, de betreffende (rationele, irrationele, algebraïsche danwel transcendente) getallen zijn in ieder geval voor de cryptografie niet relevant en we gaan hier dan ook maar niet op in (zie slechts kader 5.6).

Elementaire rekenkundige operaties op gehele getallen zijn optellen, aftrekken, en vermenigvuldigen (die we hier niet definiëren) en delen. Bij delen blijft er zoals bekend naast een *quotiënt* een *rest* over, die positief of nul kan zijn. Iets preciezer, de deling van a door b levert twee getallen, q en r (quotiënt en rest), die voldoen aan: (1) $a = q \cdot b + r$; (2) $0 \leq r < |b|$. Voor het quotiënt schrijven we $a \div b$ en voor de rest $a \,\%\, b$.

Deelbaarheid en priemgetallen. Als de deling van a door b rest 0 oplevert zeggen we dat b een deler is van a, of a een veelvoud van b; notatie: $b|a$. Elk getal deelt 0 ($\forall a : a|0$), 1 deelt elk getal ($\forall a : 1|a$), en elk getal deelt zichzelf ($\forall a : a|a$) en daaraan zien we dat elk positief getal (behalve 1) al minstens twee delers heeft: 1 en zichzelf. Een *priemgetal* is een positief getal met exact twee positieve delers; de getallen met meer delers heten *samengesteld*. Over priemgetallen is heel veel bekend, onder andere dat er oneindig veel van zijn en dat elk getal op één manier is te schrijven als product van priemgetallen.

Stelling 5.1 (Hoofdstelling van de Rekenkunde) *Ieder positief getal n kan worden geschreven als product van priemgetallen: $n = \prod_{i=1}^{k} p_i$ en afgezien van het verwisselen van de volgorde is deze representatie uniek.*

Hoe simpel de definitie van priemgetallen en de formulering van deze stelling ook mogen zijn, het is, gegeven een groot getal n, niet gemakkelijk te bepalen of n een priemgetal is, danwel de priemfactoren van n te vinden; zie opgave 5.3.

Grootste gemene deler en het Euclides-algoritme. Getallen a en b kunnen verschillende delers hebben, maar hebben altijd minstens één deler gemeenschappelijk, omdat 1 elk getal deelt. Met een grotere gemeenschappelijke deler (zeg m) kan de breuk $\frac{a}{b}$ worden vereenvoudigd door teller en noemer door m te delen (de breuk $\frac{a/m}{b/m}$ heeft dezelfde waarde). Teller en noemer van de nieuwe breuk worden zo klein mogelijk door m zo groot mogelijk te nemen en daarom is het relevant om te kijken naar de *grootste gemene deler* van getallen a en b, notatie $\mathrm{ggd}(a, b)$.

Kinderen die op Nederlandse basisscholen breuken leren vereenvoudigen moeten dit zo aanpakken: ontbind teller en noemer in priemgetallen, en zoek naar gemeenschappelijke factoren onder en boven. Voor grote getallen is deze aanpak onuitvoerbaar omdat het vinden van de factoren van een getal te moeilijk is. We behandelen hier het Euclides-algoritme (algoritme 5.2) omdat de uitgebreide versie ervan een belangrijke eigenschap van de ggd bewijst (stelling 5.3).

Stelling 5.2 *Voor $a > 0$ is $\mathrm{ggd}(a, 0) = a$ en als $b > 0$ dan $\mathrm{ggd}(a, b) = \mathrm{ggd}(b, a \,\%\, b)$.*

```
proc Euclides(a, b):
    (* a en b zijn gehele getallen waar a > 0 en a ≥ b ≥ 0 *)
    if b = 0
      then return a
      else d := Euclides(b, a % b)
           return d
```

Algoritme 5.2: Euclides-algoritme voor ggd.

Bewijs. Beschouw eerst $\text{ggd}(a, 0)$; a deelt zowel a als 0 en is dus een gemeenschappelijke deler. Elke gemeenschappelijke deler moet a delen, is dus ten hoogste zo groot als a, waarmee bewezen is dat a de grootste gemene deler is.

Beschouw nu de situatie $a \geq b > 0$. Bedenk eerst, dat een *gemeenschappelijke deler* van twee getallen ook een deler is van een lineaire combinatie van die twee: als $z|x$ en $z|y$ dan ook $z|(x + k \cdot y)$ voor elk geheel getal k.

Deze eigenschap gebruiken we om aan te tonen dat de *gemeenschappelijke delers* van a en b precies dezelfde zijn als de *gemeenschappelijke delers* van b en $a \,\%\, b$. Enerzijds, als $d|a$ en $d|b$, dan ook $d|(a - q \cdot b)$ waar $q = a \div b$, en $a - q \cdot b$ is precies $a \,\%\, b$, dus $d|(a \,\%\, b)$ (en b). Anderzijds, als $d|b$ en $d|(a \,\%\, b)$, dan ook $d|(a \,\%\, b + q \cdot b)$ waar $q = a \div b$, en $a \,\%\, b + q \cdot b$ is precies a, dus $d|a$ (en b).

Hiermee is natuurlijk ook de *grootste* gemeenschappelijke deler van a en b dezelfde als die van b en $a \,\%\, b$. □

Een belangrijke eigenschap, die we later zullen gebruiken om multiplicatieve groepen te karakteriseren, is dat $\text{ggd}(a, b)$ te schrijven is als lineaire combinatie van a en b. Anders gezegd, voor $d = \text{ggd}(a, b)$ bestaan er gehele getallen x en y zodanig dat $d = a \cdot x + b \cdot y$, en de coëfficiënten x en y worden uitgerekend door algoritme 5.3. (Deze coëfficiënten kunnen negatief zijn.)

Stelling 5.3 *Algoritme 5.3 berekent* (d, x, y)*, waarvoor* $d = \text{ggd}(a, b)$ *en* $d = a \cdot x + b \cdot y$.

Bewijs. Het algoritme berekent de waarde d conform het eerdere algoritme.

Wanneer $b = 0$ wordt $d = a$ opgeleverd, en uiteraard is $a = 1 \cdot a + 0 \cdot b$. Voor $b > 0$ levert een recursieve aanroep d op en getallen x' en y' waarvoor $d = x' \cdot b + y' \cdot (a \,\%\, b)$. Omdat $(a \,\%\, b) = a - (a \div b) \cdot b$, is dus

$$d = x' \cdot b + y' \cdot (a \,\%\, b) = x' \cdot b + y' \cdot (a - (a \div b) \cdot b) = y' \cdot a + (x' - y' \cdot (a \div b)) \cdot b \ .$$

```
proc EuclidesPlus(a, b):
    (* a en b zijn gehele getallen waar a ≥ b ≥ 0 *)
    if b = 0
      then return (a, 1, 0)
      else (d, x', y') := EuclidesPlus(b, a % b)
           x := y' ; y := x' − y' · (a ÷ b) ;
           return (d, x, y)
```

Algoritme 5.3: Uitgebreid Euclides-algoritme (delen in $\mathbb{Z}_a$).

□

De grootste gemene deler is het *kleinste* positieve getal dat een lineaire combinatie van a en b is en deelt alle lineaire combinaties (opgave 5.9). Deze karakterisering van de ggd kan ook als alternatieve definitie worden gebruikt.

5.2 Invoerblindering

Met de getallen in $\mathbb{Z}$ kun je heel veel uitdrukken en berekenen, maar voor de berekeningen in de cryptografie zijn ze minder geschikt. Deze sectie illustreert met een fictief familiedrama waarom. Heel veel economische activiteit die tegenwoordig plaatsvindt heeft te maken met het verkopen, opslaan, en bewerken van informatie. Men kan hierbij denken aan weerberichten die tegen een vergoeding worden aangeboden, of het vertalen van teksten, een vorm van informatiebewerking waarvoor het redelijk is iets te betalen.

Wie een brief met gevoelige informatie wil laten vertalen heeft een probleem: gebruikmaken van de diensten van een vertaler vereist dat de betreffende persoon de brief te zien krijgt, en als iets geheim is wil je dat niet. Een tweede probleem is dat het voor de gebruiker van de vertaaldienst niet mogelijk is, de juistheid van het geleverde werk te controleren. Kort gezegd, op het vertaalbureau kan wel een spion werken die de inhoud van de brief doorvertelt, of zelfs een schurk die de brief verandert. Om dit uit te sluiten zijn vertalers altijd gebonden aan beroepscodes en geheimhoudingsplichten: de integriteit van hun werk wordt door gedragscodes en onderling vertrouwen afgedwongen. Het zou voor de opdrachtgever bevredigender zijn wanneer het mogelijk was een brief te laten vertalen zonder dat de vertaler hem kan lezen, en met de mogelijkheid het resultaat zelf te controleren. Bij het vertalen van teksten is dit nog niet mogelijk, maar met cryptografische technieken kan het wel voor computationele diensten.

Rianne (8 jaar) en Jan-Willem (10 jaar) zijn weg van getallen en besluiten elk een rekenmachine te kopen. Jan-Willem koopt een machine van 2 euro die kan optellen, aftrekken, vermenigvuldigen en delen. Rianne spaart haar zakgeld van een maand op en koopt een prachtige machine van 5 euro die, naast allerlei 'handige' functies ook wortels kan uitrekenen; zie kader 5.4.

Jan-Willem is wel lekker goedkoop uit, maar moet voor het trekken van wortels steeds een beroep doen op zijn zus, die echter, begrijpelijk, haar investering wil terugverdienen en per getrokken wortel een kwartje rekent. Rianne leent haar machine niet uit omdat zij dan niet kan controleren hoeveel wortels Jan-Willem ermee uitrekent; ze rekent zelf wortels uit van getallen die Jan-Willem haar aanlevert (met kwartje). Vervelend voor Jan-Willem, want hij wil een vierkante taart bakken en niemand in huis mag weten hoe groot die wordt. Hoe kan Jan-Willem zijn geheim laten bewortelen zonder dat Rianne het te weten komt?

Jan-Willem blindeert zijn geheime x door vermenigvuldiging met een *willekeurig* getal b, de *blinder* genaamd; zie algoritme 5.5. Rianne krijgt van Jan-Willem het product x', maar omdat er *voor elk getal* x een b bestaat zdd $x' = b \cdot x$, weet zij hierdoor nog niets over x. Rianne rekent de wortel van x' uit en stuurt die terug, waarna Jan-Willem deelt door $\sqrt{b}$. Dit laatste lijkt onmogelijk omdat Jan-Willem geen wortels kan trekken. Maar dat Jan-Willem geen *wortel kan trekken uit een willekeurig getal* impliceert niet, dat hij geen *willekeurig getal met wortel kan genereren.* Inderdaad, Jan-Willem kiest niet b willekeurig, maar een k, en berekent b als k^2; nu is b even willekeurig als wanneer Jan-Willem rechtstreeks een willekeurige b had gekozen en Jan-Willem weet de wortel ervan.

Op dit verhaal zijn verschillende reacties mogelijk.

Kader 5.4: De rekenmachines van Jan-Willem en Rianne

De eenvoudigste rekenmachines rekenen alleen met drijvende komma zoals de rechtermachine. Hiermee kun je gehele getallen verwerken (berekeningen op $\mathbb{Z}$) en ook tussenliggende waarden, maar de getallen mogen niet al te groot of al te klein zijn. De linkermachine rekent ook met een macht van 10 erbij waarmee de reële getallen (verzameling $\mathbb{R}$) bruikbaar worden gerepresenteerd. Ook heeft deze de mogelijkheid om rechtstreeks met breuken (verzameling $\mathbb{Q}$) te werken, maar door de beperkte schermbreedte (tien cijfers) kunnen teller en noemer niet te groot worden. Zakrekenmachines waarmee je direct in eindige getalsystemen ($\mathbb{Z}_m$) kunt rekenen zijn er niet; op programmeerbare machientjes kun je dat wel zelf maken.
Een rekenmachine is mooier, en brengt in de winkel meer geld op, naarmate er meer toeters en bellen opzitten. De fabrikanten komen de functiefetisjisten graag tegemoet met bijvoorbeeld een HYP-knop waarmee de hyperbolische functies en hun inversen kunnen worden berekend: weer zes functies erbij! Wie zo'n apparaat koopt drukt een keer op die knop om te kijken wat er gebeurt maar daarna blijft de knop doorgaans ongebruikt.
De mooiste knop op de wetenschappelijke rekenmachine is de INV-knop omdat deze symboliseert dat berekeningen met deze machine altijd kunnen worden teruggedraaid. Als een berekening op de machine kan worden uitgevoerd en een inverse heeft, kan die inverse ook met deze machine worden berekend.

1. **Pedagoog:** *Ach, die arme kinderen, zo jong en dan al zo verpest!* Tja, het is niet anders, ze zullen zich moeten redden in de 21^{ste} eeuw, en daar kunnen ze beter maar vast op voorbereid zijn.
2. **Ingenieur:** *Het komt niet uit, want door afrondfouten is het eindresultaat* $\frac{\sqrt{k^2 \cdot x}}{k}$ *verschillend van* $\sqrt{x}$. De precisie van de rekenmachines zal voor culinaire doelen wel voldoende zijn, maar zoals is opgemerkt gaat het bij getallen in de cryptografie niet om hoeveelheden. Een 'ongeveer' juist antwoord is onbruikbaar, en we hebben daarom getallen nodig

---

1. Jan-Willem kiest een random k, berekent $b = k^2$ en $x' = x \cdot b$.
2. Jan-Willem stuurt x' plus een kwartje naar Rianne.
3. Rianne stuurt $y' = \sqrt{x'}$.
4. Optioneel: Jan-Willem controleert dat $y'^2 = x'$.
5. Jan-Willem berekent $y = y'/k$.

Algoritme 5.5: Basisprotocol voor invoerblindering.

---

waarmee we kunnen rekenen zonder afrondfouten.

3. **Statisticus:** *Omdat er oneindig veel getallen bestaan kan Jan-Willem in stap 1 niet uniform kiezen. De gebruikte kansverdeling, in combinatie met x', geeft Rianne informatie over x.* Dit moeilijk te begrijpen argument is juist, en daarom rekenen we in de cryptografie steeds in eindige getallenverzamelingen.
4. **Numericus:** *Jan-Willem kan met dat rekenmachientje* wel *worteltrekken hoor! Deel $1 + x$ door 2 en .....* In de reële getallen is het inderdaad zo dat je van alle (continue) functies die je kunt uitrekenen, ook de inverse kunt uitrekenen. Het is nog niet zo gemakkelijk om in dit verhaal hard te maken dat bij een ander getallensysteem het berekenen van de inversen *niet mogelijk is.* Hierop komen we terug in sectie 6.2.
5. **Econoom:** *Jan-Willem kan de y' die hij voor een kwartje heeft gekocht door verschillende getallen k delen en zo meerdere wortels voor de prijs van een krijgen.* Dit argument klinkt solide maar klopt niet. Wel is het zo, dat Jan-Willem *zoveel getallen als hij wil* kan krijgen met een wortel, namelijk door willekeurige getallen te gaan kwadrateren. Maar om vanuit y' de wortel van x af te leiden moet worden gedeeld door de k waarmee x' is gemaakt (als $x' = k^2 \cdot x$). Om wortels uit een ander gegeven getal z af te leiden moet Jan-Willem delen door $\sqrt{x'/z}$ en die heeft hij niet.
6. **Rechercheur:** *Als criminelen straks geblindeerd informatie kunnen laten bewerken verliezen we de greep op de drugshandel in Nederland. Het moet dus aan aanbieders van informatiediensten worden verboden, geblindeerde invoeren van hun cliënten te accepteren.* Het klinkt wel erg gemakkelijk, alles meteen streng verbieden wat niet in je kraam te pas komt. Helaas kan Rianne aan het getal x' niet zien of Jan-Willem het geblindeerd heeft of niet, zij kan haar klanten dus op dit punt niet dwarsbomen. Bovendien gaat deze eis voorbij aan de gerechtvaardigde behoefte aan privacy van gebruikers van informatiediensten.

De anekdote reikt enkele punten aan waarop later in dit boek wordt teruggekomen. Ondertussen tekent zich de behoefte af aan een getallensysteem dat de volgende eigenschappen heeft.

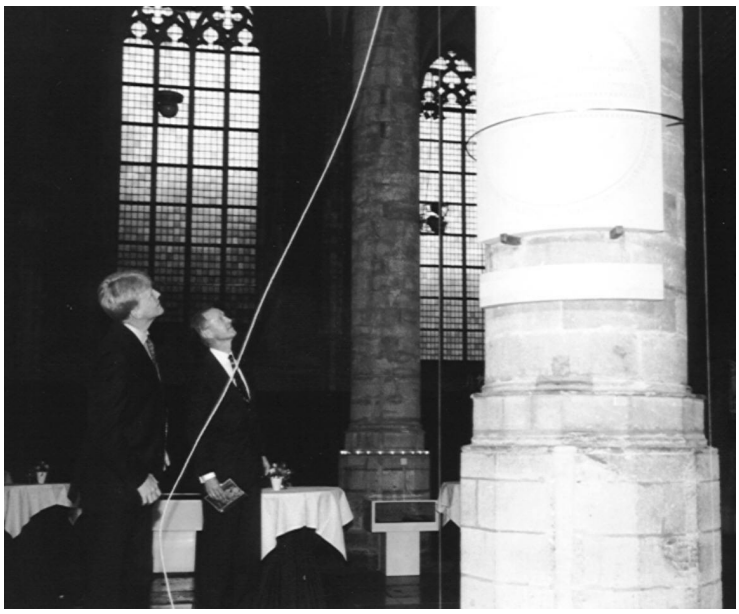
1. Er zijn rekenregels waarmee van verschillend uitgevoerde berekeningen kan worden aangetoond dat ze dezelfde uitkomst hebben.
2. Er bestaan slechts eindig veel getallen zodat er een random getal kan worden gekozen met uniforme kansverdeling.
3. Er kan exact worden gerekend zodat overeenstemming van uitkomst niet door afrondfouten wordt verstoord.
4. Er bestaan efficiënt berekenbare functies waarvan de inverse niet efficiënt berekenbaar is.

5.3 Modulaire aritmetiek

In de cryptografie wordt het meest gebruik gemaakt van *modulaire aritmetiek*, hetgeen wil zeggen dat er wordt gerekend *modulo* een vast getal, de *modulus* genaamd. De verzameling van de gebruikte getallen, de *modulaire groep*, wordt in de volgende subsectie geïntroduceerd, wellicht iets uitgebreider dan voor de meeste lezers noodzakelijk is. Na deze voorzichtige

Kader 5.6: π

Getallen kunnen meer dan hoeveelheden uitdrukken: ze kunnen ook veel enthousiasme losmaken. Er is geen getal dat zo tot de verbeelding spreekt als π, de verhouding tussen doorsnede en omtrek van een cirkel.



Ongeveer vierhonderd jaar geleden rekende Ludoph van Ceulen de waarde van π uit tot 35 decimalen, die op zijn grafsteen werden gebeiteld; de eerste wiskundige publicatie op dit medium. De steen is later zoekgeraakt, maar op 5 juli 2000 werd een replica onthuld door niemand minder dan onze eigen kroonprins.
Tot de bekende blunders rond π hoort het aannemen van een wet waarin de waarde van π werd vastgelegd op 3.125 (1897, Indiana). Maar nog in mei 2000 haalde de tachtigjarige Bussumse uitvinder Charles Doyer (die eerder een motor had uitgevonden die op water liep) alle Nederlandse kranten met de 'ontdekking' dat π geen 206 miljard, maar slechts veertien decimalen telde.
Niet alleen heeft π veel met cirkels te maken, ook in de getaltheorie duikt het vaak op. De kans dat voor random gekozen integers a en b geldt $\text{ggd}(a, b) = 1$ is $6/\pi^2$ bewees Lejeune Dirichlet in 1849 [Knu98, p. 342].

introductie volgt in sectie 5.3.2 een beschouwing over het vermenigvuldigen en delen van modulaire getallen.

5.3.1 Introductie van de modulaire groep $\mathbb{Z}_m$

In deze sectie is m een vast getal ($m \geq 2$); we gaan de gehele getallen in klassen opdelen afhankelijk van hun rest bij deling door m; twee getallen hebben dezelfde rest wanneer hun verschil door m deelbaar is.

Definitie 5.4 *Getallen a en b zijn m-verwant, notatie $a \equiv_m b$, als $m|(a - b)$.*

Verwantschap is een *equivalentie-relatie*, dat wil zeggen, de relatie is *reflexief*: $a \equiv_m a$; *symmetrisch*: $a \equiv_m b \Rightarrow b \equiv_m a$; *transitief*: $a \equiv_m b \wedge b \equiv_m c \Rightarrow a \equiv_m c$. Van equivalentierelaties is bekend dat ze een verzameling verdelen in onderling equivalente elementen, in het algemeen *equivalentieklassen* genaamd, en in dit specifieke geval restklassen:

Definitie 5.5 *De* restklasse *van a (modulo m) is de verzameling $\overline{a} = \{x : x \equiv_m a\}$.*
De verzameling van restklassen heet $\mathbb{Z}_m$.

Kader 5.7: HET REKENEN MET RESTKLASSEN

Dit voorbeeld illustreert hoe de vermenigvuldiging op getallen in $\mathbb{Z}$ een vermenigvuldiging op de restklassen van $\mathbb{Z}_m$ definieert. Wanneer men getallen uit $\overline{5}$ en $\overline{6}$ kiest en vermenigvuldigt, zijn er verschillende uitkomsten in $\mathbb{Z}$ mogelijk. Conform stelling 5.6 liggen die getallen allemaal in dezelfde restklasse (hier $\overline{2}$). We kunnen daarom het product van de *klassen* $\overline{5}$ en $\overline{6}$ definiëren als de *klasse* waarin het product van *willekeurige* getallen uit $\overline{5}$ en $\overline{6}$ valt.

Vermenigvuldiging $\overline{5} \cdot \overline{6}$ in $\mathbb{Z}_7$:

1: $\overline{5} = \{\ldots, -9, -2, 5, 12, 19, \ldots\}$
$\overline{6} = \{\ldots, -8, -1, 6, 13, 20, \ldots\}$
Nu $-9 \cdot 13 = -117 \in \overline{2}$
$5 \cdot 20 = 100 \in \overline{2}$
$(-9) \cdot (-8) = 72 \in \overline{2}$

2: $(5 \cdot 6) \,\%\, 7 = 30 \,\%\, 7 = 2$

Dit gaat (weer conform stelling 5.6) ook op voor optellen en aftrekken. Dit lijkt erg vanzelfsprekend tot men hetzelfde probeert met delen of exponentiëren.

Het aantal restklassen is precies m; voor a en b in dezelfde klasse geldt natuurlijk $\overline{a} = \overline{b}$; zo geldt:

$$\begin{array}{rcl} \overline{-m} = \overline{0} = \overline{m} & = & \{\ldots -2m, -m, 0, m, 2m, 3m, \ldots\} \\ \overline{-m+1} = \overline{1} = \overline{m+1} & = & \{\ldots -2m+1, -m+1, 1, m+1, 2m+1, 3m+1, \ldots\} \\ & \vdots & \\ \overline{-m-1} = \overline{-1} = \overline{m-1} & = & \{\ldots -2m-1, -m-1, -1, m-1, 2m-1, 3m-1, \ldots\} \end{array}$$

Met deze restklassen kunnen we rekenen als met de onderliggende getallen; tenminste, voor zover het de operaties optellen, aftrekken en vermenigvuldigen betreft. Dit komt doordat de klasse van de uitkomst niet afhangt van welk getal uit een bepaalde klasse je neemt; dit wordt geïllustreerd in kader 5.7 en bewezen in de volgende stelling.

Stelling 5.6 *Stel $a \equiv_m a'$ en $b \equiv_m b'$.*
Dan (i) $a + b \equiv_m a' + b'$, (ii) $a - b \equiv_m a' - b'$ en (iii) $a \cdot b \equiv_m a' \cdot b'$

Bewijs. Voor vermenigvuldigen (geval (iii)) gaat het zo: omdat $a \cdot b - a' \cdot b' = (a \cdot b' + a \cdot (b - b')) - a' \cdot b' = (a - a') \cdot b' + a \cdot (b - b')$ volgt $m|(a \cdot b - a' \cdot b')$ uit $m|a - a'$ en $m|b - b'$. □

Definitie 5.7 *Op de verzameling $\mathbb{Z}_m$ zijn optelling, aftrekking en vermenigvuldiging gedefiniëerd door $\overline{a} + \overline{b} = \overline{a + b}$, $\overline{a} - \overline{b} = \overline{a - b}$, $\overline{a} \cdot \overline{b} = \overline{a \cdot b}$.*

In $\mathbb{Z}_m$ gelden dezelfde rekenregels als in $\mathbb{Z}$, zoals $\overline{0} + a = a$, $\overline{1} \cdot a = a$, $a + b = b + a$, $a \cdot b = b \cdot a$, $a \cdot (b + c) = a \cdot b + a \cdot c$, etc. Een dergelijke structuur heet in de wiskunde een *ring*. Over het algemeen laat men het plaatsen van strepen boven getallen achterwege en wij zullen dit ook doen als uit de context blijkt dat elementen van een modulaire groep worden bedoeld.

5.3.2 De multiplicatieve groep $\mathbb{Z}_m^*$

We zagen dat in de ring $\mathbb{Z}_m$ een deling niet kan worden gedefiniëerd zoals dat met optellen en vermenigvuldigen werd gedaan. Sectie 5.2 maakt wel duidelijk dat het inverteren van rekenkundige operaties van belang is, met andere woorden, kan uit $y = a \cdot x$ de waarde

Kader 5.8: Twee voorbeelden van $\mathbb{Z}_m^*$

Voorbeeld 1: $m = 7$.
$\mathbb{Z}_7 = \{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\}$
0 heeft geen inverse
$1 \cdot 1 = 1,\ 2 \cdot 4 = 1,\ 3 \cdot 5 = 1,\ 4 \cdot 2 = 1,\ 5 \cdot 3 = 1,\ 6 \cdot 6 = 1$
$\mathbb{Z}_7^* = \{1,\ 2,\ 3,\ 4,\ 5,\ 6\},\ \phi(7) = 6.$

Voorbeeld 2: $m = 15$.
$\mathbb{Z}_{15} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$
0 heeft geen inverse
$1 \cdot 1 = 1,\ 2 \cdot 8 = 1$
3 heeft geen inverse want $3 \cdot a'$ is altijd een 3-voud
$4 \cdot 4 = 1$
5 heeft geen inverse en 10 ook niet
6, 9 en 12 hebben geen inverse
$7 \cdot 13 = 1,\ 8 \cdot 2 = 1,\ 11 \cdot 11 = 1,\ 13 \cdot 7 = 1,\ 14 \cdot 14 = 1.$
$\mathbb{Z}_{15}^* = \{1,\ 2,\ 4,\ 7,\ 8,\ 11,\ 13,\ 14\},\ \phi(15) = 8$

x teruggevonden worden? De oplossing hiervan is uniek als a komt uit de deelverzameling van getallen in $\mathbb{Z}_m$ die een *multiplicatieve inverse* hebben. Deze deelverzameling is gesloten onder vermenigvuldiging en bevat het neutrale element 1 en wordt daarom zelf ook een groep genoemd.

Definitie 5.8 *De* multiplicatieve groep modulo m*, notatie* $\mathbb{Z}_m^*$*, is gedefinieerd door*

$$\mathbb{Z}_m^* = \{a \in \mathbb{Z}_m | \exists a' : a \cdot a' = 1\}.$$

De Euler-functie $\phi(m)$ *is* $|\mathbb{Z}_m^*|$*; het aantal elementen van* $\mathbb{Z}_m^*$ *heet ook de* orde *van de groep.*

Omdat $0 \cdot a'$ altijd 0 is, heeft 0 in ieder geval geen inverse, en de omvang van $\mathbb{Z}_m^*$ is dus altijd kleiner dan $\mathbb{Z}_m$. Nu is het bij reële getallen zo, dat *elk* getal ongelijk 0 een multiplicatieve inverse heeft; uit kader 5.8 blijkt dat dit voor sommige waarden van m in $\mathbb{Z}_m$ ook geldt, maar voor andere waarden niet.

Stelling 5.9 $a \in \mathbb{Z}_m^*$ *dan en slechts dan als* $\text{ggd}(a, m) = 1$*; in dat geval berekent het uitgebreide Euclides-algoritme de inverse van* a *als* x*.*

Bewijs. Stel $a \in \mathbb{Z}_m^*$, dan is er een $a' \in \mathbb{Z}_m$ waarvoor $a \cdot a' = 1$ in $\mathbb{Z}_m$. Gerekend in $\mathbb{Z}$ betekent dit dat $a \cdot a'$ een veelvoud van m plus 1 is, dus $a \cdot a' = k \cdot m + 1$, dan is $a \cdot a' - k \cdot m = 1$. Omdat ggd($a$, m) zowel a als m deelt, deelt dit getal ook $a \cdot a' - k \cdot m$, dus 1.

Stel nu ggd(a, m) = 1. De aanroep *Euclides*(a, m) geeft nu als antwoord 1, en bovendien x, y waarvoor $1 = a \cdot x + m \cdot y$. In $\mathbb{Z}_m$ geldt dus $a \cdot x = 1$, ofwel, de opgeleverde x is de inverse van a. □

Als m een priemgetal is, geldt ggd(a, m) = 1 voor elke a tussen 0 en m, zodat $\mathbb{Z}_m^* = \mathbb{Z}_m \setminus \{0\}$ en $\phi(m) = m-1$. De inverse van een $a \in \mathbb{Z}_m^*$ noteren we als a^{-1}; deze zit zelf ook in $\mathbb{Z}_m^*$ en deze deelverzameling is ook gesloten onder vermenigvuldigen (opgave 5.18). Voor $k \in \mathbb{Z}$ kunnen

Kader 5.9: Berekening van orden in $\mathbb{Z}_{13}^*$

$$\begin{aligned}
\text{ord}(8)&: 8^1 = 8,\ 8^2 = 12,\ 8^3 = 5,\ 8^4 = 1\\
&\Rightarrow \text{ord}(8) = 4\\
\text{ord}(3)&: 3^1 = 3,\ 3^2 = 9,\ 3^3 = 1\\
&\Rightarrow \text{ord}(3) = 3\\
\text{ord}(2)&: 2^1 = 2,\ 2^2 = 4,\ 2^3 = 8,\ 2^4 = 3,\\
&\quad 2^5 = 6,\ 2^6 = 12,\ 2^7 = 11,\ 2^8 = 9,\\
&\quad 2^9 = 5,\ 2^{10} = 10,\ 2^{11} = 7,\ 2^{12} = 1\\
&\Rightarrow \text{ord}(2) = 12
\end{aligned}$$

we nu de exponentiatie a^k definiëren als het product van k factoren a. Uit een eenvoudig voorbeeld kan blijken dat de exponenten *niet* modulo m mogen worden gerekend; wel modulo $\phi(m)$ zoals is bewezen door Lagrange.

Stelling 5.10 (Lagrange) *Voor $b \in \mathbb{Z}_m^*$ geldt $b^{\phi(m)} = 1$.*

Je kunt deze stelling gemakkelijk bewijzen door met formules en getallen te manipuleren op een manier waar wiskundigen dol op zijn. Bekijk alle $\phi(m)$ elementen van $\mathbb{Z}_m^*$:

$$x_1 \quad x_2 \quad x_3 \quad \ldots \quad x_{\phi(m)}.$$

Omdat $\mathbb{Z}_m^*$ is gesloten onder vermenigvuldiging hebben deze getallen een product $P \in \mathbb{Z}_m^*$. Vermenigvuldig nu alle getallen met b en noem hun product P':

$$b \cdot x_1 \quad b \cdot x_2 \quad b \cdot x_3 \quad \ldots \quad b \cdot x_{\phi(m)}.$$

Tussen P en P' kunnen we twee relaties vinden:

1. Haal de $\phi(m)$ factoren b buiten haakjes en vindt $P' = b^{\phi(m)} \cdot P$.
2. De tweede rij bevat ook alleen getallen uit $\mathbb{Z}_m^*$, en omdat b een inverse heeft zijn al deze getallen verschillend. Dus de tweede rij bevat precies dezelfde getallen als de eerste, alleen in een andere volgorde. Maar dan is ook hun product gelijk: $P' = P$.

Dit bewijst $P = b^{\phi(m)} \cdot P$, en omdat P een inverse heeft impliceert dit $b^{\phi(m)} = 1$.

Een mooi bewijs, maar we kunnen meer inzicht over het gedrag van vermenigvuldiging in $\mathbb{Z}_m$ ontlenen aan een redenering die uitgaat van het begrip *orde* van een element.

Bewijs. Voor elke $b \in \mathbb{Z}_m^*$ is er een positieve k waarvoor $b^k = 1$; schrijf de machten maar eens op:

$$b^0 = 1, \quad b^1 = b, \quad b^2, \quad b^3, \quad b^4, \quad \ldots$$

De getallen zitten allemaal in $\mathbb{Z}_m^*$ en hier zijn er maar eindig veel van. Onvermijdelijk komt er dus een keer een getal opnieuw voor: $b^i = b^j$. Dan is $b^{j-i} = 1$ (zie kader 5.9).

Definitie 5.11 *De* orde *van $b \in \mathbb{Z}_m^*$ is de kleinste positieve k waarvoor $b^k = 1$.*

Kader 5.10: VERDELING VAN $\mathbb{Z}_{13}^*$ IN PARTITIES VAN GROOTTE ord(8)

Begin met een willekeurig getal bv 7:
$7,\ 7 \cdot 8 = 4,\ 4 \cdot 8 = 6,\ 6 \cdot 8 = 9,\ 9 \cdot 8 = 7$
Kies een getal dat nog niet geweest is bv 3:
$3,\ 3 \cdot 8 = 11,\ 11 \cdot 8 = 10,\ 10 \cdot 8 = 2,\ 2 \cdot 8 = 3$
Kies een getal dat nog niet geweest is bv 12:
$12,\ 12 \cdot 8 = 5,\ 5 \cdot 8 = 1,\ 1 \cdot 8 = 8,\ 8 \cdot 8 = 12.$
Alle getallen zijn nu ingedeeld:
$\{7, 4, 6, 9\},\ \{3, 11, 10, 2\},\ \{12, 5, 1, 8\}.$

Omdat de k^{e} macht van b gelijk is aan 1, geldt dit ook voor alle machten die een veelvoud van k zijn, dus $1 = b^k = b^{2 \cdot k} = b^{3 \cdot k} \ldots$. De stelling $b^{\phi(m)} = 1$ volgt dus uit $\text{ord}(b) | \phi(m)$: de *orde van een element deelt de orde van de groep*. Dit wordt bewezen door de elementen van $\mathbb{Z}_m^*$ te partitioneren en wel zo, dat elke partitie precies $\text{ord}(b)$ elementen bevat. Bij een getal x worden precies alle getallen $x \cdot b^i$ ingedeeld, waarmee inderdaad elk getal terechtkomt in een partitie met precies $\text{ord}(b)$ elementen. De gebruikte equivalentierelatie op $\mathbb{Z}_m^*$ is $x \sim y \equiv \exists i : x \cdot b^i = y$; zie kader 5.10 voor een illustratie. Voor het uitwerken van verdere details, zie opgave 5.22. □

Een direct gevolg van de stelling van Lagrange is dat we bij exponentiëren in $\mathbb{Z}_m$ de exponent mogen rekenen in $\mathbb{Z}_{\phi(m)}$. Een ander gevolg is een eigenschap van priemgetallen, bekend als de *kleine stelling van Fermat*.

Corollarium 5.12 (Fermat) *Voor p priem en $a \in \mathbb{Z}_p^*$, is $a^{p-1} = 1$.*

Bewijs. Voor p priem is $\phi(p) = p - 1$ en de gelijkheid volgt uit stelling 5.10. □

In modulaire aritmetiek kan machtsverheffen efficiënt worden berekend (zie sectie 6.1.3). Een inverse voor machtsverheffen kan op twee manieren worden geformuleerd. Met bekend grondtal g maar onbekende exponent x kan men vragen naar de oplossing voor x uit gegeven $y = g^x$, het berekenen van de discrete logaritme. Met onbekend grondtal x maar bekende exponent e kan men vragen naar de oplossing voor x uit gegeven $y = x^e$, het trekken van hogeremachtswortels. Het discrete-logprobleem is het best bestudeerd, en staat bekend als moeilijk, in groepen met priemmodulus (hoofdstuk 15). Worteltrekken is het best bestudeerd, en staat bekend als moeilijk, in groepen met samengestelde modulus. De multiplicatieve structuur van deze groepen zal in de volgende twee secties verder worden bestudeerd.

5.4 Multiplicatieve groep: priemmodulus

Volgens de stelling van Lagrange is in elke groep $\mathbb{Z}_m^*$ en voor elke $b \in \mathbb{Z}_m^*$ de orde van b een deler van de orde van de groep; dit werd in kader 5.9 geïllustreerd. Een zeer interessant speciaal geval treedt op wanneer deze twee orden aan elkaar gelijk zijn, zoals $\text{ord}(2)$ in $\mathbb{Z}_{13}^*$, die gelijk is aan 12 ($\phi(13)$ dus), want in dit geval is *elk* getal uit $\mathbb{Z}_{13}^*$ gelijk aan een macht van 2.

Kader 5.11: Symboliek van getallen in de bijbel

Bijbelschrijvers gebruikten getallen meestal *niet* om hoeveelheden aan te duiden, maar als verwijzing naar andere bijbelse begrippen. Men moet bijvoorbeeld aan Israël denken wanneer wordt gesproken over vijf (aantal boeken van Mozes) of twaalf (de stammen), en de gehele wereld wordt bedoeld met vier (windstreken) of zeven (toen bekende werelddelen).
De spijzigingsverhalen kunnen dan ongeveer zo worden uitgelegd. In Matteüs 14 voedt Jezus een menigte van vijfduizend mannen met vijf broden, waarbij er twaalf manden over blijven: Jezus ziet als zijn opdracht om vanuit de joodse traditie (de vijf boeken van Mozes) Israël te voeden (voor elke stam een mand). In Matteüs 15 zegt hij dat ook expliciet, maar dan ontmoet hij een vrouw die hem zegt dat de honden (aanduiding voor niet-joden) meeëten van broodkruimels die van de tafel vallen. Hier wordt meteen duidelijk gemaakt, dat Jezus zelf het bedoelde brood is. Later voedt Jezus een menigte van vierduizend mensen met zeven broden, waarbij er zeven manden overblijven: nu is er genoeg brood voor de hele wereld.

Definitie 5.13 *Een element $b \in \mathbb{Z}_m^*$ waarvoor geldt* $\text{ord}(b) = \phi(m)$ *heet een* generator *van $\mathbb{Z}_m^*$. Als een groep (zoals $\mathbb{Z}_m^*$) een generator heeft, noemen we deze groep* cyclisch.

Niet elke multiplicatieve groep is cyclisch. Als m een getal is dat door verschillende oneven priemgetallen wordt gedeeld, is $\mathbb{Z}_m^*$ *niet* cyclisch (te bewijzen met de Chinese reststelling, opgave 5.33). Als m een veelvoud van 4 is, is $\mathbb{Z}_m^*$ evenmin cyclisch. Als m een priemgetal is, is $\mathbb{Z}_m^*$ *wel* cyclisch; het voorbeeld van $\mathbb{Z}_{13}^*$ die wordt gegenereerd door 2 generaliseert dus naar alle priemmoduli. Deze eigenschap van multiplicatieve groepen met priemmodulus wordt in sectie 5.4.1 bewezen. In de gevallen waar de groep cyclisch is, zijn meerdere, maar niet alle, elementen een generator; in $\mathbb{Z}_{13}^*$ zijn bijvoorbeeld vier generators: 2, 6, 7 en 11. In sectie 5.4.2 zullen we eigenschappen van $\mathbb{Z}_m^*$ afleiden uit de cycliciteit.

5.4.1 $\mathbb{Z}_p^*$ is cyclisch

Het bewijs van de stelling van Gauss (dat $\mathbb{Z}_p^*$ voor p priem een cyclische groep is) mag worden overgeslagen door hen met niet meer dan een praktisch gerichte belangstelling voor cryptografie. Het is hier opgenomen voor de volledigheid en om de interesse te bevredigen van hen die een dieper begrip van getallen en hun eigenschappen willen verwerven. Voor verdere verdieping raadplege men bijvoorbeeld Shanks [Sha78]. Er worden achtereenvolgens een aantal resultaten bewezen, uitmondend in de hoofdstelling van deze sectie.

Onder *nuldelers* verstaan we in een ring getallen die ongelijk 0 zijn, maar als product 0 hebben. Als voorbeeld: in $\mathbb{Z}_{15}$ zijn 6 en 10 ongelijk 0, hun product is wel 0. Niet toevallig

gebruikt dit voorbeeld een samengestelde modulus (let ook op hoe de priemdelers van de modulus voorkomen in de getallen 6 en 10!); bij een priemmodulus komen geen nuldelers voor.

Lemma 5.14 *Voor p priem heeft $\mathbb{Z}_p$ geen nuldelers, dus $a \cdot b = 0 \Rightarrow a = 0 \vee b = 0$.*

Bewijs. Als $a \cdot b = 0$ in $\mathbb{Z}_p$, dan $p|a \cdot b$ in $\mathbb{Z}$. In $\mathbb{Z}$ geldt dat als een priemgetal een product deelt, dan deelt het minstens één der factoren, dus $p|a$ of $p|b$, waaruit $a = 0$ of $b = 0$ in $\mathbb{Z}_p$ volgt. □

Een *polynoom* is lineaire combinatie van machten van een variabele x, anders gezegd, een functie van de vorm $f(x) = a_n \cdot x^n + \ldots + a_1 \cdot x + a_0$. We zeggen dat de graad van dit polynoom n is als de coëfficiënt a_n ongelijk is aan 0. Op polynomen zijn optelling, aftrekking en vermenigvuldiging gedefinieerd, en er is een deling met rest mogelijk, vergelijkbaar met die op gehele getallen. In het bijzonder is, voor elke constante b, de functie $x - b$ een polynoom van graad 1, en is elk polynoom van graad $n > 0$ te schrijven als $f(x) = f'(x) \cdot (x - b) + c$, waar f' een polynoom van graad $n - 1$ is. Een *wortel* van f is een waarde x_0 voor de vrije variabele x die voldoet aan $f(x_0) = 0$. Polynomen zijn in deze sectie van belang omdat een element van orde e een wortel is van het polynoom $x^e - 1$, en de volgende stelling geeft ons een grens op het aantal van deze wortels.

Stelling 5.15 *Voor p priem, heeft een n-degraads polynoom in $\mathbb{Z}_p$ hoogstens n wortels.*

Bewijs. We gebruiken inductie naar n.

$n = 0$: Een nuldegraads polynoom is een functie van de vorm $f(x) = a_0$, waar $a_0 \neq 0$. Het aantal waarden van x waarvoor $f(x) = 0$ is 0.

$n + 1$: Als f een wortel heeft, zeg b, gaan we f schrijven als $f(x) = f'(x) \cdot (x - b) + c$ met f' van graad n. Omdat $f(b) = 0$ volgt hieruit dat $c = 0$, anders gezegd, het polynoom f is deelbaar door $x - b$ als b een wortel is. Maar we hebben nu gevonden $f(x) = f'(x) \cdot (x - b)$ en, omdat $\mathbb{Z}_p$ geen nuldelers heeft (lemma 5.14), volgt voor een wortel x_0 van f dat $(x_0 - b) = 0$ of $f'(x_0) = 0$. De eerste mogelijkheid geldt voor precies één element (namelijk b) en de tweede voor hoogstens n elementen (inductie). Dus heeft f hoogstens $n + 1$ wortels.

□

Om de administratie van het bewijs van stelling 5.17 rond te krijgen is de volgende rekenkundige relatie nodig. Als voorbeeld: $341 = 11 \cdot 31$, de delers van 341 zijn 1, 11, 31 en 341, en $\phi(1) + \phi(11) + \phi(31) + \phi(341) = 1 + 10 + 30 + 300 = 341$.

Lemma 5.16 *Voor elke $m > 0$ is $\sum_{d:d|m} \phi(d) = m$.*

Bewijs. We gaan de getallen $x \in \{0, \ldots, m - 1\}$ partitioneren aan de hand van $\text{ggd}(x, m)$. Zij d een deler van m en laat $e = \frac{m}{d}$.

Een getal x met $\text{ggd}(x, m) = e$ is in ieder geval een veelvoud van e, dus is een van de d getallen $i \cdot e$ met $0 \leq i < d$ (want $d \cdot e = m$ en doet niet meer mee). Maar niet al deze getallen hebben $\text{ggd}(i \cdot e, m) = e$: voor de berekening van $\text{ggd}(i \cdot e, m)$ kun je de factor e buiten haakjes halen en dus $\text{ggd}(i \cdot e, m) = e \cdot \text{ggd}(i, d)$. Daarom geldt $\text{ggd}(i \cdot e, m) = e \iff \text{ggd}(i, d) = 1$ en we weten al dat het aantal getallen i waarvoor dit geldt $\phi(d)$ is.

Kortom, er zijn precies $\phi(d)$ elementen met $\text{ggd}(x, m) = e$, en omdat elk van de m getallen in precies één van de partities zit is het totaal m: $\sum_{d:\, d|m} \phi(d) = m$. □

Kader 5.12: De cyclische structuur van $\mathbb{Z}_{13}^*$

2 is een generator van $\mathbb{Z}_{13}^*$:

i	$\in \mathbb{Z}_{12}$	0	1	2	3	4	5	6	7	8	9	10	11	12
2^i	$\in \mathbb{Z}_{13}^*$	1	2	4	8	3	6	12	11	9	5	10	7	1

$\phi(12) = 4$ dus $\mathbb{Z}_{13}^*$ heeft 4 generators.
Kies i met $\text{ggd}(i, 12) = 1$ bv $i = 5$.
$2^5 = 6$ is een generator.

x	$\in \mathbb{Z}_{13}^*$	1	2	3	4	5	6	7	8	9	10	11	12
$\text{ind}_2(x)$	$\in \mathbb{Z}_{12}$	0	1	4	2	9	5	11	3	8	10	7	6
$\text{ind}_6(x)$	$\in \mathbb{Z}_{12}$	0	5	8	10	9	1	7	3	4	2	11	6

Stelling 5.17 (Gauss) *Voor een priemgetal p, en $d|p-1$, zijn er precies $\phi(d)$ elementen van orde d in $\mathbb{Z}_p^*$.*

Bewijs. Schrijf $\psi(d)$ voor het aantal elementen van orde d. Elk element in $\mathbb{Z}_p^*$ heeft een orde die een deler is van $p-1$ omdat $p-1$ de orde van $\mathbb{Z}_p^*$ is; dit impliceert dat $\psi(d) = 0$ als d geen deler van $p-1$ is. Een element van orde d is wortel van het polynoom $x^d - 1$.

Stel nu dat a een element van orde d is. De getallen a^i voor $0 \leq i < d$ zijn allemaal verschillend (dat wordt geïmpliceerd door de definitie van $\text{ord}(a)$). Omdat $a^d = 1$ geldt ook $(a^i)^d = 1$, dus deze getallen zijn allemaal een wortel van het polynoom $x^d - 1$. Omdat dit polynoom niet meer dan d wortels heeft (stelling 5.15!) vormen de getallen a^i alle wortels van dit polynoom en dus omvatten zij alle elementen van orde d. Dit bewijst dat $\psi(d) \leq d$.

Nu hebben niet alle genoemde machten van a noodzakelijk orde d: hun orde kan namelijk kleiner zijn dan d en dat is zeker het geval voor die machten $x = a^i$ voor exponenten i met $\text{ggd}(i, e) > 1$. Zeg namelijk $j = \text{ggd}(i, d) > 1$, schrijf $i = s \cdot j$ en $d = t \cdot j$, dan is $x^t = (a^{s \cdot j})^t = (a^{j \cdot t})^s = 1^s = 1$, dus een kleinere macht van x is al gelijk aan 1, waarmee $\text{ord}(x) < d$. Dit betekent dat alle elementen van orde d worden gevonden onder de machten a^i waarvoor $\text{ggd}(i, d) = 1$ en zoals bekend zijn dat er $\phi(d)$. Dit bewijst dat $\psi(d) \leq \phi(d)$.

Omdat elk van de $p-1$ elementen van $\mathbb{Z}_p^*$ een orde heeft (die $p-1$ deelt) is $\sum_{d|p-1} \psi(d) = p-1$. Omdat $\sum_{d|p-1} \phi(d) = p-1$ (lemma 5.16), volgt dat voor geen enkele deler van $p-1$ de ongelijkheid $\psi(d) \leq \phi(d)$ strikt is, ofwel er geldt $\psi(d) = \phi(d)$ voor elke deler d van $p-1$. □

Corollarium 5.18 *Voor p een priemgetal is $\mathbb{Z}_p^*$ cyclisch.*

Bewijs. Wegens de vorige stelling zijn er $\phi(p-1)$ elementen van orde $p-1$. □

5.4.2 Eigenschappen van de cyclische $\mathbb{Z}_p^*$

Veel belangrijke eigenschappen van de getallen in $\mathbb{Z}_p^*$ laten zich begrijpen en bewijzen uit het bestaan van een generator voor deze groep. Is een generator g gekozen, dan is er voor elke $x \in \mathbb{Z}_p^*$ precies één $i \in \mathbb{Z}_{p-1}$ waarvoor $g^i = x$; deze i wordt de *index* van x genoemd, geschreven $\text{ind}(x)$ of $\text{ind}_g(x)$ als g niet uit de context blijkt.

De indexfunctie voldoet aan allerlei rekenregels die vergelijkbaar zijn met de logaritme die uit de reële getallen bekend is:

1. $\text{ind}(x^{-1}) = -\text{ind}(x)$ ofwel $x^{-1} = g^{-\text{ind}(x)}$;
2. $\text{ind}(x \cdot y) = \text{ind}(x) + \text{ind}(y)$ ofwel $x \cdot y = g^{\text{ind}(x)+\text{ind}(y)}$;
3. $\text{ind}(x^k) = k \cdot \text{ind}(x)$ ofwel $x^k = g^{k \cdot \text{ind}(x)}$.

Net zoals bij rekenen in reële getallen het grondtal van de logaritme kan worden gevarieerd, gelden deze regels onafhankelijk van de gekozen generator. De indexen voor verschillende generatoren kunnen op dezelfde manier in elkaar worden omgerekend:

4. $\text{ind}(g_1^i)^j = i \cdot j$ ofwel $\text{ind}_{g_1}(x) = \text{ind}_{g_1}(g_2) \cdot \text{ind}_{g_2}(x)$.

Vanwege deze overeenkomst met de bekende logaritme wordt de index ook wel de *discrete logaritme* genoemd. Kader 5.12 toont de elementen van $\mathbb{Z}_{13}^*$ als gegenereerd door 2 en met hun index voor de generatoren 2 en 6.

Men zou kunnen denken dat de index een handig hulpmiddel kan zijn voor het uitvoeren van berekeningen in $\mathbb{Z}_p^*$. Dit is echter niet zo, omdat er geen efficiënte manier bekend is om van een zeker getal de index te berekenen. Het gebruik van de index is daarom vooralsnog beperkt tot het begrijpen en bewijzen van eigenschappen van $\mathbb{Z}_p^*$. Omdat $g^i \cdot g^j = g^{i+j}$, gedragen de getallen uit $\mathbb{Z}_p^*$ zich onder vermenigvuldigen, respectievelijk machtsverheffen precies zo als de getallen uit $\mathbb{Z}_{p-1}$ onder optellen, respectievelijk vermenigvuldigen. Dit gebruiken we hier voor het bewijs van de volgende twee stellingen.

Stelling 5.19 (Kwadraatstelling voor priemmodulus) *Als p een oneven priemgetal is, zijn precies de helft van de getallen uit $\mathbb{Z}_p^*$ een kwadraat, en die getallen zijn het kwadraat van precies twee getallen.*

Bewijs. Een getal g^i is het kwadraat van g^j in $\mathbb{Z}_p^*$ precies dan en slechts dan als $i = j + j$ in $\mathbb{Z}_{p-1}$. Omdat $p-1$ een even modulus is, zijn in $\mathbb{Z}_{p-1}$ precies de helft van de getallen even, en als $i = j + j$ dan is ook $i = j' + j'$ voor $j' = j + \frac{p-1}{2}$. Er zijn dus twee getallen waarvan i het dubbele is. □

De twee getallen met hetzelfde kwadraat zijn altijd elkaars tegengestelde: als $a = b^2$ dan ook $a = (-b)^2$. De kwadraten in $\mathbb{Z}_p^*$ zijn dus precies de getallen met *even index* en dit is onafhankelijk van de gekozen generator. Dit impliceert dat *de pariteit van de index* niet afhangt van de gekozen generator, hetgeen in kader 5.12 wordt geïllustreerd.

Omdat de index van een getal niet gemakkelijk te berekenen is, is het ook niet zo gemakkelijk om kwadraten (getallen met even index) te herkennen; gelukkig heeft Euler uitgevonden hoe dit kan zonder de index te kennen.

Stelling 5.20 (Euler-criterium) *Voor p een oneven priemgetal is $x \in \mathbb{Z}_p^*$ een kwadraat dan en slechts dan als $x^{\frac{p-1}{2}} = 1$.*

Bewijs. Als $x = g^i$ voor een even getal $i = 2i'$, dan is $x^{\frac{p-1}{2}} = (g^{2i'})^{\frac{p-1}{2}} = g^{(p-1)\cdot i'} = g^0 = 1$ omdat de exponent modulo $p-1$ wordt gerekend. (Dat we g, i en i' niet kennen is natuurlijk voor de geldigheid van deze afleiding niet van belang.)

Als $x = g^i$ voor een oneven getal $i = 2i' + 1$, dan is $x^{\frac{p-1}{2}} = (g^{2i'+1})^{\frac{p-1}{2}} = g^{(p-1)\cdot i' + \frac{p-1}{2}} = g^{\frac{p-1}{2}} \neq 1$. □

In het tweede geval is de uitkomst altijd -1 (opgave 5.26).

5.5 Modulaire ring: samengestelde modulus

In deze sectie bekijken we het rekenen met een samengestelde modulus. De resultaten zijn gemakkelijk te generaliseren naar moduli met meerdere priemfactoren; de Chinese reststelling is geldig wanneer de getallen p en q geen deler gemeenschappelijk hebben (dus $\text{ggd}(p, q) = 1$). Met het oog op de eenvoud van presentatie gaan we uit van een modulus m die het product is van twee oneven en verschillende priemgetallen p en q. Zulke moduli worden gebruikt in het RSA-cryptosysteem, en inderdaad geldt bij verschillende priemgetallen altijd de voorwaarde voor de stelling: $\text{ggd}(p, q) = 1$.

In sectie 5.5.1 wordt de structuur van de ring $\mathbb{Z}_m$ uitgelegd. Getallen in $\mathbb{Z}_m$ kunnen een-op-een worden vereenzelvigd met *paren* getallen uit $\mathbb{Z}_p \times \mathbb{Z}_q$, waarbij alle bewerkingen puntsgewijs kunnen worden uitgevoerd. In het bijzonder geldt $a \equiv_m b \iff (a \equiv_p b \wedge a \equiv_q b)$. Sectie 5.5.2 behandelt enkele consequenties van deze structuur.

5.5.1 $\mathbb{Z}_m$ is een productring

We kennen inmiddels $\mathbb{Z}_m$ als een verzameling met m getallen; de verzameling $\mathbb{Z}_p \times \mathbb{Z}_q \equiv \{(a_1, a_2) : a_1 \in \mathbb{Z}_p, \quad a_2 \in \mathbb{Z}_q\}$ bevat precies $p \cdot q = m$ paren. De volgende functie beeldt een restklasse modulo m af op het paar van restklassen modulo p, resp. q:

Definitie 5.21 *Laat $f : \mathbb{Z}_m \to \mathbb{Z}_p \times \mathbb{Z}_q$ gedefinieerd zijn door $f(a) = (a \,\%\, p, a \,\%\, q)$.*

Omdat a een restklasse, dus een verzameling gehele getallen is, moet worden aangetoond dat het beeld van f niet afhangt van de representant die uit a is gekozen. Gelukkig geldt voor $x, y \in a$ dat $m|x - y$ en omdat p en q delers van m zijn impliceert dit $x \,\%\, p = y \,\%\, p$ en $x \,\%\, q = y \,\%\, q$, zodat f goed gedefinieerd is.

Het aantal elementen van $\mathbb{Z}_m$ is gelijk aan dat van $\mathbb{Z}_p \times \mathbb{Z}_q$ en daarom zijn de volgende drie beweringen equivalent:

1. f is surjectief (voor elk paar (a_1, a_2) is er minstens een a met $f(a) = (a_1, a_2)$);
2. f is injectief (voor elk paar (a_1, a_2) is er hoogstens een a met $f(a) = (a_1, a_2)$);
3. f heeft een inverse.

De geldigheid van de beweringen en de constructie van de inverse van f staan samen bekend als de Chinese reststelling.

Stelling 5.22 (Chinese reststelling) *Voor elk paar $(a_1, a_2) \in \mathbb{Z}_p \times \mathbb{Z}_q$ is er precies één $a \in \mathbb{Z}_m$ met $f(a) = (a_1, a_2)$.*

Bewijs. Omdat $\text{ggd}(p, q) = 1$ heeft p een multiplicatieve inverse modulo q, dat wil zeggen, er is een $y_p \in \mathbb{Z}$ waarvoor $p \cdot y_p \,\%\, q = 1$; schrijf $W_q = p \cdot y_p$ in $\mathbb{Z}_m$. Het getal W_q heeft bij deling door p rest 0 en bij deling door q rest 1, er geldt dus $f(W_q) = (0, 1)$. Evenzo is er een getal W_p te bepalen dat bij deling door p rest 1 heeft en bij deling door q rest 0, waarvoor dus geldt $f(W_p) = (1, 0)$.

Zij nu het paar (a_1, a_2) gegeven en neem $a = a_1 \cdot W_p + a_2 \cdot W_q$. Nu geldt modulo p dat $a = a_1 \cdot W_p + a_2 \cdot W_q = a_1 \cdot 1 + a_2 \cdot 0 = a_1$ en modulo q dat $a = a_1 \cdot W_p + a_2 \cdot W_q = a_1 \cdot 0 + a_2 \cdot 1 = a_2$. Dus $f(a) = (a_1, a_2)$.

Omdat zo'n getal a voor elk paar bestaat en er evenveel getallen in $\mathbb{Z}_m$ als paren in $\mathbb{Z}_p \times \mathbb{Z}_q$ zijn, is het getal uniek. □

Schrijven we nu kortheidshalve a_p en a_q voor $a \,\%\, p$ en $a \,\%\, q$, dan vertalen rekenoperaties in $\mathbb{Z}_m$ zich als volgt naar $\mathbb{Z}_p$ en $\mathbb{Z}_q$:

1. $f(a+b) = (a_p + b_p,\ a_q + b_q)$;
2. $f(a \cdot b) = (a_p \cdot b_p,\ a_q \cdot b_q)$;
3. $f(a^{-1}) = (a_p^{-1},\ a_q^{-1})$;
4. $f(a^k) = (a_p^k,\ a_q^k)$, en de exponenten kunnen per component verder worden gereduceerd: $(a_p^{k \,\%\, \phi(p)},\ a_q^{k \,\%\, \phi(q)})$;
5. et cetera.

Kortom, $\mathbb{Z}_m$ heeft de structuur van het cartesisch product $\mathbb{Z}_p \times \mathbb{Z}_q$ waarin operaties puntsgewijs worden uitgevoerd. Om deze reden duiden we $\mathbb{Z}_m$ aan als een productring.

5.5.2 Eigenschappen van de productring $\mathbb{Z}_m$

De structuurstelling voor $\mathbb{Z}_p^*$ met p priem (corollarium 5.18) wordt niet voor praktisch rekenwerk gebruikt; dit is niet mogelijk omdat de index-functie niet efficiënt berekenbaar is. De structuurstelling voor $\mathbb{Z}_m$ met $m = p \cdot q$ kan wel praktisch worden aangewend omdat zowel de functie f als zijn inverse gemakkelijk te berekenen zijn (mits de factoren van m bekend zijn).

Een voor de cryptografie belangrijk gevolg van de productstructuur van $\mathbb{Z}_m$ is de volgende karakterisering van de kwadraten in $\mathbb{Z}_m$.

Stelling 5.23 (Kwadraatstelling, samengestelde modulus) *Voor m het product van twee oneven priemgetallen is precies een kwart van de getallen uit $\mathbb{Z}_m^*$ een kwadraat, en elk daarvan is het kwadraat van precies vier getallen.*

Bewijs. Zij a een kwadraat, namelijk $a = b^2$. Beschouw de vergelijking $x^2 = a$ in $\mathbb{Z}_m$:

$$\begin{array}{rcl} x^2 = a & \iff & \text{[schrijf in componenten]} \\ & & (x_p^2,\ x_q^2) = (a_p,\ a_q) \\ & \iff & x_p^2 = b_p^2 \wedge x_q^2 = b_q^2 \\ & \iff & \text{[in } \mathbb{Z}_p \text{ en } \mathbb{Z}_q \text{ zijn dit de twee oplossingen]} \\ & & (x_p = b_p \vee x_p = -b_p) \wedge (x_q = b_q \vee x_q = -b_q) \\ & \iff & \text{[uitschrijven]} \\ & & (x_p,\ x_q) \in \{(b_p,\ b_q),\ (-b_p,\ -b_q),\ (b_p,\ -b_q),\ (-b_p,\ b_q)\} \end{array}$$

De eerste twee paren zijn $f(b)$ en $f(-b)$; de andere twee paren zijn ook elkaars tegengestelde, zeg $f(c)$ en $f(-c)$.

Omdat elk getal precies één kwadraat heeft, maar elk kwadraat van precies vier getallen het kwadraat is, is precies een kwart van de getallen een kwadraat. □

De vier getallen met hetzelfde kwadraat vallen uiteen in twee paren die onderling elkaars tegengestelde zijn: b en $-b$, respectievelijk c en $-c$. De som van twee (verschillende) getallen met hetzelfde kwadraat is geen element van $\mathbb{Z}_m^*$ want

1. $f(b + (-b)) = (b_p + (-b_p),\ b_q + (-b_q)) = (0,\ 0)$, inderdaad is $b + (-b)$ steeds 0 en heeft geen multiplicatieve inverse.
2. $f(b + c) = (b_p + b_p,\ b_q + -(b_q)) = (2b_p,\ 0)$ dus $b + c$ heeft geen inverse modulo q.

Kader 5.13: De macht van het getal

In navolging van het succes van kwantitatieve methoden in techniek en geneeskunde (kader 5.1) is men ook in politiek en bestuur steeds meer van getallen gebruik gaan maken. Gevolg is dat aspecten van een zaak invloedrijker worden naarmate ze nauwkeuriger in getallen zijn uit te drukken.
Zo werd milieubelasting door Schiphol gereduceerd tot geluidsoverlast omdat die het kwantificeerbaarheidsideaal het dichtst benadert. De schaalvergroting en specialisering in de veeteelt hebben tot hogere productie (kwantificeerbaar) geleid, maar we hebben de aandacht voor de waarde en waardigheid van het dier (niet-kwantificeerbaar) uit het oog verloren. Ook op andere gebieden worden, volgens Van den Beukel [Beu98], fundamentele waarden ondergeschikt aan de pragmatiek omdat moraal niet in cijfers is uit te drukken.

een bijdrage levert aan de veiligheid op straat. Op basis van evaluatie van de cijfers van uitgaansgeweld en door het meten van veiligheidsgevoelens kan bepaald worden of het experiment succesvol is en wordt verlengd of uitgebreid.

Maar er kleven meer bezwaren aan deze opkomende 'numerocratie'. De Utrechtse politie (folder) zal over zeer nauwkeurige 'veiligheidsgevoelmeters' beschikken, vaker is het controleren van de herkomst en de juistheid van cijfermateriaal heel moeilijk, waardoor instituten als RIVM en CBS onbedoeld invloedrijk worden. Tussen goedgedefinieerde parameters kunnen onverwachte relaties bestaan: het dalen van prijzen voor telefonie leidt tot een stijging van de uitgaven voor telefonie. Veel mensen hebben grote moeite met het interpreteren van getallen: het valt veel lezers niet op wanneer in een krantenartikel een grootheid door onjuiste vertaling een factor 1 000 of 1 000 000 groter is geworden.
Het toenemende belang van kwantitatieve methoden in politiek en bestuur kan zich daarom ontwikkelen tot een bedreiging voor de democratie.

3. $f(b+(-c)) = (b_p + (-b_p),\ b_q + b_q) = (0,\ 2b_q)$ dus $b+c$ heeft geen inverse modulo p.

Dat $b+c$ en $b+(-c)$ geen inverse hebben impliceert dat $\text{ggd}(b+c, m)$, resp. $\text{ggd}(b+(-c),\ m)$ ongelijk 1 zijn: dus gelijk aan een van de priemfactoren van m.

Deze structuur van het viertal getallen met hetzelfde kwadraat is de basis voor vrijwel alle moderne algoritmen voor het vinden van de factoren van m. Deze algoritmen proberen twee getallen x_1 en x_2 in $\mathbb{Z}_m$ te vinden met hetzelfde kwadraat, die echter niet aan elkaar gelijk, noch elkaars tegengestelde zijn. De factoren van m worden berekend als $p = \text{ggd}(x_1 + x_2,\ m)$ (en $q = m \div p$).

Samenvatting en conclusies

In moderne cryptografische methoden wordt veel gerekend en daarom is de studie van getallen en hun eigenschappen voor de cryptografie van belang. De 'gewone' getallensystemen ($\mathbb{Z}$, $\mathbb{Q}$, en $\mathbb{R}$) hebben helaas enkele tekortkomingen wanneer men ze in cryptografische berekeningen wil gebruiken. Voor de public-keycryptografie is het belangrijkste bezwaar, dat alle functies

die kunnen worden uitgerekend, ook weer kunnen worden geïnverteerd. Dit betekent dat afluisteraar Oscar, als hij de ciphertext Y ontvangt en weet hoe die uit de plaintext X is berekend, zelf X kan vinden. Bescherming van X vereist dus dat het encryptiealgoritme niet bekend is, en bij public-keycryptografie is de encryptiesleutel juist wel bekend.

Daarom wordt er in de cryptografie altijd *modulair* gerekend. Hierbij geldt een getal m als modulus, en na elke rekenoperatie wordt van het resultaat de rest bij deling door m genomen. Er zijn op die manier eindig veel getallen, namelijk 0 tot en met $m - 1$. Al zijn de getallen erg groot (m heeft typisch 155 tot 310 cijfers), de berekeningen kunnen exact, dus zonder afronding, worden uitgevoerd.

Bovendien zijn er in de modulaire rekenkunde functies die gemakkelijk zijn uit te rekenen terwijl hun inverse dat niet is. In de cryptografie wordt veel gebruikgemaakt van machtsverheffen. Deze functie is efficiënt te berekenen, maar worteltrekken of de logaritme zijn dat niet. Op deze rekenkundige functies, en hoe daaruit een cryptografisch algoritme is te maken, gaat hoofdstuk 6 verder in.

Soms wordt bij modulaire berekeningen de Chinese reststelling gebruikt. Als de modulus samengesteld is, zeg $m = p \cdot q$, dan kan de berekening apart met modulus p en met modulus q worden uitgevoerd, en de resultaten gecombineerd. Ten eerste is de berekening sneller omdat met kleinere moduli en dus met kortere getallen gerekend wordt. Ten tweede kunnen sommige operaties alleen met priemmodulus worden uitgevoerd.

Opgaven bij hoofdstuk 5

Opgave 5.1 *Jan beweert dat er meer gehele dan natuurlijke getallen zijn. Joke beweert dat er evenveel gehele als natuurlijke getallen zijn. Welke argumenten kunnen ze gebruiken, en wie heeft gelijk?*

Opgave 5.2 *Bewijs dat er, gegeven a, $b \neq 0 \in \mathbb{Z}$, slechts één paar (q, r) van gehele getallen bestaat zodanig dat $a = q \cdot b + r$ en $0 \leq r < |b|$.*

Opgave 5.3 *Zijn de volgende getallen samengesteld en zo ja, wat zijn hun priemfactoren? 87, 323, 1261, 2537, 17741, 32707.*

Opgave 5.4 *Hoeveel delers hebben 32, 27, en 25? Hoeveel delers heeft $32 \cdot 27 \cdot 25$?*

Opgave 5.5 *Er zijn 100 kaarten die aan een kant rood zijn en aan de andere kant blauw. Ze liggen genummerd op tafel, allemaal met rood boven. Er komen 100 omdraaiers langs, die echter in toenemende mate lui zijn: omdraaier 1 draait alle kaarten om, omdraaier 2 draait alle even kaarten om, omdraaier 3 draait de kaarten om waarvan het nummer door drie deelbaar is, en zo draait omdraaier k precies die kaarten om, waarvan het nummer een veelvoud van k is. Hoeveel kaarten liggen na afloop met rood boven?*

Opgave 5.6 *Zij n een positief getal, te schrijven als $n = \prod_{i=1}^{k} p_i^{e_i}$. Hoeveel (positieve) delers heeft n?*

Opgave 5.7 *Geef een iteratieve versie van (Euclides en) EuclidesPlus. Je algoritme berekent een reeks delingen met rest en stopt als rest 0 wordt gevonden. Elk van de gevonden resten wordt direct uitgedrukt als lineaire combinatie van de oorspronkelijke invoergetallen.*

Opgave 5.8 *Los op in gehele getallen x en y (en z):*

1. $38x + 43y = 1$.
2. $24x + 33y = 3$.
3. $15x + 35y = 25$.
4. $14x + 8y = 37$.
5. $12x + 20y + 4z = 23$.
6. $45x + 60y + 21z = 24$.

Opgave 5.9 *Bekijk bij gegeven a en b de verzameling lineaire combinaties van a en b:*

$$L = \{n | \exists x,\, y \in \mathbb{Z} : n = x \cdot a + y \cdot b\}.$$

Neem m als het kleinste positieve element van deze verzameling en bewijs (1) $m = \mathrm{ggd}(a, b)$ en (2) $\forall n \in L : m|n$.

Opgave 5.10 *Alice beschikt (voor zekere modulus m en exponent e) over een mechanisme om e-demachtswortels te trekken modulo m en biedt deze dienst voor geld aan: wie haar een $y \in \mathbb{Z}_m$ plus 10 euro stuurt ontvangt een x waarvoor $x^e = y$. Bob heeft er wel 10 euro voor over om de wortel uit b te weten, maar omdat b geheim is, wil hij niet dat Alice te weten komt wat b is. Beschrijf hoe Bob van de dienstverlening van Alice gebruik kan maken zonder zijn getal te onthullen.*

Opgave 5.11 De Wortelservice. *(Uit het tentamen van oktober 1998.)*
Bob biedt een interessante computationele dienst aan: hij heeft een groot samengesteld getal m op zijn homepage, en tegen inlevering van een getal x (kwadratisch residu in $\mathbb{Z}_m$) en een tientje geeft hij een wortel van x.
a. Kent Bob de factoren van m?
b. Alice wil de wortel van een getal a weten, maar a is geheim. Beschrijf hoe ze van Bob's dienst gebruik kan maken zonder dat Bob te weten komt wat haar geheime getal is.
c. Beschrijf hoe Oscar, met een investering van ongeveer 20 Euro, Bob kan gaan beconcurreren door dezelfde dienst aan te bieden.

Opgave 5.12 *Alice beschikt (voor zeker priemgetal p en generator g) over een mechanisme om logaritmen in $\mathbb{Z}_p^*$ te bepalen en biedt deze dienst voor geld aan: wie haar een $y \in \mathbb{Z}_p^*$ plus 10 euro stuurt ontvangt een x waarvoor $g^x = y$. Bob heeft er wel 10 euro voor over om de logaritme van b te weten, maar omdat b geheim is, wil hij niet dat Alice te weten komt wat b is. Beschrijf hoe Bob van de dienst gebruik kan maken zonder b te onthullen.*

Opgave 5.13 *Bepaal $\mathbb{Z}_{11}^*$ en $\mathbb{Z}_{21}^*$ op de manier van kader 5.8.*

Opgave 5.14 Euler's phi-functie. *(Uit het tentamen van oktober 1996.)*
Wat is $\phi(m)$? Hoeveel is $\phi(24)$ en hoeveel is $\phi(200)$?

Opgave 5.15 *Bepaal $\phi(2^k)$.*

Opgave 5.16 *Bewijs dat de inverse van een getal $a \in \mathbb{Z}_m^*$ uniek is.*

Opgave 5.17 *Bereken $\phi(m)$ voor een macht van een priemgetal, dus $m = p^e$ voor een priemgetal p. (Gebruik stelling 5.9.)*

Opgave 5.18 *Zij* $a,\ b \in \mathbb{Z}_m^*$. *Bewijs* $a^{-1} \in \mathbb{Z}_m^*$ *en* $a \cdot b \in \mathbb{Z}_m^*$.

Opgave 5.19 *Bereken* ord(8) *in* $\mathbb{Z}_{23}^*$.

Opgave 5.20 *Bepaal de orde van 10 in* $\mathbb{Z}_7^*$, *in* $\mathbb{Z}_{13}^*$ *en in* $\mathbb{Z}_{19}^*$.
Wat is de periodelengte van de repeterende breuken $\frac{1}{7}$, $\frac{1}{13}$ *en* $\frac{1}{19}$*?*
Verklaar wat je ziet. Werkt dit alleen voor priemgetallen?

Opgave 5.21 *Bereken* ord(2) *en* ord(3) *in* $\mathbb{Z}_{17}^*$.
Bereken ord(2) *in* $\mathbb{Z}_{97}^*$.

Opgave 5.22 *We bekijken in meer detail het bewijs van de stelling van Lagrange. Zij* m *gegeven, en een vaste* $b \in \mathbb{Z}_m^*$.

1. *Definieer op* $\mathbb{Z}_m^*$ *de relatie* $x \sim y$ *als* $\exists i : y = x.b^i$.
2. *Bewijs dat* $\sim$ *een equivalentierelatie is (reflexief, symmetrisch, transitief).*
3. *Een equivalentierelatie verdeelt een verzameling in niet-overlappende deelverzamelingen (klassen)* $\bar{x} = \{y : x \sim y\}$.
4. *Definieer de orde van* b *als de kleinste positieve* k *waarvoor* $b^k = 1$ *en bewijs dat* $k = |\bar{1}|$.
5. *Elke klasse is even groot. Bewijs dit door te laten zien dat* $f : z \to z.x^{-1}.y$ *een* bijectie *is van* $\bar{x}$ *naar* $\bar{y}$.
6. *Uit het voorgaande volgt dat* $\phi(m)$ *een veelvoud is van de orde van* b.

Opgave 5.23 *Verifieer dat de generatoren van* $\mathbb{Z}_{13}^*$ *zijn: 2, 6, 7 en 11.*
Wat zijn de generatoren van $\mathbb{Z}_{11}^*$ *en* $\mathbb{Z}_{17}^*$*?*

Opgave 5.24 *Is* $\mathbb{Z}_{15}^*$ *cyclisch? Is* $\mathbb{Z}_7^*$ *cyclisch? Is* $\mathbb{Z}_9^*$ *cyclisch? Is* $\mathbb{Z}_{25}^*$ *cyclisch?*

Opgave 5.25 Getaltheorie: Ordes. *(Uit het tentamen van april 1997.)*
In deze opgave is p *een oneven priemgetal.*
(a) Hoe is de orde van $\mathbb{Z}_p^*$ *gedefinieerd en hoeveel bedraagt de orde van* $\mathbb{Z}_p^*$*?*
(b) Wat is de definitie van een generator *van* $\mathbb{Z}_p^*$*?*
(c) Bewijs: als g *een generator is, dan is* $g^{\frac{p-1}{2}} = -1$.

Opgave 5.26 *Bewijs dat in* $\mathbb{Z}_p^*$ *(voor elke generator)* $\text{ind}(1) = 0$ *en* $\text{ind}(-1) = \frac{1}{2}(p-1)$.

Opgave 5.27 Getaltheorie: Kwadraten. *(Uit het tentamen van januari 1999.)*
Bewijs: als p *een oneven priemgetal is, dan is in* $\mathbb{Z}_p^*$ *precies de helft van de getallen een kwadraat.*

Opgave 5.28 *Alice beschikt (voor zekere modulus* m*) over een mechanisme om te bepalen of een getal een kwadraat is en biedt deze dienst voor geld aan: wie haar een getal* y *stuurt plus 5 euro krijgt te horen* of *er een* x *bestaat met* $x^2 = y$*. Bob heeft er wel 5 euro voor over om te weten of* b *een kwadraat is, maar omdat* b *geheim is, wil hij niet dat Alice te weten komt wat* b *is. Beschrijf hoe Bob van de dienstverlening van Alice gebruik kan maken zonder zijn getal te onthullen. Kan het ook zonder dat Alice er achter komt of Bobs getal een kwadraat is of niet?*

Opgave 5.29 *Bepaal voor $m = 35$ de getallen W_5 en W_7 die nodig zijn voor het terugrekenen vanuit $\mathbb{Z}_5 \times \mathbb{Z}_7$. Los op $f(x) = (2, 3)$ (en verifieer de oplossing).*

Opgave 5.30 *Bepaal voor $m = 77 (= 7 \cdot 11)$ de getallen W_7 en W_{11} nodig voor het terugrekenen vanuit $\mathbb{Z}_7 \times \mathbb{Z}_{11}$.*
Welke getallen hebben hetzelfde kwadraat als 36? Verifieer!

Opgave 5.31 Getaltheorie. *(Uit het tentamen van maart 1999.)*
Zij p een priemgetal met de eigenschap dat $\frac{p+1}{4}$ een geheel getal is.
(a) Noem twee priemgetallen met en twee zonder die eigenschap.
(b) Bewijs dat je in $\mathbb{Z}_p^*$ de wortels van een getal x kunt vinden als $y = x^{(p+1)/4}$ en $y' = -x^{(p+1)/4}$.
(c) Waar wordt in de cryptografie van deze worteltrekmethode gebruik gemaakt?

Opgave 5.32 *Beschouw het getal $N = 323$.*
(a) Kwadrateer willekeurige getallen uit het bereik $1 \ldots 161$ tot je twee getallen tegenkomt met hetzelfde kwadraat. Bereken uit de som en het verschil van deze getallen de factoren van N.
(b) Kan het op deze manier gebeuren dat je een paar vindt van de vorm b en $-b$?
(c) Kun je een schatting maken van de tijd die nodig is om op deze manier de factoren van een (veel, vééél groter) getal N te vinden?

Opgave 5.33 *Bewijs dat $\mathbb{Z}_m^*$* niet cyclisch *is als $m = p \cdot q$ (voor p en q oneven priemgetallen).*
Bewijs dat $\mathbb{Z}_m^*$ wel cyclisch *is als $m = 2p$ (voor p oneven priem).*
Bewijs dat $\mathbb{Z}_m^*$ niet cyclisch *is als $m = 2^s \cdot p$ (voor $s > 1$ en p oneven priem).*
(Hint: de Chinese reststelling karakteriseert $\mathbb{Z}_m$ als productgroep. Hoe bereken je de orde van a als $\mathrm{ord}(a_p)$ en $\mathrm{ord}(a_q)$ bekend zijn?)

Hoofdstuk 6

Getallen en public-keycryptografie

In hoofdstuk 5 hebben we kennis gemaakt met de getalsystemen die in de cryptografie worden gebruikt: de modulaire getallen, met hun operaties en eigenschappen. De bruikbaarheid van deze getalsystemen in de cryptografie komt hieruit voort, dat sommige operaties *wel*, en andere *niet* berekenbaar zijn (in praktische zin). Als gevolg hiervan kan Alice een te verzenden bericht verpakken in een rekenkundige puzzel, die voor Oscar niet oplosbaar is maar voor Bob wel omdat hij over extra informatie beschikt. Anders gezegd, hoewel Oscar weet welke berekeningen Alice heeft toegepast op haar bericht, kan hij het bericht niet terugvinden doordat de benodigde rekenkundige operaties niet kunnen worden uitgevoerd. De theoretische onderbouwing van deze onberekenbaarheid, of liever het ontbreken daarvan, wordt in sectie 6.2 besproken.

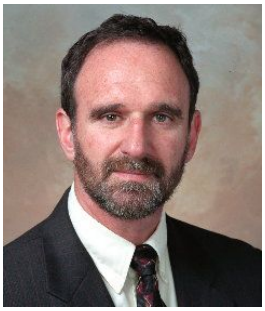
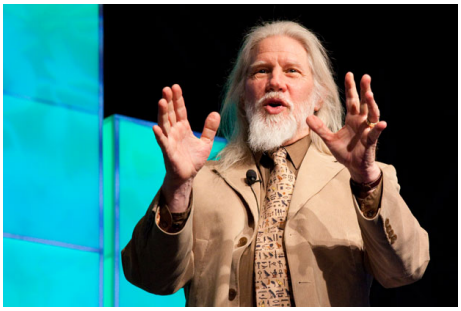
De getallen die Alice gebruikt om haar bericht voor Bob te verpakken, worden de *publieke sleutel* of *public key* genoemd, terwijl Bobs getallen voor het terugvinden van de informatie de *geheime sleutel* of *secret key* worden genoemd. De publieke sleutel, de encryptiesleutel dus, mag openbaar worden gemaakt; alleen de geheime sleutel hoeft door Bob te worden beschermd, maar die hoeft nooit te worden verzonden. Het sleutelbeheer van de resulterende *public-keycryptosystemen* is daarom veel eenvoudiger.

Voor een goed begrip moet wel direct een onderscheid worden gemaakt tussen de begrippen *berekenbaar* en *effectief berekenbaar* (boven aangeduid met de toevoeging "in praktische zin"). In de wiskunde en theoretische informatica noemt men een functie *berekenbaar* wanneer er een algoritme bestaat dat de uitkomst als resultaat geeft. Relevanter voor de cryptografie is *effectieve berekenbaarheid*, die niet alleen het bestaan van een algoritme vereist, maar bovendien grenzen stelt aan de tijd die zo'n algoritme mag nemen om het resultaat te vinden.

Als illustratie van het onderscheid mag integerfactorisering dienen: het vinden van de priemfactoren van een samengesteld getal. Een eenvoudig algoritme voor dit probleem is, het getal achtereenvolgens door oplopende integers te delen, totdat er een rest 0 ontstaat; factorisering is dus berekenbaar. De hoeveelheid rekenstappen die deze aanpak vereist is echter (als het invoergetal uit bijvoorbeeld honderd cijfers bestaat) dermate hoog dat deze aanpak niet als effectief telt. Op de *effectieve* berekenbaarheid van factorisatie gaan we in sectie 6.2.1 in. In de rest van dit hoofdstuk zullen we de term berekenbaar steeds hanteren in de praktische betekenis van effectief berekenbaar.

Dit hoofdstuk gaat eerst wat dieper in op effectieve berekenbaarheid en functies die effectief berekenbaar zijn (sectie 6.1) of juist niet (sectie 6.2). Het materiaal in hoofdstuk 5 en 6 wordt dan met de cryptografie verbonden in sectie 6.3, waar drie getaltheoretische cryptosystemen worden besproken van het public-keytype. De systemen van ElGamal (sectie 6.3.2) en

Kader 6.1: Het ontstaan van public-keycryptografie

Bij asymmetrische, of public-keycryptografie worden er voor versleutelen en ontsleutelen verschillende sleutels gebruikt, waarbij de encryptiesleutel publiek is. Het Diffie-Hellman-protocol (sectie 4.4.1) zorgt dat gebruikers over een sleutel kunnen beschikken zonder dat een privaat kanaal nodig is. Verder formuleerden Diffie en Hellman [DH76] voor een systeem waarbij de aanvaller de encryptie kan uitvoeren deze eisen:

1. Encryptie en decryptie gebruiken *verschillende* sleutels als in $k = (k_p, k_s)$ waar k_s alleen aan de ontvanger bekend is.
2. Encryptie is one-way, dat wil zeggen, niet efficiënt inverteerbaar (maar wel met kennis van k_s).
3. k_s is niet efficiënt te berekenen uit k_p.
4. Decryptie is bestand tegen een gekozen-klaretekstaanval, immers de aanvaller kan willekeurig veel paren $(E(X), X)$ maken met de encryptiefunctie.
5. Het klaretekstalfabet moet groot zijn omdat een brutekrachtaanval over de mogelijke klareteksten mogelijk wordt.

In 2000 ontvingen Diffie (foto rechts) en Hellman (links) voor hun baanbrekende werk de Marconi-onderscheiding.

Rivest, Shamir en Adleman (sectie 6.3.3) worden zeer veel gebruikt, het kwadraatsysteem van Rabin is behandeld vanwege zijn schijnbaar aantrekkelijke eigenschap van bewijsbare veiligheid (sectie 6.3.5). Een beschouwing over de relatie tussen symmetrische en asymmetrische cryptoalgoritmen besluit het hoofdstuk (sectie 6.3.6).

6.1 Berekenbare functies in modulaire groepen

Om de vraag te beantwoorden wanneer een rekenmethode als effectief wordt beschouwd hanteert de informatica meestal het criterium of de berekening *polynomiaal* is in de omvang van de invoer.

Definitie 6.1 *Een algoritme is* polynomiaal *wanneer de rekentijd op een invoer van n bits uit te drukken is als* $O(n^d)$ *voor enig getal d.*
Een probleem is polynomiaal *wanneer er een polynomiaal algoritme voor bestaat.*

De effectiviteit van berekeningen hangt soms af van de wijze van representatie van de invoer. Voor de getaltheoretische berekeningen die volgen zullen we aannemen dat getallen worden gerepresenteerd in *binaire notatie*, dat wil zeggen dat een getal m wordt gerepresenteerd door

een bitrij $m_{k-1} \ldots m_1 m_0$ zodanig dat $m = \sum_{i<k} m_i 2^i$. Voor restklassen modulo m nemen we aan dat ze worden gerepresenteerd door het kleinste niet-negatieve element, dus een getal tussen 0 en $m-1$. Andere representaties, bijvoorbeeld de decimale notatie, zijn hier snel in om te rekenen. Het getal k geeft dus de lengte (in bits) van een getal aan, en een rekenkundige operatie heeft dan een of meer bitrijen van lengte k als invoer.

6.1.1 Additieve en multiplicatieve operaties

In het vervolg zijn a, b en m getallen van k bits. Het vergelijken van getallen in $\mathbb{Z}$ ($a > b$) kan in $O(k)$ tijd door de bits vanaf de meest significante te vergelijken. Optellen en aftrekken kunnen in $O(k)$ tijd worden gedaan door vanaf de minst significante bit te werken net als bij decimale getallen. Vermenigvuldigen kan bit voor bit in $O(k^2)$ tijd en delen kan worden gedaan door een binaire staartdeling die in $O(k^2)$ tijd zowel het quotiënt als de rest oplevert.

Wanneer men optelt modulo m hoeft niet te worden gedeeld om de nieuwe restklasse te bepalen: omdat $a, b < m$ is $a + b < 2m$, zodat hoogstens eenmaal m moet worden afgetrokken. Daarom zijn optellen en aftrekken niet alleen in $\mathbb{Z}$, maar ook in $\mathbb{Z}_m$ lineair in tijd. De vermenigvuldiging van a en b levert een product van ten hoogste $2k$ bits, dat in $O(k^2)$ tijd door de modulus kan worden gedeeld. Daarom is vermenigvuldigen niet alleen in $\mathbb{Z}$, maar ook in $\mathbb{Z}_m$ kwadratisch.

Met meer geavanceerde technieken is snellere implementatie van de multiplicatieve functies mogelijk. De complexiteit van vermenigvuldigen is niet exact bekend, maar wordt in de literatuur vaak aangeduid als $M(k)$ voor getallen van k bits. Delingen kunnen in vermenigvuldigingen worden vertaald, en worden uitgevoerd in $O(M(k))$ tijd; zie Knuth [Knu98, p. 311]. Omdat snelle implementatie van aritmetiek, en de studie van de exacte complexiteit van deze operaties niet het doel van dit boek zijn, zullen we hier verder niet op ingaan, en verder een complexiteit van $O(k^2)$ veronderstellen voor multiplicatieve operaties.

6.1.2 Grootste gemene deler

Voor het berekenen van de grootste gemene deler van a en b kan het Euclides-algoritme (algoritme 5.2) worden gebruikt. De deling die nodig is voor de recursieve aanroep $\mathit{Euclides}(b, a \,\%\, b)$ kost $O(k^2)$ tijd.

De recursiediepte is lineair hetgeen men als volgt kan inzien. Beschouw een executie van *Euclides* waarin d-maal wordt gedeeld en bekijk de rij achtereenvolgende resten

$$x_{d-1} = a \,\%\, b, \quad x_{d-2} = b \,\%\, x_{d-1}, \quad \ldots, x_1 = x_3 \,\%\, x_2, \quad x_0 = x_2 \,\%\, x_1 = 0.$$

De rij wordt dus van achteren geïndexeerd en uit $x_i = x_{i+2} \,\%\, x_{i+1}$ volgt nu dat deze rij ten minste zo hard groeit als de Fibonacci-getallen, dus dat b ten minste zo groot is als het Fibonacci-getal F_d. De Fibonacci-getallen groeien exponentieel, waarmee de lengte van deze rij logaritmisch in b is, dus lineair in k!

Het berekenen van ggd(a, b) kost nu met het Euclides-algoritme $O(k^3)$ tijd. De extra operaties in het uitgebreide Euclides-algoritme kosten, net als de deling, $O(k^2)$ tijd zodat ook het berekenen van een multiplicatieve inverse modulo m in $O(k^3)$ tijd is voltooid.

Het algoritme van Stein [Ste67] (algoritme 6.2) berekent de grootste gemene deler in $O(k^2)$ tijd. Weliswaar is de recursie iets dieper, maar wanneer de getallen binair zijn gerepresenteerd kost elk niveau van recursie slechts lineaire tijd. Net zoals Euclides 2250 jaar eerder deed maakte Stein gebruik van eenvoudige relaties die gelden voor de grootste gemene deler (vergelijk met stelling 5.2).

Stein(a, b):
(* a en b zijn positieve getallen *)
if $a = b$ **then return** a
else if *even*(a) **and** *even*(b) **then return** $2 \cdot \mathit{Stein}(\frac{a}{2}, \frac{b}{2})$
else if *even*(a) **then return** $\mathit{Stein}(\frac{a}{2}, b)$
else if *even*(b) **then return** $\mathit{Stein}(a, \frac{b}{2})$
else (* beide oneven *) **return** $\mathit{Stein}(\min(a, b), |a - b|)$

Algoritme 6.2: Steins binair ggd-algoritme.

Stelling 6.2 *Voor positieve a en b geldt:*

1. *Als $a = b$ dan* $\gcd(a, b) = a$;
2. *Als $a = 2a'$ en $b = 2b'$ dan* $\gcd(a, b) = 2 \cdot \gcd(a', b')$;
3. *Als $a = 2a'$ en b is oneven dan* $\gcd(a, b) = \gcd(a', b)$;
4. *Als a is oneven en $b = 2b'$ dan* $\gcd(a, b) = \gcd(a, b')$;
5. *Als a en b oneven dan* $\gcd(a, b) = \gcd(a, |a - b|)$ *en $a - b$ is even.*

Aan een binair gerepresenteerd getal ziet men direct of het even is ($a_0 = 0$) en zoja, dan verkrijgt men direct de gewenste a' door de laatste bit te laten vallen. De tijd van een aanroep van *Stein* wordt gedomineerd door de eerste test op $a = b$ want hiervoor moeten alle bits worden vergeleken en dat kost lineaire tijd.

De recursiediepte is ook bij Steins algoritme lineair in k. Immers, is ten minste een van de getallen even, dan neemt de lengte van dat getal (of van allebei) met een bit af; beginnend met twee getallen van k bits kan dat dus hoogstens $2k$-maal gebeuren. In het laatste geval is een garantie op het afnemen van de getalgrootte niet zo makkelijk te geven, maar omdat oneven getallen een even verschil hebben komt het nooit tweemaal achtereen voor. Hierdoor is eenvoudig in te zien dat de recursiediepte door $4k$ is begrensd.

De rekentijd van *Stein* is daarom $O(k^2)$.

6.1.3 Exponentiatie

Op de gehele getallen $\mathbb{Z}$ kan machtsverheffen niet als interessante operatie worden beschouwd omdat de resulterende getallen dermate lang zijn dat ze niet effectief te berekenen zijn: de lengte van a^b is b maal de lengte van a! Bedenk dat k ruwweg de logaritme van de getallen is, dus b is zelf exponentieel in k, de lengte van de invoer. Wanneer men modulair rekent met een modulus van k bits is het resultaat ook van lengte k en het is uit te rekenen door herhaald vermenigvuldigen. Echter, beginnen met 1 (of met a) en dan b-maal (of $b - 1$ maal) met a vermenigvuldigen is te kostbaar omdat het (bijna) b vermenigvuldigingen kost.

Algoritme 6.3 berekent de machten a^c voor de achtereenvolgende getallen c waarvan de binaire representatie de prefixen van b zijn. Bijvoorbeeld voor $b = 23$ is de binaire representatie 10111, de prefixen zijn dus 1, 10, 101, 1011, 10111 ofwel (decimaal) 1, 2, 5, 11, 23. De waarde van het binaire getal 1 is 1, en als c de waarde is van de binaire string β, hebben de strings $\beta 0$ en $\beta 1$ de waarden $c + c$, respectievelijk $c + c + 1$. Dit programmafragment berekent daarom de achtereenvolgende waarden van c:

$c := 1$;

```
Macht(a, b) :
  (* Laat 1, b_{l-1}, ..., b_1, b_0 de bits van b zijn *)
  C := a ;
  for i = l − 1 downto 0 do
      C := C ∗ C ;
      if b_i = 1 then C := C ∗ a ;
  return C
```

Algoritme 6.3: Indiaas machtsverheffen.

```
for i = l − 1 downto 0 do
    c := c + c ;
    if b_i = 1 then c := c + 1 ;
```

In plaats van c bouwt het algoritme direct de waarde $C = a^c$ op met de relaties $a^1 = a$, $a^{c+c} = a^c * a^c$ en $a^{c+1} = a^c * a$. Het aantal vermenigvuldigingen is 1 of 2 per bit van de exponent (exclusief de leidende 1). Voor elke 1 in de exponent vermenigvuldigt het algoritme tweemaal en voor elke 0 eenmaal. Een exponent van $l+1$ bits kost dus minstens l en hoogstens $2l$ vermenigvuldigingen. Wanneer de exponent een random getal is, zal het verwachte aantal vermenigvuldigingen rond $1\frac{1}{2}l$ liggen.

Voor k-bits getallen, zowel grondtal als exponent, kan vermenigvuldiging in $O(k^2)$ tijd dus exponentiatie in $\mathbb{Z}_m$ is van orde $O(k^3)$ ofwel kubisch van complexiteit. De berekening verloopt ruwweg sneller als de exponent kleiner is (maar het aantal 1-en in b is ook belangrijk, zo kost de 63ste macht meer vermenigvuldigingen dan de 65ste, 10 tegen 7).

Algoritme 6.3 staat bekend als Indiaas machtsverheffen omdat het al in 200v.C. door een Indiër werd beschreven. Het gebruik van de modulus is in algoritme 6.3 impliciet gelaten: in feite is het algoritme geschikt voor exponentiëren in wat voor getallensysteem dan ook, inclusief bijvoorbeeld elliptische krommen (sectie 15.2.2). Je kunt er zelfs mee vermenigvuldigen door herhaald optellen (kader 6.4) maar dit wordt als archaïsch beschouwd.

Indiaas machtsverheffen kan natuurlijk in veel varianten worden geprogrammeerd, zoals iteratief of recursief, en met het bekijken van de bits van b van links naar rechts of van rechts naar links. Alle varianten hebben gemeen dat kwadrateren wordt gebruikt om het aantal vermenigvuldigingen beperkt te houden, daarom wordt ook wel gesproken van *kwadrateringsmethode*.

Het aantal vermenigvuldigingen bij Indiaas machtsverheffen is niet altijd het minimaal mogelijke. Met Algoritme 6.3 kost a^{15} *zes* vermenigvuldigingen. Een derdemacht kost maar twee, en een vijfdemacht drie vermenigvuldigingen. Dus wanneer je a^{15} berekent als $(a^3)^5$, kan het met vijf vermenigvuldigingen. Het vinden van de kortste reeks vermenigvuldigingen is een onopgelost probleem.

6.1.4 Worteltrekken in $\mathbb{Z}_p^*$

Tweede- en hogeremachtswortels modulo een priemgetal p kunnen worden berekend. Stelling 5.20 toont voor welke getallen $a \in \mathbb{Z}_p^*$ de vergelijking $x^2 = a$ een oplossing heeft; volgens Euler is dit het geval dan en slechts dan als $a^{\frac{p-1}{2}} = 1$. Voor priemgetallen die een viervoud min één zijn is een oplossing van deze vergelijking gemakkelijk uit te rekenen, en wel als een

Kader 6.4: Rekenen in Rusland

Wanneer men de vermenigvuldigingen in algoritme 6.3 door optellingen vervangt, ontstaat een procedure die vermenigvuldigingen uitvoert door herhaalde optelling. Als zodanig werd het vierduizend jaar geleden door de Egyptenaren gebruikt en honderd jaar geleden nog in Rusland. Men komt dan ook de benamingen Egyptisch vermenigvuldigen of Russische boerenmethode tegen. Ook andere eeuwenoude rekentradities worden in Rusland nog hooggehouden. Bijgaande foto is in 1987 gemaakt op de afdeling medaillelintjes van een groot warenhuis in Moskou. Wanneer men meerdere artikelen koopt, wordt het totaalbedrag op de vertrouwde abacus (midden) uitgerekend en daarna op de elektronische kassa aangeslagen.

macht van a:

Lemma 6.3 *Als* $4|(p+1)$ *dan zijn de (tweedemachts) wortels van* a *in* $\mathbb{Z}_p^*$ *te berekenen als* $x_1 = a^{\frac{p+1}{4}}$ *en* $x_2 = -a^{\frac{p+1}{4}}$.

Bewijs. De exponent $\frac{p+1}{4}$ is een geheel getal, zodat de macht rechtstreeks met algoritme 6.3 kan worden berekend. Verder gebruiken we dat als a een kwadraat is, dan $a^{\frac{p-1}{2}} = 1$ (Euler). Dan is

$$\begin{aligned} x_1^2 &= (a^{\frac{p+1}{4}})^2 \\ &= a^{\frac{p+1}{2}} = a^{\frac{p-1}{2}} \cdot a \\ &= 1 \cdot a = a \end{aligned}$$

en evenzo is $x_2^2 = a$. □

De tweedemachtswortel kan dus door exponentiëren worden gevonden in $O(k^3)$ tijd. Voor andere priemgetallen (viervouden plus één) bestaat ook een algoritme om tweedemachtswortels te trekken, maar dit is iets ingewikkelder.

Een wortel van hogere macht kan worden gevonden door de exponent te inverteren modulo $\phi(p)$, immers exponenten in $\mathbb{Z}_p^*$ worden modulo $\phi(p)$ gerekend volgens de stelling van Lagrange (stelling 5.10).

Lemma 6.4 *Zij* e *een getal dat voldoet aan* $\text{ggd}(e, \phi(p)) = 1$. *De vergelijking* $x^e = a$ *in* $\mathbb{Z}_p^*$ *heeft als oplossing* $x = a^d$, *waar* $d = e^{-1}$ *in* $\mathbb{Z}_{\phi(p)}$.

In plaats van een priemgetal p kan in dit lemma ook een samengesteld getal n worden gelezen. Voor het berekenen van de wortel is dan wel kennis van $\phi(n)$ nodig.

6.1.5 Het genereren van een priemgetal

In diverse cryptografische algoritmen is het voor het kiezen van de sleutel nodig om een priemgetal van een bepaalde lengte (en soms met aanvullende eigenschappen) te genereren. Het

priemgetal moet random zijn (vergelijk met het kiezen van een sleutel in een symmetrisch cryptosysteem) om te voorkomen dat het getal kan worden geraden.

Er zijn geen formules of algoritmen bekend die rechtstreeks vanuit random bits een priemgetal kunnen berekenen, toch is het mogelijk om in $O(k^4)$ tijd een priemgetal van k bits te genereren. Zo'n methode is gebaseerd op een test waarmee van een willekeurig getal kan worden bepaald of het priem is of niet; dan worden willekeurig getallen getest en uit de dichtheid van priemgetallen kan dan worden bepaald hoeveel getallen moeten worden bekeken.

Idee: de Fermat-test. De tests waarmee wordt gecontroleerd of een gegeven getal m samengesteld of priem is zijn bijna allemaal uitwerkingen en verfijningen van de kleine stelling van Fermat (stelling 5.12): als m priem is, dan $a^{m-1} = 1$ voor alle $a \in \mathbb{Z}_m^*$. Men rekent dus in $\mathbb{Z}_m$, kiest een willekeurige a en berekent of $a^{m-1} = 1$. Welke conclusie kan worden getrokken uit deze Fermat-test?

- Als de uitkomst *ongelijk is* aan 1 is m samengesteld want bij priem m komt er beslist 1 uit.
- Als de uitkomst 1 is, is geen conclusie mogelijk. Dit komt omdat de stelling van Fermat alleen iets zegt over het geval dat m priem is; wanneer m samengesteld is kan de uitkomst ongelijk zijn aan 1, maar ook gelijk aan 1.

Als voorbeeld van het laatste: 341 is niet priem ($11 \cdot 31$), maar $2^{340} = 1$ modulo 341.

De Fermat-test kan derhalve in sommige gevallen het samengesteld zijn van een getal bewijzen, namelijk als de uitkomst ongelijk 1 is. Het priem-zijn van een getal kan met de test echter niet worden bewezen omdat ook een samengesteld getal kan 'slagen' voor de test (uitkomst 1). Omdat alleen de uitkomst 'samengesteld' zeker is, noemt men deze en soortgelijke tests nooit priemtests, maar altijd samengesteldheidstests. Het verschijnsel dat een samengesteld getal kan slagen voor zo'n test is heel algemeen en de term *pseudopriem* wordt ervoor gebruikt: 341 slaagt voor de Fermat-test met $a = 2$, daarom heet 341 een 2-*Fermat-pseudopriem*.

Omdat pseudopriemgetallen de betrouwbaarheid van de tests verstoren is naar hun voorkomen veel onderzoek gedaan. Het samengestelde getal 341 is 2-Fermat-pseudopriem, maar niet 3-Fermat-pseudopriem: $3^{340} \neq 1$; er zijn (maar?) honderd getallen a tussen 0 en 341 met $a^{340} = 1$. Een voor de hand liggend idee is, de Fermat-test met een aantal waarden van a te herhalen; slaagt het getal meerdere malen achtereen, dan groeit het vertrouwen dat het inderdaad priem is. Een samengesteld getal als 341 valt dan snel door de mand, want er zijn flink wat getallen waarvoor de uitkomst ongelijk 1 is.

Helaas bestaan er ook getallen m die samengesteld zijn, maar waarvoor toch geldt $a^{m-1} = 1$ voor *alle* $a \in \mathbb{Z}_m^*$. Zo'n getal heet een *Carmichael-getal* en het kleinste is 561. Het bestaan van deze Carmichael-getallen is voor de Fermat-test ernstiger dan het bestaan van a-Fermat-pseudopriemgetallen. Immers, een getal als 341 is voor sommige a wel en voor andere a geen Fermat-pseudopriem, en door willekeurig een redelijk groot aantal waarden a te testen is de kans groot dat zo'n getal als samengesteld wordt ontmaskerd. Maar wanneer de invoer 561 is zal het altijd slagen voor de Fermat-test, er is geen kans dat de samengesteldheid wordt aangetoond. (Anders dan de kleine kans dat een getal buiten $\mathbb{Z}_m^*$ wordt gekozen.)

Anders gezegd, de a kan binnen het algoritme random worden gekozen, echter, het getal m is invoer van het algoritme. Hoe zeldzaam de Carmichael-getallen ook zijn (er zijn er slechts 252 kleiner dan 10^8), wanneer de Fermat-test ermee wordt gestart zal het getal altijd slagen met elke a, dus ongeacht hoeveel verschillende waarden er worden geprobeerd!

Vervolg: de Miller-Rabin-test. De test van Miller en Rabin bekijkt iets nauwkeuriger hoe het berekenen van de macht a^{m-1} verloopt wanneer m priem is. Als gevolg daarvan komen MR-pseudopriemgetallen minder vaak voor en bovendien kan worden bewezen dat geen enkel samengesteld getal voor heel veel waarden van a een a-MR-pseudopriemgetal is. Miller en Rabin schrijven het (even) getal $m-1$ als oneven getal maal een tweemacht: $m-1 = t \cdot 2^s$ en berekenen de macht a^{m-1} door eerst a^t te berekenen en dit s-maal te kwadrateren; beschouw de rij

$$w_0 = a^t, \quad w_1 = (w_0)^2, \quad \ldots, \quad w_s = (w_{s-1})^2.$$

De stelling van Fermat leert alleen dat als m priem is, dan $w_s = 1$. Nu is het mogelijk, dat kleinere machten van a al waarde 1 hebben, zelfs a^t kan al gelijk zijn aan 1. Miller en Rabin merkten op dat voor een priemgetal m: als $a^t \neq 1$, dan staat voor de eerste 1 in deze rij de waarde -1. Deze elementaire observatie is maar iets sterker dan die van Fermat: inderdaad verkrijgen we in dit rijtje een 1 door kwadratering, en indien m priem is, heeft behalve 1 alleen -1 het kwadraat 1.

De Miller-Rabin-test verifieert dus deze eigenschap voor een willekeurig gekozen getal a, en m is een *a-Miller-Rabin-pseudopriem* als

1. m is samengesteld;
2. $a^{m-1} = 1$;
3. $a^t = 1$ of voor zekere i is $a^{t \cdot 2^i} \neq 1, -1$ en $a^{t \cdot 2^{i+1}} = 1$.

Miller-Rabin-pseudoprimaliteit impliceert Fermat-pseudoprimaliteit (wegens de tweede eis) en is dus zeldzamer. Er zijn vrijwel geen samengestelde getallen die a-Miller-Rabin-pseudopriem zijn voor überhaupt enige a. De kracht van het concept ligt hierin, dat zelfs bij getallen die MR-pseudopriem zijn, dit maar voor erg weinig waarden van a zo is:

Stelling 6.5 *Als m een samengesteld getal is, zijn er minder dan $m/4$ getallen a in $\mathbb{Z}_m$ waarvoor m een a-Miller-Rabin-pseudopriem is.*

De complete test herhaalt de verificatie een te kiezen aantal, zeg R-maal. Wanneer m priem is slaagt de verificatie alle R malen, onafhankelijk van de keuze van de getallen a. Wanneer m samengesteld is, is de kans dat de verificatie alle R malen slaagt kleiner dan $(\frac{1}{4})^R$, en dit is afhankelijk van de gemaakte keuzen van a. Omdat er voor sommige samengestelde getallen een positieve kans bestaat dat ze door deze testprocedure worden geaccepteerd als priemgetal is de test een *Monte-Carlo-algoritme*. (Een Monte-Carlo-algoritme is een randomiserend algoritme waarvan de looptijd begrensd is, en er is een kans dat het antwoord fout is.)

De dichtheid van priemgetallen. Een geslaagde complete test kost R exponentiaties, dus $O(R \cdot k^3)$ bitoperaties. Het verwachte aantal verificaties bij een samengestelde m is kleiner dan $1\frac{1}{3}$ zodat de complexiteit van een falende test verwacht $O(k^3)$ is.

Over de frequentie van het voorkomen van priemgetallen is bekend dat in de buurt van een groot getal x ongeveer één op de $\ln x$ getallen een priemgetal is. Dit verschijnsel staat bekend als de *priemgetalstelling* en is tweehonderd jaar geleden door Gauss bewezen.

Getallen van k bits lang liggen in de buurt van 2^k en de dichtheid van priemgetallen is dus ongeveer 1 op $\ln(2^k)$ ofwel 1 op $0{,}7 \cdot k$. Dit betekent dat, wanneer men onder willekeurige k-bitsgetallen gaat zoeken naar priemgetallen, men na verwacht $0{,}7 \cdot k$ pogingen succes heeft. Zoeken onder oneven getallen zelfs al na ongeveer $0{,}35 \cdot k$ pogingen. De totale complexiteit wordt $O(k^3 \cdot (k + R))$ operaties, wat als $O(k^4)$ kan worden weergegeven omdat doorgaans k groter is dan R.

Samenvatting en overzicht. De volgende getaltheoretische functies kunnen worden berekend in tijd polynomiaal in de lengte van de getallen.

Functie	Tijd	Opmerkingen
Optellen, aftrekken	$O(k)$	
Vermenigvuldigen, delen	$O(k^2)$	
Grootste gemene deler	$O(k^2)$	
Worteltrekken	$O(k^3)$	mits factorisatie van modulus bekend is
Genereren van priemgetal	$O(k^4)$	Monte Carlo dus kans op fout antwoord

6.2 Niet-berekenbare functies in modulaire groepen

In de cryptografie is niet alleen behoefte aan snelle algoritmen voor getaltheoretische berekeningen; zoals al is aangestipt is er voor de veiligheid van de systemen een even grote behoefte aan functies die niet te berekenen zijn. In deze sectie zullen enkele functies worden behandeld, de discrete log en integer factorisatie (sectie 6.2.1) waarvan wordt aangenomen dat ze niet berekenbaar zijn. De onberekenbaarheid van andere functies kan uit die aannamen worden afgeleid, bijvoorbeeld het worteltrekken met samengestelde modulus (sectie 6.2.2). Deze aanname is helaas voor geen van die functies bewezen, en het is ook niet te verwachten dat dit binnenkort zal worden bewezen; dit wordt gemotiveerd met een beschouwing over de complexiteitsklassen P en NP (sectie 6.2.3).

6.2.1 Discrete log en factorisatie

Er zijn tot op heden geen polynomiale algoritmen bekend voor het berekenen van de discrete log of het factoriseren van getallen. Het discrete-logprobleem luidt: gegeven een priemgetal p, een generator g en een element a van $\mathbb{Z}_p^*$; geef een getal i waarvoor $a = g^i$. Het integerfactoriseringprobleem luidt: gegeven een samengesteld getal m; geef een deler van m (niet 1 of m).

Beide problemen worden al eeuwen door getaltheoretici bestudeerd zonder dat er efficiënte algoritmen zijn gevonden. Bij de stand van de techniek in 2000 konden deze problemen worden berekend tot getallen van ongeveer 512 bits, en elk jaar komt men hiermee weer iets verder. Andere rekenkundige problemen, zoals het genereren van priemgetallen en exponentiëren, kunnen zelfs op kleine computers gemakkelijk voor getallen van duizenden bits lang worden berekend. Intensieve studie van beide getaltheoretische problemen heeft doen vermoeden dat polynomiale algoritmen *niet bestaan*, of op zijn minst, wat voor de cryptografie praktisch gezien even goed is, dat de cryptografische aanvaller niet over zulke algoritmen beschikt.

Aanname 6.6 (Discrete log) *Er bestaat geen efficiënt algoritme voor het discrete-logprobleem. Gegeven goedgekozen p, g, en a, is het niet haalbaar* $\log(a)$ *te vinden.*

Aanname 6.7 (Factorisering) *Er bestaat geen efficiënt algoritme voor integerfactorisering. Gegeven een goedgekozen samengestelde n, is het niet haalbaar de factoren van n te vinden.*

Het tweede deel van de aannamen is nog iets sterker dan het eerste. Niet alleen bestaat er geen efficiënt algoritme dat *alle* instanties kan oplossen, maar de moeilijkheid van het probleem zit in elke goedgekozen instantie.

De Diffie-Hellman-relatie. Een probleem dat nauw aan de berekening van logaritmen verwant is, is de relatie die geldt tussen getallen zoals die bijvoorbeeld in het Diffie-Hellman-protocol voorkomen. De relatie geldt tussen vier getallen, maar omdat een ervan meestal de vaste generator g is, wordt deze doorgaans weggelaten in de notatie.

Definitie 6.8 *De getallen g, a, b en c voldoen aan de Diffie-Hellman-relatie, notatie $DifHel_g(a, b, c)$ als er een x bestaat waarvoor $a = g^x$ en $c = b^x$.*

Doorgaans is b ook een macht van g, zeg g^y, en dan is c tevens gelijk aan a^y ofwel g^{xy}. De relatie zegt dan dat de logaritme van een getal het product van de logaritmen van de andere getallen is.

De gedefinieerde relatie geeft aanleiding tot drie rekenkundige problemen:

1. *Diffie-Hellman-beslissing:* Gegeven a, b en c, geldt $DifHel(a, b, c)$?
2. *Diffie-Hellman-completering:* Gegeven a en b, vind c waarvoor $DifHel(a, b, c)$.
3. *Diffie-Hellman-exponent:* Gegeven a, b en c waarvoor $DifHel(a, b, c)$ geldt, vindt een x waarvoor $a = g^x$ en $c = b^x$.

Van deze problemen wordt ook aangenomen dat ze moeilijk zijn:

Aanname 6.9 (Diffie-Hellman) *Er bestaan geen efficiënte algoritmen voor Diffie-Hellman-beslissing, -completering of -exponent.*

Deze aanname is in het algemeen sterker dan aanname 6.6: het is gemakkelijk in te zien dat de Diffie-Hellman-problemen op te lossen zijn als logaritmen kunnen worden berekend. Alleen voor de Diffie-Hellman-exponent is eenvoudig equivalentie te bewijzen (opgave 6.9). De relatie tussen het Diffie-Hellman-probleem en de discrete logaritme is uitvoerig bestudeerd. In veel gevallen geldt ook het omgekeerde: als Diffie-Hellman-completering berekenbaar is dan kunnen logaritmen berekend worden (Boneh en Lipton, Maurer en Wolf [BL96, MW96]).

Cryptografisch sterk. Een algoritme heet *cryptografisch sterk* wanneer de onmogelijkheid van een aanval erop af te leiden is uit een van deze aannamen. Het zou voor de cryptografie prettig zijn als de aannamen konden worden bewezen. Het is echter in het algemeen heel moeilijk om het niet bestaan van een efficiënt algoritme voor enig probleem te bewijzen. Deze getaltheoretische problemen houden ook verband met andere beruchte onopgeloste problemen uit de informatica, zoals zal worden uitgelegd in sectie 6.2.3.

Nu zijn er voor factoriseren wel algoritmen bekend voor zogenaamde *speciale getallen*. Zo'n algoritme kan bijvoorbeeld een factor p van n vinden in een tijd die een functie is van de grootste deler van $p - 1$ (Pollards methode, zie Knuth [Knu98, p. 413]). Andere methoden hebben een looptijd die afhangt van p zelf, of van factoren van $p + 1$. Om aanvallen met deze methoden te vermijden wordt een priemgetal voor cryptografische toepassingen altijd in meerdere stappen gekozen. Men zoekt bijvoorbeeld eerst een priemgetal p' van zestig cijfers, waarna er naar priemgetallen wordt gezocht onder de getallen $k \cdot p' + 1$, voor k met twintig cijfers. De resulterende p met tachtig cijfers heeft dan in ieder geval een grote factor in $p - 1$.

Het woord 'goedgekozen' in de aannamen 6.6 en 6.7 heeft betrekking op het vermijden van deze gemakkelijke getallen. De aannamen behoeven dus als toelichting, dat er wel degelijk makkelijke getallen bestaan, maar men gaat ervan uit dat deze zwakke getallen zijn te herkennen en daarom ook te vermijden. Priemgetallen die niet met een van de bekende algoritmen voor speciale getallen kunnen worden gevonden heten *sterke priemgetallen*.

6.2.2 Worteltrekken in $\mathbb{Z}_m^*$

Soms wordt van een probleem beweerd dat de onberekenbaarheid ervan is bewezen. In de cryptografie betekent dit meestal niet meer dan dat het niet bestaan van een algoritme ervoor wordt afgeleid uit een van de aannamen in de vorige sectie. Het trekken van vierkantswortels modulo een getal met onbekende factorisatie is hiervan een voorbeeld.

Stelling 6.10 (Aanname 6.7) *Er bestaat geen efficiënt algoritme voor het trekken van (tweedemachts) wortels modulo een samengesteld getal (met onbekende factorisatie).*

Bewijs. Zij m een samengesteld getal en veronderstel dat we beschikken over een algoritme P dat, gegeven een kwadraat $a \in \mathbb{Z}_m^*$, een oplossing voor $x^2 = a$ geeft. Kies een random getal y, bereken $a = y^2$ en geef a als invoer aan P om een x met $x^2 = y^2$ te verkrijgen. Volgens de kwadraatstelling 5.23 zijn er vier getallen met kwadraat a, die paarsgewijs elkaars tegengestelde zijn. Omdat y random is gekozen is er een kans van $\frac{1}{2}$ dat y ongelijk aan x en $-x$ is, en in dat geval is $\text{ggd}(x + y, m)$ een factor van m (zie de toelichting na stelling 5.23). De aanname dat er geen algoritme bestaat om de factoren van m te vinden sluit daarom het bestaan van een worteltrekalgoritme uit. □

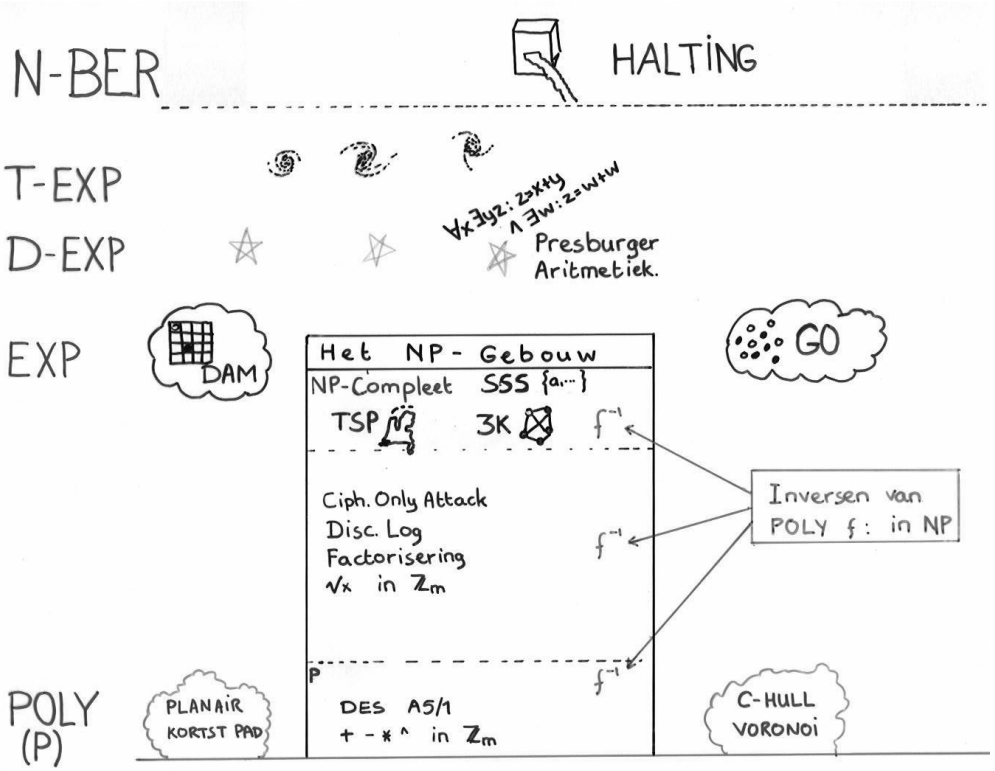
Wanneer de factoren van m, zeg p en q, bekend zijn, is worteltrekken wel mogelijk door toepassen van de Chinese reststelling. Het getal a wordt uitgedeeld naar $\mathbb{Z}_p$ en $\mathbb{Z}_q$, daar wordt de wortel bepaald volgens sectie 6.1.4, en de deelresultaten worden teruggerekend naar $\mathbb{Z}_m$. Worteltrekken in $\mathbb{Z}_m$ is dus bewijsbaar equivalent met het kennen van de factorisatie van m.

6.2.3 P versus NP

Het uitvoeren van een cryptografische aanval kan worden omschreven als een computationeel probleem, en cryptografen zouden graag willen bewijzen dat voor dit probleem geen efficiënte oplossing bestaat. In sectie 6.2.1 werden het discrete-logprobleem en integerfactorisering geformuleerd, in de eerdere hoofdstukken kwam het vinden van een sleutel voor symmetrische algoritmen aan de orde. Hoe verschillend het terugrekenen van een A5/1-sleutel ook schijnt van het berekenen van een discrete logaritme, de genoemde problemen hebben een belangrijke eigenschap gemeen: *een gevonden of geraden oplossing kan in polynomiale tijd worden geverifieerd of verworpen.* Voor de genoemde problemen geconcretiseerd:

1. **Ciphertext-Only Attack (COA):** Het is niet gemakkelijk, gegeven een DES-cryptobericht Y, de gebruikte sleutel te *vinden*; het is wel gemakkelijk, gegeven Y en k, te *verifiëren* of $D_k(Y)$ een leesbare tekst is of niet.
2. **Factorisering (Fact):** Het is niet gemakkelijk, gegeven een samengesteld getal m, een factor van m te *vinden*; het is wel gemakkelijk, gegeven m en p, te *verifiëren* of p een factor van m is.
3. **Discrete Log (DL):** Het is (bij een vaste p en generator g van $\mathbb{Z}_p^*$) niet gemakkelijk, gegeven x de index van x te *vinden*; het is wel gemakkelijk, gegeven x en i, te *verifiëren* of $g^i = x$.

Complexiteit, polynomiaal en exponentieel. De problemen die deze verificatie-eigenschap hebben worden NP-problemen genoemd, voor *Non-deterministisch Polynomiaal.* Iets preciezer gezegd kan men een NP-probleem formuleren als het zoeken naar een oplossing y bij een gegeven invoer x, waarbij de lengte van y polynomiaal moet zijn in de lengte van x, en er is een efficiënte procedure $P(x, y)$ die verifieert of de oplossing y juist is bij x.



Figuur 6.5: Artist impression van het complexiteitslandschap

Figuur 6.5 plaatst een aantal computationele problemen in hun samenhang van complexiteit. Onderin de figuur, op het polynomiale maaiveld, bevinden zich allerlei problemen waarvan de oplossing kan worden gevonden in een rekentijd die polynomiaal is in de lengte van de invoer. Hieronder bevinden zich graafproblemen zoals planariteit en kortste paden, en ook geometrische problemen zoals de convexe omhulling of het Voronoidiagram. Helemaal bovenin de figuur is de plaats voor problemen die onberekenbaar zijn in de wiskundige zin, waarvoor dus geen algoritme bestaat. Het bekendste voorbeeld hiervan is het haltingprobleem: gegeven een programma plus invoer, geeft het programma een eindige of een oneindige berekening?

In het exponentiële wolkendek staan enkele problemen waarvan bekend is dat een oplossing kan worden uitgerekend, maar alleen in rekentijd exponentieel in de lengte van de invoer en niet sneller. Bekendste voorbeelden hiervan zijn bordspelen zoals go en dammen waarvoor moet worden bepaald of de invoer (die wel groter mag zijn dan een normaal speelbord) een winnende stand is voor een van de partijen. Merk op dat van een winnende strategie, als die bestaat en kan worden gegeven, *niet* snel kan worden geverifieerd of dit inderdaad tot winst leidt. Hiervoor moeten namelijk ook weer alle mogelijke tegenzetten worden bekeken, en deze spellen hebben dus niet de polynomiale verificatie-eigenschap.

Er zijn ook problemen die meer dan exponentiële rekentijd vergen. Met Presburger aritmetiek wordt een zekere taal van logische formules over gehele getallen bedoeld; van deze klasse van formules is algoritmisch te bepalen of ze waar of onwaar zijn. De rekentijd is dubbelexponentieel: de beslissing kost 2^{2^n} tijd voor een formule met n symbolen.

Cryptografie en de klasse NP. Omdat cryptografische aanvallen de verificatie-eigenschap hebben gaat de aandacht van cryptografen niet zo hoog, maar slechts naar de klasse van NP-problemen. Van sommige problemen is bekend dat ze efficiënt kunnen worden berekend, bijvoorbeeld encryptie en decryptie in DES (met sleutel!), en de aritmetische functies uit sectie 6.1. Van andere problemen is bekend dat ze tot de moeilijkste uit de NP-klasse behoren; deze problemen noemt men *NP-Compleet* (NPC). De volgende problemen zijn bekende NPC-problemen en een uitgebreidere opsomming is te vinden in het klassieke werk over NP-compleetheid van Garey en Johnson [GJ79]:

1. **Traveling SalesPerson (TSP):** Gegeven een gewogen graaf G en een getal K, bestaat er een pad door G dat langs alle punten komt en lengte $\leq K$ heeft?
2. **Subset Sum (SSS):** Gegeven een verzameling getallen $a_1, \ldots, a_n$ en een getal S, bestaat er een deelverzameling van de getallen met som precies S?
3. **3-Kleuring (3K):** Gegeven een graaf G, kunnen de knopen met drie kleuren worden gekleurd, zodanig dat aangrenzende knopen overal verschillende kleuren hebben?

NP-problemen hebben geen hogere dan een exponentiële complexiteit, omdat een oplossing altijd met brute kracht kan worden gevonden. De lengte van de gezochte string y is immers begrensd, alle kandidaten kunnen dus in exponentiële tijd worden opgesomd en geverifieerd. Om deze reden is het NP-gebouw in figuur 6.5 niet hoger dan de wolken getekend.

Nu ligt het voor de hand, dat in het algemeen het *vinden* van een string y wel moeilijker kan zijn dan het *verifiëren* ervan. Anders gezegd: dat problemen bestaan die wel polynomiaal verifieerbaar zijn maar niet in P (de polynomiaal oplosbare problemen) zitten. Er geldt dan $\mathrm{P} \neq \mathrm{NP}$, en omdat dit zo voor de hand ligt is in figuur 6.5 het NP-gebouw *hoger getekend* dan de omringende polynomiale struikgewassen. Deze verhouding is echter een artistiek misschien acceptabele, maar wetenschappelijk niet gefundeerde interpretatie: *het is namelijk niet bewezen dat P en NP verschillen.*

Van geen enkel probleem in de NP-klasse is bewezen dat er geen polynomiaal algoritme bestaat om het op te lossen. De problemen waarvan wel is bewezen dat ze niet polynomiaal oplosbaar zijn (dammen en go) missen de verificatie-eigenschap en zitten dus niet in NP. We weten dus niet of er polynomiaal verifieerbare problemen bestaan die niet polynomiaal op te lossen zijn ($\mathrm{P} \neq \mathrm{NP}$), danwel, dat polynomiale verifieerbaarheid ook polynomiale oplosbaarheid impliceert ($\mathrm{P} = \mathrm{NP}$)! Het laatste geval zou in de meeste takken van de informatica erg goed uitkomen omdat dan allerlei praktisch relevante optimaliseringsproblemen (zoals TSP) snel oplosbaar zijn.

Van factoriseren en de discrete log is niet bekend dat ze polynomiaal zijn, en ook niet dat ze NP-compleet zijn. Public-keycryptografie vereist hoe dan ook dat het meer tijd kost om een oplossing te vinden dan om die te verifiëren. Daarom is de cryptografie een (het enige) gebied in de informatica waarin men juist hoopt dat $\mathrm{P} \neq \mathrm{NP}$.

6.3 Public-keycryptosystemen

De wiskunde uit hoofdstuk 5 en de algoritmische theorieën uit de voorgaande secties komen samen in een soms raadselachtige, maar uiterst belangrijke en fascinerende toepassing: de public-keycryptografie. In dit hoofdstuk wordt deze vorm van cryptografie eerst ingeleid, waarna de volgende secties belangrijke systemen behandelen.

6.3.1 Verschillende sleutels met onomkeerbare berekeningen

In de klassieke visie op cryptografie wordt de versleuteling en ontsleuteling uitgevoerd met een sleutel die een gedeeld geheim is van zender en ontvanger. Het sleutelbeheer (het overeenkomen van dit gedeeld geheim) is zoals we hebben gezien in sectie 4.4 een van de grootste problemen bij de toepassing van symmetrische cryptografie.

Dat de sleutel aan de tegenpartij niet bekend mag zijn is te onderbouwen vanuit de ontvangstzijde: een tegenstander die dezelfde informatie heeft als de ontvanger kan ook dezelfde berekening (ontsleutelen) ermee uitvoeren. Vanuit de zenderkant echter is de gedachte, dat iemand die weet *hoe* er versleuteld is, *daarmee* in staat is het proces om te keren en uit de cijfertekst de klare tekst te berekenen. In de loop van de jaren zestig ontstond echter het besef, dat het kennen van een berekenmethode *helemaal niet* impliceert dat ook de inverse functie te berekenen is.

De opbloei van de brede belangstelling voor cryptografie in de jaren zeventig leidde ertoe dat dit *onbewezen (!)* principe de basis werd van de nieuwe klasse van cryptografische algoritmen. Als men *wel* kan hardmaken dat de ontvangsleutel geheim moet zijn, maar *niet* dat de zendsleutel geheim moet zijn: maak dan deze twee sleutels *verschillend*. De zendsleutel wordt gebruikt in een berekening waarvan de inverse niet (zonder extra informatie) te berekenen is, en de ontvangsleutel is precies die extra informatie die voor het inverteren nodig is. Wanneer Alice een bericht aan Bob wil sturen zal Bob nu zijn *zendsleutel* aan Alice opsturen en dit hoeft niet via een beschermd kanaal te gebeuren. Uit de beschrijving van de encryptie (zendsleutel) en de uitkomst ervan (de cijfertekst) is dan niet de invoer ervan (klare tekst) af te leiden. Het kan daarmee geen kwaad als de tegenpartij de zendsleutel kent. Bob kan deze sleutel daarom gewoon in een publieke lijst of op zijn website zetten. Voor het terugsturen van berichten (van Bob naar Alice) wordt natuurlijk een compleet andere sleutel gebruikt, namelijk een waarvan Alice het ontvangdeel en Bob het zenddeel heeft.

Tussen het ontstaan van dit idee (zie kader 6.1) en de eerste praktische realisatie ervan (het RSA-algoritme) lagen ongeveer twee jaar. De eisen die Hellman formuleerde voor publickeycryptografie zijn te vervullen voor getalberekeningen, en daarom zijn de klare tekst, de cijfertekst en de sleutels ook steeds getallen. De al eerder genoemde flexibiliteit van getallen (die bij verschillende berekeningen soms dezelfde uitkomst kunnen geven) maakt dat de systemen in allerlei protocollen kunnen worden gebruikt die verder gaan dan pure geheimhouding van data (hoofdstukken 7 tot en met 9).

De secties 6.3.2 en 6.3.3 bespreken de twee meest gebruikte algoritmen, namelijk ElGamal en RSA. Hun populariteit is niet direct te verklaren vanuit wat bekend is over de moeilijkheid van het breken van deze systemen. Er wordt namelijk wel vermoed, maar er is niet bewezen, dat het breken even moeilijk is als de discrete logaritme, respectievelijk factoriseren. In figuur 6.5 betekent dit, dat de plaats van het breken van boven wordt begrensd door de plaatsing van de genoemde problemen. Het lijkt aantrekkelijker, systemen te gebruiken waarvan het breken zich zo hoog mogelijk in deze figuur bevindt. Bijvoorbeeld een systeem waarvan men kan bewijzen dat het even hoog zit als factoriseren, of zelfs bovenin de NP-klasse. Zulke systemen bestaan, maar de ervaringen ermee zijn slecht; dit wordt besproken in secties 6.3.4 en 6.3.5.

In de volgende secties staat steeds x voor de klare tekst en y voor de cijfertekst.

6.3.2 Het ElGamal-algoritme

Het algoritme van ElGamal [ElG85] is gebaseerd op de aanname van onberekenbaarheid van Diffie-Hellman-completering (aanname 6.9).

Het ElGamal-cryptosysteem	
Berekeningen in: $\mathbb{Z}_p^*$, met p een groot priemgetal en g een bekende generator van $\mathbb{Z}_p^*$	Procedure voor sleutelgeneratie: 1. Kies random $a \in \mathbb{Z}_{p-1}$. 2. Bereken $b = g^a$ in $\mathbb{Z}_p^*$. Publieke sleutel: b. Geheime sleutel: a.
Encryptie van x: 1. Kies random $k \in \mathbb{Z}_{p-1}$. 2. Bereken $u = g^k$ en $z = b^k$. 3. Bereken $v = z \cdot x$. 4. $y = (u, v)$	Decryptie van y: 1. Bereken $z' = u^a$. 2. Bereken $x' = v \cdot z'^{-1}$

Toelichting op encryptie en decryptie. De werking van ElGamals algoritme is te vergelijken met het blinderen van invoeren zoals toegepast in sectie 5.2: x wordt verborgen door te vermenigvuldigen met een *blinder* z, dus $v = x \cdot z$. Hiermee blijft x verborgen voor ieder die de blinder niet kent. Bij invoerblindering bleef kennis van de blinder voorbehouden aan één partij, want het bekendmaken van x aan anderen was juist niet de bedoeling.

Bij het versleutelen en ontsleutelen moet het bericht natuurlijk voor de bedoelde ontvanger wel te ontcijferen zijn, daarom wordt aan het bericht informatie u toegevoegd waarmee de ontvanger in staat is z te berekenen. De z wordt hiermee uitgewisseld op de manier van het Diffie-Hellman-protocol (sectie 4.4.1). De publieke sleutel van de ontvanger is een getal $b = g^a$, en de geheime sleutel is a. De blinder z wordt niet willekeurig gekozen, maar berekend als b^k voor een random k, waarna $u = g^k$ wordt meegezonden. De ontvanger kent a en gebruikt als blinder $z' = u^a$; inderdaad is

$$z' = u^a = (g^k)^a = (g^a)^k = b^k = z.$$

Voor encryptie moet tweemaal worden geëxponentieerd en voor decryptie eenmaal; daarom is decryptie iets sneller, al zijn de operaties wel van dezelfde orde van grootte.

Een aanvaller beschikt over de getallen p, g, en b (publieke parameters) en u (uit het bericht). Tussen deze getallen en de blinder z bestaat de Diffie-Hellman-relatie: $DifHel_g(b, u, z)$, immers $u = g^k$ en $z = b^k$. Een cijfertekstaanval op ElGamals algoritme is dus het oplossen van een Diffie-Hellman-completering.

Variant: ElGamal in een subgroep. In plaats van een echte generator van $\mathbb{Z}_p^*$ wordt voor g vaak een element van lagere orde, en wel een priemgetal q, gebruikt. Dat betekent dat niet alle getallen van $\mathbb{Z}_p^*$ als blinder of publieke sleutel kunnen voorkomen, maar alleen de machten van g. De modulus p wordt dan zo gekozen dat $p - 1$ een groot priemgetal q als deler heeft, en bij sleutelgeneratie en encryptie worden a, respectievelijk k uit $\{0, \ldots, q - 1\}$ gekozen. De gebruikte exponenten zijn dan korter en er is minder geheugenruimte nodig. Zoals zal blijken bij de behandeling van bekende algoritmen voor de discrete log (hoofdstuk 15) is het vinden van een logaritme in een subgroep van priemorde niet gemakkelijker dan in de gehele groep.

Een ander voordeel van het gebruik van een subgroep is dat het vrij gemakkelijk is een element van orde q te vinden (als q een priemfactor van $p - 1$ is). Schrijf r voor $\frac{p-1}{q}$. Van een willekeurig element a wordt de macht $g = a^r$ berekend; is deze ongelijk aan 1, dan is zijn orde q. Immers, $g^q = a^{p-1} = 1$, dus $\mathrm{ord}(g)|q$, en omdat q priem is impliceert dit $\mathrm{ord}(g) = q$.

Elementen waarvoor $a^r = 1$ zijn behoorlijk zeldzaam (stelling 5.15), zodat het zoeken naar een element van orde q snel succesvol wordt afgesloten.

Bij het gebruik van subgroep-ElGamal moet men bedacht zijn op een gekozen-cijfertekstaanval die gebruik maakt van een valse u waarvoor de discrete log te berekenen is. Het berekenen van de discrete log wordt in hoofdstuk 15 behandeld, waarna op de kwestie wordt teruggekomen. Hier volstaan we met op te merken dat er in $\mathbb{Z}_p^*$ elementen bestaan waarvoor de discrete log moeilijk is (zoals g en machten van g) maar er bestaan andere elementen waarvoor de discrete log gemakkelijk is.

Bij een gekozen-cijfertekstaanval (*Chosen Ciphertext Attack*) stuurt Oscar Alice een bericht (u, v) en verkrijgt de ontsleutelde waarde x, in dit geval $x = v/u^a$, waaruit $z = u^a$ te berekenen is als v/x. Oscar krijgt met deze aanval de beschikking over een compleet Diffie-Hellman-viertal (g, b, u, z) en het vinden van de geheime sleutel is nu een instantie van Diffie-Hellman-exponent. Bij een echt bericht is u een macht van g en is de discrete log moeilijk, zodat uit u^a *niet* de waarde a te berekenen is. Oscar kan echter een u kiezen waarvoor de discrete log te berekenen is en kan dan uit u^a *wel* de waarde van a berekenen. Gelukkig zijn machten van g gemakkelijk te herkennen: zij voldoen aan $u^q = 1$ (zie het bewijs van stelling 5.17). Bij subgroep-ElGamal moet daarom, tenzij er andere maatregelen tegen de gekozen-cijfertekstaanval zijn, voor ontsleuteling worden getest of $u^q = 1$.

Gedeelde parameters. Het is mogelijk dat elke gebruiker zijn eigen p en g kiest, dan zijn deze getallen onderdeel van de publieke sleutel van die gebruiker. Voor de veiligheid is het echter geen enkel probleem om een hele gemeenschap van gebruikers deze waarden te laten delen. In dat geval zijn p en g (en eventueel q) algemeen bekende systeemparameters, en alleen de waarden a en b zijn per gebruiker verschillend.

De organisatie die deze p en g kiest hoeft niet als bijzonder betrouwbaar bekend te staan. Het is namelijk niet mogelijk bij de generatie van deze getallen geheime informatie achter te houden waarmee logaritmen toch kunnen worden berekend. Wel is nodig, dat $p-1$ ten minste één grote priemfactor heeft, maar dit kan door de gebruikers worden gecontroleerd.

6.3.3 Het RSA-algoritme

Het algoritme van Rivest, Shamir en Adleman [RSA78] is zonder twijfel het bekendste, en ook het meest gebruikte algoritme voor public-keycryptografie. Het RSA-systeem is gebaseerd op de moeilijkheid wortels te trekken in $\mathbb{Z}_n$ zonder kennis van de factorisatie van n. Het algoritme gebruikt hogeremachtswortels, in tegenstelling tot Rabins algoritme dat ongeveer hetzelfde doet maar dan met tweedemachtswortels.

Het RSA-cryptosysteem	
Berekeningen in: $\mathbb{Z}_n$, voor samengestelde n.	Procedure voor sleutelgeneratie: 1. Kies priemgetallen p en q. 2. Bereken $n = p \cdot q$ en $\phi(n)$. 3. Neem $e \in \mathbb{Z}^*_{\phi(n)}$. 4. Bereken $d = e^{-1}$ in $\mathbb{Z}^*_{\phi(n)}$. Publieke sleutel: (n, e) Geheime sleutel: (n, d).
Encryptie van x: $y = x^e$.	Decryptie van y: $x' = y^d$.

Toelichting op encryptie en decryptie. De encryptie is een exponentiatie met bekende exponent e met een publiek bekende modulus en kan zonder kennis van de factoren worden uitgevoerd. Het inverteren van de berekening kan met kennis van p en q worden gedaan, maar meestal worden de factoren alleen bij het genereren van de sleutel gebruikt. Er wordt dan een speciale exponent d berekend die exponentiatie met e voor alle waarden van x ongedaan maakt. Inderdaad, omdat $e \cdot d = 1$ modulo $\phi(n)$, is voor alle x in $\mathbb{Z}_n^*$

$$(x^e)^d = x^{(e \cdot d)} = x^1 = x.$$

Dat deze relatie ook geldt voor x die door p en/of q deelbaar zijn, is gemakkelijk na te gaan.

Mogelijke aanvallen op RSA. Aanvaller Oscar zal proberen, gegeven y en de sleutel (n, e), de e-demachtswortel uit y modulo n te trekken. Van verschillende manieren om dit probleem aan te pakken kan de onmogelijkheid worden bewezen uit de aangenomen onmogelijkheid van integerfactorisering.

1. *Oscar factoriseert n.* We hebben aangenomen dat het factoriseren van de modulus computationeel niet haalbaar is, waarmee de kous af lijkt. Nu beschikt Oscar niet *alleen* over het getal n, maar hij heeft als aanvullende kennis de getallen e en y. Gelukkig kan worden bewezen dat deze aanvullende informatie het vinden van p en q niet mogelijk maakt!

   Het getal y is niet anders dan x^e voor een (willekeurig) getal x. Stel dat Oscar over een algoritme $A(n, e, y)$ beschikt dat de factoren van n vindt met gebruikmaking van e en y. Oscar kan zelf, zoveel hij wil, getallen x_i encrypten (met alleen n en e) en de resulterende y_i gebruiken in algoritme A. Anders gezegd, algoritme $B(n, e)$ kiest een x, berekent $y = x^e$ en roept $A(n, e, y)$ aan en factoriseert n.

   Bewijzen dat kennis van e niets toevoegt is iets lastiger omdat e, in tegenstelling tot y, wél informatie over de factoren van n bevat! Immers, het getal e zit in $\mathbb{Z}^*_{\phi(n)}$, dit is bij sleutelgeneratie door Bob gecontroleerd *met kennis van p en q* terwijl Oscar die test zelf niet kan uitvoeren. We gaan nu het volgende resultaat (uit Bach en Shallit [BS96, stelling 8.8.7]) gebruiken: dat inverteerbare getallen vrij veel voorkomen in $\mathbb{Z}_m$, voor elke m.

   Feit 6.11 $\phi(m) \geq \frac{m}{\log \log m}$.

   Oscar kan daarom een berekening die een inverteerbare e gebruikt, uitproberen met enkele willekeurige e en heeft grote kans op succes.

   Stel dat Oscar over een algoritme $B(n, e)$ beschikt dat de factoren van n vindt, met gebruik van een in $\mathbb{Z}_{\phi(n)}$ inverteerbare e. Algoritme $C(n)$ kiest ongeveer $\log \log n$ willekeurige e en roept $B(n, e)$ met die waarden aan; met grote kans wordt n gefactoriseerd.

   Merk op, dat hiermee nog steeds *niet is bewezen* dat factoriseren onmogelijk is, alleen maar dat de aanvullende gegevens e en y hierbij niet van significante invloed zijn. Dit soort bewijzen wordt veel gegeven in de cryptografie, en de algemene conclusie is: als Oscar probleem P met invoer X niet kan berekenen, kan hij het ook niet met aanvullende informatie Z die Oscar zelf ook kan berekenen. In het laatste geval kon Oscar zelfs geen e garanderen die aan de eigenschap van de echte e voldoet (inverteerbaar in $\mathbb{Z}_{\phi(n)}$), maar de hoge dichtheid van goede e's stond het gebruik van een random getal toe.

   Het is even veilig om bij de sleutelgeneratie d random te kiezen en e uit te rekenen als d^{-1}, maar dat is lastiger te bewijzen (Opgave 6.17).

2. *Oscar vindt $\phi(n)$.* De waarde van $\phi(n)$ is voor Oscar ook zeer bruikbaar omdat daarmee d uit e kan worden berekend. Nu is $\phi(n) = (p-1)(q-1)$, waaruit volgt dat $p+q = n-\phi(n)+1$, en uit het product van p en q (namelijk n) en hun som kunnen de getallen zelf worden opgelost. De veronderstelde onmogelijkheid van het factoriseren van n impliceert derhalve dat $\phi(n)$ niet efficiënt kan worden berekend.

3. *Oscar vindt d.* Omdat ook $\phi(n)$ niet Oscars uiteindelijke doel is, maar het inverteren van de encryptiefunctie, zou Oscar met d al heel blij zijn, ook al heeft hij niet de waarde van p, q, of $\phi(n)$.

   DeLaurentis [DeL84] bewees in 1984 dat kennis van getallen e en d met product 1 in $\mathbb{Z}_{\phi(n)}$ voldoende is voor het factoriseren van n. Volledigheidshalve en ter bevrediging van mogelijke nieuwsgierigheid ernaar wordt deze truc hier gememoreerd, hoewel hij in de praktijk nooit wordt uitgevoerd.

   Bereken $M = e \cdot d - 1$ in $\mathbb{Z}$ en merk op dat (omdat $e \cdot d$ een veelvoud is van de vooralsnog onbekende $\phi(n)$) voor alle $w \in \mathbb{Z}_n^*$ geldt $w^M = 1$. Nu is M een even getal, dat te schrijven is als product van een tweemacht 2^s en een oneven getal t, zodat de macht w^M steeds te berekenen is door het getal w^t s maal te kwadrateren. Net als in de Miller-Rabin-test wordt de rij

   $$w^t, \quad w^{2t}, \quad \ldots, \quad w^{2^i \cdot t}, \quad \ldots, \quad w^M$$

   gevormd en het laatste getal w^M is steeds 1. Er zijn weer drie mogelijke vormen van deze rij: (a) w^t is al gelijk aan 1; (b) het eerste getal is ongelijk 1, na een tijdje komt -1 en dan 1; (c) het eerste getal is ongelijk 1, en voor de eerste 1 staat *niet* -1. Bij een w die volgens route (c) bij 1 belandt wordt een niet-triviale wortel van 1 gevonden, waarmee n is gefactoriseerd! Uiteraard moet dan worden aangetoond dat geval (c) vaak voorkomt, en gelukkig blijkt dit kans $\frac{1}{2}$ te hebben. (Wanneer men een willekeurige $w < n$ kiest is er nog een vierde mogelijkheid, namelijk dat $w^M \neq 1$; in dat geval ligt w buiten $\mathbb{Z}_n^*$ en geeft $\text{ggd}(w, n)$ direct een factor van n. Dit is uiterst zeldzaam.)

   Ook dit type bewijs komt veel voor in de cryptografie, en de algemene conclusie is: als Oscar probleem P met invoer X niet kan berekenen, maar wel met X en extra informatie Z, dan kan hij Z niet zelf berekenen.

Van Oscars uiteindelijke doel (het worteltrekken) wordt wel *vermoed* dat het even moeilijk is als het factoriseren van de modulus, maar *bewezen* is dit niet. Een bewijs hiervoor zou bijvoorbeeld bestaan uit een procedure die n factoriseert, maar waarin het trekken van e-demachts wortels als gegeven wordt verondersteld.

Gedeelde parameters. Het is niet mogelijk voor meerdere gebruikers een gemeenschappelijke modulus te kiezen en alleen e (en d) per gebruiker te laten verschillen. De reden is dat een gebruiker met de rekentruc van DeLaurentis de modulus kan factoriseren en dan de geheime sleutels van zijn medegebruikers kan berekenen. Daarom heeft bij RSA elke gebruiker zijn eigen n (en e en d) nodig, en moet ook altijd voor verschillende sleutels in verschillende groepen worden gerekend.

Encryptie- en decryptietijd. Zowel voor encryptie als voor decryptie is een enkele exponentiatie nodig; duidelijk is dat dit eenvoudig kan worden gedaan in $O(k^3)$ tijd (met Indiaas machtsverheffen). In de praktijk is encryptie een factor $O(k)$ sneller, doordat voor e een klein getal wordt gekozen. Men spreekt dan van de *kleine encryptie-exponentvariant*. De waarden 3,

Kader 6.6: Chinese reststelling als achilleshiel door foutinjectie

Het gebruik van de Chinese reststelling om berekeningen te versnellen is gebleken een achilleshiel van RSA-toepassingen te zijn. Een berekening modulo n wordt gesplitst in twee deelberekeningen, modulo p en modulo q, en dit expliciete gebruik van de factoren is erg gevoelig voor het lekken van informatie erover.
De zwakte wordt uitgebuit in een actieve aanval die *foutinjectie* wordt genoemd. In de berekening van een getal x wordt geprobeerd een van de twee deelberekeningen, zeg modulo p, te verstoren. Het eindresultaat x' zal dan gelijk zijn aan x modulo q, maar ongelijk aan x modulo p zodat $x - x'$ een factor van n onthult.
Bij RSA-smartcards bleek het mogelijk een deelberekening te verstoren door de kaart tijdens de berekening te bestralen met röntgenstraling, of de voedingsspanning te laten fluctueren. Vlastimil Klima en Tomas Rosa [KR01] ontdekten in maart 2001 een bug in PGP die het mogelijk maakte de terugrekening naar $\mathbb{Z}_n$ te manipuleren. Het getal W_p (zie sectie 5.5.1) is versleuteld opgeslagen, maar de lengte is door een indringer manipuleerbaar.

17 en 65537 worden wel gebruikt (PGP gebruikt bijvoorbeeld standaard 65537) en de exponentiatie kost dan 2, 5 of 17 vermenigvuldigingen. Vooral in toepassingen waar de encryptiefunctie vaker wordt uitgevoerd (en bovendien op goedkopere hardware) dan decryptie is dit een groot voordeel.

De decryptietijd kan met een constante factor (circa 4) worden versneld door de Chinese reststelling te gebruiken. Dan wordt y omgezet naar $\mathbb{Z}_p$ en $\mathbb{Z}_q$, er wordt geëxponentieerd met $d \,\%\, (p-1)$ en $d \,\%\, (q-1)$, en het resulaat wordt teruggerekend naar $\mathbb{Z}_n$. Het heen en terugrekenen kost $O(k^2)$ tijd, en de kostbare exponentiatie is nu vervangen door twee exponentiaties op getallen van ongeveer de halve lengte (zie kader 6.6).

6.3.4 Cryptografie op NPC-basis: subset-sumsysteem

De eerste public-keysystemen waren gebaseerd op een probleem dat (mogelijk) hoger in de complexiteitshiërarchie zit dan de getaltheoretische problemen, namelijk subset sum. Deze algoritmen werden ook wel knapsacksystemen genoemd vanwege een interpretatie van het subset-sumprobleem: probeer een rugzak met capaciteit S precies te vullen met een keuze uit objecten met gewicht $a_1 \ldots a_k$. Het vinden van een deelverzameling van getallen met voorgeschreven som is NP-moeilijk, en dit geeft een manier om een bitrij x in een getal te coderen.

Het Merkle-Hellman-knapsacksysteem	
Berekeningen in: Encryptie – $\mathbb{Z}$, Decryptie – $\mathbb{Z}_p$.	Procedure voor sleutelgeneratie: 1. Kies $\vec{s}$ superstijgend. 2. Kies p en $m \in \mathbb{Z}_p^*$. 3. Bereken $\vec{a} = m \cdot \vec{s}$. Publieke sleutel: $\vec{a}$. Geheim: $\vec{s}$, m, p.
Encryptie van x: 1. $y = \sum_i x_i \cdot a_i$	Decryptie van y: 1. $y' = y \cdot m^{-1}$ in $\mathbb{Z}_p^*$. 2. Vind x' uit $\sum_i x'_i \cdot s_i = y'$.

Toelichting op encryptie en decryptie. De publieke sleutel is een reeks van k getallen a_1 tot a_k, waarvan de bitrij $x = x_1 \ldots x_k$ een deelverzameling specificeert. De codering is dus $y = \sum_i x_i \cdot a_i$ en het terugvinden van x uit deze som is natuurlijk precies een instantie van het subset-sumprobleem. Helaas is het niet voldoende dat het terugvinden van x voor Oscar moeilijk is: daarnaast moet het terugvinden van x voor Bob *wel* mogelijk zijn!

Nu is er een speciaal soort rijen waarvoor het terugvinden van x wel gemakkelijk is, en dat zijn de *superstijgende* rijen; een rij $\vec{s}$ is superstijgend als $s_i > \sum_{j<i} s_j$ (elk getal is groter dan de som van de voorgangers). Je kunt een superstijgende rij maken door s_i steeds te kiezen als som van de voorgaande getallen plus een klein random getal:

$$4, \quad 5, \quad 11, \quad 23, \quad 49, \quad 93, \quad 188, \quad 379, \quad 758, \quad 1531.$$

Voor een superstijgende rij is het decoderen van x supereenvoudig omdat je de bits vanaf het grootste getal direct kunt berekenen. Als $y \geq s_k$, dan moet $x_k = 1$, want de andere getallen zijn samen kleiner dan s_k, dus onvoldoende om som y te hebben, en als $y < s_k$ dan is natuurlijk $x_k = 0$. De overige bits worden opgelost uit $y' = y - x_k \cdot s_k$.

Centraal idee van het cryptosysteem is, dat de rij gebruikte gewichten voor Bob superstijgend is, maar voor Alice en Oscar niet. De ordening wordt verstoord door de getallen in $\mathbb{Z}_p^*$ met een multiplier m te vermenigvuldigen. De ontvanger hoeft alleen het totaal y weer door m te delen om x uit de superstijgende rij te kunnen oplossen.

Veiligheid van knapsacksystemen. Het lijkt wel alsof het breken van het knapsacksysteem net zo moeilijk is als subset sum, maar dat is niet zo. Van alle mogelijke subset-suminstanties komt maar een deelverzameling voor, namelijk die waarbij de gewichtenreeks van een superstijgende rij is afgeleid. Deze deelverzameling D is niet zo gemakkelijk te karakteriseren danwel te herkennen, maar omdat het een deelverzameling is geldt wel

1. Als Oscar subset sum kan oplossen, dan kan hij ook de problemen in D oplossen en berichten ontcijferen.

maar niet

2. Als Oscar de instanties in D kan oplossen, dan kan hij alle instanties van subset sum oplossen.

Daarom impliceert de onmogelijkheid om alle subset-suminstanties efficiënt op te lossen (de aanname $\mathrm{P} \neq \mathrm{NP}$) ook niet dat het onmogelijk is het knapsacksysteem te breken. Dit systeem

heeft ook een slechte reputatie: al in 1982 werd door Eier en Lagger [EL83] een efficiënt algoritme gevonden dat de instanties in D kan oplossen. Wanneer een rij a tot D behoort, zijn er meerdere waarden p', m' waarmee a kan worden teruggerekend tot een superstijgende rij, en een aanvaller kan zo'n paar redelijk snel vinden.

De NP-volledigheid van een probleem heeft altijd betrekking op de *worst-case*tijd van een algoritme dat alle instanties van het probleem aankan. Blijkbaar heeft het subset-sumprobleem naast 'enkele' zeer moeilijke, ook een flink aantal makkelijke instanties, en geeft het Merkle-Hellman-systeem vaak gemakkelijke instanties. Wanneer men wil bewijzen dat een cryptosysteem NP-moeilijk te breken is, zal men een constructie moeten geven waarmee het decrypten van berichten kan worden gebruikt om *willekeurige* instanties van bijvoorbeeld subset sum op te lossen. Bewijsbaar NP-moeilijke cryptosystemen zijn nog niet bekend.

De aannamen die we maakten over de getaltheoretische problemen (aannamen 6.6 en 6.7) zijn cryptografisch gezien sterker, omdat de veiligheid van elke individuele sleutel eruit volgt.

6.3.5 Bewijsbaar veilig: Rabins algoritme

Het lijkt aantrekkelijk een cryptografisch algoritme te gebruiken waarvan niet alleen wordt vermoed, maar ook is bewezen dat het breken ervan even moeilijk is als de discrete logaritme of factoriseren. Zo'n cryptografisch sterk systeem wordt nu besproken; ook wordt besproken waarom dit systeem juist helemaal niet zo aantrekkelijk is.

In Rabins systeem bestaat de encryptie uit kwadrateren met samengestelde modulus. Voor het decrypten worden de priemfactoren gebruikt om de wortel te trekken volgens lemma 6.3. De Chinese reststelling (stelling 5.22) wordt gebruikt om de berekening in de componenten $\mathbb{Z}_p$ en $\mathbb{Z}_q$ afzonderlijk te doen. Hierna worden de resultaten gecombineerd met behulp van de getallen W_p en W_q zoals beschreven in sectie 5.5.1.

Rabins kwadraatsysteem	
Berekeningen in: $\mathbb{Z}_n$, waar $n = p \cdot q$, product van grote priemgetallen.	Procedure voor sleutelgeneratie: 1. Kies p en q priem. 2. Bereken $n = p \cdot q$. Publiek: n. Geheim: p en q.
Encryptie van x: $y = x^2$ modulo n.	Decryptie van y: 1. $y'_p = y \,\%\, p$; $y'_q = y \,\%\, q$; 2. $x'_p = \sqrt{y'_p}$; $x'_q = \sqrt{y'_q}$; 3. $x' = W_p \cdot x'_p + W_q \cdot x'_q$.

Toelichting op encryptie en decryptie. Voor encryptie is slechts een enkele vermenigvuldiging nodig. Voor decryptie moet de Chinese reststelling worden gebruikt, er wordt dus naar de subgroepen heen en weer gerekend, en de wortel wordt met een priemmodulus getrokken. Voor toepassing van de methode uit lemma 6.3 kunnen p en q als viervoud min één worden gekozen. Een getal dat het product is van twee zulke priemgetallen heet een *Blum-integer*.

Het kwadraat y heeft natuurlijk vier wortels (volgens stelling 5.23) en op de een of andere manier moet duidelijk zijn welke ervan de invoer x was. Daarom is x meestal een getal dat aan een zekere eigenschap voldoet, bijvoorbeeld dat bepaalde bits op 0 staan. We gaan hier niet verder op in.

De bewezen veiligheid van Rabins systeem. Decryptie in het systeem bestaat uit het trekken van vierkantswortels, waarvan de moeilijkheid equivalent is met het factoriseren van de modulus. Dit volgt uit de eigenschappen van het kwadrateren in $\mathbb{Z}_n$ zoals behandeld in sectie 5.5.2.

Lemma 6.12 *Als Oscar een wortel van één door hem gekozen y kan verkrijgen, factoriseert hij n met kans $\frac{1}{2}$.*

Bewijs. Oscar kiest een random z en berekent $y = z^2$. Een wortel van y is een getal x met $x^2 = z^2$, en door de random keuze van z is er dan kans $1/2$ dat z niet x of $-x$ is, waarmee Oscar een factor vindt als $\text{ggd}(z + x, n)$. □

Dit ene lemma vormt de basis van de twee volgende corollaria, die dus moeten worden gezien als keerzijden van dezelfde medaille.

Corollarium 6.13 (aanname 6.7) *Er bestaat geen efficiënt algoritme om x uit zijn encryptie y te vinden.*

Bewijs. Dit algoritme geeft volgens lemma 6.12 een methode om n te factoriseren, want volgens aanname 6.7 onmogelijk is. □

Corollarium 6.14 *Rabins algoritme is kwetsbaar tegen een gekozen-cijfertekstaanval met $O(1)$ (y, x) paren.*

Bewijs. Elke x die Oscar krijgt bij een door hem gekozen y geeft hem $\frac{1}{2}$ kans om n te factoriseren, en daarmee de geheime informatie van Bob te achterhalen. □

Bij gebruik van Rabins algoritme zal er dus altijd voor moeten worden gewaakt, dat de decryptieprocedure niet een willekeurige invoer accepteert en de uitvoer bekendmaakt. Dit sluit gebruik in combinatie met invoerblindering of in allerlei cryptografische protocollen (die later in dit boek worden beschreven) uit. In een andere cryptografische toepassing, blinde overdracht, kan deze kwetsbaarheid juist weer als voordeel worden uitgebuit.

De paradoxale conclusie is: het is wel prettig als breken van het cryptosysteem even moeilijk is als een getaltheoretisch probleem, maar niet dat dit wordt bewezen.

Verschuiving over b. In een variant op het systeem kiest men bij sleutelgeneratie ook een getal b, en luidt de encryptie $y = x(x + 2b)$. De reden is een technische: we willen niet dat de encryptie van kleine x ($x < \sqrt{n}$) te doorzichtig is omdat er niet daadwerkelijk is gereduceerd na de vermenigvuldiging. Maar $y = x(x+2b) = (x+b)^2-b^2$, de toevoeging van b is dus niets anders dan het toevoegen van een verschuiving over b in de klare-tekstruimte en een verschuiving over $-b^2$ in de cijfertekstruimte. Deze toevoeging heeft dus volstrekt geen invloed op de veiligheid, die steeds gebaseerd blijft op het kwadrateren en worteltrekken.

6.3.6 Symmetrische en asymmetrische cryptografie

Een asymmetrisch, of public-keycryptosysteem is veel gemakkelijker in het gebruik dan een symmetrisch systeem. Bij een symmetrisch algoritme moet immers door twee gebruikers een sleutel worden gedeeld, die aan niemand anders bekend mag zijn. De twee partijen moeten deze sleutel zien af te spreken en deze over een beschermd kanaal aan elkaar mededelen.

Bij public-keycryptografie is het genereren van de sleutel een zaak van slechts één partij; de ontvanger. Wanneer de generatie wordt uitgevoerd op de hardware waarmee de ontvanger

ook de decryptie gaat uitvoeren, is versturen van de geheime sleutel in het geheel niet nodig, wat een veel grotere veiligheid mogelijk maakt. Elke partij die berichten wil ontvangen kiest een sleutelpaar en maakt het publieke deel daarvan aan iedereen bekend (bijvoorbeeld via de website).

Het verzenden van een boodschap door Alice aan Bob verloopt dan zo: Alice verkrijgt de publieke sleutel van Bob, berekent de encryptiefunctie op haar bericht en stuurt het resultaat naar Bob. Bob past dezelfde decryptiefunctie toe op alle ontvangen berichten, ongeacht de afzender. Voor het sturen van een antwoord aan Alice gebruikt hij een andere sleutel, namelijk de publieke sleutel van Alice.

Valkuilen bij public-keycryptografie. Het is niet bezwaarlijk dat de invoer van public-keyalgoritmen een getal moet zijn. Een willekeurige digitale boodschap kan als getal worden opgevat in het tweetallig stelsel. Een boodschap zal doorgaans langer zijn dan een enkel getal, maar het is geen probleem of risico om de boodschap in blokken te verdelen en die een voor een te encrypten.

Men hoeft bij het gebruik van een public-keycryptosysteem geen zorgen te hebben over het uitlekken van de publieke sleutel: een buitenstaander kan van die sleutel op geen enkele manier misbruik maken. Ook bij latere protocollen die we op basis van publieke sleutels zullen zien is dit steeds het geval. Toch is wel enige zorg in de omgang met sleutels vereist. Aanvaller Oscar kan namelijk proberen zijn eigen publieke sleutel aan Alice te sturen en haar in de waan te brengen dat dit de sleutel van Bob is. Dan zal Alice haar bericht met Oscars sleutel encrypten waarna niet Bob, maar Oscar het kan lezen. Ook bij toepassingen als handtekeningen en identificatie (hoofdstuk 7 en 8) is verificatie van de juistheid van de publieke sleutel steeds een belangrijk punt van zorg.

Een groter bezwaar voor public-keycryptografie is de hoge rekentijd voor encryptie en decryptie. Voor algoritmen als DES en IDEA bestaan snelle implementaties in hardware en software. Het uitvoeren van getalberekeningen op getallen van duizend bits lang kost dan in verhouding veel tijd: ruwweg kan men zeggen dat encryptie met symmetrische algoritmen een factor 1000 sneller is!

Hybride systemen. Omdat symmetrische en asymmetrische cryptografie verschillende mogelijkheden en moeilijkheden hebben, ziet men vaak systemen waarin deze twee vormen van cryptografie worden gecombineerd. Public-keycryptografie wordt dan toegepast voor taken als identificatie en ondertekening, maar voor het versleutelen van grote hoeveelheden data wordt dan een symmetrisch algoritme toegepast. Een voorbeeld van een dergelijke combinatie is het telewerkprotocol van ITsec (kader 4.6).

Ook in het populaire e-mailprogramma PGP worden symmetrische en asymmetrische cryptografie op een zinvolle manier gecombineerd. Een (mogelijk lang) e-mailbericht wordt versleuteld met een symmetrisch algoritme, waarbij de sleutel (de *sessiekey*) een random string is. Alleen de sessiekey wordt met een public-keyalgoritme versleuteld, en de versleuteling ervan wordt als header aan het bericht toegevoegd. Dus ongeacht de lengte van het bericht is slechts een enkele encryptie van het public-keysysteem nodig.

Een verder voordeel is mogelijk wanneer een gebruiker, zeg Alice, een bericht wil versturen aan meerdere ontvangers: Bob, Bert, Barbara et cetera. De body van het bericht wordt slechts eenmaal versleuteld maar nu wordt de sessiekey met verschillende publieke sleutels vercijferd, en er worden meerdere headers aan het bericht toegevoegd. Het hele bericht kan dan via een broadcastkanaal eenmaal worden verstuurd, waarbij de headers het juist voor de bedoelde ontvangers mogelijk maken het bericht te lezen.

Kader 6.7: Hybride systeem: Pretty Good Privacy

Als iemand een deurslot zou uitvinden dat *echt* veilig is: volstrekt niemand kan het huis betreden zonder de sleutel... dan zou de politie dat toch erg mooi vinden, de uitvinder voor de Nobelprijs voordragen en de toepassing van deze wondersloten onder de burgerij bevorderen? Philip Zimmermann staat bekend als de cryptoanarchist die sterke encryptie binnen ieders bereik heeft gebracht met zijn programma Pretty Good Privacy (PGP), gratis voor iedereen sinds 1991. Zimmermann werd vrijwel direct vervolgd voor illegale wapenhandel en wapenexport, want superveilige sloten werden in Amerika als munitie beschouwd. De verspreiding van PGP werd door de Amerikaanse overheid tegengewerkt voor zover dat mogelijk was. De vervolging leidde uiteindelijk niet tot een veroordeling en Zimmermann mocht als gerehabiliteerde het Congres toespreken over de liberalisering van cryptografie. PGP is inmiddels overal geaccepteerd, maar in meer en meer landen is sterke cryptografie aan handelsbeperkingen onderhevig; eind 1998 werd hierover in Wassenaar een akkoord ondertekend.
(**Foto:** Phil R. Zimmermann in Red Square, Computerworld Russia.)

Samenvatting en conclusies

Sommige rekenoperaties op getallen kunnen wel, en andere niet efficiënt worden uitgevoerd. Voor het uitvoeren van getalberekeningen is vaak niet meer nodig dan basisschoolrekenkunst, maar dan opgeschaald naar getallen van honderden cijfers lang. De rekentijd voor optellen, vermenigvuldigen en machtsverheffen is polynomiaal in de lengte: voor getallen van k bits lang wordt de rekentijd uitgedrukt als een macht van k (bijvoorbeeld k^2 of k^3). Voor de inverse van machtsverheffen, namelijk worteltrekken of de logaritme, geldt dat niet: de rekentijd van de best bekende algoritmen is uitgedrukt als een k-de macht a^k. Dit betekent, dat voor getallen van een bepaalde lengte (circa duizend bits) machten *wel*, maar wortels en logaritmen *niet* kunnen worden berekend.

De complexiteitstheorie bestudeert problemen die wel en niet efficiënt kunnen worden berekend, en dus ook de vraag of misschien in de toekomst wel polynomiale algoritmen voor wortels en logaritmen kunnen worden gevonden. De complexiteit van deze rekenproblemen is echter nog niet opgelost; het ontdekken van zulke algoritmen kan dus niet worden uitgesloten, al wordt het onwaarschijnlijk geacht.

In de getaltheoretische cryptosystemen voert de zender met het bericht x enkele rekenkundige bewerkingen uit (meestal zit hier een exponentiatie bij) en verzendt het resultaat y. Een afluisteraar ziet y en weet bovendien, hoe y uit x berekend is, maar voor het oplossen moet een wortel of logaritme worden berekend, zodat y niet kan worden gevonden. De ontvanger beschikt over een extra getal waarmee de berekening van y uit x wel mogelijk is.

De bekendste algoritmen van dit soort zijn RSA (gebaseerd op het trekken van hogeremachtswortels) en ElGamal (gebaseerd op logaritmen). Omdat de getalberekeningen veel tijd kosten, worden deze algoritmen vaak met symmetrische cryptografie gecombineerd wanneer lange berichten moeten worden versleuteld.

Opgaven bij hoofdstuk 6

Opgave 6.1 *Schrijf een programma voor de binaire staartdeling op k-bitsintegers. Laat zien dat de complexiteit niet alleen in het algemeen door $O(k^2)$ wordt begrensd, maar ook door $O(k_q \cdot k)$, waar k_q de lengte van het quotiënt is.*

Opgave 6.2 *Laat zien dat het Euclides-algoritme in $O(k^2)$ tijd te implementeren is (gebruik opgave 6.1). Geldt dit ook voor de uitgebreide versie?*

Opgave 6.3 *Bedenk een uitgebreid Stein-algoritme dat naast* $\gcd(a, b)$ *ook bepaalt welke lineaire combinatie hieraan gelijk is.*

Opgave 6.4 *Geef een algoritme dat een generator van $\mathbb{Z}_p^*$ bepaalt. Je mag aannemen dat de factorisatie van $p - 1$ bekend is.*

Opgave 6.5 *Waarom kun je lemma 6.4 niet gebruiken om tweedemachtswortels uit te rekenen?*

Opgave 6.6 *Waar moet je in figuur 6.5 het kraken van het one-time pad plaatsen?*

Opgave 6.7 Veiligheid van Cryptosystemen. *(Uit het tentamen van oktober 1996.)*
Op welke aannamen berust de veiligheid van de volgende crypto-algoritmen: (1) ElGamal; (2) RSA ; (3) DES ; (4) Merkle & Hellman's knapsack algoritme.
Wat is de verhouding tussen deze aannamen en het open "$P \neq NP$" probleem?
Wat doet men in de praktijk om de veiligheid van deze algoritmen te garanderen?

Opgave 6.8 *Wat is de invloed van de betrouwbaarheidsparameter R (zie sectie 6.1.5) op de complexiteit van priemgetalgeneratie?*

Opgave 6.9 *Bewijs dat Diffie-Hellman-exponent zo moeilijk is als het berekenen van de discrete logaritme.*
(Hint: gegeven invoer a voor het logaritmeprobleem, vul a eerst aan met getallen b en c volgens de Diffie-Hellman-relatie.)

Opgave 6.10 *Alice biedt als computationele dienstverlening het worteltrekken modulo een zeker getal m aan voor 10 euro. Tot haar verbazing krijgt zij slechts één klant: Bob, die twee wortels bij haar koopt en vervolgens dezelfde dienst aanbiedt voor 8 euro. Hoe is dit mogelijk?*

Opgave 6.11 *Alice biedt als computationele dienstverlening het derdemachtsworteltrekken modulo een zeker getal m aan voor 10 euro.*
(a) Bob wil hier wel gebruik van maken maar zijn getal x is geheim. Beschrijf hoe hij zijn getal kan laten bewortelen zonder het aan Alice bekend te maken.
(b) Wat is voor Alice de complexiteit om Bobs getal uit zijn geblindeerde invoer te bepalen?

Opgave 6.12 RSA decryptie in subgroepen. *Bij k-bits RSA zijn de decryptiesleutel d en de modulus k-bits getallen. Neem als voorbeeld $m = 77 = 7.11$ en stel $d = 47$.*

(a) Schrijf d binair; hoeveel vermenigvuldigingen (modulo 77) moet je doen om y^d rechtstreeks uit te rekenen? Bereken 16^{47}

(b) Volgens de Chinese rest Stelling kun je y uitdelen naar subgroepen en elke vermenigvuldiging mod 77 vervangen door twee vermenigvuldigingen, eentje mod 7 en eentje mod 11. Neemt de totale hoeveelheid werk hierdoor af?

(c) Merk op dat in $\mathbb{Z}_7^*$ geldt: $x^6 = 1$. Vind een kleinere exponent v zodanig dat $y^v = y^{47}$ in $\mathbb{Z} * 7$. Hoevaak moet je vermenigvuldigen om y^{47} te berekenen modulo 7?

(d) Laat zien dat je een 47ste macht in $\mathbb{Z}_{11}$ kunt vinden met slechts vier vermenigvuldigingen modulo 11. Bereken opnieuw 16^{47} modulo 77.

(e) Laat zien dat decryptie viermaal sneller gaat bij het gebruik van de subgroepen en CRT. (Vermenigvuldigen is kwadratisch.)

Opgave 6.13 *Hoe kun je (voor ElGamal in een subgroep) een priemgetal p vinden zo dat $p-1$ een grote factor q heeft?*

Opgave 6.14 Rabins Kwadraatsysteem. *(Uit het tentamen van januari 1999.)*
In Rabin's cryptosysteem is encryptie kwadrateren in $\mathbb{Z}_n$; de ontvanger kent de factoren van n en kan (als decryptie) wortels trekken in $\mathbb{Z}_n$.

(a) Men noemt het systeem "bewezen veilig". Wat is er precies bewezen en hoe wordt dat bewezen?

(b) Waarom beperkt dit bewijs de bruikbaarheid van het algoritme?

Opgave 6.15 RSA Vercijfering. *(Uit het tentamen van januari 1997.)*
Waarvoor staan de letters RSA in het RSA cryptosysteem?

Hoe ziet een RSA key-pair eruit, en welke functies worden voor encryptie en decryptie toegepast?

Wat gebeurt er, als de vercijferde plaintext niet relatief priem is met de gebruikte modulus, en levert dit in de praktijk problemen op?

Opgave 6.16 *In deze opgave wordt bewezen dat zowel ElGamal- als RSA-versleuteling* homomorf *is.*

Bewijs (voor RSA) dat als y de versleuteling van x is en y' de versleuteling van x', dan is $y \cdot y'$ de versleuteling van $x \cdot x'$.
Bewijs (voor ElGamal) dat als (u, v) een versleuteling van x is en (u', v') een versleuteling van x', dan is $(u \cdot u', v \cdot v')$ een versleuteling van $x \cdot x'$.

Opgave 6.17 *Bewijs dat het even veilig is om bij de sleutelgeneratie van RSA d random te kiezen als om e random te kiezen.*

Opgave 6.18 *Bewijs: als p een priemgetal is dat een viervoud min 1 is, dan is -1 geen kwadraat in $\mathbb{Z}_p^*$.*
Bewijs: Als n een Blum-integer is en a een kwadraat in $\mathbb{Z}_n^*$, dan is van de vier wortels van a er precies één zelf een kwadraat.

Opgave 6.19 Cryptosystemen. *(Uit het tentamen van januari 1997.)*
Wanneer noemen we een cryptosysteem symmetrisch *en wanneer spreken we van een* public-key *systeem?*

Welke cryptosystemen ken je die in de praktijk werden of worden gebruikt? Geef van elk systeem aan, of het een symmetrisch, danwel een public-key systeem is.

Hoofdstuk 7

Digitale handtekeningen

Beveiliging van informatie gaat verder dan zorgdragen voor geheimhouding zoals behandeld in de eerdere hoofdstukken. Misschien zelfs nog belangrijkere functies zijn het verzekeren van *de juistheid* van informatie en *de toegang* tot informatie (of andere objecten) voor de daartoe geauthoriseerde personen. In dit en het volgende hoofdstuk worden de cryptografische technieken behandeld die deze functies ondersteunen.

De juistheid van informatie is natuurlijk moeilijk met geautomatiseerde technieken te controleren, maar het is al een stap in de goede richting als de *herkomst* van de informatie te verifiëren is, indien nodig op een te bewijzen manier. De digitale handtekening is het mechanisme waarmee stukken informatie onlosmakelijk aan personen (of andere partijen in het cryptografische spel) kunnen worden gebonden. De handtekening voorkomt dat een indringer informatie kan vervangen door andere, en voorkomt dat de (rechtmatige) afzender van informatie later kan ontkennen deze te hebben afgegeven.

Enkele toepassingen van handtekeningen vereisen de combinatie met cryptografische technieken, te weten hashfuncties en invoerblindering. Deze technieken worden besproken, maar vergezeld van weer enkele andere toepassingen zodat het hoofdstuk als geheel zicht biedt op de veelzijdigheid van de toepassingen van cryptografie in bijvoorbeeld een eerlijke elektronische muntworp (kader 7.4) en winkelen met gegarandeerde privacy (sectie 7.4.4).

7.1 Eisen en toepassingen

Wanneer informatie wordt beveiligd tegen uitlekken spreekt men van verhulling (*confidentiality*). Verdere informatiebeveiliging wordt verkregen met eigenschappen die *authenticiteit*, *integriteit* en *onloochenbaarheid* worden genoemd.

1. **Authenticiteit:** De ontvanger van een bericht verkrijgt zekerheid over de afzender van het bericht.

2. **Integriteit:** De ontvanger van een bericht verkrijgt zekerheid over de juistheid van de inhoud van het bericht.

3. **Onloochenbaarheid:** (Engels: *non-repudiation*) De ontvanger van een bericht kan achteraf tegen een derde partij bewijzen dat de afzender het heeft gestuurd.

In de meeste gevallen gaan authenticiteit en integriteit samen. Deze twee eigenschappen kunnen worden verkregen met symmetrische cryptografie, namelijk een message authentication code (sectie 7.2.1). Een echte digitale handtekening, die ook onloochenbaarheid geeft, is alleen met public-keycryptografie mogelijk.

7.1.1 Model en eisen

De digitale handtekening is een stukje informatie S dat door een partij (Alice) aan een bericht M kan worden toegevoegd, waardoor later door een ontvangende partij (Bob) kan worden geverifieerd dat M van Alice afkomstig is.

De handtekening mag *alleen* door Alice kunnen worden gemaakt: immers, als ook Bob de handtekening kan maken is er geen onloochenbaarheid. Dit impliceert dat Alice over zekere *geheime*, aan haar gebonden informatie x moet beschikken. Deze informatie noemen we de geheime sleutel van Alice. Om de handtekening te kunnen controleren moet Bob enige kennis over deze sleutel hebben; hij moet immers de handtekening van Alice kunnen onderscheiden van de handtekeningen van andere partijen. De informatie y die Bob heeft over x noemen we de *publieke* sleutel.

Geschreven handtekeningen zijn aan de ondertekende informatie gebonden doordat ze op hetzelfde stuk papier zijn gezet. Omdat elektronische informatie gemakkelijk is te knippen en te plakken, moet de binding van S aan M op een andere manier tot stand komen, namelijk doordat S afhangt van M. Het zetten van een handtekening is dus het berekenen van een functie van M en de geheime sleutel x; deze functie noemen we $Sig(M, x)$ (voor *signature*). De controle van de juistheid van handtekening S onder bericht M met informatie y noemen we $Ver(M, S, y)$ (voor *verify*) met uitkomst *true*/*false*.

Een handtekeningschema definieert de functies *Sig* en *Ver*; deze moeten als eerste natuurlijk voldoen aan de eis dat een door Alice ondertekend bericht door de ontvanger wordt geaccepteerd. Dat wil zeggen, voor alle M en passende geheime en publieke sleutels x en y geldt

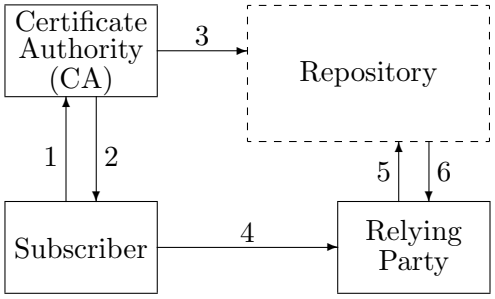
$$Ver(M, Sig(M, x), y) = true.$$

Een complementaire eis is dat het namaken van handtekeningen ondoenlijk moet zijn. De aanval die Oscar op een handtekeningschema doet is een poging om een paar M, S te construeren dat aan $Ver(M, S, y)$ voldoet. Een paar dat hieraan voldoet maar niet door Alice is getekend, noemen we een *vervalsing*. Hier onderscheiden we een universele en een existentiële vervalsing.

We zeggen dat Oscar een *existentiële vervalsing* (of *existential forgery*) maakt als hij bij de publieke sleutel y een M en een S vindt waarvoor $Ver(M, S, y)$. We zeggen dat Oscar een *universele vervalsing* maakt als hij bij de publieke sleutel y en een bericht M een S vindt waarvoor $Ver(M, S, y)$. In beide gevallen veronderstellen we de publieke sleutel voor Oscar bekend. In het eerste geval vindt Oscar het bericht als uitkomst van zijn berekening samen met S, in het tweede geval vindt hij een valse handtekening bij een van tevoren bekende of gekozen M.

Bij de meeste handtekeningschema's is een existentiële vervalsing verrassend gemakkelijk te maken. We zullen dan ook aandacht besteden aan de (eenvoudige) maatregelen waarmee deze kunnen worden verworpen. Meestal wordt ervoor gezorgd dat slechts een kleine fractie van de getallen (of strings) als boodschap M mag voorkomen. Een M die niet is gekozen maar uit een berekening werd gevonden, is dan met een kleine (verwaarloosbare) kans als vals bericht voor Oscar bruikbaar.

Kader 7.1: De public-keyinfrastructuur.



De *public-keyinfrastructuur* is het totaal aan faciliteiten waarmee zekerheid over een juiste public key kan worden verkregen. Men onderscheid hierin doorgaans een *subscriber*, die een certificaat vraagt (1) en krijgt (2) van een *Certificate Authority*. De CA slaat het certificaat ook op (3) in een publiek toegankelijke *repository* of *Registration Authority*. Wanneer de subscriber zich wil verantwoorden (4) tegenover een ander (*Relying Party*) kan deze RP het certificaat vragen (5) en krijgen (6) uit de RA.
Omdat certificaten door de handtekening van de CA beschermd zijn tegen modificatie kan elke subscriber als zijn eigen Registration Authority fungeren. Het voordeel van een centrale RA is dat er een zogenaamde *Certificate Revocation List* kan worden bijgehouden; hierin staan de certificaten die ongeldig zijn geworden, bijvoorbeeld doordat de subscriber meldt dat zijn geheime sleutel is uitgelekt.

7.1.2 Het digitale paspoort: certificaten

Alle toepassingen van public-keycryptografie vereisen dat op betrouwbare manier een publieke sleutel aan een persoon of instantie kan worden gebonden. Om met Alice te communiceren zal Bob als eerste moeten beschikken over die sleutel, zowel voor het encrypten van berichten aan Alice, als voor het controleren van haar handtekening (en, zoals we in hoofdstuk 8 zullen zien, het controleren van haar identiteit). Als Oscar erin slaagt Bob een valse sleutel aan te bieden waarvan hij zelf het geheime deel heeft, kan hij de voor Alice gecodeerde berichten lezen en documenten uit haar naam tekenen die dan door Bob worden geaccepteerd.

Het is, zeker in toepassingen voor electronic commerce, niet mogelijk voor Bob om alle personen met wie hij communiceert zelf te gaan controleren en om hun publieke sleutel te gaan vragen. De uitwisseling van publieke sleutels zal via het open netwerk moeten gebeuren, liefst met zo weinig mogelijk overhead en als het even kan als onderdeel van de communicatie met Alice. In het ideale geval kan Alice zich zonder meer bij Bob aanmelden en *zelf* haar publieke sleutel overleggen. Digitale handtekeningen vormen de basis voor het mechanisme van *certificaten* waarmee een partij inderdaad haar eigen sleutel kan overleggen.

Het mechanisme is afgekeken van de bekende papieren paspoorten. Wanneer Alice zich tegenover Bob identificeert is het van belang dat Bob Alice' paspoort ziet, maar niet hoe hij daar aan komt; Alice mag het zelf overleggen. Het paspoort beschrijft hoe de persoon eruitziet, zelfs met foto, en hoe de handtekening eruitziet. Belangrijke eis voor paspoorten is, dat Oscar niet zelf een paspoort kan maken met zijn eigen beschrijving en handtekening maar de naam van Alice erop. Dat komt doordat, behalve de inhoud van het paspoort, ook de vorm aan strenge eisen voldoet, waardoor alleen de overheid (een gemeente) een paspoort kan maken. Het paspoort kan daarom worden gezien als een door de betreffende gemeente afgegeven verklaring van de binding tussen de naam en de persoonsbeschrijving. Omdat het algemeen bekend is hoe nauwgezet iemands identiteit wordt gecontroleerd door een gemeente voordat een paspoort wordt afgegeven, kan een paspoort worden geaccepteerd als bewijs van identiteit.

Het paspoort is niet meer dan een technisch hulpmiddel om Bobs vertrouwen in de identiteitscontrole van de gemeente te kunnen gebruiken. Oscar kan zich nog op de volgende manieren tegenover Bob als Alice voordoen:

1. Oscar vervalst een paspoort, dat wil zeggen, hij produceert een object dat bij Bob de toetsing van een paspoort kan doorstaan, maar dat gegevens bevat die hij er zelf in heeft gezet.
2. Oscar verkrijgt een echt paspoort op valse naam door zich met succes bij de gemeente voor te doen als Alice.
3. Oscar heeft een document dat helemaal geen paspoort is, maar weet Bob ervan te overtuigen dat het dat wel is. Deze werkwijze is vergelijkbaar met de truc waarmee reizigers in het buitenland studentenkortingen krijgen door bijvoorbeeld een busabonnement te laten zien.

Zij nu Theo (de T staat voor *trusted party* of vertrouwenspersoon) een instantie waarvan Bob de identiteitscontroles vertrouwt. En neem aan dat (x_T, y_T) een sleutelpaar is waarvan Theo het geheime deel en Bob het publieke deel heeft. Een *public-keycertificaat* voor Alice is een string van de vorm $C =$"Alice, y" samen met Theo's handtekening $S = Sig(C, x_T)$ hierover. Ze verkrijgt dit certificaat door zich bij Theo te identificeren als Alice en hem de sleutel y te geven; in de 'echte' wereld zal Theo ongetwijfeld voor deze controle en het aanmaken van het certificaat een klein bedrag vragen. Het certificaat beschrijft dus hoe de echte Alice te herkennen is (de public key y) en bevat een 'echtheidskenmerk' S waarmee het aanmaken van het certificaat naar Theo te traceren is.

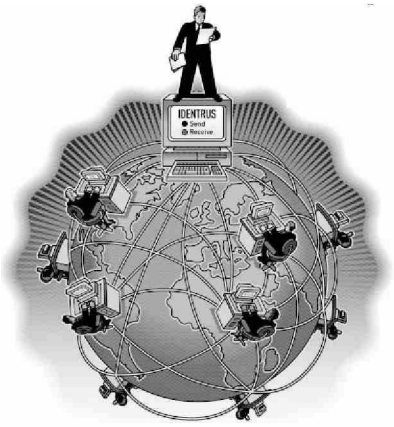
Alice kan zich bij Bob aanmelden en overlegt haar certificaat $(C,\ S)$; Bob controleert $Ver(C, S, y_T)$ en accepteert dat y de publieke sleutel van Alice is. Net als bij de papieren paspoorten zijn er voor Oscar nog verschillende mogelijkheden om Bob een valse y_O in de maag te splitsen:

1. Oscar vormt de string $C' =$"Alice, y_O" en met een universele vervalsing van het handtekeningschema produceert hij een S die aan $Ver(C', S, y_T)$ voldoet.
2. Oscar overtuigt Theo ervan dat hij Alice is, overlegt y_O en verkrijgt een echt certificaat op valse naam.
3. Oscar maakt eerst een sleutelpaar (x_1, y_1) en zorgt ervoor dat Bob y_1 binnenkrijgt maar denkt dat dit de publieke sleutel van Theo is. Vervolgens ondertekent hij de string $C' =$"Alice, y_O" zelf met x_1 en beschikt over een nepcertificaat dat door Bob wordt geaccepteerd.

Aanvallen van het eerste type moeten worden ontmoedigd door de cryptografische sterkte van het handtekeningschema. Aanvallen van het tweede type suggereren gestandaardiseerde manieren van identiteitscontroles bij de instellingen die certificaten verstrekken; hieraan wordt op EU-niveau gewerkt. Om de publieke sleutel van Theo te verifiëren kunnen per systeem verschillende strategieën worden gebruikt. Bij het opzetten van een betaalsysteem bijvoorbeeld kan in elke betaalautomaat de publieke sleutel van de bank bij het installeren worden ingebracht.

Ook zijn er systemen bedacht waarbij de certificering in meerdere niveaus gaat. Er is dan een volgende partij die de certificaatinstellingen certificeert en van deze ene partij kan dan de ene publieke sleutel op een betrouwbare manier aan alle geïnteresseerde partijen worden overgebracht. Identrus (kader 7.2) is een voorbeeld van zo'n infrastructuur.

Kader 7.2: IDENTRUS



Identrus is een samenwerkingsverband van een flink aantal banken ter ondersteuning van 'B2B', business-to-business E-commerce.
De aangesloten financiële instellingen kunnen bedrijven controleren en accepteren voor de verstrekking van een certificaat op smartcard. Voor het ondertekenen van deze certificaten wordt de RSA-handtekening gebruikt. De publieke sleutels van de aangesloten instellingen worden door Identrus zelf gecertificeerd en zo is er sprake van een hiërarchische PKI met drie lagen: (1) de deelnemende bedrijven; (2) de aangesloten financiële instituten; (3) Identrus.
De sleutels op de certificaten kunnen worden gebruikt voor encryptie, handtekeningen, en identificatie (zie hoofdstuk 8). De banken staan in voor de betrouwbaarheid van de bij hen aangesloten klanten, tot en met de mogelijke afgifte van een bankgarantie. Zo kan met minimaal risico elektronisch een transactie tot stand komen, ook tussen partijen die elkaar niet kennen.
Het afgeven van een certificaat is met grote zorg omgeven, wat de uitgifte-capaciteit drukt. ABN-AMRO bijvoorbeeld wil uiteindelijk ongeveer tienduizend certificaten per dag kunnen uitgeven. Dit is voldoende om kleine e-commercebedrijfjes te kunnen bedienen, maar onvoldoende om ook het grote publiek te bedienen.
Website van Identrus: `www.identrus.com`.

Als tegenpool van dit hiërarchische model mag het *network of trust* uit PGP dienen. In het PGP-model bestaan er geen 'lagere' of 'hogere' machten, maar gebruiker Bob bepaalt zelf een lijst van Theo's en Thea's en in welke mate hij op hun identiteitscontroles vertrouwt. Afhankelijk van de door Alice overlegde certificaten kan Bob besluiten haar in de lijst op te nemen.

7.1.3 Controle op kernwapenverdrag

Aan het eind van de jaren zeventig werd tussen de Verenigde Staten en de Unie van Socialistische Sovjetrepublieken een verdrag gesloten waarin terugdringing van het aantal kernwapenproeven werd afgesproken; zie Chaum [Cha82]. Nu is het één ding om iets af te spreken waar je allebei belang bij hebt, maar ingeval van twee partijen die 'elkaar wel kunnen schieten' is het ook nodig er een element van controleerbaarheid aan toe te voegen. Daarom werd ook in het verdrag afgesproken dat elk van de twee partijen een seismologische sensor op het andere grondgebied mocht plaatsen, die gegevens over grondtrillingen naar het thuisland zou moeten sturen.

Hier ontstond echter een nieuw ingewikkeld vertrouwensprobleem:

1. Partij A wil er zeker van zijn dat de uit land B ontvangen data inderdaad de gegevens van de eigen module zijn en niet door partij B gemodificeerd.

2. Partij A wil er ook zeker van zijn dat de gegevens die de module van B opstuurt alleen betrekking hebben op bodemtrillingen en dat er niet stiekem ook andere gegevens worden afgetapt en overgebracht.

De klassieke aanpak van probleem 1 was encryptie (zie sectie 7.2.1) maar dit sluit controle door de andere partij op de inhoud van de gegevens uit. De pas uitgevonden asymmetrische handtekeningen met public-keycryptografie boden uitkomst: de sensor van partij A was voorzien van een geheime sleutel waarmee uitgaande communicatie werd ondertekend, waarmee de inhoud door partij B nog kon worden gezien, en de herkomst door partij A kon worden geverifieerd.

7.1.4 Elektronisch geld

In diverse elektronische betaalsystemen wordt geld gerepresenteerd door informatie die door de bank is ondertekend. Een elektronisch betaaltoken is dan een string gevormd uit de waarde b en een serienummer r, en een omschrijving van wat dit token voorstelt; de complete string omschrijven we als $\langle\, \mathbf{munt}, b, r \,\rangle$. Het token wordt pas geldig als het is voorzien van de handtekening van de bank.

7.2 Handtekeningschema's

Deze sectie bespreekt enkele implementaties van digitale handtekeningen. De bekendste twee zijn afgeleid van een public-keycryptoalgoritme, namelijk de RSA-handtekening en de ElGamal-handtekening. De RSA-handtekening heeft enkele aantrekkelijke eigenschappen waardoor zij voor de meeste toepassingen de geschiktste keuze is. De tekortkomingen van de laatste kunnen echter voor speciale toepassingen als pluspunt worden uitgebuit, waardoor bespreking ervan niet achterwege kan blijven.

7.2.1 Symmetrische bescherming: authenticatiecodes

Bij gebruik van symmetrische cryptografie is er geen verschil tussen de geheime en de publieke sleutel van een partij: Alice en Bob beschikken over dezelfde sleutel k.

Encryptie van een bericht met k biedt, behalve geheimhouding van de inhoud, enige bescherming vergelijkbaar met een handtekening. Laat Bob een bericht Y ontvangen en na decryptie met sleutel k de boodschap $M = D_k(Y)$ lezen. Omdat niemand dan Bob en Alice de sleutel k kent, en Bob de boodschap zelf niet heeft verstuurd, kan Bob erop vertrouwen dat de boodschap van Alice afkomstig is.

Het systeem staat een zeer eenvoudige existentiële vervalsing toe. Immers, elke string Y die door Oscar op Bob wordt afgestuurd zal door Bob worden geaccepteerd als ondertekende $M = D_k(Y)$. Oscar heeft hiermee dus een correct 'ondertekend' bericht verstuurd, alleen kan hij niet kiezen wat hij stuurt, en niet eens *weten* wat hij stuurt. Tegen deze 'blinde vervalsing' kan men zich echter eenvoudig wapenen door af te spreken dat elk bericht met een reeks van twintig nullen moet beginnen. De kans dat het door Oscar vervalste bericht hieraan voldoet is minder dan 1 op een miljoen.

In deze opzet is er nog een koppeling met versleuteling (dus verhulling van de inhoud), maar deze wordt losgelaten in de zogenaamde *authenticatiecodes* (Engels: *Message Authentication Code*, MAC). Alice en Bob gebruiken behalve hun geheime sleutel k een publiekbekende hashfunctie H (zie sectie 7.3). Een hashfunctie beeldt een willekeurig lange bitrij af op een rij van vaste lengte (bijvoorbeeld 160 bits) en wel zo, dat het niet mogelijk is om uit het resultaat informatie over de invoer af te leiden.

De authenticatiecode onder bericht M wordt berekend als $S = H(k, M)$ en de code S bij bericht M wordt met dezelfde formule geverifieerd. Een aanvaller, die over M en bijbehorende S beschikt, kan hieruit k niet afleiden al kent hij H, en kan dus ook niet M wijzigen omdat hij niet in staat is de bijpassende S te berekenen.

Let erop dat het 'ondertekenen' van de boodschap en het 'controleren' van de handtekening met *dezelfde* informatie k gebeuren. Bob verkrijgt daardoor wel zelf het vertrouwen dat de boodschap van Alice afkomstig is, maar kan een derde partij hiervan niet overtuigen. Ten eerste zal een derde pas overtuigd zijn van de juistheid van de code ($S = H(k, M)$) na onthulling van de geheime sleutel k, ten tweede is het zelfs dan nog mogelijk dat Bob zelf de boodschap met code heeft gemaakt. Er is in deze vorm van ondertekening dus wel sprake van authenticiteit en integriteit, maar niet van onloochenbaarheid.

Authenticatiecodes worden heel veel gebruikt, bijvoorbeeld in protocollen als Kerberos en UMTS (hoofdstuk 10). In deze toepassingen is onloochenbaarheid niet van belang, en de kosten van berekening en verificatie van een authenticatiecode zijn lager dan voor een echte handtekening.

7.2.2 De RSA-handtekening

De RSA-functies kunnen direct worden gebruikt voor het zetten van handtekeningen. De geheime en publieke sleutel worden gekozen als gebruikelijk, dat wil zeggen, n een groot getal, e en d elkaars inverse modulo $\phi(n)$, publiek is (n, e) en geheim is (n, d).

Als ondertekening van M wordt de decryptiefunctie gebruikt, dus $Sig(M, (n, d)) = M^d \bmod n$ en voor de verificatie wordt getest of dit resultaat klopt: $Ver(M, S, (n, e)) = (S^e \bmod n = M)$. Zowel het zetten als het verifiëren van een handtekening kost dus één exponentiatie. Door gebruik van de variant met kleine e kan het verifiëren wel een orde van grootte sneller worden gemaakt.

Ook de RSA-handtekening is eenvoudig existentieel te vervalsen: bij een willekeurig getal S kan Oscar direct een bericht $M = S^e$ uitrekenen waarvoor S een geldige handtekening is. Met de publieke e kan Oscar ook uitrekenen welk bericht dat is, dus de reeks van twintig nullen uit de vorige sectie is niet meer afdoende om de vervalsing ongevaarlijk te maken. Bij de RSA-handtekening kunnen Alice en Bob afspreken dat de eerste tachtig bits van M nullen moeten zijn; dan moet Oscar dan 2^{80} getallen S proberen om een bericht te vinden dat door Bob wordt geaccepteerd. Oscar kan dit bericht wel zien, maar niet beïnvloeden. In plaats van een reeks vast afgesproken nullen kan er ook een andere string zijn die in M moet voorkomen, bijvoorbeeld een hashwaarde (zie sectie 7.3).

7.2.3 De ElGamal-handtekening

In het ElGamal-cryptosysteem zijn er gedeelde parameters p (priemmodulus) en g, een element van grote orde. Een geheime sleutel is een getal a en de publieke sleutel is $b = g^a$. Deze informatie kan ook in een handtekeningsysteem worden gebruikt. Net als bij versleuteling (zie sectie 6.3.2) wordt er bij ondertekening een random getal gekozen en een handtekening bestaat uit een paar van twee getallen.

De berekening van een handtekening bij M:

1. Kies een random getal k met inverse k^{-1} modulo ord(g);
2. Bereken $c = g^k$;
3. Bereken $d = (M - a \cdot c) \cdot k^{-1}$.
4. De handtekening is (c, d).

De verificatie bestaat uit de controle dat $c < p$ en $b^c \cdot c^d = g^M$.

Inderdaad geldt dat als (c, d) bij M is berekend als boven aangegeven, dat

$$\begin{aligned} b^c \cdot c^d &= (g^a)^c \cdot (g^k)^{(M-a\cdot c)\cdot k^{-1}} \\ &= g^{a\cdot c + k\cdot(M-a\cdot c)\cdot k^{-1}} \\ &= g^M \end{aligned}$$

Net als bij het RSA-schema is een existentiële vervalsing mogelijk, en wel met de volgende berekening:

1. Kies willekeurige i en j.
2. Bereken $c = g^i \cdot b^j$.
3. Bereken $d = -c \cdot j^{-1}$.
4. Bereken $M = -c \cdot i \cdot j^{-1}$.

Het paar (c, d) is een geldige handtekening voor M (zie opgave 7.2). Er is ook een vervalsing mogelijk wanneer de verificatieprocedure voor c getallen groter dan p accepteert. Dan kunnen met de Chinese reststelling, (c modulo p) en (c modulo ord(g)) onafhankelijk worden gekozen en is een universele vervalsing mogelijk (opgave 7.4).

Gebruik van het schema vereist bijzondere zorg voor de geheimhouding en keuze van de gebruikte random getallen k. De geheime sleutel a is ten eerste af te leiden uit een ondertekend bericht met de daarbij gebruikte k, en wel als volgt: $a = (M - k \cdot d) \cdot c^{-1}$.

Het is ook mogelijk de sleutel af te leiden uit twee berichten met handtekening waarvan slechts bekend is dat ze met dezelfde k zijn getekend, ook als die k zelf niet bekend is. Immers, zij bekend M_1 met $c_1 = g^k$ en $d_1 = (M_1 - a \cdot c_1) \cdot k^{-1}$ en bericht M_2 met $c_2 = g^k$ en $d_2 = (M_2 - a \cdot c_2) \cdot k^{-1}$. (Omdat het eerste deel, c, van de handtekening alleen van k afhangt en niet van M, is het hergebruik van k direct zichtbaar aan gelijke c in de twee handtekeningen.) Nu geldt:

$$\begin{aligned} d_1 d_2 &= d_2 d_1 & \\ &\Rightarrow d_1 \cdot (M_2 - a \cdot c_2) \cdot k^{-1} = d_2 \cdot (M_1 - a \cdot c_1) \cdot k^{-1} & \text{invullen} \\ &\Rightarrow d_1 \cdot (M_2 - a \cdot c_2) = d_2 \cdot (M_1 - a \cdot c_1) & \text{vermenigvuldig met } k \\ &\Rightarrow a \cdot (d_1 \cdot c_2 - d_2 \cdot c_1) = d_1 M_2 - d_2 M_1 & \text{omschrijven} \\ &\Rightarrow a = (d_1 M_2 - d_2 M_1) \cdot (d_1 c_2 - d_2 c_1)^{-1} & \text{modulo ord}(g). \end{aligned}$$

Opgave 7.5 werkt deze berekening nog iets verder uit.

In de meeste toepassingen van handtekeningen is deze eigenschap van het ElGamal-schema een zwakte, maar ze kan ook worden uitgebuit als voordeel. Het is namelijk mogelijk een sleutel te verstrekken waarmee precies eenmaal (veilig) een bericht kan worden ondertekend, omdat bij het ondertekenen van een tweede bericht de sleutel uitlekt. Dit wordt bijvoorbeeld bij de implementatie van elektronisch geld gebruikt om te voorkomen dat een elektronisch bankbiljet meerdere malen wordt uitgegeven. Bij het opnemen van een bankbiljet verkrijgt de gebruiker een geheime sleutel waar zijn identiteit in is gecodeerd, en waarmee bij het uitgeven van het biljet een handtekening moet worden gezet. Het eenmaal uitgeven van het biljet kan anoniem gebeuren. Een poging om het biljet tweemaal uit te geven impliceert dat een tweede bericht wordt ondertekend, waarmee later bij de bank de identiteit van de betrokkene kan worden berekend (sectie 13.2.3).

De keuze van een random getal bij het ondertekenen geeft aanleiding tot een probleem dat bekend staat als *onderbewust kanaal* of *subliminal channel*. In plaats van een random k kan een getal worden gekozen afhankelijk van zekere informatie die bij het zetten van de handtekening beschikbaar is, en die is dan in de handtekening gecodeerd. Simmons [Sim84]

noemt als voorbeeld een paspoort waarvan de echtheid voor iedereen te controleren is, maar waaraan douaniers kunnen zien of iemand van terrorisme of smokkel verdacht is geweest. De informatie is echter onzichtbaar voor wie niet van de wijze van codering op de hoogte is, en het is zelfs niet uit een serie handtekeningen af te leiden dat er subliminaal informatie in is gecodeerd. De aanwezigheid van een subliminal channel in het handtekeningschema maakt het ongeschikt voor toepassingen waarbij moet worden gecontroleerd wat voor informatie er wordt overgebracht, zoals bij de kernwapenproefcontrole.

7.2.4 Vergelijking van RSA- en ElGamal-handtekeningen.

De meest voorkomende schema's zijn het RSA-schema en diverse varianten van de ElGamal-handtekening. Bij het vergelijken van de kosten kan men kijken naar de tijd voor ondertekenen en verifiëren en naar de grootte van handtekeningen en sleutels.

Bij RSA kosten zowel ondertekenen als verifiëren één exponentiatie; bij gebruik van de kleine publieke exponent kan men het verifiëren zelfs terugbrengen tot enkele vermenigvuldigingen. In veel toepassingen komt verificatie veel vaker voor dan ondertekenen en dan is dit een voordeel. Bij ElGamal kost een ondertekening één exponentiatie en een verificatie drie; dit is dus voor veel toepassingen minder aantrekkelijk.

De handtekening bestaat bij RSA uit één getal, dat zo groot is als de RSA-modulus en dat is om aan huidige veiligheidseisen te voldoen minstens 1024 bits. Bij ElGamal is de handtekening een paar, maar deze kan men nemen in het bereik 0 tot ord(g). En dan is er een voordeel van toepassing van ElGamal in een subgroep: er is voldoende veiligheid wanneer de orde van g ongeveer 2^{160} is, waarmee de handtekening uit twee getallen van 160 bits bestaat en dus kleiner dan de RSA-handtekening is. Vooral bij toepassingen waar meerdere handtekeningen op een chipkaart moeten worden opgeslagen (elektronisch geld) is dit een voordeel.

Patentkwesties zijn als argument in het voordeel van ElGamal gebruikt: het RSA-algoritme is namelijk lange tijd gepatenteerd geweest, maar het patent is in september 2000 afgelopen, waarmee dit argument is vervallen.

Het ElGamal-schema is iets gecompliceerder en het bestaan van een subliminal channel doet de populariteit van het algoritme geen goed. Het schema van ElGamal is gebruikt als *Digital Signature Algorithm* in het Amerikaanse Capstone-project (kader 7.3). Het wordt in Amerika het meest gebruikt, terwijl in Europa RSA populairder is.

7.3 Hashfuncties

Een methode om getallen of strings van een bepaalde lengte te ondertekenen kan niet zonder meer worden gebruikt voor het ondertekenen van langere berichten. Je kunt een bericht wel in blokken opdelen en blok voor blok ondertekenen (vergelijkbaar met de aanpak bij encryptie) maar dit biedt een aanvaller de mogelijkheid om zelf een bericht samen te stellen uit delen van andere berichten. Voor het ondertekenen van berichten langer dan een getal is daarom een nieuw concept nodig en wel de hashfunctie.

7.3.1 Definitie en gebruik: fingerprinting

Een cryptografische hashfunctie beeldt bitrijen van willekeurige lengte af op bitrijen van vaste lengte. Het is dus een functie $H : \mathbb{B}^* \to \mathbb{B}^k$ en als eerste moet nu duidelijk zijn dat, omdat $\mathbb{B}^*$

Kader 7.3: Het Capstone-project

In het begin van de jaren negentig lanceerde de Amerikaanse overheid een programma genaamd Capstone om te komen tot een pakket van cryptografische standaarden met een gemeenschappelijke veiligheid van tachtig bits. Onderdelen zijn een symmetrische encryptiealgoritme (met 80-bitssleutels), een hashfunctie (met 160-bitshashwaarden om sterk botsingsvrij te zijn), en een handtekeningalgoritme. De *Digital Signature Standard* (DSS) die onderdeel is van Capstone gebruikt het ElGamal-schema.
Het gehele Capstone-project is in diskrediet gebracht door de controverse rond het encryptiealgoritme, *Skipjack* geheten. Dit algoritme werd geheimgehouden (op zich al reden tot controverse), en alleen beschikbaar gesteld in de vorm van de *Clipperchip*. Deze chip implementeerde niet alleen Skipjack, maar in combinatie met een vorm van *key escrow*, hetgeen betekent dat de Amerikaanse overheid van elke chip de sleutel bewaart! De overheid zou dus zelf alle met Skipjack versleutelde communicatie kunnen lezen, in principe alleen op gerechtelijk bevel tegen een verdachte. De weerstand tegen key escrow was enorm en de pogingen van de Amerikaanse regering om Clipper in te voeren toonden voor velen aan waar het de regering om gaat bij de toepassing van cryptografie: de greep op burgers vergroten.
De DSS is technisch niet omstreden (Skipjack overigens ook niet) maar nu wordt begrijpelijk waarom velen niet blij zijn met een handtekeningschema dat een subliminal channel bevat, en dus een mechanisme om digitale paspoorten onzichtbaar van merktekens te voorzien.

een oneindige en $\mathbb{B}^k$ een eindige verzameling is, deze afbeelding *niet injectief* is. Er zijn (vele!) X_1 en X_2 waarvoor geldt $X_1 \neq X_2$ en $H(X_1) = H(X_2)$.

Een vast gekozen hashfunctie maakt onderdeel uit van het ondertekenen en controleren van langere berichten, en wel op deze manier. De parameters van de handtekeningfucties en de k van de hashfunctie worden zo gekozen dat een k-bitsgetal in één keer als argument van *Sig* kan worden aangeboden. Als ondertekening van bericht M geldt nu de *Sig*-waarde van $H(M)$, voor dit doel ook wel de *fingerprint* van M genoemd. Het ondertekenen van bericht M gaat dus zo:

1. Bereken $F = H(M)$;
2. Bereken $S = Sig(F, x)$;
3. De handtekening is S.

De controle van deze handtekening omvat nu ook de toepassing van H op M:

1. Bereken $F = H(M)$;
2. $Ver(F, S, y)$.

Bij ondertekening en controle wordt natuurlijk dezelfde waarde F berekend en daarom zal de controle van een correct gezette handtekening slagen.

Het toevoegen van fingerprinting aan het handtekeningschema biedt extra mogelijkheden voor het vervalsen van handtekeningen en daaruit volgen enkele eisen waaraan H moet voldoen.

Poging 1: De existentiële vervalsing. Oscar gaat uit van een correct paar F en S. Dit kan een paar zijn dat door Alice is ondertekend, maar we hebben ook gezien dat de gebruikte schema's doorgaans het vinden van zo'n paar toelaten (existentiële vervalsing). Oscar zoekt een bericht M waarvoor $H(M) = F$, en beschikt hiermee over een vervalsing M, S.

Definitie 7.1 *Hashfunctie H is* one-way *als het ondoenlijk is uit F een M te berekenen waarvoor $H(M) = F$.*

Let wel, het gaat hier niet om het *bestaan* van dergelijke berichten M, maar om de moeilijkheid van het vinden ervan.

Als deze vervalsing lukt, beschikt Oscar over een getekend bericht, geheel zonder dat Alice er aan te pas is gekomen. Er is dan sprake van schending van authenticiteit. Oscar kan naar een geschikte M zoeken met een brutekrachtaanval en als we willen dat hij verwacht 2^x berekeningen moet doen, zal de lengte van de hash ten minste x bits moeten bedragen.

Poging 2: Het verplaatsen van de handtekening. Oscar gaat uit van een door Alice ondertekend bericht M met handtekening S. De handtekening is geldig voor elk bericht M' met $H(M') = H(M)$ en Oscar kan proberen zo'n bericht te vinden. De aanval lijkt op de vorige, maar Oscar heeft nu als extra informatie alvast een bericht dat op de begeerde fingerprint wordt afgebeeld en dit leidt daarom tot een net iets sterkere eis op H.

Definitie 7.2 *Hashfunctie H is* zwak botsingsvrij *als het, gegeven M, ondoenlijk is een $M' \neq M$ te vinden met $H(M') = H(M)$.*

Als deze aanval lukt, heeft Alice wel moeten ondertekenen, maar Oscar heeft de inhoud van het bericht veranderd en er is dus schending van integriteit waarbij Oscar slechts één door Alice ondertekend bericht hoeft te zien (passieve aanval). Ook hier is een brutekrachtaanval mogelijk en een x-bitsveiligheid impliceert een hash van ten minste x bits breed.

Poging 3: De verjaardagsaanval. Oscar maakt twee berichten met dezelfde fingerprint (bijvoorbeeld waarbij M_1 gunstig is voor Alice en M_2 gunstig voor Oscar). Alice wil M_1 wel ondertekenen, en de afgegeven handtekening S is meteen ook voor M_2 geldig, waarmee Oscar beschikt over een handtekening onder een bericht dat Alice niet heeft ondertekend.

Definitie 7.3 *Hashfunctie H is* sterk botsingsvrij *als het ondoenlijk is, berichten M_1 en M_2 te vinden met $M_1 \neq M_2$ en $H(M_1) = H(M_2)$.*

Als de aanval lukt is er weer sprake van schending van integriteit, maar het *zien* van een echt bericht van Alice is niet voldoende: Oscar moet Alice een *door hem gemaakt* bericht laten tekenen (actieve aanval) en daarvan kan hij de handtekening overzetten.

Een berekening die Oscar kan opzetten om zo'n paar te vinden kan uitgaan van de verjaardagsstelling (stelling A.7). Hieraan ontleent de aanval ook zijn naam. Oscar produceert een reeks 'goede' berichten M_1 en een reeks 'slechte' berichten M_2 en berekent steeds de fingerprint tot hij een botsing vindt. Dit gebeurt verwacht na ongeveer $\sqrt{2^k}$ pogingen als de fingerprint lengte k heeft. Daarom vereist sterke botsingsvrijheid een dubbele lengte van de fingerprints: wil men dat een aanval verwacht 2^x stappen kost, dan is een fingerprint van minstens $2x$ bits nodig.

Dit verklaart meteen waarom het Capstone-project (kader 7.3), met als uitgangspunt een 80-bitsveiligheid, wel kan volstaan met een encryptiealgoritme met 80-bitskeys, maar een 160-bitshashfunctie nodig heeft.

Onderlinge implicaties. De eisen op een hashfunctie kunnen zo worden samengevat:

1. Een hashfunctie moet one-way zijn voor het behoud van authenticiteit.

2. Een hashfunctie moet zwak botsingsvrij zijn voor het behoud van integriteit tegen een passieve aanval.

3. Een hashfunctie moet sterk botsingsvrij zijn voor het behoud van integriteit tegen een actieve aanval.

Zijdelings merken wij op dat als $NP = P$ (sectie 6.2.3), er in het geheel geen one-wayfuncties bestaan. De drie genoemde eisen zijn oplopend in sterkte en daarom hoeft men bij het ontwerp van een hashfunctie alleen met sterke botsingsvrijheid rekening te houden.

Lemma 7.4 *Als H sterk botsingsvrij is, is H zwak botsingsvrij.*

Bewijs. Stel dat H *niet* zwak botsingsvrij is. Bij een gegeven M_1 kan een M_2 worden gevonden met dezelfde hashwaarde. Dan kan, zonder kennis van een M_1 vooraf, een paar worden gevonden door M_1 random te kiezen en M_2 erbij te zoeken met dezelfde hashwaarde. Dus H is niet sterk botsingsvrij. □

Lemma 7.5 *Als H zwak botsingsvrij is, is H one-way.*

Bewijs. Veronderstel dat H *niet* one-way is. Bij een gegeven fingerprint F kan een M worden gevonden met $H(M) = F$. Maar dan kan bij een gegeven (willekeurig) bericht M_1 eerst de fingerprint $F = H(M_1)$ worden berekend en vervolgens wordt een M berekend met F als fingerprint. De kans dat M gelijk is aan M_1 is klein vanwege het grote aantal berichten met hashwaarde F. Er is dus een tweede bericht gevonden met dezelfde hashwaarde, waarmee H niet zwak botsingsvrij is. □

De omgekeerde implicaties gelden niet (mits $P \neq NP$).

7.3.2 Hashfuncties: MD5, SHA, Chaum

Men kan hashfuncties zien als een soort eenrichtingsencryptie: het bericht wordt op een andere bitrij afgebeeld, alleen is er geen inverse bij. De constante lengte van de uitvoer wordt meestal verkregen door een soort chaining toe te passen. De bouwsteen van een k-bitshashfunctie is dan een *blokfunctie* met vaste invoer- en uitvoerlengte, dus $h : \mathbb{B}^m \times \mathbb{B}^k \to \mathbb{B}^k$. Het bericht M wordt in blokken $b_1, \ldots, b_l$ van m bits verdeeld, die elk een blokhash krijgen toebedeeld volgens de relatie $f_i = h(b_i, f_{i-1})$. De eerste blokhash wordt met een constante f_0 uitgerekend. Om aan een veelvoud van m bits te komen wordt het bericht aangevuld met nullen en de lengte van het oorspronkelijke bericht wordt opgenomen in het laatste blok.

De botsingsvrijheid van de gehele hashfunctie wordt nu overgenomen van de blok-functie:

Propositie 7.6 *Als er M_1 en M_2 zijn met $M_1 \neq M_2$ en $H(M_1) = H(M_2)$, dan vindt men ergens in de berekening van H-waarden f_1, f_2, b_1, b_2 met (1) $(b_1, f_1) \neq (b_2, f_2)$ en (2) $h(b_1, f_1) = h(b_2, f_2)$.*

Het ontwerp van de hashfunctie kan zich concentreren op het ontwerp van een (sterk botsingsvrije) blokfunctie.

Functies gebaseerd op bit- en byteoperaties. Een populaire hashfunctie is MD5, onder andere toegepast in PGP (kader 6.7). De blokken zijn 512 bits lang en de breedte van de hash is 128 bits. Het algoritme begint met de initialisatievector f_0 en voor ieder blok van het bericht wordt de functie h uitgevoerd. Deze behelst 64 achtereenvolgende bewerkingen op de string, afhankelijk van de data in het blok.

Als onderdeel van het Capstone-project werd in Amerika een 160-bitshashfunctie gedefinieerd, genaamd *Secure Hash Algorithm*. SHA levert een hash van 160 bits, dit is immers nodig om de gewenste 80-bitsveiligheid ook te geven tegen verjaardagsaanvallen. De initialisatievector is dan ook 160 bits lang, en per datablok worden er tachtig operaties mee uitgevoerd.

De precieze operaties van MD5 en SHA zijn te vinden in Schneier [Sch96, Hf. 18]. Deze hashfuncties zijn snel en hebben met symmetrische cryptografie gemeen dat de gebruikte operaties weinig structuur hebben. We spreken wel van *ad-hocalgoritmen* en de werkwijze is te vergelijken met die van symmetrische encryptiealgoritmen. Door het gebrek aan structuur is er over de veiligheid van de functies weinig te bewijzen.

Getaltheoretische hashfunctie: Chaum, Van Heijst, Pfitzmann Naast de ad-hocfuncties bestaan er ook systemen die met de asymmetrische cryptografie vergelijkbaar zijn in die zin dat ze op manipulatie van getallen zijn gebaseerd. De veiligheid ervan kan dan bewijsbaar zijn, althans, af te leiden uit een van de aannames in sectie 6.2.1, en de functies worden dan cryptografisch sterk genoemd. Als voorbeeld beschrijven we de hashfunctie van Chaum e.a. [CHP91], gebaseerd op onberekenbaarheid van de discrete log.

Zij p een priemgetal en g een element van hoge orde q, ook een priemgetal. De blokfunctie H wordt gedefinieerd door het *random* kiezen van een tweede element h van orde q. De hashfunctie verdeelt het argument x in twee delen x' en x'' (zoals steeds bij een blokfunctie) en luidt: $H(x) = g^{x'} h^{x''}$.

Deze functie is *sterk botsingsvrij* (onder aanname van onberekenbaarheid van index met grondtal g). Uit een paar x, y met $H(x) = H(y)$ volgt namelijk

$$\begin{array}{rcll} H(x) = H(y) & \Rightarrow & g^{x'} h^{x''} = g^{y'} h^{y''} & \text{uitschrijven van definitie} \\ & \Rightarrow & g^{x'-y'} = h^{y''-x''} & \text{overbrengen van factoren} \\ & \Rightarrow & g^{(x'-y')(y''-x'')^{-1}} = h & \text{want } h^1 = h. \end{array}$$

Omdat met een botsing de index van h wordt gevonden, is het vinden van een botsing zo moeilijk als het berekenen van de index.

Het kiezen van de parameter h is een delicate zaak omdat iemand deze stiekem kan kiezen als g^a voor een bekende a. Met kennis van a is H *niet meer zwak botsingsvrij*. Immers, gegeven $x = (x', x'')$ kan y worden gevormd als $(x' + a.r, x'' - r)$ en dan is een botsing gecreëerd. De functie blijft *wel one-way*, ook als ind(h) bekend is.

7.3.3 Toepassing: blobs

Hashfuncties kunnen ook gebruikt worden voor de implementatie van een ander handig cryptografisch primitief, namelijk de *blob*. Een blob wordt gebruikt in een cryptografisch protocol wanneer het nodig is dat een partij zich vastlegt op bepaalde informatie, zonder dat die op dat moment al door anderen gezien mag worden. De meest elementaire toepassing is de 'eerlijke muntworp', geïllustreerd in kader 7.4.

De elementaire operaties voor een blob zijn het *inpakken* van x en het *openen* van een blob, en ze zijn vergelijkbaar met een envelop waar je x instopt. Bij het *inpakken* van x wordt er een waarde bl berekend aan de hand van x: $bl = Inpak(x)$. Bij het openen van een blob wordt

Kader 7.4: De eerlijke muntworp

- *Met Linda.*
- Hoi Linda, dit is Jon. Ga je vanavond mee uit? Geef me een kans!
- *(zucht) Oké, Ik neem een getal in gedachten, als je in één keer raadt of het even is.*
- Eeehhh..... oneven.
- *Sorry Jon, 't was 6. (klik)*

Jon kan voorkomen dat hij door Linda wordt afgescheept door haar te vragen haar getal als blob aan hem te geven voor hij raadt:

- *Met Linda.*
- Hoi Linda, dit is Jon. Ga je vanavond mee uit?
- *(zucht) Als je in één keer raadt of mijn random x even is.*
- Geef je me eerst $H(x)$? Geef me een kans!
- *VtIDxwaWV0QGNzLnJ1.*
- Eeehhh..... oneven.
- *Sorry Jon, 't was 6.*
- Maar $H(6) \neq VtIDxwaWV0QGNzLnJ1$, Linda. Ik ben er om acht uur.

Als H sterk botsingsvrij is vindt Linda geen even *en* oneven x met dezelfde hashwaarde zodat ze na het opgeven van y niet meer van x kan veranderen (binding). Als H one-way is kan Jon niet uit y een oplossing voor $H(x) = y$ berekenen dus het vernemen van y helpt Jon niet (concealing).

een bewijs gegeven van de inhoud van de blob: deze operatie $Open(bl, x)$ kan slagen als de maker van de blob inderdaad x heeft ingepakt.

Het moet onmogelijk zijn om:

1. Aan een blob bl te zien als welke x hij kan worden geopend.
   De blob is dan *verbergend* ofwel *concealing*.
2. Een als $Inpak(x)$ gemaakte blob te openen als $x' \neq x$.
   De blob is dan *bindend* ofwel *binding*.
3. Een blob bl te maken die met twee verschillende waarden kan worden geopend.

De verbergende en bindende eigenschap kunnen in een bepaalde implementatie computationeel of perfect aanwezig zijn, maar niet beide perfect.

Blobs kunnen met een hashfunctie H worden geïmplementeerd door als $Inpak(x)$ te nemen de waarde $H(x)$. Het openen van blob bl met x bestaat dan uit het overleggen van x, waarna de tegenpartij controleert dat $H(x)$ inderdaad bl is.

Je kunt met een blob ook iets ingewikkelder dingen doen, bijvoorbeeld een digitaal kraslot maken. Er zijn dan een aantal prijzen, bijvoorbeeld *Niets*, *BigMac*, *Cola* of *Droomreis*. Alice prepareert de loten en verkoopt ze, maar wil natuurlijk niet dat iemand voordat hij een lot koopt kan zien wat ermee wordt gewonnen. Bob wil een lot kopen en dan zien wat hij gewonnen heeft, maar eist dat Alice na het kopen niet meer de inhoud van het lot kan veranderen. Ook hier biedt een hashfunctie uitkomst, maar omdat er maar vier verschillende prijzen zijn kan x

niet worden ingepakt door $H(x)$ te berekenen. Door de verschillende mogelijkheden te hashen zou Bob kunnen zien wat er in elk lot zit.

Alice pakt waarde x in door een random getal r te kiezen en als blob te berekenen $bl = H(r \cdot x)$. Het random getal moet zij bewaren. Het openen van bl als x bestaat uit het overleggen van een string r zodanig dat $bl = H(r \cdot x)$. Als Alice r kwijtraakt kan zij later niet meer bewijzen wat er in de blob is ingepakt en Bob zal niet met minder dan een *Droomreis* genoegen nemen.

7.4 Blinde handtekeningen

Een handtekening wordt meestal gezet om aan te geven dat men het ergens mee eens is of dat men iets heeft gezien. De berekening van de handtekening door Alice vereist het bericht M en de geheime sleutel x en dit lijkt te impliceren dat Alice M heeft gezien. Deze sectie behandelt de *blinde handtekening*, waarbij Bob Alice een bericht laat ondertekenen zonder dat Alice ziet wat ze tekent, en geeft enkele andere toepassingen van het hiervoor gebruikte primitief *invoerblindering*.

De blinde handtekening is als het ondertekenen van een document in een enveloppe van carbonpapier. Op het document verschijnt de handtekening en die bewijst dat er bij het totstandkomen interactie met Alice was, maar de enveloppe heeft voorkomen dat Alice zag wat ze tekende.

7.4.1 Motivatie

In een elektronisch betaalsysteem kan geld worden gerepresenteerd door digitale objecten (tokens), elk met een registratienummer en een waarde (hoofdstuk 13). Om het bijmaken van tokens tegen te gaan moet elk token worden voorzien van een handtekening van de bank, en uiteraard gaat een dergelijke tokenopname vergezeld van het verlagen van de bankrekening. Een digitale winkelier krijgt het bedrag bijgeschreven bij inlevering van het token.

Een betaalsysteem is *anoniem* als het niet mogelijk is een betaling te traceren naar een individu. Consumenten in de digitale economie hebben een gerechtvaardigd verlangen naar de mogelijkheid anoniem te betalen. De registratienummers in de tokens staan anonimiteit in de weg, omdat een ingeleverd token aan de hand van het serienummer kan worden verbonden met het individu dat het heeft opgenomen.

Het is niet mogelijk om tokens anoniem op te nemen, immers het bedrag van de tokens moet bij opname worden afgeschreven van de rekening. Een anoniem betaalsysteem gaat daarom altijd uit van een procedure waarmee tokens kunnen worden betrokken van de bank zonder dat de bank het serienummer ziet. Hiertoe vormt de geldopnemer *zelf* een token en verkrijgt hij hiervoor de handtekening van de bank, zonder dat hij het serienummer op het token laat zien.

Een andere toepassing, namelijk geheime verkiezingen, volgt in Sectie 14.3.3. Daar laat een kiezer een ingevuld stembiljet ondertekenen zonder te laten zien waarop hij heeft gestemd.

Uit de voorbeelden blijkt al dat er bij het blind ondertekenen altijd sprake is van interactie tussen twee partijen, namelijk een die het bericht kent en een die over de geheime sleutel beschikt.

7.4.2 Implementatie door blindering

Bij het RSA-systeem bestaat de handtekening uit het trekken van een e-demachtswortel en in hoofdstuk 5 hebben we al gezien dat hiervoor een elementair protocol bestaat dat uitgaat van *invoerblindering*. De blindering is uiteraard zichtbaar in de handtekening, maar kan door de andere partij ongedaan worden gemaakt. Laat Bob beschikken over een bericht M en Alice over een RSA-sleutelpaar (n, d) waarvan Bob het publieke deel (n, e) kent.

Bob		Alice
Kies random k; $b := k^e$	1	
$M' := M.b$; zend M'	2	Ontvang M'
Ontvang y'	3	Zend M'^d
Bereken $y = y'.k^{-1}$	4	

Bob beschikt nu over een y die voldoet aan $y^e = M$, dus een geldige handtekening voor M. De blindering is *perfect veilig* omdat Alice alleen M' te zien krijgt en

$$\forall M' : \forall M : \exists k : M' = M.k^e$$

(namelijk $k = (M'.M^{-1})^d$). Bobs belangen worden door het protocol goed beschermd.

Kan Bob Alice bedriegen door de verkregen handtekening y' voor *twee berichten* M_1 en M_2 te gebruiken? Het omrekenen van y' naar y is mogelijk doordat Bob een k kent waarvoor $M' = M.k^e$. Wanneer Bob zijn y' wil omrekenen naar een y_1 (geldig voor M_1) *en* naar y_2 (geldig voor M_2) dan moet hij dus zowel een k_1 kennen met $M_1 \cdot k_1^e = M'$ en een k_2 met $M_2 \cdot k_2^e = M'$. Dit kan als hij beschikt over een k_0 met $k_0^e = M_2 \cdot M_1^{-1}$ want dan kan hij k_1 random kiezen en $k_2 = k_1 \cdot k_0$ nemen. Omgekeerd impliceert het kennen van zulk een k_1 en k_2 ook het kennen van een k_0 met $k_0^e = M_2 \cdot M_1^{-1}$ want $k_2 \cdot k_1^{-1}$ voldoet. De mogelijkheid om y' voor twee berichten M_1 en M_2 te gebruiken is dus gebonden aan het vinden van berichten waarvan de wortel uit het quotiënt bekend is.

7.4.3 Controle op inhoud

Bij toepassingen van blinde handtekeningen moet steeds voor de ondertekenaar verborgen blijven welk token wordt ondertekend. Aan de andere kant wil de ondertekenaar wel zeker zijn dat een token van het juiste soort wordt ondertekend. Alice wil zeker zijn dat er in de door Bob aangeboden enveloppe een tientje zit en geen honderdje.

Oplossing 1: Verschillende sleutels. Alice gebruikt voor de verschillende waarden van tokens verschillende sleutels: verschillende handtekeningen dus. Als Bob een tientje opneemt zet Alice haar 'tientjeshandtekening' op de door Bob aangeboden enveloppe, en wat daar ook in mag zitten: het ondertekende token is nooit meer dan een tientje waard.

Oplossing 2: Cut-and-choose. Alice laat Bob twintig enveloppes met een token maken, laat hem er ter controle negentien openmaken en tekent blind de twintigste als de eerste negentien in orde zijn. De vorm van een token is bijvoorbeeld de string $\langle\, \mathbf{munt}, b, r \,\rangle$, waar b de waarde en r het serienummer is.

1. Bob biedt twintig getallen m_1 tot m_{20} aan.
2. Alice kiest een index i en vraagt negentien getallen k_j voor $j \neq i$.
3. Alice checkt dat m_j/k_j^e van de vorm $\langle\, \mathbf{munt}, 10, r \,\rangle$ is.
4. Als het klopt geeft ze m_i^d.

Kader 7.5: KAARTEN VIA HET INTERNET.

Alice en Bob hebben zin in een spelletje pesten maar bevinden zich aan verschillende kanten van de aardbol en het spel moet dus via informatieuitwisseling worden gespeeld. Alice schudt de 54 kaarten door ze met 54 random strings aan te vullen en te encrypten met haar eigen public key; de omgekeerde kaart k_i is dus $y_i = (r_i \cdot k_i)^e$. Bob beheert en deelt de pot.

Bob neemt zelf kaarten van de pot door zeven y_i te kiezen en blind te laten decrypten door Alice. Hij geeft Alice zeven andere strings door haar zeven van de y_i terug te sturen. Net als bij een kartonnen kaartspel kan Bob niet zien welke kaarten hij geeft. Bob speelt een kaart uit door k_i samen met r_i te overleggen; voor Alice bewijst de kennis van r_i dat deze kaart inderdaad onder de door haar geopende kaarten zat. Alice speelt een kaart uit door k_i en r_i te overleggen, waarna Bob kan controleren dat het getal $y = (r_i \cdot k_i)^e$ onder de getallen zit die hij Alice heeft gegeven.

Bobs enige kans op valsspelen is om één verkeerd token tussen de twintig te stoppen in de hoop dat dit token zal worden getekend. Er is altijd een kans van $\frac{19}{20}$ om betrapt te worden, en uiteraard staat op valsspelen een zware boete!

Oplossing 3: Zero-knowledge proof. De perfecte veiligheid van blindering impliceert dat het aangeboden token M' in feite alles kan zijn: voor *elke* M, dus ook die van ongeoorloofde vorm, bestaat er een b zo dat $M' = b.M$. Voor elke M kan Bob die blinder b ook berekenen, maar voor het verwijderen ervan heeft hij de e-demachtswortel nodig en we nemen steeds aan dat Bob maar voor één (relevante) blinder die wortel kent. Eerlijkheid van token M' volgt dus uit *kennis van Bob* (en niet het bestaan) van een getal k zodanig dat M'/k^e van de juiste vorm is $(\langle \mathbf{munt}, 10, r \rangle)$.

Het is mogelijk deze kennis van Bob te verifiëren met een geavanceerde wiskundige techniek genaamd *zero-knowledge proof*; zie sectie 8.4.1.

7.4.4 Blindering en blinde overdracht

In cryptografische protocollen is er soms de noodzaak dat Bob enige informatie van Alice verkrijgt zonder dat Alice precies weet welke informatie dat is (zie ook kader 7.5). Dit probleem staat bekend als blinde overdracht (*oblivious transfer*) en wordt verder besproken in sectie 14.4.1.

Het volgende eenvoudige protocol geeft Bob een kans van exact een half om Alice' geheim te weten, en Alice weet na afloop niet of Bob het geheim kent. Het geheim van Alice is een tweetal priemgetallen p, q en Bob kent het product.

Bob		Alice
Kies random $x \in \mathbb{Z}_n$; zend $a = x^2$	1	Ontvang a
Ontvang x'	2	Zend x', een wortel van a
Probeer of ggd$(x + x', n) > 1$	3	

Alice kan allevier de wortels van a uitrekenen, maar heeft geen manier om te weten welke x Bob al kent. Geeft ze x of $-x$ dan weet Bob nog niets en anders vindt Bob de factoren. Alice weet achteraf niet of Bob n heeft kunnen factoriseren of niet.

Het protocol voor een blinde RSA-ondertekening kan ook worden gebruikt voor een *blinde ontsleuteling* en hieruit kan een zeer algemeen protocol voor oblivious transfer worden afgeleid. Bij blinde ontsleuteling beschikt een partij over een decryptiesleutel en een andere over een cijfertekst; deze partij moet de klare tekst te weten komen maar niet de sleutel.

Blinde overdracht kan als volgt opgelost worden. Alice heeft s geheimen x_1 tot x_s; ze gebruikt een RSA-sleutel en geeft Bob y_1 tm y_s waar $y_i = x_i^e$ (ze versleutelt dus haar eigen geheimen). Bob kan een van de geheimen, zeg x_i, openen door y_i te laten ontsleutelen door Alice, en door dit blind te laten doen blijft voor Alice verborgen welk geheim Bob heeft geopend. In dit protocol kan Bob kiezen welk geheim hij opent, maar wanneer dit onwenselijk is kan Alice haar geheimen willekeurig permuteren (als in kader 7.5).

Toepassing: discrete geheimenwinkel. Alice drijft op het internet een handeltje in informatie van het soort waar mensen graag iets voor betalen: mooie plaatjes, handige weetjes, leuke programmaatjes. Volgens haar catalogus kost elk soort informatie precies evenveel, namelijk 10 euro. Bob heeft wel interesse in een van de mooie stukjes informatie, maar wil niet dat Alice te weten komt in welke informatie hij geïnteresseerd is. Geen probleem: Alice verkoopt haar geheimen met verzekerde discretie.

De bestanden G_1 tot G_s worden elk met een symmetrisch algoritme versleuteld maar met verschillende sleutels x_1 tot x_s. Alice plaatst de aldus versleutelde bestanden op haar website, samen met de RSA-encrypte sleutels $y_i = x_i^e$ (en haar catalogus). De afrekening van de geheimen verloopt bij Alice zo: je kunt er voor 10 euro een e-demachtswortel modulo n kopen, en wat je ermee doet laat Alice koud. Bob kan geheim i kopen door de wortel uit y_i te vragen, maar gezien zijn geheimzinnigheid over zijn interesses kan hij er ook voor kiezen y_i eerst te blinderen.

Samenvatting en conclusies

De digitale handtekening verzorgt de authenticiteit en onloochenbaarheid van informatie; dit betekent dat de herkomst van de informatie later met zekerheid is te controleren voor alle partijen. Ze worden bijvoorbeeld gebruikt in sleutelcertificaten en elektronisch geld, maar ook voor het ondertekenen van e-mails.

De bekendste implementaties zijn de RSA-handtekening en de ElGamal-handtekening, afgeleid van de gelijknamige cryptosystemen. Bij RSA gaat het verifiëren van een handtekening veel sneller dan het zetten, bij ElGamal is het verifiëren juist iets duurder. Wel kunnen bij ElGamal de handtekeningen korter zijn, waardoor dit algoritme met name in bijvoorbeeld smartcards populair is.

Om een langer bericht te ondertekenen wordt gebruik gemaakt van een hashfunctie die elk bericht afbeeldt op een bitrij van vaste lengte. Het is onmogelijk om uit de bitrij het bericht

terug te vinden of om twee berichten te vinden die op dezelfde bitrij worden afgebeeld. In plaats van het complete bericht wordt dan alleen deze bitrij (de hashwaarde) ondertekend.

Soms wordt blindering gebruikt in combinatie met handtekeningen. Een partij zet dan een handtekening onder een bericht dat eerst door een tweede partij onherkenbaar is gemaakt, en de ondertekenaar weet dan niet wat hij heeft ondertekend. Blindering wordt ook in andere cryptografische protocollen gebruikt, bijvoorbeeld om de privacy van een partij te garanderen.

Opgaven bij hoofdstuk 7

Opgave 7.1 *Alice en Bob gebruiken het protocol uit kader 7.4 om een random keuze te maken; laten we aannemen dat Bob belang heeft bij een true en Alice bij een false.*

(a) Bewijs dat als Alice en *Bob hun keuze uniform maken (ze kiezen elk voor even of oneven met kans een half), de uitkomst ook uniform is.*
(b) Bewijs dat als een van de twee *de keuze uniform maakt en onafhankelijk van de ander, de uitkomst uniform is.*
(c) Kan een partij 'smokkelen' door de eigen keuze niet uniform te maken?

Opgave 7.2 *Bewijs dat de existentiële vervalsing bij ElGamal (sectie 7.2.3) inderdaad een geaccepteerde handtekening oplevert.*

Opgave 7.3 RSA Handtekeningen. *(Uit het tentamen van maart 1999.)*
Hoe zet een partij zijn RSA handtekening onder string x?
Men zegt dat bij dit systeem een Existential Forgery mogelijk is. Wat is dat en wat kun je eraan doen?

Opgave 7.4 *In de verificatie van de ElGamal-handtekening ($b^c \cdot c^d = g^M$) wordt c tweemaal gebruikt, maar de eerste keer is de waarde modulo* $\mathrm{ord}(g)$ *van belang en de tweede keer de waarde modulo p. Laat zien hoe je een universele vervalsing kunt berekenen als het getal c in het bereik* $0 \ldots p \cdot \mathrm{ord}(g)$ *mag liggen.*

Opgave 7.5 *Bewijs dat bij ElGamal-handtekeningen de geheime sleutel is te vinden uit twee getekende berichten als het quotiënt van de twee gebruikte random getallen bekend is (dat wil zeggen, een getal r is bekend zodat $k_2 = r.k_1$ geldt).*

Opgave 7.6 *Bewijs propositie 7.6. Laat zien dat het belangrijk is dat de lengte van het bericht in het laatste, en niet het eerste blok wordt opgenomen.*

Opgave 7.7 *Bewijs dat de blokfunctie van Chaum e.a.* one-way *blijft ook met kennis van* $\mathrm{ind}(h)$.

Opgave 7.8 *Maak een blobschema dat perfect verhullend en computationeel bindend is.*
Maak een blobschema dat computationeel verhullend en perfect bindend is.
Bewijs dat er geen blobschema bestaat dat perfect verhullend en perfect bindend is.

Opgave 7.9 Blind Signatures. *(Uit het tentamen van januari 1997.)*
Een instituut ondertekent berichten met RSA (de publieke sleutel is (n, e)).

Laat zien hoe Alice een bericht voor ondertekening kan blinderen en hoe ze de handtekening kan deblinderen om over een getekend, maar onbekend bericht te beschikken.

Welke drie methoden zijn er om te garanderen dat er een bericht van de juiste inhoud wordt ondertekend en hoe werken ze?

Opgave 7.10 *Alice gebruikt cut-and-choose bij de uitgifte van tokens maar haar protocol bevat een implementatiefout. Er wordt voor negentien van de geblindeerde tokens niet gevraagd om het random getal k_j, maar om de blinder b_j. (In stap 3 controleert Alice natuurlijk m_j/b_j.) Laat zien hoe Bob van deze misser gebruik kan maken en dat hij voor 10 euro een hoger gewaardeerd token kan verkrijgen.*

Opgave 7.11 *Bob probeert het blinderingsschema te misbruiken door een tientje en een honderdje te zoeken met een quotiënt waarvan hij de wortel weet.*

(a) Bob legt een lijst aan van vele k_x met bijbehorende $b_x = k_x^e$, vele $t_y = \langle\, \mathbf{munt}, 10, r_y \,\rangle$ en $h_z = \langle\, \mathbf{munt}, 100, r_z \,\rangle$. Hoe lang moeten de lijsten zijn om x, y, z te bevatten met $b_x = t_y h_z^{-1}$?
(b) Hoe kun je die x, y, z zo snel mogelijk vinden?
(c) Hoe kun je de gevonden getallen gebruiken om 90 euro te verdienen?

Opgave 7.12 *Bij pesten moet na een tijdje een nieuwe pot worden gemaakt door de al uitgespeelde kaarten opnieuw te schudden. Kan dat ook bij pesten over het internet (kader 7.5)?*

Opgave 7.13 *Geef een cryptografisch protocol voor het eerlijk werpen van een dobbelsteen. (Hint: dit kan met blobs als primitief en het protocol is een generalisatie van de eerlijke muntworp.)*

Hoofdstuk 8

Identificatie

Het ondubbelzinnig kunnen vaststellen van de identiteit van een partij is een van de meest centrale toepassingen van de cryptografie. Identificatie is nodig bij het beschikken over informatie of banktegoeden, en voor toegang tot computersystemen, gebouwen en landen. Hierbij hoeft identificatie niet altijd een koppeling met naam (en adresgegevens) te betekenen, maar het kan ook een koppeling zijn met een eerder contact met deze partij.

Het belangrijkste cryptografische mechanisme voor identificatie is het *zero-knowledge proof*, besproken in sectie 8.3. Allereerst wordt bekeken aan welke eisen zo'n protocol moet en kan voldoen, aan de hand van enkele bestaande systemen wordt de samenhang van de diverse eisen en technieken duidelijk gemaakt (sectie 8.1). De zero-knowledge proofs hebben nog verdere toepassingsmogelijkheden en hierover gaat sectie 8.4. Sectie 8.5 belicht enkele basisbegrippen uit de computergebaseerde biometriek die ook voor identificatie (van personen) wordt gebruikt.

8.1 Definities en aanvallen, voorbeelden

De interactie waarmee partij Alice haar identiteit bewijst tegenover partij Bob noemen we een *protocol*. Alice en Bob wisselen enkele boodschappen uit, waarna Bob ervan overtuigd is dat hij inderdaad met Alice converseert.

Classificatie van protocollen. Afhankelijk van de mate van bescherming die deze protocollen bieden worden ze, volgens Fiat en Shamir [FS86], in drie groepen verdeeld.

1. **Authenticatieprotocol:** Alice kan tegenover Bob haar identiteit bewijzen, anderen kunnen dat niet.
2. **Identificatieprotocol:** Alice kan tegenover Bob haar identiteit bewijzen, anderen kunnen dat niet, en Bob kan zich niet als Alice voordoen tegenover anderen.
3. **Handtekeningschema:** Alice kan tegenover Bob haar identiteit bewijzen, anderen kunnen dat niet, en Bob kan zich niet als Alice voordoen tegenover anderen en zelfs niet tegenover zichzelf.

Het verschil tussen authenticatie en identificatie is dat in het eerste geval de mogelijkheid bestaat dat Bob de informatie die hij bezit om Alice te herkennen, misbruikt om zich bij een

andere partij als Alice voor te doen. De extra eis die voor een identificatieprotocol geldt sluit dit uit.

Het verschil tussen identificatie en handtekening is subtieler en zit in de vraag of Bob later tegenover derden kan aantonen met Alice te hebben gesproken. Bij de identificatieprotocollen uit sectie 8.3 kan Bob zich weliswaar niet *tegenover anderen* als Alice voordoen, maar wel kan hij een complete 'echte' conversatie met Alice *zelf genereren*. Daarom bewijst Bobs afschrift van deze conversatie niet dat hij inderdaad met Alice heeft gecommuniceerd. De extra eis in een handtekeningschema sluit uit dat Bob een conversatie uit het protocol genereert zonder dat Alice erbij is. In zo'n schema kan Bob met de informatie die hij van Alice heeft gekregen wel bewijzen dat hij met Alice heeft gesproken.

Zoals de term 'handtekeningschema' al doet vermoeden is er verband tussen deze vorm van identiteitscontrole en digitale handtekeningen. Een methode om berichten te ondertekenen (als in hoofdstuk 7) kan altijd voor identiteitscontrole worden ingezet. Bob laat Alice een bericht tekenen en kan dan zelf aan de handtekening zien dat hij met Alice converseert, maar hij kan de handtekening ook bewaren om dit achteraf te bewijzen.

In het vervolg wordt hoofdzakelijk over identificatie gesproken.

Aanvallen op de identificatie. Tegenstander Oscar kan zich met de conversatie bemoeien en heeft daarbij een van deze doelen:

1. **Sleutelinformatie:** Oscar probeert achter de geheime informatie (of een deel daarvan) van Alice te komen.
2. **Impersonificatie:** Oscar probeert het protocol met Bob uit te voeren met als resultaat dat Bob gelooft dat hij Alice is.

De werkwijze van Oscar kan worden beschreven aan de hand van de mate van inmenging in de conversatie.

1. **Passieve aanval:** Oscar luistert een conversatie tussen Alice en Bob af.
2. **Actieve aanval:** Oscar gaat zelf eerst enkele conversaties aan met (meestal) Alice en/of Bob.
3. **Coalitie:** Oscar heeft medewerking van Alice en/of Bob en heeft daarom toegang tot hun geheime informatie.

Coalities met Bob kunnen zinvol zijn bij authenticatieprotocollen waarmee op meerdere punten kan worden aangemeld. Oscar steelt het slot van Alice' achterdeur, leest de informatie eruit en kan daarmee de voordeur binnen. Bij een identificatieprotocol helpt een coalitie met Bob niet om een impersonificatie te doen.

Een coalitie met Alice lijkt niet voor de hand te liggen, toch zijn er vele situaties waarin een gebruiker een ander helpt met de impersonificatie. Bankpasjes plus pincode en computerpasswords worden bijvoorbeeld uitgeleend aan huisgenoten. Verder is illegale immigratie tegenwoordig een omvangrijke tak van de criminele industrie met een jaaromzet van tien miljard dollar. Een van de werkwijzen is dat een medewerker van de organisatie zijn paspoort afstaat en daarmee een ander het land binnen laat gaan.

Een coalitie met Alice is niet met behulp van cryptografie tegen te gaan, immers, Alice kan alle geheimen die zij heeft ten behoeve van identificatie overgeven aan Oscar. Dit betekent dat waar coalities ongewenst zijn, cryptografie met andere middelen moet worden gecombineerd. Soms worden identificatiepunten (geldautomaten bijvoorbeeld) van videocamera's voorzien om

Kader 8.1: Hoe Bob Evers zich identificeert

Willy van der Heide schreef in de jaren vijftig de jongensboekenserie *Bob Evers*, waarin Bob Evers, Jan Prins en Arie Roos de strijd aanbinden tegen dieven, smokkelaars, muiters en goudzoekers. Uiteraard zijn bij deze avonturen diverse methoden van identificatie nodig, en hiervoor worden voorwerpen, geheimen en kenmerken gebruikt.
In *Cnall-Effecten in Casablanca* gaat het drietal in Afrika zoeken naar agent Cnall die daar honderdduizend dollar moet afgeven aan een persoon die beschikt over een passende helft van een doorgescheurde folder (voorwerp). "Perfect," merkt Bob hierover op (p. 22), "Niemand heeft ooit nog een manier ontdekt om twee verschillende stukken papier zo doormidden te scheuren, dat je de helften kunt omwisselen. Er is altijd maar één helft die aan zijn wederhelft past." Gelukkig had men in die tijd ook nog geen papier ontdekt dat niet kan worden afgepakt, anders was het boek niet spannend!
Een minder perfecte methode gebruiken boeven in *Vreemd Krakeel in Californië*: contact wordt gelegd door te vragen om een Hollandse krant. Deze identificatie-methode heeft een veel te hoge *false acceptance rate*, want prompt wordt Arie na het kopen van een krant met enkele anderen ontvoerd.
In *Kabaal om een Varkensleren Koffer* figureert de Groene Man met zijn opvallende persoonskenmerken: gekleed in een lichtgroen zomerpak met lichtgroen overhemd en groene das en rijdend in een groene twoseater. In een hotelkamer met lichtgroene wanden, donkergroen zeil, een bleekgroene beddesprei en groene stoelen steelt hij de Koffer. Maar (p. 63): "Als wij op zoek gaan naar een Groene Man met een groene Austin Healey.... dan zijn wij groentjes, want volgens mij is al dat groen met opzet gedaan."
Website `www.bobevers.nl`.

conflicten achteraf te kunnen beslechten. Ook kan de identificatie van biometrie gebruikmaken. Hierover handelt sectie 8.5.

8.1.1 Geheimen, tokens, kenmerken

De diverse mogelijkheden die er voor identificatie zijn worden nu besproken, in historische of maatschappelijke context. Voor het identificeren van een partij kunnen veel verschillende methoden worden gebruikt, maar vrijwel alle van oudsher gebruikte methoden kunnen in drie

Kader 8.2: De Vasco Identikey

Deze Vasco Identikey is een identificatiesysteem dat een combinatie gebruikt van *kennis* en *voorwerp*. De sleutelhanger kan een challenge-responsprotocol uitvoeren, ook via het internet, waardoor de tegenpartij de zekerheid verkrijgt met dit exemplaar te communiceren. Echter, voordat de Identikey bereid is zijn kunsten te vertonen moet hij worden ontsloten door het intypen van een vijfcijferige pincode.
Om het gebruik via internet van het token te vergemakkelijken is hij voorzien van lichtgevoelige cellen waarmee het de challenge van een beeldscherm kan lezen.
De Identikey wordt veel gebruikt bij telebankieren. Het getoonde exemplaar is afkomstig van de SnS-bank.
Zie `www.vasco.com`.

groepen worden ingedeeld. Er is sprake van *bezit van een voorwerp*, *kennis van een geheim*, of *persoonsgebonden kenmerken*. De drie groepen van technieken verschillen in de mate waarin de toegang, al dan niet vrijwillig, kan worden overgedragen. Daarom worden vaak combinaties van deze drie groepen gebruikt, bij een pinbetaling bijvoorbeeld het *bezit* van de kaart en het *kennen* van de viercijferige code.

Een keuze voor een van de drie strategieën of een combinatie ervan is doorgaans niet voldoende om een veilig systeem te verkrijgen. Het controleren van de echtheid van aangeboden voorwerpen, de juistheid van overlegde geheimen, de binding van geheimen of kenmerken aan partijen: het moet allemaal worden geregeld en veel hiervan kan met cryptografische ondersteuning worden gedaan.

Gebruik van een geheim of password. Het idee om iemand toegang te geven op basis van kennis van een geheim is ouder dan het gebruik van computers: al in de oudheid vroegen soldaten die een stad bewaakten naar een wachtwoord aan ieder die toegang verlangde. Omdat de poortwachter zelf het wachtwoord moest kennen en dus zelf toegang kon krijgen, spreken we van *authenticatie*. De controle door een persoon zorgde hierbij altijd voor een combinatie met andere kenmerken: iemand van het vijandige volk kon bijvoorbeeld door een verkeerde uitspraak (van *Scheveningen*) door de mand vallen. In feite is hiermee een primitieve koppeling met een biometrisch kenmerk gemaakt.

De opkomst van computersystemen, waar op verschillende plaatsen kon worden ingelogd, maakte het nodig deze vorm van toegangscontrole opnieuw te bekijken. Computerterminals waren immers primair voor het intypen en uitlezen van informatie gemaakt, en er is geen extra hardware nodig wanneer de controle slechts op basis van informatie-uitwisseling gebeurt. Daarom kan er meestal op werkstations worden ingelogd met uitsluitend een password, zonder token (een materiële sleutel of kaart).

Al spoedig werd de mythe gevormd dat toegang op basis van een password eigenlijk veiliger is dan op basis van een voorwerp (contactsleutel voor computer bijvoorbeeld). De argumenten kwamen erop neer, dat een voorwerp kan worden verloren of gestolen en dan door een ander

worden gebruikt, en dat dit bij een geheim password niet kan. Resterende twijfel werd weggenomen met het argument dat 'het onmogelijk is alle mogelijkheden te proberen' en dat het 'dus' niet mogelijk is zonder password toegang te krijgen.

Deze redenering is natuurlijk niet steekhoudend en was dit vanaf het begin niet. Het is juist (te) gemakkelijk om een geheim te raden, maar daarnaast kun je het stelen, afluisteren, afpersen of kopen en geheimen zijn ook vrijwillig overdraagbaar (coalitie). De banken geven altijd erg hoog op van de veiligheid van hun eigen systemen, maar misbruiken deze laatste mogelijkheid door hun klanten op te laten draaien voor schade door slechte beveiliging. Wie zich namelijk beklaagt over een zogenaamde spookopname krijgt steevast te horen dat het systeem heel veilig is, en de klant 'dus' zelf onzorgvuldig is geweest met de pincode. Bijvoorbeeld door deze aan een huisgenoot mee te delen, die dan direct als verdachte wordt aangemerkt.

De overdracht van een geheim is niet meer ongedaan te maken: kent een ander eenmaal het geheim dan kan het hem niet meer worden afgenomen.

Bezit van een voorwerp. Zoals het voorbeeld van de bank al aangeeft, zijn er ook veel systemen die juist wel gebruik maken van een materieel voorwerp, bijvoorbeeld de bankpas (al dan niet in combinatie met kennis van een geheim) of de ouderwetse huissleutel. Een voorwerp kan worden gestolen of nagemaakt, waarbij nog onderscheid kan worden gemaakt naar de mate van controle die de namaker hiervoor moet hebben over het originele voorwerp.

Een zegelring om wetten van Meden en Perzen te tekenen (zie Ester 3:10 in de bijbel) is heel gemakkelijk na te maken zonder de ring ooit te zien: een stempel kan worden gemaakt van één ondertekende wet. Een sleutel (voorwerp) kan worden nagemaakt vanaf een wasafdruk van een echte sleutel: 'uitlenen is weggeven' is het idee. Iemand die een sleutel leent kan daarom na het teruggeven nog steeds over (een kopie van) de sleutel beschikken.

Een smartcard die wortels trekt modulo een geheim, samengesteld getal kan worden nagemaakt door hem enkele wortels uit door de namaker gekozen getallen te laten trekken (met lemma 6.12). Zo'n kaart is in uitleenbaarheid met een deursleutel te vergelijken. Met een smartcard die hogeremachtswortels trekt is dit (waarschijnlijk) niet mogelijk. Als de kaart is uitgeleend mag men na het teruggeven veronderstellen dat de tijdelijke bezitter niet langer over het voorwerp beschikt.

Een voorbeeld van een smartcard die in combinatie met een geheim wordt gebruikt staat in kader 8.2.

Aanvallen op codes en passen. Wanneer toegang mogelijk is via een combinatie van een voorwerp en een geheim zullen aanvallers dus ook zoeken naar combinaties van stelen of namaken en afluisteren, raden of afpersen. De pinpas is een voorbeeld van een systeem waarbij de code (pin, persoonlijk identificatie nummer) is gecombineerd met bezit van een voorwerp, namelijk de betaalpas zelf. De mogelijkheid van het afkijken van de pincode (passieve aanval) is de primaire zwakte van deze passen en moet dus met het stelen of namaken van de pas worden gecombineerd. Beide werden in Nederland gedaan.

Zo is geconstateerd dat dieven vaseline op de cijfertoetsen van een geldautomaat smeerden om na een opname te kunnen zien welke van deze toetsen waren gebruikt. Ook zijn wel kleine camera's ontdekt in kaartjesautomaten op stations: juist de afscherming die het afkijken moest voorkomen bood een prima dekking voor de camera's. In deze gevallen, die worden gerangschikt onder *passieve* aanvallen, werd van gebruikers van een pas de bankpas na afloop gerold.

Een meer geavanceerde, actieve aanval gebruikte nepbetaalautomaten in snoep- en sigarettenautomaten op luchthavens. Kooplustigen voerden hun pas en de pincode in, waarna de

informatie op de magneetstrip met de code werden opgeslagen. Magneetpassen zijn eenvoudig na te maken door de afgelezen informatie weer op een andere ('witte') pas te schrijven.

In deze gevallen was misbruik mogelijk door het afkijken (en hergebruiken) van de code. Het is niet duidelijk in hoeverre criminelen kunnen frauderen door bij een gestolen pas zelf een correcte PIN te genereren. Anderson [And94] rapporteert dat dit in Groot-Brittannië werd gedaan in samenwerking met corrupt bankpersoneel.

8.2 Authenticatie op basis van cryptografie

Deze sectie bespreekt enkele authenticatietoepassingen die zijn gebaseerd op symmetrische cryptografie.

8.2.1 Kennen en herkennen: UNIX

In een computersysteem is het ongewenst dat passwords worden opgeslagen omdat dan een partij die eenmaal onrechtmatig is binnengedrongen zo over meer en echte passwords kan gaan beschikken. Opslag van passwords werd al in UNIX voorkomen door gebruik te maken van een one-wayfunctie H (hash). Voor een gebruiker met password a werd de waarde $y = H(a)$ opgeslagen: deze y kan niet voor inloggen worden gebruikt. Uiteraard wordt bij inloggen met password a' de waarde $H(a')$ vergeleken met y, en kennis van y is niet voldoende om een succesvolle a' te berekenen.

Wel is het steeds nodig dat een gebruiker bij het inloggen zijn password intypt. Het is vrij gemakkelijk om dit intypen af te kijken en ook werden er wel speciale programmaatjes geschreven die de ingetypte tekens van het toetsenbord naar een andere partij doorstuurden (*password snooping*).

8.2.2 One-time passwords: S-key

Password snooping is net als het afkijken van pincodes gebaseerd op het hergebruik van een password. Met name bij inloggen via een netwerk is het gevaar van het afluisteren van het password reëel. Het *S-keysysteem* sluit hergebruik van passwords uit doordat elk password maar een keer kan worden gebruikt: voor een volgende aanmelding van dezelfde gebruiker is een ander password nodig. Het benodigde geheugen wordt in de hand gehouden doordat alle passwords uit eenzelfde startwaarde worden afgeleid door de hashfunctie herhaald toe te passen. Een aangeboden password a wordt gecontroleerd door $H(a)$ te vergelijken met de opgeslagen y, maar hierna wordt y door a vervangen.

Preciezer gezegd verloopt het installeren van gebruiker Alice zo:

1. Alice kiest een geheim A.
2. Alice itereert H duizendmaal: $y = H^{1000}(A)$.
3. y wordt in de computer (Bob) opgeslagen en Alice houdt A.

Het (remote) inloggen gaat zo:

Alice		Bob
Meld met password $a = H^{1000-i}(A)$	1	Ontvang a
	2	Controleer $H(a) = y$
	3	Bij acceptatie $y := a$

Alice		Bob
Stuur "Alice"	1	Ontvang Username en zoek bijbehorende k
	2	Kies een random x
Ontvang x	3	Stuur x
Bereken $z := E_k(x)$	4	Bereken $y := E_k(x)$
Stuur z	5	Ontvang z
	6	Accepteer als $z = y$

Protocol 8.3: Symmetrisch authenticatieprotocol.

Een afgeluisterd password a kan niet worden hergebruikt omdat er de volgende keer juist moet worden ingelogd met een waarde a' die voldoet aan $H(a') = a$. Het S-keysysteem is populair voor het ondersteunen van remote login, typisch voor medewerkers van een organisatie tijdens tijdelijk verblijf in het buitenland.

S-key beschermt *niet* tegen een *actieve aanval* van het volgende soort. Oscar kijkt een keer bij Bob hoe Bobs login-prompt eruit ziet. Als Alice een internetcafé binnengaat laat Oscar de loginprompt van Bob op het scherm verschijnen. Alice typt haar password en Oscar genereert een foutmelding. Oscar beschikt nu over een ongebruikt password en kan zich tegenover Bob als Alice voordoen. Deze aanval kan alleen voorkomen worden door het password (mede) afhankelijk te laten zijn van de *tijd* of iets wat Bob vraagt (*challenge*).

De veiligste implementatie van het S-keyidee zou die zijn waarbij Alice de A onthoudt en op de plaats waar zij in wil loggen de hashfunctie itereert. Dit doet zij dan niet op ter plaatse beschikbare hardware maar op *eigen* hardware, bijvoorbeeld een sleutelhangersmartcard. Hier hoeven geen persoonlijke geheimen in te zitten, maar Alice moet er zeker van kunnen zijn dat er niet mee is geknoeid.

De praktijk is dat gebruikers de passwords *niet* ter plekke uitrekenen, maar dat bij het installeren van de gebruiker een briefje wordt gegenereerd met een flink aantal achtereenvolgens te gebruiken passwords. Uiteraard is hieraan een risico verbonden, maar de voordelen van de eenmaligheid van de passwords weegt hier doorgaans tegenop.

Met het S-keysysteem kan slechts een beperkt aantal logins worden uitgevoerd voordat de gebruiker opnieuw moet worden geïnstalleerd.

8.2.3 De chipknip

Uit de voorafgaande voorbeelden is gebleken dat het niet voldoende veilig is wanneer bij de authenticatie *kennis* van een geheim wordt gedemonstreerd door *het geven* van het geheim. De chipknip bevat een protocol waarmee het geheim in een berekening (namelijk een encryptie) wordt gebruikt waarvan het *resultaat* wordt overlegd en gecontroleerd.

In dit systeem is E een encryptiefunctie (3DES) en k het geheim van Alice, dat echter bij dit protocol ook bij Bob bekend moet zijn. De wijze waarop dit bij de chipknip gebeurt (masterkeysysteem) is al in sectie 4.4.4 uitgelegd. Alice bewijst haar kennis van k door een berekening met k te maken, maar de invoer hiervoor moet zij van Bob krijgen; anders zou het afgeluisterde antwoord kunnen worden gebruikt voor impersonificatie (als in sectie 8.2.2). De invoer die Bob geeft wordt de *challenge* genoemd en het antwoord de *respons*; zie protocol 8.3.

Alice, die zelf over k beschikt, kan de gevraagde waarde in stap 4 uitrekenen en wordt door

Kader 8.4: Authenticatie in de chipknip

Het authenticatieprotocol van de chipknip beschermt tegen passieve en actieve aanvallen van buitenaf, maar niet tegen coalitie. De masterkey in betaalautomaten maakt het mogelijk om van een willekeurige kaart x de authenticatiesleutel $E_M(x)$ te berekenen. Bovendien kan een *nieuw* serienummer worden gegenereerd *met* bijbehorende sleutel. Maar om deze fraude te voorkomen bevat elke kaart ook de RSA-handtekening van de bank over het serienummer: $y = x^d$. Door deze toevoeging is fraude met *nieuw aangemaakte* nummers vrijwel onmogelijk. Fraude moet dus worden gepleegd door het namaken van een bestaande pas waarvan eerst de RSA-handtekening is verkregen.

Bob geaccepteerd. Zonder kennis van k is het niet mogelijk een goede z terug te geven, zelfs niet als er een vorige conversatie is afgeluisterd, immers, de challenge x is iedere keer nieuw. Bij de chipknip is de identificatie tweezijdig: de kaart kiest ook een challenge en controleert de respons om van de echtheid van de betaalautomaat overtuigd te raken.

Het protocol biedt de ideale mogelijkheid voor een gekozen-klaretekstaanval. Immers, als challenge kan een gekozen x worden aangeboden waarna als respons de waarde $E_k(x)$ wordt ontvangen. De sleutellengte moet dus voldoende groot zijn om hiertegen bestand te zijn, en (enkelvoudige) DES is niet veilig.

Protocol 8.3 heeft belangrijke toepassingen (chipknip) en is verder van belang omdat het de idee van challenge en respons introduceert. Het heeft echter ook twee belangrijke nadelen, die in sectie 8.3 met zero-knowledgeprotocollen zullen worden verholpen.

1. Als Oscar het protocol met Alice (of Bob) aangaat kan hij weliswaar niet de sleutel k vinden uit de respons, maar toch in het bezit komen van de encryptie van een door hem gekozen x. Het is onbekend wat dat hem voor voordeel zou kunnen opleveren (waarschijnlijk geen enkel), maar liever zouden we kunnen *bewijzen* dat Oscar zelfs met een actieve aanval geen nuttige informatie ontvangt.

2. Het (symmetrische) protocol heeft de beperkingen van authenticatieschema's. Bob beschikt over het geheim k van Alice en dit betekent dat Bob zich tegenover andere partijen als Alice kan impersonifiëren. Het systeem is goed beschermd tegen een aanval *van buitenaf* zoals door criminelen die apparaten namaken zonder over de geheime sleutels te beschikken. Het systeem biedt geen bescherming tegen een aanval *van binnenuit*, bijvoorbeeld door betaalautomaathouder die het slechte pad op gaat. (Bij de chipknip zit deze sleutel niet in de betaalautomaat maar wordt door die automaat berekend uit het serienummer van de kaart; dit doet echter niet af aan deze zwakte.)

8.3 Zero-knowledge proof voor identificatie

Identificatie met een zero-knowledge proof gaat nog een stap verder dan protocol 8.3. Net als in protocol 8.3 bewijst Alice het geheim te kennen zonder het te laten zien en daarom spreken we van *proof*. Nu kan ook worden bewezen dat Bob *niets heeft* aan de informatie die hij tijdens het protocol van Alice krijgt (of eigenlijk: dat hij *geen informatie* krijgt) en daaraan ontleent men de term *zero-knowledge*.

Er worden hier drie identificatieprotocollen behandeld, die sterk op elkaar gelijken en als basis een bepaald soort one-wayfunctie f hebben. Het schema van Feige, Fiat en Shamir gebruikt voor f het kwadrateren modulo een samengesteld getal (als in Rabins cryptosysteem, sectie 6.3.5). Het schema van Schnorr gebruikt voor f de exponentiatie met vast grondtal (als in ElGamals cryptosysteem, sectie 6.3.2). Het schema van Gilliou en Quisquater gebruikt als f de exponentiatie met vaste exponent (als in het RSA-algoritme, sectie 6.3.3).

In alle gevallen beschikt Alice over een geheim getal a en Bob over een publiek getal b gelijk aan $f(a)$. Steeds beweert Alice voor een aantal getallen x de waarde $f^{-1}(x)$ te kennen, waarna Bob een getal mag uitkiezen waarvan Alice de inverse moet geven. Bob accepteert Alice als het antwoord klopt. Over de correctheid worden steeds drie uitspraken gedaan.

1. **Acceptatie.** Als Alice het protocol volgt wordt zij geaccepteerd.
   Dit wordt bewezen door het narekenen van het gegeven antwoord.
2. **Weigering.** Een partij die a niet kent wordt *niet* geaccepteerd.
   Er wordt bewezen dat een partij die de inverse van *twee* van de waarden x kent, hieruit a kan berekenen. Een partij zonder kennis van a (en de mogelijkheid a uit te rekenen) kent dus ten hoogste een antwoord en heeft een kleine kans te worden geaccepteerd.
3. **Zero-knowledge.** Bob verkrijgt *geen informatie* over de waarde van a.
   We laten zien dat Bob de gehele conversatie in het protocol zelf kan genereren *zonder Alice*, dus dat de aanwezigheid van Alice hem niets leert dat hij zonder haar niet zou weten.

De drie protocollen verschillen natuurlijk in allerlei uitwerkingsdetails, waarop aan het eind van de sectie wordt ingegaan.

8.3.1 Feige, Fiat en Shamir: vierkantswortel

Als eerste wordt een vereenvoudigde versie van Feige, Fiat en Shamirs protocol beschreven, waarin Alice de kennis van een wortel eenmaal bewijst. Een gedeelde parameter voor Alice en Bob is een groot samengesteld getal n, waarvan echter noch Bob, noch Alice de factoren kent. Alice beschikt over een random $a \in \mathbb{Z}_n^*$ en Bob beschikt over $b = a^2$. Wegens aanname 6.7 is het niet mogelijk wortels modulo n te berekenen, er is dus geen enkele berekening die Bob kan uitvoeren waarmee hij a te weten komt.

Eénschotsidentificatie. Als Alice zich meldt bij Bob, zoekt deze de publieke sleutel van Alice op; zie protocol 8.5. De sleutel kan lokaal zijn opgeslagen of door Alice als certificaat worden aangeboden. Alice stuurt (stap 3) een getal s op en maakt hierdoor impliciet de claim, *wortels van s en $s.b$ te kennen.* Dit komt omdat zij het getal s heeft verkregen door een random r te kwadrateren: dus r is een wortel uit s, en dan is $r.a$ een wortel uit $s.b$.

De hele identificatie draait om de omgekeerde implicatie: *wie van zowel s als van $s.b$ een wortel kent, kent daarmee een wortel van b* (namelijk het quotiënt van die twee wortels).

Alice		Bob
Meld username	1	Zoek b behorende bij Alice
Kies random $r \in \mathbb{Z}_n$	2	
$s := r^2$; zend s	3	Ontvang s
	4	Kies random $c \in \{0, 1\}$
Ontvang c	5	Zend c
Zend $y = r.a^c$	6	Ontvang y
	7	Accepteer als $y^2 = s.b^c$

Protocol 8.5: Identificatieronde volgens Feige, Fiat en Shamir.

De stappen 4 tot en met 7 dienen precies hiervoor dat Bob ervan overtuigd raakt dat zijn tegenpartij inderdaad over de twee wortels beschikt. Het gemakkelijkst zou dit kunnen als Alice ze opstuurt, maar dan kent Bob ook *beide* wortels en dus a.

Het verloop is daarom als bij een examen, waar niet *alle* vragen over de gehele stof worden gesteld, maar slechts enkele. Professor Bob mag student Alice één vraag stellen, maar omdat hij mag kiezen welke, kan hij daardoor toch de overtuiging krijgen dat Alice meer dan één antwoord weet. Bob kiest (stap 4) een *random challenge*, namelijk 0 als hij een wortel uit s wil zien en 1 als hij een wortel uit $s.b$ wil zien. Alice beantwoordt deze ene vraag en slaagt als het antwoord goed is.

Acceptatie. Omdat $r^2 = s$ en $a^2 = b$ is inderdaad $(r.a^c)^2$ gelijk aan $s.b^c$ voor zowel $c = 0$ als $c = 1$. Dus als Alice en Bob het protocol volgen wordt Alice geaccepteerd.

Weigering. Oscar, die *geen wortel van b kent*, kan voor geen enkel getal s zowel van s als van $s.b$ een wortel weten. Nu kun je best slagen voor een examen zonder de gehele stof te kennen, als je maar weet wat er wordt gevraagd.

Als Oscar weet of vermoedt dat Bob challenge 0 zal geven, maakt hij s net als Alice, namelijk als kwadraat van een random r. Hij kan dan op challenge 0 antwoorden, maar op challenge 1 niet. Als Oscar weet of vermoedt dat Bob challenge 1 zal geven, kiest hij random r en berekent s als $s = r^2/b$. (Bob ontvangt wel het getal s maar kan natuurlijk niet zien hoe zijn tegenpartij dat gekozen heeft!) Oscar kan dan op challenge 1 antwoorden, maar op challenge 0 niet.

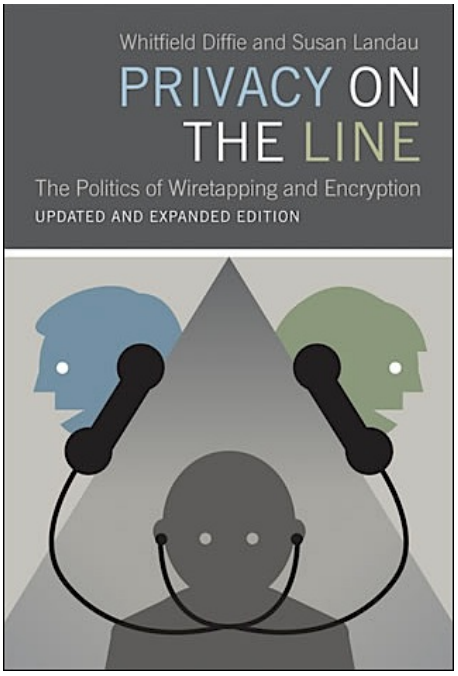
Oscar kan proberen Bobs challenge te anticiperen en slaagt voor de identificatie met kans $\frac{1}{2}$.

Lemma 8.1 *Een partij die een kans van meer dan $\frac{1}{2}$ heeft om door Bob te worden geaccepteerd kent een wortel van b.*

Bewijs. In het protocol kiest Bob de challenge pas na ontvangst van s, en een partij die een kans van meer dan $\frac{1}{2}$ heeft om te worden geaccepteerd, beschikt dus over een passende respons voor meer dan de helft van de mogelijke challenges. Deze partij beschikt dus (of kan deze berekenen) over twee wortels, zeg y_0 een wortel van s en y_1 een wortel van $s.b$. Dan geldt voor $y = y_1/y_0$ dat $y^2 = b$. □

Omdat Oscar met vrij grote kans kan slagen, is herhaling van het protocol nodig. Als er van protocol 8.5 k rondes worden gedaan, kan Alice in *elke* ronde worden geaccepteerd, terwijl voor Oscar de kans dat dit in alle ronden gebeurt slechts $\frac{1}{2^k}$ is.

Kader 8.6: Zero-knowledge bij Ministerie van Defensie?



Het Amerikaanse Ministerie van Defensie (Department of Defense, DoD) greep de patentaanvraag van Feige, Fiat en Shamir aan om zich wereldwijd belachelijk te maken. De patentaanvraag werd beantwoord met een *secrecy order*. Dit gebeurt incidenteel wanneer iemand iets heeft uitgevonden waarvan het DoD meent dat openbaarmaking de veiligheid van Amerika schaadt. De secrecy order verbiedt het over de uitvinding te spreken met buitenlanders, en verplicht de uitvinder tot het geven van een lijst van alle personen met wie hij er al over heeft gesproken.
Feige, Fiat en Shamir waren echter zelf buitenlander (Israëli) en hadden hun uitvinding in Israël gedaan en er op meerdere conferenties openlijk over gesproken. Shamir voorzag wel problemen bij de presentatie op een Amerikaans congres, maar gelukkig werd de secrecy order na wat telefoontjes achter de schermen weer even snel opgeheven als hij was gekomen.
Achtergronden van het verhaal en van de toepassing van secrecy orders zijn te vinden bij Landau [Lan88]. Whitfield Diffie en Susan Landau [DL98] bestudeerden honderden cases uit de verhouding tussen overheid en burgers op het gebied van de cryptografie.

Zero-knowledge. We zullen nu laten zien dat het protocol *zero-knowledge* is voor zowel Bob als Oscar. Hiermee wordt bedoeld dat Bob, dan wel Oscar, uit het protocol wel de zekerheid krijgt dat zijn tegenpartij over een wortel van b beschikt, maar geen enkele informatie over die wortel. De ontvangen informatie kan namelijk ook door Bob of Oscar zelf worden uitgerekend.

Definitie 8.2 ([FFS88]) *Een protocol is* zero-knowledge *voor een deelnemer X als X de mogelijkheid heeft om, zonder participatie van de andere deelnemer(s), informatie te genereren volgens dezelfde waarschijnlijkheidsverdeling als de informatie die hij tijdens het protocol ontvangt.*

Hierbij maken we een onderscheid waarbij de deelnemer eerlijk speelt (zero-knowledge voor Bob) danwel smokkelt (zero-knowledge voor Oscar). Oscar kan, als Alice zich bij hem meldt, van protocol 8.5 afwijken in de keuze van c, namelijk door deze niet random te kiezen maar af te laten hangen van s (en eventuele eerdere protocolronden). Na het bewijs komen we hierop terug.

Lemma 8.3 *Protocol 8.5 is zero-knowledge voor Bob en voor Oscar.*

Bewijs. Het afschrift van het protocol is een reeks van k tuples (s, c, y) waarin (1) s is een random kwadraat, (2) c is een random bit en (3) $y^2 = s.b^c$.

Bob kan een tuple $(s, 0, y)$ genereren door r random te kiezen en te nemen $s = r^2$ en $y = r$. Bob kan een tuple $(s, 1, y)$ genereren door r random te kiezen en te nemen $s = r^2/b$ en

$y = r$. Ondanks de deling door b is s hierbij een random kwadraat, want nog steeds heeft elk kwadraat dezelfde kans om als s voor te komen. Bob kan een complete conversatie genereren door k maal een c als random bit te kiezen, en een s en y erbij te genereren. Merk op dat Bob de waarde van s berekent *nadat* hij c heeft gekozen.

Oscar heeft ingeval van een echte interactie met Alice de mogelijkheid om zijn keuze van c te laten afhangen van de ontvangen s (en eventuele eerdere informatie die hij ontvangen heeft). Zij $\gamma(s)$ de bit die Oscar kiest als Alice in stap 3 de waarde s stuurt. Om deze ronde zonder Alice te genereren kiest Oscar een c random, genereert een bijpassende s (en y) als boven, maar gooit het hele tuple weer weg als $\gamma(s) \neq c$. Omdat er twee challenges mogelijk zijn (0 of 1) hoeft Oscar verwacht slechts ongeveer de helft van tuples weg te gooien. Het genereren van een reeks van k tuples kost hem dan zo'n $2k$ pogingen. □

Een protocol dat zero-knowledge is voor Bob beschermt tegen een passieve aanval. Immers hierbij kan Oscar een conversatie afluisteren, maar niet beïnvloeden, en dus is die conversatie totstandgekomen door interactie met een eerlijke Bob. Zero-knowledge voor Oscar is nodig om ook tegen actieve aanvallen (inclusief coalities) te beschermen. Hierbij neemt Oscar immers de plaats in van Bob, en kan hij door de interactie met Alice een conversatie verkrijgen waarin hij zijn keuzen niet volgens het protocol maakt, maar manipuleert.

Het herhalen van het protocol. Aangezien protocol 8.5 een onechte Alice slechts met kans $\frac{1}{2}$ ontmaskert (lemma 8.1), moet het meerdere keren worden uitgevoerd om de gewenste zekerheid te verkrijgen. Om te garanderen dat bedriegers slechts een kans van één op een miljoen hebben om te worden geaccepteerd is een twintigvoudige herhaling voldoende.

Dit kan zonder concessies aan de veiligheidseigenschappen door een simpele iteratie: Protocol 8.5 wordt dan twintigmaal *na elkaar* uitgevoerd. De random keuzen (r voor Alice en c voor Bob) worden in elke ronde opnieuw gemaakt.

Nadeel van deze sequentiele herhaling is dat het vaak wisselen van communicatierichting veel tijd kan kosten. Het is ook mogelijk de herhaling (deels) parallel te doen, waarbij Alice in één keer een aantal, zeg k, waarden s aanbiedt, en Bob antwoordt met k challenges. Het aantal parallelle challenges mag echter niet te groot zijn, anders is het niet meer mogelijk te bewijzen dat het protocol zero-knowledge is voor Oscar! Oscar kan namelijk de k challenges (totaal 2^k mogelijkheden) gezamenlijk laten afhangen van de k getallen die hij van Alice ontvangt (via een functie γ). Bij het zelf genereren van een conversatie moet hij dan alle combinaties weggooien waarbij $(c_1, \ldots, c_k) \neq \gamma(s_1, \ldots, s_k)$. Nu moet hij verwacht 2^k pogingen doen om een ronde van de conversatie te genereren.

8.3.2 Schnorr: discrete log

In het schema van Schnorr [Sch89] wordt gerekend modulo een groot priemgetal p waarbij er een element g is van hoge orde q, ook een priem. Het geheim van Alice is een getal $a \in \mathbb{Z}_q$ en Bob beschikt over $b = g^a$. Wegens aanname 6.6 is het niet mogelijk discrete logaritmen te berekenen, er is dus geen enkele berekening die Bob kan uitvoeren waarmee hij a te weten komt.

Het protocol. De manier waarop Alice haar kennis demonstreert loopt parallel aan protocol 8.5. Met het aanbieden van getal s claimt Alice nu bij elk getal c de index van $s.b^c$ te kennen. Bob neemt een steekproef, en als Alice daarvoor slaagt is de kans groot dat Alice ten minste twee antwoorden kent, hetgeen genoeg is als bewijs van identiteit.

Alice		Bob
Meld username	1	Zoek b behorende bij Alice
Kies random $r \in \mathbb{Z}_q$	2	
$s := g^r$; zend s	3	Ontvang s
	4	Kies random $c \in \mathbb{Z}_q$
Ontvang c	5	Zend c
Zend $y = r + a.c$	6	Ontvang y
	7	Accepteer als $g^y = s.b^c$

Protocol 8.7: Identificatie volgens Schnorr.

Acceptatie. Omdat $g^r = s$ en $g^a = b$ is inderdaad $g^{r+a.c}$ gelijk aan $s.b^c$. Dus als Alice en Bob het protocol volgen wordt Alice geaccepteerd.

Weigering. Oscar, die de index van b niet kent, kan voor geen enkel getal s bij elke c de index van $s.b^c$ noemen. Maar net als bij het vorige protocol kan hij voor het examen slagen als hij de challenge van Bob raadt. Als Oscar weet of vermoedt dat Bob de challenge c_0 zal geven, kiest hij r random en berekent $s = g^r/b^{c_0}$. Hij kan dan op challenge c_0 de respons r geven, maar op andere challenges kan hij niet antwoorden. Oscar kan proberen Bobs challenge te anticiperen en slaagt voor de identificatie met kans $\frac{1}{q}$.

Lemma 8.4 *Een partij die een kans van meer dan $\frac{1}{q}$ heeft om door Bob te worden geaccepteerd kent de index van b.*

Bewijs. In het protocol kiest Bob de challenge pas na ontvangst van s, en een partij die een kans van meer dan $\frac{1}{q}$ heeft om te worden geaccepteerd, beschikt dus over een passende respons voor meer dan een fractie $\frac{1}{q}$ van de mogelijke challenges. Deze partij beschikt dus (of kan deze berekenen) over ten minste twee indices, zeg y_1 de index van $s.b^{c_1}$ en y_2 de index van $s.b^{c_2}$. Nu

$$\begin{array}{lll} & g^{y_1} = s.b^{c_1} \ \wedge \ g^{y_2} = s.b^{c_2} & \\ \Rightarrow & g^{y_1-y_2} = b^{c_1-c_2} & \text{deel links door rechts} \\ \Rightarrow & y_1 - y_2 = a.(c_1 - c_2) & \text{Neem links en rechts index} \\ \Rightarrow & a = \frac{y_1-y_2}{c_1-c_2} & \text{in } \mathbb{Z}_q \text{ kun je delen} \end{array}$$

Bobs tegenpartij beschikt dus over a. □

Omdat q een groot getal is, is Oscar's kans om zonder kennis van a te worden geaccepteerd verwaarloosbaar, ook als er maar één ronde wordt uitgevoerd.

Zero-knowledge. De ruime keuze die Bob heeft bij het kiezen van zijn challenge (c uit $\mathbb{Z}_q$ in plaats van één bit zoals bij Feige e.a.) maakt niet alleen dat het protocol aan één ronde genoeg heeft, maar ook dat het niet meer zero-knowledge is voor Oscar.

Lemma 8.5 *Protocol 8.7 is zero-knowledge voor Bob.*

Bewijs. De conversatie levert een drietal (s, c, y) waarin s en c random zijn en y voldoet aan $g^y = s.b^c$. Bob kan zo'n drietal genereren door c en r random te kiezen en te berekenen $s = g^r/b^c$. □

Alice		Bob
Meld username	1	Zoek b behorende bij Alice
Kies random $r \in \mathbb{Z}_n$	2	
$s := r^e$; zend s	3	Ontvang s
	4	Kies random $c \in \{0, \ldots, e-1\}$
Ontvang c	5	Zend c
Zend $y = r.a^c$	6	Ontvang y
	7	Accepteer als $y^e = s.b^c$

Protocol 8.8: Identificatie volgens Guillou en Quisquater

Als Oscar in Bobs plaats met Alice converseert kan hij de challenge c kiezen afhankelijk van het getal s dat hij ontvangt. Er is op dit moment niemand die enige aanwijzing heeft kunnen aandragen hoe dit Oscar kan helpen enige nuttige informatie aan het antwoord y te ontlenen. Als Oscar zelf een tuple maakt (als in lemma 8.5) moet hij c als eerste kiezen. Het is daarom niet meer te bewijzen dat hij een conversatie die hij met Alice heeft zelf kan simuleren zoals vereist in de definitie van zero-knowledge (definitie 8.2).

In een implementatie van het protocol is het manipuleren van c overigens eenvoudig uit te sluiten. Na stap 1, dus *voordat* hij het getal s heeft gezien, moet Bob c kiezen en als blob naar Alice sturen. Alice ontvangt de blob, maar kan hieruit niet c vinden en kan haar s dus niet van c laten afhangen. Na ontvangst van s stuurt Bob c aan Alice, die nu kan controleren of dit inderdaad de waarde is waarop Bob zich na stap 1 heeft vastgelegd.

8.3.3 Guillou en Quisquater: hogeremachtswortels

Het schema van Guillou en Quisquater [GQ88] is een variant op dat van Feige e.a., maar gebruikt hogere machten. Net als het schema van Schnorr gebruikt het een challenge van meerdere bits. Alice en Bob kennen een publieke RSA-sleutel (n, e), waarbij e een priemgetal is. De geheime sleutel hoeven zij niet te kennen. Alice' geheim is $a \in \mathbb{Z}_n^*$ en Bob kent $b = a^e$.

Acceptatie. Omdat $r^e = s$ en $a^e = b$ is inderdaad $(r.a^c)^e$ gelijk aan $s.b^c$ voor elke c. Dus als Alice en Bob het protocol volgen wordt Alice geaccepteerd.

Weigering. Oscar kan de challenge c proberen te raden en hij kan worden geaccepteerd door $s = r^e/b^c$ in te sturen. Evenals in de vorige twee protocollen geldt dat het kennen van *twee* antwoorden kennis van a impliceert.

Lemma 8.6 *Een partij die een kans groter dan $\frac{1}{e}$ heeft om door Bob te worden geaccepteerd kent a.*

Bewijs. Omdat er precies e verschillende challenges mogelijk zijn moet zo'n partij op ten minste twee challenges, zeg c_1 en c_2, de juiste respons kunnen geven. De partij beschikt over y_1 en y_2 die voldoen aan $y_1^e = s.b^{c_1}$ en $y_2^e = s.b^{c_2}$. Bereken nu f als de inverse van $c_1 - c_2$ modulo e; dit kan omdat e een priemgetal is, en er geldt nu: $f.(c_1 - c_2) = h.e + 1$ voor zekere

h. Nu

$$
\begin{array}{lll}
 & y_1^e = s.b^{c_1} \ \wedge \ y_2^e = s.b^{c_2} & \\
\Rightarrow & (y_1/y_2)^e = b^{c_1-c_2} & \text{deel links door rechts} \\
\Rightarrow & y_1/y_2 = a^{c_1-c_2} & \text{neem links en rechts } e^{\text{e}} \text{ wortel} \\
\Rightarrow & (y_1/y_2)^f = a^{f.(c_1-c_2)} & \text{neem links en rechts } f^{\text{e}} \text{ macht} \\
\Rightarrow & (y_1/y_2)^f = a^{h.e+1} = b^h.a & \text{gebruik keuze van } f \\
\Rightarrow & a = \frac{(y_1/y_2)^f}{b^h} &
\end{array}
$$

Bobs tegenpartij beschikt dus over a. □

Zero-knowledge. Protocol 8.8 is zero-knowledge voor Bob.

Het is zero-knowledge voor Oscar als e zo klein is dat hij bij het genereren van een conversatie e pogingen kan besteden om één ronde te genereren.

8.3.4 Vergelijking van de protocollen

Hoe kan een conversatie waarvan *alle informatie* zonder kennis van a te genereren is, Bob overtuigen dat zijn tegenspeler a kent? De paradox wordt opgelost door te kijken naar de *volgorde* waarin de diverse getallen worden gekozen. Wanneer men eerst s kiest is a nodig om de respons te kunnen geven voor *elke* c. Wanneer men eerst c kiest kan *één* s worden gekozen waarvoor de respons kan worden gegeven. Bob dwingt af dat s wordt gekozen vóór zijn tegenpartij de challenge ontvangt. Aan het conversatieafschrift is niet te zien in welke volgorde s en c zijn gekozen.

Vooruitlopend op de verdere bespreking geeft deze tabel vast een overzicht van de drie gepresenteerde protocollen.

Naam	**Functie**	**in**	**Kans**	**Generatie**	**Opmerkingen**
Fiat, Feige, Shamir	$a \to a^2$	$\mathbb{Z}_m$	$1/2$	User / MK	m samengesteld
Schnorr	$a \to g^a$	$\mathbb{Z}_p$	$1/q$	User	p priem, $q = \text{ord}(g)$
Guillou, Quisquater	$a \to a^e$	$\mathbb{Z}_n$	$1/e$	User / MK	(n, e) is RSA key

Sleutelgeneratie door centrale partij. In de schema's van Fiat e.a. en Guillou-Quisquater wordt gerekend modulo een samengesteld getal. Deze modulus kan door Alice zelf worden gekozen (en onderdeel uitmaken van de public key), maar vaker wordt deze door een centrale partij gekozen en voor alle users gebruikt. De gebruikte functies (kwadrateren en machtsverheffen) zijn inverteerbaar voor wie de factoren van de modulus kent, in het tweede geval dus voor de centrale partij. Die beschikt hiermee over een soort masterkey, in de tabel als MK aangegeven. De discrete log (uit Schnorrs schema) kent deze achterdeur niet.

Dit geeft de mogelijkheid het systeem *zelfcertificerend* te maken. Hiermee wordt bedoeld, dat de publieke sleutel b van Alice kan worden gekozen, bijvoorbeeld als de hashwaarde van Alice' naam en adresgegevens. De systeembeheerder berekent dan voor Alice haar geheime sleutel a als de wortel uit b. Bob kan nu zonder certificaten in te zien aan Alice' publieke sleutel komen. Alice meldt zich bij Bob met bekendmaking van haar persoonsgegevens; Bob berekent b als de hashwaarde ervan en accepteert Alice als zij over de wortel van b blijkt te beschikken.

Aan de andere kant geeft inverteerbaarheid natuurlijk ook een risico: als de centrale partij onzorgvuldig is met de geheimhouding van de masterkey, of kwaadwillend is, kan Alice' sleutel gecompromitteerd raken.

Kwantitatieve eigenschappen. Omdat een keuze tussen de diverse protocollen vaak op kwantitatieve gronden wordt gemaakt, wordt nu gekeken hoe de diverse parameters van een protocol samenhangen.

Het *certainty level* of de *acceptatiekans* is de kans dat een aanvaller slaagt in een impersonificatiepoging. Omdat een aanvaller steeds de waarde van s kan kiezen om in ieder geval op één challenge te kunnen antwoorden, is met een challenge van k bits op z'n best een acceptatiekans van $\frac{1}{2^k}$ mogelijk.

De *lengte* van het protocol is het aantal malen dat van communicatierichting moet worden gewisseld en wordt uitgedrukt in het aantal achtereenvolgende ronden. Wanneer een basisprotocol t-maal achtereen wordt herhaald, kan Alice met haar respons van een ronde vast haar s van de volgende ronde meesturen. In totaal moet Alice dan $t + 1$ keer zenden, en daartussenin Bob t-maal. Zeker wanneer de communicatie een lange vertraging heeft (in het internet bijvoorbeeld) is een kort protocol wenselijk.

Belangrijk, vooral bij toepassing van smartcards, zijn ook de hoeveelheid geheugen voor opslag van Alice' geheim, de hoeveelheid rekenoperaties, en de hoeveelheid uit te wisselen bits.

Rondes, acceptatiekans, zero-knowledge. De acceptatiekans van een protocol kan worden verkleind door het te herhalen. Als het basisprotocol acceptatiekans ρ heeft en t-maal wordt herhaald, kan de acceptatiekans dalen tot ρ^t.

Deze herhaling kan in de lengte (sequentieel) of in de breedte (parallel) worden gedaan. Bij sequentiële herhaling stuurt Alice één s, en pas na ontvangst van Bobs challenge de s voor de volgende iteratie. Bij parallelle herhaling stuurt Alice een aantal (zeg k) waarden van s tegelijk, Bob antwoordt met k challenges en Alice geeft k responses.

Om het aantal ronden laag te houden is parallelle herhaling te prefereren, echter de breedte van het protocol moet klein blijven, wil men de zero-knowledge-eigenschap voor Oscar behouden. Deze eigenschap vereist immers dat Oscar een conversatie met Alice kan simuleren zonder haar aanwezigheid. Zoals na het bewijs van lemma 8.4 is opgemerkt geeft een k-bitchallenge een vertraging van 2^k bij deze simulatie.

Gelukkig staan realistische keuzen van de parameters een protocol van één ronde toe. Zo wordt een acceptatiekans van 1 op een miljoen doorgaans voldoende klein geacht. Een poging tot impersonificatie vindt immers plaats in een interactie met Bob, en bij ontmaskering kan Oscar van verdere pogingen worden uitgesloten. Oscar heeft dus doorgaans niet de gelegenheid zijn pogingen straffeloos te herhalen tot hij succes heeft. Simulatie van het protocol door Oscar is een offline proces, waarbij het niet problematisch is als Oscar een berekening een miljoen maal moet herhalen voordat hij een conversatie heeft gegenereerd. In praktische zin gesproken heeft dus een enkele ronde met een challenge van twintig bits zowel een voldoende kleine acceptatiekans als de zero-knowledge-eigenschap voor Oscar.

8.4 Meer over zero-knowledge proofs

In deze sectie worden mogelijkheden van zero-knowledge proofs besproken die verder gaan dan identificatie. In plaats van kennis van een getal kan ook kennis van een oplossing van een willekeurig (maar verifieerbaar) probleem mee worden bewezen (sectie 8.4.1). Door een schema van het interactieve aspect te ontdoen kan een handtekeningschema ontstaan (sectie 8.4.3); wel verdwijnt hierbij de zero-knowledge-eigenschap.

Alice		Bob
Kies $\rho \in S_3$	1	
Kies random r_x per $x \in V$	2	
$S = [H(r_x, \rho(A(x)) : x \in V]$	3	
Zend S	4	Ontvang S
	5	Kies random $c = xy \in E$
Ontvang c	6	Zend c
Zend r_x, $\rho(A(x))$, r_y, $\rho(A(y))$	7	Ontvang r_1, k_1, r_2, k_2
	8	Accepteer als $\begin{cases} k_1 \neq k_2, \\ k_1, k_2 \in \{1, 2, 3\}, \\ H(r_1, k_1) = S[x], \\ H(r_2, k_2) = S[y] \end{cases}$

Protocol 8.9: Bewijs van driekleuring

8.4.1 Zero-knowledgebewijs van driekleuring

We laten zien dat met een zero-knowledge proof niet alleen kennis van een geheim (maar herkenbaar) getal kan worden aangetoond, maar ook kennis van een oplossing van een NP-volledig probleem. Hiermee is het zero-knowledge proof bruikbaar voor bijvoorbeeld de verificatie van de inhoud van een blind ondertekend bericht (zie sectie 7.4.3). Immers, elk probleem waarvan de oplossing gecontroleerd kan worden zit in de NP-klasse.

Zoals aangegeven in figuur 6.5 behoort het graafkleuringsprobleem tot de moeilijkste problemen uit de klasse NP. Graafkleuring is NP-compleet en dit betekent dat alle NP-problemen tot graafkleuring kunnen worden herleid. Dat wil zeggen dat een NP-probleem als de kleuring van een bepaalde graaf kan worden geformuleerd, zó dat een oplossing voor het oorspronkelijke probleem wordt omgerekend in een kleuring.

Deze sectie laat zien dat Alice aan Bob kan bewijzen over een kleuring van een zekere graaf te beschikken, zonder dat Bob iets over de kleuring te weten komt. Alice stuurt Bob een 'kraslot' met daarop de ingekleurde graaf, waarvan Bob twee aangrenzende knopen mag openkrassen om te constateren dat hun kleur verschillend is. Alice maakt elk kraslot door de kleuren van haar oplossing op random wijze door elkaar te halen, zodat informatie van één kraskaart niet met die van een andere te vergelijken is.

Notaties. Om aan te sluiten bij de notaties uit de vorige sectie wordt de graaf aangegeven als $B = (V, E)$, waarbij V de verzameling van n knopen en E de verzameling van m kanten van de graaf is. In het vervolg gaan we uit van *drie*kleuring: zonder moeite is de methode voor meer kleuren te gebruiken. Alice beschikt over een driekleuring A; dit is een functie $A : V \to \{1, 2, 3\}$ met de eigenschap dat voor alle $xy \in E$ geldt $A(x) \neq A(y)$.

Het protocol gebruikt blobs als primitief en hiervoor moet er een botsingsvrije one-wayhashfunctie H bekend zijn. Om waarde v in te pakken wordt een random getal r gekozen, en de waarde $H(r, v)$ is de blob. De blob kan als v worden geopend door v en r te onthullen. Een blob die niet is geopend ziet er uit als een random getal, of anders gezegd, de enige manier om van een blob te bewijzen dat het geen random getal is, is hem te openen.

Het protocol. Alice prepareert haar kraskaart als volgt. Ze kiest een *random permutatie* ρ op de drie kleuren en berekent voor elke knoop $x \in V$ de tijdelijke kleur $\rho(A(x))$. Om de kleuren in blobs te pakken kiest Alice voor knoop x een random getal r_x en vormt $s_x = H(r_x, \rho(A(x)))$. De kraskaart S bestaat uit de collectie blobs.

Bobs challenge is een kant van de graaf, waarvan hij de kleuren van de verbonden knopen wil zien.

Als response stuurt Alice de twee kleuren, met de random getallen, waarna Bob controleert dat de kleuren verschillende getallen uit $\{1, 2, 3\}$ zijn, en dat de blobs juist geopend zijn.

De eigenschappen van het protocol. Als Alice en Bob het protocol volgen wordt Alice geaccepteerd, omdat Bob precies de constructie van Alice controleert, zij het slechts voor twee aangrenzende knopen.

De acceptatiekans (voor bedriegers) is bij dit protocol nog onacceptabel groot. Oscar, die geen kleuring van B kent, kan geen S construeren waarvan hij voor *elke* kant xy kan laten zien dat hij x en y verschillend heeft gekleurd. Maar misschien kent hij een toekenning van kleuren die slechts op één kant conflicteert. Dan is er een kans van $\frac{1}{m}$ dat hij niet slaagt, en een kans van $\frac{m-1}{m}$ dat hij wordt geaccepteerd. Protocol 8.9 zal dus een veelvoud van m keer moeten worden herhaald om een redelijke acceptatiekans te krijgen.

Een conversatie (S, c, Y) kan als volgt worden gegenereerd.

(1) Kies $c = xy \in E$.
(2) Kies kleuren k_1 en k_2 en randoms r_1 en r_2 voor x en y.
(3) Neem $S[x] = H(r_1, k_1)$, $S[y] = H(r_2, k_2)$, $S[z]$ random voor $z \neq x, y$.
(4) Neem $Y = (r_1, k_1, r_2, k_2)$.

(Let weer op de volgorde van het kiezen van S en c!)

Dit toont de zero-knowledge-eigenschap voor het protocol aan. De hoeveelheid communicatie is erg groot, door de grote omvang van S en het grote aantal ronden dat nodig is.

8.4.2 Niet-interactieve zero-knowledge

Het uitwisselen van de challenge en respons maakt een interactie tussen Alice en Bob nodig. Voor een identificatieprotocol is dit niet erg, maar in andere toepassingen van zero-knowledge proofs is het prettiger om van deze interactie af te komen. Alice genereert dan een *niet-interactief zero-knowledge proof*, dit is een bewijs van haar kennis dat door iedereen kan worden geaccepteerd, maar geen informatie over haar geheimen prijsgeeft.

De betekenis van de challenge c van Bob in een zero-knowledgeprotocol is tweeledig. Ten eerste ligt de zeggingskracht van het drietal (s, c, y) geheel in de volgorde van het kiezen van s en c, en de interactie met Bob dwingt Alice s te genereren voordat ze c kent. Ten tweede maakt de challenge elke conversatie verschillend zodat een afgeluisterde conversatie niet kan worden gebruikt voor impersonificatie. De eerste rol kan worden overgenomen door een hashfunctie en het tweede is voor bewijs van kennis niet nodig. (Voor het vormen van een handtekeningschema wordt de tweede rol overgenomen door een wisselend bericht, zie sectie 8.4.3.)

Een niet-interactief bewijs bestaat alleen uit de waarde s en de response y. De manier waarop de volgorde van berekenen aan het afschrift te zien is, is deze: de challenge c ligt impliciet vast door toepassing van een hashfunctie H op s. De berekening voor de challenge wordt bij het controleren van het bewijs herhaald.

In de verdere uitwerking van dit idee veronderstellen we Schnorrs schema, maar op de andere schema's is het ook toepasbaar. Het niet-interactieve bewijs is een paar (s, y) met de

Alice		Bob
Meld username	1	Zoek b behorende bij Alice
	2	$m = \langle \textbf{login}, \texttt{Alice}, \texttt{Bob}, tijd, u \rangle$
Ontvang m	3	Zend m
Verifiëer inhoud van m	4	
Kies random r	5	
$s := g^r$; $c = H(s, m)$	6	
$y := r + a.c$; zend (s, y)	7	Ontvang (s, y) ; $c = H(s, m)$
	8	Accepteer als $g^y = s.b^c$

Protocol 8.10: Handtekeningen volgens Schnorr

eigenschap

$$g^y = s \cdot b^{H(s)}.$$

Het paar kan, met kennis van a, zo worden gegenereerd:

	Genereren van niet-interactief bewijs
1	Kies $s = g^r$ voor random r
2	Kies $c = H(s)$
3	Kies $y = r + a \cdot c$.

Zoals gezegd bestaat het bewijs uit alleen (s, y); bij verificatie wordt $c = H(s)$ berekend en dan $g^y = s.b^c$ gecontroleerd. Omdat er geen interactie meer is, is het bewijs overdraagbaar: dat Bob het van een dame ontvangt, wil nog niet zeggen dat deze zelfde dame het heeft berekend. Daarom is een niet-interactief zero-knowledge proof niet als identificatiemiddel te gebruiken.

Als de challenge k bits breed is, kan zo'n drietal in ongeveer 2^k pogingen als volgt gevonden worden zonder kennis van a: kies y random, c random, $s = g^y/b^c$ en herhaal tot $H(s) = c$. Voor voldoende brede challenge (bijvoorbeeld $k = 80$) is deze methode ondoenlijk en dan kan zo'n drietal *alleen* met kennis van a worden berekend. Omgekeerd is het niet mogelijk a uit het drietal te berekenen.

Behalve dat er geen interactie meer nodig is met de controleur van het bewijs, heeft een niet-interactief bewijs nog een voordeel. De berekening van c met een hashfunctie dwingt af dat c eerlijk wordt gekozen en daarom is het niet nodig de zero-knowledge-eigenschap apart tegenover Bob of tegenover Oscar te bekijken. Als het interactieve protocol zero-knowledge is tegenover Bob, dan is de niet-interactieve versie veilig, onafhankelijk van de eerlijkheid van de tegenpartij.

8.4.3 Handtekeningschema

Een handtekeningschema verschilt van een identificatieschema in die zin, dat Bob achteraf een bewijs heeft dat Alice zich bij hem heeft aangemeld. Een handtekening schema is *nooit* zero-knowledge; immers, als Bob de conversatie zelf kan simuleren bewijst deze niet dat Alice bij hem is geweest. Het is mogelijk een zero-knowledgeprotocol om te vormen tot een handtekeningschema.

Weer bewijst Alice haar kennis van a niet-interactief, door het opsturen van een paar (s, y). Dit paar kan alleen met kennis van a worden berekend, maar dat betekent nog niet dat de partij die het aanbiedt zelf a kent. Daarom wordt er een koppeling met een (wisselend)

'bericht van de dag' gemaakt, en wel door c mede van dit bericht af te laten hangen. Wanneer Alice zich bij Bob meldt, vormt Bob het bericht m door vermelding van tijd en plaats van identificatie, samen met een random getal u. Bob vraagt Alice dit bericht te ondertekenen en accepteert Alice als haar handtekening klopt.

De controle op de inhoud van m (stap 4) voorkomt de volgende aanval van Oscar: Oscar wil op dinsdag Alice impersonifiëren bij Carol en speelt daarom op maandag voor Bob tegenover Alice. Alice wil zich identificeren tegenover Bob, maar Oscar laat haar alvast een bericht voor dinsdag tekenen, dat hij dan de volgende dag tegenover Carol gebruikt.

Overigens wordt deze aanval ook al onmogelijk gemaakt door het random getal u dat Carol in het te ondertekenen bericht opneemt. Dit getal u zorgt er ook voor dat als Alice gelijktijdig twee identificaties bij Bob doet, Bob toch over twee verschillende ondertekende berichten komt te beschikken.

Het bericht m dat wordt ondertekend kan ook elk willekeurig ander bericht zijn. De handtekeningschema's voor identificatie zijn niet wezenlijk anders dan die, welke voor het ondertekenen van berichten worden gebruikt.

8.5 Biometrie

De laatste jaren is er een toenemende belangstelling voor persoonsidentificatie op basis van lichaamsgebonden kenmerken. Hoewel deze techniek zelf niet cryptografisch van aard is, past een korte bespreking van de mogelijkheden goed bij het onderwerp identificatie. Sectie 8.5.1 belicht de theoretische achtergrond, werking, en begrippen die bij alle vormen van biometrie een rol spelen. Sectie 8.5.2 vergelijkt de diverse biometrische kenmerken die voor identificatie worden of kunnen worden gebruikt. Sectie 8.5.3 behandelt enkele toepassingen.

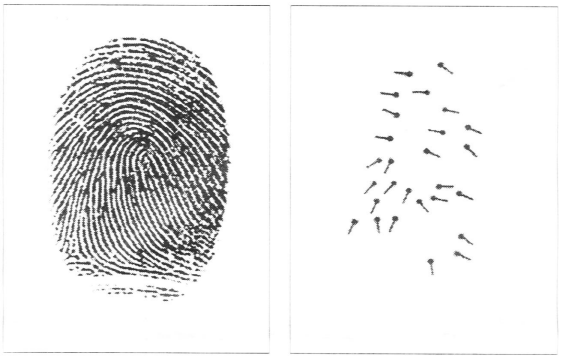
8.5.1 Principes van biometrische identificatie

Het persoonskenmerk waarop de identificatie is gebaseerd zal in het systeem dat de identificatie uitvoert aanwezig zijn in een digitale en samengevatte vorm die we *template* noemen. Het opgeslagen template is vergelijkbaar met een publieke sleutel (b in de vorige secties) en het aanbieden ervan is dan de persoonlijke sleutel a. Het verschil is dat a niet geheimgehouden hoeft te worden omdat niemand het kan aanbieden dan de echte houder ervan.

Controle op juiste template. Betrouwbare identificatie vereist dat het juiste template b van de gebruiker aanwezig is. In open systemen kan b onderdeel zijn van een certificaat, dat een ondertekende verklaring over het template vormt en door de gebruiker zelf aan het systeem wordt aangeboden. Een persoonlijke smartcard (bijvoorbeeld bankpas of paspoort) kan bij het uitgeven worden voorzien van het template van de houder en is daarna niet meer te wijzigen.

Vorming en vergelijking van templates. Zowel bij het installeren van een gebruiker als bij de identificatie wordt het kenmerk opgemeten met een sensor (bijvoorbeeld camera, microfoon) en samengevat in het template, b of a. Het template bestaat uit enkele karakteristieken van het lichaamskenmerk waarvan wordt verondersteld dat ze robuust zijn onder meetfouten en ruis; zie bijvoorbeeld kader 8.11. De identificatie slaagt als de twee templates in voldoende mate overeenstemmen.

Kader 8.11: Template van een vingerafdruk



Een vingerafdruk ziet er op het eerste gezicht uit als een onontwarbare kluwen lijnen. Er bestaan echter, volgens Köhler [Köh99], een zevental karakteristieke patronen (*minutiae*), zoals een splitsing, een 'oog' of lijneinde. Het template bevat de types, richtingen en onderlinge posities van de minutiae. Het is niet mogelijk vanuit het template de vingerafdruk te reconstrueren.
Door variatie in de plaatsing van de hand, stof of ruis kan een volgend opgenomen template van dezelfde persoon in een of meerdere minutiae afwijken. Daarom moet bij de identificatie een marge tussen het opgeslagen en aangeboden template worden toegestaan.

Gelijkheid van de templates kan niet worden verlangd vanwege de fluctuatie in het kenmerk en ruis bij de meting. Om deze reden is het ook niet zinvol de biometrische meetgegevens integraal op te slaan.

Onjuiste acceptatie of weigering. De mate waarin opgeslagen en aangeboden template overeen moeten komen is een in te stellen parameter; deze heeft invloed op de foutfrequentie (FAR en FRR) van de methode.

Bij een ruime marge kan een verkeerde persoon ten onrechte worden geaccepteerd als zijn kenmerk ongeveer met dat van de doelpersoon overeenkomt. Dit heet een *false acception* en de *False Acception Rate* of FAR is het percentage van de gevallen waarin een onjuist persoon wordt geaccepteerd. Bij een krappe marge kan een aangeboden template van de juiste persoon al te veel van het opgeslagen template afwijken waardoor de persoon ten onrechte wordt geweigerd. Dit heet een *false rejection* en de *False Rejection Rate* of FRR is het percentage afwijzingen van de juiste persoon.

Verkleinen van de marge verlaagt de FAR maar verhoogt de FRR. Afhankelijk van de toepassing kan een kleine FAR of juist een kleine FRR belangrijk zijn, waarmee in de instelling van de marge rekening kan worden gehouden. Verlaging van FAR en FRR kan soms worden bereikt door preciezere informatie in het template op te slaan, maar dit kan leiden tot hogere geheugen- en processorbelasting.

8.5.2 Vormen van biometrie

Er is inmiddels een grote verscheidenheid aan lichaamskenmerken die voor identificatie kunnen worden gebruikt. Voor sommige is toepassing nog in het laboratoriumstadium, zoals DNA, geur en het menselijk elektrisch veld, waarop we hier kortheidshalve niet ingaan.

Vingerafdruk. De vingerafdruk van elk mens is uniek, en zelfs per vinger verschillend, maar de afdruk blijft wel gedurende het hele leven hetzelfde. Vingerafdrukken zijn bovendien ongevoelig voor stemmingen of ziekte.

Kader 8.12: Toetsenbord met biometrische ontsluiting

Biometrie op basis van vingerafdrukken is goedkoop. Er is een breed scala van lezers op de markt beschikbaar, van aparte scanners tot geïntegreerde producten zoals deze Wombat Midas. Van deze lezers kunnen meerdere personen gebruik maken omdat meerdere templates zijn opgeslagen (in de pc of server). Kleine scanners kunnen ook worden toegepast om gebruiksvoorwerpen te personaliseren. Zo kan in de kolf van een pistool een sensor worden ingebouwd, waardoor alleen de eigenaar ermee kan schieten (de zogenaamde 'smart gun', `www.colt.com`). Dit voorkomt dat eerzame huisvaders door inbrekers of hun kinderen met hun eigen afgepakte blaffer overhoop worden geknald. Maar er zijn ook smartcards met geïntegreerde lezers; deze kunnen als bankpas worden ingezet.

Met een zeer eenvoudige optische sensor is de vingerafdruk op te nemen, en de technieken om ze in templates samen te vatten zijn vergevorderd. Vanuit technisch oogpunt biedt de vingerafdruk vele voordelen.

Schrijfgedrag. Bij het opmeten van schrijfgedrag wordt de persoon gevraagd een schriftelijke handtekening te plaatsen. Behalve het resultaat (dat met wat oefening heel gemakkelijk is te reproduceren) wordt continu de druk en snelheid van de pen geregistreerd. De penbewegingen blijken, meer dan de uiteindelijke handtekening, representatief voor de persoon die de handtekening heeft gezet.

Irisherkenning. Het patroon van lijnen en vlekken in het oog is een uniek kenmerk, dat bovendien zonder lijfelijk contact kan worden bemeten: de persoon hoeft slechts in een camera te kijken.

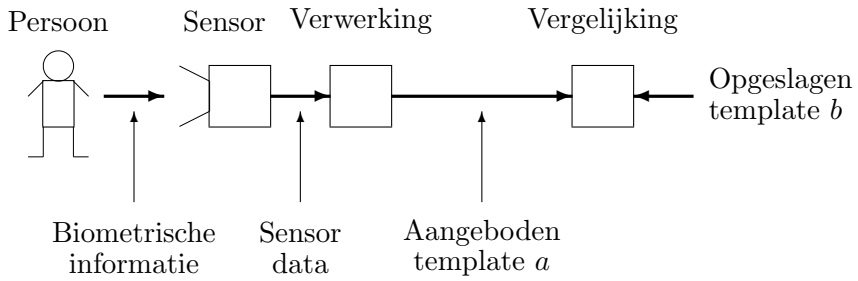
Handgeometrie. Met een sensor, vergelijkbaar met een vingerafdrukscanner, kan de vorm van de gehele hand worden bepaald.

Stemherkenning. Omdat vrijwel elke pc al van een microfoon is voorzien is het erg gemakkelijk om de stem op te nemen bij het uitspreken van een bepaald woord. De verwerking tot een template dat een goede balans biedt tussen FAR, FRR en opslagruimte is nog een probleem.

8.5.3 Huidige toepassingen

De gang van informatie van een persoon naar de vergelijking van templates is weergegeven in figuur 8.13. Uiteraard kan de identificatie niet betrouwbaarder zijn dan de route die deze informatie volgt.

Een eerste poging tot misleiding is het simuleren van de biometrische informatie voor de sensor. Bijvoorbeeld met een bandopname van de stem, een foto van de iris die voor de camera wordt gehouden, of een afgehakte vinger. De duurdere vingerafdruk-sensoren gebruiken



Figuur 8.13: Sensoring en verwerking van data

warmtemeting en aanwezigheid van bloedcirculatie om te verifiëren dat de aangeboden vinger levend is. Een irisscanner kan dit controleren door de iris op verschillende licht-golflengten te scannen.

Vervolgens kan men proberen een echt template van de gebruiker te bemachtigen en af te spelen voor de vergelijking. Deze aanval is mogelijk als de aanvaller het datapad tussen de sensor (en verwerking) en de vergelijking kan manipuleren. Hieruit volgt al dat biometrie niet geschikt is om identificatie over het internet te ondersteunen, maar ook toetsenbord met vingerafdruklezer geeft op dit gebied al reden tot zorg.

Köhler [Köh99] merkt op dat identificatie met vingerafdrukken het populairst is in verband met de technische mogelijkheden op dit gebied. Daarnaast gelden in verschillende landen wetten omtrent de opslag van persoonsgegevens die ook in het voordeel van vingerafdrukken werken. Europese richtlijnen beperken het opslaan en gebruiken van gegevens betreffende gezondheid, seksuele geaardheid, godsdienst en ras. De iris verandert onder invloed van bepaalde ziekten, een gezichtsscan onthult het ras, de stem verandert bij verkoudheid, de geur verraadt alcoholgebruik. Alleen de vingerafdruk bevat geen informatie over deze privacygevoelige kwesties.

Samenvatting en conclusies

Met identificatie wordt bedoeld dat een persoon zijn identiteit of zijn recht op gebruik ergens van bewijst. Identificatie is een van de belangrijkste toepassingen van cryptografie.

Cryptografische identificatie gebeurt steeds op basis van een geheim dat de betreffende persoon moet kennen. Probleem hierbij is dat het overleggen van het geheim de waarde ervan vermindert, omdat daarna ook een afluisteraar of de controleur het kent. Soms is dit acceptabel (de controleur wordt zelf vertrouwd), soms zijn cryptografische algoritmen nodig.

Beveiliging tegen afluisteraars wordt bijvoorbeeld verkregen met eenmalige passwords of door het geheim niet over te zenden, maar alleen in een berekening te gebruiken en het resultaat over te zenden.

Identificatie zonder geheime informatie over te dragen aan de controleur is ook mogelijk, namelijk met een zero-knowledge proof. Hierbij stuurt de persoon eerst een getal, waarna de controleur mag antwoorden met een willekeurige challenge en de persoon moet hierop een antwoord geven. De persoon geeft maar één antwoord, maar omdat hij niet vantevoren weet

welke challenge wordt gevraagd geeft een goed antwoord het vertrouwen dat hij ook andere challenges zou kunnen beantwoorden.

Zero-knowledge proofs worden veel gebruikt in cryptografische protocollen omdat een partij ermee kan bewijzen het protocol eerlijk te spelen, zonder informatie over de eigen geheimen te geven.

De laatste jaren is er ook een groeiende belangstelling voor biometrische identificatie. Hierbij wordt van een persoon een lichaamskenmerk opgemeten en vergeleken met een opgeslagen kenmerk. Vooral de vingerafdruk is populair omdat deze gemakkelijk is op te nemen.

Opgaven bij hoofdstuk 8

Opgave 8.1 *Een dief smeert vaseline op de knoppen van een geldautomaat en rolt de pinpas na gebruik. Hoe groot is zijn kans op misbruik als hij*

(a) ziet dat de toetsen 2, 3, 5 en 9 zijn gebruikt?
(b) ziet dat de toetsen 2, 5 en 9 zijn gebruikt?
(c) ziet dat de toetsen 2 en 5 eenmaal, en 9 tweemaal is gebruikt.

Is een code met drie verschillende cijfers veiliger of minder veilig dan een met vier verschillende cijfers?

Opgave 8.2 *Waarom is het belangrijk dat Schnorrs identificatieschema gebruikmaakt van een subgroep van $\mathbb{Z}_p^*$, en niet van een echte generator g van $\mathbb{Z}_p^*$?*

Opgave 8.3 Zero-knowledge identificatie. *(Uit het tentamen van januari 1997.)*
In het Gilliou-Quisquater identificatie systeem is er een systeemparameter (n, e), en voor gebruiker X is een getal b_X publiek opgeslagen. Gebruiker X identificeert zich door te tonen dat hij beschikt over een getal a_X waarvoor geldt $a_X^e = b_X \pmod{n}$. Geef een protocol waarmee X zijn kennis kan bewijzen, zonder enige informatie over a_X prijs te geven.

Opgave 8.4 *Is het in het bewijs in lemma 8.3 belangrijk dat bij Oscars γ-functie de waarden 0 en 1 even vaak voorkomen?*

Opgave 8.5 Guillou en Quisquater Identificatie. *(a) Wat is het belangrijkste nadeel van een identificatieprotocol op basis van symmetrische cryptografie (zoals bv. toegepast in betaalpassen)?*
(b) In het systeem van Guillou en Quisquater is een RSA publieke sleutel (n, e) bekend. Het geheim van Alice is a en haar publieke sleutel is b; welke relatie bestaat tussen a en b?
(c) Beschrijf hoe de Zero-Knowledge identificatie verloopt.
(d) Waar wordt de geheime sleutel d behorend bij (n, e) gebruikt?

Opgave 8.6 *Een drietal (s, c, y) als in sectie 8.4.3 kan* alleen *met kennis van a worden berekend. Wat is er mis met dit identificatieprotocol: Alice meldt zich bij Bob en geeft hem zo'n drietal, Bob accepteert als $c = H(s)$ en $g^y = s.b^c$.*

Opgave 8.7 *Alice is verdacht van een computerinbraak op dinsdagavond vanuit Bobs systeem. Bob beweert dat Alice op dinsdagavond bij hem heeft ingelogd met protocol 8.10; Alice beweert niet bij Bob te hebben ingelogd. Beschrijf welke informatie de partijen kunnen aanvoeren en hoe een rechter over dit bewijsmateriaal kan oordelen.*

Opgave 8.8 *Alice is verdacht van een autokraak op dinsdagavond. Alice beweert op dinsdagavond bij Bob te hebben ingelogd met protocol 8.10; Bob beweert dat Alice op dinsdagavond niet bij hem heeft ingelogd. Beschrijf welke informatie de partijen kunnen aanvoeren en hoe een rechter over dit bewijsmateriaal kan oordelen.*

Project 8.9 *Maak een scriptie over de smart gun.*

Hoofdstuk 9

Verdeelschema's en groepscryptografie

De uitspraak "kennis is macht" karakteriseert treffend waar het in de cryptografie om gaat, immers, informatie of bevoegdheden worden beschermd door middel van kennis van cryptografische sleutels. Veel situaties vragen om gedeelde macht: in een vereniging moet een besluit bijvoorbeeld statutair door ten minste drie bestuursleden worden bekrachtigd. Ook kan men denken aan bankkluizen, die met meerdere sleutels moeten worden geopend. Een dergelijke verdeling van bevoegdheid wordt ondersteund door een cryptografisch *verdeelschema*, waarbij een sleutel pas kan worden gebruikt wanneer voldoende bezitters van deelsleutels meewerken.

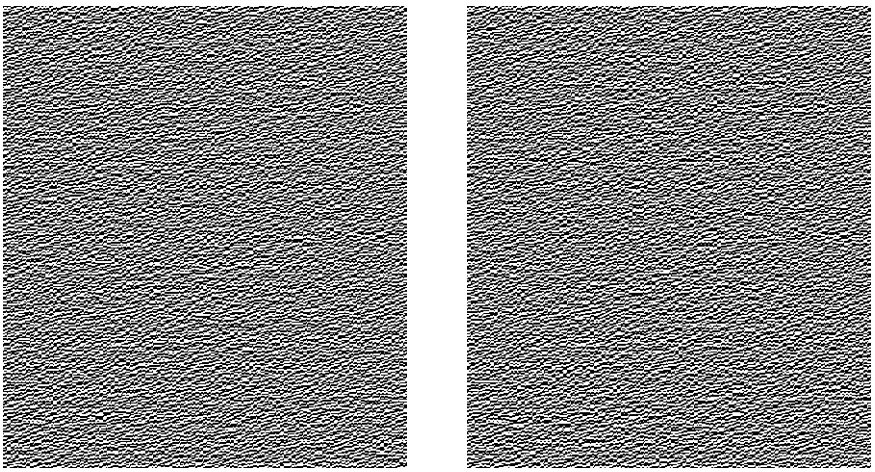
Verdeelschema's hebben binnen de cryptografie echter ook toepassingen die tot nu toe geen equivalent in de dagelijkse wereld hebben. Het is mogelijk informatie van meerdere partijen gecombineerd te verwerken zonder dat de partijen deze informatie beschikbaar hoeven te stellen. Deze *secure multiparty computations* worden besproken in hoofdstuk 14, maar in dit hoofdstuk wordt een illustrerend voorbeeld behandeld, namelijk 'de villabewoners' (sectie 9.1.2).

Sectie 9.1.1 introduceert enige terminologie en beschrijft eigenschappen die de diverse schema's al dan niet hebben. Een belangrijk onderscheid, dat ook samenhangt met de gebruikte techniek, betreft de vraag of *alle* deelsleutels nodig zijn voor het gebruik van de hoofdsleutel. Is dit het geval dan spreken we van een *vetoschema*. De gebruikte techniek is hier zeer elementair: de deelsleutels worden gekozen als willekeurige getallen met de hoofdsleutel als som. Soms worden er meer deelsleutels uitgedeeld dan nodig voor het gebruik van de hoofdsleutels en dan spreken we van een *drempelschema*; dit vereist een ingewikkelder verdeling.

De verdeelde sleutels moeten voor cryptografische berekeningen te gebruiken zijn zonder ze te reconstrueren. Ontsleuteling van een bericht wordt uitgevoerd door een protocol waarin de deelsleutelhouders gezamenlijk deelnemen. Er wordt dan gesproken van *groepscryptografie* (*threshold cryptography*).

De opzet van het hoofdstuk is zo, dat diverse mogelijkheden en eigenschappen van de schema's eerst worden uitgelegd aan de hand van een vetoschema (sectie 9.2). Vervolgens wordt het (polynoomgebaseerde) drempelschema behandeld, en hoe diverse eigenschappen hieraan kunnen worden gekoppeld (sectie 9.3). Hoewel dit voor het drempelschema gecompliceerder is, zijn belangrijke beveiligingsdoelen ook haalbaar voor drempelschema's. Sectie 9.1 introduceert de benodigde terminologie, veiligheidsvereisten en aanvallen.

Kader 9.1: Visuele cryptografie.



Het is mogelijk om aandelen op transparanten uit te geven en de reconstructie optisch uit te voeren. Er wordt dan gesproken van *visuele cryptografie* (*visual cryptography*). In het additieve veto-schema is elk aandeel een random getal en informatie over het geheim is alleen in hun correlatie aanwezig. Net zo zijn de twee visuele aandelen elk op zich een random verdeling van zwarte en witte pixels zodat een grijs vlak ontstaat dat geen informatie bevat. De informatie is alleen aanwezig in de correlatie tussen de twee aandelen en wordt zichtbaar wanneer de vlakken precies over elkaar gelegd worden. Om dit te zien moet je deze bladzijde kopiëren op een transparant en een van de grijze vlakken exact bovenop de andere leggen. Je kunt de kopie bovenop deze bladzijde leggen of de plaatjes van de kopie uitknippen en op elkaar leggen. (Zie ook kader 9.5.)

9.1 Inleiding: schema's en gebruik

9.1.1 Terminologie en classificatie

Een verdeelschema wordt gebruikt om informatie door verdeling te beschermen. De betreffende informatie is het *geheim* (*secret*). De delen waarin het is opgedeeld, worden de *aandelen* of *shares* genoemd en de partijen die in het bezit ervan zijn de *aandeelhouders* of *shareholders*. Het proces waarbij de aandelen worden gevormd is het *delen* (*dealing*). Vaak is er een partij die in het begin over het geheim beschikt en de aandelen uitgeeft, deze partij heet dan de *deler* (*dealer*). Het geheim kan worden terugberekend wanneer voldoende aandelen bijeen worden gebracht; we spreken dan van *reconstructie* (*reconstruction*) van het geheim. Reconstructie is een functie *Rec* met k argumenten. In dit hoofdstuk staat k steeds voor het aantal aandelen dat nodig is om de reconstructie uit te voeren. Verder is X de verzameling waaruit het geheim genomen is en S de verzameling waaruit de aandelen komen.

Aan een verdeelschema worden twee elementaire eisen gesteld; de eerste laat zich doorgaans eenvoudig verifiëren en betreft de correctheid van het gereconstrueerde geheim:

- **Reconstrueerbaarheid.** Wanneer de reconstructie wordt uitgevoerd met k aandelen is de uitkomst gelijk aan het geheim.

De tweede eis is dat het geheim niet bekend kan worden door het samenvoegen van minder dan k aandelen en deze eis kan hetzij computationeel, danwel in perfecte zin worden gesteld.

- **Perfecte verhulling:** Bij elke combinatie van $k-1$ aandelen $s_1, \ldots, s_{k-1}$ en elke waarde $x \in X$ is er een $s_k \in S$ zodanig dat reconstructie uit $(s_1, \ldots, s_k)$ de waarde x oplevert.
- **Computationele verhulling:** Het is computationeel ondoenlijk om, gegeven $k-1$ aandelen van een geheim x, de waarde x te bepalen.

Perfecte verhulling is *sterker* dan te eisen dat $k-1$ aandelen de waarde niet vastleggen, waarvan hier een voorbeeld dat vooruitloopt op de villabewoners.

Hellingschema voor villabewoners. De deler is eigenaar van een riant pand waarvan de waarde met zeven cijfers wordt geschreven. Hij geeft elk van de zeven aandeelhouders een van deze cijfers (met vermelding van de positie). De geheime waarde van het huis is reconstrueerbaar uit de zeven aandelen, die immers tezamen het betreffende getal van zeven cijfers vormen. Wanneer echter minder dan zeven aandelen bijeen worden gelegd kan het bedrag niet exact worden gereconstrueerd, maar wel kunnen veel mogelijkheden worden uitgesloten. Is het aantal in aanmerking komende bedragen eerst 10^7, elk bijkomend aandeel reduceert het aantal mogelijkheden met een factor 10. Anders gezegd, samenwerking van minder dan k aandeelhouders maakt in dit schema het geheim gedeeltelijk zichtbaar. Perfecte verhulling sluit dit uit: zelfs als $k-1$ aandelen bekend zijn is het geheim nog geheel onzichtbaar (elke waarde is nog mogelijk).

Bij deze verdeling is het doorgaans aantrekkelijker het eerste cijfer te weten dan om het laatste te weten, omdat het eerste cijfer immers de waarde ongeveer aangeeft. Deze overweging speelt in de verder te behandelen schema's geen rol.

Het voorbeeld heeft een plezierige eigenschap, namelijk dat elk aandeel klein is (slechts één cijfer). De lengte van de aandelen *samen* is gelijk aan de lengte van het geheim en er is dus voor de verdeelde opslag niet meer ruimte nodig (in totaal) dan voor centrale opslag. Een schema waarin geleidelijk meer over het geheim bekend wordt naarmate er meer aandelen gegeven zijn wordt een *hellingschema* (Engels: *ramp scheme*) genoemd.

Zo'n schema kan bijvoorbeeld worden gebruikt om een 3DES-sleutel te verdelen: elke aandeelhouder krijgt dan een van de drie DES-deelsleutels zodat ze samen een bericht kunnen ontsleutelen. De bescherming naar buiten toe is gelijk aan die van 3DES: een buitenstaander moet alle 168 bits achterhalen, onafhankelijk van de verdeelde opslagwijze. De bescherming tegen aanvallen van binnenuit is kleiner. Samenwerking van twee sleutelhouders reduceert een aanval tot het achterhalen van de derde deelsleutel, wat precies zo moeilijk is als het breken van DES. Je kunt dus zeggen dat 3DES met een hellingschema tegen aanvallen van buitenaf zo sterk is als 3DES, maar tegen aanvallen van binnenuit zo sterk als DES.

De prettige eigenschap van het schema dat de aandelen klein zijn is voor perfecte verdeling niet realiseerbaar.

Stelling 9.1 *Perfecte verhulling impliceert dat elk aandeel zo groot is als het geheim.*

Bewijs. Gegeven $k-1$ aandelen kan men de reconstructie opvatten als een functie van S naar X, die het k^{e} aandeel afbeeldt op het geheim. Perfecte verhulling vereist dat elk punt in X als beeld voorkomt (de afbeelding is surjectief) en dit impliceert dat $|S| \geq |X|$. De lengte van het aandeel (in bits) is dus ten minste de lengte van het geheim. □

Doorgaans zijn S en X dezelfde verzameling.

Vetoschema's en drempelschema's. Het eerste onderscheid dat we maken tussen schema's betreft de vraag of het mogelijk is om meer shares uit te delen dan er nodig zijn voor de reconstructie. Bij de eenvoudigste schema's, *vetoschema's* of ook wel *secret splitting* genaamd, is dit

niet het geval: er worden exact k aandelen verspreid en deze zijn alle nodig voor reconstructie. We noemen dit een vetoschema omdat elke aandeelhouder de reconstructie van het geheim kan verhinderen. Het geheim is niet beschikbaar wanneer een aandeelhouder afwezig is wegens ziekte of ontslag, of als een aandeel is kwijtgeraakt. Dit kan soms acceptabel zijn in eenvoudige toepassingen als spelletjes of in secure multiparty computing (bijvoorbeeld de villabewoners). Meestal echter is een goede beschikbaarheid van het geheim vereist ook bij afwezigheid van een aandeelhouder en dan is een schema nodig dat het uitdelen van meer aandelen toestaat.

In een *drempelschema* kan de deler meer aandelen uitdelen dan nodig voor de reconstructie. Er wordt wel gesproken van een k-uit-n-schema wanneer er n aandelen worden uitgedeeld, waarvan k willekeurige voor reconstructie kunnen worden gebruikt (uiteraard moet $n \geq k$ gelden).

Het realiseren van een drempelschema is op zich niet heel gecompliceerd. Wel is het toevoegen van extra beveiliging, bijvoorbeeld tegen bedrog en verdeeld gebruik (zie verderop), voor deze schema's ingewikkelder dan voor een vetoschema. Daarom is er bij de opzet van dit hoofdstuk voor gekozen om een vetoschema in sectie 9.2 te behandelen inclusief de uitbreidingen, hoewel veel ervan praktisch niet relevant zijn. Sectie 9.3 behandelt dan het drempelschema van Shamir, weer inclusief enkele uitbreidingen, maar de lezer kan ervoor kiezen die over te slaan danwel er vluchtig kennis van te nemen.

Verifieerbaarheid. De eisen die verder aan de schema's kunnen worden gesteld hebben te maken met de mogelijkheid dat een of meerdere van de deelnemende partijen (deler of aandeelhouders) oneerlijk spel spelen.

Een aandeelhouder kan de uitkomst van de reconstructie verstoren door het inleveren van een *vals aandeel*, een onjuiste waarde, dat wil zeggen, een getal s' in plaats van het echte aandeel s. De deler kan al bij de verdeling valsspelen, en hier onderscheiden we twee vormen:

1. Het delen van *inconsistente aandelen*, dat wil zeggen, er worden aandelen uitgedeeld waaruit geen reconstructie mogelijk is, of (in een drempelschema) waarbij de uitkomst afhangt van *welke* aandelen worden gecombineerd.

2. Het delen van een vals geheim, dat wil zeggen, er worden aandelen uitgedeeld die wel een correcte en unieke reconstructie toestaan, maar de uitkomst is niet het echte geheim.

Bedrog kan worden tegengegaan door het verspreiden van extra informatie (publieke aandelen) waarmee correcte aandelen kunnen worden herkend. Schema's waarin dit gebeurt heten *verifieerbare* (*verifiable*) schema's. Naast de reconstructieprocedure is er dan een verificatieprocedure $Corr(s)$ waaraan een aandeel wordt onderworpen vóór reconstructie; de uitkomst is *true* als s het door de deler verstrekte aandeel is. Dit predikaat $Corr$ heeft behalve s andere parameters zoals de index van de aandeelhouder en informatie die door de deler bekendgemaakt kan zijn.

De bescherming tegen bedrog in een of andere vorm kan ook weer van *perfecte* of *computationele* aard zijn en we zullen aan een voorbeeld zien hoe deze twee kunnen worden uitgewisseld.

- **Perfecte verifieerbaarheid:** Er bestaat behalve het echte aandeel s *geen* $s' \in S$ waarvoor $Corr(s')$ geldt.

- **Computationele verifieerbaarheid:** Het is ondoenlijk, zelfs gegeven een s waarvoor $Corr(s)$ geldt, een $s' \neq s$ te berekenen waarvoor $Corr(s')$ geldt.

Perfecte bescherming betekent natuurlijk dat bedrog ronduit onmogelijk is omdat er maar één waarde bestaat waarmee de betreffende partij voor de dag kan komen. Computationele

bescherming laat de mogelijkheid open dat er verschillende waarden bestaan, maar het is computationeel onhaalbaar een andere dan de juiste te vinden.

Verifieerbaarheid staat op gespannen voet met verhulling.

Lemma 9.2 *Een schema kan niet zowel perfect verhullend als perfect verifieerbaar zijn.*

Bewijs. Perfecte verhulling vereist dat er, gegeven $k-1$ aandelen, nog verschillende mogelijkheden voor de uitkomst, en dus voor het laatste aandeel zijn. Perfecte verifieerbaarheid vereist juist dat er voor een aandeelhouder maar één mogelijk aandeel bestaat dat door de anderen wordt geaccepteerd. □

Reconstructie of verdeeld gebruik. De aandeelhouders kunnen hun aandelen gebruiken om samen het geheim te berekenen, maar dit is lang niet altijd gewenst. Vaak is het de bedoeling dat het geheim door de aandeelhouders samen wordt gebruikt, maar dat het daarna weer door verdeling moet zijn beschermd. Helaas maakt reconstructie het geheim op één plaats zichtbaar, en de aandeelhouders hebben geen zicht op wat er vervolgens met het geheim gebeurt (kader 9.12).

Verdeelschema's kunnen de mogelijkheid bieden om met het geheim te rekenen zonder het te reconstrueren. De berekening met het geheim als invoer wordt vervangen door een protocol, dat door de aandeelhouders samen wordt uitgevoerd. Het resultaat is de uitkomst van de berekening, en het geheim blijft verdeeld. Uiteraard speelt ook bij deze gedistribueerde rekenschema's de verifieerbaarheid een belangrijke rol, want aandeelhouders kunnen de berekening verstoren door in het protocol valse waarden in te sturen.

Het is zelfs mogelijk dat de aandelen door de aandeelhouders in een gedistribueerde berekening worden vastgesteld, dus zonder tussenkomst van een deler. Het geheim bestaat dan van meet af aan alleen in verdeelde vorm.

9.1.2 Toepassingen van verdeelschema's

Er zullen nu enkele toepassingen van verdelen worden toegelicht, waaruit ook zal blijken dat bij elk ervan de nadruk op verschillende veiligheidseisen kan liggen.

Sleutelverdeling bij banken en overheid. Belangrijke sleutels in bancair of overheidsverkeer zijn vaak verdeeld om af te dwingen dat meerdere functionarissen bijeen zijn voordat de sleutel kan worden gebruikt. Een bankkluis of belangrijke safe had bijvoorbeeld twee sloten, waarbij de directeur de sleutel van de ene had en elke klerk een sleutel van de andere.

De extra bescherming die dit (boven onverdeelde sleutels) biedt is drieledig.

1. Het is moeilijker voor indringers om de sleutel zonder meer te stelen omdat ze de deelsleutels van verschillende plaatsen moeten halen.

2. Er is bescherming tegen misbruik van binnenuit, dat wil zeggen tegen de eigen medewerkers die uit hoofde van hun taken over sleutels moeten kunnen beschikken. Dit is een belangrijk punt want het voorkomen van misbruik van binnenuit is namelijk alleen door sleutelverdeling mogelijk.

3. De medewerkers worden beschermd tegen aanvallen op hun persoon. Sleutelverdeling ontmoedigt criminele organisaties om medewerkers met geweld onder druk te zetten om aan illegaal gebruik mee te werken.

Kader 9.2: James Bond gepensioneerd door verdeelschema's.

In *Operatie Donder* (*Thunderball*, 1965) moet James Bond '007' het opnemen, hoe kan het ook anders, tegen het oude bekende S.P.E.C.T.R.E. De organisatie heeft twee atoombommen gestolen en dreigt die te laten ontploffen in Engelse en Amerikaanse steden als er niet een enorm losgeld wordt betaald.
De beveiliging op moderne atoomwapens zou de film in deze tijd minder realistisch maken. Atoomwapens zijn namelijk met pincodes beveiligd en deze codes zijn bovendien verdeeld waarbij reconstructie plaatsvindt in een chip die in de splijtbare massa is opgenomen; zie Anderson en Kuhn [AK96].
Omdat de wapens voor eenmalig gebruik zijn gemaakt en bovendien de reconstructiehardware afdoende wordt vernietigd, is reconstructie geen bezwaar en is een procedure voor gedistribueerd gebruik overbodig. Wel is verificatie nodig; de beoogde ontvanger van de bom kan namelijk proberen te infiltreren en valse aandelen te introduceren waardoor de bom onbedoeld niet zou barsten.

In het elektronische bancaire verkeer bestaan zeer belangrijke sleutels, bijvoorbeeld die waarmee bankpassen worden gevalideerd of certificaten ondertekend. Die worden dan ook altijd verdeeld zodat voor deze handelingen de aanwezigheid en deelname van meerdere medewerkers is vereist. De aandelen worden op een smartcard uitgegeven, die eventueel extra kan worden beveiligd met een PIN of vingerafdruk.

Voor het feitelijke ondertekenen van een bankpas of certificaat komen de aandeelhouders bijeen en voeren de smartcards in in de speciale hardware die de ondertekening uitvoert. Deze hardware is een zogenaamde secure device dat voor deze taak is geprogrammeerd en waarvan wordt vertrouwd dat er niet mee is geknoeid. Het is daarom zeker dat de sleutel na reconstructie en gebruik in deze speciale machine weer wordt gewist, zodat er geen noodzaak is voor een speciaal protocol dat gedistribueerd gebruik mogelijk maakt.

Multiparty secure computing: De villabewoners. In veel toepassingen is het wenselijk om berekeningen uit te voeren op een manier die alleen het eindresultaat, maar niet de invoeren of tussenresultaten onthult. Bij deze berekeningen, *multipartyberekeningen* genaamd, kunnen meerdere partijen betrokken zijn met diverse belangen op het gebied van geheimhouding van hun invoer of beïnvloeding van de uitkomst. Het ontwerp ervan is daarom een ingewikkelde zaak die wordt behandeld in hoofdstuk 14. Aan de hand van een eenvoudige probleemstelling (de villabewoners) wordt hier geïllustreerd hoe verdeling een rol kan spelen in multipartyberekeningen, en aan welke eisen de gebruikte schema's moeten voldoen.

	Bewoner i
1	Kies n getallen $s_j[i]$ $(j = 1 \ldots n)$ met som $x[i]$.
2	Zend $s_j[i]$ naar bewoner j.
3	Ontvang $s_i[j]$ van bewoner j (voor elke j).
4	Bereken $U_i := \sum_j s_i[j]$.
5	Publiceer U_i
6	Lees alle U_j
7	$SOM := \sum_j U_j$; $AVE := SOM/n$

Protocol 9.3: De villabewoners: oplossing.

Het probleem van de villabewoners. De n bewoners van een villawijkje hebben belang bij het bepalen van de gemiddelde waarde van hun woningen (bijvoorbeeld om bezwaar te kunnen maken tegen gemeentelijk OZB-beleid). Bewoner i kent de waarde $x[i]$ van zijn eigen pand, maar geen van de bewoners wil hierover openheid geven uit angst onder het gemiddelde te liggen en dan door de anderen als armoedzaaier te worden beschouwd. Hun probleem is: hoe kan het gemiddelde (of, equivalent: de som) van de $x[i]$ worden bepaald zonder dat iemand de waarde van zijn huis aan anderen hoeft te vertellen?

Het berekenen van het gemiddelde is een gezamenlijk belang, en we nemen dan ook aan dat geen van de deelnemers de uitkomst van de berekening wil verstoren, en dat de partijen elkaar ook in dit opzicht vertrouwen. Wel wordt elke bewoner gedreven door een natuurlijke nieuwsgierigheid die hem zal doen pogen om stiekem toch achter de waarde van andere panden te komen.

De oplossing: rekenen met aandelen. De berekening beschermt elke individuele invoer door verdeling: de berekening wordt uitgevoerd met aandelen van de invoerwaarden en alleen het eindresultaat wordt gereconstrueerd; zie protocol 9.3. Bewoner i deelt aandelen van zijn eigen invoer $x[i]$; het deel voor bewoner j heet $s_j[i]$. Dit schema loopt vooruit op het additionele schema van sectie 9.2: de aandelen zijn random getallen met som $x[i]$, waarbij wordt gerekend modulo een groot getal q. Elke bewoner ontvangt dus een aandeel van elk bedrag.

Het simpele additionele schema heeft de eigenschap te commuteren met optelling: de sommen van aandelen zijn aandelen van een som. Elke bewoner telt zijn n aandelen van de verschillende huizen op, en de bewoners hebben nu elk een aandeel van de beoogde som, die door reconstructie zichtbaar wordt gemaakt.

Eigenschappen van de oplossing. De oplossing gebruikt, zoals veel protocollen voor verdeling en multipartyberekeningen, zowel een-op-eencommunicatie als verzending naar alle deelnemers. De communicatie binnen een protocol moet natuurlijk beveiligd worden afhankelijk van de eisen die aan het protocol als geheel worden gesteld. Van een-op-eencommunicatie is de beveiliging tegen afluisteren of veranderen inmiddels uitgebreid behandeld. Onder het publiceren van informatie wordt verstaan dat deze informatie aan alle deelnemers bekend wordt gemaakt, waarbij wordt gegarandeerd dat elke deelnemer hetzelfde bericht ziet.

De bewoners zijn goed beschermd tegen het uitlekken van hun geheim. De waarde x_i kan worden berekend wanneer *alle* bewoners behalve i hun informatie combineren, maar dit kan niet als een zwakte van het protocol worden gezien. Hoe de gemiddelde waarde ook is berekend, als alle bewoners behalve één hun informatie en het gemiddelde bij elkaar leggen kunnen ze de

ontbrekende waarde uitrekenen. Daarom biedt dit protocol een optimale bescherming van de privacy van elke deelnemer.

Beperkingen van de oplossing. Zoals al is opgemerkt moet worden aangenomen dat de bewoners geen belang hebben bij het verstoren van de uitkomst van de berekening. De bewoner die als laatste zijn U_i publiceert heeft hiervoor de beste mogelijkheden: hij kan namelijk eerst de $n-1$ andere aandelen van de uitkomst afwachten. Om dit te voorkomen moet het protocol worden uitgebreid met verificatie, maar de verificatie-informatie moet dan ook voor de aandelen van de uitslag te berekenen zijn.

De bewoner die zijn aandeel als laatste publiceert, kent de uitslag als eerste en kan, zelfs bij een verifieerbaar schema, de procedure verstoren als de uitslag hem niet bevalt. Hij kan weigeren zijn U_i te publiceren.

Verdeling bij sleutelafgifte. Onder de bezwaren die worden opgevoerd tegen een verplichte afgifte van sleutels (zie kader 9.4) is de mogelijkheid tot misbruik ervan. Uiteraard is in ons land de politie in haar handelen aan wetten en regels gebonden, maar hiervan wordt wel eens afgeweken. Ook vanuit beheerders van de telecommunicatie-infrastructuur vinden illegale afluisterpraktijken plaats.

Het risico van oneigenlijk gebruik van de afgegeven sleutels kan, volgens Micali [Mic92], worden verkleind door de sleutels verdeeld op te slaan. De gebruiker van een sleutel maakt (bijvoorbeeld bij de aanvraag van een certificaat voor de publieke sleutel) aandelen van de geheime sleutel en die worden over meerdere opslagcentra verdeeld. De opslagcentra zijn verplicht hun deel ter beschikking te stellen voor opsporing wanneer er daartoe een justitieel bevel wordt overlegd. Deze opslagcentra zouden natuurlijk onder strenge controle op het gebied van veiligheid moeten staan, maar kunnen verschillende politieke, religieuze of maatschappelijke signatuur hebben. Dit vermindert het risico van misbruik tegen politieke tegenstanders, want de tegenstanders van het ene centrum zijn de vrienden van het andere.

De kosten van een dergelijke structuur zijn hoog en de baten ervan gering, en het is daarom onwaarschijnlijk dat verdeelde afgifte in de geschetste vorm ooit werkelijkheid wordt. Als voorbeeld is het echter erg interessant omdat er aan veel eisen op verdeelschema's tegelijk voldaan moet zijn:

1. **Drempelschema.** Snel politiewerk vereist dat geheime sleutels *snel* beschikbaar kunnen zijn, ook 's nachts, en mogelijk hebben een of meerdere centra bedenkingen tegen het afluisteren van de betreffende verdachte. Schema's waarin de medewerking van alle centra vereist is voor gebruik van de sleutel zijn onbruikbaar.

2. **Aandeelhouderverifieerbaar.** Een verdachte kan vrienden hebben binnen de opslagcentra, en daarom kan een centrum proberen het gebruik van de sleutel te frustreren door een vals aandeel in te leveren.

3. **Delerverifieerbaar.** Bij toepassing in gedwongen sleutelafgifte heeft de deler er groot belang bij dat de latere reconstructie zal mislukken. Het moet daarom aan de aandelen te zien zijn dat ze bij reconstructie een echt geheim opleveren, namelijk de geheime sleutel behorend bij de publieke sleutel waarvoor een certificaat wordt afgegeven.

4. **Verdeeld gebruik.** Het afgeven van de aandelen aan de politie elimineert de privacy van de gebruiker totaal. De medewerking van de centra is gemakkelijker te verkrijgen wanneer zij controle hebben over wat er met de sleutel gebeurt. Het is daarom wenselijk een schema te hebben waarmee de centra gezamenlijk een aantal verdachte berichten

Kader 9.4: Sleutelafgifte ten behoeve van opsporingsinstanties.

Voor politie en justitie, die gewend zijn bij verdachten de telefoon af te tappen of post te onderscheppen, is de beschikbaarheid van sterke cryptografie een doorn in het oog. Vanuit met name de Amerikaanse opsporingsinstanties klinkt vaak de roep om het gebruik van versleuteling, ook voor persoonlijke communicatie, te binden aan de afgifte van een kopie van de geheime sleutel ten behoeve van opsporing: *sleutelafgifte* of *key escrow*.

Net als het reguliere bedrijfsleven maken internationale criminele organisaties steeds meer gebruik van electronische communicatie. Door sleutelafgifte hoopt men terrorisme, drugshandel en kinderporno effectiever te kunnen bestrijden. Ook vanuit het bedrijfsleven is belangstelling voor sleutelafgifte, om toegang tot alle bedrijfsinformatie te kunnen hebben ook als werknemers afwezig, ziek of ontslagen zijn.
Tegenstanders wijzen op de hoge kosten van de benodigde infrastructuur en de beveiliging ervan (zie bijvoorbeeld `http://www.cdt.org/crypto/risks98/` of Koops en Jong [KJ98] over aansprakelijkheidsrisico's). De veiligheid van public-keysystemen neemt af wanneer de geheime sleutels gaan worden verstuurd. Bovendien is de keuze van het veiligheidsniveau niet langer een persoonlijke, omdat een hoger niveau dan dat gekozen door het sleutelbeheer niet meer mogelijk is. Het risico van misbruik van de afgegeven sleutels kan door verdeling ervan worden verkleind. Het gebruik van een tweede ontcijfersleutel bij een bericht (zoals de *Additional Decryption Key* bij PGP) kan als alternatief bruikbaar zijn.
Een voorstel tot wet om sleutelafgifte zonder meer verplicht te stellen voor gebruikers van versleuteling heeft het in Nederland in 1993 niet gehaald. Ook in de Verenigde Staten en andere landen halen regeringen hiermee bakzeil. Vervolgens werd geprobeerd de software-industrie over te halen, bijvoorbeeld door het opheffen van exportbeperkingen voor cryptoproducten met sleutelafgifte. Het is ook mogelijk om afgifte van een geheime sleutel te verplichten voor het verkrijgen van een certificaat voor het publieke deel ervan.

kunnen ontsleutelen, zonder meteen alle communicatie van de verdachte voor de politie zichtbaar te maken.

9.1.3 Vaste notaties

In alle verdeelschema's in dit hoofdstuk staat a voor het geheim en a_i voor het i^{e} aandeel. Waar gebruik wordt gemaakt van private en publieke informatie (voor verifieerbaarheid) is b het publieke geheim en b_i het i^{e} publieke aandeel. Het aantal aandelen dat voor reconstructie nodig is is k. Er wordt steeds gerekend modulo een (groot) priemgetal q. Zowel het geheim als de aandelen zijn getallen modulo q, dus $S = X = \mathbb{Z}_q$.

Om de overige variabelen (waarvoor het alfabet maar net groot genoeg is) uit elkaar te houden kan de lezer deze tabel gebruiken:

a	(deel van) geheim	n	aantal aandeelhouders
b	publiek geheim of aandeel	o	
c	challenge	p	modulus van publieke aandelen
d	differentieel aandeel	q	priem, modulus van aandelen
e	publiek deel van d of f	r	random
f	coëfficiënt van polynoom	s	opening in ZKP
g	element van orde q	t	tweede opening in ZKP
h		u	gecodeerde ElGamalblinder
i	index van aandeelhouder	v	geblindeerd ElGamalbericht
j	index van aandeelhouder	w	quorumfactor
k	aantal benodigde aandelen	x	bericht bij ElGamal
l	index van aandeelhouder	y	response bij ZKP
m	nieuwe aandeelhouder	z	(delen van) blinder bij ElGamal

Er worden in dit hoofdstuk twee verdeelschema's behandeld, beide met diverse variaties in gebruik. Het zijn het additieve schema (een vetoschema) in sectie 9.2 en het polynoomschema van Shamir [Sha79] (een drempelschema) in sectie 9.3.

9.2 Een additief vetoschema

In het additieve schema bestaat tussen het geheim a en de aandelen a_i deze eenvoudige relatie:

$$a = \sum_i a_i.$$

Directe varianten erop zijn een multiplicatief schema (waar het geheim het product van de aandelen is) of een bitgeoriënteerd schema waarin het geheim als de XOR van de aandelen wordt verkregen.

9.2.1 Delen en reconstructie

Alle berekeningen met het geheim en de aandelen worden uitgevoerd modulo een groot priemgetal q. De deler kiest $k-1$ aandelen random en berekent vervolgens de waarde van het laatste aandeel zo, dat aan de relatie is voldaan:

	Deler
1	Kies a_1 tm a_{k-1} random, uniform en onafhankelijk.
2	Bereken $a_k := a - (a_1 + \ldots + a_{k-1})$.
3	Zend a_i aan aandeelhouder i.

Het geheim a is alleen bij het maken van a_k gebruikt en het lijkt daarom alsof aandeel a_k aantrekkelijker is dan de andere. Dit is echter niet zo, want a_k is net als de andere aandelen een uniform getrokken random getal.

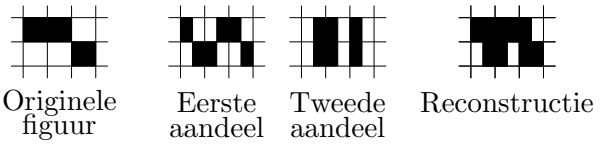
Propositie 9.3 *Voor elk getal $s \in S$ is $Pr(a_k = s) = \frac{1}{q}$.*

Bewijs. Er geldt $a_k = s$ dan en slechts dan als $a_1 = a-(a_2+\ldots+a_{k-1}+s)$. De kans hierop is $\frac{1}{q}$ omdat a_1 met uniforme verdeling is gekozen onafhankelijk van de andere getrokken getallen en het geheim. □

Kader 9.5: HOE WERKT VISUELE CRYPTOGRAFIE?

Van het verborgen plaatje is elke pixel een klein vierkantje dat of wit, of zwart is. De pixels in de aandelen zijn verder opgedeeld: van elke pixel in een aandeel is of de linkerhelft, of de rechterhelft zwart.

Deze keuze is random, maar net als in het additieve schema zit de informatie in de correlatie tussen de twee aandelen. Voor een *witte* pixel in het geheim zijn in de aandelen de *zelfde* helften zwart. Voor een *zwarte* pixel in het geheim zijn in de aandelen de *tegenoverliggende* helften zwart.



Wat gebeurt er als de aandelen over elkaar worden gelegd? Bij een witte pixel overlappen de twee zwarte delen elkaar zodat een helft van het vierkantje onbedekt blijft. Bij een zwarte pixel vullen de twee zwarte delen elkaar aan zodat het gehele vierkant wordt bedekt.

Door dit idee verder uit te werken is het ook mogelijk om figuren met grijstinten te verdelen. Een groep in Leuven werkt al aan kleurenafbeeldingen: zie `http://www.esat.kuleuven.ac.be/~rijmen/vc/`.

Dat alle aandelen even belangrijk zijn is ook af te zien aan de reconstructie, die geheel symmetrisch in de aandelen is:

$$Rec(a_1, \ldots, a_k) = \sum_i a_i.$$

Stelling 9.4 *Additieve verdeling is perfect verhullend.*

Bewijs. Zij gegeven $k-1$ aandelen, dwz. de getallen a_i voor $i \neq i_0$, en een getal x. Met de keuze $a_{i_0} = x - \sum_{i \neq i_0} a_i$ wordt verkregen dat $Rec(a_1, \ldots, a_k) = x$, dus elke waarde voor het geheim is nog mogelijk gegeven $k-1$ aandelen. □

9.2.2 Verifieerbaarheid met publieke aandelen

Het controleren van de juistheid van een ingeleverd aandeel (of andere waarde in een protocol) kan in principe op twee manieren gebeuren.

1. **Publiek aandeel.** De deler maakt, behalve de private aandelen waaruit de aandeelhouders het geheim kunnen reconstrueren, ook publieke aandelen. Het is niet mogelijk uit de publieke aandelen het geheim te reconstrueren of het bijbehorende private aandeel, maar gegeven het publieke aandeel kan het juiste private aandeel worden herkend.

2. **Zero-knowledge proof.** Een partij die een waarde overlegt (die met een private waarde is berekend), verzorgt tevens een bewijs van juistheid ervan. Het bewijs overtuigt de andere spelers van de juistheid van de waarde, maar omdat het zero-knowledge is, verschaft het geen informatie over de private waarde. Deze controle zien we in sectie 9.2.3.

Perfecte verhulling of perfecte verifieerbaarheid. Door de wijze waarop de deler de publieke aandelen beschikbaar stelt kan men kiezen om ofwel de verhulling, ofwel de verifieerbaarheid perfect te maken (en de andere eigenschap computationeel). De publieke aandelen

zijn het beeld van de private aandelen onder een one-wayfunctie. De private aandelen zijn getallen in $\mathbb{Z}_q$, maar de publieke aandelen zijn getallen in $\mathbb{Z}_p$, waar p een groter priemgetal is. De functie gebruikt een getal g van orde q en luidt $H(x) = g^x$. Tussen een private waarde v en de bijbehorende publieke waarde w bestaat dus steeds deze relatie:

$$w = g^v.$$

Merk op dat kennis van w de waarde v niet beschikbaar maakt (omdat de discrete log moeilijk is, aanname 6.6) maar wel vastlegt (gegeven w is er maar één mogelijkheid voor v). De bescherming van de private waarde, gegeven de publieke, is dus niet perfect maar computationeel.

De publieke aandelen b_i worden tijdens het delen aangemaakt en de deler kan ze publiceren, of alleen (ondertekend) aan de betreffende aandeelhouder geven.

1. **Perfecte verhulling en computationele verifieerbaarheid:**
   De deler geeft elke aandeelhouder zijn private aandeel en geeft hem ook het ondertekende publieke aandeel: het drietal (a_i, b_i, s_i) waar s_i de handtekening over b_i is. De ondertekening door de deler is nodig zodat de andere aandeelhouders zich van de echtheid van het publieke aandeel kunnen overtuigen. Bij reconstructie overlegt de aandeelhouder dit drietal en het wordt correct bevonden als $g^{a_i} = b_i$ en s_i is geldig over b_i (formeel $Ver(b_i, s_i, y)$ waar y de sleutel van de deler is).

   Deze variant is perfect verhullend omdat gegeven $k-1$ aandelen nog steeds elke waarde van het k^{e} aandeel (met publiek deel en ondertekening) mogelijk is.

   Een aandeelhouder die wil valsspelen moet een acceptabele combinatie (a'_i, b'_i, s'_i) maken. Hij kan eerst a'_i kiezen, en dan zijn publieke aandeel $b'_i = g^{a'_i}$, maar moet dan de handtekening van de deler onder b'_i vervalsen. Een alternatief is eerst een handtekening onder b'_i te vervalsen (bijvoorbeeld als een existentiële vervalsing), maar dan moet a'_i als de discrete log hiervan worden berekend. Van deze en andere pogingen een acceptabel drietal te berekenen kan worden aangenomen dat ze computationeel onmogelijk zijn.

2. **Computationele verhulling en perfecte verifieerbaarheid:**
   De deler geeft elke aandeelhouder zijn private aandeel en publiceert de publieke aandelen. We nemen aan dat deze publicatie zo wordt gedaan dat alle deelnemers van de echtheid van de publieke aandelen overtuigd zijn, zodat we de handtekening achterwege laten. Bij reconstructie levert aandeelhouder i het getal a_i in, en er wordt gecontroleerd of $g^{a_i} = b_i$.

   Bedrog door aandeelhouder i is nu perfect uitgesloten omdat er maar één getal bestaat dat door de anderen zal worden geaccepteerd.

   De verhulling is niet perfect omdat de aandelen, en daarmee het geheim, vastliggen door de gepubliceerde publieke aandelen. Het berekenen van het geheim of de aandelen is het het vinden van de discrete logaritme van de publieke aandelen, en dus computationeel onmogelijk (stelling 9.6).

Hoewel perfecte verhulling erg aantrekkelijk klinkt, is deze eigenschap in de praktijk moeilijk waar te maken, zelfs als het verdeelschema perfect verhult. De reden is dat de aandelen van de deler naar de aandeelhouders verzonden worden, en vaak wordt hierbij publickeycryptografie gebruikt zodat de geheimhouding van de aandelen niet perfect maar computationeel is. Daarom zullen we in dit hoofdstuk verder naar schema's kijken waarbij de publieke aandelen worden gepubliceerd.

Deler		Aandeelhouder i
Kies geheime sleutel a	1	
Publiceer $b = g^a$	2	Lees b
Vorm a_j zdd. $\sum_j a_j = a$	3	
Publiceer $b_j = g^{a_j}$	4	Lees b_j, controleer $\prod_j b_j = b$
Stuur a_j aan j	5	Ontvang a_i, controleer $g^{a_i} = b_i$

Protocol 9.6: MICALI'S DELERVERIFIEERBARE VETOSCHEMA.

Delerverificatie met publiek geheim. Verificatie van de aandeelhouders geeft het vertrouwen dat reconstructie de waarde oplevert die door de deler is verdeeld. Echter, dit is nog geen garantie dat de aandeelhouders iets hebben aan dit getal, want de deler kan voor de a in de deelprocedure elk willekeurig getal nemen. Voor delerverificatie is ten eerste nodig dat de aandeelhouders weten welk getal voor hen waardevol is, met andere woorden, dat ze het juiste geheim kunnen herkennen. Ten tweede moet deze herkenning kunnen worden uitgevoerd zonder over het geheim te beschikken, maar aan de hand van aandelen ervan.

Om herkenning van het echte geheim a mogelijk te maken nemen we aan dat er, behalve het private geheim a van de deler een publiek geheim b bekend is, uiteraard gelijk aan $b = a^g$. Een geheim waarvan een publieke versie beschikbaar is kan niet worden geconstrueerd (ondoenlijkheid van discrete log) maar is slechts computationeel beschermd.

Nu kan men aanvoeren dat als de deler een vals geheim a' wil uitdelen, hij ook een vals publiek geheim $b' = g^{a'}$ kan publiceren. Voor de vorming van de geheimen a en b onderscheiden we *expliciete* en *impliciete* geheimen. Een geheim is expliciet als de waarde ervan vastligt door relaties met andere zaken (zoals de waarde van een huis). Voor een expliciet geheim nemen we aan dat de publieke versie ervan ook vastligt, dus niet door een publicatie van de deler maar vanuit een andere partij. Een geheim is impliciet als de waarde ervan in principe alles mag zijn, zoals bijvoorbeeld een random gekozen sleutel. De deler is dan niet a priori aan een waarde gebonden, maar moet wel aandelen verstrekken die het juiste getal, dat past bij de publieke sleutel, opleveren. In dit hoofdstuk gebruiken we als voorbeeld het ElGamal-algoritme, omdat de publieke sleutel horend bij a ook gelijk is aan $b = g^a$. De publieke sleutel is dus gelijk aan de publieke versie van de geheime sleutel.

Protocol 9.6 (van Micali [Mic92]) gebruikt het feit dat de additieve relatie tussen de private aandelen zich vertaalt naar een multiplicatieve relatie tussen de publieke aandelen. De relatie $b = \prod_j b_j$ noemen we de *globale consistentie* van de (publieke) aandelen en deze wordt na publicatie van alle publieke waarden door de aandeelhouders gecontroleerd. Vervolgens controleert elke aandeelhouder of zijn eigen private aandeel correspondeert met de gepubliceerde publieke versie. De relatie $b_i = g^{a_i}$ noemen we de *interne consistentie* van het i^{e} aandeel en die wordt in stap 5 gecontroleerd. Globale en interne consistentie van de aandelen impliceert dat de aandeelhouders gezamenlijk over de logaritme van b beschikken. Dus als er berichten worden verzonden met public key b, dan zijn de aandeelhouders samen in staat het te ontsleutelen.

Stelling 9.5 *Als* $b = \prod_j b_j$ *en* $\forall j : b_j = g^{a_j}$ *dan* $g^{\sum_j a_j} = b$.

Het schema is niet perfect verhullend, immers, van alle private waarden zijn publieke versies bekend. Wel is het computationeel verhullend omdat het berekenen van a uit $k-1$ private aandelen en alle publieke waarden equivalent is met de discrete logaritme; het schema is dus cryptografisch sterk.

Stelling 9.6 *Micali's schema is computationeel verhullend.*

Bewijs. Veronderstel dat Oscar beschikt over een procedure Rec^* die het geheim berekent uit $k-1$ private aandelen, dat wil zeggen, de waarden a_i voor $i \leq k-1$. Verdere invoer van Rec^* zijn het publieke geheim b en de k publieke aandelen b_i.

De procedure Rec^* kan als volgt worden gebruikt om de logaritme x van een willekeurig getal y te vinden:

	Het vinden van de logaritme van y
1	$b_k := y$
2	a_i is random; $b_i := g^{a_i}$ voor $i \leq k-1$
3	$b := \prod_i b_i$
4	$a := Rec^*(a_1, \ldots, a_{k-1})$
5	$a_k := a - (a_1 + \ldots + a_{k-1})$
6	$x := a_k$

Stappen 1 tot en met 3 maken y onderdeel van een verdeeld geheim a met $k-1$ bekende private aandelen. Stap 4 gebruikt de veronderstelde procedure die a berekent. Stappen 5 en 6 vinden de logaritme van y uit dit geheim.

De aanname dat de logaritme van y niet kan worden berekend impliceert dus dat Rec^* niet bestaat, waarmee het schema inderdaad cryptografisch sterk is. □

Deze methode kent nog enkele problemen, die met de gecompliceerdere protocollen later in dit hoofdstuk kunnen worden opgelost.

1. *Wat te doen bij bedrog?* Protocol 9.6 ontmaskert bedriegende partijen wel, maar voorkomt niet altijd de schade. Als de deler vals blijkt te spelen, wordt dit bij het delen gemerkt en kan men besluiten overnieuw te beginnen, eventueel met een andere deler. Maar als bij reconstructie (of gebruik) van de geheime sleutel een aandeelhouder vals blijkt te spelen kan hij hoogstens worden gestraft, maar het geheim is verloren.

   Dit probleem is op te lossen met een drempelschema, waarbij er meer aandelen verspreid zijn dan nodig voor reconstructie. Blijken er bij reconstructie aandeelhouders te liegen over hun aandeel, dan kan de reconstructie worden uitgevoerd met alleen de correcte aandelen (mits dit er minstens k zijn).

2. *Deler laat het geheim lekken.* De deler kan niet voorkomen dat het geheim later wordt gereconstrueerd, maar hij kan wel de verhulling van het geheim voorkomen. Hij kan bijvoorbeeld met een aandeelhouder i^* een getal a^* afspreken en dat de som van de overige aandelen a^* bedraagt. Uiteraard is a^* een random gekozen getal, waardoor deze coalitie niet te detecteren is. Het gevolg is dat niet alleen de aandeelhouders samen over het geheim kunnen beschikken, maar ook aandeelhouder i^* alleen.

   Dit probleem is op te lossen door het geheim niet door één partij te laten kiezen, maar in een verdeelprotocol dat door de aandeelhouders samen wordt uitgevoerd (sectie 9.2.4).

Verifieerbaar rekenen met aandelen. Met publieke aandelen kunnen ook berekeningen met aandelen worden gecontroleerd, mits deze zich vertalen naar relaties tussen de publieke versies ervan. Protocol 9.7 bevat protocol 9.3, maar is aangevuld met verificatie. De waarde $x[i]$ van huis i is een expliciet geheim, zodat we moeten aannemen dat de publieke versie $y[i]$ ervan al beschikbaar is (er geldt dus $y[i] = g^{x[i]}$).

	Bewoner i
1	Kies n getallen $s_j[i]$ $(j = 1 \ldots n)$ met som $x[i]$.
2	Publiceer publieke versies $t_j[i] = g^{s_j[i]}$.
3	Controleer dat voor j geldt: $\prod_l t_l[j] = y[j]$.
4	Zend $s_j[i]$ naar bewoner j.
5	Ontvang $s_i[j]$ van j en controleer $g^{s_i[j]} = t_i[j]$.
6	Bereken $U_i := \sum_j s_i[j]$.
7	Publiceer U_i
8	Lees alle U_j en controleer $g^{U_j} = \prod_l t_j[l]$.
9	$SOM := \sum_j U_j$; $AVE := SOM/n$

Protocol 9.7: Verifieerbare villabewoners.

In stap 1 verdeelt bewoner i de waarde van van zijn huis en in stap 2 publiceert hij de publieke aandelen, die in stap 3 door de anderen worden gecontroleerd. Stap 4 en 5 zijn de verspreiding en controle van de private aandelen, nog steeds volgens Micali's protocol. Stappen 6 en 7 berekenen en publiceren een som, maar dankzij de publieke aandelen kan het antwoord worden gecontroleerd en dat gebeurt in stap 8.

9.2.3 Groepscryptografie

De vorige sectie liet zien dat er met geheimen kan worden gerekend zonder ze te reconstrueren en zelfs dat dit op een verifieerbare manier kan. Deze sectie gaat verder in op een specifieke berekening met het geheim, namelijk het ontcijferen van een bericht. Het gedeelde geheim wordt als sleutel gebruikt, maar zonder het te reconstrueren.

We spreken hier van *groepscryptografie* omdat de aandeelhouders alleen als groep gebruik maken van hun geheime sleutel. De gebruikte cryptografische functies zijn gelijk aan de normale ElGamal-functies (zie sectie 6.3.2), zodat de groep naar buiten toe transparant is (dat wil zeggen, als één partij te zien is).

Het ElGamal-algoritme gebruikt een geheime sleutel a en een publieke sleutel b, gelijk aan g^a. Bij groepscryptografie wordt de publieke sleutel op de normale manier verspreid en deze kan door iedereen worden gebruikt om berichten te versleutelen. Voor het versleutelen van getal x wordt x vermenigvuldigd met een blinder $z = b^k$ (met k random), waarna $u = g^k$ en $v = z \cdot x$ samen worden opgestuurd.

Uit het paar (u, v) kan x met gebruikmaking van a worden berekend als $x = v/u^a$, maar a is verdeeld zodat de aandeelhouders alleen samen kunnen ontsleutelen.

Ontcijferen volgens ElGamal. Het geheim a wordt alleen gebruikt als exponent en daarom is de ontsleuteling goed te distribueren. Omdat $u^{\sum a_i} = \prod u^{a_i}$, hoeft elke aandeelhouder alleen maar de waarde $z_i = u^{a_i}$ op te sturen. Belangrijk op te merken is, dat het insturen van deze waarde niet het private aandeel blootgeeft omdat a_i de logaritme van z_i is (met grondtal u) en dus niet is te berekenen.

De ontsleuteling wordt gecoördineerd door een partij die we coördinator noemen. Deze hoeft niet over geheime waarden te beschikken, en vaak zal het een van de aandeelhouders zijn.

Coördinator		Aandeelhouder i.
Publiceer (u, v)	1	Lees (u, v) (en controleer $u^q = 1$)
Ontvang z_j van elke aandeelhouder	2	Zend $z_i = u^{a_i}$ aan coördinator
$z := \prod_j z_j$; $x = v/z$	3	

De coördinator kan met een gekozen-cijfertekstaanval (als beschreven in sectie 6.3.2) de aandelen proberen uit te rekenen. De test $u^q = 1$ beveiligt hiertegen: als de gelijkheid niet geldt, geven de aandeelhouders hun z_i niet. Deze test moet bij alle protocollen in dit hoofdstuk worden uitgevoerd (maar wordt kortheidshalve niet steeds vermeld).

Verificatie bij ontsleuteling. De coördinator ontvangt van aandeelhouder i waarde z_i, die geacht wordt te zijn berekend als u^{a_i}. De coördinator kent a_i niet, maar beschikt wel over de waarde b_i die ook met a_i is berekend, namelijk als $b_i = g^{a_i}$. De eerlijkheid van de aandeelhouder is equivalent met de gelijkheid van de twee exponenten, ofwel dat g, b_i, u en z_i aan de Diffie-Hellman-relatie (sectie 6.2.1) voldoen. De aandeelhouder kent bovendien de verborgen exponent, en kan daarom een zero-knowledge proof van geldigheid van *DifHel*(b_i, u, z_i) geven.

Voor het bewijzen van kennis van een logaritme is Schnorrs zero-knowledge proof (protocol 8.7) te gebruiken. De verificatiestap door Bob wordt uitgebreid zodat de coördinator controleert dat het niet alleen om ${}^{g}\log b_i$ maar ook om ${}^{u}\log z_i$ gaat:

Coördinator		Aandeelhouder i
Ontvang s en t	1	Kies r en zend $s = g^r$ en $t = u^r$.
Zend random challenge c	2	Ontvang c
Ontvang y	3	Zend $y = r + c \cdot a_i$
Controleer $\begin{cases} g^y = s \cdot b_i^c \\ u^y = t \cdot z_i^c \end{cases}$	4	

Stelling 9.7 *Dit protocol is een zero-knowledge proof (tegenover een eerlijke coördinator) van* *DifHel*(b_i, u, z_i).

Bewijs. Het protocol moet voldoen aan *acceptatie* (als de aandeelhouder eerlijk is accepteert de coördinator), *weigering* (als de aandeelhouder vals speelt is de acceptatiekans uiterst klein) en *zero-knowledge* (de conversatie is te construeren zonder aandeelhouder).

Acceptatie: Als de aandeelhouder eerlijk is, geldt $g^y = g^{r+c\cdot a_i} = g^r \cdot (g^{a_i})^c = s \cdot b_i^c$ en $u^y = u^{r+c\cdot a_i} = u^r \cdot (u^{a_i})^c = t \cdot z_i^c$. Het bewijs wordt dus geaccepteerd.

Weigering: Veronderstel dat de aandeelhouder, na het geven van s en t, op twee verschillende challenges, c_1 en c_2, een antwoord kan geven, namelijk y_1 en y_2. Dan geldt

$$\begin{array}{lll} g^{y_1} = s \cdot b_i^{c_1} & \text{en} & g^{y_2} = s \cdot b_i^{c_2} \\ u^{y_1} = t \cdot z_i^{c_1} & \text{en} & u^{y_2} = t \cdot z_i^{c_2}. \end{array}$$

Uit de eerste twee relaties volgt $g^{y_1-y_2} = b_i^{c_1-c_2}$ en dus $b_i = g^{\frac{y_1-y_2}{c_1-c_2}}$. Uit de volgende twee relaties volgt $u^{y_1-y_2} = z_i^{c_1-c_2}$ en dus $z_i = u^{\frac{y_1-y_2}{c_1-c_2}}$. Dit impliceert *DifHel*(b_i, u, z_i) (en dat de aandeelhouder de verborgen exponent kent).

Wanneer de Diffie-Hellman-relatie niet geldt, is er dus hoogstens één waarde van c waarvoor de aandeelhouder een acceptabel antwoord kan geven, en de kans dat die challenge gekozen wordt is $1/q$.

Zero-knowledge: De uitgewisselde getallen zijn (s, t, c, y) en zo'n viertal met willekeurige c kan zo worden gekozen: (1) Kies random y en c. (2) Bereken $s = g^y/b_i^c$. (3) Bereken $t = u^y/z_i^c$.

Coördinator		Aandeelhouder i.
Publiceer (u, v)	1	Lees (u, v)
	2	$z_i := u^{a_i}$
	3	Neem random r; $s_i := g^r$; $t_i := u^r$
	4	$c := H(s_i, t_i)$; $y_i := r + c \cdot a_i$
Ontvang (z_j, s_j, t_j, y_j)	5	Zend (z_i, s_i, t_i, y_i)
$c_j := H(s_j, t_j)$	6	
Controleer $\begin{cases} g^{y_j} = s_i \cdot b_i^{c_j} \\ u^{y_j} = t_i \cdot z_i^{c_j} \end{cases}$	7	
$z := \prod_j z_j$; $x = v/z$	8	

Protocol 9.8: Verifieerbaar ontsleutelen in veto-ElGamal.

De getallen voldoen aan $g^y = s \cdot b_i^c$ en $u^y = t \cdot z_i^c$, en kennis noch gelijkheid van de logaritmen is nodig om ze te vinden. □

Om het bewijs niet-interactief te maken gebruiken we (als in sectie 8.4.2) een hashfunctie H, waarmee de challenge c wordt berekend als $c = H(s, t)$. Er is dan minder interactie nodig tussen coördinator en aandeelhouder, en de zero-knowledge-eigenschap hangt niet meer af van eerlijkheid van de coördinator.

De inzending van de aandeelhouder bestaat dus uit z_i en een niet-interactief zero-knowledge proof van de juistheid van z_i; zie protocol 9.8. Na controle van de bewijzen kan de coördinator het bericht ontsleutelen.

9.2.4 Sleutelbeheer in een groep

Tot slot van deze sectie over groepscryptografie met een additioneel verdeelschema behandelen we enkele procedures ter ondersteuning van het veilig beheer van sleutels.

Het grootste bezwaar van het additieve schema blijft de kwetsbaarheid voor afwezigheid van een of meer aandeelhouders, en dit wordt met deze procedures niet opgeheven. Daarvoor is een drempelschema nodig zoals behandeld in sectie 9.3. De hier behandelde procedures kunnen ook voor dit complexere schema worden uitgevoerd, maar die aangepaste procedures behandelen we niet. De procedures in deze sectie kunnen dus worden gezien als vereenvoudigde versies van procedures die het sleutelbeheer bij drempelgroepscryptografie kunnen ondersteunen.

Delen zonder deler. Tot nu toe veronderstelden we dat de sleutel door een partij werd gegenereerd en vervolgens verdeeld, maar hieraan zijn verschillende risico's verbonden:

1. Een oneerlijke deler kan een sleutel genereren die niet random is gekozen, waardoor een zwakte kan ontstaan.

2. Een oneerlijke deler kan de sleutel na het genereren laten uitlekken.

3. De aandelen moeten worden verstuurd van deler naar aandeelhouders, waarbij ze misschien kunnen uitlekken.

Omdat de sleutel een impliciet geheim is, is een alternatief mogelijk waarbij de sleutel impliciet wordt gedefinieerd door keuzen van de aandeelhouders zonder ooit op een centrale plaats

bekend te zijn geweest. Hiertoe kiest elke aandeelhouder *zelf* zijn aandeel a_i als een random getal, en berekent en publiceert hij het publieke deel $b_i = g^{a_i}$. De publieke sleutel wordt door de aandeelhouders berekend als $b = \prod b_i$.

Deze manier van sleutelgeneratie lost alle bovengenoemde problemen op:

1. Zelfs als ook maar één van de aandeelhouders zijn aandeel echt als een random getal kiest, is het geheim a gedefinieerd door $a = \sum a_i$ een random getal.

2. Geen aandeelhouder of coalitie van $k-1$ aandeelhouders kan de sleutel laten uitlekken omdat $k-1$ aandelen geen informatie over de sleutel geven.

3. Elke aandeelhouder kiest zijn aandeel zelf en er is zodoende dus nooit verzending van private aandelen nodig.

Nog steeds kan een aandeelhouder het schema verstoren, bijvoorbeeld door een b_i te geven waarvan hij de logaritme niet kent, of door later niet aan ontsleuteling mee te werken. Hij kan dit echter niet ongemerkt doen.

Herverdeling van het geheim. Help! De aandeelhouders vrezen dat een of meer aandelen zijn uitgelekt, mogelijk omdat er een verrader onder hen is. Als de tegenpartij erin slaagt ook de andere aandelen te lezen, kan hij het geheim reconstrueren. Om dit te voorkomen willen de aandeelhouders het geheim *herverdelen*. Dat wil zeggen dat de oude aandelen a_j worden vernietigd, en dat de aandeelhouders nieuwe aandelen a_j^* krijgen die hetzelfde geheim verdelen (dus $\sum a_j^* = \sum a_j$).

Wat wordt er met herverdeling bereikt? Omdat de nieuwe aandelen dezelfde sleutel definiëren kunnen ze na herverdeling dezelfde publieke sleutel blijven gebruiken. Omdat de oude en nieuwe aandelen geen relatie met elkaar hebben, kan de tegenpartij geen oude en nieuwe aandelen combineren om het geheim te berekenen. De uitgelekte a_j worden dus waardeloos.

In vier stappen zullen we een procedure ontwikkelen die de herverdeling uitvoert onder aannamen dat (1) ten minste één aandeelhouder eerlijk is; (2) verraders niet ontmaskerd willen worden.

1. *Reconstructie en nieuwe uitgifte.* Een aandeelhouder (of buitenstaander) wordt als coördinator aangewezen. Elke aandeelhouder stuurt zijn aandeel naar de coördinator, die het geheim reconstrueert (met de reconstructiefunctie) en opnieuw verdeelt (volgens het verdeelprotocol).

   Deze methode (en elke andere die het geheim reconstrueert) is natuurlijk onacceptabel: de coördinator kan een verrader zijn en het geheim is dan meteen verloren.

2. *Het bijtellen van een nulgeheim.* De coördinator leidt het veranderen van de aandelen zonder ze te zien: hij deelt namelijk *differentiële* aandelen d_j van het getal nul uit, en aandeelhouder i berekent $a_i^* = a_i + d_i$. Omdat $\sum d_j = 0$ (immers de coördinator deelt 0 uit) is voldaan aan $\sum a_j^* = \sum a_j$. Aanvallers die oude en een aantal nieuwe aandelen combineren vinden als som a plus enkele random getallen, en hieruit is a niet te vinden.

   De coördinator ziet in deze oplossing niet het geheim, maar heeft toch nog te veel macht. (1) Hij kan het geheim *veranderen* in $a + d$ door niet 0 maar een getal $d \neq 0$ te delen. (2) Hij kent met de d_j de relatie tussen oude en nieuwe aandelen, waarmee hij aanvallers kan helpen die oude en nieuwe aandelen willen combineren. Het eerste probleem lossen we op met verificatie (oplossing 3), het tweede door elke aandeelhouder een 0 te laten delen (oplossing 4).

	Aandeelhouder i
1	Kies $d_j[i]$ voor $j = 1 \ldots n$ met $\sum_j d_j[i] = 0$
2	Bereken $e_j[i] := g^{d_j[i]}$
3	Publiceer $e_j[i]$ en stuur $d_j[i]$ aan j
4	Lees alle $e_l[j]$ en ontvang $d_i[j]$ van j
5	Controleer dat voor elke j geldt $\prod_l e_l[j] = 1$
6	Controleer dat voor elke j geldt $g^{d_i[j]} = e_i[j]$
7	Bereken $a_i^* = a_i + \sum_j d_i[j]$
8	Bereken voor alle j: $b_j = b_j \cdot \prod_l e_j[l]$
9	Wis a_i, alle $d_i[j]$ en alle b_j

Protocol 9.9: Herverdeling van aandelen.

3. *Verifieerbaar bijtellen van een nulgeheim.* De coördinator verstrekt private aandelen d_j van het getal 0, en publieke aandelen e_j. De aandeelhouders controleren dat $\prod e_j = 1$ (globale consistentie) en aandeelhouder i controleert $g^{d_i} = e_i$. Als dit klopt berekent de aandeelhouder $a_i^* = a_i + d_i$, en de publieke aandelen worden aangepast als $b_j^* = b_j \cdot e_j$.

   De coördinator heeft bij verificatie geen mogelijkheid meer om het geheim te veranderen, maar hij kent nog steeds de relatie tussen oude en nieuwe aandelen.

4. *Verifieerbaar bijtellen van k nulgeheimen.* De relatie tussen oude en nieuwe aandelen wordt zelf ook verdeeld over de aandeelhouders door *elke aandeelhouder* als deler van 0 te laten optreden; zie protocol 9.9.

   Het differentiële aandeel van i voor j heet $d_j[i]$ en heeft publiek deel $e_j[i]$. De relatie $\sum_j d_j[i] = 0$ laat zich verifiëren als $\prod_j e_j[i] = 1$. Geen aandeelhouder krijgt daardoor kans ongemerkt het geheim te veranderen door iets anders dan 0 te delen. Doordat elke aandeelhouder 0 deelt blijft het geheim gelijk.

   Zij nu $N \subset \{1, \ldots, k\}$ en veronderstel dat de tegenpartij de beschikking krijgt over oude aandelen a_j waar $j \notin N$ en de nieuwe aandelen a_j^* waar $j \in N$. De som van al deze aandelen is $a + \sum_{j \in N} \sum_i d_j[i]$ en zolang er ook maar één aandeelhouder i is die de waarden $d_j[i]$ niet afstaat is hieruit geen reconstructie mogelijk.

9.3 Een drempelschema: Shamirs polynoomschema

Bij een vetoschema zijn alle uitgedeelde aandelen weer nodig voor reconstructie, maar bij een drempelschema kunnen er extra aandelen worden uitgedeeld, waarna er toch een vast aantal voor reconstructie kan worden gebruikt. Een drempelschema is flexibeler dan een vetoschema omdat voor reconstructie of gebruik van het geheim niet de medewerking van *elke* aandeelhouder is vereist. Het eerste voordeel is dat het geheim ook bij afwezigheid van enkele aandeelhouders beschikbaar is. Bovendien is de bescherming tegen bedriegende aandeelhouders veel groter. Door verificatie kan bedrog worden opgemerkt, waarna de valse inbreng kan worden genegeerd en met behulp van correcte inbreng (mits het aantal k bedraagt) berekeningen kunnen worden voltooid.

Kader 9.10: Geheime stemming: verdelen van stem.

Nog uitdagender dan 'de villabewoners' is het houden van een geheime stemming. In het eenvoudigste geval is de keuze van de stemmer beperkt tot VOOR (waarde 1) of TEGEN (waarde 0) en het tellen van de stemmen is vergelijkbaar met het sommeren van de huizenprijzen. De geheimhouding van de stemming kan, net als bij de villabewoners, worden verkregen door verdeling van de stemmen. Het uitbrengen van de stem gebeurt door de stem te delen over een aantal stembureaus. De stembureaus krijgen van elke stem alleen een aandeel en kunnen dus niet zien waarop een kiezer stemt, maar ze kunnen de aandelen wel optellen. Alleen de uitslag wordt gereconstrueerd en bekendgemaakt.
Omdat stembureaus kunnen uitvallen moet een drempelschema worden gebruikt. Omdat verkiezingen controleerbaar eerlijk moeten verlopen, is aandeelhouderverifieerbaarheid nodig. En: omdat iedereen maar één keer mag stemmen is deler-verifieerbaarheid nodig, en dat is in deze oplossing moeilijk te verkrijgen. Een publieke stem kan niet worden gepubliceerd omdat het aantal mogelijke waarden klein is, zodat uit de publieke stem de private stem kan worden gevonden.
Vrije verkiezingen worden in sectie 14.3 behandeld.

Steeds is k het aantal aandelen dat nodig is, en n het totale aantal aandeelhouders; er wordt dan gesproken van een k-uit-n-schema. Het bekendste drempelschema is het polynoomschema van Shamir [Sha79], waarin tussen het geheim en de aandelen de volgende relatie bestaat:

de punten $(0, a)$ en (i, a_i) liggen op een polynoom van graad $k-1$.

Het geheim wordt dus gecodeerd in een $(k-1)^{\text{e}}$-graads kromme die de y-as snijdt in het punt $(0, a)$. Deze relatie kan nog verder worden uitgewerkt door de coëfficiënten van het polynoom expliciet te maken:

$$\exists f_0, \ldots, f_{k-1} : \ [\forall i : a_i = \sum_l f_l \cdot i^l] \ \wedge \ [f_0 = a].$$

9.3.1 Delen en reconstructie

De definitie van de graad van een polynoom wijkt iets af van de gebruikelijke, omdat voor een graad $k-1$ niet wordt verlangd dat de coëfficiënt f_{k-1} ongelijk is aan 0:

Definitie 9.8 *Een* polynoom van graad $k-1$ *is een functie van de vorm*

$$f(z) = f_0 + f_1 \cdot z + \ldots + f_{k-1} \cdot z^{k-1} = \sum_l f_l \cdot z^l.$$

De getallen f_l heten de coëfficiënten *van f.*
Het nulpolynoom *is het polynoom waarvan alle coëfficiënten gelijk 0 zijn (en dus $f(z) = 0$ voor elke z).*
Een nulpunt *van f is een oplossing in z van $f(z) = 0$.*

Een polynoom van graad $k-1$ dat niet het nulpolynoom is heeft ten hoogste $k-1$ nulpunten (stelling 5.15). De som en het verschil van polynomen van graad $k-1$ zijn weer polynomen van graad $k-1$. Zowel de reconstructie als de verhulling van dit schema zijn gebaseerd op dit bekende feit, dat verderop bij de reconstructieprocedure bewezen zal worden.

Feit 9.9 *Door elk k-tal punten met verschillende x-coördinaat gaat precies één polynoom van graad $k-1$.*

Het delen. De deler kiest een willekeurig polynoom f door $(0, a)$ en geeft aandeelhouder i het punt $(i, f(i))$:

	Deler
1	Neem $f_0 = a$ en $f_1, \ldots, f_{k-1}$ random
2	Laat $a_j = \sum_l f_l \cdot j^l$
3	Zend (j, a_j) aan aandeelhouder j

Omdat voor reconstructie k aandelen nodig zijn, moeten er ten minste zo veel worden uitgedeeld, maar elk aantal $n \geq k$ is mogelijk. In deze procedure wordt voor aandeelhouder j het polynoom f berekend in punt j; het is ook mogelijk voor elke aandeelhouder een willekeurig punt x_j te kiezen. Omdat $f(0)$ de waarde a is wordt er natuurlijk nooit een aandeel $(0, f(0))$ uitgedeeld.

Stelling 9.10 *Het polynoomschema is perfect verhullend.*

Bewijs. Veronderstel dat een partij beschikt over $k-1$ aandelen, dus punten $(x_1, y_1), \ldots, (x_{k-1}, y_{k-1})$ van het polynoom; zij x_k een waarde die nog niet als x-coördinaat voorkomt.

Voor een willekeurige waarde van a kan aan de $k-1$ punten als k^{e} punt $(0, a)$ worden toegevoegd. Volgens feit 9.9 gaat er door deze punten één polynoom f, zodat de waarde a mogelijk is als reconstructiewaarde. De waarde a wordt namelijk verkregen door het aandeel $(x_k, f(x_k))$ toe te voegen. □

Reconstructie. Uit k punten op het polynoom kunnen de k coëfficiënten ervan worden berekend, waarna het polynoom in het punt 0 kan worden berekend om het geheim te vinden. In plaats daarvan leiden we een formule af die rechtstreeks, zonder tussenkomst van de coëfficienten, het geheim oplevert.

Stelling 9.11 *Laten $(x_1, y_1), \ldots, (x_k, y_k)$ punten zijn op een polynoom f van graad $k-1$. Dan snijdt f de y-as in het punt $(0, a)$ waar*

$$a = \sum_{i=1}^{k} y_i \cdot \prod_{j \neq i} \left(\frac{-x_j}{x_i - x_j} \right).$$

Bewijs. Met de k gegeven punten kan de volgende functie in z worden gedefinieerd:

$$A(z) = \sum_{i=1}^{k} y_i \cdot \prod_{j \neq i} \left(\frac{z - x_j}{x_i - x_j} \right).$$

De waarden x_i en y_i zijn hierin constanten.

1. A is een polynoom van graad $k-1$. Omdat x_i en x_j constanten zijn, is elke factor $\frac{z-x_j}{x_i-x_j}$ een polynoom van graad 1. Voor een vaste i worden deze polynomen vermenigvuldigd voor $j \neq i$, er worden dus $k-1$ polynomen van graad 1 vermenigvuldigd en daarom is $\prod_{j\neq i} \left(\frac{z-x_j}{x_i-x_j}\right)$ een polynoom van graad $k-1$. Vermenigvuldiging met de constante y_i doet de graad niet toenemen dus is ook $y_i \cdot \prod_{j\neq i} \left(\frac{z-x_j}{x_i-x_j}\right)$ een polynoom van graad $k-1$. Het optellen van deze polynomen verhoogt de graad niet dus is $\sum_i y_i \cdot \prod_{j\neq i} \left(\frac{z-x_j}{x_i-x_j}\right)$ een polynoom van graad $k-1$.

2. Voor elk van de k punten (x_l, y_l) is $A(x_l) = y_l$. Immers

$$\begin{array}{rcll} A(x_l) & = & \sum_i y_i \cdot \prod_{j\neq i} \left(\frac{x_l-x_j}{x_i-x_j}\right) & \text{neem } z = x_l \\ & = & \sum_{i\neq l} y_i \cdot \prod_{j\neq i} \left(\frac{x_l-x_j}{x_i-x_j}\right) + y_l \cdot \prod_{j\neq l} \left(\frac{x_l-x_j}{x_l-x_j}\right) & \text{splits som} \\ & = & \sum_{i\neq l} y_i \cdot 0 + y_l \cdot 1 & \textit{toelichting} \\ & & & \text{en } \frac{x_l-x_j}{x_l-x_j} = 1 \\ & = & y_l. & \end{array}$$

Toelichting stap 3: Voor $i \neq l$ komt in het product $\prod_{j\neq i} \left(\frac{x_l-x_j}{x_i-x_j}\right)$ eenmaal (namelijk voor $j = l$) de factor $\frac{x_l-x_l}{x_i-x_l}$ voor en die is 0, zodat het hele product 0 is.

3. Het gedefinieerde polynoom A is gelijk aan f. Het verschil van de twee polynomen is een polynoom van graad $k-1$. Uit punt 2 blijkt dat dit verschil ten minste k nulpunten heeft, want $A(x_l) - f(x_l) = 0$ voor elk van de k punten waarvoor f gegeven was. Omdat een polynoom van graad $k-1$ niet meer dan $k-1$ nulpunten heeft wanneer het niet het nulpolynoom is, is dit verschil dus *wel* het nulpolynoom, waaruit $A = f$ volgt.

4. $A(0) = \sum_{i=1}^{k} y_i \cdot \prod_{j\neq i} \left(\frac{-x_j}{x_i-x_j}\right)$. Vul $z = 0$ in in de definitie van A. □

De reconstructie van a uit k aandelen (x_l, a_l) verloopt dus volgens de formule

$$a = \sum_{i=1}^{k} a_i \cdot \prod_{j \neq i} \left(\frac{-x_j}{x_i - x_j} \right).$$

Het product $\prod_{j\neq i} \left(\frac{-x_j}{x_i-x_j}\right)$ noemen we de *quorumfactor* van i en korten we af tot w_i. De reconstructie is dan te schrijven als $a = \sum_i a_i \cdot w_i$. Dit lijkt op de formule $a = \sum a_i$ die voor het vetoschema werd gebruikt, maar nu moet elk aandeel a_i worden vermenigvuldigd met de quorumfactor. De quorumfactor van i kan per reconstructie verschillen, omdat hij afhangt van welke aandeelhouders meedoen in de reconstructie. Hij hangt echter alleen af van *welke* aandelen er zijn (dat wil zeggen, de x_j) en *niet* van de *waarde* van de overige aandelen (de a_j).

9.3.2 Verifieerbaarheid

Het additieve schema kende twee controles bij het delen, namelijk de interne consistentie van de aandelen (de relatie tussen privaat en publiek aandeel) en de globale consistentie (de aandelen stellen het echte geheim voor). Bij het polynoomschema is een derde controle nodig en die heeft

Deler		Aandeelhouder i
Neem $f_0 = a$ en $f_1, \ldots, f_{k-1}$ random	1	
Publiceer $e_l = g^{f_l}$ $(l < k)$	2	Controleer $e_0 = b$
Laat $a_j = \sum_l f_l \cdot j^l$ $(j \leq n)$	3	
Publiceer $b_j = g^{a_j}$	4	Controleer $b_j = \prod_l e_l^{j^l}$
Stuur a_j aan j	5	Ontvang a_i en controleer $g^{a_i} = b_i$

Protocol 9.11: VERIFIEERBAAR DELEN IN POLYNOOMSCHEMA.

ermee te maken dat niet elk n-tal punten als verzameling aandelen kan voorkomen. Anders gezegd, de deler kan niet alleen punten uitdelen die op een polynoom liggen dat niet door $(0, a)$ gaat, maar kan ook punten uitdelen die helemaal niet op één polynoom van graad $k-1$ liggen. In dat geval kan de uitkomst van de reconstructie afhangen van welke aandelen ervoor worden gebruikt. Er moet dus ook worden gecontroleerd of de n punten op één polynoom van graad $k-1$ liggen.

Net als in het additieve schema wordt verifieerbaarheid bereikt door naast de private aandelen a_i publieke tegenhangers b_i te gebruiken, waarmee de relatie tussen de private waarden wordt gecontroleerd. Het bestaan van de (geheime) coëfficiënten van het polynoom f wordt door de deler bewezen door publicatie van de publieke coëfficiënten $e_l = g^{f_l}$. De relatie

$$a_i = f_0 + f_1 \cdot i + f_2 \cdot i^2 + \ldots + f_{k-1} \cdot i^{k-1}$$

tussen de private aandelen en private coëfficiënten vertaalt zich in de relatie

$$b_i = e_0 \cdot e_1^i \cdot e_2^{i^2} \cdot \ldots \cdot e_{k-1}^{i^{k-1}}$$

tussen de publieke aandelen en coëfficiënten. Overeenkomstig het schema van Micali wordt de publieke relatie door de aandeelhouders geverifieerd, evenals de interne consistentie van hun eigen aandeel.

Het delen gaat volgens protocol 9.11, waarbij wordt aangenomen dat het geheim expliciet is en $b = g^a$ al bekend, of dat de deler het geheim a kiest en b zelf publiceert. De aandeelhouders controleren in stap 2 dat de constante term f_0 van het polynoom is wat hij moet zijn, dus dat het polynoom de juiste waarde codeert. In stap 4 controleren ze dat de aandelen inderdaad op een polynoom van graad $k-1$ liggen. In stap 5 controleren ze de interne consistentie van hun aandeel.

9.3.3 Groepscryptografie

Omdat de reconstructie in het polynoomschema enige gelijkenis vertoont met die in het additieve schema, is de groepsontsleuteling volgens veto-ElGamal (protocol 9.8) de basis van ontsleuteling in het drempelschema. De ontsleuteling bij drempel-ElGamal is gegeven als protocol 9.13 en is ontleend aan Desmedt en Frankel [DF89].

Neem weer aan dat een ElGamal-bericht (u, v) is ontvangen; de coördinator leidt de berekening van $x = v/u^a$. Wanneer de berekening wordt uitgevoerd met aandelen $(i_1, a_1), \ldots, (i_k, a_k)$ kan voor elk aandeel de quorumfactor w_l worden berekend en dan geldt $a = \sum_l a_l \cdot w_l$. Voor de macht u^a betekent dit dat hij kan worden gevonden als

$$u^a = u^{(\sum_l a_l \cdot w_l)} = \prod_l u^{a_l \cdot w_l} = \prod_l (u^{a_l})^{w_l} = \prod_l z_l^{w_l}.$$

Kader 9.12: KEY SHARING IN PGP.

Het beveiligingsprogramma PGP biedt in zijn commerciële versie een vorm van sleutelverdeling. Voor het gebruik van de sleutel wordt reconstructie toegepast bij de coördinator: elke aandeelhouder stuurt hiervoor zijn aandeel aan de coördinator. Het PGP-programma is zo gemaakt, dat de gebruiker deze aandelen niet kan zien, en dat de gereconstrueerde sleutel wordt gewist zodra het programma wordt afgesloten.
Toch is er bij PGP al een vorm van misbruik mogelijk waaraan de voordelen van verdeeld gebruik kunnen worden afgelezen. Veronderstel dat de aandeelhouders gezamenlijk besluiten een belangrijk contract te ondertekenen, en Alice wordt als coördinator aangewezen. Alice krijgt aandelen, reconstrueert de sleutel en tekent hiermee het contract. Zolang zij het programma niet afsluit kan zij de geheime sleutel blijven gebruiken, en hiermee dus meer en andere berichten ondertekenen dan die in de vergadering zijn goedgekeurd.

Dit betekent dat de berekening met a_l kan worden uitgevoerd voordat de quorumfactor bekend is, en dat bij het gebruik van de quorumfactor (door de coördinator) de waarde a_l niet meer nodig is. De inzending van elke aandeelhouder hoeft dus niet af te hangen van welke aandeelhouders meedoen aan de ontsleuteling, maar is (net als in het additieve schema) het getal $z_i = u^{a_i}$. De toevoeging van s_i, t_i en y_i als niet-interactief bewijs van juistheid is ook hetzelfde als in veto-ElGamal.

Het schema gaat afwijken van protocol 9.8 vanaf stap 7, de controle op de inzendingen. Een aandeelhouder kan niet ongemerkt valsspelen, maar in protocol 9.8 staat de coördinator machteloos als een aandeelhouder niets of een onjuiste waarde instuurt. In het drempelschema kan het aantal aandeelhouders n groter zijn dan k, en de coördinator hoeft maar k juiste inzendingen af te wachten. Ongeacht afwezigheid of bedrog van enkele aandeelhouders slaagt de ontsleuteling zodra er k goede inzendingen zijn ontvangen.

Zodra bekend is welke inzendingen voor de berekening worden gebruikt, bepaalt de coördinator per inzending de quorumfactor en hij vindt u^a, en daarmee x.

Coördinator		Aandeelhouder i.
Publiceer (u, v)	1	Lees (u, v)
	2	$z_i := u^{a_i}$
	3	Neem random r; $s_i := g^r$; $t_i := u^r$
	4	$c := H(s_i, t_i)$; $y_i := r + c \cdot a_i$
Ontvang (z_j, s_j, t_j, y_j)	5	Zend (z_i, s_i, t_i, y_i)
$c_j := H(s_j, t_j)$	6	
Controleer $\begin{cases} g^{y_j} = s \cdot b_j^{c_j} \\ u^{y_j} = t \cdot z_j^{c_j} \end{cases}$	7	
Gebruik k correcte (j, z_j)	8	
Bereken quorumfactor w_j	9	
$z := \prod_j z_j^{w_j}$; $x = v/z$	10	

Protocol 9.13: VERIFIEERBAAR ONTSLEUTELEN IN DREMPEL-ELGAMAL.

9.3.4 Sleutelbeheer voor groepscryptografie

Net als het vetoschema kent het drempelschema procedures om gezamenlijk een sleutel te kiezen, en om de sleutel te herverdelen. Omdat het aantal aandeelhouders willekeurig groter dan k kan zijn, kent drempelcryptografie ook procedures om een aandeelhouder toe te voegen. Uiteraard kunnen al deze zaken worden uitgevoerd zonder dat het geheim wordt gereconstrueerd, zonder dat een malafide deelnemer het geheim kan veranderen of beïnvloeden, en zonder dat een coalitie van minder dan k aandeelhouders het geheim kan berekenen.

Delen zonder deler. Elke aandeelhouder i kiest en deelt een geheim $a[i]$. Partij i kiest geheime coëfficiënten $f_0[i]$ $(= a[i])$, $\ldots$, $f_{k-1}[i]$ en geeft deelnemer j punt $a_j[i] = \sum_l f_l[i] \cdot j^l$. Elke deelnemer telt de ontvangen punten op, en de verkregen punten liggen op een polynoom dat het geheim $a = \sum_i a[i]$ verdeelt.

Herverdeling. Elke deelnemer kiest een polynoom $f^{(i)}$ dat 0 verdeelt (dus $f_0[i] = 0$) en geeft aan j het punt $d_j[i] = f^{(i)}(j)$. Deelnemer j telt bij a_j de waarden $d_j[i]$ voor alle i op. De publieke aandelen en publieke coëfficiënten worden bijgewerkt met publieke versies van de coëfficiënten $f_l[i]$ en de differentiële aandelen $d_j[i]$.

Bij de herverdeling worden aandeelhouders die betrapt zijn op valsspelen buitenspel gezet: zij krijgen geen differentiële aandelen, en kunnen dus hun aandeel niet aan het nieuwe polynoom aanpassen.

Het toevoegen van een aandeelhouder. Om een nieuwe partij als aandeelhouder m te kunnen toevoegen moet deze partij over de waarde $a_m = f(m)$ komen te beschikken. Deelnemer i kiest een polynoom $f^{(i)}$, dit keer met de eigenschap dat $f^{(i)}(m) = 0$. Hiervan geeft hij punten $d_j[i] = f^{(i)}(j)$ aan deelnemer j. Deelnemer j telt alle waarden $d_j[i]$ (tijdelijk) bij zijn aandeel op en nu liggen de punten (j, a_j^*), met $a_j^* = a_j + \sum_i d_j[i]$, op het polynoom $f^* = f + \sum_i f^{(i)}$.

Elke aandeelhouder stuurt zijn waarde a_j^* aan m, die hieruit de volledige functie f^* en daarmee de waarde $f^*(m)$ kan berekenen. Door de keuze van $f^{(i)}$ is er tussen f en f^* maar één relatie: $f^*(m) = f(m)$. De nieuwe aandeelhouder vindt daarom op deze manier zijn eigen aandeel, maar kan zijn kennis over f^* niet gebruiken om het geheim of andere aandelen te berekenen.

Verifeerbaar sleutelbeheer. Gedistribueerd delen, herverdeling en het toevoegen van een aandeelhouder hebben ook allemaal een verifieerbare versie. Voor al deze protocollen kan ervoor worden gezorgd dat valsspelende aandeelhouders worden ontmaskerd, waarna het protocol kan worden voortgezet zolang ten minste k aandeelhouders het protocol volgen.

Samenvatting en conclusies

Verdeelschema's worden in de cryptografie gebruikt om belangrijke informatie te beschermen tegen misbruik door een of enkele personen. De informatie wordt zo over meerdere partijen verdeeld, dat er ten minste een van te voren bepaald aantal personen moet meewerken aan het terugvinden of gebruiken van het geheim. Een belangrijke toepassing is groepscryptografie;

het verdelen van cryptografische sleutels, waarna het ondertekenen of ontsleutelen van een bericht alleen door meerdere personen tezamen kan worden gedaan.

Additioneel schema en polynoomschema. De twee verdeelschema's die in dit hoofdstuk zijn behandeld, zijn het additionele schema en het polynoomschema. Het eerste is eenvoudiger, maar het is niet mogelijk meer aandelen uit te delen dan die welke nodig zijn voor het gebruik van het geheim. Het schema is daardoor een *vetoschema*. In het polynoomschema kunnen wel meer aandelen worden uitgedeeld, zodat bij het gebruik van het geheim ook enkele aandeelhouders kunnen worden gemist. Het schema is daarom een *drempelschema*.

Verificatie. Een belangrijke toevoeging aan verdeelschema's is een verificatie-eigenschap, waarmee kan worden gecontroleerd of een aandeelhouder zijn aandeel eerlijk gebruikt. Voor verificatie zijn twee belangrijke technieken. Een *publiek aandeel* is een waarde die vanuit het geheim is berekend, en waarmee berekeningen met het geheim rechtstreeks kunnen worden gecontroleerd. Met een *zero-knowledge proof* kan een partij bewijzen eerlijk te spelen, zonder zijn geheime waarden te onthullen.

Het nut van verificatie is bij drempelschema's groter dan bij vetoschema's. Bij een vetoschema beschermt verificatie alleen tegen partijen die niet als valsspeler ontmaskerd willen worden. Een aandeelhouder kan zichzelf echter 'opofferen' door op zeker moment medewerking te weigeren of foute waarden in te sturen. Weliswaar wordt dit door de verifieerbaarheid ontdekt, maar het geheim is dan verloren: er zal een nieuw geheim gekozen moeten worden en oude berichten zijn verloren, met alle gevolgen van dien. Bij een drempelschema blijft het geheim beschikbaar, ook als er aandeelhouders valsspelen. Immers, het totale aantal aandeelhouders is groter dan het aantal dat voor het gebruik van het geheim moet meewerken.

Groeps-RSA en groepshandtekeningen. Dit hoofdstuk behandelde groepscryptografie met het ElGamal-algoritme, en niet zonder reden. Groepscryptografie met RSA is veel lastiger; de geheime functie is $D(y) = y^d \bmod n$, maar het is niet mogelijk de geheime exponent d volgens een al behandeld schema te verdelen. De reden is dat d een getal uit $\mathbb{Z}_{\phi(n)}$ is, maar geen van de aandeelhouders mag $\phi(n)$ kennen. Zoals in sectie 6.3.3 is beschreven, is kennis van $\phi(n)$ voldoende om de modulus te factoriseren, en daarmee de geheime sleutel volledig te berekenen.

Ook is het ingewikkelder om ondertekening van berichten in een groep te laten gebeuren. Het berekenen van een ElGamal-handtekening vanuit een verdeelde sleutel is erg gemakkelijk, maar de deelsleutels zijn uit hun deelresultaten te berekenen.

Opgaven bij hoofdstuk 9

Opgave 9.1 *Bewijs dat de uitkomst van protocol 9.3 juist is.*

Opgave 9.2 *Geef een verifieerbare versie van de villabewoners. Je mag aannemen dat van elk pand de waarde x_i privaat bekend is aan de eigenaar en dat de publieke waarde $y_i = g^{x_i}$ aan iedereen bekend is.*

Opgave 9.3 *Bewoners aan de even kant van de villawijk denken graag dat de huizen aan hun kant meer waard zijn dan die aan de oneven kant. Hoeveel even bewoners moeten, na afloop van protocol 9.3, samenwerken om de gemiddelde waarde van de oneven huizen te achterhalen?*

Project 9.4 *Schrijf een programma voor en een scriptie over visuele cryptografie.*

Opgave 9.5 *Bewijs stelling 9.5.*

Opgave 9.6 *Neem aan dat de prijs van een villa een bedrag onder de tien miljoen in hele euro's is.*
Laat zien dat protocol 9.7 de $x[i]$ niet verhult.
Geef een gewijzigd protocol dat wel verhullend is.

Opgave 9.7 *Laat zien dat protocol 9.8 geen verificatie biedt als c_i wordt berekend als $H(s_i)$ in plaats van $H(s_i, t_i)$.*

Opgave 9.8 *Bedenk een protocol voor het ondertekenen van een bericht met veto-groepscryptografie.*

Opgave 9.9 *De standaarddefinitie van een polynoom van graad $k-1$ eist dat de $(k-1)^{\text{e}}$ coëfficiënt (f_{k-1}) ongelijk 0 is. Laat zien dat een polynoomschema met deze definitie* niet *perfect verhullend is. Laat zien dat wel alle waarden uit $\mathbb{Z}_q$ nog mogelijk zijn voor a als er $k-2$ aandelen bekend zijn.*

Opgave 9.10 Polynoom-evaluatie. *Gegeven coëfficienten f_0 t/m f_{k-1} en z, allen in $\mathbb{Z}_q$, hoeveel bewerkingen (mod q) kost het om $f(z) = \sum_{l=0}^{k-1} f_l \cdot z^l$ te berekenen?*
Hint: $ax^2 + bx + c = (ax + b)x + c$.

Opgave 9.11 Polynoom-interpolatie in $\mathbb{Z}_7$. *Een deler heeft een tweedegraadspolynoom gekozen over $\mathbb{Z}_7$ en verdeelt vier aandelen: de punten (1,5), (2,6), (3,1) en (4,4).*
(a) Hoeveel aandelen zijn nodig om het geheim te vinden?
(b) Wat zijn de quorumfactoren bij reconstructie uit de x-wwarden 1, 2 en 3?
(c) Wat zijn de quorum-factoren bij 1, 2 en 4?
(d) Wat is de geheime waarde?
(e) Kun je de drie polynoom-coëfficienten vinden?

Project 9.12 *Is protocol 9.11, net als protocol 9.6, computationeel verhullend?*

Opgave 9.13 *Een atoombomsleutel moet worden verdeeld onder generaals en kolonels en wel zo, dat de sleutel moet kunnen worden gebruikt door twee generaals samen, een generaal met twee kolonels, of vijf kolonels. Beschrijf hoe dit kan worden gedaan.*

Opgave 9.14 Computationele Verifieerbaarheid. *In dit hoofdstuk keken we naar verificatie met een publiek aandeel b_i, gedefinieerd als $b_i = g^{a_i}$. De volgende oplossing behoudt de perfecte verhulling van het geheim a.*
Zij h een random *macht van g, dwz., de orde van h is q. Er bestaat een α waarvoor $g^\alpha = h$, maar deze α kent niemand. Voor de verificatie beschikt aandeelhouder i over een* randomizer *r_i, waarvoor geldt $b_i = g^{a_i} \cdot h^{r_i}$.*
(a) Laat zien hoe de deler de publieke aandelen en randomizers kan uitrekenen.
(b) Breidt het Reconstructie-protocol uit met controle van randomizers en publieke aandelen.
(c) Laat zien dat de perfecte verhulling blijft gelden: gegeven b_i is nog steeds elke waarde uit $\mathbb{Z}_q$ mogelijk als waarde voor a_i.
(d) Laat zien dat Computationele Verifieerbaarheid geldt: het vinden van een valse *a'_i en een valse randomizer r'_i die samen met a'_i door de controle komt, is zo moelijk als het berekenen van de logaritme van h.*
(e) Bedenk een NIZKP waarmee aandeelhouder i kan bewijzen, dat een ingestuurde z_i (voor ontsleuteling) juist is berekend.

Opgave 9.15 *Een sleutel moet worden verdeeld onder Tsjechische en Slowaakse ministers en wel zo, dat het geheim kan worden gebruikt als er drie Tsjechen én drie Slowaken aanwezig zijn.*

Opgave 9.16 *Een sleutel moet onder voetballers van achttien eredivisiclubs worden verdeeld en wel zo, dat de sleutel gebruikt kan worden als van meer dan de helft van de clubs (tien) een compleet team aanwezig is (elf spelers).*

Opgave 9.17 *Werk het protocol voor delen zonder deler in het drempelschema uit.*

Opgave 9.18 *Werk een verifieerbaar protocol uit voor delen zonder deler in het drempelschema.*

Opgave 9.19 *Bedenk een protocol voor het ondertekenen van een bericht met drempel-groepscryptografie.*

Hoofdstuk 10

Toepassingen

In de afgelopen hoofdstukken heeft de nadruk gelegen op de asymmetrische cryptografie en de toepassingen die daarmee kunnen worden ontwikkeld. In dit hoofdstuk worden enkele grote cryptografische systemen bekeken, waarbij zal blijken dat ook op basis van symmetrische encryptie krachtige en bruikbare systemen kunnen worden gebouwd. Deze systemen genieten het voordeel van een efficiënt te implementeren, maar snelle en sterke encryptie. Enkele beperkingen van symmetrische cryptografie zullen ook worden bekeken.

Kerberos (sectie 10.1) is een systeem waarmee in een *client-serveromgeving* de toegang tot servers beheerd kan worden. UMTS (sectie 10.2) is de opvolger van gsm als standaard voor mobiele communicatie. SSL (sectie 10.3) is een protocol dat communicatie met internetsites beveiligt.

Het is niet het doel van dit hoofdstuk om deze systemen zo volledig te beschrijven dat de lezer er als gebruiker mee aan de slag kan. Beoogd wordt te beschrijven hoe cryptografische technieken, deels eerder in dit boek beschreven, hun weg vinden in op grote schaal toegepaste beveiliging.

10.1 Kerberos

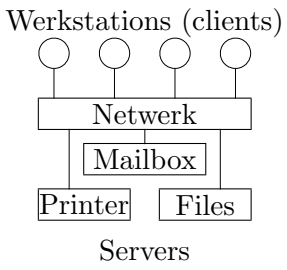
Kerberos is een authenticatiesysteem voor lokale netwerken met een client-serverarchitectuur, gebaseerd op een *Trusted Third Party*: een partij die door alle anderen wordt vertrouwd. Het beschermt servers tegen gebruik door niet-geauthoriseerde partijen, en clients tegen interactie met valse servers. Ook is voorzien in het genereren van een sessie-sleutel voor de communicatie tussen client en server, zodat indringers geen lopende sessies kunnen overnemen of afluisteren.

10.1.1 Doel en uitgangspunten van Kerberos

Gedurende de jaren tachtig werd de client-serverarchitectuur (kader 10.1) steeds vaker gebruikt in allerlei omgevingen, van bedrijf tot universiteit. De openheid van dergelijke netwerken schept een aantal problemen op het gebied van veiligheid.

1. **Clientauthenticatie.** De servers moeten hun diensten alleen verlenen aan gebruikers (clients) die in het systeem zijn geregistreerd, en bovendien de voor de dienst vereiste permissies hebben.

Kader 10.1: De client-serverarchitectuur



In een netwerk met client-serverarchitectuur worden allerlei hulpbronnen, zoals fileopslag en printers, aangestuurd via servers. Een gebruiker die vanachter zijn pc of werkstation een bestand wil afdrukken, doet dit door het bestand naar een printserver te sturen. Kenmerkend voor het client-servermodel is dat dienstverlenende programma's *passief* wachten op opdrachten.
Het onderscheid tussen clients en servers is niet strikt: een server kan, om de gevraagde dienst uit te voeren, als client bij een andere server aankloppen. Dit gebeurt bijvoorbeeld als de opdracht aan de printserver niet het bestand, maar slechts de naam ervan bevat: de printserver meldt zich als client voor de fileserver (disk) om het bestand te krijgen.
Een client-servernetwerk is erg open: dagelijks worden clients en servers toegevoegd of verwijderd. Ook is het vrij gemakkelijk voor buitenstaanders om toegang tot het netwerk te krijgen, bijvoorbeeld door een extra pc aan te sluiten of door een bestaande te modificeren.

2. **Serverauthenticatie.** Gebruikers moeten alleen diensten accepteren van de echte, geregistreerde servers en niet communiceren met willekeurige partijen.

3. **Vertrouwelijkheid.** De communicatie tussen client en server moet worden versleuteld, en hiervoor moeten client en server een sleutel delen.

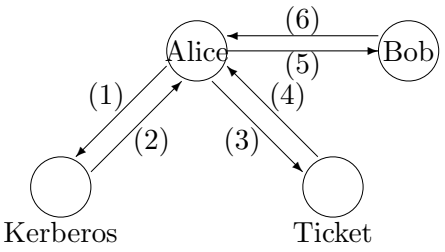
Authenticatie en sleutelbeheer kunnen in principe volledig tussen client en server worden geregeld, bijvoorbeeld door elke client een sleutel te laten delen met elke server. (Met publickeycryptografie kan worden volstaan met één sleutelpaar per gebruiker, maar daarbij moet elke server betrouwbaar over de publieke sleutels beschikken.) In de praktijk is dit echter niet voldoende flexibel omdat informatie over elke gebruiker op vele plaatsen (in elke server) wordt gedupliceerd. Telkens wanneer een nieuwe gebruiker in het systeem wordt ingevoerd, moeten alle servers over de gebruiker en zijn sleutel worden ingelicht. Als een gebruiker zijn password verandert moet hij dat op elke server afzonderlijk doen.

Kerberos vereenvoudigt het beheer door de informatie die op authenticatie betrekking heeft op één plaats te concentreren, namelijk in een extra server, de *authenticatieserver*. Deze Kerberos-server kent de (geheime) sleutel van elke gebruiker. Ook alle informatie over de permissies van de gebruikers is op een centrale plaats opgeslagen; om redenen die straks uitgelegd worden is dit een tweede speciale server, de *ticketserver*. Bij het toevoegen of verwijderen van een gebruiker of dienst, en bij het wijzigen van de permissies van een gebruiker hoeft alleen de authentication server, respectievelijk de ticketserver maar te worden bijgewerkt.

Globale werking: tickets. In de beschrijving van het systeem is Alice een client en Bob een server; Alice wil de dienst van Bob gebruiken. De authenticatieserver wordt hier Kerberos genoemd en deze beschikt over sleutels k_A en k_B die met Alice, respectievelijk Bob, worden gedeeld. Het mechanisme waarmee het contact tussen Alice en Bob tot stand wordt gebracht is te vergelijken met een bioscoop met meerdere filmzalen. Het afrekenen (met eventuele controle van kortingsrechten en reserveringen) gebeurt aan een centrale kassa, waarna de bezoeker een kaartje voor de gewenste filmzaal krijgt. Bij de filmzaal worden geen ingewikkelde controles

Kader 10.2: Interactie in Kerberos

Alle communicatie in Kerberos verloopt op initiatief van de client; er is geen communicatie tussen de verschillende servers.
(1 en 2) Alice gebruikt haar sleutel k_A om van Kerberos een ticket te krijgen voor de ticketserver.
(3 en 4) Met de toegang tot de ticketserver verkrijgt Alice een ticket voor Bob.
(5 en 6) Alice benadert Bob met het ticket en verkrijgt service en zekerheid over de echtheid van Bob.



meer uitgevoerd: het kaartje is voldoende om te worden toegelaten.

Om toegang tot server Bob te verkrijgen, vraagt Alice aan Kerberos een ticket voor Bob, en in deze fase gebruikt zij haar sleutel k_A. Vervolgens kan Alice dit kaartje aan Bob aanbieden om de gewenste dienst te verkrijgen. Meer dan in een bioscoop moet ervoor worden gewaakt dat het ticket alleen kan worden gebruikt door de partij voor wie het is afgegeven. Het ticket staat daarom op naam van Alice, en bij het aanbieden aan de server wordt extra authenticatie-informatie meegestuurd.

Zoals blijkt uit figuur 10.2 verloopt de werkelijke interactie iets ingewikkelder, namelijk via een tweede speciale server, de *ticketserver*. De belangrijkste reden voor het toevoegen van deze server is dat hierdoor Alice niet voor elk gebruik van een server haar password hoeft in te voeren. Het is niet wenselijk dat wanneer een gebruiker (Alice) uren of zelfs dagen achtereen met een werkstation op een netwerk is ingelogd, haar password in het station wordt bewaard. Daarom moet bij iedere handeling waarvoor k_A nodig is het password opnieuw worden ingevoerd; door de toevoeging van de ticketserver hoeft dit slechts eenmaal per loginsessie te gebeuren.

Wanneer Alice inlogt is er een eenmalige interactie met Kerberos, waarbij ze haar password (sleutel) gebruikt; Kerberos is de server die de sleutels van alle gebruikers (clients) en van de ticketserver kent. Haar werkstation komt in bezit van een ticket voor de ticketserver en een nieuwe sleutel k_{AT} waarmee dit ticket meerdere malen kan worden gebruikt. De sleutel en het ticket hebben een beperkte gebruiksduur en mogen in het werkstation worden bewaard. Telkens wanneer Alice een server wil gebruiken worden het ticket voor de ticketserver en de bijbehorende sleutel k_{AT} gebruikt om een server ticket te vragen. De ticketserver is de server die de sleutels van alle andere servers en de permissies van de gebruikers kent. Wordt een server Bob meermalen achtereen gebruikt, dan kan hiervoor een keer een ticket worden verkregen.

10.1.2 Het ticketprotocol

De berichten die uitgewisseld worden in Kerberos staan in protocol 10.3. Beschreven is versie 4; de nieuwere versie 5 voorziet in een iets grotere flexibiliteit, maar in de wijze waarop authenticatie wordt gerealiseerd, verschillen de versies niet. In het overzicht staat de notatie $E_X\langle y\rangle$ voor de encryptie van y onder sleutel k_X. De gebruikte symmetrische encryptie in versie 4 van Kerberos is DES, maar in versie 5 kunnen ook andere algoritmen worden gebruikt.

1. A $\rightarrow$ K : $\langle A,\ T\rangle$
2. K $\rightarrow$ A : $E_A\langle k_{AT},\ T,\ Ticket_T\rangle$,
   waar $Ticket_T = E_T\langle k_{AT},\ A,\ T\rangle$
3. A $\rightarrow$ T : $\langle B,\ Ticket_T,\ Auth_1\rangle$,
   waar $Auth_1 = E_{AT}\langle A\rangle$
4. T $\rightarrow$ A : $E_{AT}\langle k_{AB},\ B,\ Ticket_B\rangle$,
   waar $Ticket_B = E_B\langle k_{AB},\ A,\ B\rangle$
5. A $\rightarrow$ B : $\langle Ticket_B,\ Auth_2\rangle$,
   waar $Auth_2 = E_{AB}\langle A,\ x\rangle$
6. B $\rightarrow$ A : $E_{AB}\langle x+1\rangle$

Protocol 10.3: Berichten in het ticketprotocol

Aanvraag en gebruik van het eerste ticket. In stap 1 vraagt Alice een ticket voor de ticketserver door vermelding van haar naam A en de gewenste server T. Men zou verwachten dat zij hierbij al haar password gebruikt (bijvoorbeeld meestuurt) zodat Kerberos zich van de authenticiteit van de aanvrager kan overtuigen. De aannamen over de openheid van het netwerk en het gebruik van alleen symmetrische encryptie sluiten echter uit dat Alice haar password, of andere geheime informatie, over het netwerk stuurt. Verrassend gevolg is dat er bij de authenticatieserver Kerberos *geen sprake* is van authenticatie. Kerberos accepteert de aanvraag als Alice voorkomt onder de geregistreerde gebruikers en stuurt een ticket, zonder enige zekerheid dat bericht 1 inderdaad van Alice afkomstig is. Authenticatie vindt feitelijk pas plaats bij het gebruik van het ticket.

Het afgegeven ticket, $Ticket_T$, bevat de identiteit van Alice en de ticketserver. De eerste om te voorkomen dat een andere partij het ticket onderschept en gebruikt, de tweede om te voorkomen dat Alice een ticket voor een andere server gebruikt dan de ticketserver. Bovendien genereert Kerberos een nieuwe sleutel k_{AT} voor de communicatie tussen Alice en de ticketserver en deze sleutel is ook in het ticket opgenomen. Het ticket is versleuteld met k_T, de sleutel van de ticketserver. Aan Alice wordt een bericht (stap 2) gestuurd met het ticket en de sleutel k_{AT}, het geheel versleuteld onder k_A.

Het is deze laatste versleuteling die de feitelijke authenticatie van Alice tegenover de ticketserver verzorgt, want alleen Alice kan het bericht 2 lezen en over sleutel k_{AT} beschikken. In stap 3 vraagt Alice van de ticketserver een ticket voor Bob, en overlegt haar $Ticket_T$ en een authenticator $Auth_1$. De ticketserver ontvangt een ticket versleuteld onder k_T, en verkrijgt de zekerheid dat dit door Kerberos is gemaakt omdat geen andere partij k_T kent; de versleuteling onder k_T fungeert als een handtekening van Kerberos (vergelijk sectie 7.2.1). De ticketserver weet van Kerberos dat een sessiesleutel k_{AT} die in een ticket voor Alice is opgenomen, verder alleen versleuteld onder k_A wordt verstuurd. De authenticator bewijst (opnieuw als in sectie 7.2.1) dat de afzender k_{AT} kent, en dus indirect dat die over k_A beschikt. De ticketserver accepteert daarom het ticket en geeft een ticket voor Bob af (stap 4).

Gebruik van het tweede ticket. De structuur en gebruik van het tweede ticket komen vrijwel overeen met die van het eerste ticket. De ticketserver genereert een nieuwe sleutel k_{AB} die voor de communicatie tussen Alice en Bob gaat worden gebruikt; deze wordt opgenomen in $Ticket_B$ en versleuteld aan Alice gestuurd. Weer wordt het ticket versleuteld met de sleutel

van de betreffende server (nu k_B), en voor de verzending naar Alice wordt gebruikgemaakt van een sleutel die Alice en de ticketserver delen (nu k_{AT}). Alice, en alleen zij, kan bericht 4 ontsleutelen en naast het ticket over k_{AB} beschikken. Bob ontvangt een verzoek om service, tezamen met een ticket en een authenticator die bewijst dat de maker ervan over de sessiesleutel beschikt. (Een indringer kan dit bericht later opnieuw naar Bob sturen om Alice te 'verrassen' met een bericht van stap 6, maar door de versleuteling kan deze indringer hiermee geen service van Bob krijgen.)

Bij de laatste interactie is echter ook een mechanisme opgenomen waarmee Alice zekerheid krijgt over de echtheid van de server. Bij de interactie met Kerberos en de ticketserver is dit niet noodzakelijk, omdat Alice aan deze servers geen vertrouwelijke gegevens stuurt. Het contact met Bob is wel kritisch: ten eerste omdat Alice aan een server vertrouwelijke gegevens kan zenden, ten tweede omdat Alice moet kunnen vertrouwen op gegevens die zij van de server ontvangt.

Voor serverauthenticatie wordt een getal x gebruikt in Alice' authenticator. Bob kan dit getal lezen en moet het, opgehoogd met 1, versleutelen onder k_{AB} en terugsturen. Van het ontvangen bericht 6 heeft Alice de zekerheid dat het is verstuurd door een partij die k_{AB} kent, en dus indirect k_B.

Netwerkadressen en tijdstempels. Het bovenbeschreven gebruik van gedeelde sleutels is het belangrijkste veiligheidsmechanisme in Kerberos. Enkele aanvullende maatregelen moeten het afvangen en hergebruik van tickets en authenticators voorkomen.

In elk bericht is een tijdstempel opgenomen dat aangeeft wanneer het is verzonden. Is de ontvangsttijd te veel later dan het zendtijdstip, dan wordt het bericht geweigerd; hierdoor heeft een aanvaller slechts enkele seconden de tijd om een replay-aanval uit te voeren. Ook de authenticators bevatten een tijdstempel en zijn slechts enkele seconden geldig. Dit voorkomt dat een aanvaller bijvoorbeeld bericht (3) onderschept, en na het vertrek van Alice opnieuw aanbiedt aan de ticketserver.

Verder wordt, naast de identiteit van Alice, ook haar netwerkadres opgenomen in de tickets. Een ticket kan dan niet vanaf een ander werkstation worden gebruikt, omdat de servers weigeren een antwoord te sturen naar een ander adres dan dat in het ticket.

10.1.3 Evaluatie en BAN-logica

Kerberos is een ingenieus maar bruikbaar protocol. Het wordt in veel bedrijfsomgevingen gebruikt en biedt dan een flexibele bescherming tegen indringers in het netwerk. Toch kent het ook beperkingen, zowel van operationele als van wetenschappelijke kant, waarvan er nu een aantal wordt belicht.

1. *Afhankelijkheid van tijd.* De bescherming tegen replay-aanvallen vereist dat de klokken in het systeem binnen nauwe marges gelijklopen. Wanneer een aanvaller de klok van een werkstation kan laten voorlopen, of die van een server kan laten achterlopen, vergroot hij de tijd die hij heeft om een ticket of authenticator te hergebruiken. Betrouwbare gelijkloop van klokken in een netwerk is helaas op zich al een technisch zeer lastig probleem.

2. *Kerberos en ticketserver als trusted party.* De speciale servers (Kerberos en ticketserver) zijn van enorm belang voor de veiligheid: alle sleutels zijn er opgeslagen en een inbraak in een ervan geeft een aanvaller de controle over het hele netwerk. De servers moeten daarom extra worden beveiligd, zowel met firewalls als fysiek (door opstelling in een

Symbool	Gebruik	Betekenis
$\models$	$P \models X$	Partij P is overtuigd van de juistheid van X
$\triangleleft$	$P \triangleleft X$	Partij P ziet X, ofwel heeft de beschikking over X
$\models\!\!\Rightarrow$	$P \models\!\!\Rightarrow X$	Partij P heeft zeggenschap over X, maw., als P zegt X, dan geldt X
$\mid\!\sim$	$P \mid\!\sim X$	Partij P heeft (eerder gedurende het protocol) X gezegd
$\sharp$	$\sharp X$	Formule X is recent, dat wil zeggen, niet voorgekomen in eerdere runs van het protocol
$\overset{\cdot}{\longleftrightarrow}$	$P \overset{K}{\longleftrightarrow} Q$	Partijen P en Q delen K als geheime sleutel, met andere woorden, P en Q, en verder geen andere partij, kennen K.
$\overset{\cdot}{\models\!\!\Rightarrow}$	$\overset{K}{\models\!\!\Rightarrow} P$	Partij P heeft publieke sleutel K (de geheime sleutel is dan K^{-1})
$\rightarrow$	$P \rightarrow Q : X$	In het protocol stuurt P een bericht met vorm/inhoud X aan Q
	$\langle X \rangle_K$	Boodschap X, versleuteld onder K

Tabel 10.4: De taal van BAN-logica.

afgesloten ruimte). Ze kunnen zelf altijd van alle servers gebruik maken (onder hun eigen of valse naam).

3. *Symmetrische cryptografie.* Kerberos kent alle beperkingen die inherent zijn aan het gebruik van symmetrische cryptografie. Er is geen identificatie, slechts authenticatie. De Kerberos en ticketservers hebben de mogelijkheid ongeauthoriseerde gebruikers tickets af te geven en zo toegang tot servers te verlenen. De gevolgen van een gecompromitteerde server gaan echter verder dan het verlenen van ongeauthoriseerde toegang. Alle communicatie tussen geauthoriseerde gebruikers en echte servers gebruikt sleutels die door deze servers zijn aangemaakt. Door gecompromitteerde servers kan daarom alle vertrouwelijke communicatie uitlekken.

4. *Defensief ontwerp.* De veiligheidsmechanismen in Kerberos zijn ontworpen met het oog op het tegengaan van specifieke vormen van aanvallen zoals het onderscheppen van berichten of het genereren van nieuwe berichten. Er kan wel worden beredeneerd dat zulke aanvallen vervolgens niet kunnen worden uitgevoerd, maar het is erg moeilijk te bewijzen of te beredeneren wat in positieve termen het resultaat van een Kerberos-interactie is.

BAN-logica: afleidingssysteem voor authenticatieprotocollen. Omdat het erg moeilijk is om over Kerberos en soortgelijke protocollen te redeneren, zijn er verschillende redeneersystemen hiervoor ontworpen. We geven een zeer korte beschrijving van de BAN-logica van Burrows, Abadi en Needham [BAN90]. Deze logica kent een taal (waarin kennis van een partij, en het delen van sleutels kunnen worden geformuleerd) en afleidingsregels, vergelijkbaar met bewijsregels in de normale logica. De taal is samengevat in tabel 10.4.

De afleidingsregels geven aan welke redeneerstappen geldig zijn bij het redeneren over interacties tussen partijen; hier volgen er slechts enkele als voorbeeld.

1. $\mid\!\sim$*-introductie:* Als P gelooft dat hij sleutel k deelt met Q en een onder k versleuteld bericht ziet, gelooft hij dat Q X heeft gezegd. Als P gelooft dat K de geheime sleutel van Q is en een bericht ontvangt dat met K^{-1} is ondertekend, gelooft hij dat Q X heeft gezegd:
$$\frac{P \models (Q \overset{k}{\longleftrightarrow} P) \qquad P \triangleleft \langle X \rangle_k}{P \models (Q \mid\!\sim X)} \qquad \frac{P \models (\overset{K}{\models\!\Rightarrow} Q) \qquad P \triangleleft \langle X \rangle_{K^{-1}}}{P \models (Q \mid\!\sim X)}.$$

2. $\models\!\Rightarrow$*-eliminatie:* Als P gelooft dat Q zeggenschap heeft over X en bovendien X zegt, dan gelooft P zelf X:
$$\frac{P \models (Q \models\!\Rightarrow X) \qquad P \models (Q \mid\!\sim X)}{P \models X}.$$

3. $\triangleleft$*-introductie:* Als P een bericht aan Q stuurt ziet Q dit:
$$\frac{P \to Q : X}{Q \triangleleft X}.$$

4. $\sharp$*-introductie:* Als P een random getal genereert is dit recent:
$$\frac{P \text{ genereert } x}{P \models \sharp x}.$$

De regels $\mid\!\sim$-introductie en $\models\!\Rightarrow$-eliminatie worden bijvoorbeeld gebruikt bij het redeneren over een bericht ontvangen van een certificaatserver. Partij P kan deze server CA pas gebruiken als hij (1) gelooft dat het een gezaghebbende server is, dus $P \models (CA \models\!\Rightarrow (\overset{l}{\models\!\Rightarrow} Q))$; (2) de publieke sleutel van CA kent, dus $P \models (\overset{K}{\models\!\Rightarrow} CA)$. Het protocol waarmee P de sleutel van Q te weten komt bestaat uit één bericht $(\overset{l}{\models\!\Rightarrow} Q)$, dat ondertekend door CA aan P wordt gestuurd.

1.	$P \models (CA \models\!\Rightarrow (\overset{l}{\models\!\Rightarrow} Q))$	Aanname
2.	$P \models (\overset{K}{\models\!\Rightarrow} CA)$	Aanname
3.	$CA \to P : \langle \overset{l}{\models\!\Rightarrow} Q \rangle_{K^{-1}}$	Protocol
4.	$P \triangleleft \langle \overset{l}{\models\!\Rightarrow} Q \rangle_{K^{-1}}$	$\triangleleft$-introductie, 3
5.	$P \models (CA \mid\!\sim \overset{l}{\models\!\Rightarrow} Q)$	$\mid\!\sim$-introductie, 2 en 4
6.	$P \models (\overset{l}{\models\!\Rightarrow} Q))$	$\models\!\Rightarrow$-eliminatie, 1 en 5

Door een protocol te analyseren met BAN-logica kan men erachterkomen dat bepaalde elementen ervan overbodig zijn (zoals de dubbele encryptie van tickets in stap 2 en 4 van Kerberos). Ook is het een geschikte manier om de aannamen van een protocol expliciet te krijgen. De basis van de logica is echter zwak: uit een 'correctheidsbewijs' voor een protocol mag nog niet worden geconcludeerd dat er geen aanvallen tegen dat protocol mogelijk zijn.

Kader 10.5: De UMTS-frequentieverdeling

(1)	1900 – 1920 MHz	T	Asym. Comm.
(2)	1920 – 1980 MHz	F	Gepaard met (4)
(3)	2010 – 2025 MHz	T	Asym. Comm.
(4)	2110 – 2170 MHz	F	Gepaard met (2)

Aan UMTS is in totaal 155 MHz etherbandbreedte toegewezen; dit is erg weinig in vergelijking met gsm, waar al 220 MHz aan is toegekend. Mogelijk worden later frequenties van 2500 tot 2690 MHz en de huidige gsm-banden ook voor UMTS gebruikt. Gepaarde banden (F) gebruiken frequentieverdeling tussen mobiel en vast station: er is een frequentie voor de vaste post en een andere frequentie (190 MHz hoger) voor de mobiele post. De ongepaarde banden (T) gebruiken tijdverdeling: het vaste en mobiele station zenden beurtelings op dezelfde frequentie en dit is vooral geschikt wanneer de datastromen in de twee richtingen in omvang verschillen.
De verdeling van de frequenties was een nationale verantwoordelijkheid, waarbij gekozen kon worden tussen een veiling of een *beauty contest* (schoonheidswedstrijd). Bij de tweede mogelijkheid dienen bedrijven een exploitatieplan in, waarna een commissie bepaalt aan wie de frequenties worden toegewezen. Opbrengsten van veilingen waren enorm in Duitsland (100 miljard mark) en Groot-Brittannië (22 miljard pond), maar de hooggespannen verwachtingen in Nederland en België werden niet waargemaakt. In Nederland betaalden Libertel, KPN, Dutchtone, Telfort en Ben samen vijf miljard gulden voor vijf vergunningen. In België was nauwelijks belangstelling en werden maar drie van de vier vergunningen verkocht, aan Belgacom, KPN en Mobistar, elk voor de bodemprijs van 150 miljoen euro. Toetreding van nieuwe bedrijven vond niet plaats: alle vergunningen werden verkocht aan bestaande gsm-aanbieders.
De hoge investering in vergunningen bracht KPN al in problemen van een te hoge schuldenlast. Uiteindelijk zullen de UMTS-gebruikers deze investeringen voor de aanbieders moeten terugverdienen.
Bron: `www.planet.nl/~frequentieland/`.

10.2 Universal Mobile Telecommunication System (UMTS)

De afkorting UMTS kreeg in Nederland bekendheid door de openbare frequentieveilingen in ons land en de omringende landen. Er werd echter al jaren eerder gewerkt aan de ontwikkeling van dit nieuwe communicatiesysteem, dat binnen enkele jaren de mobiele gsm-telefoons moet gaan vervangen.

In deze sectie wordt UMTS bekeken vanuit een technisch standpunt, waarbij natuurlijk de beveiliging en de daarvoor gebruikte cryptografie de meeste aandacht krijgen. Als eerste worden de ontwikkeling, mogelijkheden en vooruitzichten van UMTS geschetst (sectie 10.2.1). Vervolgens komen de architectuur van het netwerk en de beveiligingseisen aan bod (sectie 10.2.2). De beveiliging is geheel op symmetrische cryptografie gebaseerd, en het gebruikte blokalgoritme Kasumi en de ervan afgeleide functies f8 en f9 worden in sectie 10.2.3 behandeld. Hoe deze algoritmen in authenticatie- en versleutelprotocollen worden gebruikt komt in sectie 10.2.4 aan de orde.

10.2.1 Opkomst van UMTS

UMTS wordt wel omschreven als de derde generatie mobiele communicatie. Wat waren de eerste twee generaties, en waarom is er behoefte aan een derde generatie?

1. *Analoge telefonie: ATF en Greenpoint.* De eerste mobiele systemen voor een groot publiek waren de autotelefoons die werden ontwikkeld in de jaren tachtig. Het geluid werd analoog overgebracht (net als radiosignalen van de omroep) en hieraan kleven diverse nadelen. Iedereen met een geschikt radioapparaat kan rechtstreeks meeluisteren en de storingsgevoeligheid is groot. Omdat een analoog kanaal steeds (al is het tijdelijk) een eigen frequentie nodig heeft is het niet mogelijk efficiënt met het radiospectrum om te gaan.

2. *Digitale telefonie: gsm.* Omzetting van geluid in digitale signalen biedt vele voordelen. Door foutcorrectie is stooronderdrukking mogelijk; door encryptie wordt meeluisteren onmogelijk gemaakt; digitale techniek is goedkoop en betrouwbaar; integratie met andere diensten is mogelijk, zoals SMS (tekstberichten) en WAP (internetinformatie op de mobiel). Door de lage kosten (met name van apparatuur) werd mobiele telefonie een doorslaand succes: eind 2001 had ruim de helft van de Nederlanders een mobieltje.

   De technische inrichting van het gsm-netwerk is ontworpen voor spraak: communicatie vindt steeds in twee richtingen tegelijk plaats met 9,6 kbps (kilobit per seconde). Andere diensten kunnen hierop worden geïmplementeerd, maar vaak is de transmissiesnelheid te laag en soms is communicatie slechts in één richting nodig. Zo is een videoverbinding niet mogelijk, en het oversturen van bijvoorbeeld foto's is traag.

3. *Digitale communicatie: UMTS.* Het derdegeneratienetwerk benadert communicatie breder dan spraak: uitgangspunt is de mogelijkheid digitale informatie te versturen. Dit moet in pieken kunnen (bij telefonie is de datastroom constant), met snelheden tot 2000 kbps, en asymmetrisch (meer data in een van de richtingen). Op deze communicatielaag kunnen dan allerlei diensten worden aangeboden, waarbij spraak waarschijnlijk voorlopig de belangrijkste, maar zeker niet de enige zal zijn.

Rollen binnen UMTS. De organisatie van UMTS kent ruwweg drie soorten partijen (rollen): eindgebruikers, netwerkbeheerders en aanbieders. De *eindgebruikers* zijn de individuen of bedrijven die informatie willen ontvangen of verzenden, typisch met mobiele apparatuur (mobile equipment). De *netwerkbeheerders* bezitten de infrastructuur (servicing networks), zoals antennemasten, basisstations en bekabeling. Geen enkele beheerder zal een wereldwijd dekkend net gaan opbouwen, en communicatie van een eindgebruiker zal dus via wisselende en vaak meerdere netwerkbeheerders verlopen. De *aanbieders* zijn contractpartij voor de eindgebruikers, houden binnen het netwerk informatie bij over elk van hun gebruikers (home environment) en zorgen ervoor dat eindgebruikers de infrastructuur van de netwerkbeheerders kunnen gebruiken.

Vaak zal een bedrijf zowel aanbieder als netwerkbeheerder zijn: het bedrijf heeft dan zowel klanten als infrastructuur. Als de infrastructuur een grote dekking heeft kan veel verkeer van de klanten in het eigen netwerk worden afgehandeld, maar dit zal nooit helemaal lukken omdat de klanten een wereldwijde dekking eisen. De eigen eindgebruikers maken dus toch vaak gebruik van vreemde servicing networks, een verschijnsel dat *roaming* wordt genoemd.

Kader 10.6: UMTS-diensten en apparatuur

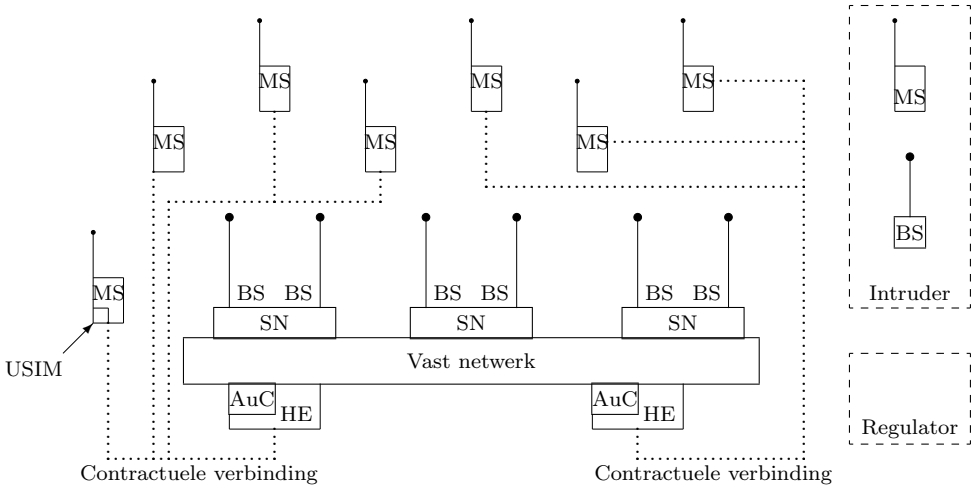
Zoals vermeld zijn de diensten van UMTS niet beperkt tot spraak: er wordt ook tekstuele informatie overgebracht, audio van hoge kwaliteit, foto's en zelfs video. De mobiele apparatuur lijkt daarom vaak niet meer op de mobiele telefoons uit het gsm-tijdperk. Vrijwel altijd is een flink kleuren-LCD-scherm ingebouwd en knoppen om over dit scherm te kunnen 'surfen'. Een simpele digitale camera (DCC-chip en lensje) maakt dan meteen beeldtelefonie mogelijk.
Een UMTS-setje kan ook een insteekkaart voor de pc of laptop zijn. Ook ziet men wel apparaten die uit meerdere delen bestaan, die onderling ook draadloos communiceren met het Bluetooth-protocol. De mobiele manager heeft dan in de ene zak een telefoontje met UMTS toegang, in de andere zak een organizer die via de UMTS-telefoon automatisch met het secretariaat communiceert. Het zakkantoor is compleet met een pen waarmee van alles kan worden opgeschreven dat dan automatisch in de organizer wordt opgeslagen of op de thuisbasis wordt meegelezen.

Cellulaire dekking. Voor de gebruiker zijn gsm en UMTS draadloze systemen: de radiografische communicatie verleent de systemen hun kenmerkende mobiliteit. De meeste communicatie verloopt echter altijd via het vaste deel van het netwerk: radiocommunicatie vindt plaats tussen het mobiele station en een nabij basisstation. De basisstations zijn onderling verbonden met vaste verbindingen: meest glasvezelkabel, maar ook satellietverbindingen en incidenteel straalverbindingen. Mobiele stations communiceren nooit rechtstreeks met elkaar.

Een netwerkbeheerder bezit een aantal basisstations en de vaste verbindingen ertussen, en heeft verbinding met de structuur van andere netwerkbeheerders. Het gebied dat door een basisstation wordt bestreken heet een *cel* en het station gebruikt een aantal frequentiekanalen om te communiceren met de mobiele stations binnen de cel. Aangrenzende cellen moeten andere frequenties gebruiken, maar de frequenties kunnen in de daarachter gelegen cellen tegelijkertijd worden gebruikt.

Ook gsm gebruikt deze cellenstructuur, maar bij UMTS zijn de cellen veel kleiner. In gsm is een cel enkele tot enkele tientallen kilometers groot, bij UMTS zijn de grootste cellen een kilometer in doorsnede en in stedelijke gebieden zijn de cellen maar tweehonderd meter groot. Omdat voor UMTS weinig frequentieruimte beschikbaar is, kan elke cel namelijk maar voor weinig stations, dus voor een klein gebied de communicatie verzorgen. Er zijn daarom voor de aanleg van het vaste UMTS-netwerk hoge investeringen vereist, maar de mobiele stations kunnen volstaan met een kleiner zendvermogen.

Het netwerk houdt van elke gebruiker de locatie bij, dat wil zeggen, de cel waarin hij zich bevindt, ook wanneer de gebruiker niet actief is. Wanneer een gebruiker zich beweegt en zich van de ene cel in de andere begeeft wordt hij administratief doorgegeven; dit heet een *hand-over*. Een hand-over kan plaatsvinden zonder dat lopende communicatie wordt onderbroken, want gedurende korte tijd (een seconde) is het mobiele station met beide basisstations in verbinding.



Figuur 10.7: Overzicht van het UMTS-netwerk.

10.2.2 Structuur en Beveiliging

Partijen in UMTS. De relatie tussen de diverse partijen en apparatuur (volgens [UMTSb]) in UMTS zijn weergegeven in figuur 10.7. (Dezelfde structuur is op gsm van toepassing.) In het vervolg wordt van Engelse terminologie gebruikgemaakt omdat deze internationaal is overeengekomen, en gestandaardiseerde Nederlandse equivalenten niet voorhanden zijn.

De *user* (eindgebruiker) heeft toegang tot het netwerk via een *Mobile Station* (MS), ook wel *User Equipment* (UE) genaamd. Het mobile station bevat een *User Services Identity Module* (USIM), een smartcard vergelijkbaar met de SIM-kaart in gsm-toestellen. De USIM bevat de *International Mobile User Identity* (IMUI) van de gebruiker, dit is het registratienummer waaronder de gebruiker bekend staat bij zijn home environment. De gebruiker kan van een ander MS gebruik maken door zijn USIM in dit toestel te plaatsen.

De user heeft een *subscription* (contractuele band) met een *Home Environment* (HE), die de administratie en afrekening van de diensten tussen netwerkbeheerders en gebruikers verzorgt. Ook fungeert de HE als trusted party voor de authenticatie tussen user en servicing network. Hiertoe beheert de HE een *Authentication Center* (AuC) in het netwerk, waarmee de servicing networks via het vaste net communiceren.

Een *Servicing Network* (SN) bestaat uit een vast deel en een collectie *Base Stations* (BS), die via de ether met de MS communiceren. Een SN laat een MS als gebruiker toe na authorisatie door de user's HE en rekent de communicatie af via de HE.

De *intruder* (aanvaller) kan uiteraard zelf over een MS beschikken (indien nodig met subscription en/of gemodificeerd), maar ook over een base station of zelfs een servicing network. In de gsm-tijd was netwerkapparatuur duur en specialistisch en de wereld van netwerkbeheerders was overzichtelijk. Nu is netwerkapparatuur goedkoop en wordt aan veel partijen verkocht in een open, dus onoverzichtelijke markt. Een aanvaller kan daarom relatief gemakkelijk aan netwerkapparatuur komen of deze bouwen, en het is ook mogelijk dat normale netwerkaanbieders aan illegale praktijken beginnen.

Onder *regulators* worden diverse overheden en opsporingsinstanties verstaan.

Aanvallen op mobiele communicatiesystemen. De beveiligingseisen van UMTS zijn opgesteld met het oog op allerlei mogelijke aanvallen die door de ontwerpers zijn voorzien [UMTSd]. Een aanval kan gericht zijn op het verkrijgen van informatie over een gebruiker (data, locatie, communicatiegedrag), het gebruik van het netwerk op kosten van een ander, of het uitsluiten van een gebruiker (denial-of-service).

Aanvalstechnieken. Met een gewijzigd MS kan de intruder een *passieve* aanval doen, dat wil zeggen, communicatie, zowel stuurinformatie als data, beluisteren. Met een gewijzigd MS kan de intruder al een *actieve* aanval uitvoeren: berichten aan het SN sturen namens een gebruiker. De intruder die een BS heeft kan zich tegenover de gebruiker als SN voordoen: de MS legt contact met het valse BS en zolang dit in de lucht is zal de MS niet met een ander BS communiceren. De intruder die over een gemodificeerd MS en een BS beschikt kan een *man-in-the-middleaanval* uitvoeren; zie kader 10.8.

Denial-of-service. Een radiografisch systeem is altijd kwetsbaar voor een *Denial-of-Service-aanval* (DoS), dat wil zeggen dat de aanvaller het de gebruiker onmogelijk maakt diensten te gebruiken. De aanvaller kan met sterke stoorpulsen alle radiografische communicatie in de omgeving van de gebruiker verstoren. Een dergelijke aanval is eenvoudig uitvoerbaar en niet met cryptografie te voorkomen; de mogelijkheid zal dus geaccepteerd moeten worden. Omdat deze aanval zo eenvoudig is, heeft het geen zin cryptografie te gebruiken om allerlei ingewikkelde DoS-aanvallen onmogelijk te maken.

Met actieve aanvallen zijn in principe diverse Denial-of-Service-aanvallen mogelijk. De intruder kan namens het MS een afmeldingsbericht sturen, waarna het MS permanent onbereikbaar is, of de intruder kan het MS contact laten maken met een vals BS. De tweede aanval is in UMTS mogelijk, de eerste wordt voorkomen door encryptie van stuurinformatie.

Overnemen van verbindingen. Oscar kan communiceren op kosten van Alice door een uitgaande verbinding over te nemen met een man-in-the-middleaanval. Als Alice een verbinding aanvraagt stuurt Oscar haar berichten door tot en met de authenticatie tegenover SN. Dan verbreekt hij de verbinding met Alice en gebruikt hij de verbinding voor zijn eigen communicatie.

Ook het overnemen van een inkomende verbinding kan voordeel opleveren. Oscar en Alice zijn op reis in een ver land. Olga is thuisgebleven en wil met Oscar bijpraten, maar liefst op kosten van Alice. Olga belt Alice; volgens de tariefstructuur moet Olga dan het binnenlands traject betalen, terwijl Alice het internationale deel van het binnenkomende gesprek betaalt. Oscar is weer 'man-in-the-middle' tussen Alice en het SN; nadat Alice het gesprek van Olga heeft aangenomen en zich heeft geauthenticeerd neemt Oscar de verbinding met Olga over.

UMTS maakt het moeilijk om onversleutelde verbindingen af te dwingen (kader 10.8), waarmee het ook moeilijk wordt verbindingen over te nemen. Op onversleutelde verbindingen wordt periodiek een hernieuwde authenticatie uitgevoerd zodat Oscar de verbinding slechts kort kan gebruiken.

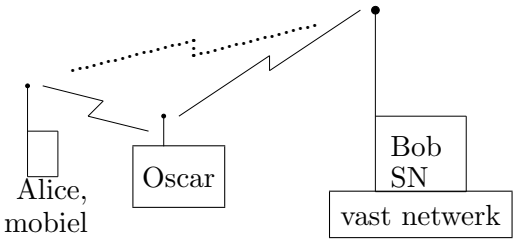
Veiligheidseisen. De belangrijkste doelen en randvoorwaarden van de UMTS-beveiliging zijn de volgende.

1. *User data confidentiality.* De inhoud van communicatie is niet uit de ethersignalen af te leiden. Hiertoe wordt, als in gsm, versleuteling op het radiokanaal toegepast.

Kader 10.8: Omzeilen van encryptie in gsm

Zowel in gsm als UMTS kan Oscar de communicatie van Alice onderscheppen met een *vals basisstation*. Wanneer Alice haar mobiel aanzet zoekt deze contact met een basisstation, en als Oscar op de oproep reageert 'vangt' hij Alice en haar mobiel zoekt niet verder. Oscar is nu een 'man-in-the-middle' tussen Alice en Bob.

Gsm kent meerdere algoritmen voor encryptie en wanneer een gesprek begint, onderhandelen mobiel en basisstation over het te gebruiken algoritme. Als Alice, die denkt rechtstreeks met Bob te communiceren, meldt welke encryptiealgoritmen zij kan gebruiken, vervangt Oscar dit bericht en vermeldt algoritmen die Bob niet kent. Het gevolg is dat er in het komende gesprek geen versleuteling wordt gebruikt, en Oscar kan het gesprek onversleuteld beluisteren.



UMTS maakt deze aanval onmogelijk door de integriteit van controleberichten te garanderen met een Message Authentication Code (MAC). Wel kan Oscar op deze manier Alice het communiceren onmogelijk maken.

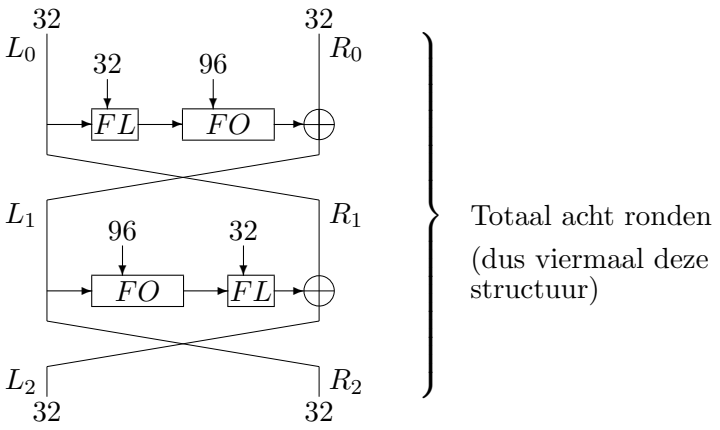
2. *User location/identity confidentiality.* De inhoud van de radiocommunicatie onthult niet welke user het kanaal gebruikt, en het is niet mogelijk de achtereenvolgende locaties (cellen) van een gebruiker uit de berichten te traceren.
   Het gebruikte mechanisme (sectie 10.2.4) is ontleend aan gsm, maar uitgebreid omdat het gsm-mechanisme niet meer voldoet onder de UMTS-aannamen.

3. *Data integrity.* Het is niet mogelijk controledata te wijzigen.
   Voor een spraaknetwerk (gsm) is de mogelijkheid, data te veranderen niet relevant, omdat een intruder onmogelijk een spraaksignaal door een vals signaal kan vervangen. Anders gezegd, het menselijk oor aan de ontvangkant is de integriteitscontrole. Omdat UMTS voor algemene datacommunicatie is ontworpen is het gebrek aan integriteitscontrole op user data opmerkelijk.

4. *User authentication and authorisation.* Gebruik van het netwerk is alleen mogelijk door geregistreerde users, en dan nog uitsluitend ten laste van hun eigen account en met instemming van de home environment.
   Omdat alle netwerkinvesteringen uiteindelijk uit communicatieopbrengsten worden terugverdiend mag dit wel als een van de belangrijkste eisen worden beschouwd. Netwerkaanbieders moeten de zekerheid hebben dat de geleverde dienst via de home environment wordt betaald. Users moeten de zekerheid hebben dat anderen niet op hun kosten het netwerk gebruiken.

5. *Network authentication.* Een MS verkrijgt de zekerheid dat het communiceert met een echt BS, dat wil zeggen, een BS dat via het vaste netwerk in verbinding staat met de user's home environment.

6. *Interoperability.* De beveiligingsmechanismen zijn gestandaardiseerd en wereldwijd worden gemeenschappelijke algoritmen ondersteund zodat er ook bij wereldwijde roaming



Figuur 10.9: De Feistel-structuur van Kasumi.

altijd beveiliging is.

7. *Flexibility.* De veiligheidsmechanismen kunnen later worden uitgebreid of aangepast wanneer nieuwe bedreigingen dit noodzakelijk maken.

Ook binnen het vaste netwerk zijn veiligheidsmechanismen nodig: tussen een home environment en servicing networks moet sprake zijn van authenticatie en versleuteling. Op deze beveiliging gaan we in dit boek niet in.

10.2.3 Cryptografische algoritmen in UMTS

Alle beveiliging binnen UMTS is gebaseerd op symmetrische cryptografie. Voor de diverse algoritmen gelden namelijk erg krappe eisen op de gebruikte rekenkracht: op een 8-bitsprocessor met 3MHz kloksnelheid mag bijvoorbeeld sleutelgeneratie maar een halve seconde duren [UMTSe]. Er worden diverse algoritmen voor sleutelgeneratie gebruikt, f0 tot f5, die echter per home environment mogen verschillen en dus niet zijn gestandaardiseerd. De algoritmen voor versleuteling (f8) en integriteit (f9) zijn wel gestandaardiseerd, en wel gebaseerd op het blokalgoritme Kasumi.

Het blokalgoritme Kasumi. Kasumi [UMTSa] is gebaseerd op een eerder algoritme, *Misty*, en werd in besloten kring ontwikkeld, maar daarna werd het ontwerp vrijgegeven. Kasumi heeft sleutels van 128 bits en versleutelt blokken van 64 bits data volgens het Feistel-principe. Het aantal ronden is acht; zie figuur 10.9.

Net als in DES wordt de invoer beschreven als twee deelrijen L_0 en R_0 van 32 bits en de uitvoer bestaat uit L_8 en R_8. Uit de sleutel K worden deelsleutels K_i (ook 128 bits) afgeleid, en per ronde is de versleuteling

$$\begin{aligned} R_i &= L_{i-1} \\ L_i &= R_{i-1} \oplus f(L_{i-1}, K_i) \end{aligned}$$

De functie f is in even en oneven ronden iets verschillend.

De rondefunctie f kent twee operaties, namelijk FL en een wat ingewikkelder FO. In FL worden 32 sleutelbits gebruikt en wordt elke helft (zestien bits) van de data geXORd met de andere datahelft en sleutelbits.

De functie FO kent weer een interne functie FI, die zestien databits verdeelt in een stuk van zeven en van negen bits, en de data wordt in vier achtereenvolgende deelronden verwerkt met S-boxen van zeven en negen bits breed. De functie FO gebruikt driemaal 32 sleutelbits (totaal 96) en heeft een interne structuur die zelf op een Feistel-netwerk lijkt maar het net niet is. De datahelften A en B (hier elk weer 16 bits) worden namelijk zo bewerkt:

$$\begin{aligned} A_i &= B_{i-1} \\ B_i &= B_{i-1} \oplus FI(A_{i-1}) \end{aligned}$$

en B_i wordt dus niet niet verkregen uit A_{i-1} maar uit B_{i-1}. (De A wordt in de functie FI gecombineerd met sleutelbits maar dat is hier niet weergegeven.) Door de verandering is de functie FO niet, zoals een Feistel-netwerk, inverteerbaar! Men bedenke echter dat de functie FO niet zelf de versleuteling van data verzorgt, maar alleen als rondefunctie binnen het grotere Feistel-netwerk wordt gebruikt. Inverteerbaarheid is dus niet noodzakelijk.

In oneven ronden wordt op L_{i-1} eerst FL toegepast en dan FO, en in even ronden wordt eerst FO toegepast en dan FL (zoals aangegeven in figuur 10.9).

Een sterke kant van Kasumi is de goede implementeerbaarheid met weinig hardware en weinig tijd. Een aanval met bijvoorbeeld differentiële cryptanalyse op Kasumi is moeilijk omdat in elke ronde alle 128 sleutelbits worden gebruikt. Daar staat tegenover dat, als eenmaal de sleutelbits van de laatste ronde bekend zijn, de aanvaller hierdoor ook de gehele sleutel kent. Volgens Kühn [Küh01] heeft een aanval op een Kasumi met slechts zes ronden al 2^{55} gekozen berichten en ongeveer 2^{100} encrypties nodig.

Stroomversleuteling: f8. Data wordt versleuteld met f8 [UMTSc], een stroomversleuteling gebaseerd op Kasumi. Data is in UMTS, net als in gsm, verdeeld in stukjes die *frames* worden genoemd en ook in UMTS wordt vanuit de encryptiesleutel steeds een rij bits gegenereerd waarmee het frame wordt geXORd. Er zijn echter ook verschillen, voortvloeiend uit het ontwerp voor telefonie, danwel algemene datacommunicatie. Bij spraak (gsm) is steeds elk frame even lang (namelijk 114 bits) en is de communicatie symmetrisch, er gaan dus in beide richtingen evenveel frames. Daarom genereert A5 (zie sectie 4.3.1) voor elk frame 228 bits, waarvan voor elke richting de helft wordt gebruikt.

Bij UMTS is de framelengte variabel en kan er communicatie in hoofdzakelijk één richting plaatsvinden, zoals bijvoorbeeld bij het downloaden van muziek of video. Daarom wordt er steeds een bitrij gegenereerd voor een frame van één richting van de communicatie; de waarde van de bit *DIRECTION* is 0 voor een frame van MS voor BS, en 1 voor een frame van BS naar MS. Verdere invoer voor de generator zijn een frame serienummer, weergegeven als *COUNT* en 32 bits lang, en de sleutel of *Confidentiality Key*, weergegeven als *CK* en 128 bits lang.

In A5 wordt de bitrij gegenereerd door drie schuifregisters die worden geïnitialiseerd met sleutel en framenummer. Een dergelijke opzet wordt in f8 weer gebruikt, de elementaire structuur van het schuifregister is echter vervangen door aanroepen van Kasumi zodat blokken van 64 bits tegelijk worden gegenereerd. Het frame kan veel langer zijn dan 64 bits (namelijk 1 tot 5114 bits) zodat per frame meestal meerdere blokken nodig zijn. Voor elk blok versleutelt f8 het vorige blok met Kasumi (onder sleutel *CK*), maar geXORd met aanvullende informatie.

De interne toestand van f8 wordt beschreven door twee rijen van 64 bits, namelijk A (die in het begin wordt berekend en niet verandert) en *KSB*, het *Key Stream Block*. De vaste bitrij KM (*Key Modifier*) van lengte 128 is 010101...0101. Bij initialisatie (dat wil zeggen,

het begin van elk frame) krijgt A de waarde van het framenummer en de richting, versleuteld onder $(CK \oplus KM)$:

$$A = Kasumi_{CK \oplus KM}(COUNT,\ DIRECTION,\ 00000).$$

(Omdat *COUNT* en *DIRECTION* samen minder dan 64 bits bevatten worden ze met nullen aangevuld tot 64 bits.)

Nu kunnen blokken van 64 bits worden gemaakt; steeds wordt het vorige blok geXORd met A en het bloknummer, en versleuteld onder CK.

$$KSB_i = Kasumi_{CK}(KSB_{i-1} \oplus (i-1) \oplus A).$$

Voor de berekening van KSB_1 wordt voor KSB_0 de waarde 0 gebruikt. Er worden zoveel blokken gemaakt tot de totale lengte minstens zo groot is als die van het frame, en van het laatste blok worden mogelijk niet alle bits gebruikt.

Is de klare tekst van het frame X, en geven we de gegenereerde bitrij (alle blokken achter elkaar) weer als KSB, dan is de cijfertekst $Y = X \oplus KSB$. De andere partij beschikt over dezelfde waarden van *COUNT*, *DIRECTION* en CK en kan ook KSB genereren en vindt X als $Y \oplus KSB$.

Berichtauthenticiteit: f9. De integriteit van data wordt bewaakt met een *message authentication code* (zie sectie 7.2.1) berekend met het algoritme f9 [UMTSc], eveneens gebaseerd op Kasumi. Kasumi wordt hier gebruikt als blokfunctie in een hashalgoritme.

Behalve de reeds bekende parameters *COUNT* en *DIRECTION* gebruikt de hashfunctie een 32-bitsgetal *FRESH* dat bij het begin van iedere connectie wordt gekozen. Dit getal voorkomt dat een tegenpartij een bericht kan vervangen door een bericht met hetzelfde serienummer uit een voorgaande connectie. De gebruikte sleutel van 128 bits heet de *integrity key* (IK).

Een reeks van 64-bitsblokken wordt gevormd door *COUNT*, *FRESH*, het bericht (1 tot 5114 bits) en *DIRECTION* achter elkaar te plaatsen en er een 1 aan toe te voegen. Met nullen wordt dit aangevuld tot een *Padded String* PS waarvan de lengte een veelvoud van 64 bits is; het i^{e} blok ervan noemen we PS_i en het aantal blokken k.

Een register A wordt steeds met een blok van PS geXORd en versleuteld onder IK, en de hashwaarde is de combinatie van alle blokken. Er geldt $A_0 = B_0 = 0$ (een reeks van 64 nullen), en

$$\begin{aligned} A_i &= Kasumi_{IK}(A_{i-1} \oplus PS_i) \\ B_i &= B_{i-1} \oplus A_i \end{aligned}$$

De hashwaarde is de linkerhelft (32 bits slechts) van $Kasumi_{IK \oplus KM}(B_k)$ (hier is KM de string 1010..10).

10.2.4 Cryptografische protocollen in UMTS

UMTS bevat een heel scala aan beveiligingsalgoritmen en -protocollen. We behandelen hier slechts een greep eruit, namelijk die voor bescherming van de user identity, de authenticatieprocedure, en de manier waarop sleutels worden gebruikt tijdens een contact tussen MS en SN.

Subscriber identity confidentiality. Elk MS wordt in UMTS geïdentificeerd met een *International Mobile Equipment Identifier* (IMEI), in feite een pseudoniem voor het telefoonnummer van dat station. Als dit nummer zou worden gebruikt om de communicatie tussen een MS en een BS te identificeren, zou een intruder de locatie en het communicatiegedag van een gebruiker via de ethersignalen kunnen volgen. Het is niet zo eenvoudig om het IMEI in alle communicatie versleuteld te versturen, omdat er immers voor versleuteling eerst authenticatie en sleutelgeneratie nodig is.

Daarom krijgt in UMTS elk MS tevens een tijdelijke identiteit, de *Temporary Mobile Equipment Identifier* (TMEI), die steeds kan veranderen en dat in ieder geval bij een hand-over doet. De TMEI wordt steeds toegekend via een cel, en identificeert het station uniek in combinatie met het nummer van die cel, de *Location Area Identifier* (LAI). Radioberichten worden geïdentificeerd met de TMEI/LAI combinatie en de IMEI wordt nooit over de ether verstuurd. MS en BS houden de correspondentie tussen IMEI en TMEI/LAI bij.

De normale procedure voor een hand-over van basisstation BS1 naar BS2 is als volgt.

1. Het MS meldt zich bij het nieuwe basisstation BS2 met vermelding van de oude TMEI/LAI.
2. De LAI identificeert voor BS2 de cel waarbinnen de TMEI is uitgegeven. BS2 neemt via het vaste netwerk contact op met BS1 en vraagt het IMEI van het MS.
3. Als BS2 de identiteit van het MS kent, vindt authenticatie plaats en wordt er een encryptiesleutel afgesproken.
4. BS2 kent het MS een nieuwe TMEI toe en stuurt deze, met de LAI van BS2, versleuteld naar MS.

Het MS bewaart de TMEI/LAI ook wanneer het wordt uitgezet, zodat altijd na het aanzetten de normale procedure kan worden gevolgd. Hierdoor kunnen BS1 en BS2 ook willekeurig ver uit elkaar liggen en tot verschillende SN behoren.

Merk op dat het IMEI inderdaad niet wordt verzonden bij een hand-over. Een intruder die een bepaald MS wil volgen kan dus uit de communicatie niet afleiden welke gegevens voor dit station bestemd zijn of ervandaan komen. Een intruder die in een cel communicatie van een MS oppikt en die wil gaan volgen loopt vast bij de volgende hand-over. Hij kent immers de TMEI in de eerste cel (maar weet niet welk toestel hierbij hoort) en kan nog horen dat het MS zich meldt bij een volgende cel. Dan krijgt het toestel een nieuwe TMEI, maar de intruder weet niet welke dat is en verliest dus het contact zodra de nieuwe TMEI in de nieuwe cel wordt gebruikt.

De normale procedure faalt als BS1 onbereikbaar is (of er zijn gegevens over de IMEI verloren gegaan) en wanneer het MS voor het eerst in gebruik wordt genomen. In dat geval kan BS2 aan MS vragen de IMEI zelf op te sturen. In gsm gebeurt dit onversleuteld. Een passieve aanvaller kan in zo'n geval de IMEI rechtstreeks ontvangen, maar omdat het weinig gebeurt wordt het veiligheidsrisico ervan acceptabel geacht. Met een actieve aanval (vals BS) zit het anders. De intruder laat het MS zich melden bij een vals station en genereert dan de melding dat de IMEI niet meer van het oude BS te krijgen is, om zo de IMEI onversleuteld te horen. Bij het ontwerp van gsm werd met valse BS geen rekening gehouden, maar binnen UMTS is deze procedure hierdoor onacceptabel.

Het MS kent niet alleen IMEI, maar beschikt ook over de versleuteling Y ervan onder een sleutel KI. Terwijl het IMEI uniek is, wordt de KI door een grote groep mobile stations gedeeld. Het MS stuurt aan BS de waarde Y en de identificatie van de groep, dus: welke sleutel KI is gebruikt. Het BS kan via het vaste netwerk KI opvragen en IMEI uitrekenen

door decryptie, een vals BS kan niet aan KI komen. De keuze van de groepsgrootte is een compromis. Zijn de groepen klein, dan geeft de identificatie van de groep te veel informatie over het MS. Zijn de groepen groot, dan wordt de sleutel KI veel gebruikt en is het risico van uitlekken groter.

Authentication and Key Agreement (AKA). Wanneer een MS contact maakt met een BS wordt een tweezijdige authenticatieprocedure toegepast. Het BS verkrijgt hiermee zekerheid over de identiteit van het MS (user authentication), en het MS verkrijgt zekerheid te communiceren met een echt BS (network authentication). Om het overnemen van verbindingen na authenticatie te voorkomen worden ook sleutels afgesproken voor versleuteling (CK) en voor een message authentication code (IK). De versleuteling dient uiteraard ook de vertrouwelijkheid van de gebruikersdata.

Uitgangspunt van de procedure is dat de USIM in het MS en het AuC (Authentication Center) een geheim getal K (128 bits) delen. Hiermee worden uit een random getal $RAND$ (eveneens 128 bits) vier waarden berekend.

Waarde	bits	functie	gebruik
$MACA$	64b	$f1_K(SQN, RAND)$	network authentication
RES	32–128b	$f2_K(RAND)$	user authentication
CK	128b	$f3_K(RAND)$	confidentiality key (versleuteling)
IK	128b	$f4_K(RAND)$	integrity key (bericht authenticiteit)

Wanneer een MS zich via een BS meldt bij een SN worden de volgende stappen uitgevoerd.

1. SN $\longrightarrow$ AuC: $\langle IMEI \rangle$.

   Op ontvangst van het $IMEI$ zoekt AuC de bijbehorende K en SQN, het aantal malen dat reeds authenticatie van deze gebruiker heeft plaatsgevonden. Dit wordt uiteraard elke keer verhoogd, zodat elke keer een andere $MACA$ voor user authentication wordt gebruikt. Het AuC berekent nu:

   $$\begin{aligned} MACA &= f1_K(SQN, RAND) \\ XRES &= f2_K(RAND) \\ CK &= f3_K(RAND) \\ IK &= f4_K(RAND) \end{aligned}$$

   De gebruikte functies $f1$ to $f4$ moeten in de USIM en het AuC gelijk zijn, maar overeenstemming tussen verschillende home environments is niet nodig en daarom zijn ze niet in de standaard vastgelegd. Hier staat $XRES$ voor eXpected RESponse, namelijk het antwoord dat SN straks verwacht van MS.

2. AuC $\longrightarrow$ SN: $\langle RAND, XRES, CK, IK, SQN, MACA \rangle$.

   Dit bericht wordt via het vaste netwerk verstuurd.

3. SN $\longrightarrow$ MS: $\langle RAND, SQN, MACA \rangle$.

   Dit bericht wordt via de ether verstuurd. Een man-in-the-middle verkrijgt dus de waarde $MACA$ waarmee het netwerk zich tegenover MS authentificeert, maar niet de bijbehorende sleutels CK en IK.

   De USIM beschikt over K en kan dus uit $RAND$ (en SQN) de authenticatiecodes en sleutels berekenen:

   $$\begin{aligned} XMACA &= f1_K(SQN, RAND) \\ RES &= f2_K(RAND) \end{aligned}$$

De USIM verifieert dat $MACA = XMACA$, waarmee het netwerk is geauthenticeerd. De USIM geeft CK en IK door naar het MS, waar de versleuteling wordt uitgevoerd.

4. MS $\longrightarrow$ SN: $\langle RES \rangle$.
   SN verifieert $XRES = RES$, waarmee MS is geauthenticeerd. Na de wederzijdse authenticatie worden CK en IK als sleutels voor encryptie en bericht-authenticiteit gebruikt.

Versleuteling en integriteit. Naast f8 en f9 kunnen nog andere algoritmen voor versleuteling en integriteit worden gedefinieerd. Het MS stuurt een bericht aan SN met welke algoritmen het ondersteunt. Als het MS en het SN geen integriteitalgoritme gemeenschappelijk hebben wordt de verbinding verbroken; in dit geval zou er namelijk geen bescherming van controleberichten zijn en dit leidt tot een te groot risico. Als het MS en het SN geen versleutelingsalgoritme gemeenschappelijk hebben, kan er worden gekozen voor een onversleutelde verbinding.

Versleuteling en integriteitsbewaking werken op het hele traject tussen MS en de centrale van het SN, dus niet alleen op de radioverbinding maar ook op het vaste deel van het servicing network. Tussen de verschillende servicing networks wordt geen versleuteling toegepast.

Voor de integriteitsbewaking berekent de zender van een bericht een authenticatiecode met f9 en stuurt deze mee. De ontvanger berekent ook de code en vergelijkt deze met de meegezonden code. Deze bewaking vindt plaats voor controleberichten, maar *niet* voor applicatiedata. Deze data wordt geacht door de versleuteling voldoende beschermd te zijn (zie opgave 10.4).

10.3 Secure Socket Layer

Het World Wide Web werd ontworpen voor het opvragen van allerlei soorten informatie, zoals wetenschappelijke publicaties, maar ook weerberichten, bioscoopprogramma's, treinschema's et cetera. De behoefte aan beveiliging van dit soort informatie is gering. Echter, na enkele jaren werd het WWW ook gebruikt als platform voor elektronische handel: tegenwoordig worden er aanbiedingen (offertes) gedaan, transacties afgesloten en betalingsopdrachten verstuurd, en er wordt voor informatie betaald. Verschillende incidenten uit de eerste jaren hebben de noodzaak van goede beveiliging duidelijk gemaakt. In meerdere gevallen werden creditcardnummers afgeluisterd en later misbruikt. Ook werden connecties tussen banken en hun klanten overgenomen, waarna de gebruikers hun betaalopdrachten en PINs onbewust naar de indringers verzonden.

Netscape, belangrijke aanbieder van de gelijknamige internetbrowser, ontwikkelde voor de beveiliging het Secure Socket Layer (SSL) protocol. SSL voorziet in authenticatie, integriteit en versleuteling en gebruikt een combinatie van symmetrische en asymmetrische cryptografie.

1. *Versleuteling:* Voor de versleuteling van (grote hoeveelheden) data is symmetrische encryptie geschikt, mits op een veilige manier een sleutel kan worden overeengekomen; dat gebeurt in SSL bij het opzetten van een sessie.

2. *Authenticatie:* Zoals blijkt uit de Kerberos- en UMTS-protocollen, is ook authenticatie met symmetrische cryptografie mogelijk, maar interactie met een trusted party tijdens de authenticatieprocedure is dan vereist. Omdat dit bij het bekijken van webpagina's niet wenselijk is, bevat SSL authenticatieprotocollen op basis van public-keyalgoritmen.

3. *Integriteit:* SSL genereert geen digitale handtekening bij elk bericht, maar gebruikt een Message Authentication Code (MAC) die berekend is uit een gedeelde informatie (zie sectie 7.2.1). Deze codes voorzien wel in integriteit van de berichten, maar niet in onloochenbaarheid. Een client heeft dus niet achteraf de mogelijkheid, tegenover derden te bewijzen wat de inhoud van bijvoorbeeld een pagina met aanbiedingen was. Onloochenbaarheid moet dus op het applicatieniveau worden verkregen, en dit is meestal ook wenselijk.

De eigenlijke communicatie tussen de twee partijen gebeurt met symmetrische cryptografie door middel van het *recordprotocol*. De gebruikte symmetrische sleutels worden in het voorafgaande *handshakeprotocol* op veilige wijze tussen de partijen overeengekomen met asymmetrische cryptografie. Uiteraard moeten de partijen daarvoor over elkaars publieke sleutel beschikken, en hiervoor worden certificaten in het standaard formaat X.509 gebruikt.

Het handshakeprotocol. Communicatie tussen client en server vindt steeds plaats op initiatief van de client; ruwweg zijn drie stappen in het protocol aan te wijzen.

1. De client begint het handshakeprotocol door het sturen van een *client_hello* bericht met daarin een random getal en informatie over welke algoritmen en protocollen door de client worden ondersteund.

2. De server maakt nu een keuze over de te gebruiken algoritmen en meldt deze in een *server_hello* bericht, waarin eveneens een random getal is opgenomen. De random getallen voorkomen dat verschillende incarnaties van het protocol interacteren. Verder stuurt de server, als er authenticatie nodig is, zijn public-keycertificaat en kan hij, als hij geen anonieme clients wil toestaan, de client om een certificaat vragen.

3. De client stuurt, als hierom gevraagd is, zijn certificaat en hierna wordt een sleutel gegenereerd.

   Als hiervoor Diffie-Hellman wordt gebruikt, vinden client en server de publieke getallen van de tegenpartij in het ontvangen certificaat, en rekenen de sleutel M uit met de eigen geheime parameter (zie sectie 4.4.1). De andere partij is nu geauthenticeerd: verdere communicatie wordt immers symmetrisch beveiligd, en de tegenpartij die M kent moet ook de partij zijn die de geheime parameters van het ontvangen certificaat kent.

   Wanneer RSA is overeengekomen voor sleutelgeneratie, kiest de client M en stuurt die op, versleuteld onder de publieke sleutel van de server. De server heeft dan echter wel een certificaat ontvangen, maar heeft geen zekerheid dat hij communiceert met de partij die de bijbehorende geheime sleutel kent. Daarom stuurt de client zijn handtekening over de eerdere berichten en M.

De sleutel M wordt niet direct als sleutel in het recordprotocol gebruikt: de benodigde sleutels worden met hashfuncties afgeleid uit M en de random waarden in de *hello* berichten.

Het recordprotocol. Is eenmaal de veilige verbinding gelegd, dan worden berichten uitgewisseld met symmetrische cryptografie. Elk bericht wordt voorzien van een message authentication code (voor integriteit) en versleuteld (voor vertrouwelijkheid). Samen met de lengte en een aanduiding van de inhoud wordt het pakket aangeboden aan het standaard transportprotocol TCP.

Tijdens het handshakeprotocol of het recordprotocol kunnen allerlei uitzonderingen optreden, die leiden tot het afbreken van de verbinding. Dit gebeurt bijvoorbeeld als een onverwacht bericht werd ontvangen, of een bericht met incorrecte MAC.

X.509 Certificaten. Een certificaat is een ondertekende verklaring van de binding tussen een persoon (of instantie) en zijn publieke sleutel. De vorm van deze certificaten werd gestandaardiseerd door de International Telecommunications Union in aanbeveling X.509.

Structuur van de certificaten. Certificaten worden uitgegeven door een Certification Authority (CA) nadat deze de binding tussen persoon en sleutel heeft geverifieerd. Een X.509-certificaat bevat (onder andere) de volgende informatie:

- **Serienummer:** Een unieke identificatie van dit certificaat.
- **Uitgever:** De naam van de CA.
- **Geldigheid:** De periode dat het certificaat mag worden gebruikt.
- **Houder:** De naam van de gebruiker die in bezit is van de geheime sleutel die hoort bij de publieke sleutel in het certificaat.
- **Public key:** De sleutel waarover het certificaat is afgegeven.
- **Handtekening:** De digitale handtekening van de CA over alle voorgaande velden.

Een gebruiker kan certificaten van een of meerdere CA's accepteren; de lijst van CA's wordt meegestuurd als in SSL om een certificaat wordt gevraagd. Hij moet dan wel de publieke sleutels van die CA's hebben (om de handtekeningen te controleren) en inzicht hebben in hoe de controle bij elke CA verloopt. In Netscape (SSL) kan per CA worden aangegeven of een certificaat van die CA wordt geaccepteerd van een website, of voor e-mail.

Vreemde CA's en kruiscertificatie. Wanneer een partij wel over een certificaat beschikt, maar dat niet werd uitgegeven door een CA die voor de ander acceptabel is, kunnen meerdere certificaten worden gestuurd. Alice heeft bijvoorbeeld een certificaat van CA1 terwijl Bob certificaten van CA2 accepteert. Nu kunnen de CA's onderling aan *kruiscertificatie* doen: hierbij geven de CA's certificaten voor elkaars public key, dus CA2 geeft aan CA1 een certificaat af. Dit certificaat is door CA1 ook aan Alice ter beschikking gesteld, en Alice stuurt het met haar eigen certificaat naar Bob. Bob verkrijgt door het tweede certificaat de zekerheid dat CA1 inderdaad een certificate authority is en tevens zekerheid over de public key van CA1, waarna hij Alice' certificaat kan accepteren.

Intrekking (revocation). De CA's kunnen hulp bieden ingeval een gebruiker merkt dat zijn geheime sleutel is uitgelekt (gecompromitteerd). Het certificaat wordt dan ingetrokken, wat wil zeggen dat de CA het niet meer ondersteunt; uiteraard kunnen kopieën ervan nog op diverse plaatsen circuleren. De CA publiceert regelmatig een lijst van uitgegeven certificaten die zijn ingetrokken, de zogenaamde *revocation list*. Partijen zijn niet verplicht om bij het aannemen van een certificaat na te vragen of het is ingetrokken, maar bij erg belangrijke interacties kan het worden gedaan.

De certificate authorities, de certificaten, het proces van kruiscertificatie en de revocation lists vormen samen een goede basis voor het veilig werken met public-keycryptografie. Deze gezamenlijke structuur wordt aangeduid als de *public-keyinfrastructuur*.

Samenvatting en conclusies

In dit hoofdstuk keken we naar het authenticatiesysteem Kerberos, het mobiele communicatiesysteem UMTS, en het SSL-protocol (Secure Socket Layer) voor de beveiliging van internetsites. De meeste grootschalige cryptografische systemen en protocollen zijn geheel of voor het grootste deel gebaseerd op symmetrische cryptografie. In de mobiele telefonie, maar ook in drukbezochte internetservers, gelden hoge eisen aan de snelheid waarmee cryptografische functies berekend kunnen worden. Alleen symmetrische cryptografie kan op het moment aan deze eis voldoen, en de huidige algoritmen mogen door uitgebreid onderzoek en goeddoordachte ontwerpen als veilig worden beschouwd.

Bij de behandelde systemen komen ook de tekortkomingen van symmetrische cryptografie aan het licht.

1. *Authenticatie versus identificatie (sectie 8.1).* De sleutel waarmee een gebruiker zich authentiseert is ook op een andere plaats opgeslagen, waarmee misbruik in principe mogelijk is. De gebruiker moet deze centrale partij vertrouwen. Bij identificatie, alleen mogelijk met asymmetrische cryptografie, is de sleutel van een gebruiker niet opgeslagen en is geen misbruik mogelijk.

2. *Interactie met third party.* Authenticatieprotocollen op basis van symmetrische cryptografie vereisen dat tijdens de authenticatie contact wordt gelegd met een trusted party. Dit is de Kerberos-server of, in UMTS, de Home Environment van de gebruiker. Deze interactie maakt de authenticatie duurder (langer) en betekent dat twee partijen niet met elkaar kunnen communiceren als deze derde partij niet bereikbaar is. Secure Socket Layer gebruikt in de authenticatiefase public-keycryptografie en -certificaten. De certificaten worden wel door een derde partij afgegeven, maar contact met die partij tijdens de procedure is niet nodig.

3. *Integriteit versus onloochenbaarheid.* Message Authentication Codes laten de ontvanger verifiëren dat het bericht inderdaad van de juiste tegenpartij afkomstig is, maar dit is niet achteraf bewijsbaar tegenover een onafhankelijke partij. Een geschil tussen de partijen over de inhoud van de communicatie is dus niet gemakkelijk op te lossen. Omdat Secure Socket Layer claimt ook e-commerce over het Internet te faciliteren, is het gemis aan onloochenbaarheid bij SSL een minpunt.

Doordat goedkope hardware steeds sneller wordt, is in de toekomst een toename van het gebruik van asymmetrische cryptografie te verwachten. Ook wordt onderzoek gedaan naar asymmetrische functies die sneller te berekenen zijn dan de nu gebruikte, bijvoorbeeld op basis van elliptische krommen (sectie 15.2.2).

Opgaven bij hoofdstuk 10

Opgave 10.1 *Neem aan dat Oscar erin is geslaagd de Kerberos-server te corrumperen (en dus, de berichten te laten verzenden die hij kiest).*

1. *Beschrijf hoe Oscar toegang kan krijgen tot Bob op Alice' naam (aanval op client authentication).*

2. *Beschrijf hoe Oscar kan zorgen dat Alice contact met Oscar krijgt in plaats van met Bob (aanval op server authentication).*

3. *Beschrijf hoe Oscar Alice en Bob kan laten communiceren maar zelf alle berichten kan inzien (aanval op data confidentiality).*

Opgave 10.2 *Welke aanval is mogelijk op de authenticatie in UMTS wanneer de parameter SQN in $f1$ ontbreekt?*

Opgave 10.3 *Laat zien dat f9* niet one-way *is. Gegeven COUNT, FRESH, een bitrij M en een 32-bitsstring H_0, construeer een bericht dat bestaat uit M aangevuld met enkele tientallen random bits, dat door f9 op H_0 wordt afgebeeld.*
Met hoeveel random bits wordt M ongeveer aangevuld? Is deze constructie als aanval uit te buiten?

Opgave 10.4 *Stroomversleuteling is een erg zwak mechanisme voor integriteit. Laat zien hoe Oscar een versleuteld, maar bekend bericht of deel van een bericht kan veranderen in een door hem gekozen bericht (of deel daarvan) met dezelfde lengte.*

Hoofdstuk 11

Regulering van cryptografie

Men zou verwachten dat individuen vrij zijn in het uitwisselen van informatie en vrij in het kiezen van de manier van uitwisseling omdat dit een zaak tussen deze partijen is. Dit is echter niet het geval: in vrijwel alle landen houdt de overheid met wetgeving greep op de uitwisseling van informatie. Het aftappen van telefoon- en communicatielijnen is voor opsporingsdoelen altijd toegestaan, en soms maakt de overheid het zichzelf hierbij gemakkelijker door versleuteling te beperken. Daarnaast kennen veel landen beperkingen op de handel in en export van cryptografische producten.

In dit hoofdstuk worden de belangrijkste wetten op dit gebied in Nederland en België genoemd, maar dat is niet het voornaamste doel van het hoofdstuk. Een overzicht hiervan is in een boek bijna niet te geven omdat de wetgeving aan snelle verandering onderhevig is. Een bron op dit gebied is het werk van Koops [Koo01], waaraan alle gegevens over wetgeving in dit hoofdstuk zijn ontleend. Ook bevat de wetgeving vaak veel onduidelijkheden omdat de achterliggende techniek door de wetgevers onvolledig of te laat wordt begrepen. Belangrijker in dit hoofdstuk is het, inzicht te krijgen in de belangen van de diverse partijen die bij de regelgeving zijn betrokken. Verder worden de argumenten voor of tegen bepaalde wettelijke regelingen besproken.

Het uitgebreidst en meest omstreden is de regelgeving die geldt rond versleuteling (sectie 11.3). Ook de restricties die gelden op export (sectie 11.2) zijn ingegeven door zorgen over gebruik (door zogenaamde 'schurkenstaten') van cryptografie voor versleuteling. Het gebruik van cryptografie voor identificatie en authenticatie is nergens verboden. Het belang van digitale handtekeningen voor e-commerce wordt nu wel in veel landen ingezien, en hierover ontstaan nu ook wettelijke regels (sectie 11.4.1).

Sectie 11.1 belicht eerst de partijen die bij de totstandkoming van regelgeving zijn betrokken en de randvoorwaarden die daarbij gelden.

11.1 Totstandkoming van wetgeving

In een parlementaire democratie worden wetten aangenomen door de volksvertegenwoordiging, het parlement. De wetten worden voorbereid door ambtenaren onder verantwoordelijkheid van ministers. Enige tijd nadat de wetten door het parlement zijn goedgekeurd, treden ze in werking.

De overheid is in het maken van wetgeving beperkt door de Grondwet en door internationale afspraken en verdragen. Verder moet de overheid natuurlijk rekening houden met de belangen van de burgers (in binnen- en buitenland), de bedrijven en de eigen organisaties en diensten.

De Grondwet. Vrijwel elk land heeft een *grondwet* waarin de verhouding tussen de burgers en de overheid wordt geregeld. In de Grondwet worden de randvoorwaarden gesteld waarbinnen de overheid wetten kan maken. In het bijzonder vermeldt de Grondwet steeds bepaalde *grondrechten*, die door de wetgevende overheid gerespecteerd moeten worden. In Nederland en België omvatten deze grondrechten onder andere het recht op *drukpersvrijheid*, *privacy* en *briefgeheim*.

In Nederland noch in België kan wetgeving door de rechter aan de Grondwet worden getoetst; wetten die in strijd zijn met de Grondwet kunnen dus niet eenvoudig buiten werking worden gesteld.

Internationale verdragen en afspraken. Ondertekening van internationale verdragen kan een regering verplichten of verbieden bepaalde dingen bij wet te regelen. De regelingen in zo'n internationaal verdrag zijn doorgaans niet direct in elk land geldig, maar pas als de nationale wetgeving eraan is aangepast. Voorbeelden van zulke verdragen zijn de *Verklaring van de Rechten van de Mens*, het *Akkoord van Wassenaar* (sectie 11.2), of het *Verdrag van Maastricht* over de vorming van de Europese Unie. In Nederland en België kan een rechter de nationale wetten toetsen aan internationale verdragen, en wetten buiten werking verklaren als ze ermee in strijd zijn.

De Europese Unie stuurt tegenwoordig sterk de regelgeving in de lidstaten. Een belangrijk doel van de Europese Unie is, de diverse nationale wetgevingen met elkaar in overeenstemming te brengen. Dit doet de Unie door het uitgeven van *verordeningen* en *richtlijnen*. Een verordening geldt als een soort Europese wet en is direct geldig voor burgers en bedrijven in de Unie. Verordeningen worden meest uitgegeven op het gebied van economie en handel (bijvoorbeeld eisen op in de handel gebrachte producten), en een individu heeft met de meeste niet rechtstreeks te maken. Een richtlijn is een verplichting voor de nationele overheden om de nationale wetgeving aan te passen. Een richtlijn is niet zelf geldig voor partijen binnen de Unie, men hoeft zich er pas aan te houden als de richtlijn inderdaad door de nationale overheid in wetgeving is geïmplementeerd.

Een ander internationaal verdrag is het *Europees Verdrag voor de Rechten van de Mens* (1950). Inhoudelijk is er veel overlap met de *Universele Verklaring van de Rechten van de Mens* (Verenigde Naties, 1948), maar het Europese verdrag verschilt hiervan in die zin, dat het bindend is voor de landen die het ondertekend hebben. Het verdrag spreekt over vrijheid van meningsuiting, en politieke en godsdienstige vrijheid.

Partijen en hun belangen. De burgers die in een land wonen willen zo weinig mogelijk in hun doen en laten door de overheid worden beperkt. Maar zij verwachten van de overheid ook bescherming tegen (kwaadwillende) andere partijen, en deze belangen kunnen wel eens botsen.

Bedrijven zijn er doorgaans op uit (veel) geld te verdienen en zijn dus gebaat bij een klimaat waarin er zo weinig mogelijk regels zijn die economische activiteiten aan banden leggen. Wel vragen de bedrijven om een mate van rechtszekerheid, dat wil zeggen, een infrastructuur waarin bijvoorbeeld partijen worden gedwongen zich aan onderlinge afspraken te houden, en waarin wordt opgetreden tegen concurrerende bedrijven die regels overtreden.

Verder zijn er de opsporings- en veiligheidsdiensten. Dit zijn publieke organen (dat wil zeggen, staan onder overheidscontrole) die als doel hebben criminaliteit te bestrijden en de

veiligheid van het land te waarborgen. Voor veiligheidsdiensten is het van belang, op de hoogte te zijn van het doen en laten van vijandige partijen (zoals terroristen en vijandelijke landen). Voor opsporingsinstanties komt hier nog bij, dat er bewijzen hierover moeten worden verzameld. Deze belangen van informatievergaring zijn vaak strijdig met het belang van de burger in een goede bescherming van zijn privacy.

11.2 Verkoop en export van cryptografie

Tot het werk van inlichtingen- en veiligheidsdiensten behoort het, op de hoogte te blijven van wat er in andere landen gebeurt. Een van de manieren om dit te doen, is het berichtenverkeer te beluisteren, bijvoorbeeld tussen ambassades of ministeries van de andere landen. Het spionagewerk wordt echter moeilijker wanneer de tegenpartij van versleuteling gebruik blijkt te maken. Het breken van versleutelingsalgoritmen (*cryptanalyse*) hoort dan ook tot de klassieke taken van inlichtingendiensten, en de National Security Agency in de Verenigde Staten heeft cryptanalyse als een van zijn belangrijkste taken.

De frustratie loopt hoog op wanneer de vijandelijke berichten onleesbaar blijken te zijn omdat ze zijn versleuteld met algoritmen die in het eigen land zijn ontwikkeld. Hieruit komt de wens voort, de export van cryptografische technologie aan banden te leggen, met name naar landen waarmee men op gespannen voet staat. Aan zulke landen verkoopt men het liefst alleen cryptografische producten die de eigen geheime dienst kan kraken. Vooral vanuit de Verenigde Staten is er de wens export van cryptografie naar 'schurkenstaten' aan banden te leggen, en de VS ijveren ook voor exportrestricties in andere landen.

Deze redenering is begrijpelijk, maar of ze ook in de spionagepraktijk van alledag stand houdt is de vraag. De verspreiding van wetenschappelijke kennis over cryptografie kan niet worden beperkt. Als een vijandig land geen kant en klare cryptosoftware kan kopen, kan het die altijd zelf ontwikkelen. Het nieuwe systeem is in ieder geval voor de eigen inlichtingendiensten onbekend. Men kan een geheel eigen algoritme ontwikkelen, of zelf de bekende algoritmen implementeren, eventueel met meer ronden of een grotere sleutellengte. Of het gemakkelijker te kraken is, is nog maar de vraag. Bovendien zullen buitenlandse partijen (niet alleen staten, maar ook internationale criminele organisaties) zich van een exportverbod weinig aantrekken en gebruik kunnen maken van illegaal geëxporteerde software.

Tegenwoordig is cryptografie niet meer alleen een zaak van militairen en diplomaten, maar zij wordt in allerlei producten gebruikt: besturingssystemen, netwerkapparatuur, telefoons. Deze producten hebben geen militaire, maar een civiele toepassing. De producenten ervan ervaren bij de verkoop grote problemen door een exportverbod. Rivest [Riv98] schat dat de Amerikaanse softwarebedrijven per jaar miljarden dollars aan omzet mislopen door exportbeperkingen. Door intensieve lobby vanuit het bedrijfsleven zijn de exportbeperkingen de laatste jaren minder streng geworden.

Akkoord van Wassenaar. In de tijd van de Koude Oorlog zag het *COCOM* (COordinating COMmittee for Multilateral Export Controls) toe op de export van militaire en dual-useproducten. (Dual-usegoederen zijn producten die zowel een militaire als een civiele toepassing hebben.) Belangrijkste bedoeling was te voorkomen dat deze producten ruim beschikbaar zou komen in 'gevaarlijke' landen als Libië en Irak. COCOM werd in 1994 ontbonden en werd opgevolgd door het Akkoord van Wassenaar.

Het Akkoord van Wassenaar werd in 1995 door 28 landen getekend, waaronder Nederland

en België. Het is niet direct rechtsgeldig voor partijen in de ondertekenende landen, maar pas nadat de ondertekenaars het in nationale wetgeving hebben geïmplementeerd. De export van wapens en dual-usegoederen werd gebonden aan vergunningen, die door de nationale overheden worden uitgegeven. Cryptografie wordt in het akkoord expliciet als dual-useproduct genoemd, maar cryptografie voor de consumentenmarkt was uitgezonderd: die mocht dus vrij worden geëxporteerd.

In 1998 werd het verdrag herzien: export van symmetrische cryptografie met sleutels tot 56 bits en asymmetrische cryptografie met sleutels tot 512 bits was vrij. Ook werd een uitzondering gemaakt voor cryptografie die de toegang tot auteursrechtelijk beschermde werken beveiligt; hierbij gaat het om kopieerbeveiligingen op bijvoorbeeld cd's en dvd's, die dus zonder vergunning mogen worden geëxporteerd.

Export uit Nederland en België. In Nederland en België zijn de restricties van het Akkoord van Wassenaar van kracht. Voor export binnen de Benelux is echter geen vergunning vereist. Het is onduidelijk of ook export door elektronische middelen (bijvoorbeeld het aanbieden van software als download via WWW) onder het verbod valt.

Gevolgen van exportrestricties. In enkele gevallen kunnen gebruiks- en exportrestricties leiden tot een slechtere cryptografische beveiliging, ook in landen waarvoor de restricties helemaal niet gelden. Dit is uiteraard een onbedoeld neveneffect van de regelgeving.

Communicatie in gsm-telefoons, bijvoorbeeld, wordt versleuteld met het algoritme A5, waarvan diverse versies in gebruik zijn omdat de sterkste versie, A5/1, niet overal is toegestaan. Bij elke verbinding moet daarom door de partijen informatie worden uitgewisseld over welke versie wordt ondersteund, en dit creëert een zwakte in de beveiliging. Een aanvaller kan deze berichtuitwisseling verstoren en daarmee afdwingen dat er in het geheel geen versleuteling wordt gebruikt (zie kader 10.8). Een van de eisen [UMTSe, blz. 8] op het versleutelingsalgoritme in UMTS is, dat minstens één versie ervan wereldwijd exporteerbaar is, zodat kan worden gegarandeerd dat altijd versleuteling wordt toegepast.

11.3 Regels rond geheimhouding

Overheidsdiensten als de politie en veiligheidsdiensten hebben belang bij het verkrijgen van informatie en worden dus door versleuteling belemmerd als die wordt toegepast. Aan de andere kant hebben bedrijven belang bij veilige communicatie met leveranciers en afnemers. Volgens de Grondwet is privacy een grondrecht, en dit houdt in dat burgers zelf kunnen bepalen hoeveel informatie over hen aan anderen ter beschikking komt.

In deze sectie wordt eerst gekeken naar de mogelijkheden en het gebruik van aftappen van informatie, en daarna naar de wettelijke regels rond cryptografische bescherming daartegen.

11.3.1 Aftappen van telecommunicatie

Wie tapt er af? Het is erg gemakkelijk, toegang te krijgen tot informatie die wordt verzonden over telecommunicatienetwerken. De beheerders van de netwerken hebben direct toegang tot alle informatie die wordt verstuurd via hun schakelstations. Door de gebruikte routeringstechniek in het internet kan informatie door heel veel stations lopen, ook in andere landen,

en welke het zijn is erg onvoorspelbaar. Het kan bijvoorbeeld gebeuren dat e-mailverkeer tussen Europese landen 's morgens via Amerika wordt verstuurd omdat het daar nacht is en de netwerken dan licht zijn belast.

Veel communicatie vindt draadloos plaats, en radiosignalen zijn door hun aard zeer gemakkelijk op te vangen door iedereen met een geschikt ontvangtoestel. Dat geldt ook voor straalverbindingen; bij deze verbindingen, meestal tussen twee hoge torens, wordt met richtantennes de zendenergie zo nauwkeurig mogelijk op de ontvangstantenne gericht. In het gebied tussen de torens is het signaal gemakkelijk op te pikken. Een groot deel van de zendenergie schiet de ontvanger voorbij en verdwijnt de ruimte in, maar grote landen (de Verenigde Staten met name) hebben speciale satellieten gelanceerd om deze signalen op te vangen. Onder de codenaam *Echelon* bouwen de Verenigde Staten en Groot-Brittannië aan een wereldwijd luisternetwerk waarmee vrijwel alle telefoongesprekken beluisterd kunnen worden. Campbell [Cam99] geeft een overzicht van de aftapmogelijkheden van de overheden van diverse grote landen.

Ook is het voor buitenstaanders mogelijk, telefoonlijnen af te tappen door in te breken in schakelkasten. Hiermee kunnen gesprekken worden beluisterd, maar ook kan data die via een modem wordt verstuurd worden onderschept.

Regels rond aftappen. Het afluisteren van gesprekken of data is in Nederland verboden (Artikel 139c van het Wetboek van Strafrecht), maar er zijn hierop heel veel uitzonderingen. Radioverkeer is vrij, want het artikel is niet van toepassing op "door middel van een radioelektrische ontvanginrichting ontvangen gegevens, tenzij om de ontvangst mogelijk te maken een bijzondere inspanning is geleverd of een niet toegestane ontvanginrichting is gebruikt". Wat hierbij onder een "bijzondere inspanning" wordt verstaan is niet precies duidelijk, maar het kraken van encryptie is in ieder geval een bijzondere inspanning. Het is niet toegestaan de verkregen informatie aan anderen bekend te maken.

In Nederland en België kan de politie de communicatie van verdachten aftappen als hiervoor toestemming is verkregen van de Rechter-Commissaris, respectievelijk de Onderzoeksrechter. Dit gebeurt alleen als iemand wordt verdacht van een misdrijf waarop ten minste vier jaar gevangenisstraf staat. In alle Europese landen moeten de telecommunicatiebedrijven aan deze aftapping meewerken. Het is zelfs zo, dat de uitgifte van vergunningen om een gsm- of UMTS-netwerk te exploiteren is gekoppeld aan de mogelijkheid gesprekken af te tappen. Ook veiligheidsdiensten hebben uitgebreide aftapbevoegdheden.

Conclusie. Wetgeving is niet het meest geschikte middel om gegevensverkeer tegen afluisteren te beschermen. De overheid zal het afluisteren door de eigen instellingen (politie en veiligheidsdiensten) nooit willen verbieden omdat dit een ernstige belemmering voor rechtshandhaving zou zijn. Een verbod op afluisteren is vrijwel niet te handhaven, omdat het afluisteren ongemerkt gebeurt.

Kortom, wie vertrouwelijkheid van zijn communicatie op prijs stelt kan hierbij niet op wettelijke bescherming vertrouwen, maar zal zelf technische maatregelen moeten nemen. Instellingen met een eigen netwerk kunnen proberen hun infrastructuur fysiek ontoegankelijk te maken, maar voor wie over netwerken van anderen communiceert is cryptografische versleuteling de enige manier om vertrouwelijk te kunnen communiceren.

11.3.2 Gebruik van cryptografie

Waar burgers (en daaronder criminelen!) hun communicatie alleen voor opsporingsinstanties kunnen verbergen door cryptografie is het begrijpelijk dat deze instanties vragen om inperking van het gebruik daarvan. Daartegenover staat steeds het belang van burgers en bedrijven, die een veilige communicatie eisen. In de recente geschiedenis heeft de roep om een verbod op cryptografie steeds het luidst geklonken in de Verenigde Staten. Waar encryptie niet kan worden verboden, pleiten de VS voor *key escrow* (zie verderop).

Discussie in Nederland. In 1994 werd er in Nederland een ontwerpwet opgesteld waarin het gebruik van cryptografie aan strenge banden werd gelegd. Voor het gebruik van encryptie zou een vergunning worden vereist, waarbij de aanvrager zou moeten aantonen een belang bij versleuteling te hebben. Bovendien zou een kopie van de sleutel aan de overheid moeten worden afgegeven (key escrow). Het ontwerp stuitte op zeer veel kritiek: het zou technisch onuitvoerbaar zijn, de kosten van de sleutelopslag zouden groot zijn, en handhaving van het verbod vrijwel onmogelijk. Als belangrijkste bezwaar werd aangevoerd dat het middel een te zware inbreuk op de privacy doet om het doel (opsporing van criminelen) te rechtvaardigen. Het wetsontwerp is ingetrokken en leidt nu een stoffig bestaan in een Haagse bureaulade.

Discussie in Europa. De discussie over inperking van cryptografie is in Amerika nog steeds gaande. Voorstanders van inperking grepen de terreuraanslag op het WTC (11 september 2001) aan om hun argumenten opnieuw naar voren te brengen. In Europa echter kan deze discussie inmiddels als een achterhoedegevecht worden beschouwd. Men is er sterker geneigd erop te letten dat waar vrijheden van de burger worden ingeperkt ten behoeve van nationale veiligheid of opsporing, de maatregelen effectief zijn en de inperking van vrijheden in verhouding staat met het verwachte voordeel. Een uitgebreid overzicht van argumenten voor en tegen inperking vindt men in de dissertatie van Koops [Koo98].

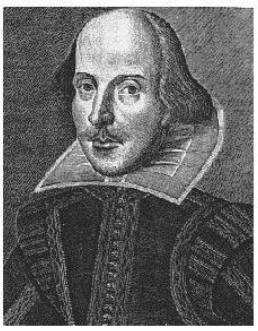
Nu is een verbod op cryptografie vrijwel niet te handhaven, wat voor een deel komt doordat versleutelde data in feite niet is te onderscheiden van random bits (behalve door wie de sleutel kent). Handhaving van een cryptografieverbod zou dus een verbod betekenen op het verzenden van random bits. Nu kan men altijd random bits 'verstoppen' in onschuldig uitziende data met technieken uit de zogenaamde *steganografie* (kader 11.1). In het algemeen verstaat men onder steganografie elke techniek waarmee wordt gecommuniceerd, zonder dat voor buitenstaanders zichtbaar is dat er sprake is van communicatie. Zo kan bijvoorbeeld het buiten bedrijf stellen van een op zich onschuldige website een afgesproken teken zijn om tot een bepaalde handeling over te gaan.

Verder kan men zich afvragen of zware criminelen zich überhaupt zullen storen aan een verbod op cryptografie. Men zal bij het beramen van een terreuraanslag of zware misdaad toch van cryptografie gebruikmaken, want de straf die daarop staat zal veel lager zijn dan die op het beraamde misdrijf. Een vergelijking met de vuurwapenwet kan hier worden gemaakt: criminelen overtreden deze wet liever dan 'bloot' over straat te gaan. De wet is dus *niet* effectief in het beïnvloeden van het gedrag van criminelen, maar alleen 'brave burgers' worden erdoor beperkt. Van vuurwapens kan men echter zeggen, dat ze in de handen van brave burgers meer kwaad dan goed aanrichten, waarmee de wapenwet kan worden gemotiveerd. Cryptografie echter is een belangrijke en nuttige techniek, ook voor goedwillende burgers en bedrijven, die er hun legitieme belangen mee beschermen.

In Europa werden extreme posities ingenomen door Noorwegen en Frankrijk. Frankrijk kende rond 1992 een bijna totaal verbod op het gebruik van cryptografie: voor verkoop van

Kader 11.1: Hoe werkt steganografie? Een voorbeeld.

Een digitale foto is een lange rij getallen, die helderheidswaarden van pixels voorstellen. De waarden zijn getallen tussen 0 en 255, en het plaatje wordt nauwelijks merkbaar beïnvloed als de getallen met 1 worden verhoogd of verlaagd. Nu kan een geheime boodschap in een fotobestand worden verweven door steeds de laatste (minst belangrijke) bit van elk getal te vervangen door een bit van de boodschap. Programmatuur om deze vervlechting uit te voeren op beeld, muziek of tekst is op het internet beschikbaar.
De linkerafbeelding is een portret van William Shakespeare (door Droeshout); in de rechterafbeelding zijn alle minst significante bits vervangen door de boodschap `Steganography is the art and science of communicating in a way which hides the existence of the communication. In contrast to cryptography, where the enemy is allowed to detect, intercept and modify messages without being able to violate certain security premises guaranteed by a cryptosystem, the goal of steganography is to hide messages inside other harmless messages in a way that does not allow any enemy to even detect that there is a second secret message present.`



De plaatjes zijn op het oog al vrijwel niet te onderscheiden, en als het origineel ontbreekt is er aan het behandelde plaatje niets merkwaardigs te zien. Het gebruikte programma (S-Tools) kan ook boodschappen verstoppen in MP3-bestanden of in tekst (waarbij de keuze tussen één of twee spaties tussen een woord wordt gebruikt om een bit te coderen). Er bestaat weer andere software die plaatjes kan screenen op ingebrachte boodschappen van bepaalde, bekende steganografiesoftware. Daarom moet steganografie steeds met cryptografie worden gecombineerd: zelfs als de boodschap wordt ontdekt, kan zij nog niet worden gelezen.
Er wordt gesuggereerd dat terroristen deze techniek gebruiken om instructies naar hun agenten via internet te kunnen versturen, maar de bewijzen hiervoor zijn erg dun: "Suspected ringleader Mohamed Atta was seen repeatedly by witnesses using his Hotmail account at public libraries in Florida to surf the Internet, downloading what appeared to be pictures of children and scenes of the Middle East."

producten was een vergunning nodig, waarvoor men technische details over het systeem aan de overheid moest geven. Noorwegen heeft zich steeds tegen beperking van cryptografie verzet en overwoog als eerste land een wet waarin cryptografie juist werd verplicht (bij opslag van medische gegevens). In de jaren negentig was er in Europa een duidelijke tendens in de richting van liberalisering van cryptografie.

De Raad van Europa sprak in 1995 uit dat er maatregelen overwogen moeten worden om het negatieve effect van cryptografie op justitieel onderzoek te beperken, maar zonder het legitieme gebruik ervan meer dan nodig aan banden te leggen. In 1999 ging de Raad nog verder: cryptografie werd genoemd als legitiem middel om de privacy te beschermen, zodat lidstaten haar niet zouden kunnen verbieden. Wel werd de mogelijkheid open gelaten dat overheden sleutels kunnen opvragen in het kader van een justitieel onderzoek.

In de Europese Unie werd vanaf 1996 gesproken over voorwaarden waaraan Trusted Third Parties zouden moeten voldoen. In het begin werd key escrow als mogelijkheid opengehouden om te voorzien in toegang tot informatie door opsporingsinstanties, maar vanaf 1997 werd het economische en maatschappelijke belang van cryptografie benadrukt. Hiermee werd afstand genomen van key escrow, maar de Ministers van Justitie en Binnenlandse Zaken pleitten in 1998 weer voor toegang tot informatie door opsporingsinstanties. De huidige situatie (2002) is dat men afwacht hoe het gebruik van cryptografie door zware criminelen en terroristen zal uitpakken.

Situatie in Nederland en België. De wetgeving is inmiddels in Frankrijk sterk geliberaliseerd. Voor verkoop van versleutelingsproducten is nog een meldingsplicht, maar het gebruik van cryptografie is vrij.

In Nederland zijn geen beperkingen op het gebruik van cryptografie van kracht. België kende vanaf 1994 een regeling dat communicatieapparatuur in beslag kan worden genomen als die justitiële aftapping onmogelijk maakte. Toen men begreep dat deze bepaling feitelijk een verbod op encryptie (zonder key escrow) inhield, werd in 1997 deze wet veranderd. In België zijn nu geen beperkingen op het gebruik van cryptografie van kracht.

11.3.3 Sleutelafgifte

Wanneer versleuteling niet zonder meer wordt verboden, kan de overheid haar opsporingsinstanties nog een aftapmogelijkheid geven door te zorgen dat die instanties over de sleutel kunnen beschikken. Zo is de communicatie tussen verschillende partijen wel goed beveiligd tegen afluisteren door andere burgers of criminelen, maar toch toegankelijk voor de overheid.

De overheid kan stimuleren of eisen dat alle cryptografische sleutels zonder meer ter beschikking van de overheid worden gesteld; men spreekt dan van *key escrow* (of *key recovery*). Een andere mogelijkheid is, de sleutel op te vragen op het moment dat dit in het kader van een onderzoek nodig is.

Key escrow of key recovery. De situatie waarin decryptiesleutels bij voorbaat bij een andere partij worden gedeponeerd werd vanouds aangeduid met *key escrow*. Tegen key escrow bestond veel weerstand, omdat burgers het onaangenaam vonden dat de overheid alle berichten 'over de schouder' mee zou kunnen lezen. Voorstanders van key escrow begonnen erop te wijzen dat sleutelafgifte ook haar nut zou bewijzen ingeval de gebruiker zijn geheime sleutel zou verliezen of vergeten. In dat geval zou de sleutel immers ook ten behoeve van de gebruiker uit de opslag kunnen worden opgevraagd; men begon ook te spreken over *key recovery* in plaats van de beladen term key escrow.

Reguleren van key escrow. Voor de invoering van key escrow zijn voor wat de regelgeving betreft verschillende mogelijkheden.

1. *Wettelijke verplichting.* Een gebruiker mag zelf bepalen welk cryptosysteem hij gebruikt, maar is wettelijk verplicht de gebruikte sleutel af te geven. Deze weg werd gekozen in het voorontwerp van een wet dat in 1993 in Nederland circuleerde.

2. *Keuze voor een cryptosysteem met escrow.* De overheid kiest een of enkele cryptografische algoritmen uit die in het land gebruikt mogen worden en brengt deze systemen, met ingebouwde key escrow, op de markt. Gebruik van andere systemen wordt dan ontmoedigd of ronduit verboden.

Kader 11.2: Geheimhouding met kaf en koren

Discussies over key escrow betreffen altijd sleutels voor geheimhouding; het is buiten discussie dat sleutels voor handtekeningen of berichtauthenticatie niet voor afgifte in aanmerking komen. Ronald Rivest liet zien dat je op basis van (symmetrische) berichtauthenticatie ook vertrouwelijkheid kunt krijgen, waarbij er dan wel voor elke bit van de boodschap twee berichten worden gestuurd. Voor de i^{e} bit van de boodschap stuurt Alice berichten $(i, 0)$ én $(i, 1)$, maar van een ervan klopt de Message Authentication Code (MAC) wel en voor de andere niet. Bob verifieert de MAC en verwerpt het foute bericht (het kaf) en accepteert alleen de juiste bit (het koren).
Omdat een afluisteraar (die niet over de sleutel beschikt waarmee de MAC wordt berekend en geverifieerd) niet kan zien welk bericht een goede MAC heeft, is niet te zien of de bit een 0 of een 1 moet zijn. De methode kan nauwelijks worden verboden omdat toepassing nooit bewijsbaar is: van berichten met een incorrecte MAC kan immers niet worden bewezen dat Alice ze heeft gestuurd.
Rivest ontwierp dit schema niet met de bedoeling dat het ook echt gebruikt zou gaan worden, maar om aan te tonen dat escrow van encryptiesleutels niet zinvol is zonder ook escrow van authenticatiesleutels toe te passen. En daarmee, dat key escrow een onwerkbaar principe is.

3. *Regeling via Certificate Authorities.* Een wettelijke verplichting wordt opgelegd aan Trusted Third Parties die public-keycertificaten afgeven. Wanneer de sleutels voor encryptie kunnen worden gebruikt is de CA verplicht de geheime sleutel in bezit te hebben en op justitieel bevel af te staan.

Discussie rond key escrow. In het publieke debat rond key escrow overheerst de mening dat escrow een te zwaar middel is om het doel te heiligen, en bovendien dat key escrow het belangrijkste probleem in het geheel niet oplost; zie Rivest [Riv98].

Key escrow moet ervoor zorgen dat opsporingsinstanties toegang blijven houden tot informatie die door criminelen wordt opgeslagen of verstuurd. Nu blijkt dit probleem ten eerste helemaal niet zo groot te zijn. Het aantal zaken waarin justitie met versleutelde gegevens te maken krijgt is klein, het aantal zaken waarin de beveiliging niet kan worden gekraakt is nog kleiner, en in de meeste van die gevallen is er voldoende ander bewijs om de verdachte te kunnen veroordelen. (De meeste zaken betreffen overigens niet, zoals vaak wordt gedacht, terrorisme of drugshandel, maar handel in kinderporno.) Verder zullen juist criminelen zich van reglementering van cryptografie niets aantrekken en gebruik blijven maken van systemen zonder escrow, zoals PGP. De bijdrage die van escrow kan worden verwacht aan wetshandhaving is dus gering.

Voor partijen die zich wel onderwerpen aan het escrowregime zijn er belangrijke risico's. De opgeslagen sleutels kunnen worden misbruikt, namelijk doordat partijen binnen de overheid

de sleutels gaan gebruiken zonder dat er justitiële aanleiding voor is, bijvoorbeeld om politieke tegenstanders uit te kunnen schakelen. De opslagcentra zullen een belangrijk doelwit zijn voor criminelen die computermisdrijven willen plegen: een inbraak in zo'n opslagcentrum kan in één keer een flink aantal belangrijke sleutels opleveren. Ten slotte zijn er de technische risico's die verbonden zijn aan het transport en de uitvoer van geheime sleutels. In principe kan een cryptografische module zeer veilig worden ontworpen als de generatie van de geheime sleutel binnen de module plaatsvindt, en alleen de publieke sleutel kan worden uitgevoerd. Verplichte key escrow maakt een uitvoerkanaal voor de geheime sleutel nodig en dit opent een mogelijke aanval op de module. De kosten van een key-escrowsysteem, inclusief beveiligingen tegen misbruik, zijn hoog.

De belangrijkste discussie moet zich echter afspelen rond het volgende zeer principiële punt: mag de overheid, om wat voor goede reden dan ook, zo diep ingrijpen in de persoonlijke vrijheid van de burgers als door het verbieden van vertrouwelijke communicatie? Het antwoord dat door de meeste specialisten wordt gegeven is 'nee'.

Key escrow binnen organisaties. Key escrow (of een aanvullende decryptiesleutel, zie sectie 11.3.4) wordt soms toegepast binnen een bedrijf of organisatie. Medewerkers ontvangen of beheren informatie die niet primair persoonlijk is, maar van belang is voor de bedrijfsvoering. Bij ziekte of ontslag van een medewerker, of een conflict tussen medewerkers, moet de continuïteit van het bedrijf gewaarborgd zijn.

De overheersende mening in dezen is dat de belangen die een bedrijf heeft bij escrow van sleutels van werknemers gerechtvaardigd zijn, en dat toepassing van escrow dus acceptabel is. Uiteraard moeten werknemers ervan op de hoogte zijn dat anderen in het bedrijf toegang hebben tot alle informatie op de systemen binnen het bedrijf. De werknemer kan er dan voor kiezen om persoonlijke informatie niet binnen het bedrijf op te slaan.

De inrichting van de sleutelopslag is in bedrijven doorgaans erg eenvoudig: meestal is er een partij, bijvoorbeeld een manager of systeembeheerder, die toegang tot alle sleutels heeft. Soms worden ook verdeelschema's (zie hoofdstuk 9) toegepast zodat alleen meerdere functionarissen tezamen toegang tot de informatie kunnen krijgen.

Een alternatief, dat een compromis kan vormen tussen de belangen van bedrijven en die van hun werknemers, is het gebruik van een aanvullende decryptiesleutel. Hierbij wordt een bericht zo versleuteld, dat het niet alleen met de geheime sleutel van de geadresseerde, maar ook met een tweede sleutel kan worden geopend. Dit mechanisme (*Additional Decryption Key* genaamd) is toegevoegd aan PGP om aan wensen uit het bedrijfsleven tegemoet te komen; zie kader 11.3.

Decryptiebevel bij onderzoek. Wanneer de overheid geen a priori toegang tot versleutelde gegevens heeft, kan de medewerking van sleutelhouders worden gevraagd wanneer in het kader van een onderzoek versleutelde gegevens worden aangetroffen. Personen van wie justitie aan kan nemen dat ze over de sleutel beschikken worden dan verplicht aan het onderzoek mee te werken, hetzij door de sleutel af te staan, dan wel door de gegevens te ontsleutelen.

Een dergelijke regeling is in veel landen van kracht, waarbij echter meestal de verdachte zelf is uitgezonderd. Het is namelijk altijd een recht van een verdachte om niet mee te werken aan zijn eigen veroordeling. Ook familieleden van de verdachte kunnen zich beroepen op een verschoningsrecht, dat wil zeggen dat zij geen informatie hoeven te geven die belastend kan zijn voor hun familielid.

Kader 11.3: Additional Decryption Key (ADK) in PGP

Omdat voor bedrijven verlies van data soms erger is dan verlies van privacy, werd vanaf versie 5.5 in PGP de mogelijkheid van een *Additional Decryption Key* gegeven.
Alice voegt aan haar publieke sleutel een tweede publieke sleutel toe, bijvoorbeeld die van haar werkgever Wilma. Wanneer Bob een bericht aan Alice stuurt, vraagt zijn PGP-software hem of hij gebruik wil maken van de tweede sleutel. Als Bob toestemt, wordt de sessiesleutel ook versleuteld met Wilma's sleutel en toegevoegd aan het bericht. Dan kan niet alleen Alice, maar ook Wilma het bericht lezen.
Meestal zal Wilma van deze mogelijkheid nooit gebruikmaken: Alice leest het bericht en verwerkt de informatie binnen het bedrijf. Pas wanneer Alice afwezig is en haar zaken binnen het bedrijf door een collega worden overgenomen gebruikt Wilma haar sleutel om het bericht te lezen. Het mechanisme geeft Bob de mogelijkheid om per bericht te beslissen of het voor Wilma leesbaar is; key escrow binnen het bedrijf maakt alle berichten aan Alice zonder meer toegankelijk voor Wilma. Als Bob een privé-bericht aan Alice' werkadres stuurt gebruikt hij de ADK niet.
Sommige gebruikers van PGP waren fel tegen deze toevoeging die zij zagen als een inbreuk op hun privacy. Erg pijnlijk voor de makers was het dat een van de ontevreden gebruikers, Ralph Senderik, een bug in de implementatie ontdekte. Oscar kon zelf, zonder inmenging van Alice, aan haar sleutel een ADK toevoegen en de combinatie doorspelen aan Bob; de controle dat de ADK echt door Alice was toegevoegd werkte niet. Zie Otterloo [Ott01].

Sleutelafgifte in Nederland en België. In Nederland werd in 1993 een voorontwerp van een wet opgesteld waarin het gebruik van cryptografie werd gekoppeld aan het vooraf afgeven van de geheime sleutels. Deze wet regelde dus feitelijk een verplichte key escrow, maar de wet is nooit verder gekomen dan een ontwerp.

In Nederland en in België is het gebruik van cryptografie vrij, en er bestaat geen plicht tot sleutelafgifte bij gebruik.

In beide landen kan een sleutelhouder worden verplicht mee te werken aan een strafrechtelijk onderzoek. In Nederland kan de politie, wanneer bij een huiszoeking versleutelde gegevens worden aangetroffen, opdracht geven deze te ontsleutelen aan personen van wie kan worden verwacht dat ze de sleutels kennen. Deze opdracht kan niet worden gegeven aan de verdachte of aan personen met verschoningsrecht.

In België is verplichting geregeld in Artikel 88 van de *Wet inzake Informaticacriminaliteit* (november 2000). Een onderzoeksrechter kan personen eisen informatie te geven over de toegang tot de versleutelde gegevens, of zelfs eisen dat de persoon de gegevens voor de onderzoeksrechter ontsleutelt. Van het laatste zijn de verdachte en personen met verschoningsrecht uitgezonderd.

11.3.4 Binding ElGamal

In de discussie over sleutelafgifte wordt het onderscheid tussen symmetrische en public-keycryptografie meestal niet gemaakt; de impliciete aanname is dat de sleutel van Oscar toegang geeft tot Oscars communicatie. Er is echter voor de bewijsvoering wel degelijk een relevant verschil.

Laten wij aannemen dat Oscar door justitie wordt verdacht van een ernstig misdrijf en dat zijn geheime sleutel mag worden gebruikt om zijn communicatie af te tappen. Het probleem ontstaat doordat slechts de helft van Oscars communicatie met deze sleutel te openen is. Berichten *aan* Oscar zijn met Oscars publieke sleutel versleuteld en kunnen met de geheime

sleutel worden geopend. Maar berichten *van* Oscar (aan zijn vermeende handlangers) zijn door Oscar versleuteld met andere public keys en kunnen met Oscars geheime sleutel niet worden geopend! Kortom, een justitieel bevel tegen Oscar geeft toegang tot zijn inkomende, maar niet tot zijn uitgaande post. Het is twijfelachtig of Oscar kan worden veroordeeld op basis van een e-mailbericht dat hij ontvangen heeft. Misschien heeft Oscar wel geantwoord op een verdacht e-mailbericht, maar de ontvanger woont misschien in een ander land of zijn sleutels kunnen om andere reden niet door justitie worden gebruikt.

Verplicht gebruik van ADK. Voor dit probleem is door Verheul en Van Tilborg [VT97] een technische oplossing ontworpen; vermeld moet worden dat deze nergens in wetgeving is geïmplementeerd. In dit systeem, *Binding ElGamal* genaamd, heeft de verzender van een bericht de plicht gebruik te maken van een Additional Decryption Key. De bijbehorende geheime sleutel is uiteraard in handen van een overheidsinstelling, het Data Recovery Center (DRC).

Wanneer Alice normaal een bericht M aan Bob stuurt, bevat het bericht twee belangrijke velden: (1) de body van het bericht is de encryptie van M met een symmetrich algoritme onder een *random* sleutel x (de sessiesleutel); (2) de header bevat de versleuteling van x met de publieke sleutel van Bob. Het DRC verspreidt nu zelf een publieke sleutel, en de overheid eist toevoeging van (3) de versleuteling van x met de publieke sleutel van het DRC.

Het systeem maakt voor de versleuteling gebruik van ElGamal, dus Bob heeft public key b (en kent a zodanig dat $g^a = b$) en het DRC heeft sleutel b^*. De normale encryptie van x in ElGamal bestaat uit een paar $(g^k,\ x \cdot b^k)$, voor een random getal k (zie sectie 6.3.2). Voor de versleuteling van x met sleutel b^* moet de verzender dezelfde waarde van k gebruiken, zodat het paar $(g^k,\ x \cdot b^{*k})$ wordt verkregen. Het tweemaal opsturen van g^k is natuurlijk niet nodig; daarom worden slechts drie getallen $u = g^k$, $v = x \cdot b^k$ en $v^* = x \cdot b^{*k}$ in de berichtheader opgenomen. Bob verkrijgt x als v/u^a en het DRC als v/u^{*a^*}.

Controle op toegevoegde informatie. Een laatste probleem dat moet worden opgelost is dat Alice, als zij inderdaad kwade plannen in haar e-mail bespreekt, liever niet met de overheid samenwerkt en in plaats van $x \cdot b^{*k}$ een random getal u^* aan haar bericht kan toevoegen. Controle op de juistheid van het toegevoegde getal is dus nodig.

Deze controle is technisch het gemakkelijkst door $x = v/u^{*a^*}$ te berekenen en te kijken of de body van het bericht ermee te ontsleutelen is. Deze methode stuit echter op een organisatorisch en een juridisch bezwaar. Ten eerste is voor de controle de geheime sleutel a^* nodig, en wordt dus vereist dat het DRC alle controles uitvoert. Ten tweede houdt de controle in dat het bericht wordt geopend, en dat mag wettelijk pas als Alice al ergens van wordt verdacht, en nooit als onderdeel van een routinecontrole of burgers zich aan communicatieprotocollen houden.

In het ontwerp van Verheul en Van Tilborg is de controle op de juistheid van u^* mogelijk zonder gebruik van a^* en de controle maakt het niet mogelijk het bericht te lezen. Om deze controle uit te kunnen voeren werd boven de eis gesteld dat Alice bij de twee encrypties van x (namelijk voor Bob en voor het DRC) dezelfde waarde van k gebruikt. De twee encrypties zijn door de gemeenschappelijke waarde van k aan elkaar gebonden volgens een rekenkundige relatie. Er geldt namelijk $u = g^k$ en (ga na!) $v/v^* = (b/b^*)^k$, met andere woorden, er geldt de Diffie-Hellman-relatie

$$\mathit{DifHel}_g(u,\ b/b^*,\ v/v^*),$$

waarbij Alice bovendien de verborgen exponent k kent.

Van een Diffie-Hellman-relatie is een zero-knowledge proof te geven en dit kan ook in niet-interactieve vorm worden gedaan; zie sectie 9.2.3. Hiervan wordt op deze manier gebruikge-

maakt: Alice moet een vierde veld aan haar bericht toevoegen, namelijk (4) een niet-interactief bewijs dat de encrypties in velden (1) en (2) hetzelfde getal verbergen. (We laten de uitwerking van de details, waarbij s, t en y worden berekend als in protocol 9.8, over aan de lezer.) De controle van het bewijs gebruikt alleen informatie uit het bericht, en de publieke sleutels van Bob en het DRC, en kan dus door iedere partij die het bericht kan zien worden uitgevoerd. Service providers kunnen berichten weigeren als dit bewijs niet klopt, en opsporingsinstanties kunnen bij het afluisteren van de telecommunicatie-infrastructuur gericht naar zulke 'foute' berichten zoeken.

Verheul en Van Tilborg bewijzen dat de toegevoegde informatie het systeem *niet* minder veilig maakt dan het normale ElGamal-algoritme.

11.4 Verdere wetgeving

Omdat cryptografie in allerlei sectoren van de samenleving een rol kan spelen, zijn ook allerlei regels op zeker moment voor de cryptografie van belang. Als voorbeeld noemen we hier de Kieswet, waarmee rekening moet worden gehouden bij het ontwerp van protocollen voor elektronische verkiezingen (sectie 14.3). Bij de implementatie van elektronisch betaalverkeer (hoofdstuk 13) zijn regels rond geld van belang.

11.4.1 Digitale handtekening

Rond het belang van een digitale handtekening is nauwelijks controverse. Er is brede overeenstemming dat een wettelijke status voor deze handtekening van belang is voor de ontwikkeling van elektronische handel over het internet.

De Europese Unie heeft in 1999 een richtlijn aangenomen (1999/93/EG) die de lidstaten verplicht digitale handtekeningen in hun wetgeving gelijk te stellen aan geschreven handtekeningen. Lidstaten mogen wel technische eisen aan de digitale handtekening of aan de publickeycertificaten stellen, maar dan moet ook elke handtekening die eraan voldoet wettig worden geaccepteerd. Lidstaten mogen het uitgeven van certificaten niet aan vergunningen binden, moeten de aansprakelijkheid van certificaatuitgevers regelen en zijn verplicht ook certificaten en handtekeningen uit andere lidstaten te erkennen.

In Nederland wordt de richtlijn uitgevoerd in de *Wet Elektronische Handtekeningen*, waarin aanvullingen op het Burgerlijk Wetboek worden gegeven. De elektronische handtekening krijgt in principe een rechtsgeldigheid die gelijk is aan een geschreven handtekening (B.W. artikel 15a), mits er gebruik is gemaakt van een gekwalificeerd certificaat (art. 15b). Als een partij certificaten aanbiedt is die partij ook aansprakelijk voor schade die ontstaat wanneer anderen in vertrouwen op dat certificaat handelen (art. 196b). Wel kunnen in het certificaat door de uitgever beperkingen op het gebruik, en de aansprakelijkheid, worden opgenomen.

De implementatie van de richtlijn in België is inhoudelijk vrijwel gelijk en vastgelegd in wetsvoorstel 0322/006 "vaststelling van bepaalde regels in verband met het juridisch kader voor elektronische handtekeningen en certificatiediensten". Certificatiedienstverleners hebben een registratieplicht en moeten zich tegen de aansprakelijkheid verzekeren (art. 4 lid 2).

De informatie in deze sectie is ontleend aan de Digital Signature Law Survey van Simone van der Hof, `http://rechten.kub.nl/simone/ds-lawsu.htm`.

Kader 11.4: Wie een hond wil slaan vindt in DMCA een stok.

Sony introduceerde met de *Aibo* een digitaal huisdier, in de hoop dat het idee van huisrobots bij het publiek ingang zou vinden. Een echte Aibo-liefhebber heeft soms wel vijf of tien van deze robots in huis.

Een hacker met schuilnaam AiboPet vond door reverse engineering uit hoe de robot werkt, schreef nieuwe software en publiceerde die op zijn website, tot groot plezier van de vele fans. Helaas, de software in de robot was versleuteld, waardoor AiboPet niet alleen een opmerkelijke technische prestatie leverde, maar ook een kopieerbeveiliging doorbrak.
Sony schreef AiboPet dat hij de Digital Millennium Copyright Act had overtreden en beval hem de software van internet te verwijderen, wat hij deed. Een storm van protesten van Aibo-houders bracht Sony tot andere gedachten, en er ontstond zelfs een samenwerking tussen Sony en AiboPet. Het bleek niet het grootste economische belang van Sony te zijn om op dat moment de rechten uit de DMCA uit te oefenen.
`http://www.sciam.com/explorations/2002/012102aibo/`

11.4.2 Bescherming van intellectueel eigendom

Boeken, muziek en films zijn beschermd door het auteursrecht, dat bepaalt dat alleen de schrijver of maker van deze werken ze mag publiceren of vermenigvuldigen. Cryptografische technieken maken het mogelijk om werken te beveiligen tegen inbreuk op dit recht, bijvoorbeeld door video's of cd's te beveiligen tegen kopiëren. De techniek kan de toegang tot het werk verder inperken dan de wettelijke bescherming reikt. Video's kunnen bijvoorbeeld door een technische bescherming zo worden gemaakt, dat ze wel op Amerikaanse, maar niet op Europese spelers kunnen worden bekeken. Het auteursrecht spreekt alleen over het kopiëren van werken, niet over het bekijken ervan.

Verenigde Staten: Digital Millennium Copyright Act. Rond deze bescherming van intellectueel eigendom is in de afgelopen jaren in de Verenigde Staten een sterk belangenconflict gegroeid. Onder druk van uitgevers en platenmaatschappijen is een wet van kracht geworden, de *Digital Millennium Copyright Act*, die het intellectueel eigendom regelt. Deze DMCA verbiedt ook het doorbreken van enige technische beveiliging op een werk of het meewerken aan of mogelijk maken van het doorbreken van een technische beveiliging.

Enkele wetenschappelijk onderzoekers hebben al ervaren dat ze op grond van deze wet niet meer konden publiceren over cryptografisch onderzoek. Een andere kritiek op de wet is, dat de technische beveiligingen soms gebruikmaking van een werk blokkeren, terwijl dat gebruik wel door de wet is toegestaan, zoals citeren of het maken van een reservekopie. Ook voor deze legale doeleinden mag de beveiliging niet worden doorbroken, zodat de positie van de consument wordt verzwakt.

Europese Unie: harmonisatie van het auteursrecht. Vanaf 1994 werd er in de Europese Unie over de auteursrechten nagedacht, met als doel te komen tot harmonisatie van de wetgeving in de lidstaten, en aanpassing van de regels aan de eisen van het digitale tijdperk. In 2001 werd richtlijn 2001/29/EG aangenomen en de bepalingen hiervan moeten binnen achttien maanden door de lidstaten in wetgeving zijn omgezet.

De richtlijn kent aan auteurs (hieronder verstaan we ook uitvoerend kunstenaars, filmmakers, mucici en omroeporganisaties) het recht toe te beslissen over reproductie (art. 2), vertoning (art. 3) en distributie (art. 4) van hun werk. Opslag in computergeheugens of buffers in een communicatienetwerk, nodig voor gebruik van het werk, in EU-jargon 'tijdelijke reproductiehandelingen' genaamd, zijn hiervan uitgezonderd (art. 5). Verder worden een twintigtal uitzonderingen genoemd waarover de lidstaten zelf mogen beslissen, zoals het gebruik van een afbeelding van een gebouw voor de wederopbouw ervan.

Lidstaten zijn volgens Artikel 6 verplicht technische voorzieningen wettelijk te beschermen, en producten en diensten voor het kraken ervan te verbieden. Een technische voorziening is elk onderdeel dat beoogt inbreuken op het recht van de auteur te voorkomen. Volgens Koelman [Koe01] is ook in deze richtlijn de bescherming van technische voorzieningen te sterk. Rechthebbenden kunnen allerlei vormen van gebruik die niet door het auteursrecht zijn verboden, afschermen achter technische voorzieningen. Als diezelfde voorzieningen ook zijn aangebracht om gebruik tegen te gaan dat wel is verboden, vallen de voorzieningen onder de bescherming en mogen ze niet worden doorbroken.

Auteurswet in Nederland. Volgens de aangepaste Auteurswet zijn onder andere deze manieren van gebruik altijd toegestaan: doorverkopen (art. 12b); tijdelijke reproductiehandelingen (art. 13a); citeren uit kranten en tijdschriften (art. 15); onderwijs en onderzoek (art. 16); kopiëren voor eigen gebruik (art. 16b). De bescherming van technische voorzieningen volgt de Europese richtlijn nauwgezet en is geregeld in artikel 29a en 29b.

Samenvatting en conclusies

Cryptografie en wetshandhaving staan met elkaar in een haat-liefdeverhouding. Cryptografie kan veel misdrijven voorkomen en beschermt de politie tegen afluisteren door criminelen, maar belemmert ook de informatie- en bewijsvergaring door de politie. Soms worden daarom wettelijke maatregelen genomen zoals beperking van verkoop en export van cryptografie, beperking van gebruik van versleuteling of de verplichting om decryptiesleutels af te geven aan de wetshandhavers.

In de afgelopen jaren is het besef gegroeid dat deze maatregelen weinig effectief zijn omdat ze vaak gemakkelijk technisch te omzeilen zijn, dat ze duur zijn, dat ze de veiligheid van cryptografische systemen verminderen, en dat ze een te grote inbreuk vormen op democratische rechten van de burger. De maatregelen worden dan ook in de meeste Europese landen, voor zover ze al waren ingevoerd, versoepeld.

In België en Nederland gelden geen beperkingen op de verkoop en het gebruik van cryptografische producten. Voor export van deze producten buiten de Benelux is een vergunning vereist.

Opgaven bij hoofdstuk 11

Project 11.1 *Onderzoek de wettelijke status van digitale handtekeningen en schrijf hierover een scriptie.*

Project 11.2 *Onderzoek de wettelijke bescherming van intellectueel eigendom en schrijf hierover een scriptie.*

Opgave 11.3 Multi-ElGamal. *(Uit het tentamen van januari 1999.)*
In het ElGamal systeem zijn een priemgetal p en een element g in $\mathbb{Z}_p^*$ afgesproken. Een client met private key a heeft als public key $b = g^a$. De Multi-ElGamal codering *van x is de rij $(g^k,\ b_1^k.x,\ b_2^k.x,\ \ldots,\ b_n^k.x)$.*

(a) Wat is hierin k, $b_1, \ldots, b_n$ en x?
(b) Hoe vindt gebruiker i uit deze codering S?
(c) Waar kun je Multi-ElGamal voor gebruiken?
(d) Hoe kun je bewijzen dat Multi-ElGamal niet minder veilig is dan ElGamal?

Hoofdstuk 12

Smartcards

In toenemende mate wordt in digitale systemen gebruikgemaakt van smartcards. Zo'n kaart bevat niet alleen data, maar implementeert ook de procedures om met deze data te werken. Niet alleen is dat erg handig, maar het biedt beheerders van systemen ook de mogelijkheid om gebruikers bepaalde gevoelige data ter beschikking te stellen en tegelijkertijd in de hand te houden wat de gebruiker met die data gaat doen. Ander gebruik, in het bijzonder kopiëren van data of kaart, is in principe niet mogelijk.

De beveiliging van smartcards is geen cryptografie; vergelijk sectie 12.3 met definitie 0.1. Toch is een hoofdstuk over smartcards in een boek over cryptografie zinvol, ten eerste omdat veel cryptografische systemen tegenwoordig van smartcards gebruikmaken. Daarnaast is het goed zich van de beperkingen van deze kaarten bewust te zijn, zodat de risico's van de systemen kunnen worden ingeschat. Ten slotte stelt de smartcard ook eisen aan het ontwerp van cryptografische algoritmen; dit punt wordt op diverse plaatsen in het hoofdstuk toegelicht.

Sectie 12.1 vertelt over het gebruik van smartcards, wat ze kunnen, en wat we ervan verwachten. Het grootste deel van het hoofdstuk gaat over het aanvallen van smartcards, namelijk sectie 12.2 over aanvallen van buitenaf en sectie 12.3 over het openbreken van de kaarten.

Kader 12.1: De smartcard als verzamelobject.

De kaartuitgevers proberen de kaarten er met kleuren zo aantrekkelijk mogelijk uit te laten zien. Door afwisseling in de aangeboden afbeeldingen worden 'opgebruikte' smartcards een leuk verzamelobject. De afgebeelde kaarten: een Nederlandse chipknip, en een Duitse, Tsjechische en Franse telefoonkaart.
"Toch knap, een complete processor met geheugen, zo klein dat hij in een bankpas past." De prestatie die door de fabrikanten wordt geleverd is nog knapper, want het actieve deel van de smartcard vult bij lange na niet de hele pas. De processor en geheugen (RAM, ROM en EEPROM) zijn geïntegreerd op een enkele chip met een afmeting van 2 à 3 millimeter, die zich midden achter de metalen contacten bevindt. Wat er verder omheen zit is alleen plastic en dient als advertentieruimte en om te voorkomen dat mensen de kaart in hun broekzak kwijtraken.

12.1 Inleiding

12.1.1 Het Gebruik van smartcards

De verspreiding van smartcards heeft de laatste jaren een grote vlucht genomen.

De bekendste wegwerpkaart is waarschijnlijk de telefoonkaart (kader 12.1), sinds de vroege jaren negentig in diverse Europese landen te verkrijgen. De afmetingen van deze kaarten zijn gekozen als het al langer bestaande 'creditcardformaat", 86×54 millimeter, met een dikte van 0,76 millimeter. De oudere telefoonkaarten waren even groot, maar hadden alleen data op een magneetstrip.

Ook de elektronische betaalpas chipknip is een smartcard. Deze heeft dezelfde afmeting en heeft aan de achterzijde een magneetstrip om pinpasfuncties te ondersteunen. Gegevens op een magneetstrip liggen open en bloot, net als wanneer gegevens op papier zijn opgeschreven. Het enige verschil is dat het magneetformaat alleen machinaal te lezen en te schrijven is. De gegevens in een smartcard kunnen niet zonder meer worden veranderd, gekopieerd of geanalyseerd.

Een volgend voorbeeld van een smartcard is de SIM-kaart (Subscriber Identity Module) in een mobiele telefoon. Deze is veel kleiner dan een bankpas, maar voor de elektronica maakt dat niet uit; in beide gevallen bevindt die zich geheel achter het elektrische contact.

Verder vinden we smartcards in decryptieapparatuur voor betaal-tv. Ook hier is sprake van een zeer kleine chip ingebed in plastic behuizing, maar om aan de kennelijke behoefte van de gebruiker aan visualisatie te voldoen heeft de behuizing vaak de vorm van een sleuteltje.

De genoemde voorbeelden hebben gemeen dat de smartcard steeds toegang verleent tot systemen van ongekend grote commerciële waarde. Televisie en telefonie zijn bedrijfstakken met een miljardenomzet, die voor een deel via de smartcards moet worden afgerekend. Naast deze kaarten die voor een groot publiek zijn bestemd, vindt men ook tal van kaarten verspreid binnen een organisatie. De PrimaCryptionkaart (kader 4.1) dient voor versleuteling, maar ook worden er vaak kaarten toegepast voor toegangscontrole.

12.1.2 De techniek van smartcards

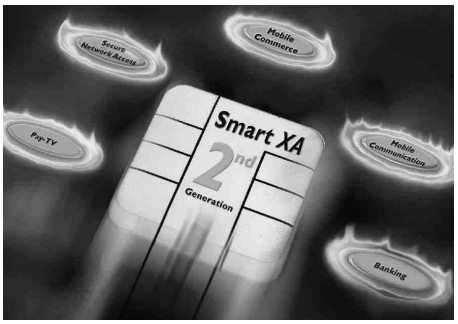
Een smartcard bevat ten minste een datageheugen en een processor; zie bijvoorbeeld Handschuh en Paillier [HP98]. Het geheugen bestaat uit drie delen met verschillende eigenschappen.

1. **Read-Only Memory (ROM).** Geheugen dat alleen kan worden gelezen maar niet geschreven. Hierin bevinden zich de onveranderlijke gegevens, met name de programmatuur van de kaart. De omvang van het ROM is typisch 20 à 40 kB.

2. **Random Access Memory (RAM).** Geheugen dat kan worden gelezen en geschreven. Het dient voor de tijdelijke opslag van gegevens, zoals tussenresultaten van berekeningen. Dit geheugen houdt alleen gegevens vast zolang de voedingsspanning aanwezig is, alles wordt dus gewist wanneer de kaart wordt uitgeschakeld (uit de lezer wordt gehaald). De omvang van het RAM is typisch 1 à 2 kB.

3. **Electrically Erasable Programmable Read-Only Memory (EEPROM)**. Dit geheugen kan worden gelezen en geschreven. Bovendien blijft de inhoud bij het uitschakelen van de voedingsspanning aanwezig. Het EEPROM is dus de plaats waar de kaart data kan bewerken en opslaan tot de volgende activering. De cryptografische sleutels van een

Kader 12.2: De Philips smartcardprocessor



Philips claimt met de SmartXA een smartcard te leveren met de kracht van een pc. Deze uitspraak doet de ontwikkeling in pc's van de laatste tien jaar geen recht, maar de prestaties van SmartXA zijn niettemin indrukwekkend.
De architectuur is 16 bits en voorziet in een interne firewall om verschillende applicaties tegen elkaar te beschermen. De kaart kan (zomer 2001) geleverd worden met 128kB ROM, 5kB RAM en 64kB EEPROM. De SmartXA heeft een speciale coprocessor voor 3DES en een FameX coprocessor voor getalberekeningen. FameX staat voor *Fast Accelerator for Modular Exponentiation Extended*. Een RSA-berekening op 1024-bitsgetallen kost hiermee ongeveer 400 milliseconden.
`www.semiconductors.philips.com/identification/products/smartxa/`

smartcard bevinden zich ook in het EEPROM. De omvang van het EEPROM is typisch 8 à 16 kB.

De processor in een smartcard is primitief in vergelijking met zijn collega's in pc's. Vaak zijn het 8-bitsprocessoren, een enkele kaart heeft 16-bitsprocessoren. Kaarten die voor asymmetrische cryptografie worden gebruikt hebben meestal een coprocessor voor getalberekeningen. De kloksnelheid is 5 à 20 MHz. De processor heeft een voorziening om data te ontvangen en te verzenden via de goudkleurige contacten op de kaart, waarbij de datasnelheid 100 tot 200 kbps bedraagt.

De rekentijd voor RSA of ElGamal met 1024-bitsmoduli kan voor een zo beperkte processor een probleem zijn. Daarom is gezocht naar cryptografische algoritmen die een even grote veiligheid kunnen bieden maar minder rekentijd vragen. Uit deze behoefte zijn onder andere de cryptografie op elliptische krommen en XTR ontstaan; zie sectie 15.2.

Een smartcard moet enkele jaren kunnen worden gebruikt, en geen batterij kan in de beschikbare ruimte zo lang energie leveren. Een smartcard heeft daarom vrijwel nooit een eigen stroomvoorziening: energie moet altijd vanuit de kaartlezer worden aangevoerd. De benodigde spanning is meestal 3 tot 5 volt en de afgenomen stroom is typisch 5mA.

Bij sommige kaarten wordt, behalve de energie, ook het kloksignaal vanuit de kaartlezer betrokken. Gebruikelijker is echter, dat de kaart een eigen klokoscillator aan boord heeft die wordt gesynchroniseerd vanuit de lezer (als bescherming tegen tijdaanvallen, zie verderop). Een smartcard heeft geen doorlopende klok waarmee de kaart zelf de datum en tijd bijhoudt; zo'n klok moet immers continu van stroom worden voorzien. Datum en tijd moet de kaart dus altijd van de kaartlezer vernemen, en de kaart 'weet' zelf nooit hoe lang hij ongebruikt is geweest.

De technische ontwikkelingen maken de smartcards ieder jaar 'smarter' met snellere processoren en meer geheugen. Daarnaast zijn er speciale voorzieningen die de veiligheid van de kaart kunnen vergroten, zoals een eigen stroombron (sectie 12.2.2), een keypad (kader 8.2) of een vingerafdruklezer. Als deze voorzieningen goedkoper worden, kunnen die in de toekomst ook in massaproducten worden toegepast.

12.1.3 De smartcard als secure platform

Een smartcard is een zeer geschikt platform voor de opslag van cryptografische sleutels en het uitvoeren van cryptografische berekeningen. In de eerste plaats is een smartcard hiervoor gewoon ontzettend handig. Je kunt de kaart erg gemakkelijk meenemen en beschikt dan over alle benodigde data, programmatuur en hardware. De gebruiker hoeft dus niet een 1024-bitssleutel uit zijn hoofd te leren en bij de pinautomaat de exponentiatie op een papiertje uit te rekenen. (Een exponentiatie van 1024-bitsgetallen zou met de hand ongeveer één mensenleven duren en bovendien 1200 kilogram papier kosten.)

Nog belangrijker is dat de smartcard wordt beschouwd als een zogenaamd *veilig platform* of *secure device*. Hiermee wordt bedoeld dat de data in de kaart beschermd is tegen alles wat niet al in de programmatuur staat. Er wordt aangenomen dat het niet mogelijk is om de data te benaderen anders dan via de ingebouwde software, en dat het niet mogelijk is die software te veranderen. Het veilig platform is ongevoelig voor virussen, de software werkt in alle gevallen correct, en de hardware is betrouwbaar. Kortom, 'knoeien' met de smartcard is niet mogelijk en daarom geldt de kaart als *knoeibestendig* (*tamper proof*).

Geen enkele kaart kan echter deze claim tegenover elke aanval en elke aanvaller weerstaan en daarom worden de aanvallen en aanvallers geclassificeerd. Anderson en Kuhn [AK96] onderscheiden drie typen aanvallers:

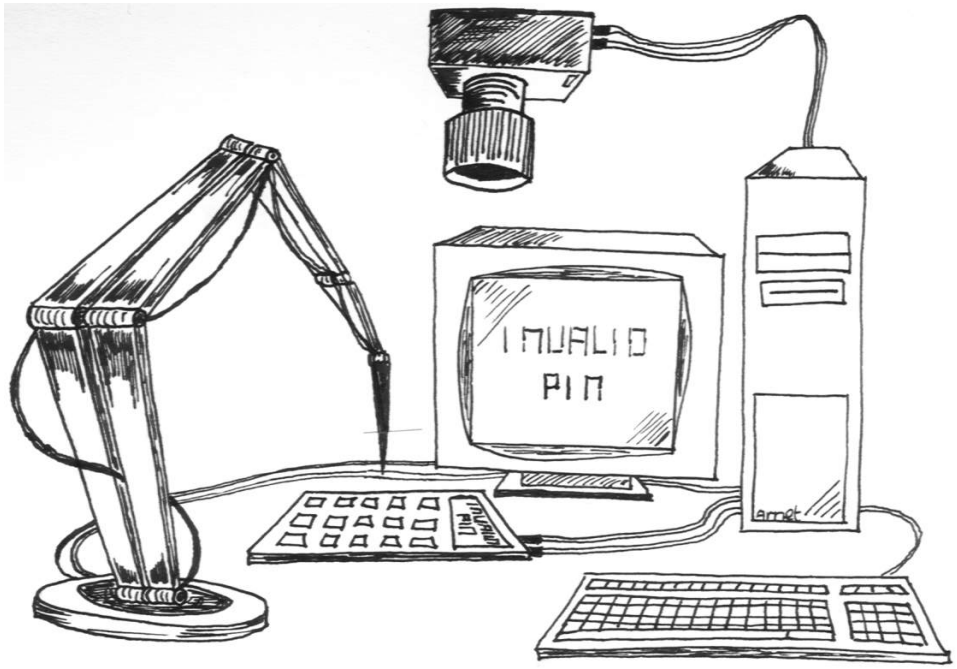
- **Klasse 1: slimme buitenstaander.** Aanvaller met matige kennis van het systeem en beschikkend over eenvoudige apparatuur.
- **Klasse 2: specialisten.** Mogelijk met detailkennis over de werking van het systeem, mogelijk toegang tot delen ervan en beschikkend over geavanceerde apparatuur.
- **Klasse 3: machtige organisaties.** Is in staat om meerdere teams op een aanval te zetten, kan veel apparatuur kopen en beschikt over specialistische, mogelijk zelfs geheime apparatuur.

Ook kijken we naar de mate van toegang tot de kaart die een aanvaller nodig heeft.

- **Observatie.** De aanvaller heeft interactie met de kaart tijdens normaal gebruik, maar de eigenaar neemt de kaart hierna weer mee. Voorbeeld: de aanvaller beheert de kaartterminal waarop met een betaalpas een of enkele malen wordt betaald.
- **Vrij gebruik.** De aanvaller heeft tijdelijk de beschikking over de kaart en kan hem dus heel veel gebruiken, maar de kaart moet onbeschadigd weer aan de eigenaar worden teruggegeven. Voorbeeld: de aanvaller neemt een creditcard 'even' mee naar achteren en geeft hem een tijdje later met de betaalbon weer terug.
- **Vernielen.** De aanvaller maakt de kaart kapot. Hij gebruikt er eventueel meerdere om te oefenen op dit specifieke model en wil dan uit een volgend exemplaar de geheimen achterhalen. Voorbeeld: de aanvaller koopt enkele videodecoders en vernielt deze om zelf nieuwe piraatdecoders te kunnen maken.

Tegen machtige organisaties die bereid zijn iets te vernielen is vrijwel niets bestand. Daarom worden bijvoorbeeld inspectiesondes voor de controle van kernwapenverdragen regelmatig geïnspecteerd. Knoeien is dan misschien niet geheel onmogelijk, maar op zijn minst hoopt men dit dan bij de volgende inspectie te kunnen merken. Veiligheid tegen machtige organisaties is vermoedelijk (volgens Anderson en Kuhn [AK96]) op slechts één terrein bereikt: de detonatie van kernwapens. De processoren liggen ingebed in het radioactieve plutonium en

Kader 12.3: Testen van pincodes



Data in organizers is vaak met een vier- of vijfcijferige pincode beveiligd. De afgebeelde machine neemt de saaie taak over van het achtereenvolgens uitproberen van tien- of honderdduizend codes. De camera herkent de cijfertoetsen en hiermee wordt de arm gestuurd, maar daarnaast wordt ook de respons van het display herkend en zodra de machine 'beet heeft' wordt de belangstellende met een bescheiden piepje achter de koffie vandaan geroepen.
De Identikey (kader 8.2) en veel soortgelijke apparaatjes gaan definitief kapot wanneer er driemaal achtereen een foute code wordt ingetoetst. De aanval wordt dan wel eens gecombineerd met foutinjectie om het ophogen of testen van de valsepinteller te frustreren.

er wordt aangenomen dat het voor 'schurkenstaten' makkelijker is om zelf een atoombom te maken dan om een Amerikaans exemplaar te stelen en de beveiliging te doorbreken.

12.2 Aanvallen op smartcards

De eerste aanvallen die we bekijken zijn niet-destructief. De aanvaller observeert de kaart tijdens berekeningen, waarbij hij wel de mogelijkheid heeft de kaartlezer te manipuleren.

Een aanvaller kan proberen de sleutel (deels) te raden door het invoer-uitvoergedrag van de kaart in kaart te brengen. Hiertegen moet het systeem beveiligd zijn op het niveau van de gebruikte protocollen. Als gebruikte identificatieprotocollen zero-knowledge zijn, geeft de uitvoer van de kaart geen informatie over de sleutel. Een handtekeningschema moet veilig zijn tegen een gekozen-cijfertekstaanval.

12.2.1 Tijdaanval

Niet alleen de uitvoer van de berekening, maar ook het verloop ervan hangt af van de geheime sleutel in de kaart en in het bijzonder geldt dit voor de duur van de berekening. Bij een *tijdaanval* (*timing attack*) beheert Oscar de kaartlezer en is hij in staat de tijd tussen het geven van een opdracht aan de kaart en het ontvangen van het antwoord nauwkeurig te meten.

Tijdaanval op RSA. Dat tijdmetingen informatie over de sleutel kunnen lekken werd geopperd door Kocher, en een dergelijke aanval werd uitgevoerd door Dhem e.a. [DKL+98]. De betreffende kaart berekent RSA-handtekeningen m^d over een aangeleverd bericht m en exponentieert met algoritme 6.3:

$S := m$;
for $i = l - 2$ **downto** 0 **do**
    $S := S * S$;
    if d_i **then** $S := S * m$

Hierin bestaat het geheim d uit de bits $d_{l-1}d_{l-2}\ldots d_1d_0$. In achtereenvolgende ondertekeningen is steeds m verschillend, maar de exponent d blijft gelijk en daarom realiseert het programma steeds evenveel vermenigvuldigingen (ongeveer l plus het aantal 1-en in d). Een vermenigvuldiging duurt iets langer wanneer aan het eind een reductie modulo n nodig is; dit kan op grond van de te vermenigvuldigen getallen worden beslist. Dus niet elke vermenigvuldiging duurt even lang en daarom is er in de totale responstijd een zekere spreiding.

Berichten m kunnen dan worden verdeeld in berichten M_1 en M_2 waarvoor d_{l-2} een andere invloed heeft op de duur van berekeningen op invoer uit M_1 dan op invoer uit M_2. Voor de eerste sleutelbit (d_{l-2}) wordt gekeken of in de berekening van $(m * m) * m$ een reductie wordt toegepast. Als deze berekening lang is, is $m \in M_1$ en als hij kort is, is $m \in M_2$. De duur van alle andere vermenigvuldigingen is onafhankelijk van de vraag of m in M_1 of M_2 zit.

Als $d_{l-2} = 0$ wordt de vermenigvuldiging op grond waarvan de verdeling is gemaakt niet uitgevoerd; de gemiddelde tijd is dan hetzelfde voor berichten in M_1 en M_2. Als $d_{l-2} = 1$ is er een vermenigvuldiging (de tweede) die in ieder geval lang is voor $m \in M_1$ en kort voor $m \in M_2$; de gemiddelde tijd is dan korter voor berichten in M_2. Er worden nu meerdere berichten aangeboden en er wordt gekeken naar een statistisch significant verschil in berekeningsduur tussen invoer uit M_1 en uit M_2. De volgende sleutelbits worden op soortgelijke manier aangepakt.

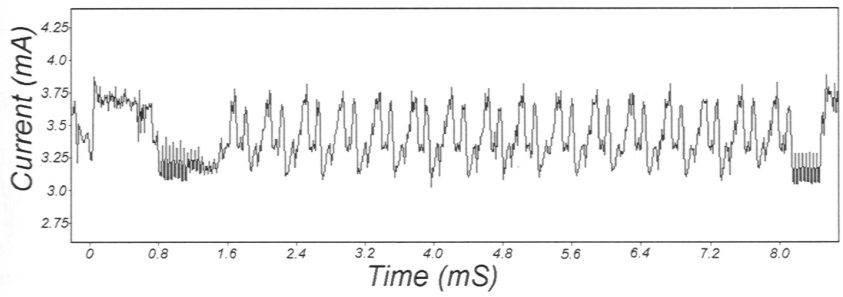
De te meten variatie is erg klein, bijvoorbeeld 400 processorcycles op een totale berekeningduur van 7,4 miljoen cycles! Daarom moeten per sleutelbit een flink aantal handtekeningen worden gezet. Voor een 512-bitssleutel werden ongeveer driehonderdduizend handtekeningen gevraagd, waarmee de smartcard enkele uren zoet is! Voor de aanval is een normale pc (met smartcardlezer) voldoende; hij kan worden uitgevoerd door een aanvaller van klasse 1 mits die enkele uren vrij gebruik van de kaart heeft.

Bescherming tegen tijdaanvallen. Een gebruikelijke bescherming tegen tijdaanvallen is het uitvoeren van overbodige berekeningen; deze bescherming wordt uitgevoerd op het niveau van softwarecodering. De laatste regel van het algoritme wordt dan bijvoorbeeld zo aangepast:

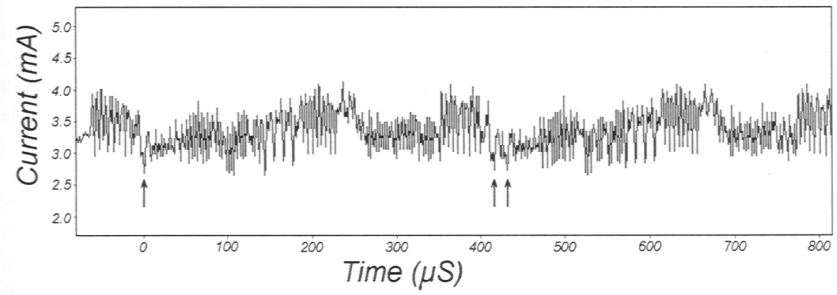
if d_i **then** $S := S * m$ **else** $T := S * m$

De vermenigvuldiging wordt uitgevoerd, ongeacht of d_i 0 of 1 is, en het tijdverschil wordt veel kleiner. Uiteraard wordt de kaart door het uitvoeren van allerlei overbodige stappen wel veel langzamer.

Kader 12.4: Stroomverbruik van een DES-smartcard



Het stroomgebruik van een smartcard kan in de tijd worden uitgezet met een oscilloscoop. Met een duurder apparaat kunnen de metingen sneller achtereen worden gedaan (horizontale resolutie) en nauwkeuriger (verticale resolutie). Voor een elektronisch laboratorium is een samplefrequentie van 1GHz (een meting per 10^{-9}s) bij een precisie van 1 procent normaal. De hobbyist kan onder de 500 euro beschikken over een sample frequentie van 20MHz, wat voldoende is voor deze afbeeldingen (ontleend aan Kocher, Jaffe en Jun [KJJ99]).



De eerste grafiek toont de stroomafname tijdens een DES-berekening; de zestien ronden van DES zijn in de grafiek gemakkelijk te herkennen. In de tweede grafiek zijn rondes 2 en 3 preciezer aangegeven. Elke ronde begint met een rotatie van de sleutelbits (zie sectie 2.1.2): in ronde 2 over één, en in ronde 3 over twee bits. De rotaties zijn in het stroomverbruik te zien (pijltjes).

Het AES-algoritme Rijndael gebruikt de sleutel alleen als operand in een XOR-operatie, waardoor de invloed van de sleutel op de tijdsduur nihil is.

12.2.2 Stroomaanval

Een smartcard heeft geen batterij, maar betrekt stroom uit de kaartlezer, die daarmee ook inzicht krijgt in *hoeveel* stroom de kaart op elk moment gebruikt; zie kader 12.4.

Principe van de stroomaanval. Op veel momenten tijdens het uitvoeren van een algoritme is het stroomverbruik afhankelijk van de bewerkte data. Bij DES zijn de momenten van sleutelrotatie gemakkelijk in het stroomverbruik te herkennen. De rotatie verschuift de bits een voor een, waarbij er eerst aan het begin een 0 wordt ingevuld, en op grond van de laatste bit wordt (met een **if**) besloten of de eerste positie wordt overschreven met 1. Op een nog nauwkeuriger grafiek is direct te zien of deze overschijving al dan niet plaats vindt, waarmee de sleutelbits direct kunnen worden afgelezen.

Wanneer informatie wordt gewonnen uit een enkele executie van de smartcard spreken we van een *enkele stroomaanval* (*simple power attack*). Het is ook mogelijk meerdere runs te bemeten en naar de verschillen te kijken; er wordt dan gesproken van *differentiële stroomaanval*

(*differential power attack*). Deze laatste techniek is nog krachtiger. Stroomaanvallen kunnen door een aanvaller van klasse 1 worden uitgevoerd, vaak op basis van een enkele observatie van normaal kaartgebruik. De aanval is daarmee veel krachtiger dan de tijdaanval.

Bescherming tegen stroomaanval. De software moet zo veel mogelijk constructies vermijden waar uitvoering van instructies afhankelijk is van sleutelbits. Helaas is het bij veel instructies zo dat het stroomverbruik afhangt van de waarde van de operanden, en dan is deze bescherming niet genoeg. Het stroomverbruik van het algoritme kan worden gemaskeerd door het toevoegen van enkele stoorcircuits die volgens een random patroon stroom gebruiken. Shamir [Sha00] suggereert het gebruik van een condensatorbuffer tussen de kaartlezer en de processor.

Het AES-algoritme Rijndael gebruikt de sleutel alleen als operand in een XOR-operatie, waardoor de invloed van de sleutel op het stroomgebruik nihil is.

12.2.3 Foutinjectie

De smartcard is tijdens zijn rekenwerk buitengewoon afhankelijk van zijn omgeving. Het actieve deel is zeer klein en verdwijnt geheel in de kaartlezer. Het is mogelijk gebleken, een kaart aan extreme omgevingsomstandigheden bloot te stellen en hiermee de correcte werking van de kaart te verstoren. De kaart geeft na deze *foutinjectie* een incorrecte uitvoer en in sommige gevallen kan hieruit geheime informatie worden afgeleid.

De techniek van foutinjectie. Men kan de kaart tijdens een berekening blootstellen aan een zeer hoge of juist zeer lage temperatuur, aan straling (röntgen of ultraviolet) of aan mechanische krachten. Als de kaart een extern kloksignaal nodig heeft kan men de klok in de kaartlezer te snel, te langzaam of onregelmatig laten lopen. Om deze reden hebben veel kaarten tegenwoordig een eigen klok.

De meest succesvolle aanvallen maken gebruik van fluctuaties in de voedingsspanning, vaak met een tijdsduur korter dan een schakeltijd van de gebruikte logica. De spanning en tijdsduur van onderbreking ervan zijn namelijk zeer nauwkeurig te doseren en, in combinatie met een stroommeting, kan een spanningsonderbreking op een precies gekozen moment worden geplaatst. In alle gevallen is het resultaat dat een of meerdere instructies niet of onjuist worden uitgevoerd.

Aanvallen op RSA en FFS. Een smartcard die RSA-handtekeningen berekent met de Chinese reststelling is zeer eenvoudig aan te vallen; zie Boneh e.a. [BDL97] en kader 6.6. Deze aanval vereist vrijwel geen kennis van de implementatie omdat een willekeurige verstoring van de kaartwerking met grote kans succes heeft.

Minder bekend is de aanval, eveneens van Boneh, DeMillo en Lipton [BDL97], op het identificatieprotocol van Feige, Fiat en Shamir (protocol 8.5). Terwijl de kaart wacht op een challenge bewerkt Oscars kaartlezer de kaart om in de opgeslagen r één bit te wijzigen (opgave 12.1); de nieuwe waarde is $r' = r + e$. (Omdat r in het RAM-geheugen staat en a in het ROM is r veel gevoeliger voor storingen in de voedingsspanning.) Oscar geeft challenge $c = 1$ en ontvangt de waarde $y = r' \cdot a$, waaruit $r'^2 = y^2/sb$ wordt berekend.

Als Oscar de waarde e van de geïnduceerde fout weet of raadt kan hij r oplossen. Immers, $r^2 = s$ is bekend, en $r'^2 - r^2 = 2re + e^2$ daarmee ook, en we vinden $r = (r'^2 - r^2 - e^2)/(2e)$. Oscar vindt hiermee ook r' en dus het geheim $a = y/r'$. Dat de berekening juist is controleert hij met de test $a^2 = b$.

Helaas kent Oscar e niet, maar als de storing in de kaart inderdaad tot een éénbits-fout in r heeft geleid, is $e = \pm 2^i$, voor een i kleiner dan k (de lengte van de getallen). Oscar hoeft dus maar $2k$ waarden van e te proberen voor hij weet of de aanval is geslaagd. De aanval kan mislukken doordat geen of juiste meerdere bits in r zijn gewijzigd.

Deze aanval kan succesvoller worden naarmate meer implementatiedetails van de kaart bekend zijn, omdat de storing preciezer op de opslag van r wordt gericht.

Geheugen uitlezen door foutinjectie. Fouten kunnen met grote nauwkeurigheid worden veroorzaakt wanneer er ook stroomanalyse wordt gebruikt om te kijken waar de processor precies mee bezig is. Anderson en Kuhn [AK96] rapporteren de volgende aanval. Een processor bevat bijvoorbeeld de volgende routine om a bytes informatie vanaf adres b naar de uitgang te schrijven:

(* b is het beginadres, a het aantal bytes *)
while $a \neq 0$ **do**
    $output(*b)$; (* Schrijf byte b *)
    $b++$; $a--$

Er wordt nu gezocht naar een verstoring, meestal een spanningsfluctuatie, die hetzij de verlaging van a, danwel de test op a verstoort. Door deze verstoring iedere luscycle te herhalen stuurt de processor niet slechts a bytes, maar de gehele inhoud van het geheugen vanaf byte b naar de uitvoer.

Beveiliging tegen foutinjectie. Op algoritmisch niveau kan een oplossing worden geëvalueerd op bestendigheid tegen foutinjectie; voor RSA zal dit betekenen dat gebruik van de Chinese reststelling minder aantrekkelijk wordt. Boneh, DeMillo en Lipton [BDL97] suggereren bescherming op protocolniveau, namelijk elk resultaat te controleren voor het wordt uitgevoerd. Voor een RSA-handtekening kan dit vrij snel (als een kleine exponent is gebruikt), maar voor andere protocollen is de controle duur of onmogelijk. Op hardwareniveau is controle op de juistheid van gegevens mogelijk met foutdetecterende codes.

De smartcard kan worden voorzien van sensoren om spanning, klokfrequentie en temperatuur te meten. Ook mechanische spanning kan worden bemerkt, met lagen van piëzo-elektrisch kristal. Elk afwijkend gedrag wordt als een aanval gezien en blokkeert de kaart.

12.3 Openen van smartcards

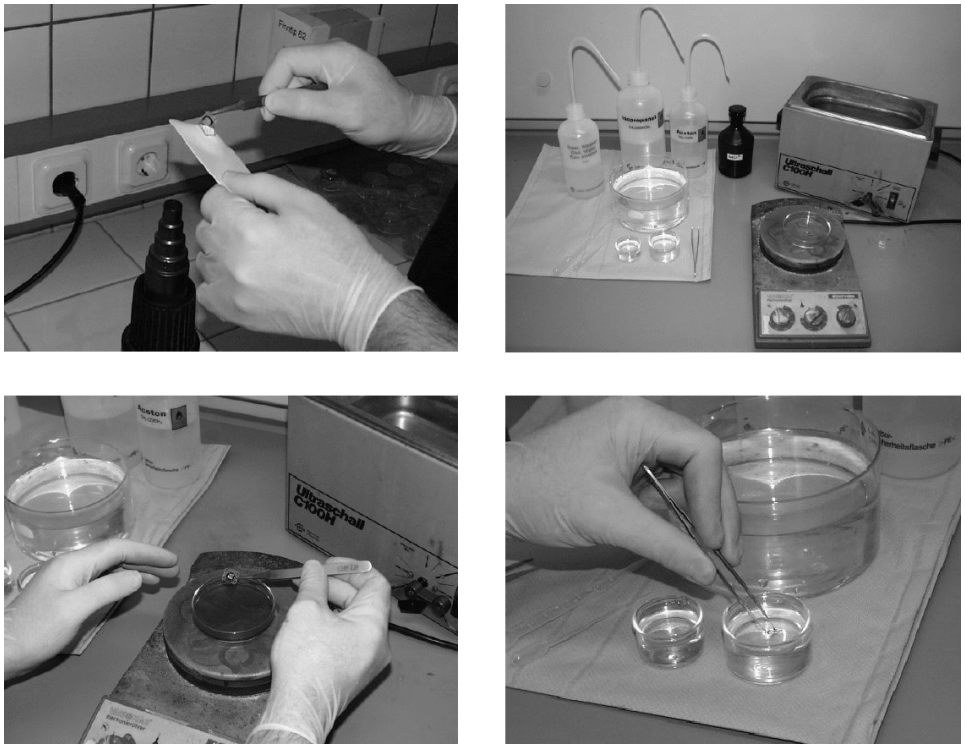
De tot nu toe beschreven aanvallen laten de kaart intact. De aanvaller is er korte of lange tijd mee bezig, maar de kaart kan na de aanval zonder een spoor ervan aan de eigenaar worden teruggegeven. Soms leiden deze externe aanvallen nog niet tot het gewenste resultaat. De bescherming tegen de aanvallen tijdens het gebruik van de sleutel kan te goed zijn, of de kaart is beveiligd met een pincode zodat de aanvaller de kaart niet kan laten werken. Een aanvaller kan dan proberen tot in de kaart door te dringen.

Deze sectie is grotendeels ontleend aan Anderson en Kuhn [AK96].

12.3.1 De processor bereiken

Bij de productie van de smartcard wordt de chip (processor en geheugen) gelijmd aan de metalen contactplaatjes en voor de elektrische verbinding worden er dunne gouddraadjes aan

Kader 12.5: De smartcardprocessor blootleggen



Het openen van de smartcard volgens het proces van Kömmerling en Kuhn [KK99]: de kaart wordt bij opwarmen zacht waardoor de chip eruit kan worden getrokken. Het pakketje met de chip gaat achtereenvolgens in warme salpeterzuur, aceton, gedestilleerd water en isopropanol.
De losse chip kan nu in een voetje met bedrading worden geplaatst en op allerlei manieren verder worden aangevallen. Apparatuur voor stroomanalyse en het fluctueren van de voedingsspanning kunnen worden aangekoppeld. Met een microscoop en microprobes kunnen individuele verbindingen op de chip worden opgezocht en uitgelezen; zie ook kader 13.10.
`http://www.cl.cam.ac.uk/~mgk25/sc99-tamper-slides.pdf`

gesoldeerd. Vervolgens wordt de chip afgedekt met epoxyhars. Epoxy is een klasse chemische verbindingen die zich kenmerkt door kleine ringen van twee koolstof- en een zuurstofatoom. Het contactplaatje met chip wordt dan in de plastic omhulling vastgeplakt.

De chip kan ook weer uit de plastic kaart worden gehaald. Het plastic kan met een mes worden weggesneden waarna het epoxy zichtbaar wordt. Epoxyhars lost op in enkele druppels salpeterzuur (HNO_3), dat echter het silicium van de chip geheel onaangetast laat; zelfs de inhoud van ROM- en EEPROM-geheugen blijven intact. (Waarschuwing: ook vingers, ogen en longen verdwijnen in salpeterzuur als sneeuw voor de zon. In professionele bedrijven wordt deze fase altijd in een zuurkast uitgevoerd!)

De chipcircuits liggen nu bloot, maar de chip werkt nog net zo als tevoren. Via de contacten kan hij op een lezer worden aangesloten.

Op diverse manieren proberen fabrikanten hun chips te beschermen tegen chemische aan-

vallen. Er worden bijvoorbeeld afwisselende lagen van harsen gebruikt die in verschillende stoffen moeten worden opgelost. Het liefst stoffen waarin ook bepaalde elektrische onderdelen van de chip oplossen, zodat de chip na afloop onbruikbaar is.

De chip kan zo worden gemaakt dat hij de aanwezigheid van epoxy of een coating kan bemeten; ontbreekt de laag, dan weigert de chip. Als de processor zelf een batterij heeft, kan er ook een sensor worden ingebouwd die de aanwezigheid van salpeterzuur detecteert en dan het geheugen wist. Ook kunnen er speciale dunne detectiedraden rond de chip worden gelegd; verwijdering van de omgeving onderbreekt de draden of veroorzaakt sluiting waarna de chip de sleutels wist.

12.3.2 Analyse van de chip

Als het chipoppervlak bereikbaar is, zijn er diverse mogelijkheden om waar te nemen wat er precies in de chip gebeurt. Zo kunnen verbindingen worden opgemeten met microprobes, een soort superkleine meetpennen. Voordeel van deze probes is dat er niet alleen signalen gemeten, maar ook kunnen worden ingebracht. Wel is er door de elektrische belasting een verstoring van sommige circuits mogelijk. Ook is het mogelijk te meten zonder elektrisch contact. Zo kan de elektrische uitstraling van een verbinding met gevoelige sensoren worden gemeten. Op een onderdeel kan een speciaal kristal worden aangebracht waarvan de brekingsindex varieert met de elektrische spanning, waarna het signaal met een laser kan worden uitgelezen.

In het begin weet de aanvaller vaak niet waar de gevoelige informatie zich op de chip bevindt. Voor een aanval worden dan meerdere kaarten van hetzelfde type gekocht en in achtereenvolgende analyserende tests komt men meer en meer over de werking van de kaart te weten. Het is zelfs mogelijk door reverse engineering de volledige werking van een chip in kaart te brengen.

12.3.3 Wijzigingen in de chip

De werking van een chip kan worden veranderd door elektrische verbindingen te verbreken of juist aan te brengen. Dit is mogelijk met een ionenstraalmicroscoop; een dergelijk apparaat is duur (miljoenen euros) maar werktijd erop kan worden gehuurd.

Geheimen uitlezen door wijziging. Meestal heeft zo'n wijziging tot doel, een bescherming tegen andere aanvallen op te heffen, of via een nieuwe weg tot de geheime gegevens te geraken. Zo kan een processor worden beschadigd op zo'n manier dat hij alle informatie uit het EEPROM achtereenvolgens gaat lezen. Een enkele probe op de databus kan dan alle geheime sleutels aflezen.

Van DES is bekend dat sleutelbits uit de uitvoer kunnen worden afgeleid als bepaalde delen van het algoritme niet werken (bijvoorbeeld S-boxen of de XOR-operatie).

Gebruik van gemodificeerde kaarten. De wijziging kan ook tot doel hebben de kaart in gewijzigde versie weer in gebruik te nemen, bijvoorbeeld als telefoonkaart die nooit opraakt. Natuurlijk kan de chip opnieuw in epoxyhars en plastic worden verpakt en op het plastic kan desgewenst de oorspronkelijke afbeelding worden aangebracht.

De kaart kan dan, als alles zorgvuldig is gedaan, zonder meer worden gebruikt bij automaten zonder toezicht (telefoons, videodecoders). Gebruik is ook mogelijk bij heel veel plaatsen met toezicht, bijvoorbeeld betaalterminals; winkeliers controleren immers niet het ongeschonden uiterlijk van de aangeboden kaarten. De gebruiker van een gemodificeerde kaart moet er wel op bedacht zijn dat de kaart bij elke transactie statusinformatie naar de kaartuitgever stuurt.

De beheerders van het systeem kunnen dus bemerken dat er met deze kaart iets aan de hand is en als de kaart op naam is verkocht, kan men de kaarthouder hierop aanspreken.

Er bestaan technieken die juist het onzichtbaar knoeien met de kaarten bemoeilijken. De kaart wordt bijvoorbeeld verpakt in een hard materiaal waarop mechanische druk wordt uitgeoefend. Hierdoor ontstaat een patroon van haarscheurtjes dat voor deze verpakking net zo karakteristiek is als een vingerafdruk; een scan ervan kan in de chip zelf of in de kaartlezer worden opgeslagen. Na het openen van de kaart kan hij wel opnieuw in hard materiaal worden verpakt, maar het is onmogelijk om in de nieuwe verpakking hetzelfde patroon van breuken te laten ontstaan.

Samenvatting en conclusies

Smartcards zijn, behalve een zeer handig hulpmiddel voor gebruikers van systemen, ook een belangrijke techniek om de veiligheid van cryptografische systemen te vergroten. Met een smartcard kan een gebruiker de beschikking krijgen over sleutels en data, maar alleen voor de doelen die de systeemontwerper heeft voorzien. Gebruik van de sleutels in andere algoritmen of het kopiëren van sleutels en data wordt onmogelijk geacht.

De bescherming die een smartcard biedt tegen uitlekken of ongeoorloofd gebruik van data is groot, maar niet absoluut. Een kaartlezer kan worden gemanipuleerd om bijvoorbeeld een stroomaanval uit te voeren: uit het stroomverbruik van de kaart wordt geheime data afgeleid. Ook is het in veel gevallen mogelijk de kaart te openen en de binnenkant ervan uit te lezen of te wijzigen.

Opgaven bij hoofdstuk 12

Opgave 12.1 *Een spanningsdip wijzigt een enkele bit in het RAM-geheugen met kans α. Welke α geeft de hoogste kans op een éénbitsfout in een k-bitsregister?*

Opgave 12.2 *Toon aan dat Schnorrs identificatieprotocol kwetsbaar is tegen foutinjectie. (***Hint:** *als in r de fout $e = \pm 2^i$ is opgetreden onthult dit de waarde van de i^{e} bit van r; zie Boneh e.a. [BDL97].)*

Hoofdstuk 13

Elektronisch geld

Dit hoofdstuk over elektronisch geld behandelt voornamelijk systemen, waarin geldswaarde is gekoppeld aan uitwisselbare elektronische objecten. Bij veel betaalsystemen is er geen sprake van geldswaarde die in het systeem aanwezig is, maar slechts van het faciliteren van directe overschrijvingen tussen bankrekeningen. Dergelijke systemen worden slechts kort genoemd; de belangrijkste technische uitdagingen ervan liggen op het gebied van communicatiebeveiliging en identificatie en beide gebieden zijn reeds behandeld.

In dit boek spreken we pas van 'elektronisch geld' als er, zoals gezegd, waarde in het systeem aanwezig is en tussen betaler en ontvanger kan worden uitgewisseld. Als inleiding op de behandeling van dergelijke systemen geeft sectie 13.1 een overzicht van wat geld eigenlijk is, wat het niet is, en welke eigenschappen voor elektronisch geld van belang zijn.

Een elektronische munt heeft doorgaans de vorm van een speciaal geldswaardig certificaat, dus een door de bank ondertekende verklaring van waarde. Betaling gebeurt door het overdragen van zo'n certificaat, en de ontvanger kan de handtekening controleren om zich van de echtheid van het geld te overtuigen. Een dergelijk systeem werd ontwikkeld door het Amsterdamse bedrijf DigiCash en is in zijn eenvoudigste vorm ook in gebruik (sectie 13.2).

De geschetste gang van zaken is echter te simpel: er moeten aanvullende maatregelen worden genomen om te voorkomen dat men meerdere malen met hetzelfde geld betaalt. Verder kan men voor de betalingen aanvullende wensen hebben, zoals de mogelijkheid anoniem betalingen te doen, eisen op de efficiëntie van het systeem of de mate van interactie die tussen de partijen nodig is bij een betaling. Daarom zullen we, voortbouwend op het model van de DigiCash-betalingen, naar complexere systemen kijken.

De werkelijkheid voor wat betreft grootschalig ingevoerde systemen is echter heel anders. Bij het ontwerp van het chipknipsysteem heeft men gekozen voor een vorm van waardeopslag die simpel en goedkoop te implementeren is, gebruikmakend van smartcards waarvan op de integriteit te vertrouwen is. We zullen de globale werking van dit systeem, en de belangrijkste risico's ervan bekijken in sectie 13.3.

13.1 Geld en eisen op elektronisch geld

Wat is geld? Een gangbare definitie is de volgende van Korteweg en Keesing [KK74]:

Definitie 13.1 *Geld omvat alle activa die dienst doen als ruilmiddel, rekeneenheid en oppotmiddel, en die in deze functie algemeen worden aanvaard.*

Kader 13.1: Geslagen geld in de bijbel.

In de bijbelse tijd werd er tussen volksgenoten of familie nog veel in natura betaald; ook gold vee als betaalmiddel, zo was een geitebokje de normale beloning voor de diensten van een prostituee (Genesis 38:17). Handel met vreemdelingen werd met metaal afgerekend: Jozef werd voor twintig zilverstukken verkocht (Genesis 37:28), en Abraham betaalde een graf voor Sara met zilver (Genesis 23:16). De geldeenheid *sikkel* was een gewichtsmaat (14,5 gram): de hoeveelheid zilver moest voor elke betaling worden afgewogen.
Gemunt geld maakte wegen overbodig en dit was beschikbaar vanaf de Perzische tijd (ca. 900 voor Christus). Om bedrog met de hoeveelheid gebruikt metaal tegen te gaan was het muntrecht, dus het vervaardigen van de munten, voorbehouden aan de staat. Wel kon in principe iedereen zijn metaal door deze instelling in munten laten omzetten. De munten droegen een teken van de vorst onder wiens verantwoordelijkheid ze waren geslagen.

Tot het geld worden in onze moderne maatschappij munten en bankbiljetten gerekend (stoffelijk of *chartaal* geld), maar ook tegoeden op betaalrekeningen (onstoffelijk of *giraal* geld). Hoe de activa van deze soorten zijn ontstaan wordt beschreven in sectie 13.1.1. In sectie 13.1.2 worden de belangrijkste begrippen rond en eigenschappen van elektronisch geld uiteen gezet.

13.1.1 Van zouttablet tot euro

In een zeer ver verleden hadden de mensen geen behoefte aan geld of ruilen omdat iedereen voedsel en kleding produceerde voor de eigen behoefte, meestal gemeenschappelijk. Ongeveer vijfduizend jaar geleden was er in het economisch verkeer al sprake van specialisatie: de bakker bakte brood, de visser ving vis en de kapper knipte haar. Toch kon men het lange tijd nog stellen zonder geld, omdat de producten en diensten werden uitgewisseld in *directe ruilhandel*. De kapper knipte voor een vis of een brood, de bakker leverde brood voor een vis, en iedereen at brood met vis, met geknipt haar, zonder dat er geld nodig was.

Directe ruilhandel heeft beperkingen: de visser wil zich laten knippen maar niet de kapper, maar de bakker heeft honger; de vis is op maandag gevangen, maar het haar is pas op vrijdag lang. Daarom deden rond 2500 voor Christus in Egypte kleine waardevolle objectjes, namelijk ringetjes of schijfjes van metaal, hun intrede in het economisch verkeer. Een vis wordt geruild tegen ringetjes, niet omdat de visser graag ringetjes heeft, maar omdat de ringetjes (eventueel dagen later) kunnen worden geruild tegen een knipbeurt.

De ringetjes zijn gemaakt van een kostbaar materiaal (*commoditeit*) zoals koper of goud, en zijn daarom waardevol: we spreken van *commoditeitsgeld*. Omdat dit geld van zichzelf, dus intrinsiek, waarde heeft, wordt het ook *intrinsiek* geld genoemd. Zo deed het geld zijn intrede als ruilmiddel, maar na enkele eeuwen vervulde het de drie functies in het economische leven die in definitie 13.1 zijn genoemd:

1. **Betaalmiddel.** Ruilobject was de oorspronkelijke rol van het geld.
2. **Waardestandaard.** Behalve dat je metaal kunt accepteren als betaling voor een brood in plaats van een vis, kun je ook de prijs van het brood uitdrukken in metaal in plaats van in vissen.
3. **Waardeopslag.** Anders dan een gevangen vis of gebakken brood kunnen metaalstukjes worden bewaard zodat consumptie (het eten van een brood) op een andere tijd kan gebeuren dan productie (het vangen van een vis).

Als voorbeeld van commoditeit werd al metaal genoemd, en zowel koper, zilver als goud kwamen in de oudheid al voor als betaalmiddel. Maar er werd ook wel betaald met schelpen of zout (het woord *salaris* komt van het Latijnse *sel*, zout). De gebruikte commoditeit moest aan enkele eisen voldoen:

1. **Compact.** Het moet gemakkelijk zijn een behoorlijke waarde ervan mee te nemen.
2. **Houdbaar.** Het materiaal mag niet bederven.
3. **Waardevast.** De waarde van het materiaal mag niet plotseling sterk dalen.
4. **Verdeelbaar.** Het moet mogelijk zijn hoeveelheden ervan te meten en te verdelen.

Metaal voldoet aan al deze eisen en is daarom ook het meest als commoditeit gebruikt, maar in Virginia kon bijvoorbeeld lange tijd met tabak worden betaald. De gebruikte metalen zoals zilver, goud en brons waren in elke vorm even kostbaar, maar bij het betalen had men steeds een weegschaal nodig om de juiste hoeveelheid af te wegen (kader 13.1). Omdat tellen gemakkelijker is dan wegen ontstond het *munten*: hierbij werden plakjes met een precies afgemeten hoeveelheid materiaal vervaardigd. De maker ervan stond ervoor in dat de juiste hoeveelheid metaal was gebruikt. In het begin waren de zijkanten glad waardoor snode lieden de munt konden *snijden*: er werd wat metaal afgeschraapt waarvan weer extra nieuwe munten konden worden gemaakt. Deze praktijk was natuurlijk illegaal en kon later worden tegengegaan door een randschrift (dat de oudere lezer zich misschien nog herinnert van de gulden) of karteling (kwartje).

Het commoditeitsgeld had een monopolie tot het eind van de Middeleeuwen. Door de toegenomen welvaart en het opzetten van grote ondernemingen gingen mensen (sommige althans) over steeds grotere hoeveelheden geld beschikken en uiteindelijk werd het gesleep met goud te omslachtig. Ook was er natuurlijk altijd het risico van diefstal. Zo verschenen aan het eind van de Middeleeuwen de *banken* als instituten waar men zijn geld in bewaring kon geven. Als bewijs kreeg men een papier waarop de afgegeven hoeveelheid geld stond vermeld, de *goudbon*.

Wilde Alice iets betalen aan Bob, dan wisselde zij haar goudbon in voor geld en gaf het geld aan Bob, die het natuurlijk meteen bij de bank in bewaring gaf tegen een een nieuwe bon. Dat kon natuurlijk efficiënter; de eer voor deze stap gaat naar China waar men rond 1300 inzag dat Alice net zo goed haar goudbon direct aan Bob kon geven. Door de goudbonnen overdraagbaar te maken ontstonden de eerste bankbiljetten, die ter betaling werden geaccepteerd in plaats van 'echt' geld.

Bankbiljetten hebben geen intrinsieke waarde, maar onleenden hun waarde aan de inwisselbaarheid voor goud (of andere commoditeiten). Anders gezegd, ze *representeren* een intrinsieke waarde en daarom werden bankbiljetten wel *representatief geld* genoemd. De formele inwisselbaarheid was de *dekking* van het bankbiljet, en de meeste landen hadden dekking in termen van goud, ofwel *gouddekking* of *gouden standaard* genaamd.

Kader 13.2: Nederland verlaat den Gouden Standaard.

De economische crisis van de jaren twintig van de vorige eeuw werd bestreden door het loslaten van de formele gouddekking van bankbiljetten, waardoor meer geld in omloop kon worden gebracht. De meeste landen namen deze maatregel rond 1930, maar Nederland, met zijn enorme goudvoorraad, handhaafde de Gouden Standaard. Het gevolg was dat de Nederlandse gulden een hoge wisselkoers had, waardoor de handel met het buitenland verstoord raakte en er nauwelijks sprake was van economisch herstel. In september 1936 werd de Nederlandse gulden fiduciair, devalueerde prompt zo'n twintig procent en hierna ging het Nederland een paar jaar economisch beter.
Philips reageerde op het thema met deze reclameposter voor de toen bekende goudkleurige radiolampen. Ook bij Philips was het goud vergankelijk, want vanaf 1938 werden de lampen uit de gouden reeks niet meer toegepast.

De gouddekking hield het uit tot in de twintigste eeuw. De inwisselbaarheid gaf de bankbiljetten hun waarde, maar er werd in de praktijk nauwelijks gebruik van gemaakt; de inwisselbaarheid werd beperkt en zelfs afgeschaft. Er was bij wet geregeld dat iedereen verplicht is bankbiljetten te accepteren ter voldoening van een betaalverplichting. In Nederland worden bankbiljetten sinds 1814 uitgegeven door De Nederlandsche Bank en sinds 1904 zijn ze *wettig betaalmiddel*. De bankbiljetten hebben dus noch intrinsieke, noch representatieve waarde, maar ontlenen hun waarde aan de zekerheid dat ze ook door een volgende partij als betaling zullen worden geaccepteerd. De waarde van de bankbiljetten is daarom volledig gebaseerd op het vertrouwen dat ze iets waard zijn en daarom worden ze wel als *fiduciair* geld aangeduid.

Ook muntgeld werd in toenemende mate fiduciair, omdat de waarde van het verwerkte metaal kleiner werd dan de nominale waarde van de munt. Dat betekent dat het munten van metaal een waardevermeerdering inhoudt, en het recht munten te laten vervaardigen werd voorbehouden aan de staat. Ook werden deze zogenaamde *onvolwaardige* munten een wettig betaalmiddel voor kleine bedragen, omdat zonder wettelijke plicht de bereidheid om ze te accepteren natuurlijk klein zou zijn.

Een volgende verandering in het gebruik van geld vond plaats in de jaren zestig. Steeds minder had men geld 'in de hand' in de vorm van munten en biljetten, maar steeds meer 'op de bank' in de vorm van een tegoed op de bankrekening. Wilde Alice iets betalen aan Bob, dan nam zij geld op van haar rekening (debitering van haar saldo), en gaf het aan Bob, die het natuurlijk meteen op zijn bankrekening zette (creditering van zijn saldo). Ook dat kon natuurlijk efficiënter! Er kan direct via de bank worden betaald, namelijk door het geven van een opdracht aan de bank om het bedrag van Alice' tegoed af te schrijven en het aan Bobs tegoed toe te voegen. Deze betaalmogelijkheid bestond al vanaf 1609, toen de Amsterdamse Wisselbank, in plaats van het geven van een reçu voor afgegeven geld een boekhouding ging bijhouden. Het contante geld (munten en biljetten) heet *chartaal*, het tegoed op direct opeisbare bankrekeningen wordt tot de geldhoeveelheid gerekend en heet *giraal*. Een betaling met giraal geld kan plaatsvinden door middel van een overschrijving of cheque. Als de rekeningen bij verschillende banken worden gehouden moet er ook tussen deze banken onderling worden verrekend; dit heet *clearing* en wordt door De Nederlandsche Bank gedaan,

Kader 13.3: Chartaal en giraal geld in Nederland

In de twintigste eeuw verdubbelde de hoeveelheid geld in Nederland ongeveer elk decennium zoals blijkt uit bijgaand overzicht (bron: CBS StatLine, eenheid: miljoen gulden). Tot 1960 waren de chartale en girale geldhoeveelheden in Nederland ongeveer gelijk. Het gebruik van giraal geld bleef beperkt tot de gegoede burgerij: in 1960 kwam de Nederlandse arbeider op vrijdag thuis met z'n loonzakje. In 1970 kenden wij geen arbeiders meer maar werknemers en die kregen hun salaris giraal gestort.

Jaar	Totaal	Chartaal	Giraal
1900	331	265	66
1910	488	321	167
1920	2 458	1 120	1 338
1930	1 937	895	1 042
1940	2 994	1 695	1 299
1950	6 801	3 014	3 787
1960	11 303	5 087	6 216
1970	25 950	9 951	15 999
1980	65 585	21 996	43 589
1990	123 934	36 473	87 461
1998	256 003	37 550	218 453

In de dertig jaar daarna nam de hoeveelheid chartaal geld nauwelijks toe, en rond 2000 bedroeg die een zesde deel van de hoeveelheid giraal geld. Chartaal geld is verworden tot een randverschijnsel in de Nederlandse economie, een hulpmiddel voor het doen van kleine betalingen.

waar elke bank een rekening heeft.

Uiteraard ontstond de behoefte om ook in winkels direct met giraal geld af te kunnen rekenen en hieruit ontstonden in 1967 de betaalcheques. In hetzelfde jaar werd de BankGiroCentrale (kader 13.8) opgericht om de clearing te ondersteunen en te versnellen. In Bobs winkel vult Alice een betaalcheque in, die door Bob wordt opgestuurd naar de bank en tot overschrijving van saldo leidt. Er werd over deze cheques een beslissing genomen met belangrijke consequenties, namelijk dat ze *gegarandeerd* waren (tot een bepaald bedrag). Dit betekent dat Bob, als hij de cheque volgens de regels accepteert (pasje en handtekening controleert), *zeker is* dat hij het geld op zijn rekening ontvangt, *ook als Alice het geld niet op haar rekening heeft!* In dit laatste geval komt Alice rood te staan. De bank laat Alice hiervoor flink boeten, maar loopt ook een risico omdat het wel voorkwam dat rekeninghouders in het rood kwamen en het tekort nooit meer aanvulden. In de Nederlandse praktijk waren deze risico's acceptabel zodat Nederland het grote voordeel genoot van op veilige betaalcheques gebaseerde waardeoverdracht.

In Amerika daarentegen is een gegarandeerde betaalcheque volstrekt ondenkbaar. Er is geen burgerlijke stand of andere sluitende persoonsregistratie, waardoor het erg gemakkelijk is een rekening te openen, die met cheques in het rood te brengen, en te verdwijnen. Het uitschrijven van ongedekte cheques werd wel zwaar gestraft, maar de winkelier zat met de strop. Je kon in Amerika wel een cheque uitschrijven, maar die werd dan eerst door de winkelier aan de bank ter controle aangeboden en je kon dan een paar dagen later in de winkel je spullen ophalen. Voor de meeste aankopen is dit te omslachtig en daarom ontstond in Amerika de *creditcard* als betaalmiddel. De uitgevende instantie betaalt aan de winkelier (er is dus sprake van garantie) en verhaalt het bedrag later op de consument. De creditcardmaatschappij draagt het risico, maar schiet het bedrag ook enkele weken voor. Hiervoor worden stevige kosten in rekening gebracht (een heffing van vier tot zes procent) maar die worden voor de consument onzichtbaar gehouden.

Net zomin als van de inwisselbaarheid van representatieve bankbiljetten wordt van de opeisbaarheid van banksaldi veel gebruik gemaakt. In toenemende mate wordt het girale geld als echt geld ervaren en contant geld als een afgeleide vorm ervan, die door storting weer

Kader 13.4: Credit-debetbetalingen: Rabo Direct Betalen

Met Rabo Direct Betalen kun je tijdens het bezoek aan een internetwinkel een betaalopdracht via de bankrekening laten uitvoeren. Wanneer je opdracht tot aankoop geeft, verschijnt een venster waar gegevens over de betaalrekening en de aankoop worden ingevuld. De gegevens worden ondertekend met een Digipas (vergelijkbaar met de Identikey uit kader 7.2). De opdracht wordt verstuurd naar de bank (dus *niet* naar de winkel) en uitgevoerd. De winkel ontvangt bericht dat haar rekening is gecrediteerd, maar zonder vermelding van de identiteit van de betaler.
Klant en winkel moeten een Rabo-betaalrekening hebben, daarnaast moet de klant een Digipas aanvragen en bij de winkel is software nodig. Rabo Direct Betalen is veilig, eenvoudig en snel. Het is niet anoniem en de noodzaak van interactie met de bank maakt het afhankelijk van de goede werking van infrastructuur.

in banksaldo kan worden omgezet. Anders gezegd, het contante geld is representatief voor saldo en is niet meer dan een hulpmiddel om saldo van een rekeninghouder naar een andere te verplaatsen.

13.1.2 Betaalmodellen

Voor de bespreking van betaalsystemen zullen we het girale geld als uitgangspunt nemen. Betaling van Alice aan Bob betekent dan dat het saldo van Alice moet worden verlaagd (*debitering*) en het saldo van Bob verhoogd (*creditering*). Aan de hand van de relatie tussen deze twee deelopdrachten kunnen we drie hoofdvormen van betaling onderscheiden:

1. **Debetbetaling.** Deze betaling gebruikt (chartaal of elektronisch) geld als overdraagbaar medium. Bij opname ervan wordt gedebiteerd, en het banksaldo wordt omgezet in overdraagbare objecten, die ter betaling worden gegeven. De ontvanger kan de objecten afstorten en wordt dan gecrediteerd.

   Alleen bij debetbetalingen is anonimiteit mogelijk, mits de geldobjecten niet van elkaar te onderscheiden zijn. De bank ziet dan wel dat Alice betaalt, maar niet aan wie. De bank ziet wel dat Bob ontvangt, maar niet van wie.

2. **Credit-debetbetaling.** De overschrijving wordt als een geheel aan de bank aangeboden. Dit kan in de vorm van een schriftelijke overschrijving of machtiging, maar ook met een betaalcheque, een pinbetaling, of een internettransactie. Debitering en creditering vinden ongeveer tegelijk plaats en de bank ziet aan wie Alice betaalt en van wie Bob ontvangt.

3. **Creditbetalingen.** De ontvanger wordt snel na de betaalhandeling gecrediteerd, maar niet direct ten laste van de betaler. Een derde partij schiet het geld voor en verhaalt het op de betaler. Creditering vindt plaats *voor* debitering.

Alleen in het eerste model is geldswaarde daadwerkelijk in het betaalsysteem aanwezig en spreken we van elektronisch geld. Zowel credit-debetbetalingen als creditbetalingen komen in elektronische vorm voor, bijvoorbeeld voor betalingen over het internet (kader 13.4). Het belangrijkste probleem dat bij de implementatie ervan moet worden opgelost is identificatie. Alle partijen moeten er zeker van zijn dat degene die een betaling accordeert of ervan profiteert

dezelfde is als de houder van de rekening. De betaler moet er zeker van zijn dat degene aan wie wordt betaald inderdaad degene is voor wie de betaling is bedoeld.

Het gebruik van creditcards op het internet heeft een slechte naam op het gebied van veiligheid, en de enige reden hiervan is dat de identificatie bij creditcards erg slecht is. Vaak wordt alleen al het opsturen van een creditcardnummer als identificatie bij een betaling geaccepteerd; het opgestuurde nummer kan worden onderschept, of door de winkelier worden 'hergebruikt'. De veiligheid kan worden verbeterd door de communicatie te versleutelen, bijvoorbeeld met het *Secure Socket Layer* (SSL, sectie 10.3) protocol dat in veel webbrowsers is ingebouwd. Een nog grotere bescherming biedt *Secure Electronic Transactions* (SET). Hierbij wordt de betaalopdracht direct naar de creditcardmaatschappij gestuurd en de winkelier ontvangt alleen een betalingsbevestiging. De winkelier krijgt de creditcardgegevens niet te zien en kan die dus ook niet misbruiken. Behalve de veiligheid is bij SET ook de privacy van de klant beter beschermd.

Dit hoofdstuk gaat verder over betalingen met behulp van overdraagbare waardedragende objecten (model 1), en alleen over die systemen waarbij deze objecten zelf uit informatie bestaan. De overdraagbare objecten, samen met de protocollen die de overdracht regelen, noemen we *elektronisch geld*. Elektronisch geld kan een waarde hebben als normaal geld (euro's), maar er zijn ook veel systemen in omloop waarin waardepunten (AirMiles) of telefoontikken elektronisch zijn. Van de drie functies van commoditeitsgeld vervult elektronisch geld er maar één:

1. **Betaalmiddel.** Elektronisch geld is een hulpmiddel bij waardeoverdracht.

2. **Geen waardestandaard.** We kennen wel de euro als waarde-eenheid maar die is niet van elektronisch geld afgeleid.

3. **Geen waardeopslag.** Wie waarde wil bewaren zet het liever op de bank zodat er rente wordt ontvangen. Niemand riskeert graag een diskcrash als er een miljoen elektronische euro's in de pc zitten!

In het vervolg van deze sectie wordt een aantal eisen en aspecten van elektronisch geld belicht. Bij het bekijken van de protocollen onderscheiden we drie deelnemende partijen: (1) de betaler (of klant, Alice); (2) de ontvanger (of winkelier, Bob); (3) de bank. We zullen voor de beschrijving uitgaan van de situatie dat er maar één bank is waar Alice en Bob beide een rekening hebben.

Wettig betaalmiddel? Met contant geld kan altijd worden betaald omdat het aannemen ervan een wettelijke verplichting is; contant geld is wettelijk dan ook een *overdraagbaar instrument*. Dit betekent dat het bezit ervan in principe de mogelijkheid impliceert het rechtens over te dragen, zoals wordt verduidelijkt aan de hand van een voorbeeld.

1. Anneke is contractueel verplicht een koe aan Bert te leveren maar heeft er geen. Ze leent een koe van Carla en geeft hem aan Bert, die er erg blij mee is. Carla vraagt haar koe terug, maar Anneke heeft die niet meer en Carla herkent haar koe in Berts wei. Omdat de koe op het moment van de overdracht aan Bert niet van Anneke was, is de eigendomsoverdracht onwettig geweest. Bert is er dus geen eigenaar van geworden en Carla krijgt de koe van Bert terug, waarna Bert uiteraard alsnog van Anneke een nieuwe koe kan eisen.

2. Anneke is contractueel verplicht 1000 euro aan Bert te betalen maar heeft het geld niet. Ze leent geld van Carla en geeft het aan Bert, die er erg blij mee is. Carla vraagt haar

geld terug, maar Anneke is blut en Carla herkent haar biljet in Berts portemonnee. Helaas, hij heeft het geld van Anneke rechtens ontvangen en heeft met Carla niets te maken. Carla krijgt het geld *niet* van Bert en houdt dus haar vordering op Anneke, die het recht heeft haar financiële verplichting te voldoen met een ander biljet dan ze van Carla heeft geleend.

Er is nog geen status van elektronisch geld als wettig betaalmiddel, wel kan elektronisch geld een overdraagbaar instrument zijn. Uiteraard is het in het belang van winkeliers om elektronische betalingen te accepteren, omdat de klant anders bij de concurrent zaken gaat doen.

Banken als geldscheppende instanties. Zoals al is vermeld, worden munten en bankbiljetten (chartaal geld) en opeisbare tegoeden (giraal geld) tot de geldhoeveelheid gerekend. Munten worden vervaardigd in opdracht van de staat, die daarmee de geldhoeveelheid kan vermeerderen en dus geld scheppen. Bankbiljetten worden gemaakt in opdracht van de (Europese) Centrale Bank, die daarmee een mogelijkheid tot geldschepping heeft en de biljetten ook weer kan innemen. Giraal geld kan door alle banken worden geschapen door het principe van *wederzijdse schuldaanvaarding*. De bank geeft dan een direct opeisbaar krediet (giraal geld dus) in ruil voor een langlopende (dus *niet* direct opeisbare) vordering. Het aflossen van het krediet doet de hoeveelheid giraal geld weer slinken. Girale geldschepping is aan strenge regels onderworpen en niet elke instelling krijgt hier zomaar toestemming voor.

Elektronisch geld in het chipknipsysteem wordt toegevoegd en ingenomen door Interpay. Deze organisatie heeft niet de status van bank, en het saldo op chipknips wordt vooralsnog niet tot de geldhoeveelheid gerekend. In de zin van de gangbare economische theorieën wordt elektronisch geld dus niet beschouwd als geld, maar hoogstens als een overdraagbare vordering (op een instelling, Interpay).

De systemen voor elektronisch geld kunnen in een beperkte omgeving of toepassing worden ingezet, bijvoorbeeld voor de registratie van spaarpunten of voor een telefoon- of kopieerkaart. In dat geval is gebruik alleen mogelijk voor een specifiek doel, en de vordering is vaak niet eens opeisbaar.

Veiligheid voor alle partijen. Voor geen van de partijen mag de mogelijkheid bestaan zich te verrijken door van het protocol af te wijken. Evenmin mag het mogelijk zijn dat twee van de partijen door samen te werken de derde kunnen benadelen. Ook mag het niet mogelijk zijn dat een buitenstaander (Oscar) zich met de interactie bemoeit en hieruit voordeel haalt, of dat elektronische munten worden vervalst.

Bij elektronisch geld moet steeds gewaakt worden voor het probleem van *geldverdubbeling*. Digitale informatie kan altijd worden gekopieerd en kopieën zijn niet van originelen te onderscheiden. Daardoor kan Alice proberen met een elektronische munt tweemaal te betalen, of Bob kan proberen hem tweemaal af te storten. Alle geldsystemen moeten dit probleem op de een of andere manier aanpakken. Dat kan gebeuren door geldverdubbeling onmogelijk te maken (het betaalprotocol kan niet voor een tweede maal doorlopen worden) of door opsporing achteraf.

Anonimiteit en privacy. Bij veel economische transacties kennen de partijen elkaar wederzijds, maar bij kleine aankopen in winkels is dat niet zo. Alice kan iets kopen zonder dat Bob weet aan wie hij verkoopt, en zonder dat de bank weet waar zij haar geld uitgeeft. Bij elektronische aankopen, bijvoorbeeld via internet, is anoniem kopen soms eenvoudigweg niet

mogelijk omdat de gekochte goederen moeten worden afgeleverd. Maar bij betalingen voor informatie of het bezoek aan een website is het niet nodig dat de winkelier de identiteit van de klant kent. Een betalingssysteem heet *anoniem* (of *privacybeschermend*) als de winkelier en de bank niet aan de betaling kunnen zien wie er betaalt.

De meningen over de noodzaak van anonimiteit lopen uiteen. Voor sommige ondernemers is anonimiteit belangrijk omdat zij producten leveren (seks, gokken, drugs) waar de klant niet graag voor uitkomt. Andere ondernemers willen juist graag weten wie er koopt om daarna gericht aan marketing te kunnen doen. Ook is in sommige bedrijven prijsdifferentiatie belangrijk en daarvoor is het nodig de klant te kennen.

Voor de klant heeft anonimiteit geen nadelen; als hij wil kan hij alsnog zijn identiteit bekend maken aan de winkelier, buiten de betaling om. In een niet-anoniem systeem kunnen alle betalingen van de klant worden opgeslagen en op een onverwacht moment *tegen hem* worden gebruikt.

1. *In de rechtbank:* Alice blijkt het pistool te hebben afgerekend waar haar buurman mee is neergeschoten. Alice blijkt in een café pils te hebben afgerekend vlak voor het veroorzaken van een verkeersongeval.

2. *Bij de verzekering:* Alice krijgt geen levensverzekering omdat ze vaak in de tabakswinkel afrekent.

3. *Thuis:* Alice heeft een verrassing voor haar echtgenoot gekocht... dacht ze... want haar man werkt bij de bank.

De banken hebben geen bezwaar tegen of belang bij anonimiteit, maar vinden anonimiteit in ieder geval ondergeschikt aan kosten en veiligheid van het systeem. In bancaire kringen spreekt men al van een anoniem systeem wanneer de identiteit van de betaler wordt afgeschermd van de ontvanger. In Direct Betalen (kader 13.4) ziet de bank alle transacties, en de anonimiteit voor de klant bestaat uit een afspraak van de bank (*privacy convenant*) deze gegevens niet naar buiten te brengen. De overheid ziet daar wel wat in: anonimiteit van de betaler tegenover de winkelier, maar uiteraard kan er op gerechtelijk bevel wel inzicht in Alice' betaalpatroon worden verkregen!

De partij die het grootste belang bij anonimiteit heeft (de klant) heeft ook de zwakste stem in de keuze voor een bepaald betaalsysteem. De banken, de meeste ondernemingen en de overheid willen de ontwikkelingen sturen in de richting van niet-anonieme systemen, met als argumenten: (1) hogere veiligheid en lagere kosten; (2) noodzaak van wetshandhaving; (3) gebrek aan alternatieven. Voor internetbetalingen vanuit de pc is zo'n alternatief er wel: DigiCash, waarvan de anonieme variant in sectie 13.2.2 wordt besproken. Ook elektronische portemonnees zoals de chipknip zijn in een anonieme versie te maken; bij de prepaid-chipknip zijn betalingen wel tot de kaart te traceren, maar niet aan een persoon te koppelen.

Online en offline systemen. Bij een betaalhandeling met geld zijn steeds de klant en de winkelier betrokken. We onderscheiden betalingen naar *offline* en *online* al naar gelang er bij de betaling ook interactie met de bank nodig is. Bij een pinbetaling is dit wel zo, de betaalterminal maakt tijdens de betaling verbinding met de bank, en pinbetalingen zijn dus online. Een chipknipbetaling is offline, want bij het chippen wordt geld van de chipknip naar de terminal gestuurd zonder dat de bank er aan te pas komt. Rabo Direct Betalen (kader 13.4) is online.

Online betalingen kunnen soms niet plaatsvinden door een storing in de verbinding of in de computer van de bank. Ze kosten ook meer tijd wegens de extra interactie die nodig is en

vereisen in alle gevallen een verbinding met de bank. Deze nadelen zijn vooral merkbaar in situaties waar wordt betaald bij direct contact, zoals in winkels of bij aflevering van goederen. De chipknip is daar vaak een handig alternatief voor de pinbetaling. Voor betalingen op afstand, zoals in een internetwinkel, zijn deze nadelen nauwelijks relevant: klant en winkelier zijn op het moment van betalen toch al op het netwerk.

Open en gesloten systemen. Voor de opslag en verwerking van elektronische munten zal aan klant en winkelier software en/of hardware beschikbaar worden gesteld. Een smartcard als portemonnee is handig mee te nemen en te gebruiken, en goede software voorkomt het onbedoeld kwijtraken of verdubbelen van munten.

In sommige systemen is de verstrekte hardware (chipkaart) een essentieel deel van de beveiliging van het systeem: de klant of winkelier zou zich kunnen verrijken door data of procedures te veranderen. De chipkaart is dan niet alleen een ondersteuning voor de betaalfunctie, maar ook afscherming van de data voor de gebruiker. De veiligheid van het betaalsysteem berust dan niet alleen op cryptografische methoden, maar ook op de (geclaimde) onmogelijkheid de hardware te openen of te wijzigen. Dit type systemen, waarvan de chipknip een voorbeeld is, noemen we *gesloten* systemen.

Bij andere systemen dient standaard software en hardware wel als ondersteuning voor de gebruiker, maar er is geen bedrog mogelijk door het veranderen van data of procedures. Er hoeft dan over de gebruikte hardware niets te worden aangenomen, eventueel kan de gebruiker ervoor kiezen eigen hardware (pc) of software te gebruiken; Dergelijke systemen heten *open* betaalsystemen.

13.2 E-muntsystemen: Ecash

Deze sectie behandelt systemen voor elektronisch geld op basis van cryptografisch gedefinieerde elektronische munten. Dit soort systemen komt in diverse varianten voor en er vinden veel ontwikkelingen op dit gebied plaats. Als 'moeder' van deze systemen wordt wel *Ecash* beschouwd, een betaalsysteem ontworpen voor Internet door het Amsterdamse bedrijf DigiCash (kader 13.5).

Het vormen en herkennen van munten. Elektronische munten in het systeem zijn door de bank ondertekende waardecertificaten, elk met een waarde (w) en een uniek serienummer (r). Het certificaat wordt gevormd door r en w te combineren met een vaste string ('Dit is een munt') en het resultaat schrijven we als $\langle\, \mathbf{munt}, w, r \,\rangle$. De lengte van deze string is enkele honderden bits en we kunnen de string opvatten als een getal x. Elke partij heeft een RSA-sleutel, die van de bank heeft de getallen n, e, d en de publieke sleutel is aan alle partijen bekend. Bob en Alice hebben sleutels n_B, e_B, d_B, n_A, e_A, d_A en ook hiervan veronderstellen we de publieke delen bekend. Door de ondertekening ontstaat een getal $y = x^d$.

De waarde van de munt kan in principe elk bedrag zijn. Als Alice al vantevoren weet dat ze Bob 37,52 euro moet betalen, kan zij een munt van 37,52 vragen. Het is ook mogelijk een aantal munten van verschillende waarden te vragen en later bij de betaling meerdere munten te combineren tot het gewenste bedrag.

De controle of een gegeven getal y een munt is, gebruikt alleen de publieke sleutel. Van y wordt $x = y^e$ berekend, en als hierin de bekende string (op de juiste plaats) voorkomt is y een munt, waarvan serienummer en waarde in x af te lezen zijn. We voeren voor de controle twee

Kader 13.5: DIGICASH: OPKOMST EN ONDERGANG.

DigiCash werd in 1990 opgericht in Amsterdam door David Chaum (hier op een tekening van Max Kisman). Gestimuleerd door RijksWaterStaat, dat een mogelijke techniek voor elektronische tolheffing zag, was rond 1993 Ecash ontwikkeld. Een eerste proef gebruikte een virtuele munteenheid, de *cyberbuck*, als geldeenheid, maar al snel was de Amerikaanse Mark Twain Bank bereid een proef met echte dollars in e-vorm te wagen. Het Ecashsysteem werd alom als een technisch hoogstandje beschouwd.

Minder hoogstaand was de managementstrategie van DigiCash in de jaren negentig. De geniale wiskundigen die een klein land cryptografisch groot hadden gemaakt slaagden er niet in hun plaats op de wereldkaart te consolideren. Verschillende mogelijkheden het systeem of bedrijf te verkopen werden afgeslagen, uit wantrouwen of in de verwachting een betere deal te kunnen sluiten. In 1997 werd de vestigingsplaats na interne ruzies naar Californië verplaatst, en in 1998 sloot de Amsterdamse vestiging.

Inmiddels waren er alternatieve betaalmodellen voor het internet ontstaan, bijvoorbeeld credit-debetsystemen (kader 13.4) en beveiligd gebruik van creditcards. Ecash is nu niet meer dan een van de vele betaalmogelijkheden op internet, en wordt nog steeds door diverse banken en internetwinkels ondersteund.

`www.digicash.com`.

functies in: $waarde(y)$ is w als er een r bestaat zodanig dat $y^e = \langle\, \mathbf{munt}, w, r \,\rangle$ en 0 anders. $sn(y)$ is r als er een w bestaat zodanig dat $y^e = \langle\, \mathbf{munt}, w, r \,\rangle$ en 0 anders.

Het is niet mogelijk een munt te maken zonder de geheime sleutel van de bank. Een existentiële vervalsing van een RSA-handtekening levert een willekeurig getal x op en de kans dat een gewenste string van l bits hierin voorkomt is $\frac{1}{2^l}$. Een valsemunter kan een munt vervalsen door 2^l getallen y uit te proberen maar met bijvoorbeeld $l = 120$ is dit volstrekt ondoenlijk.

Hetzelfde principe wordt gebruikt voor de vorming en ondertekening van de diverse certificaten die in de protocollen worden gebruikt. Merk op dat de omvang van een munt of ander certificaat gelijk is aan één RSA-getal, bijvoorbeeld 1024 bits en dat de controle slechts één exponentiatie vraagt met de publieke exponent. Door deze klein te kiezen (3 of 17) kost de verificatie slechts enkele vermenigvuldigingen.

13.2.1 Rondgang van elektronische munten

Het eenvoudigste betaalsysteem met zulke munten, weergegeven als protocol 13.6, biedt geen anonimiteit. Zoals elk betaalsysteem kent het drie protocollen, namelijk voor opnemen, overdragen, en afstorten van de munt.

Bij opname krijgt Alice de munt tegen debitering van haar banksaldo. Het serienummer

Fase 1: Opname

Alice		Bank
Identificeert zich	1	Accepteer identificatie van Alice
Vraag munt van w	2	Debiteer Alice voor w
	3	Kies random r; $x = \langle \mathbf{munt}, w, r \rangle$
Ontvang y	4	Zend x^d
Controleer $waarde(y) = w$	5	
Stop y in e-knip	6	

Fase 2: Overdracht

Alice		Bob
Identificeert zich	1	Accepteer identificatie van Alice
Stuur y uit e-knip	2	Controleer $waarde(y) = w$
Ontvang u	3	Geef een random getal u
Stuur $z = \langle \mathbf{overd}, r, u, Bob \rangle^{d_A}$	4	Controleer z
	5	Accepteer de betaling, stop y in kassa

Fase 3: Afstorten

Bob		Bank
Identificeert zich	1	Accepteer identificatie
Geef y uit kassa	2	Controleer of $sn(y)$ al gestort is
	3	Crediteer Bob voor w

Protocol 13.6: Betaalprotocol met e-munten.

wordt door de bank gekozen en Alice controleert alleen de waarde.

Bij overdracht geeft Alice de munt aan Bob en tekent ze een certificaat van overdracht $\langle \mathbf{overd}, r, u, Bob \rangle$, waarin het serienummer van y, de begunstigde, en een door hem gekozen random getal zijn opgenomen. Dit certificaat wordt natuurlijk met de sleutel van Alice ondertekend en gecontroleerd; de omvang is gelijk aan de lengte van de RSA-modulus. Als Bob het overdrachtscertificaat heeft gecontroleerd, accepteert hij y als betaling. Bij afstorten geeft Bob de munt aan de bank en wordt zijn rekening gecrediteerd (tenzij dezelfde munt al eerder is afgestort, zie verderop).

De overdrachtscertificaten dienen voor het opsporen van geldverdubbelaars; een verdubbelde munt wordt tweemaal (of vaker) afgestort en de bank merkt dit direct. Elke partij die de munt heeft afgestort, wordt gevraagd een overdrachtscertificaat te overleggen waarmee de munt in zijn bezit is gekomen. Als beide partijen een door Alice ondertekend certificaat kunnen overleggen is Alice schuldig; als een van de partijen zo'n certificaat niet heeft is die schuldig. Het random getal u in het certificaat voorkomt dat Alice tweemaal met dezelfde munt bij dezelfde winkelier kan betalen.

Eigenschappen van het systeem. Het protocol is niet anoniem. Bob moet de identiteit van Alice kennen om het overdrachtscertificaat met haar sleutel te kunnen controleren. De bank kan ook zien wie er aan Bob heeft betaald omdat de bank weet aan wie de door Bob gestorte munt is afgegeven.

De munten in dit systeem zijn zeer flexibel overdraagbaar: het is mogelijk dat Bob, in plaats

Fase 1: Opname

Alice		Bank
Identificeert zich	1	Accepteer identificatie van Alice
Vraag munt van w	2	Debiteer Alice voor w
Kies random r; $x = \langle\, \mathbf{munt}, w, r \,\rangle$	3	
Kies random k; stuur $x \cdot k^e$	4	Ontvang x'
Ontvang y'; $y = y'/k$	5	Stuur x'^d
Stop y in e-knip	6	

Fase 2: Overdracht en afstorten

Onbekend persoon		Bob	Bank
Stuur y uit e-knip	1	Controleer $waarde(y) = w$	
	2	Stuur y aan Bank	Ontvang y
	3		Kijk of $sn(y)$ is afgestort
	4	Ontvang accordering	Crediteer Bob en accordeer
	5	Accepteer betaling	

Protocol 13.7: Ecash anonieme betaalprotocollen.

van de munt af te storten, hem gebruikt om aan een volgende partij te betalen. Elke partij die de munt in bezit heeft gehad, moet wel het overdrachtscertificaat waarmee hij hem ontvangen heeft bewaren! Het opsporen van geldverdubbelaars begint bij de afstorters van de munt, en elke betrokkene moet een certificaat van zijn voorganger kunnen overleggen. Uiteindelijk komt de bank terecht bij een partij die meer certifcaten voor de munt heeft afgegeven dan ontvangen.

Betalingen in deze eenvoudige vorm zijn offline. Het systeem is open: weliswaar kan een partij door de portemonnee of kassa vals te administreren geld verdubbelen, maar dit wordt altijd bemerkt en gestraft. Met een omvang van 1024 bits per munt is het zeer geschikt voor internetbetalingen vanuit een pc. Voor betalingen vanaf een smartcard is het, door de beperkte geheugenomvang van de kaart, minder geschikt.

13.2.2 Betaleranonimiteit

In het basis-Ecashsysteem is het mogelijk om anoniem betalingen te doen: Bob weet dan niet wie hem betaalt, en ook uit de administratie van de bank mag niet blijken wie Bob heeft betaald, of wat Alice met haar geld doet. Natuurlijk is voor anonimiteit nodig, dat er meerdere betalers zijn die anoniem betalen en meerdere winkeliers die anonieme betalingen aannemen.

Serienummers van munten en identificatie van de betaler spelen in protocol 13.6 een essentiële rol in het opsporen van geldverdubbelaars. In het protocol van deze sectie dragen munten ook serienummers, maar bij opname krijgt Alice een munt met een voor de bank onbekend serienummer. Een centraal probleem is: als Alice een munt met onbekend serienummer heeft, en zich niet bekend hoeft te maken bij betaling, wat weerhoudt haar er dan van de munt ongebreideld te verdubbelen?

De eenvoudigste oplossing hiervoor is, de offline eigenschap van betalingen op te offeren en controle met de administratie van de bank tijdens de betaling uit te voeren. Geldverdubbelen is hiermee onmogelijk geworden; zie protocol 13.7. De opname is vergelijkbaar met het eerdere protocol 13.6: Alice moet zich identificeren omdat de opgenomen munt ten laste van haar

bankrekening komt. Echter, in afwijking van het eerdere protocol is het niet de bank, maar Alice die een serienummer voor de munt kiest, en de bank ondertekent de munt *blind* volgens het principe uit sectie 7.4.2. Zo verkrijgt Alice een munt waarvan het serienummer voor de bank *perfect verhuld* is. Perfecte verhulling is hier belangrijk: de bank kan niet alleen r niet berekenen uit x', maar ook is het na het afstorten van een serie munten met nummers $r_1, \ldots, r_m$ *niet mogelijk* te zien welke van die munten met de opname van Alice correspondeert.

Wanneer Bob een munt ontvangt stort hij hem direct af, nog voordat hij de betaling accepteert. De bank bekijkt het serienummer en dit moet een geheel nieuw nummer zijn: de bank heeft het nummer immers nog nooit gezien! Is ditzelfde nummer echter al eens eerder afgestort, dan wordt de munt geweigerd. Geldverdubbelen is dus onmogelijk. Een poging van Alice tot geldverdubbelen blijft ongestraft, omdat haar anonimiteit ook bij weigering van de munt gehandhaafd blijft.

De bank voert de opname uit met meerdere *bekende* betalers, en de overdracht en afstorting met meerdere *bekende* ontvangers. Door de perfecte verhulling van serienummers bij opname is het niet mogelijk te bepalen welke opname met welke overdracht correspondeert.

Controle op waarde bij opname. Aan protocol 13.7 ontbreekt nog een belangrijk deel, namelijk de controle dat de munt die Alice blind laat ondertekenen inderdaad de waarde w heeft. Ze kan gemakkelijk een munt van 10 euro aanvragen maar er een van 100 euro laten tekenen. In sectie 7.4.3 zijn al drie manieren behandeld om deze controle uit te voeren:

1. **Cut-and-choose:** Alice geeft meerdere (bijvoorbeeld twintig) geblindeerde munten aan de bank en moet er dan negentien ter controle openen, waarna de twintigste ongeopend wordt ondertekend.

2. **Slechts één waarde mogelijk:** De bank gebruikt de sleutel (n, e) maar voor het ondertekenen van munten van een bepaalde waarde. Een munt die een andere waarde vermeldt is ongeldig.

3. **Zero-knowledge proof:** Alice bewijst aan de bank dat ze over een k beschikt waarvoor de waarde van (x'/k^e) gelijk aan w is, zonder die k te onthullen.

Ophefbare anonimiteit. Het systeem kan ook worden uitgebreid met een mogelijkheid voor de betaler om zijn anonimiteit op te heffen, bijvoorbeeld in de volgende gevallen:

1. Alice beweert dat zij Bob heeft betaald maar dat Bob, na het afstorten van de munt (stap 4) heeft geweigerd haar de spullen mee te geven. Bob beweert van Alice helemaal geen munt te hebben gekregen. Door haar anonimiteit op te heffen kan Alice haar claim bewijzen dat de door Bob afgestorte munt door haar is gemaakt.

2. Alice heeft anoniem een product bij Bob gekocht. Het gaat echter kapot en Alice wil een beroep doen op de garantie. Ze moet dan bewijzen dat zij de betaling heeft gedaan.

3. Alice is undercoverrechercheur en koopt bij Bob wat illegale waar, die met een anoniem betaalsysteem is afgerekend. Voor de rechtbank wil ze bewijzen dat Bob degene is die haar geld heeft aangenomen.

In alle protocollen waarin een random getal wordt gegenereerd is het mogelijk dit zo te doen, dat de herkomst van het getal te bewijzen is. Het getal (r in dit protocol) wordt dan berekend als $r = H(s)$, waar H een hashfunctie is en s een random getal. Alice bewaart s, en kan deze later opvoeren als bewijs dat zij bij het genereren van r betrokken was. Alice' identiteit

kan expliciet in de berekening worden opgenomen, als in $r = H(Alice, s)$, zodat ook na het bekendmaken van s geen onduidelijkheid kan bestaan over het genereren van r.

Eigenschappen van het protocol. Er is aan het bezit van ongeregistreerd geld een risico verbonden: bij verlies van de munt is Alice haar geld onherroepelijk kwijt! In protocol 13.6 kon Alice bij de bank beargumenteren dat de door haar opgenomen munten niet waren afgestort en zodoende vergoeding krijgen, maar juist door de anonimiteit van de munten is dat niet meer mogelijk.

Het betaalsysteem is open: noch Alice, noch Bob kunnen voordeel halen uit geldverdubbeling. Het systeem is online: het directe afstorten is de enige manier om in een anoniem systeem geldverdubbelen te voorkomen. De grootte van een munt is gelijk aan de RSA-modulus.

Een nauwkeuriger beschrijving van de protocollen zoals die worden gebruikt in Ecash wordt gegeven door Schoenmakers [Sch97]. Een boeiende mogelijkheid is het uitgeven van geld met verschillende markeringen door het gebruik van verschillende RSA-sleutels door de bank. Zo kan bijvoorbeeld verschillend geld per leeftijdscategorie worden gemaakt, waarmee men wettelijke restricties op het kopen van tabak of drugs kan implementeren in elektronische betalingen.

13.2.3 Betaleranonimiteit bij offline betalen

Protocol 13.6 heeft de prettige eigenschap offline te zijn, maar biedt geen anonimiteit, terwijl het anonieme protocol 13.7 online betalingen heeft. Het ontwerp van een protocol dat *anonimiteit* met *offline* betalingen combineert stuit op het volgende paradoxale probleem. Met offline betalingen is geldverdubbeling niet te voorkomen: Alice kan altijd haar munt kopiëren en het overdrachtsprotocol bij meerdere winkeliers uitvoeren. Bij anonieme betalingen kan een geldverdubbelaar niet achteraf worden opgespoord en gestraft.

De uitweg uit deze paradox is, dat een betaler bij de overdracht enige informatie moet afgeven; uit deze informatie is de identiteit *niet* te achterhalen, maar uit de informatie van *twee* overdrachten *wel*.

1. **Verdeling van de identiteit.** In het offline schema van Ecash wordt de identiteit van de betaler verdeeld in een verdeelschema (met aandeelhouderverifiëerbaarheid). Bij de overdracht van de munt moet de betaler een aandeel afgeven. Wordt er tweemaal betaald, dan kan de bank uit de afgegeven aandelen de identiteit van de verdubbelaar berekenen. Alle gebruikte cryptografische technieken zijn in eerdere hoofdstukken aan bod gekomen, en het systeem wordt verderop besproken.

2. **Ondertekening met ElGamal.** Biham en Shulman [BS98] gebruiken per munt een geheime sleutel, bestaande uit de de identiteit van de opnemer en random informatie. Een overdrachtscertificaat wordt ondertekend met het ElGamal-schema (sectie 7.2.3), waarbij het gebruikte random getal k afhangt van de munt. Bij geldverdubbeling worden er twee overdrachtscertificaten ondertekend, en wel met hetzelfde random getal. Zoals besproken in sectie 7.2.3 is dan uit de twee handtekeningen de geheime sleutel te berekenen. Het systeem gebruikt diverse zero-knowledge proofs om de eerlijkheid bij opnemen en overdracht af te dwingen.

In het vervolg van deze sectie wordt een uitbreiding van het protocol beschreven die anonimiteit met offline betalingen combineert.

Structuur en opnemen van de e-munt. Net als in protocol 13.7 maakt Alice haar eigen munt zodat het serienummer voor de bank verborgen blijft. Naast de vaste string ('Dit is een munt'), de waarde en het serienummer bevat de munt tien paren blobs (sectie 7.3.3) waarin Alice zich vastlegt op aandelen van haar identiteit:

	Genereren van een w-certificaat (door Alice)
1	$r := random$
2	voor $i \leq 10$: $a_{i0} := random$; $a_{i1} := A - a_{i0}$
3	voor $i \leq 10$, $y \in \{0, 1\}$: $b_{iy} := H(a_{iy})$
4	$x = \langle \textbf{munt}, w, r, \vec{b} \rangle$
5	Verkrijg y waarvoor $y^e = H(x)$

Als A de identiteit als getal representeert, bestaat het i^{e} paar aandelen uit getallen a_{i0} en a_{i1} met som A (additief schema, sectie 9.2), berekend in stap 2. Het muntcertificaat bevat niet deze getallen, maar blobs ervan, dat wil zeggen, getallen $b_{iy} = H(a_{iy})$, waar H een sterk botsingsvrije hashfunctie is (stap 3). De getallen b_{iy} gelden als publieke aandelen en maken de verdeling aandeelhouderverifieerbaar. Alice is zelf de enige aandeelhouder: zij bewaart alle a_{iy}.

Het totale certificaat x is nu te groot geworden om het in één keer met RSA te ondertekenen, daarom wordt de handtekening y berekend met een hashfunctie: $y = H(x)^d$. Om de handtekening te verkrijgen moet Alice de munt door de bank laten ondertekenen en hierbij wordt zij uiteraard voor w gedebiteerd. Het ondertekenen gebeurt blind, zodat de bank het serienummer en verdere inhoud van het certificaat niet te zien krijgt. Wel wordt er cut-and-choose toegepast zodat de bank kan controleren dat de opgegeven waarde juist is en dat het certificaat inderdaad blobs van aandelen van Alice' identiteit bevat.

Overdracht van de munt. Als de munt x aan Bob wordt aangeboden ziet hij de waarde en het serienummer, en tien paren blobs van aandelen. De identiteit van de aanbieder is aan het certificaat niet te zien omdat de waarde van de aandelen niet uit de blobs te berekenen is. De geldigheid van x wordt gecontroleerd met de losse handtekening y, die de eigenschap $y^e = H(x)$ heeft, waardoor Bob weet dat er tegenover het certificaat een debitering staat.

Als onderdeel van de betaling mag Bob van elk paar blobs er een laten openen: Bob kiest als challenge tien bits $c_1 \ldots c_{10}$ en vraagt opening van b_{ic_i}. De betaler moet met aandelen a_{ic_i} voor de dag komen en de publieke aandelen maken controle mogelijk: Bob controleert voor elk van de tien ontvangen getallen dat $H(a_{ic_i}) = b_{ic_i}$. Bob bewaart het certificaat x met handtekening y, en zijn gekozen bits c_i en de antwoorden a_{ic_i}.

Verzilveren en fraudeopsporing. Bob biedt het certificaat met de bijbehorende informatie aan aan de bank. Deze controleert of er al eerder een certificaat met dit serienummer is afgestort. Normaal is dit niet het geval en dan wordt Bob gecrediteerd en alle gegevens worden opgeslagen.

De opsporing van de fraudeur ingeval een certificaat voor de tweede maal wordt afgestort is gebaseerd op deze gedachte: als er voor een tweede keer met hetzelfde certificaat wordt betaald, is de kans dat er de tweede keer *exact dezelfde* c_i-bits worden gevraagd 1 op 1024. Deze kans beschouwen we hier als verwaarloosbaar. Veronderstel dat het serienummer van de door Bob afgestorte munt gelijk is aan dat van een munt die eerder door Carol is afgestort. De bijgeleverde challenges c'_i en aandelen $a_{ic'_i}$ worden erbij gehaald. Als de door Carol aangeleverde aandelen alle tien dezelfde zijn als die door Bob werden aangeleverd ($c'_i = c_i$ voor alle i), zijn

ze waarschijnlijk in dezelfde betaling afgegeven! Bob en Carol worden ter verantwoording geroepen wegens een poging een munt dubbel af te storten.

In het andere geval is er minstens één i waarvoor $c_i \neq c'_i$. De aandelen a_{ic_i} en $a_{ic'_i}$ worden opgeteld en hierbij verschijnt een getal: de naam van een dame die de wereld binnenkort van achter tralies gaat bekijken.

Eigenschappen van het protocol. De betalingen zijn offline en ook anoniem, zolang er tenminste met elke munt maar één keer wordt betaald. Door de omvang van de muntcertificaten en de omslachtige opname en overdracht is het echter niet voor grootschalige toepassing geschikt.

Anonieme offline systemen geven, zelfs als de gebruikte cryptografie solide is, altijd een groot risico. Geldverdubbelen wordt immers niet voorkomen, alleen getraceerd door de afgegeven informatie. Als echter een munt, inclusief de geheime informatie om hem uit te geven, per ongeluk in verkeerde en onbekende handen valt, is een ongelimiteerde schade mogelijk. Oscar, die Alice' munt x met y en de bijbehorende a_{iy} steelt kan de munt willekeurig vaak uitgeven en de bank kan Alice niet meer verwijten dan onzorgvuldigheid met haar sleutels.

De fraudeopsporing in het systeem is niet waterdicht, er is immers een kleine kans dat Alice haar certificaat tweemaal uitgeeft en er tweemaal dezelfde challenge wordt gegeven. Alice kan aan een tweede betaling beginnen, maar kan de transactie afbreken als de gevraagde challenge haar niet bevalt. Deze problemen kunnen worden opgelost met een langere challenge, bijvoorbeeld tachtig bits, waarmee de kans op een herhaalde challenge echt verwaarloosbaar is. De omvang van het certificaat en de kosten van overdracht lopen dan echter evenredig op.

In een geavanceerdere variant van dit idee wordt de identiteit niet verdeeld door meerdere (tien in ons voorbeeld) vetoschema's, maar in een drempelschema. Het certificaat bevat dan publieke informatie over een eerstegraads polynoom dat de identiteit verbergt. De challenge van Bob dwingt de betaler om een aandeel af te geven, en de challenge kan een 80-bitsgetal zijn, zodat met dit ene schema de kans op succesvolle fraude verwaarloosbaar is.

13.3 Registersystemen

Zoals al is aangegeven zijn de behandelde systemen met elektronische munten nog niet rijp voor grootschalige invoering op smartcards. De Nederlandse chipknip gebruikt dan ook helemaal geen discrete, waardedragende objecten; er wordt zelfs vrijwel geen public-keycryptografie in gebruikt. Geld is in de chipknip niet aanwezig in de vorm van objecten, maar is gerepresenteerd als de waarde van een saldoregister in de kaart. Dit registermodel wordt ook toegepast in telefoonkaarten en spaarpuntenregistraties.

In deze sectie beschrijven we de algemene werking van dergelijke systemen. De smartcard van de klant en de kassa van de winkelier houden een saldo bij. Verhoging van saldo mag alleen plaatsvinden op verzoek van een *geautoriseerde partij*, en een speler in het spel geeft zo'n verzoek alleen af tegen verlaging van het eigen saldo.

De partijen die saldo kunnen bevatten of afgeven zijn: (1) de bank; (2) de chipknippen; (3) de betaalautomaten. Alle gebruikte hardware is gesloten: het is duidelijk dat Alice of Bob zich kunnen verrijken door hun data te modificeren. De (*niet-cryptografische*) aanname dat smartcards niet kunnen worden geopend is daarom essentieel voor de veiligheid van een register-systeem. De kaart van Alice en de kassa van Bob verschijnen in dit systeem als afzonderlijke partijen (wat vooral bij opname opvalt).

Kader 13.8: Interpay en elektronisch betalen

In 1994 ontstond Interpay uit een fusie van de BankGiroCentrale, BeaNet en Eurocard Nederland. De BankGiroCentrale was in 1967 opgericht om betalingen tussen rekeningen bij verschillende banken te vergemakkelijken. BeaNet bestond sinds 1988 als de organisatie die elektronische betalingen met bankpassen afhandelde.
Interpay is een dochteronderneming van de Nederlandse banken en verzorgt alle betalingen tussen banken onderling. Creditcardgebruik en pinnen zijn snelgroeiend, maar het chippen is in Nederland nog niet erg populair geworden. In 2000 werd de hoop van Interpay op het rekeningrijden gevestigd: een tolpoortsysteem met chipknipmogelijkheid zou een grote impuls kunnen betekenen.
Interpay vervult de rol van de bank in de betaalprotocollen, maar stuurt de opdrachten voor creditering en debitering door naar de banken waar Alice en Bob hun rekening hebben.

Beschrijving van protocollen. Bij het opladen van de kaart (fase 1) moet Alice zich identificeren omdat de saldoverhoging ten laste van haar rekening gaat. Hierna is het belangrijk dat de bank zich tegenover de kaart identificeert: anders zou Alice de kaart zelf kunnen opladen met een 'nepbank' die wel een opdracht tot saldoverhogen geeft, maar geen rekening debiteert.

Bij het betalen (fase 2) krijgt de kassa de opdracht het saldo te verhogen. Hierbij moet erop

Fase 1: Opname.

Alice		Kaart	Bank
Identificeer naar Bank	1		Accepteer echtheid Alice
Geef bedrag w	2		Debiteer Alice voor w
	3	Accepteer echtheid Bank	Identificeer naar Kaart
	4	Ontvang w van bank	Authoriseer w voor Kaart
	5	Verhoog saldo met w	

Fase 2: Overdracht

Kaart		Kassa
Identificeer als kaart	1	Accepteer echtheid van kaart
Ontvang w	2	Stuur bedrag w
Als saldo $< w$: breek af	3	
Verlaag saldo en stuur acceptatie	4	Ontvang acceptatie
Ontvang betalingsbevestiging	5	Stuur betalingsbevestiging
	6	Verhoog saldo met w

Fase 3: Afstorten

Bob		Kassa	Bank
Identificeer naar bank	1	Identificeer naar bank	Accepteer Bob en Kassa
	2	Stuur saldo	Ontvang saldo
	3	Zet saldo op 0	Crediteer Bob

Protocol 13.9: Waardeoverdracht in een registersysteem.

worden gelet dat deze opdracht van een echte partij komt, de kaart moet zich dus tegenover de kassa identificeren. Interessant om op te merken hierbij is dat er geld kan verdwijnen, namelijk als de communicatie wordt verstoord tijdens stap 4. De kaart boekt het geld dan af, maar het komt niet in de kassa. Bij de kaart blijft de betalingsbevestiging dan uit en het geld kan worden teruggewonnen door de transacties van de kaart en de kassa met elkaar te vergelijken.

Bij het afstorten wordt Bob gecrediteerd op last van de kassa, zodat het van belang is dat de kassa echt is: die moet zich dus identificeren.

Eigenschappen. Als Alice haar kaart verliest, is ze het saldo erop kwijt: het kan immers door iedereen worden gebruikt omdat bij het betalen alleen de echtheid van de kaart wordt gecontroleerd, en niet of de aanbieder ook eigenaar is. De vinder kan de kaart niet ten laste van Alice opladen, omdat voor opname wel identificatie nodig is.

Een registersysteem is altijd gesloten: een gebruiker kan anders altijd geld 'bijmaken' door het eigen register te verhogen.

Privacy. De authenticatie bij diverse handelingen staat privacy in de weg; immers, betalingen worden opgeslagen in kaart en kassa, en al kan deze informatie voor de winkelier worden afgeschermd, de bank krijgt toch een overzicht van alle betalingen. De authenticatie in de chipknip gebeurt met protocol 8.3, waarbij kaart en kassa over een gedeelde geheime sleutel beschikken.

De identificatie van de kaart bij betalen dient er eigenlijk alleen voor dat het betalen alleen met een *echte* kaart kan gebeuren. Welke kaart dat precies is, is niet van belang als smartcards inderdaad niet te compromitteren zijn, immers, elke echte kaart verlaagt het interne saldo zodat er geen geld wordt gecreëerd. Volledige privacy is dan te bereiken door elke kaart te voorzien van dezelfde sleutel. Identificatie overtuigt de kassa ervan dat er wordt gecommuniceerd met een kaart die de sleutel kent, dus echt is, maar er wordt geen onderscheid gemaakt tussen de verschillende kaarten. (Bij opname speelt de kaart een rol als hulpmiddel bij de identificatie van de gebuiker, en die functie is natuurlijk wel voor elke kaart verschillend.)

Als smartcards worden gebruikt die wel te compromitteren zijn, kan een gebruiker (klant of winkelier) wel frauderen en dan is volledige identificatie belangrijk om de gemodificeerde kaarten op te sporen. De bank registreert alle transacties per kaart, en kan zo een schaduwadministratie van elk kaartsaldo bijhouden. Als er met een kaart meer wordt betaald dan opgenomen, wordt de houder verzocht het exemplaar in te leveren. De chipknip gebruikt unieke serienummers, en winkeliers zijn verplicht ten minste wekelijks af te storten zodat geknoei met de kaart in redelijke tijd wordt ontdekt.

Tussen volledige traceerbaarheid en volledige privacy bestaat ook een middenweg: het gebruik van een vast aantal, zeg duizend, verschillende sleutels. Blijkt een sleutel gecompromitteerd, dan moet slechts een duizendste van de klanten een nieuwe kaart krijgen. Individuele transacties zijn niet te traceren naar individuele kaarten, maar wel naar een groep die ongeveer een duizendste van de klanten omvat.

Samenvatting en conclusies

Geld speelt in onze samenleving een belangrijke rol, die echter in de loop van de geschiedenis steeds op een andere manier is vervuld. Tegenwoordig zien we geld als hulpmiddel voor het uitvoeren van betalingen, namelijk om giraal geld van een rekening naar een andere over

Kader 13.10: Modificatie van een chipkaart

Op enkele tientallen plaatsen in Nederland is apparatuur aanwezig waarmee elektrische schakelingen zo klein als een chip kunnen worden bemeten en gemodificeerd. Het oplossen van de buitenkant (met speciale chemicaliën) legt het elektrisch gedeelte bloot (zie kader 12.5), waarna de meting en verandering wordt gedaan onder een microscoop met microgereedschap.
De te onderzoeken chip ligt op een werkbank die door een vertragingsmechanisme heel precies kan worden gepositioneerd. Hierboven is een optische microscoop aangebracht die details kleiner dan 1 micron in kleur kan weergeven op een computerscherm; individuele transistors en verbindingen op de chip zijn zichtbaar. Om de kaart heen staan enkele microprobes: dit zijn nauwkeurig te positioneren contactpennen waarmee signalen op de chip kunnen worden gemeten of ingebracht.
De investering voor een complete uitrusting bedraagt honderdduizend tot tweehonderdduizend euro. Het is mogelijk op deze manier de authenticatiegegevens uit de chip te lezen, waarna een nieuwe chip volgens 'vrij ontwerp' kan worden gemaakt. De nieuwe chip kan zich tegenover betaalterminals als een echte chip voordoen (met de authenticatiegegevens), maar is bijvoorbeeld op te laden zonder interactie met de bank. Je kunt ook de bestaande chip kapot maken, bijvoorbeeld zo, dat het saldo niet meer wordt verlaagd, en dan is de chip geschikt voor eindeloos betalen.

te boeken. Chartaal geld (munten en bankbiljetten) kent een scherp omschreven juridische betekenis en bescherming. Die gelden voor elektronisch geld niet, zodat elektronisch geld niet in wettelijke zin als geld kan worden beschouwd.

De uitvoering van elektronisch geld en het betalen ermee moeten en kunnen met cryptografie worden ondersteund. Elektronisch geld bestaat, net als fysiek geld, uit overdraagbare objecten, elk met een bepaalde waarde. Net als fysiek geld heeft het een echtheidskenmerk, namelijk een digitale handtekening van de uitgevende instantie (bank). Anders dan fysiek geld is het zeer gemakkelijk te kopiëren, inclusief het echtheidskenmerk, waardoor het probleem van geldverdubbeling ontstaat.

Geldverdubbeling kan gemakkelijk worden getraceerd en gestraft in een niet-anoniem systeem. Daarvoor is nodig dat de betaler zich identificeert en dat er bij elke betaling een certificaat van overdracht voor elke munt wordt getekend. Geldverdubbeling kan onmogelijk worden gemaakt bij online betalingen, waarbij elke betaalde munt direct in de administratie van de bank wordt gecontroleerd. Bij anonieme offline betalingen is geldverdubbeling tegen

te gaan door de betaler een aandeel van zijn identiteit af te laten geven. Een enkele betaling laat de anonimiteit bestaan, maar een tweede betaling met dezelfde munt onthult de identiteit. Geavanceerde sleutelverdeling en zero-knowledge proofs zijn nodig om aan alle eisen van veiligheid en anonimiteit te voldoen.

Spaarkaarten en telefoonkaarten, maar ook de chipknip, gebruiken een veel goedkopere techniek. Hierin bestaat het geld niet uit afzonderlijke objecten, maar de kaart houdt intern een saldo als getal bij. Cryptografie ondersteunt de identificatie en communicatie tussen de verschillende partijen in het systeem. De gebruikte kaarten moeten in dit geval veilig zijn, dat wil zeggen, dat het openen ervan en het veranderen van de werking of het uitlezen van geheimen onmogelijk moet zijn.

Opgaven bij hoofdstuk 13

Opgave 13.1 *Is Rabo Direct Betalen een open of gesloten betaalsysteem?*

Opgave 13.2 *Bekijk een variant op protocol 13.6, waarin het overdrachtscertificaat niet een door Bob gekozen random getal, maar (1) een door Alice gekozen random getal; of (2) de tijd bevat. Is het protocol nog veilig?*

Opgave 13.3 *Bereken de kans dat er bij protocol 13.7 twee munten worden gevormd met hetzelfde serienummer.*
(Maak aannamen over het aantal gebruikers, aantal opnamen, lengte van serienummers etcetera, en gebruik stelling A.6.)

Opgave 13.4 *Wat gebeurt er wanneer bij protocol 13.7 twee gebruikers bij het opnemen van een munt hetzelfde serienummer genereren? Hoe kan de gedupeerde schadeloos worden gesteld?*

Opgave 13.5 Betaler-Anoniem Geld. *(Uit het tentamen van maart 1999.)*
In elementaire betaalsystemen wordt een geldstuk gerepresenteerd door de handtekening van de bank over een string die de waarde en een serienummer vermeldt.
(a) Waarom is het serienummer nodig?
(b) In een betaler-anoniem systeem moet iemand geldstukken kunnen opnemen zonder dat bekend is wie welke geldstukken krijgt. Hoe weet de bank dan ten laste van wiens rekening het opgenomen geld wordt afgeschreven?
(c) Beschrijf hoe de opname in een betaler-anoniem protocol verloopt.
(d) Hoe wordt in zo'n systeem double spending voorkomen?

Opgave 13.6 *Veronderstel dat de hashfunctie die in sectie 13.2.3 wordt gebruikt om blobs te maken niet sterk, maar slechts zwak botsingsvrij is; met andere woorden, Oscar heeft een manier bedacht om strings x_1 en x_2 te vinden met dezelfde hashwaarde. Beschrijf hoe Oscar een tientje kan maken dat hij ongestraft zo vaak kan uitgeven als hij wil.*

Opgave 13.7 *Bij de behandelde authenticatie van de chipknip (protocol 8.3, sleutelbeheer in sectie 4.4.4) kan een valse knip met nieuw serienummer x worden gemaakt met de masterkey M.*

(a) Beschrijf in detail hoe.

Om deze vervalsing tegen te gaan heeft elke chipknip de RSA-handtekening van de bank over

zijn serienummer, dus een y zdd. $y^e = x$.

(b) Laat zien hoe de authenticatieprocedure moet worden uitgebreid om deze kennis te controleren.

(c) Bedenk een methode om valse passen te maken (zonder RSA-handtekeningen te vervalsen).

(d) Voor wie wordt het systeem door de RSA-toevoeging veiliger?

Opgave 13.8 *Veronderstel dat verlies van geld in fase 2 van protocol 13.9 zo wordt opgelost: als bij de kaart de betalingsbevestiging uitblijft wordt w weer aan het saldo toegevoegd. Hoe kan een gebruiker zich verrijken?*

Hoofdstuk 14

Secure computing

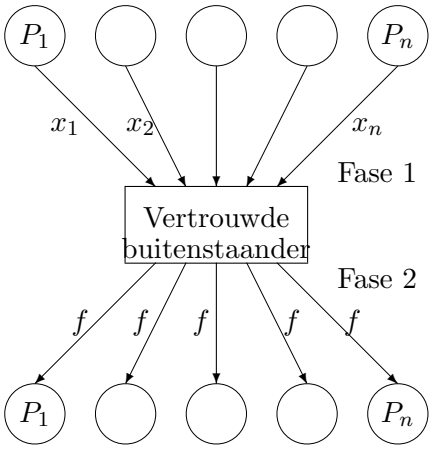
Het probleem dat centraal staat in dit hoofdstuk is het uitvoeren van berekeningen, waarvan de invoerwaarden verdeeld zijn over meerdere partijen, en waarvan deze partijen niet bereid zijn hun waarden openbaar te maken. Op de een of andere cryptografische manier moet er een balans worden gevonden tussen het *niet* onthullen van de eigen invoer en *wel* die invoer deel laten nemen aan de berekening. In dit hoofdstuk komen enkele klassieke en moderne problemen van dit soort aan bod, maar ook de belangrijkste technieken voor het vinden van deze balans.

Een klassiek probleem, van Yao [Yao82], is dat van twee miljonairs die willen weten wie van hen het rijkst is, zonder te vertellen hoeveel ze bezitten. Ook zullen we terugkomen op het probleem van de villabewoners (sectie 9.1.2) die de gemiddelde waarde van hun huizen willen uitrekenen (secties 14.1.3 en 14.2.3). Zijn deze problemen nog opgenomen om de ontwikkeling en het gebruik van de technieken te illustreren, het probleem van geheime verkiezingen is actueel en maatschappelijk relevant. Hier moeten uiteraard de stemmen worden geteld zonder dat een individuele stem zichtbaar wordt. Gezien het grote democratische belang van vrije verkiezingen zijn de eisen op een acceptabel protocol streng; we zullen enkele protocollen voor verkiezingen bekijken in sectie 14.3.

Om de invoerwaarden van de deelnemers geheim te houden kunnen verschillende technieken worden gebruikt. Er kan een verdeelschema worden gebruikt, mits de functie zo is dat deze door bewerkingen op de aandelen kan worden uitgevoerd; alleen het resultaat wordt dan gereconstrueerd; zie sectie 14.1.3. Een modernere aanpak is encryptie van de invoer (en tussenresultaten); hier moet de functie op versleutelde getallen kunnen worden geëvalueerd, een eigenschap die homomorfie wordt genoemd. Deze aanpak kan efficiënter zijn omdat elke invoer (en tussenresultaat) nu als één waarde is gerepresenteerd en niet als meerdere aandelen; zie sectie 14.2. In een voorbeeld worden de invoerwaarden niet onleesbaar gemaakt, maar anoniem ingestuurd waardoor niemand ziet *welke* waarde *van wie* afkomstig is (sectie 14.3.3).

Om de eerlijkheid van partijen af te dwingen kan ook van een veelheid aan technieken, en meestal combinaties daarvan, gebruik worden gemaakt. Publieke informatie om verdeelschema's verifieerbaar te maken en zero-knowledge proofs om de juistheid van tussenresultaten aan te tonen zijn we al eerder tegengekomen (hoofdstuk 9). Nieuw in dit hoofdstuk is het gebruik van *oblivious transfer*; hierbij moet een partij een waarde overleggen, die met een zekere kans al bij de tegenpartij bekend is. Is dit het geval, dan wordt een leugenaar meteen ontmaskerd; zie sectie 14.4.2.

Sectie 14.1 geeft enkele inleidende definities en voorbeelden. Sectie 14.2 behandelt het gebruik van homomorfe encryptie. Sectie 14.3 gaat in op geheime verkiezingen. Dit materiaal



Figuur 14.1: Referentie: het ideale protocol.

heeft allemaal betrekking op het berekenen van functies, maar in sectie 14.4 komen algemenere protocollen aan de orde.

14.1 Inleiding

14.1.1 Definities en eisen

We beschouwen n deelnemers P_1 tot en met P_n, waarbij P_i als enige beschikt over een waarde w_i. Verder nemen we aan dat een functie f op n invoerwaarden is gedefinieerd en aan alle partijen bekend is. Het doel van secure computing is de waarde $f(w_1, \ldots, w_n)$ door een gezamelijk protocol te bepalen, echter zonder dat anderen dan P_i de waarde w_i te weten komen.

Het is echter niet altijd mogelijk een waarde volledig geheim te houden; een elementair voorbeeld is waar Alice en Bob de som f van hun getallen w_A en w_B gaan berekenen. Bob immers kent w_B, en met de uitkomst f van het protocol kan hij w_A altijd uitrekenen als $f - w_B$, hoe de berekening ook verloopt. Daarom relateren Canetti e.a. [CDD+01] de veiligheid van een protocol aan dat van het zogenaamde *ideale protocol*, dat van een vertrouwde buitenstaander (Trusted Third Party) gebruikmaakt; zie figuur 14.1. Aan deze buitenstaander (soms *black box* genoemd) worden de invoeren opgestuurd, waarna deze f berekent en de waarde terugstuurt. Van de black box wordt aangenomen dat hij f eerlijk berekent en aan niemand enige informatie anders dan f teruggeeft.

Deze black box is niet meer dan een cryptografische abstractie; f wordt berekend door een cryptografisch protocol $\mathcal{P}$ dat door de deelnemers wordt uitgevoerd. Partijen (waaronder de deelnemers) kunnen proberen het protocol te verstoren door berichten af te luisteren, tussenresultaten van malafide deelnemers bij elkaar te leggen, of valse berichten te sturen. Het doel ervan kan zijn een verkeerde uitkomst te laten berekenen, of de invoerwaarde van een deelnemer te kunnen achterhalen. Het protocol wordt echter als *veilig* beschouwd wanneer er voor elke denkbare aanval *ook in het ideale protocol* een mogelijkheid bestaat om hetzelfde

resultaat te behalen.

Voor het gedrag van kwaadwillende deelnemers worden twee modellen onderscheiden.

1. **Nieuwsgierige deelnemers:** De kwaadwillende deelnemers zijn nieuwsgierig naar de invoer van andere deelnemers, en zijn bereid hun informatie met elkaar te delen, maar volgen wel eerlijk het protocol.

2. **Malafide deelnemers:** Om aan extra informatie te komen, of zelfs de correctheid van de uitkomst te verstoren, liegen malafide deelnemers over de uitkomst van hun deelberekeningen.

Voor de meeste problemen zullen we een oplossing die veilig is tegen nieuwsgierige deelnemers in detail uitwerken, en van bescherming tegen malafide deelnemers alleen schetsmatig aangeven hoe die kan worden bereikt. De definitie van veiligheid wordt in de volgende sectie geïllustreerd aan de hand van het miljonairsprobleem.

14.1.2 Het miljonairsprobleem

De miljonairs Alice en Bob willen weten wie van hen het rijkst is maar geen van beiden wil aan de ander onthullen hoe rijk hij precies is. Het vermogen van Alice wordt afgerond op a miljoen en dat van Bob op b miljoen (in deze kringen kijkt men niet op een half miljoentje meer of minder) en de getallen a en b liggen in het bereik $1 \leq a, b \leq 100$. Encryptie en decryptie met Alice' publieke en geheime sleutel worden als de functies E_A en D_A (in $\mathbb{Z}_n$) weergegeven.

Het protocol van Yao [Yao82] bepaalt of $a \leq b$:

1. *Bob kiest een random x, berekent $k = E_A(x)$ en stuurt $l = k - b$ naar Alice.*

2. *Alice ontsleutelt de honderd getallen na l, dat wil zeggen, berekent $y_i = D_A(l + i)$ voor $i = 1 \ldots 100$.*

   Merk op dat $y_b = x$, de andere y liggen willekeurig verdeeld.

3. *Alice verhoogt de getallen vanaf index a, dat wil zeggen, berekent*

$$t_i = \begin{cases} y_i & \text{als } i < a \\ y_i + 1 & \text{als } i \geq a \end{cases} .$$

   Merk op dat nu $t_b = x \iff b < a$. Hoe kunnen we testen of $t_b = x$? Als Bob x naar Alice stuurt kan Alice (met l) b berekenen. Als Alice de rij t opstuurt kan Bob deze getallen encrypten en ziet hij de plaats waar de resultaten ophouden een opeenvolgende reeks te vormen. We laten Alice niet de rij t opsturen, maar de restklassen modulo een nieuw priemgetal p; met deze getallen kan Bob wel de gelijkheid met x testen, maar geen decryptie doen.

4. *Alice kiest een priemgetal $p \approx \sqrt{n}$, berekent $u_i = t_i \bmod p$, en stuurt alle u_i en p naar Bob.*

5. *Bob test of $u_b = x \bmod p$ en vertelt het antwoord aan Alice.*

Wat zien de partijen van elkaars rijkdom? Alice ziet in stap 1 $l = E_A(x) - b$, maar omdat x (en dus $E_A(x)$) een random getal is vertelt dit niets over b. Bob ziet de getallen u_i en kan, omdat hij de waarde van u_b kent, zien of het getal y_b was opgehoogd (in stap 3). Maar omdat voor Bob de andere y_i niet te onderscheiden zijn van random getallen, kan Bob niet zien welke

andere y_i waren verhoogd. Beide partijen krijgen dus (bij eerlijk spel) geen andere informatie dan of de ander meer of minder bezit dan hijzelf.

We beschouwen nu twee scenario's voor oneerlijk spel. In het eerste wil Alice helemaal niet weten of Bob rijker is dan zijzelf, maar of hij rijker is dan haar (huidige!) verloofde en c-voudig miljonair Charles. Door in stap 3 niet de getallen vanaf index a, maar vanaf index c te verhogen, bepaalt Bob onbewust in stap 5 hoe zijn vermogen (en dus zijn huwelijkskans!) zich met dat van Charles verhoudt. Alice kan dus het protocol misbruiken om iets anders uit te rekenen dan ze met Bob had afgesproken. Zij zou echter hetzelfde kunnen doen in het ideale protocol: ook tegenover een vertrouwde partij kan zij liegen over haar vermogen, en dat van Charles opgeven, zodat de vertrouwde derde vertelt of Bob meer dan c bezit. Dit misbruik beschouwen we daarom niet als een zwakte in het protocol.

In het tweede scenario wil Alice niet weten of Bob rijker is dan zijzelf, maar of Bobs vermogen een even getal is. Ze verhoogt in stap 3 niet de getallen y vanaf een zekere index, maar verhoogt precies die y_i voor *oneven* i. Dan geldt $t_b = x \iff even(b)$, en onbewust bepaalt Bob in stap 5 de pariteit van zijn vermogen. Dit misbruik is in het ideale protocol onmogelijk en is dus een zwakte van het protocol.

14.1.3 Homomorfe verdeelschema's

Hoewel sommige ideeën uit het protocol in sectie 14.1.2 ook voor andere probleemstellingen bruikbaar zijn, is het protocol als geheel toch probleemspecifiek ontworpen. We zullen nu *algemene* technieken beschrijven, die voor een grote klasse van functies een geschikt protocol opleveren.

Homomorfie: rekenen met aandelen. Uitgangspunt is, dat er voor de functie f een manier is om de waarde uit te rekenen bij bekende invoeren; deze rekenwijze bestaat uit elementaire rekenstappen op invoer en tussenresultaten. Vervolgens wordt een *homomorf verdeelschema* gebruikt. Verdeelschema's kennen we uit hoofdstuk 9: van een geheim worden meerdere aandelen uitgedeeld over verschillende partijen, en alleen door samenwerking kan het geheim worden gevonden. De berekening van f kan nu door de partijen samen worden uitgevoerd als het verdeelschema *homomorf* is, wat het volgende wil zeggen. Voor elke elementaire bewerking $z = x \star y$ die in de berekening van f voorkomt, bestaat er een protocol waarin partij P_i moet deelnemen met een aandeel van x en een aandeel van y, en waarbij hij na afloop een aandeel van z verkrijgt.

Van een homomorf verdeelschema is in hoofdstuk 9 al een voorbeeld behandeld, namelijk de villabewoners, waarin de uitvoer de som van de (geheime) invoeren moest zijn; de enige bewerking in de berekening van f is dus optellen. Nu is optellen (in het additieve verdeelschema) erg eenvoudig te doen: is namelijk x verdeeld in aandelen x_i en y in aandelen y_i, dan geldt $x = \sum_i x_i$ en $y = \sum_i y_i$, en dus $x + y = \sum_i x_i + \sum_i y_i = \sum_i (x_i + y_i)$. Voor het 'protocol' dat de aandelen van z berekent is dus in het geheel geen interactie tussen de aandeelhouders nodig, want het aandeel z_i wordt lokaal gevonden door optelling: $z_i = x_i + y_i$.

De algemene structuur van een protocol voor functie f laat zich nu in protocol 9.3 al herkennen:

1. Elke partij deelt zijn invoer.
2. De functie wordt stap voor stap op de aandelen uitgevoerd.
3. De einduitkomst wordt gereconstrueerd.

Boolse functies. Een functie waarvan de invoer uit een rijtje bits bestaat en de uitvoer uit één bit, kan altijd worden beschreven als een boolse functie, dat wil zeggen, een combinatie van logische operatoren $\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$. Als voorbeeld geven we een boolse uitdrukking voor het miljonairsprobleem; laat de getallen a en b zijn geschreven als 3-bitsgetallen $a_2a_1a_0$ en $b_2b_1b_0$. De vergelijking $a \leq b$ geldt als $a_2 < b_2$ of $(a_2 = b_2 \wedge a_1a_0 < b_1b_0)$; immers we kijken eerst naar de meest significante bit en als daar gelijkheid is naar de rest van de getallen. De laatste vergelijking verder uitwerkend vinden we dat het antwoord $a \leq b$ gelijk is aan

$$(\neg a_2 \wedge b_2) \vee ((a_2 \Leftrightarrow b_2) \wedge [(\neg a_1 \wedge b_1) \vee ((a_1 \Leftrightarrow b_1) \wedge [\neg a_0 \wedge b_0])]) .$$

Ook voor getallen van meer bits is zo'n formule op te stellen; hij wordt dan natuurlijk langer.

Verdeling met XOR-schema. Door Yao [Yao82] werd aangetoond dat elke boolse functie met een protocol kan worden uitgerekend; we volgen de beschrijving van Cramer [Cra99]. Elke bit x (invoer, tussenresultaat of eindresultaat) wordt verdeeld over aandelen x_A en x_B voor Alice en Bob, waarbij geldt $x = x_A \oplus x_B$. Dit XOR-schema is perfect verhullend.

Voor het verdelen van invoerbit w van Alice kiest zij een random bit r, geeft ze $w_B = r$ aan Bob en houdt ze $w_A = w \oplus r$ zelf. Voor de reconstructie van het antwoord u uit u_A en u_B moeten Alice en Bob hun aandeel vertellen, en dan wordt u berekend als $u_A \oplus u_B$.

Deelprotocollen voor elementaire operaties. Neem nu aan dat bits x en y zijn verdeeld als x_A en x_B, respectievelijk y_A en y_B. We bekijken protocollen voor het berekenen van aandelen van $\neg x$, $x \Leftrightarrow y$ en $x \wedge y$. De eerste twee protocollen bestaan alleen uit een lokale berekening, het derde is ingewikkelder en vereist echte interactie tussen Alice en Bob.

Negatie: $z = \neg x$. Voor het berekenen van een negatie moet een van de deelnemers, zeg Alice, de eigen bit negeren

$$\begin{aligned} z_A &= \neg x_A \\ z_B &= x_B \end{aligned}$$

De juistheid van het resultaat volgt uit

$$\begin{aligned} z &= z_A \oplus z_B && \text{Definitie van reconstructiewaarde} \\ &= (\neg x_A) \oplus x_B && \text{Keuze van } x_A,\ x_B \\ &= \neg(x_A \oplus x_B) && \text{Omschrijven} \\ &= \neg x \end{aligned}$$

Equivalentie: $z = x \Leftrightarrow y$. Merk eerst op dat $(x \Leftrightarrow y) = \neg(x \oplus y)$. Beide deelnemers XOR-en hun eigen shares, en een van hen, zeg Bob, negeert het resultaat:

$$\begin{aligned} z_A &= x_A \oplus y_A \\ z_B &= \neg x_B \oplus y_B \end{aligned}$$

Weer is geen interactie tussen de partijen nodig; de correctheid volgt uit:

$$\begin{aligned} z &= z_A \oplus z_B && \text{Definitie van reconstructiewaarde} \\ &= (x_A \oplus y_A) \oplus (\neg x_B \oplus y_B) && \text{Keuze van } x_A,\ x_B \\ &= \neg(x_A \oplus x_B) \oplus (y_A \oplus y_B) && \text{Omschrijven} \\ &= \neg(x \oplus y) \\ &= x \Leftrightarrow y \end{aligned}$$

Conjunctie: $z = x \wedge y$. Bij conjunctie is er echt interactie tussen Alice en Bob nodig. Het gewenste resultaat, namelijk $(x_A \oplus x_B) \wedge (y_A \oplus y_B)$ kan worden geschreven als de som van vier termen (zie verderop), waarvan er een door Alice lokaal kan worden berekend en een door Bob. De lokaal berekende termen maken deel uit van z_A, respectievelijk z_B, en de andere twee termen worden bepaald met een hulpprotocol.

Er wordt gebruikgemaakt van een hulpprotocol dat een vorm van *blinde overdracht* (of *oblivious transfer*) implementeert. Bij blinde overdracht heeft Alice twee bits, a_0 en a_1, en Bob één selectiebit b. Bob kan door zijn selectiebit kiezen welk van de twee bits van Alice hij wil zien, maar Alice ziet niet welke bit Bob kiest. Na afloop van het protocol kent Bob a_b, maar Alice kent b *niet*. De uitkomst van het protocol bij Bob, weergegeven als $u = OT(a_0, a_1; b)$, is logisch uit te drukken als

$$u = (\neg b \wedge a_0) \oplus (b \wedge a_1).$$

Een mogelijke implementatie van blinde overdracht (zie ook sectie 14.4.1) is de volgende. Alice kiest een nieuwe RSA-sleutel (n, e, d) en twee random getallen r_0 en r_1, waarbij de laatste bit van r_0 gelijk is aan a_0 en de laatste bit van r_1 gelijk aan a_1. Alice stuurt de sleutel (n, e) en de versleutelde randoms: r_0^e en r_1^e. Bob kiest een van de versleutelde getallen en mag dit *blind* laten ontsleutelen door Alice (volgens sectie 7.4.2). Bob ziet het gekozen getal r, in het bijzonder de laatste bit. De andere bit blijft voor Bob onzichtbaar door de versleuteling, en de keuze van Bob blijft voor Alice onzichtbaar door de blindering.

Voor de berekening van de conjunctie bedenken we dat

$$(x_A \oplus x_B) \wedge (y_A \oplus y_B) = (x_A \wedge y_A) \oplus (x_A \wedge y_B) \oplus (x_B \wedge y_A) \oplus (x_B \wedge y_B).$$

De eerste en laatste term kunnen door Alice, respectievelijk Bob, lokaal worden berekend. Voor de andere twee termen wordt blinde overdracht gebruikt. De eerste keer kiest Alice een random bit ρ_0 en biedt ze de bits ρ_0 en $\rho_0 \oplus x_A$ aan, waaruit Bob selecteert met y_B; de uitkomst u_0 is

$$\begin{aligned} u_0 &= OT(\rho_0, \rho_0 \oplus x_A; y_B) \\ &= (\neg y_B \wedge \rho_0) \oplus (y_B \wedge (\rho_0 \oplus x_A)) \\ &= (x_A \wedge y_B) \oplus \rho_0 \end{aligned}$$

De tweede keer kiest Alice een random bit ρ_1 en biedt ze de bits ρ_1 en $\rho_1 \oplus y_A$ aan, waaruit Bob selecteert met x_B; de uitkomst u_1 van $OT(\rho_1, \rho_1 \oplus y_A; x_B)$ is $(y_A \wedge x_B) \oplus \rho_1$. Na de twee blinde overdrachten worden de nieuwe shares berekend als

$$\begin{aligned} z_A &= (x_A \wedge y_A) \oplus \rho_0 \oplus \rho_1 \\ z_B &= (x_B \wedge y_B) \oplus u_0 \oplus u_1 \end{aligned}$$

waarna de juistheid volgt uit

$$\begin{aligned} z &= z_A \oplus z_B && \text{Definitie van het schema} \\ &= ((x_A \wedge y_A) \oplus \rho_0 \oplus \rho_1) \oplus ((x_B \wedge y_B) \oplus u_0 \oplus u_1) && \text{Berekening door A en B} \\ &= (x_A \wedge y_A) \oplus \rho_0 \oplus \rho_1 \oplus (x_B \wedge y_B) \oplus \\ &\qquad (x_A \wedge y_B) \oplus \rho_0 \oplus (y_A \wedge x_B) \oplus \rho_1 && \text{Invullen } u_0, u_1 \\ &= (x_A \wedge y_A) \oplus (x_B \wedge y_B) \oplus (x_A \wedge y_B) \oplus (y_A \wedge x_B) && \text{Elimineren } \rho_0, \rho 1 \\ &= (x_A \oplus x_B) \wedge (y_A \oplus y_B) && \text{Distributie} \\ &= x \wedge y && \text{Definitie } x, y \end{aligned}$$

Andere connectieven: disjunctie en implicatie. Voor de andere logische connectieven kunnen protocollen worden ontworpen, maar nodig is dat niet omdat alle connectieven die er zijn al kunnen worden uitgedrukt in $\neg$ en $\wedge$:

$$\begin{aligned} x \vee y &= \neg(\neg x \wedge \neg y) \\ x \Rightarrow y &= \neg(x \wedge \neg y) \\ x \Leftrightarrow y &= \neg(\neg(x \wedge y) \wedge \neg(\neg x \wedge \neg y)) \end{aligned}$$

Volgens de laatste regel kan het protocol voor $\Leftrightarrow$ worden gemist, maar het bovenbeschreven protocol is erg simpel en het is inefficiënt om het te vervangen door een protocol dat drie $\wedge$-evaluaties nodig heeft.

Het complete protocol. Om een functie f te berekenen moeten Alice en Bob niet alleen f kennen, maar ook een bepaalde logische formule afspreken die f implementeert. Vervolgens delen ze ieder hun invoer, rekenen stap voor stap de tussenwaarden in de formule uit op de aandelen, en reconstrueren het resultaat.

De complexiteit (hoeveelheid werk) is evenredig met het aantal operaties in de functie f. Door de verdeling van de bits is de privacy van de invoeren perfect beschermd tegen nieuwsgierige medespelers. In de geschetste opzet is er echter geen bescherming tegen malafide spelers: waarden kunnen worden veranderd door valsspelers, al weten die spelers niet wat de oude en nieuwe waarde dan zijn.

14.2 Homomorfe encryptie

Een modernere aanpak is om invoerwaarden en tussenresultaten niet te verhullen door verdeling, maar door encryptie. Invoerwaarden worden versleuteld, waarna de versleutelde invoer publiek wordt gemaakt; op de publieke invoerwaarden en tussenresultaten worden de rekenstappen van f uitgevoerd. Ten opzichte van verdeling biedt encryptie de volgende twee grote voordelen.

1. *Het is efficiënter.* Bij verdeling wordt waarde x verdeeld over een aantal aandelen x_i, die allemaal in de berekening moeten worden betrokken. De versleuteling resulteert in een enkele waarde, waardoor de berekening compacter kan zijn.

2. *Het is controleerbaarder.* Bij het rekenen met een verdeling moet elke partij zijn aandelen geheimhouden en het is dus niet gemakkelijk te zien of hij er wel eerlijk mee rekent. Dit is soms op te lossen met publieke aandelen (voorbeelden hiervan zijn in hoofdstuk 9 te vinden), maar dit gaat weer ten koste van extra werk. De versleutelde invoeren worden publiek gemaakt, waarna de hele berekening voor iedereen controleerbaar in het openbaar gebeurt. Bij het rekenen met versleutelde waarden kan er altijd alleen bij het inleveren van de invoer worden gesmokkeld.

14.2.1 Groepscryptografie

Om zeker te zijn dat alleen het eindresultaat, en geen invoerwaarden of tussenresultaten, wordt ontsleuteld, moet gebruik worden gemaakt van groepscryptografie. Bij het gebruik van verdeelschema's bestaat deze zorg niet: eerlijke deelnemers zullen slechts eenmaal meewerken aan een reconstructie, en daarbij hun aandelen van de einduitkomst gebruiken. Bij het gebruik

van versleuteling zijn alle invoeren en tussenresultaten voor iedereen versleuteld te zien, en daarom is het onacceptabel dat een partij over de decryptiesleutel beschikt. Zelfs als deze partij niet malafide maar alleen nieuwsgierig is, weerhoudt niets hem ervan om invoerwaarden te ontsleutelen.

Er moet daarom groepscryptografie worden gebruikt, waarbij de deelnemers alleen samen een ontsleuteling kunnen uitvoeren. Wanneer de sleutels worden aangemaakt door een centrale partij, moeten alle deelnemers zich ervan kunnen overtuigen dat de geheime sleutel wordt gewist nadat aandelen ervan over de deelnemers zijn verdeeld. Meestal wordt de sleutelgeneratie door een gemeenschappelijk protocol door de deelnemers gemaakt.

Uiteraard is er de keuze tussen een vetoschema en een drempelschema. Een vetoschema is alleen te gebruiken wanneer het zeker is dat alle deelnemers tot het eind blijven functioneren.

14.2.2 Homomorfe eigenschap

De gebruikte encryptie moet de eigenschap hebben dat bewerkingen op de klare tekst kunnen worden uitgevoerd op de cijfertekst. Zowel het RSA-schema als ElGamals algoritme zijn *multiplicatief*, dat wil zeggen, dat een encryptie van $x_1 \cdot x_2$ eenvoudig verkregen kan worden uit encrypties van x_1 en x_2.

Voor RSA is deze eigenschap het eenvoudigst te bewijzen. Immers, is $y_1 = x_1^e$ en $y_2 = x_2^e$, dan is $y_1 \cdot y_2 = x_1^e \cdot x_2^e = (x_1 \cdot x_2)^e$. Bij de ontsleuteling van $y_1 \cdot y_2$ vindt men dan $x_1 \cdot x_2$. Door het vermenigvuldigen van de twee cijferteksten $(y_1 \cdot y_2)$ worden dus ook de achterliggende klare teksten met elkaar vermenigvuldigd. Bij het vermenigvuldigen blijven de operanden (x_1 en x_2) en het resultaat onzichtbaar.

Hoewel het RSA-algoritme de gewenste homomorfe eigenschap heeft, is het voor secure computing niet bruikbaar, en de reden hiervoor is dat er voor elke klare tekst maar één cijfertekst mogelijk is. De encryptie beeldt x deterministisch af op x^e. In veel toepassingen komt de invoer van een deelnemer uit een vrij kleine verzameling van mogelijkheden: een bit kan alleen 0 of 1 zijn; bij een stemming kan men voor- of tegenstemmen, of soms een keuze maken uit een tiental partijen; de waarde van een villa kan in centen worden uitgedrukt, maar voor panden tot tien miljoen euro zijn er dan nog 'slechts' een miljard bedragen mogelijk.

Wanneer een deelnemer P_i (met invoer w_i) de waarde $y_i = w_i^e$ publiceert, kan iedereen w_i achterhalen met een brutekrachtaanval over de invoerverzameling. Alle mogelijke invoeren w worden bekeken, tot er een gevonden wordt die voldoet aan $w^e = y_i$.

Probabilistische encryptie: ElGamal. Het ElGamal-algoritme werkt probabilistisch. Net als in eerdere hoofdstukken noemen we de geheime sleutel a en de publieke sleutel $b = g^a$. Tijdens encryptie van x wordt een random getal k gekozen, en de cijfertekst is (u, v) met $u = g^k$ en $v = x \cdot b^k$, en de decryptie van (u, v) is $v \cdot (u^a)^{-1}$.

Door de randomisering zijn er bij één klare tekst vele mogelijkheden voor de resulterende cijfertekst. Het algoritme verschilt nu hierin van RSA, dat het niet alleen onmogelijk is de klare tekst te *vinden* uit de cijfertekst, maar het is ook onmogelijk, voor een klare tekst x_0, te *herkennen* of de cijfertekst x_0 verbergt.

Stelling 14.1 (aanname 6.9) *Het is, gegeven een ElGamal-cijfertekst (u, v) en een klare tekst x_0, ondoenlijk te bepalen of bij (u, v) de klare tekst x_0 hoort.*

Bewijs. De cijfertekst (u, v) is een encryptie van x_0 als er een k bestaat waarvoor $u = g^k$ en $v = (b^k) \cdot x_0$, met andere woorden, als $DifHel_g(u, b, v/x_0)$ geldt (zie definitie 6.8). Een procedure die voor een willekeurige cijfertekst en klare tekst beslist of dit het geval is, kan

als volgt worden gebruikt om voor willekeurige getallen α, β, γ te bepalen of *DifHel*(α, β, γ) geldt. Neem als publieke sleutel $b = \beta$, neem $u = \alpha$ en $v = \gamma \cdot x_0$. Als de cijfertekst (u, v) (onder publieke sleutel b) de boodschap x_0 bevat geldt *DifHel*(α, β, γ) en omgekeerd. Volgens aanname 6.9 is het niet mogelijk dit te beslissen. □

Het algoritme van ElGamal is, net als RSA, multiplicatief:

Stelling 14.2 *Laat* (u_1, v_1) *en* (u_2, v_2) *versleutelingen zijn van* x_1*, respectievelijk* x_2 *(onder dezelfde publieke sleutel* b*). Dan is* (v_1v_2, u_1u_2) *een versleuteling van* x_1x_2*.*

Bewijs. Er zijn k_1 en k_2 zodat $u_1 = g^{k_1}$ en $v_1 = b^{k_1} \cdot x_1$, en $u_2 = g^{k_2}$ en $v_2 = b^{k_2} \cdot x_2$. Dan is $u_1u_2 = g^{k_1} \cdot g^{k_2} = g^{k_1+k_2}$, en $v_1v_2 = b^{k_1} \cdot x_1 \cdot b^{k_2} \cdot x_2 = b^{k_1+k_2} \cdot x_1x_2$. Als k_1 en k_2 random waren gekozen, is $k_1 + k_2$ weer een random getal. □

Exponentiële codering. Het algoritme van ElGamal is volgens de vorige stelling *multiplicatief*, dat wil zeggen dat klare teksten kunnen worden *vermenigvuldigd* door een bewerking op de cijferteksten. (Dat die bewerking zelf ook vermenigvuldiging is, is voor dit punt minder belangrijk.) Voor de meeste problemen is echter het *optellen* van beschermde waarden belangrijker en het optellen van twee klare teksten kan bij ElGamal *niet* rechtstreeks op de cijfertekst worden gedaan.

Om getallen te kunnen optellen wordt daarom een *exponentiële codering* toegepast, waarbij een getal w wordt gecodeerd als h^w. Hier is h een vast element van orde q. Door de exponentiële codering wordt het optellen van waarden vervangen door het vermenigvuldigen van hun coderingen: immers als $x_1 = h^{w_1}$ en $x_2 = h^{w_2}$, dan is $x_1 \cdot x_2 = h^{w_1+w_2}$. Deze vermenigvuldiging kan op cijferteksten worden gedaan, dat wil zeggen, de getallen x_1 en x_2 worden met ElGamal versleuteld en de vermenigvuldiging verloopt volgens stelling 14.2.

Ook aftrekken kan op cijferteksten worden gedaan. Als $(u_1, v_1) = (g^{k_1}, b^{k_1} \cdot h^{w_1})$ en $(u_2, v_2) = (g^{k_2}, b^{k_2} \cdot h^{w_2})$ versleutelde coderingen zijn van w_1 respectievelijk w_2, dan is (u_1u_2, v_1v_2) een versleutelde codering van $w_1 + w_2$, (u_1^{-1}, v_1^{-1}) een versleutelde codering van $-w_1$, en $(u_1u_2^{-1}, v_1v_2^{-1})$ een verleutelde codering van $w_1 - w_2$.

Het eindresultaat van de berekening is niet alleen versleuteld (met ElGamal), maar ook exponentieel gecodeerd. Anders gezegd, de ontsleuteling van de eindwaarde (u, v) van de berekening levert een getal x^*, dat gelijk is aan h^{w^*}, waar w^* het antwoord van de berekening is. De berekening van w^* uit x^* is het berekenen van de discrete logaritme van x^* met grondtal h. Hoewel de discrete logaritme in het algemeen niet te berekenen is (conform aanname 6.6), kan het wel als het aantal mogelijke uitkomsten van de berekening klein is. Van w^* is dan bekend dat deze tussen 0 en M ligt; bijvoorbeeld bij een stemming onder een miljoen stemgerechtigden is het aantal voorstemmers een getal tussen nul en een miljoen.

Bij een klein uitvoerbereik is het mogelijk voor elk getal w te testen of $h^w = x^*$, en zodra dit het geval is, is w^* gevonden; dit kost ongeveer M vermenigvuldigingen. De *grote stappen, kleine stappenmethode* van Shanks (sectie 15.3.1) vindt het antwoord in ongeveer $\sqrt{M}$ rekenstappen.

14.2.3 Nogmaals de villabewoners

In sectie 9.1.2 werd het probleem van de villabewoners, die de gezamenlijke waarde van hun woningen willen berekenen, opgelost met een homomorf verdeelschema. Om de mechanismen uit de vorige sectie in werking te zien, bekijken we nu een oplossing gebaseerd op homomorfe encryptie. Voor de groepscryptografie wordt een vetoschema gebruikt en de geheime sleutel

wordt in een protocol (volgens sectie 9.2.4) bepaald. De deelnemers zijn de bewoners P_1 tot en met P_n, en we nemen aan dat er al een groot priemgetal p en een element $g \in \mathbb{Z}_p^*$ van orde q bekend zijn.

1. *Keuze van h.* Er wordt een getal h van orde q gekozen.
2. *Sleutelgeneratie, keuze van aandeel.* Bewoner P_i kiest een random getal a_i als aandeel in de geheime sleutel a, die nu impliciet is gedefinieerd als $a = \sum_j a_j$.
3. *Sleutelgeneratie, berekenen van publieke sleutel.* Bewoner P_i berekent $b_i = g^{a_i}$ en maakt b_i bekend; als elke bewoner dit heeft gedaan is de publieke sleutel te berekenen als $b = \prod_j b_j$.
4. *Codering en versleuteling van invoer.* Bewoner i codeert de waarde w_i van zijn huis (in euro) als $x_i = h^{w_i}$. Hij versleutelt x_i met b door een random k_i te kiezen en te berekenen $u_i = g^{k_i}$ en $v_i = b^{k_i} \cdot x_i$.
5. *Bekendmaking van versleutelde invoer.* Bewoner i maakt het paar (u_i, v_i) bekend.
6. *Berekening.* Als elke bewoner zijn versleutelde waarde heeft bekendgemaakt, wordt (u^*, v^*) berekend als $u^* = \prod_j u_j$ en $v^* = \prod_j v_j$.
7. *Ontsleuteling, gebruik van aandeel.* De bewoners moeten samen (u^*, v^*) ontsleutelen, en geen andere waarde. Bewoner i berekent met zijn aandeel a_i van de geheime sleutel de waarde $z_i = (u^*)^{a_i}$ en maakt deze bekend.
8. *Ontsleuteling, vormen van x^*.* Als elke bewoner het getal z_j heeft bekendgemaakt, wordt het product z berekend als $z = \prod_j z_j$, en x^* als $x^* = v^* \cdot z^{-1}$.
9. *Decodering.* De uitkomst w^* wordt berekend als de logaritme van x^*. Als de totale waarde van de huizen minder dan een miljard euro is, kost deze berekening met Shanks' algoritme ongeveer zestigduizend vermenigvuldigingen.

Steeds waar berekeningen op publieke waarden worden gedaan (stappen 3, 6, 8 en 9) kunnen een of enkele bewoners worden gevraagd deze te doen. Iedereen die dat wil kan de resultaten narekenen.

Omdat de feitelijke berekening op publieke waarden wordt uitgevoerd (stap 6) is de eerlijkheid van de deelnemers veel gemakkelijker te controleren. De enige stappen die in het bovenstaande protocol niet rechtstreeks controleerbaar zijn, zijn het inleveren van de invoer (stap 4 en 5) en de ontsleuteling met de eigen deelsleutel (stap 7). Voor beide stappen kan een zero-knowledge proof worden toegevoegd.

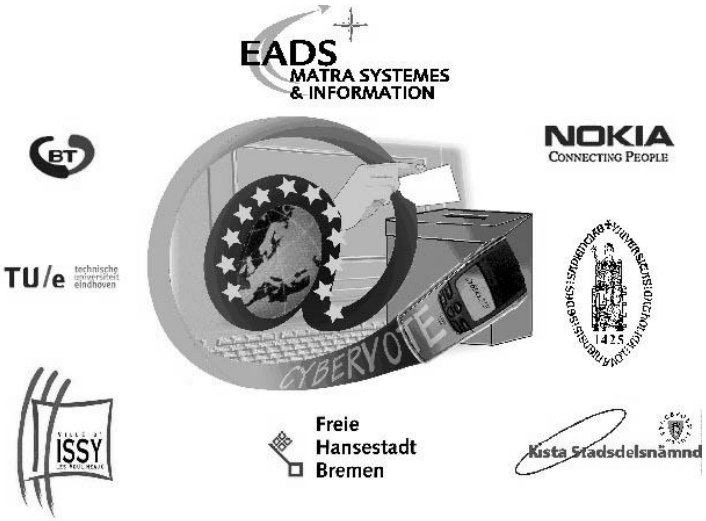
14.2.4 Vermenigvuldigen

Een berekening bevat vaak meer operaties dan alleen optellen of aftrekken; vaak moet worden vermenigvuldigd of zijn nog andere operaties op getallen nodig.

Het is niet moeilijk het verborgen getal x te vermenigvuldigen met een *bekend* getal s; hiertoe wordt het paar (u_x^s, v_x^s) gevormd, en dit paar verbergt $x \cdot s$. Immers, $u_x^s = (g^k)^s = g^{(ks)}$, en $v_x^s = (b^k \cdot h^x)^s = b^{(ks)} \cdot h^{xs}$.

Cramer, Darmgård en Nielsen [CDN01] geven een protocol waarmee ook vermenigvuldigingen kunnen worden uitgevoerd op cijferteksten. Gegeven cijferteksten van x en y, vormt het protocol een cijfertekst van $x \cdot y$.

Kader 14.2: Het CyberVote-project



De Europese Commissie hoopt de deelname aan verkiezingen te vergroten door elektronisch stemmen via internet en mobiele telefoons mogelijk te maken. Volgens Schalekamp [Sch02] is de animo voor verkiezingen het laagst bij bevolkingsgroepen die gemakkelijk met de nieuwe technologieën omgaan, zoals jongeren.
Bestaande cryptografische protocollen kunnen niet aan alle eisen voldoen: geheimhouding van individuele stemmen, en toch controleerbare zekerheid dat elke stem is meegeteld. Speciale protocollen worden ontworpen door een consortium met industriële (EADS Matra, Nokia, British Telecom) en academische (TU Eindhoven, KU Leuven) partners.
In het project worden ook juridische aspecten bekeken. Zo is in België het deelnemen aan verkiezingen verplicht, en dit levert een extra technische eis aan het systeem op, namelijk dat zichtbaar moet zijn wie er wel en niet hebben gestemd. Uiteraard lopen de kieswetten in de verschillende landen op diverse punten uit de pas. Het systeem moet in 2003 worden getest in Duitsland, Frankrijk en Zweden.
Zie `www.eucybervote.org`.

Uiteraard is het niet wenselijk y te ontsleutelen. Daarom wordt y niet rechtstreeks ontsleuteld, maar zo, dat de verhulling van y wordt omgezet van een versleuteling naar een additief verdeelschema. Anders gezegd, deelnemer P_i krijgt een getal s_i dat een additief aandeel is, dat wil zeggen $\sum_i s_i = y$. Dan kan deelnemer P_i met bovenstaande regel de versleutelde codering van $x \cdot s_i$ vormen, en deze waarden worden opgeteld tot een versleuteling van $x \cdot y$.

14.3 Geheime verkiezingen

Het houden van verkiezingen is een speciaal voorbeeld van secure computing: doel is immers te bepalen hoeveel kiezers een bepaald alternatief ondersteunen, terwijl geen kiezer hoeft te vertellen welk alternatief hij ondersteunt. Het organiseren van elektronische verkiezingen geldt binnen het gebied van secure computing als een van de meest uitdagende problemen. Dit komt door de grote belangen die bij verkiezingen op het spel staan en de veelheid aan partijen en juridische aspecten die erbij betrokken zijn.

14.3.1 Eigenschappen van protocollen voor verkiezingen

De afgelopen twintig jaar zijn veel protocollen voor elektronische verkiezingen bedacht, en er komen nog steeds nieuwe bij. Er zijn voor deze protocollen erg veel eisen te formuleren, en er zijn nog geen protocollen bekend die alle wensen in zich verenigen.

Allereerst bekijken we welke partijen bij verkiezingen een rol spelen. Om te beginnen natuurlijk de kiezers, die hun keuze mogen maken uit twee of meer alternatieven; soms alleen maar voor of tegen, soms een keuze uit een lijst van politieke partijen of personen. Het aantal kiezers is l. De kandidaten zelf spelen in de meeste protocollen geen rol, en blijven in deze sectie buiten beschouwing. De verkiezing vindt plaats onder verantwoordelijkheid van een centraal orgaan, hier de overheid genaamd. De overheid creëert en beheert voor de verkiezing een of meerdere stembureaus, waarmee de kiezers moeten interacteren om hun stem uit te brengen. Het aantal stembureaus is n. Ten slotte onderscheiden we waarnemers: dit zijn partijen die niet aan de verkiezingen deelnemen, maar die moeten controleren of de verkiezing eerlijk verloopt.

Een verkiezingsprotocol moet voldoen aan zoveel mogelijk van de volgende eisen.

1. **Bevoegdheid:** Alleen de daartoe aangewezen personen, de kiezers, kunnen een stem uitbrengen.

   Hiertoe moet de overheid beschikken over een lijst van stemgerechtigden en, wanneer de verkiezing over een netwerk plaatsvindt, over een manier om kiezers over afstand te identificeren.

2. **Eenduidigheid.** Elke kiezer kan slechts één stem laten meetellen.

3. **Geheimhouding.** De stem van een kiezer blijft voor elke andere partij verhuld.

   Zoals veel cryptografische eisen kan geheimhouding in perfecte of computationele zin gelden en in dit geval kan het onderscheid van groot belang zijn. Er is sprake van *perfecte geheimhouding* indien, gegeven de informatie die een derde partij over het stemmen door een bepaalde kiezer heeft, elke toegestane waarde van de stem mogelijk is. Er is sprake van *computationele geheimhouding* indien het, gegeven de informatie die een derde partij over het stemmen door een bepaalde kiezer heeft, ondoenlijk is de waarde van diens stem te berekenen.

   Bij computationele veiligheid moet men bedenken dat een berekening die nu ondoenlijk is, over dertig of vijftig jaar misschien wel uitvoerbaar is. De informatie over de stemming kan worden bewaard totdat cryptografie met de gebruikte sleutellengte te kraken is. Omdat het niet ongebruikelijk is dat mensen nog dertig of vijftig jaar na dato worden aangesproken op hun politieke keuzen (of die van hun voorouders) moet hiermee rekening worden gehouden bij het bepalen van de sleutellengte. Niet alleen mag de sleutel met de techniek van het moment niet te kraken zijn, maar de sleutel moet nog minstens vijftig jaar veilig blijven.

   De geheimhouding bij klassieke stembusverkiezingen is tijdloos; de stembiljetten worden vernietigd en nooit zal iemand erachterkomen wat een bepaalde kiezer heeft gestemd. Een protocol dat aan perfecte geheimhouding voldoet heeft deze eigenschap ook. Zelfs als een CD-ROM met verkiezingsgegevens over duizend jaar wordt opgegraven, en ongeacht hoe ver de techniek dan is voortgeschreden, is het onmogelijk de stem van bepaalde kiezers te berekenen.

4. **Correctheid.** De einduitslag representeert het aantal uitgebrachte stemmen; niemand kan stemmen veranderen, toevoegen of verwijderen.

Ook correctheid kan in perfecte of computationele zin gelden. Vaak is een perfect geheim protocol slechts computationeel correct en een computationeel veilig protocol, perfect correct; een hogere correctheid gaat dus ten koste van het niveau van geheimhouding.

Omdat de uitslag van de verkiezing binnen enkele uren of dagen wordt opgemaakt en vastgesteld, is het belang van perfecte correctheid beperkt. Bij een computationeel correct protocol heeft een aanvaller immers maar een beperkte tijd beschikbaar om het resultaat te beïnvloeden, en hij moet dit met de techniek van het moment doen.

5. **Verifieerbaarheid.** Het is voor waarnemers controleerbaar dat de einduitslag correspondeert met de keuzes van de kiezers.

6. **Vrijheid van dwang.** Het is niet mogelijk, een kiezer onder druk te zetten of om te kopen om een bepaalde keuze te maken.

   De technische implementatie van deze wens is dat het voor kiezers niet mogelijk is tegenover derden te bewijzen welk alternatief zij hebben gekozen. Stel nu dat Alice (onder dwang van geweld of drang van geld) Bob onder druk zet om voor alternatief X te stemmen. Bob kan zeggen hieraan toe te geven, maar omdat het protocol geen bewijs van stemgedrag afgeeft kan Bob rustig op alternatief Y stemmen en tegen Alice beweren dat hij op X heeft gestemd.

   Kruijswijk [Kru02] meent dat deze bescherming onvoldoende is. Bij alle elektronische stemmingen via internet, mobiele telefoons of onbemande stemmachines is het mogelijk, samen te stemmen. In het gegeven voorbeeld kan Alice dus het uitbrengen van de stem door Bob bijwonen en erop toezien dat hij voor X stemt. Een omkoop- of afpersscenario lijkt hier vergezocht, maar volgens Kruijswijk kan samen stemmen gemakkelijk sociaal worden afgedwongen in een gezin (waar een van de partners, meestal de man, overheersend is) of in een religieuze gemeenschap. Door de sociale druk binnen gezin of gemeenschap is de stemming niet vrij en Kruijswijk beargumenteert dat het stemmen in afzondering niet alleen een recht, maar ook een grondwettelijke plicht is. Volgens deze argumentatie is vrijheid van dwang niet te verkrijgen in een systeem dat vanaf willekeurige lokaties te gebruiken is.

7. **Opkomstcontrole.** Stembureaus en waarnemers kunnen zien, wie van de kiezers wel en niet hebben gestemd.

   In sommige landen (waaronder België) is het deelnemen aan verkiezingen verplicht, en het protocol moet controle op deze verplichting mogelijk maken.

8. **Gebruikswensen.** Het stemmen is gebruiksvriendelijk. Stemmen kan vanaf verschillende platforms (speciale machines, pc, mobiel). Het protocol is efficiënt. Een kiezer kan zijn stem na het uitbrengen nog herzien.

Het tellen van stemmen is computationeel gezien een zeer eenvoudig proces. Het is daarom goed mogelijk, verkiezingsprotocollen te ontwerpen volgens de algemene technieken uit sectie 14.1.3 of 14.2.

14.3.2 Schema van Cramer, Gennaro en Schoenmakers

Het verkiezingsprotocol van Cramer, Gennaro en Schoenmakers [CGS97] is gebaseerd op homomorfe groepscryptografie. De berekeningswijze lijkt op het protocol voor de villabewoners uit de vorige sectie, met de nodige aanpassingen:

Kader 14.3: VoteHere.net

VoteHere is een Amerikaans bedrijf dat (sinds 1996) allerlei oplossingen voor elektronisch stemmen verkoopt. Het bedrijf levert stemmachines die lokaal stemmen tellen, maar ook stemmachines die via netwerken de stemmen doorsturen en oplossingen die volledig elektronisch zijn, zodat de kiezer vanaf de eigen pc aan de stemming kan deelnemen.
De producten van VoteHere kunnen bijvoorbeeld binnen een bedrijf of vereniging worden gebruikt (VoteHere Gold) maar uitgebreidere systemen voor politieke verkiezingen (VoteHere Platinum) zijn ook leverbaar. De techniek van stemmingen per internet is gebaseerd op homomorfe groepscryptografie.
Zie `www.votehere.net`.

- De sleutel is niet verdeeld onder de deelnemers (de villabewoners danwel kiezers), maar over de stembureaus.
- Om een storing of fraude in stembureaus te kunnen ondervangen wordt gebruikgemaakt van een drempelschema voor de geheime sleutel.
- Om stemfraude tegen te gaan wordt aan elke inzending een zero-knowledge proof van juistheid toegevoegd.

Voor het coderen van stemmen wordt een getal G van orde q gebruikt, en een voor- of tegenstem wordt als G^1, respectievelijk G^{-1} gecodeerd.

De verkiezing in detail. De verkiezing begint met de gedistribueerde sleutelgeneratie door de stembureaus; de methode van sectie 9.3.4 wordt gebruikt, zie ook Pedersen [Ped91]. De uitkomst van het protocol is dat er een publieke ElGamal-sleutel b bekend is, en de geheime sleutel is volgens het polynoomschema verdeeld over de stembureaus. Bij elk aandeel a_i hoort een publiek aandeel $b_i = g^{a_i}$.

Nu b bekend is kunnen kiezers hun voorkeur aangeven. Een voor- of tegenstem wordt met $w = +1$, respectievelijk $w = -1$ aangegeven en met een random k versleuteld tot $(g^k, b^k \cdot G^w)$. Bij dit paar wordt een niet-interactief bewijs gemaakt dat de inhoud van het bericht $+1$ *of* -1 is, maar uiteraard zonder dat aan het bewijs te zien is welke waarde het is. Voor details van dit bewijs verwijzen we naar Cramer e.a. [CGS97]. Vervolgens ondertekent de kiezer de versleutelde stem en het bewijs met zijn eigen geheime sleutel, en stuurt het naar een publieke locatie.

Alleen stemmen die naar de publieke locatie worden verstuurd, worden meegeteld. Van alle stemmen wordt de handtekening gecontroleerd en gekeken of de afzender wel in de kieslijst voorkomt (voor bevoegdheid). Meerdere stemmen van dezelfde kiezer (te herkennen aan de handtekening) worden verwijderd. Als de tijd van stembussluiting is aangebroken, worden de versleutelde stemmen geteld; dit komt neer op het optellen van de versleutelde getallen en werkt volgens stelling 14.2.

De stembureaus werken nu samen om het resultaat (u^*, v^*) te ontsleutelen. Bureau i met aandeel a_i van de sleutel moet de waarde $(u^*)^{a_i}$ insturen, samen met een bewijs dat de waarde correct is berekend. Hoe dit bewijs eruitziet is in protocol 9.13 aangegeven. De waarden van bureaus die geen of een onjuist bewijs leveren worden genegeerd. Ontsleuteling resulteert in een getal x^*, waaruit de verkiezingsuitslag als de logaritme wordt gevonden. Hoewel de discrete logaritme in het algemeen niet te bepalen is, kan dat hier wel omdat de

Kader 14.4: Fraude bij internetstemming in Vlietstede

Fraude met internetstemmen gemeentenaam

Metro, 17/10/01

De fusiegemeenten Leidschendam en Voorburg hebben de internet-peiling voor een nieuwe naam gestaakt wegens stemfraude.

Inwoners die hun voorkeur kenbaar willen maken, moeten nu hun naam en adres opgeven. "Dat kunnen wij makkelijk controleren", aldus de woordvoerster.(ANP)

Dat het niet makkelijk is om een eerlijke en geheime elektronische stemming te houden bleek in oktober 2001 pijnlijk in de gemeenten Voorburg en Leidschendam. Deze gemeenten werden gefuseerd en inwoners konden via een website hun voorkeur voor de naam van de fusiegemeente kenbaar maken. Er kwam echter een onwaarschijnlijk groot aantal stemmen binnen, waarna de website werd gesloten en de peiling ongeldig werd verklaard.
In zulke gevallen is de privacy meestal het eerste slachtoffer; de stemming werd per e-mail voortgezet, waarbij naam en adres moesten worden opgegeven. Hierdoor kan worden gecontroleerd dat iedereen maar één keer stemt, maar de stemming is niet langer geheim.
De fusiegemeente zal Vlietstede gaan heten.
Ook in 2006 woedt de discussie over de veiligheid van elektronisch stemmen nog volop.

uitkomst in een vrij klein bereik ligt. De waarde ligt tussen $-l$ en $+l$, en kan eenvoudig met ongeveer l vermenigvuldigingen (dus één per kiezer) worden gevonden. Het algoritme van Shanks (hoofdstuk 15) kan de logaritme zelfs in $O(\sqrt{l})$ stappen bepalen.

Eigenschappen van het protocol. Door de controle op de handtekening onder de stemmen wordt bevoegdheid en eenduidigheid verkregen. Ook is hierdoor eenvoudige controle op opkomst mogelijk. Het bewijs van de juistheid van een stem voorkomt dat kiezers een waarde anders dan $+1$ of -1 kunnen stemmen.

Geheimhouding wordt gerealiseerd door de versleuteling van de stem en is niet perfect, maar computationeel. De stemmen verschijnen op een publieke plaats, waardoor iedereen de stemmen kan opslaan en bewaren tot ElGamal-encryptie met de gebruikte sleutellengte te kraken is.

Het stemmen is gemakkelijk, aangezien slechts het inzenden van een enkel bericht nodig is om een stem uit te brengen. Het protocol is ook zo aan te passen dat een kiezer zijn stem nog kan herzien tot de sluitingsdatum: de regel wordt ingevoerd dat als er meerdere stemmen van een kiezer zijn, de laatst uitgebrachte telt.

Het protocol realiseert geen vrijheid van dwang. Een kiezer kan immers tegenover een afperser of omkoper de waarde van k, waarmee hij zijn stem heeft versleuteld, onthullen en zo bewijzen wat hij heeft gestemd.

14.3.3 Permutatieschema

Bij het protocol in sectie 14.3.2 is van elke uitgebrachte stem te zien, door wie hij is uitgebracht, maar door de encryptie is de stem niet te lezen. Bij traditionele verkiezingen met papier en stembus is de situatie juist andersom: elke stem is direct leesbaar, maar aan de stemmen in de stembus is niet te zien wie deze stem heeft uitgebracht. De verkiezingsprotocollen van Chaum [Cha81, Cha88] gebruiken dit principe in een cryptografische context.

Een kiezer vult een stembiljet in, en laat dit blind ondertekenen door een van de stembureaus. De stembiljetten worden *anoniem* naar een stembureau gestuurd.

De verkiezing in detail. De overheid zet twee stembureaus op, namelijk een *validatiebureau* en een *telbureau*. Het validatiebureau krijgt de lijst van stemgerechtigden (met hun publieke sleutel) en begint met het genereren van een RSA-sleutelpaar. Het publieke deel (n, e) wordt openbaar gemaakt en is de sleutel waarmee stembriefjes moeten worden ondertekend.

Als deze sleutel bekend is gemaakt kunnen kiezers een stembiljet invullen en laten ondertekenen. De kiezer die op alternatief v wil stemmen, vormt het bericht $x = \langle\, \mathbf{stem}, r, v \,\rangle$, waar r een random getrokken getal is. Vervolgens laat de kiezer het bericht blind ondertekenen door het validatiebureau, als in sectie 7.4.2. De kiezer neemt een random getal k en vormt $x' = x \cdot k^e$. Nu identificeert de kiezer zich tegenover het validatiebureau en verkrijgt hij het getal $y' = x'^d$. De kiezer berekent $y = y' \cdot k^{-1}$ en beschikt over een getal y dat voldoet aan $y^e = x$.

Het getal y wordt nu naar het telbureau gestuurd, maar dit moet *anoniem* gebeuren. Dat betekent, dat de berichten die het telbureau ontvangt niet zijn te traceren naar hun afzender. Dit is te vergelijken met stembriefjes in een bus te stoppen: er is dan misschien te zien wie een briefje inlevert, maar na het schudden van de bus is niet meer te zien welk briefje van welk persoon kwam. De stemmen worden via een reeks van tussenstations verstuurd, waarbij elk station de berichten permuteert.

Het telbureau controleert of alle stemmen geldig zijn ondertekend: van elke ingeleverde y wordt y^e berekend, en de uitkomst moet een getal zijn van de vorm $\langle\, \mathbf{stem}, r, v \,\rangle$. Geldige stemmen worden gepubliceerd en geteld, waarbij duplicaat-serienummers maar één keer meetellen.

Eigenschappen van het protocol. Een stem kan alleen meetellen als hij is getekend, en elke kiezer kan hoogstens één stem laten ondertekenen; dit geeft bevoegdheid en eenduidigheid.

De geheimhouding wordt door de interactie met het validatiebureau volstrekt niet bedreigd. De blindering van stemmen is perfect veilig, en het is dus onmogelijk, zelfs als het RSA-systeem gebroken wordt, om uit informatie van het validatiebureau te achterhalen wat iemand heeft gestemd. Bij oppervlakkige beschouwing lijkt het protocol daarom een perfecte geheimhouding te geven.

Echter, wat met stembriefjes de eenvoudigste zaak van de wereld is, namelijk het door elkaar schudden, is met cryptografische berichten verre van eenvoudig. Chaum [Cha81] laat de stemmen, die eerst meervoudig worden versleuteld, via een aantal tussenstations lopen. Elk station past eenmaal ontsleuteling toe (zodat het aantal toegepaste versleutelingen met één afneemt) en permuteert de berichten. De anonimiteit van een ingezonden stem kan op twee manieren worden aangevallen. (1) Een buitenstaander kan van een bepaalde kiezer proberen de meervoudige encryptie te doorbreken; omdat er public-keycryptografie is gebruikt is de verhulling ten hoogste computationeel. (2) De tussenstations kunnen door samenwerking de stem van een bepaalde kiezer volgen tot in het telbureau, of een bepaalde stem in het telbureau traceren naar de kiezer. Door het gebruik van de publieke sleutels van alle tussenstations is het anoniem inzenden omslachtig, en door het herhaald doorsturen is het inefficiënt in communicatie.

Het protocol als geheel is maar beperkt verifieerbaar, en met name de twee bureaus hebben mogelijkheden om te frauderen. Het validatiebureau kan extra stemmen ondertekenen, die verder niet van stemmen van kiezers te onderscheiden zijn; wanneer niet alle stemgerechtigden opkomen valt dit niet eens op. Een onafhankelijk waarnemer kan niet controleren of alle

ingestuurde stemmen ook op de lijst staan; het telbureau kan bepaalde stemmen niet opnemen, en zeggen dat er minder stemmen zijn ingestuurd dan gevalideerd. Een kiezer kan in de lijst van gepubliceerde stemmen wel zien of zijn eigen stem is meegeteld. Als dit niet het geval is kan hij het meetellen van zijn eigen stem alsnog afdwingen door hem te onthullen, maar dit verbreekt de geheimhouding.

Stemmen is meer werk dan in het vorige protocol, omdat de kiezer achtereenvolgens met twee bureaus moet communiceren. Het herzien van stemmen is niet mogelijk. Ten slotte kan als nadeel worden genoemd, dat tussenresultaten van de stemming bij het telbureau kunnen worden uitgelezen. Het is een bekend sociologisch verschijnsel dat tussenresultaten de verdere verkiezing kunnen beïnvloeden, zodat er voor het uitlekken van de tussentelling moet worden gewaakt zolang het inzenden van stemmen nog mogelijk is.

Het permutatieschema is ouder dan oplossingen met homomorfe encryptie. Met dit protocol werd een begin gemaakt van de studie naar verkiezingsprotocollen en, hoewel het enkele interessante technische ingrediënten bevat, kleven er voor de meeste toepassingen te veel bezwaren aan.

14.4 Voorbij functie-evaluatie

Niet alle interacties tussen partijen kunnen worden beschreven als het berekenen van een functie. In deze sectie kijken we naar enkele problemen die met de beschreven technieken niet rechtstreeks kunnen worden opgelost.

14.4.1 Blinde overdracht

Onder *blinde overdracht* (*oblivious transfer*) worden enkele communicatieproblemen verstaan, waarbij een zender een of meer berichten heeft die de ontvanger kan zien, maar de zender weet niet welk bericht de ontvanger ziet.

Overdrachtsproblemen. In sectie 7.4.4 kwam al een overdrachtsprobleem ter sprake, namelijk bij het opzetten van een discrete geheimenwinkel. Daar kon Bob een geheim kopen zonder te laten weten welk geheim hij kocht. Nu behandelen we overdracht in een meer elementaire vorm, namelijk waar er hoogstens twee geheimen zijn.

In het *keuzeoverdracht-probleem* heeft de zender twee berichten, x_0 en x_1, die hij de ontvanger aanbiedt. De ontvanger heeft een bit s, de selectiebit. Als uitkomst van het protocol verkrijgt de ontvanger bericht x_s. De veiligheid voor de zender eist dat de ontvanger op geen enkele manier meer dan één bericht kan zien. De veiligheid voor de ontvanger eist dat de zender geen informatie krijgt over de selectiebit.

In de *random* versie van het probleem heeft de ontvanger geen selectiebit, maar een random keuze hiervoor wordt tijdens de uitvoering van het protocol gemaakt. De zender heeft dus weer twee berichten, en de ontvanger krijgt er hiervan één, maar kan niet kiezen welke. Veiligheid voor de zender is dat de ontvanger hoogstens één bericht ziet, maar dat hij niet kan beïnvloeden welk. Veiligheid voor de ontvanger eist dat hij tenminste één bericht ziet, dat de zender niet kan beïnvloeden welk bericht hij ziet, en ook niet kan achterhalen welk bericht hij ziet.

In de *random-succes* versie heeft de zender maar één bericht x, en de ontvanger krijgt dit met kans $\frac{1}{2}$ te zien. De zender weet niet of de transmissie geslaagd is, de ontvanger wel.

In elke versie kan de boodschap een enkele bit, of juist een reeks van meerdere bits zijn.

Protocollen voor keuzeoverdracht. Het volgende protocol kan worden gebruikt voor keuzeoverdracht. De zender versleutelt beide berichten x_0 en x_1 (aangevuld met een aantal random bits) onder RSA, en staat de ontvanger één blinde decryptie toe.

1. De zender kiest een RSA-sleutel $(N,\ e)$ met geheim deel d, en stuurt de ontvanger $y_0 = x_0^e$ en $y_1 = x_1^e$.

2. De ontvanger kiest een random k en stuurt de zender $y = y_s \cdot k^e$.

3. De zender antwoordt met $x = y^d$ (hetgeen de ontvanger kan controleren door te verifiëren dat $x^e = y$). De ontvanger berekent x_s als $x \cdot k^{-1}$.

De aanvulling met random bits voorkomt dat de ontvanger bijvoorbeeld $x_0 \cdot x_1$ te weten kan komen door $y_0 \cdot y_1$ blind te laten decrypten.

Met een kleine aanpassing kan hetzelfde idee voor random-keuzeoverdracht worden gebruikt: voorafgaand aan het versleutelen verwisselt de zender de berichten met kans $\frac{1}{2}$. Dat wil zeggen, de zender kiest een random bit r en berekent $y_i = x^e_{i \oplus r}$.

Als het protocol meerdere malen achtereen wordt uitgevoerd, moet de zender elke keer een nieuwe RSA-sleutel aanmaken. Wordt het meerdere malen met dezelfde sleutel uitgevoerd, dan kan de ontvanger kiezen om niet van elk paar één, maar van een paar beide berichten te zien.

Een protocol voor random-succesoverdracht. Voor random-succesoverdracht kan het volgende protocol (Rabin, 1981) worden gebruikt. De zender versleutelt zijn bericht x met Rabins kwadraatsysteem (sectie 6.3.5), en de ontvanger krijgt een kans van $\frac{1}{2}$ om de sleutel te vinden.

1. *Sleutelkeuze.* Zender kiest priemgetallen p en q en berekent $n = p \cdot q$. De zender is in staat vierkantswortels te berekenen in $\mathbb{Z}_n$.

2. *Eén aanvalspoging.* Ontvanger mag van één getal uit $\mathbb{Z}_n$ de wortel vragen. Ontvanger kiest een random u, berekent $w = u^2$ en vraagt hieruit een wortel. Met een kans van $\frac{1}{2}$ is het antwoord, v, niet gelijk aan u of $-u$, en in dat geval vindt ontvanger p en q (zie voor details sectie 6.3.5). De zender weet niet of de poging van de ontvanger is geslaagd.

3. *Verzending.* De zender berekent $y = x^2$ (eventueel wordt x aangevuld met een random getal) en stuurt het aan de ontvanger. De ontvanger kan x berekenen als hij in de vorige stap n heeft gefactoriseerd.

Bij succesvolle overdracht kent de ontvanger de factoren van n, en de zender weet niet of dat het geval is. Daarom moet ook hier bij herhaalde uitvoering van het protocol steeds een nieuwe sleutel worden gekozen.

14.4.2 Het contractprobleem

In hoofdstuk 7 werd besproken hoe een bericht kan worden ondertekend, zo, dat de handtekening de instemming van de ondertekenaar met de inhoud van het bericht bewijst. Voor het veilig ondertekenen van contracten (wederzijdse overeenkomsten) is meer nodig omdat de instemming van een partij afhankelijk is van de instemming van de andere partij.

Probleemstelling: gelijktijdig ondertekenen. Laten we aannemen dat Alice en Bob het na onderhandelingen eens zijn geworden over de inhoud van een contract waaraan zij zich beiden willen houden. Handtekeningen onder een contract dienen als bewijs voor de wilsovereenstemming op dit moment, voor het geval in de toekomst een van de partijen onder het contract uit wil.

Laten we het contract C noemen; het is niet voldoende als Alice en Bob elkaar eenvoudigweg hun handtekening over C sturen. Een partij, zeg Bob, kan nadat Alice haar handtekening heeft gestuurd, nalaten de zijne te sturen en verkeert in een sterke machtspositie. Als Alice later onder het contract uit wil, heeft zij geen kans omdat Bob over haar handtekening beschikt. Als Bob onder het contract uit wil kan dat heel gemakkelijk omdat Alice niet over Bobs handtekening beschikt.

We hebben een protocol nodig dat de handtekeningen gelijktijdig uitwisselt. Elke partij stuurt de handtekening in een reeks berichten, waarbij het sturen van de eigen handtekening direct wordt gestaakt wanneer van de andere zijde bedrog wordt bemerkt.

Even en Yacobi bewezen in 1981 dat een deterministisch protocol het probleem niet kan oplossen. In zo'n protocol worden de handtekeningen van Alice en Bob verdeeld over reeksen berichten $a_1, \ldots, a_m$ en $b_1, \ldots, b_m$, die beurtelings worden verstuurd. Er is echter in zo'n uitwisseling altijd een moment aan te wijzen, waarin een der partijen al aan het contract gebonden is, en de andere niet. Die laatste partij kan ervoor kiezen op dat moment de conversatie af te breken, en heeft dan wel een handtekening gekregen maar niet gegeven.

Verdeling in herkenbare aandelen. Even, Goldreich en Lempel [EGL82] gebruiken als principe dat handtekeningen bit voor bit worden uitgewisseld. Als een van de partijen de conversatie staakt, heeft hij hoogstens één bit meer van de handtekening dan de andere partij. De uitwerking van dit idee vereist de oplossing van nog een probleem, namelijk dat een partij, in plaats van bits van zijn handtekening, willekeurige bits kan sturen, en de tegenpartij komt er pas aan het eind achter dat hij genept is.

Daarom gaan we er, met blinde overdracht, voor zorgen dat de andere partij al een aantal bits kent; als een partij wil proberen valse bits te sturen, weet hij niet over welke bits hij 'veilig' kan liegen. Er worden hiertoe aandelen van de handtekening uitgegeven, waarbij de ene partij steeds met keuzeoverdracht over één aandeel komt te beschikken. Het originele protocol gebruikt een verdeling van de handtekening over een aantal (typisch honderd) aandelen. Net als in de beschrijving van Schneier [Sch96] van het protocol gaan we hier uit van een aantal verdelingen over twee aandelen.

Dat wil zeggen, dat Alice n paren van berichten vormt, $H_{0,1}$ en $H_{1,1}$ tot en met $H_{0,k}$ en $H_{1,k}$, waarbij steeds $H_{0,i}$ en $H_{1,i}$ samen een geldige handtekening over het contract vormen. De aandelen moeten verifieerbaar zijn, dat wil zeggen dat van bericht $H_{s,i}$ te zien is dat het samen met bericht $H_{\bar{s},i}$ een handtekening vormt. De lengte van deze aandelen is l bits.

De structuur van deze berichten is bijvoorbeeld als volgt. Laat c de hashwaarde van het contract c zijn, dan is $H_{s,i}$ Alice' handtekening over de boodschap (c, s, i). Daarnaast tekent Alice een verklaring waarin zij zich bindt aan contract C wanneer de tegenpartij voor enige i haar handtekening over $(c, 0, i)$ *en* $(c, 1, i)$ overlegt. Wanneer Bob de boodschap $H_{s,i}$ ontvangt, kan hij verifiëren dat deze van de beschreven vorm is, en hij weet dan dat Alice gebonden is aan C als hij de andere helft van dit paar verkrijgt.

Randomiserend protocol met keuzeoverdracht. Het protocol laat de tegenpartij één aandeel van elk paar kiezen met keuzeoverdracht, waarna de aandelen bit voor bit worden

onthuld. Alice verdeelt haar handtekening over berichten $H_{s,i}$ als boven beschreven, en Bob verdeelt zijn handtekening op dezelfde manier over berichten $G_{s,i}$.

1. *Uitwisseling van eerste helft.* Voor elk van de n paren $(H_{0,i}, H_{1,i})$ mag Bob met keuze-overdracht een deel kiezen. Voor elk van de n paren $(G_{0,i}, G_{1,i})$ mag Alice met keuze-overdracht een deel kiezen.

   Elke partij kan hier direct controleren dat er een echt aandeel is ontvangen, omdat de aandelen als zodanig herkenbaar zijn.

2. *Uitwisseling van de complete handtekening.* Er worden nu l deelronden gehouden. In deelronde j sturen Alice en Bob elkaar $2n$ bits, te weten de j^{e} bit van alle aandelen $H_{s,i}$, respectievelijk $G_{s,i}$, voor $s \in \{0, 1\}$ en $1 \leq i \leq n$.

   Als een partij wil proberen valse bits te sturen, loopt hij een groot risico dat de ander dit merkt omdat hij niet weet, welk aandeel de ander al volledig kent.

Als Alice en Bob het protocol tot het eind afmaken, beschikken ze allebei over n complete paren aandelen, dus in feite over n handtekeningen van de ander.

Stel nu dat een partij, zeg Alice, het protocol afbreekt na j deelronden van de tweede uitwisseling (dan wel na die ronde begint met het sturen van valse bits in plaats van de bits van H). Bob merkt dit en staakt het protocol ook, maar mogelijk heeft Alice al wel de bits van Bob uit deelronde $j+1$ ontvangen. Alice heeft dan van elk paar een bericht, en van het andere bericht van dat paar heeft ze $j+1$ bits; ze moet er dan nog $l-(j+1)$ raden, en dit betekent dat ze na 2^{l-j-1} rekenwerk over een handtekening van Bob beschikt. Bob heeft echter van elk paar al een bericht, en j bits van de andere helft, en heeft aan nog 2^{l-j} werk genoeg om Alice eveneens te binden. Alice behaalt dus door het afbreken geen groot voordeel, want zij wordt door Bob gebonden aan C met hoogstens dubbel zoveel werk als zij nog nodig heeft om Bob aan C te binden.

14.4.3 Lootjes trekken

Het laatste onderwerp van dit hoofdstuk is minder serieus van aard maar behoeft voor de Belgische lezers enige toelichting; zie kader 14.5. Sinterklaasavond nadert maar de n vrienden V_1 tot en met V_n hebben geen gelegenheid om op de gebruikelijke manier te loten voor wie iedereen een kado moet kopen.

Wat wordt bereikt met lootjes trekken? De bedoeling van lootjes trekken is de bepaling van een permutatie π, waarna V_i iets koopt voor $V_{\pi(i)}$, met de volgende randvoorwaarden.

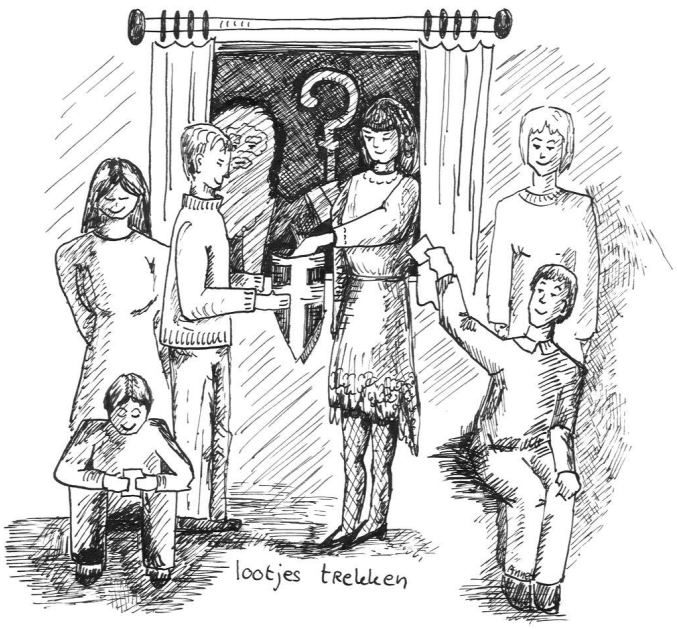
1. V_i trekt niet zichzelf, dus $\pi(i) \neq i$. Een i waarvoor $\pi(i) = i$ heet een *vast punt*.
2. V_i krijgt geen andere informatie over π dan de waarde $\pi(i)$.
3. V_i kan de keuze van π niet beïnvloeden.

Het werken met briefjes laat verschillende vormen van bedrog toe.

- *Geen permutatie.* Bij het maken van de briefjes kan iemand in plaats van zichzelf een ander op het briefje zetten, met als gevolg dat die persoon meerdere, en hijzelf geen kado's krijgt.

  De vrienden kunnen de briefjes controleren voor ze de hoed in gaan. Als ze dit niet doen, zal het bedrog pas op pakjesavond worden ontdekt.

Kader 14.5: Lootjes trekken voor Sinterklaasavond



Niet alleen wordt Sinterklaas in Nederland een dag eerder gevierd dan in België (namelijk op 5 december), er is ook een ander ritueel verbonden aan het geven van kado's. Als de vrienden Alice, Bob, Carla, Dick, Eva en Freek het feest samen vieren, kopen ze kado's voor elkaar, en het lot bepaalt voor wie elk iets koopt. Enkele weken voor het feest komen ze al bij elkaar voor het *lootjes trekken*. De namen worden op briefjes geschreven en elk trekt een briefje uit een hoed, en voor die persoon koopt hij een kado. Als iemand zichzelf trekt, wordt de hele loting over gedaan. De kans dat niemand zichzelf trekt is ongeveer $\frac{1}{e}$ (e is hier de basis van de natuurlijke logaritme, 2.7182...), en het verwachte aantal trekkingsronden is dus e.

- *Markering.* De briefjes kunnen bij het opstellen en dichtvouwen worden gemerkt op een manier die voor anderen onzichtbaar is. Hierdoor is beïnvloeding van de keuze en het illegaal verkrijgen van informatie mogelijk.
- *Vast punt.* Een persoon die zichzelf trekt kan het briefje houden in plaats van de trekking over te laten doen.
- *Valse vast-puntmelding.* Een persoon die iemand trekt voor wie hij liever geen kado koopt, kan beweren zichzelf te hebben getrokken en het lootje inleveren of de hele trekking over laten doen.

  Valse meldingen zijn eenvoudig te voorkomen door te eisen dat een melder het lootje met zijn eigen naam toont.

Omdat het hier gaat om vrienden die een 'heerlijk avondje' tegemoet hopen te gaan, valt het bedrog in de praktijk meestal mee. Valse vast-puntmeldingen komen wel eens voor, en het vergaren van informatie over π gebeurt doorgaans achteraf door 'social engineering'.

Lootjes trekken over internet. Als de vrienden niet voor lootjes trekken bij elkaar kunnen komen, moet de loting per brief, telefoon of internet worden verricht. Er voor deze situatie third parties die hun diensten gratis aanbieden (`www.sintlootjes.nl`), maar de vrienden lossen hun probleem liever zelf op. Crépeau en Kilian [CK93] beschrijven hoe een speler met kaarten een random permutatie zonder vast punt kan genereren zonder die zelf te weten te komen. Hun methode kan worden gebruikt als de vrienden van een van hen vertrouwen dat hij niet onder de kaarten zal kijken.

Het volgende protocol voldoet nog niet aan alle eisen maar kan een aanzet zijn om verder over het probleem na te denken. Een van de vrienden (V_n) wordt aangewezen om de lootjes aan te maken en 'dicht te vouwen' door ze te versleutelen met zijn eigen publieke RSA-sleutel. De anderen nemen elk een lootje uit 'de hoed' en laten het blind door V_n openen.

1. *Aanmaken van lootjes.* De deler, V_n, maakt n lootjes door de naam van V_j aan te vullen met een random getal: $x_j = \langle\, \mathbf{lootje}, r_j, V_j \,\rangle$.

2. *Dichtvouwen van lootjes.* De deler heeft een publieke RSA-sleutel (n, e) en berekent $y_j = x_j^e$.

3. *Vullen van de hoed.* De deler schudt de lootjes met een random permutatie σ en vormt de hoed als de reeks lootjes $H = (y_{\sigma(1)}, \ldots, y_{\sigma(n)})$. Iedereen mag deze complete reeks zien.

4. *Rondgaan van de hoed.* De deler stuurt de hoed naar V_1. Deelnemer V_i ontvangt van V_{i-1} een hoed met $n + 1 - i$ lootjes, kiest een random $r \leq n + 1 - i$, verwijdert het r^{e} getal uit de hoed en stuurt de hoed door naar V_{i+1}. Als laatste ontvangt V_n de hoed met slechts één lootje erin.

   Het lootje van V_i is een van de y die door V_n is aangemaakt; het nummer noemen we $\pi(i)$.

5. *Openen van lootjes.* Deelnemer V_i laat $y_{\pi(i)}$ blind door V_n openen.

6. *Vast-puntmelding.* Als V_i zichzelf heeft getrokken meldt hij dit en overlegt r_i als bewijs. De anderen gaan na dat $\langle\, \mathbf{lootje}, r_i, V_i \,\rangle^e$ inderdaad in de hoed voorkwam.

Als de trekking wordt overgedaan moeten er nieuwe lootjes worden gemaakt, dat wil zeggen, er worden nieuwe getallen r_i gekozen. De oude lootjes zijn namelijk, als er al enkele zijn ontsleuteld, voor de deelnemers van buitenaf herkenbaar.

Eigenschappen van het protocol. De versleuteling van lootjes door V_n is een prima variant op het dichtvouwen: de inhoud is voor de andere deelnemers onzichtbaar. Alleen V_n kan de 'dichtgevouwen' lootjes allemaal herkennen, maar hij mag dan ook niet kiezen maar moet het laatste lootje uit de hoed nemen.

- *Geen permutatie.* De deler kan de briefjes zo maken dat niet iedereen één keer voorkomt. Dit wordt niet tijdens het protocol bemerkt. Het is niet mogelijk, de deler alle getallen r_i te laten tonen voor de versleuteling, omdat dan de deelnemers zelf ook de individuele lootjes kunnen herkennen.

- *Markering.* Het aanbrengen van onzichtbare markeringen is onmogelijk. De deler kan wel met V_j samenwerken en hem r_j onthullen. Dan kan V_j, wanneer de hoed hem passeert, zien of zijn lootje er nog in zit of niet en het desgewenst nemen.

  Lootjes kunnen niet worden herkend aan hun positie in de reeks. De deler past de random permutatie σ toe. Evenmin kan de deler de trekking beïnvloeden door de lootjes in een bepaalde volgorde op te nemen in de hoed, omdat elke deelnemer er een random gekozen lootje uitpakt.

  Bij het rondgaan van de hoed is het belangrijk dat V_n *niet ziet* welke lootjes steeds worden doorgegeven. Immers, V_n zou aan de weggenomen lootjes precies kunnen zien wie wie heeft getrokken.

- *Vast punt.* Een deelnemer die zichzelf trekt kan het lootje ongemerkt houden.
- *Valse vast-puntmelding.* Van een vast-puntmelding door V_i moet een bewijs worden overlegd, namelijk het getal r_i; dit maakt een valse melding door anderen dan de deler onmogelijk. De deler kan r_n wel produceren zonder dat hij lootje y_n echt heeft gekregen, maar degene die V_n wel heeft getrokken kan dit bewijzen en een valse melding van V_n ontmaskeren; dit maakt een valse melding door de deler riskant.
- *Een ander lootje openen.* Deelnemer V_i kan, in plaats van het lootje dat hij getrokken heeft, een lootje laten openen dat hij heeft doorgegeven of juist één dat al ontbrak in de hoed. Hiermee verkrijgt hij informatie over π die hij niet hoort te hebben, maar hij heeft niet meer de mogelijkheid om zijn eigen lootje te lezen. Als bij een herhaalde trekking dezelfde RSA-sleutel wordt gebruikt, kan een deelnemer zelfs nog lootjes uit de vorige trekking laten openen.

De veiligheidseigenschappen van het protocol zijn goed vergelijkbaar met die van papieren lootjes.

Samenvatting en conclusies

Onder secure computing wordt verstaan dat een berekening wordt uitgevoerd, waarbij de invoerwaarden, tussenresultaten en eindresultaten voor een of meerdere deelnemers verborgen blijven. Het berekenen van een functie is het meest voorkomende geval: hierbij moet er een eindresultaat worden berekend, dat aan alle deelnemers bekend moet worden.

Soms kan voor een bepaalde probleemstelling een specifiek protocol worden ontworpen, wat we in dit hoofdstuk zagen voor het miljonairsprobleem of voor lootjes trekken. Dit vereist per probleemstelling een gedegen onderzoek, en de oplossingen zijn weinig flexibel met betrekking tot kleine wijzigingen in het probleem.

Er zijn verschillende algemene technieken waarmee functies kunnen worden berekend, namelijk technieken gebaseerd op homomorfe verdeelschema's of homomorfe groepscryptografie. De invoerwaarden worden door verdeling of versleuteling afgeschermd, waarna de berekening wordt uitgevoerd op de aandelen, respectievelijk de cijfertekst. Alleen het eindresultaat wordt gereconstrueerd, respectievelijk ontsleuteld. We hebben in dit hoofdstuk gezien hoe deze technieken voor het berekenen van klassen van functies kunnen worden gebruikt.

De algemene technieken zijn ook generiek uitbreidbaar naar een situatie waarin deelnemers willen liegen om het eindresultaat te beïnvloeden. Verdeelschema's kunnen hiervoor verifieerbaar worden gemaakt, waarbij wel bij elke berekeningsstap ook de publieke informatie moet kunnnen worden aangepast. Bij het gebruik van homomorfe encryptie is de mogelijkheid tot bedrog kleiner omdat de berekening zelf voor het grootste deel publiek te volgen is.

Een van de belangrijkste toepassingen van deze technieken is het houden van geheime verkiezingen. Voor verkiezingen is een oud protocol behandeld dat gebruik maakt van een speciaal cryptografisch primitief, namelijk anonieme kanalen. Moderne protocollen voor grootschalige verkiezingen, zoals CyberVote en VoteHere, maken gebruik van homomorfe encryptie.

Opgaven bij hoofdstuk 14

Opgave 14.1 *Alice en Bob beschikken over 3-bitsgetallen $a = a_2a_1a_0$ en $b = b_2b_1b_0$. De som van de getallen is $c = c_3c_2c_1c_0$. Geef logische functies die de bits c_i bepalen.*

Opgave 14.2 *Bewijs dat het XOR-schema uit sectie 14.1.3 perfect verhullend is.*

Opgave 14.3 *Generaliseer het bitverdeelschema uit sectie 14.1.3 en de bijbehorende deelprotocollen naar meer dan twee deelnemers.*

Opgave 14.4 *Een bericht x dat met RSA wordt versleuteld is onherkenbaar te maken door toevoeging van een random bitrij r (zoals bijvoorbeeld bij het lootjes trekken gebeurt). Kan hiermee een veilig protocol voor de villabewoners met RSA worden ontwikkeld?*

Opgave 14.5 *Kun je aan het protocol voor de villabewoners in sectie 14.2.3 verificatie toevoegen? Bij de bekendmaking van de invoer (u_i, v_i) moet een zero-knowledge proof van juistheid van de waarde worden gegeven.*

Opgave 14.6 Geheime stemming. *(Uit het tentamen van januari 1997.)*
Een commissie gebruikt dit protocol voor geheime stemming tussen een aantal alternatieven: ieder lid moet een stembiljet maken met een uniek nummer en voor elk alternatief een (ongeopende) blob. Men laat één stembiljet ondertekenen door een kiesbureau, waarna de blob van het alternatief naar keuze wordt geopend en het biljet (anoniem) wordt ingestuurd naar een stembureau.
(i) Beschrijf hoe elk lid één stembiljet kan laten ondertekenen, zonder dat bekend is wat het nummer daarvan is.
(ii) Wat zijn de consequenties als de hashfunctie die voor de blobs wordt gebruikt niet sterk, maar slechts zwak botsingsvrij is?
Wat zijn de consequenties als de hashfunctie niet one-way is?

Opgave 14.7 *Ga voor de blinde-overdrachtsprotocollen uit sectie 14.4.1 na, of de veiligheid voor zender en ontvanger perfect of computationeel gelden.*

Project 14.8 *Ontwerp een protocol voor lootjes trekken op basis van groepscryptografie. Welke bezwaren van het papieren lootjes trekken los je op?*

Hoofdstuk 15

De discrete logaritme

Het laatste hoofdstuk van dit boek behandelt enkele geavanceerde algoritmen voor een moeilijk rekenkundig probleem, namelijk de discrete logaritme. Van dit probleem en het factoriseren van integers is aangenomen dat er geen efficiënt algoritme voor bestaat (sectie 6.2.1). De algoritmen die hier worden beschreven kosten dan ook dermate veel tijd, dat een aanval ermee op een gebruikt cryptografisch algoritme zinloos moet worden geacht.

Toch kan voor het ontwerpen van een cryptografisch protocol en het kiezen van de parameters ervan, kennis van deze algoritmen van groot belang zijn. Immers, de aanname dat de logaritme of de factorisering niet kunnen worden berekend is altijd een simplificatie. Elk probleem kan voor kleine invoeren wel, en voor grote invoeren niet worden berekend. Voor de keuze van de parameters (lengte van de gebruikte getallen) is het van het grootste belang te weten, tot hoe ver een sterke tegenstander kan gaan. Men moet dus weten, tot hoe veel het best bekende algoritme in staat is, om vervolgens de parameters net iets groter te kiezen zodat toepassing ervan voor de tegenstander inderdaad zinloos is.

Van de twee belangrijke getaltheoretische problemen (discrete log en factorisering) geniet factorisering de meeste belangstelling. De reden is dat priemgetallen bij een groot publiek bekend zijn en het factoriseringsprobleem erg gemakkelijk is uit te leggen. Er zijn zelfs factoriseringswedstrijden in de vorm van de RSA Challenges (kader 15.1). De ontwikkeling van factorisatiealgoritmen loopt ongeveer gelijk met de ontwikkeling van algoritmen voor het logaritmeprobleem. Dit is te verklaren doordat een belangrijk concept, de *relatie*, in beide typen algoritmen een grote rol speelt, zoals zal worden uitgelegd in sectie 15.4.5.

In dit boek is ervoor gekozen, niet factorisatie maar de logaritme met enige diepgang te behandelen. De redenen hiervoor zijn de volgende. Ten eerste is het, vanwege de grote belangstelling voor factoriseren, gemakkelijker daarover andere bronnen te vinden.

Ten tweede is de logaritme voor meerdere cryptografische systemen van belang. Er bestaan namelijk varianten van het ElGamal-algoritme in diverse groepen, waarvan de elliptische krommen als eerste voorbeeld kunnen worden genoemd. Binnen een bespreking van het logaritmeprobleem past een behandeling van de gemeenschappelijke wiskundige achtergrond van al deze systemen, namelijk het begrip *groep*. Vervolgens zal blijken dat het ElGamal-algoritme en modernere ontwikkelingen als elliptische krommen en ECSTR in dit gemeenschappelijk kader overzichtelijk kunnen worden behandeld.

De behandeling van algoritmen voor de discrete logaritme geeft vervolgens inzicht in de claims die over deze nieuwe systemen worden gemaakt, namelijk dat ze veel efficiëntere implementaties toestaan bij een vergelijkbaar niveau van veiligheid. Kennis van algoritmen voor

Kader 15.1: De RSA Challenge

Binnen het bedrijf *RSA Security Inc.*, dat het RSA-algoritme commercieel exploiteert, bestaat een onderzoeksafdeling *RSA Laboratories*. Uiteraard is een van haar belangrijkste taken op de hoogte te blijven van hoe ver de techniek van factorisatie al is voortgeschreden. Een andere is het verzinnen van aanvallen op systemen die mogelijk een concurrent van RSA kunnen worden.

Cijfers	**Jaar**	**Werk**
Kwadratische zeef:		
100	1991	7
110	1992	75
120	1993	835
129	1994	5000
Getaltheoretische zeef:		
130	1996	1000
140	1999	2000
155	1999	8400

De RSA Challenge is een doorlopende wedstrijd, waarbij RSA Laboratories samengestelde getallen van oplopende lengte publiceert. Wie de priemfactoren kan insturen verdient een prijs. In juni 2001 stond er een beloning van 10.000 dollar op de factoren van RSA-576 (een getal van 576 bits) en 200.000 dollar op de factoren van RSA-2048 (een getal van 2048 bits).
De prijzen worden meestal gewonnen door een team rond de Nederlandse wiskundige Arjan Lenstra. De eenheid van werk in de tabel is MIPS-jaar, het werk dat in een jaar gedaan kan worden door een machine die een miljoen instructies per seconde verwerkt. De tabel laat zien dat de hoeveelheid werk die een aanvaller kan inzetten erg hard groeit: een factor duizend binnen een decennium! De grootte van de te factoriseren getallen groeit gestaag mee met een tempo van vijf à tien cijfers per jaar.
Geheel onvoorspelbaar is de vooruitgang die wordt gemaakt door de ontwikkeling van nieuwe algoritmen. De introductie van de getaltheoretische zeef in 1996 gaf een kleine stagnatie maar daarna is de groei weer sneller. De factorisatie van een 155-cijferig getal in 1999 werd als een mijlpaal beschouwd, want 155 cijfers komt overeen met 512 bits en 512-bits-RSA werd nog op grote schaal gebruikt! Rond 2010 kwam het factoriseren van een 768-bits modulus binnen bereik. Het NIST raadde zelfs aan, van 1024-bits moduli af te stappen en voor nieuwe keys 2048 bits te gebruiken.
WWW: `http://www.rsasecurity.com/rsalabs/`.

het berekenen van logaritmen is dus van direct belang voor het maken van afwegingen tussen diverse cryptografische algoritmen gebaseerd op een vergelijkbare wiskundige structuur.

Samenvattend kan men zeggen dat de studie van het logaritmeprobleem aanleiding geeft tot bredere en praktisch meer inzetbare kennis dan studie van factorisatie. Het hoofdstuk is als volgt ingedeeld. Een definitie van het wiskundige begrip *groep* en wat het logaritmeprobleem hiervoor inhoudt staan in sectie 15.1. Als voorbeelden van deze structuren worden in sectie 15.2 drie groepen behandeld, namelijk de multiplicatieve groep $\mathbb{Z}_p^*$ (gebruikt in het ElGamal-algoritme), elliptische krommen, en de XTR-groep. In sectie 15.3 komen groepsalgoritmen aan bod, dit wil zeggen, algoritmen die alleen de wiskundige operaties van een groep gebruiken. Deze algoritmen zijn in elke groep bruikbaar, maar in een groep met q elementen kost het berekenen van een logaritme ongeveer $\sqrt{q}$ stappen en dat is erg veel. Het algoritme in sectie 15.4 maakt specifiek gebruik van de representatie van de groep $\mathbb{Z}_p^*$ als gehele getallen. De complexiteit is op het eerste gezicht veel gunstiger, maar de complexiteit hangt niet af van het aantal getallen, maar van de lengte ervan. Dit verklaart waarom het ElGamal-algoritme moet werken met een relatief kleine subgroep van een modulaire groep met een veel grotere modulus.

15.1 Groepen en het logaritmeprobleem

De theorie van groepen is de tak van wiskunde die verzamelingen met een binaire operator bestudeert. De operator moet aan enkele eisen voldoen die de vijf *groepsaxioma's* worden genoemd. Uit deze vijf axioma's kunnen allerlei resultaten worden afgeleid die dan geldig zijn in *elke* verzameling waarin de axioma's geldig zijn. Men hoeft dus voor de toepassing van de gehele groepentheorie op een bepaalde gegeven structuur alleen maar deze vijf axioma's als stelling te bewijzen.

Wanneer de bestudeerde operatie een optelling is spreekt men van een *additieve* groep en als het een vermenigvuldiging is van een *multiplicatieve* groep. De lezer moet zich echter realiseren dat het verschil tussen deze twee operaties voor de structuur van de groep irrelevant is: het moet als een puur notationele kwestie worden beschouwd.

•	A	B	C	D
A	A	B	C	D
B	B	A	D	C
C	C	D	A	B
D	D	C	B	A

Alleen de eigenschappen van de operator zijn van belang en niet de structuur van de elementen van G of de berekeningswijze van de operator. Als voorbeeld kan daarom dienen de 'fruitverzameling' $F = \{A, B, C, D\}$ (appel, banaan, citroen, dadel) met de operator $\bullet$ als gedefinieerd in deze tabel.

15.1.1 De groepsstructuur

In deze sectie gebruiken we het symbool $\circ$ om de abstracte groepsoperatie weer te geven, en de letters G en H worden voor groepen gebruikt.

Definitie 15.1 *De structuur* $(G, \circ)$ *is een* commutatieve *of* Abelse groep *als aan de volgende vijf axioma's is voldaan:*

$\mathcal{G}1$ ***Gesloten:*** *Voor elke* $a, b \in G$ *is* $a \circ b \in G$.

$\mathcal{G}2$ ***Identiteit:*** *Er is een* $e \in G$ *waarvoor* $a \circ e = a$ *en* $e \circ a = a$, *voor elke* $a \in G$.

$\mathcal{G}3$ ***Inverse:*** *Voor elke* $a \in G$ *bestaat er een* $a' \in G$ *waarvoor* $a \circ a' = e$ *(de identiteit uit* $\mathcal{G}2$*).*

$\mathcal{G}4$ ***Associatief:*** *Voor elke* $a, b, c \in G$ *geldt* $a \circ (b \circ c) = (a \circ b) \circ c$.

$\mathcal{G}5$ ***Commutatief:*** *Voor elke* $a, b \in G$ *geldt* $a \circ b = b \circ a$.

De fruitstructuur $(F, \bullet)$ is een groep. Geslotenheid is hieraan te zien, dat op elke positie in de $\bullet$-tabel een element van F staat. Het bestaan van een identiteit is hieraan te zien, dat er een rij in de tabel is die gelijk is aan de rij boven de tabel (en een kolom die gelijk is aan de kolom links van de tabel). De betreffende rij is de rij van element A, dat dus de identiteit van de structuur is. Het bestaan van een inverse voor elk element is hieraan te zien, dat elke rij in de tabel het element A bevat. De commutativiteit aan de symmetrie van de tabel ten opzichte van de hoofddiagonaal.

Er bestaan ook interessante wiskundige structuren waarin wel aan de axioma's $\mathcal{G}1$ tot en met $\mathcal{G}4$ is voldaan, maar niet aan commutativiteit. Zulke structuren worden niet-commutatieve groepen genoemd (kader 15.3).

De fruitgroep heeft een eigenschap die voor alle groepen geldt (omdat hij te bewijzen is): er is maar één rij gelijk aan de toprij (de rij van A) en in elke rij vinden we slechts éénmaal de A.

Kader 15.2: Niels Hendrik Abel (1802-1829)

De aanduiding 'Abels' voor een commutatieve groep eert de Noorse wiskundige Niels Hendrik Abel, die als de grondlegger van de groepentheorie wordt beschouwd. Aan het begin van de negentiende eeuw was Europa in de ban van de vijfdegraadsvergelijkingen; zie Guadj [Gue99]. Voor de tweedegraadsvergelijking $ax^2 + bx + c = 0$ bestond al sinds de oudheid de bekende *abc*-formule: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ en soortgelijke formules waren ook voor derde- en vierdegraadsvergelijkingen ontdekt. Aanvankelijk (16 jaar oud!) meende Abel een oplossing voor dit probleem gevonden te hebben, maar zijn formule bevatte een fout. Hierdoor kwam hij op het goede spoor en in december 1823 bewees hij: algebraïsche vergelijkingen van de vijfde graad zijn niet oplosbaar.
Abel stuurde zijn bewijs naar alle grote wiskundigen van zijn tijd, maar niemand nam de moeite het te lezen. (Zijn handschrift was slecht leesbaar.) Hetzelfde lot trof zijn latere artikel over integratie en een over transcendente getallen. Uiteindelijk werd zijn werk toch erkend en gewaardeerd en er werd hem zelfs een professoraat aangeboden vanuit Berlijn. Helaas kwam de aanstellingsbrief te laat: enkele dagen voordat de brief in Noorwegen aankwam was Abel, 26 jaar oud, overleden aan tuberculose.
Behalve met de aanduiding voor commutatieve groepen wordt hij geëerd met dit standbeeld in Oslo (Gustav Vigeland, 1908).

Kader 15.3: De symmetrieën van een veelhoek

Er zijn precies $2n$ lineaire transformaties van het vlak (combinaties van rotatie, spiegeling en translatie) die een regelmatige n-hoek op zichzelf afbeelden. De (denkbeeldige) pijl kan namelijk op n posities tegen de klok in en op n posities met de klok mee wijzen. De verzameling van deze transformaties heet A_n.
De samenstelling van twee transformaties uit A_n beeldt ook de n-hoek op zichzelf af en behoort dus tot A_n ($\mathcal{G}1$). De identiteit (afbeelding die elk punt van het vlak op zichzelf afbeeldt) beeldt de veelhoek op zichzelf af en behoort dus tot A_n; bovendien is de samenstelling van deze identiteit met afbeelding a altijd gelijk aan a ($\mathcal{G}2$). Elke transformatie van het platte vlak heeft een inverse, en als een afbeelding de veelhoek op zichzelf afbeeldt dan doet de inverse dat ook ($\mathcal{G}3$). Het samenstellen van afbeeldingen is altijd associatief ($\mathcal{G}4$) maar is vaak niet commutatief en in dit geval ook niet.
Daarmee is A_n een voorbeeld van een niet-commutatieve (of: niet-Abelse) groep.

Stelling 15.2 *In een groep $(G, \circ)$ zijn de identiteit en de inverse van elk element uniek.*

Bewijs. Veronderstel eerst dat e_1 en e_2 identiteiten zijn. Dan geldt voor alle a dat $a \circ e_1 = a$, dus geldt $e_2 \circ e_1 = e_2$. Omdat e_2 een identiteit is geldt voor alle a dat $e_2 \circ a = a$, dus geldt $e_2 \circ e_1 = e_1$. Hieruit volgt $e_1 = e_2$, er is dus maar één identiteit.

Veronderstel nu dat a de inversen a_1 en a_2 heeft. Dan geldt

$$\begin{array}{rcll} a_1 & = & e \circ a_1 & \mathcal{G}2 \\ & = & (a \circ a_2) \circ a_1 & a_2 \text{ is inverse van } a \\ & = & (a_2 \circ a) \circ a_1 & \mathcal{G}5 \\ & = & a_2 \circ (a \circ a_1) & \mathcal{G}4 \\ & = & a_2 \circ e & a_1 \text{ is inverse van } a \\ & = & a_2 & \mathcal{G}2 \end{array}$$

Er is dus maar één inverse van a. □

Deze stelling rechtvaardigt het gebruik van de notatie e voor *de* identiteit van de groep en a^{-1} voor *de* inverse van a. Het aantal elementen in een groep kan eindig of oneindig zijn, maar voor de cryptografie zijn alleen eindige groepen relevant. Daarom zullen we van nu af alleen eindige groepen bekijken en we schrijven $\#G$ voor het aantal elementen van G (of de *orde* van G).

Een deelverzameling H van G heet een *subgroep* als hij zelf een groep is (met dezelfde operator). De verzameling H moet dan aan dezelfde vijf axioma's voldoen, en hierbij volgen associativiteit en commutativiteit direct uit deze eigenschappen voor G. De andere drie eigenschappen volgen uit een enkele conditie, namelijk dat voor $a, b \in H$ ook $a \circ b^{-1} \in H$.

Stelling 15.3 *Zij H een niet-lege deelverzameling van een groep $(G, \circ)$ met de eigenschap dat voor $a, b \in H$ ook $a \circ b^{-1} \in H$. Dan is $(H, \circ)$ een groep.*

Bewijs. Omdat $a \circ (b \circ c) = (a \circ b) \circ c$ en $a \circ b = b \circ a$ gelden voor alle elementen van G, gelden ze zeker voor de elementen van H, waarmee $\mathcal{G}4$ en $\mathcal{G}5$ bewezen zijn. Verder bevat H ten minste één element a, en de gegeven eigenschap impliceert dat dan ook $a \circ a^{-1} = e$ in H zit, waarmee $\mathcal{G}2$ bewezen is. Maar dan zit voor elke $a \in H$ ook $e \circ a^{-1} = a^{-1}$ in H, hetgeen $\mathcal{G}3$ bewijst. Ten slotte volgt hieruit dat als $a, b \in H$, dan ook b^{-1} en dus $a \circ (b^{-1})^{-1} = a \circ b$ (Opgave 15.3), waarmee $\mathcal{G}1$ geldt. □

Veel resultaten uit de groepentheorie gaan over wat er gebeurt als een element meerdere malen met zichzelf wordt gecombineerd, als in $a \circ a \cdots a$.

Definitie 15.4 *Voor $a \in G$ is a^k gedefinieerd als*

$$a^k = \begin{cases} e & \text{als } k = 0 \\ a \circ a^{k-1} & \text{als } k > 0 \\ (a^{-1})^k & \text{als } k < 0 \end{cases} .$$

Let op dat in deze definitie k een geheel getal is en a een groepselement.

Voor elk element a zijn er positieve getallen k waarvoor $a^k = e$; het kleinste van deze getallen heet de orde van a of $\mathrm{ord}(a)$. Deze orde is een deler van $\#G$ hetgeen impliceert dat $a^{\#G} = e$ voor alle a. De bewijzen van deze stelling zijn volledig gelijk aan die van stelling 5.10, die er een speciaal geval van is (voor G de multiplicatieve groep $\mathbb{Z}_p^*$).

Stelling 15.5 *Als $g \in G$ een element van orde q is, dan is*

$$H = \{g^i : i \in \mathbb{Z}\}$$

een subgroep van G met q elementen.

```
function macht (a, k):
    if k = 0 then return e ;
    if k < 0 then return macht (a^{-1}, -k) ;
    if odd(k) then return macht (a, k - 1) ∘ a ;
    (* k is even *)
    b := macht (a, k/2) ;
    return b ∘ b
```

Algoritme 15.4: Groepsalgoritme voor a^k.

Bewijs. Als $a = g^i$ en $b = g^j$ dan is $a \circ b^{-1} = g^{i-j}$, dus bevat in H en daarom is H een subgroep.

Omdat $g^l \neq e$ voor alle $0 \leq l < q$ zijn alle getallen g^i voor i tussen 0 en $q-1$ verschillend; immers zou $g^i = g^j$ dan is $g^{j-i} = e$. Omdat $g^q = e$ is elk element g^i gelijk aan een van die q getallen, namelijk $g^{i \bmod q}$. Dit bewijst dat H precies q elementen heeft. □

De subgroep H gegenereerd door g wordt ook wel als $\langle g \rangle$ genoteerd; g heet een *voortbrenger* of *generator* van de subgroep. Als $\mathrm{ord}(g) = \#G$ dan is H gelijk aan G en g een voortbrenger van de hele groep G; deze situatie is in cryptografische toepassingen niet van bijzonder belang.

15.1.2 Rekenen in groepen

Om groepen in cryptografische algoritmen te gebruiken moet men ermee kunnen rekenen. Allereerst is hiervoor nodig dat informatie (berichten of sleutels) als een groepselement kan worden gecodeerd. Verder moeten voor het manipuleren van groepselementen de identiteit e, de groepsoperator $\circ$ en het vinden van inversen $^{-1}$ zijn gedefinieerd. Soms is het nodig een random element in G te genereren (bijvoorbeeld algoritme 15.5). Soms wordt de grootte van de groep $\#G$ bekend verondersteld en soms ook de factorisatie van $\#G$.

Een algoritme dat de groepselementen alleen met deze genoemde functies manipuleert is natuurlijk in elke groep bruikbaar en we noemen zo'n algoritme een *groepsalgoritme*. Het is ook mogelijk dat de elementen van de groep verder kunnen worden bewerkt met andere functies omdat over de elementen meer bekend is dan de groepsstructuur. Een algoritme dat deze functies gebruikt, kan dan in deze speciale groep werken, maar is niet in een andere groep toepasbaar waar de aanvullende structuur ontbreekt. Zo'n algoritme heet een *speciaal algoritme*.

We zullen nu twee veel gebruikte groepsalgoritmen zien, namelijk voor (gegeneraliseerd) machtsverheffen (algoritme 15.4) en voor het vinden van een element met gewenste orde (algoritme 15.5). Algoritme 15.4 werkt geheel analoog aan algoritme 6.3 en op de werking gaan we niet in. De conclusie is dat er een groepsalgoritme bestaat voor gegeneraliseerd machtsverheffen dat polynomiaal (in de lengte van k) veel groepsoperaties kost.

Algoritme 15.5 vindt een element van orde q, waar q een priemgetal is en een deler van $\#G$ maar niet van $\frac{\#G}{q}$ ($\#G$ bevat dus precies één factor q). Het berekent r als $\#G/q$ en gebruikt dat de r^{e} macht van een element, indien ongelijk aan e, zelf orde q heeft. Inderdaad is $g^q = (a^r)^q = a^{\#G} = e$, dus de orde van g deelt q. Als g ongelijk is aan e is de orde groter dan 1, en de enige deler die q (priemgetal) verder heeft is q. Voor elk priemgetal q dat $\#G$ deelt bestaan er elementen van orde q en wel ten minste $q-1$ stuks, maar op het bewijs hiervan en het precieze aantal gaan we niet in. Het is wel belangrijk op te merken dat de resultaten hierover uit sectie 5.4.1 niet zonder meer geldig zijn.

$r := \#G/q$;
repeat $a :=$ *random* in G ;
$g := a^r$
until $g \neq e$;
return g

Algoritme 15.5: Een element van orde q (priem).

15.1.3 Logaritmen en cryptografie in een groep

Zij nu g een element van orde q (priem) en H de door g voortgebrachte subgroep. De generator van een groep is doorgaans niet uniek; de lezer kan zich hiervan overtuigen door in de groep van opgave 15.2 de subgroepen $\langle C \rangle$ en $\langle D \rangle$ te vergelijken. We zullen daarom steeds aannemen dat met de keuze van een subgroep ook een vaste voortbrenger g is gekozen. Met de voortbrenger en subgroep zijn een aantal computationele problemen verbonden.

1. **Inclusie:** Gegeven $y \in G$, is $y \in H$?
2. **Discrete logaritme:** Gegeven $y \in H$, vind x waarvoor $g^x = y$.
3. **Diffie-Hellman-probleem:** Elementen a, b en c voldoen aan de Diffie-Hellman-relatie, notatie *DifHel*(a, b, c) als er een x bestaat waarvoor $a = g^x$ en $c = b^x$. Het herkenningsprobleem vraagt, gegeven a, b, c, geldt *DifHel*(a, b, c)? Het completeringsprobleem vraagt, gegeven a en b, voor welke c geldt *DifHel*(a, b, c)?

Met name de discrete logaritme en het Diffie-Hellman-probleem zijn voor de cryptografie interessant omdat ze de inverse betreffen van functies die efficiënt uit te rekenen zijn. Het Diffie-Hellman-sleutelprotocol en het ElGamal-algoritme kunnen direct naar groepen worden gegeneraliseerd.

In het (gegeneraliseerde) Diffie-Hellman-protocol heeft een gebruiker een geheime sleutel $x \in \mathbb{Z}_q$ en publieke sleutel $y = g^x \in H$. Als Alice wil communiceren met Bob gebruikt zij als sleutel $k = y_B^{x_A}$, berekend in H, en Bob gebruikt $y_A^{x_B}$; deze waarden zijn gelijk. Omdat *DifHel*(y_A, y_B, k) geldt, is het vinden van k voor Oscar hetzelfde als Diffie-Hellman-completering.

Het ElGamal-algoritme uit sectie 6.3.2 laat zich zo generaliseren:

Het algemene ElGamal-schema	
Berekeningen in: G, met g een element van orde q	Procedure voor sleutelgeneratie: 1. Kies random $a \in \mathbb{Z}_q$. 2. Bereken $b = g^a$ in G. Publieke sleutel: b. Geheime sleutel: a.
Encryptie van x: 1. Kies random $k \in \mathbb{Z}_q$. 2. Bereken $u = g^k$ en $z = b^k$. 3. Bereken $v = z \circ x$. 4. $y = (u, v)$	Decryptie van y: 1. Bereken $z' = u^a$. 2. Bereken $x' = v \circ z'^{-1}$

Het is eenvoudig na te rekenen dat de gevonden z' gelijk is aan z en dus x' gelijk aan x. Een aanvaller onderschept u en kent b; deze staan met z in de Diffie-Hellman-relatie en het vinden van z (en x) is weer het completeringsprobleem.

In feite is deze generalisatie mogelijk voor alle cryptografische technieken die op exponentiëren zijn gebaseerd. We noemen de ElGamal-handtekening (sectie 7.2.3), Chaums hashfunctie (sectie 7.3.2), Schnorrs identificatieprotocol en publieke aandelen (sectie 9.2.2). Met de invoering van de groep als wiskundige abstractie kunnen meerdere cryptografische technieken, waaronder ElGamal, elliptische krommen en het recente XTR, onder een gemeenschappelijke noemer worden geplaatst. Al deze algoritmen gebruiken dezelfde groepsberekeningen, maar verschillen alleen in het gebruik van de onderliggende groep.

Voor al deze cryptografische technieken kan een keuze worden gemaakt tussen een *gemeenschappelijke* of een *individuele* groep. In het eerste model worden G en g door een centrale instantie gekozen en door alle gebruikers gedeeld, en alleen x en y verschillen per gebruiker. In het tweede model kiest elke gebruiker zelf een G en g (en x en y). De beschrijving van G en g is dan onderdeel van de publieke sleutel, die daardoor uiteraard langer wordt.

Al deze gegeneraliseerde algoritmen kunnen efficiënt worden aangevallen als een algoritme voor de discrete logaritme in de betreffende groep beschikbaar is. De groep G en de orde van g moeten dus dermate groot worden gekozen dat het berekenen van logaritmen in H met de best bekende technieken als onhaalbaar moet worden beschouwd. In sectie 15.3.4 wordt bewezen dat een groepsalgoritme voor logaritmen ten minste $\sqrt{q}$ stappen gebruikt (als q priem is). Wanneer men aanneemt dat de aanvaller alleen over een groepsalgoritme beschikt en dat een berekening van 2^{80} stappen ondoenlijk is, biedt een generator van orde $q \approx 2^{160}$ voldoende veiligheid. Om het systeem aan te vallen moet Oscar met iets beters komen dan een groepsalgoritme. Hij moet zich in de structuur van de elementen van G verdiepen en proberen hieruit een speciaal algoritme voor logaritmen in G te ontwikkelen.

De kosten voor toepassing van deze algoritmen hangen natuurlijk af van de gekozen groep. De lengte van de geheime sleutel x is $\log(q)$ bits. De lengte van de publieke sleutel y is $\log(\#G)$ bits (aannemende dat elementen van G efficiënt zijn gecodeerd). Een exponentiatie (met algoritme 15.4) kost maximaal $2\log(q)$ berekeningen van $\circ$ met een gemiddelde van $1\frac{1}{2}\log(q)$.

15.1.4 Isomorfie en representatie

$\oplus$	00	11	01	10
00	00	11	01	10
11	11	00	10	01
01	01	10	00	11
10	10	01	11	00

Zoals eerder opgemerkt, is voor de bestudering van een groep van belang welke eigenschappen de groepsoperator heeft, en niet wat de aard van de elementen of de berekeningswijze van de operator is. Daarom is het mogelijk dat twee groepen er op het oog geheel verschillend uitzien, maar toch dezelfde structuur hebben. Als voorbeeld mag dienen de groep die bestaat uit bitrijen van lengte 2 en als operatie de bitsgewijze XOR; deze groep heet $(\mathbb{Z}_2)^2$. Door het herbenoemen van de vier elementen volgens 00 $\rightarrow$ A, 11 $\rightarrow$ B, 01 $\rightarrow$ C en 10 $\rightarrow$ D wordt de tabel van de fruitgroep F verkregen. Dit betekent dat, hoewel $(\mathbb{Z}_2)^2$ en F verschillende verzamelingen zijn, hun algebraïsche structuur gelijk is. Uitspraken die gelden voor de structuur van F gelden dus ook voor $(\mathbb{Z}_2)^2$, bijvoorbeeld:

1. Elk element ongelijk de identiteit heeft orde 2.
2. De groep heeft precies drie subgroepen van grootte 2.
3. De groep is commutatief.

4. De groep bevat geen element van orde 4.

Uitspraken over F die niet over de algebraïsche structuur gaan gelden niet voor $(\mathbb{Z}_2)^2$, bijvoorbeeld: 'de elementen van F bederven wanneer men ze te lang in een warme kamer laat liggen'. Groepen met dezelfde structuur worden in de groepentheorie feitelijk als dezelfde abstracte groep beschouwd en de relatie ertussen heet *isomorfie*.

Definitie 15.6 *De groepen* $(G, \bullet)$ *en* $(H, \star)$ *heten* isomorf*, notatie* $G \cong H$*, als er een bijectieve afbeelding* $f : G \to H$ *bestaat zodanig dat* $f(a \bullet b) = f(a) \star f(b)$.

De afbeelding f is hierin de herbenoeming van de elementen.

Een groepsalgoritme dat werkt voor G werkt ook in H en gebruikt evenveel operaties (de berekening van $\circ$ hoeft niet even duur te zijn als de berekening van $\star$). Eigenschappen die gelden voor de structuur van G gelden ook voor de structuur van H; hiervan werd reeds in stelling 5.19 en 5.20 gebruikgemaakt. De groepen $(\mathbb{Z}_p^*, \cdot)$ en $(\mathbb{Z}_{p-1}, +)$ zijn isomorf. In de tweede groep herkent men een 'even getal' x (dit is een x waarvoor er een y bestaat met $y + y = x$) doordat $(\frac{p-1}{2}) \cdot x = 0$. Voor de eerste groep betekent dit dat een kwadraat x (dit is een x waarvoor er een y bestaat met $y \cdot y = x$) wordt herkend doordat $x^{(\frac{p-1}{2})} = 1$. In $(\mathbb{Z}_{p-1}, +)$ is een even getal ook te herkennen door rechtstreeks een deling door 2 uit te voeren. Dit is echter geen groepsoperatie, het moet dus als een speciaal algoritme worden beschouwd en het is in $(\mathbb{Z}_p^*, \cdot)$ ook niet toepasbaar om kwadraten te herkennen.

Een zeer bekende groep, waarvan de eigenschappen redelijk gemakkelijk te begrijpen zijn, is $(\mathbb{Z}_q, +)$, de verzameling resten modulo q (zowel priem als samengesteld) met als operatie de optelling. Deze groep heeft een generator, namelijk het getal 1. Omdat dit een additieve groep is wordt de gegeneraliseerde macht a^k, zoals gedefinieerd voor groepen in definitie 15.4, geschreven als $k \cdot a$.

Een voor de hand liggende vraag is nu: als voor de cryptografie verschillende groepen worden gebruikt, waarin verschillen die groepen dan? Het antwoord is: de diverse groepen verschillen *niets* in hun groepsstructuur, want een subgroep gegenereerd door een element g is altijd isomorf aan de cyclische groep van grootte $\text{ord}(g)$.

Stelling 15.7 *Als* g *een element van orde* q *is, dan geldt voor de additieve groep* $\mathbb{Z}_q$ *dat* $\mathbb{Z}_q \cong \langle g \rangle$.

Bewijs. Beschouw de functie $f : \mathbb{Z}_q \to \langle g \rangle$ gedefinieerd door $f(i) = g^i$. (Deze definitie is toegestaan omdat voor gehele getallen geldt $i_1 \equiv_q i_2 \Rightarrow g^{i_1} = g^{i_2}$.) De functie f is een bijectie, en er geldt $f(i + j) = g^{i+j} = g^i \circ g^j = f(i) \circ f(j)$. □

Kortom, de groepen die in de diverse ElGamal-varianten worden gebruikt zijn allemaal isomorf aan $\mathbb{Z}_q$, en hebben alle structurele eigenschappen van $\mathbb{Z}_q$. Het verschil is dat de elementen anders 'heten', of, technischer gezegd, anders zijn gerepresenteeerd als bitrij. De groepen verschillen in *representatie*, maar niet in structuur.

De volgende voor de hand liggende vraag is dan: welk doel wordt nagestreefd met het op een ingewikkelde manier representeren van de elementen van steeds dezelfde groep? Het antwoord is: er wordt bewust gezocht naar een representatie die de structuur van $\mathbb{Z}_q$ zó ontoegankelijk maakt, dat de logaritme van een element moeilijk te vinden is. De representatie moet uiteraard wel het uitvoeren van de groepsoperatie en de inverse efficiënt toestaan. Als verdere wens geldt dat de elementen van G zo klein mogelijk te representeren zijn en dat de groepsoperatie op G weinig tijd kost.

De cryptografische functies (kiezen van g, encryptie en decryptie) kunnen allemaal met efficiënte groepsalgoritmen worden uitgevoerd. (Dit geldt overigens alleen voor het rekenen in

de groep G en niet voor het kiezen van een specifieke groep van een bepaalde soort; hierop komen we nog terug.) Dus zodra een groep is gekozen en de groepsoperator is geïmplementeerd kan men aan de slag. Echter, zoals al is opgemerkt bestaat er voor de discrete logaritme geen groepsalgoritme dat minder dan $\sqrt{q}$ tijd kost. Een aanvaller moet dus voor elke nieuw gedefinieerde groep proberen een speciaal algoritme te verzinnen.

15.2 Groepen voor cryptografie

Deze sectie behandelt kort een aantal groepsstructuren (volgens stelling 15.7 in feite een aantal verschillende representaties van een cyclische groep) die voor cryptografie worden gebruikt. Het zijn, behalve de al bekende multiplicatieve groep $\mathbb{Z}_p^*$ (sectie 15.2.1), de elliptische krommen (sectie 15.2.2) en XTR (sectie 15.2.3).

We zullen eerst zien waarom het nodig is de groepsstructuur van de additieve cyclische groep te maskeren. Zij G de additieve groep $(\mathbb{Z}_q, +)$ met voortbrenger 1. De logaritme van x is het aantal malen dat 1 bij elkaar moet worden opgeteld om x te krijgen als $x = 1 + \ldots + 1$. Uiteraard is dit aantal gelijk aan x! Er geldt dus $\log x = x$ en de logaritme is dus niet moeilijk te berekenen; omdat deze gelijkheid alleen in $\mathbb{Z}_q$ met generator 1 geldt, is dit een speciaal algoritme.

Ook wanneer een generator g ongelijk 1 wordt genomen is de logaritme te gemakkelijk te berekenen: gezocht wordt dan naar het aantal malen dat g moet worden opgeteld om x te krijgen en dit aantal x/g is door een deling in $\mathbb{Z}_q^*$ te berekenen. Weer is sprake van een speciaal algoritme, want deling is geen operatie van de groep $(Z_q, +)$.

15.2.1 De multiplicatieve groep

Het standaard ElGamal-algoritme rekent met vermenigvuldigingen modulo p, dus in $\mathbb{Z}_p^*$ voor een groot priemgetal p. (Voor samengestelde p is $\mathbb{Z}_p^*$ met de Chinese reststelling in subgroepen te splitsen en is de totale sterkte niet groter dan wanneer met een priemdeler van p wordt gerekend.)

De orde van $\mathbb{Z}_p^*$ is $p - 1$, en om een element van orde q te bevatten moet q een deler zijn van $p-1$. Het is mogelijk, om voor g een generator van $\mathbb{Z}_p^*$ te kiezen; dan is $q = p-1$, dus niet priem. In subgroep-ElGamal wordt voor de keuze van p gezocht naar een groot priemgetal q (zeg 160 bits) en dan naar een veelvoud van q plus 1 ($p = r \cdot q + 1$) dat eveneens priem is. Als generator wordt a^r gekozen voor random a, mits dit ongelijk 1 is (zie algoritme 15.5). Het vinden van een geschikte groep G en een geschikte subgroep H zijn dus erg gemakkelijk en snel; dit geldt ook voor berekeningen in $\mathbb{Z}_p^*$.

Het enige nadeel van de multiplicatieve groep is dat er vrij efficiënte speciale algoritmen bestaan om met getallen te rekenen. De complexiteit van het vinden van logaritmen in $\mathbb{Z}_p^*$ is vergelijkbaar met het factoriseren van een samengesteld getal met net zoveel bits als p. Waar voor RSA moduli van 512 bits anno 2001 niet meer veilig zijn en er met moduli van 768 of 1024 bits wordt gerekend, moet ook voor logaritmegebaseerde cryptografie een p van 768 of 1024 bits worden gebruikt. Er wordt dus gerekend in een subgroep van q elementen, maar er moeten steeds 1024 bits worden opgeslagen en de berekeningen zijn op 1024-bitsgetallen.

15.2.2 Elliptische krommen

Elliptische krommen hebben niets te maken met ellipsen en de elliptische krommen die in de cryptografie worden gebruikt doen ook weinig denken aan een wiskundige kromme. Elliptische krommen worden al bijna honderd jaar gebruikt als een hulpmiddel om de eigenschappen van getallen beter te begrijpen. Zo is er bijvoorbeeld een algoritme voor factorisering van een samengesteld getal n dat gebruikmaakt van een elliptische kromme over $\mathbb{Z}_n$. In 1985 suggereerde Miller [Mil86] dat de structuur van de elliptische kromme over $\mathbb{Z}_p$ een generalisatie van het Diffie-Hellman-protocol mogelijk zou maken.

Punten van de elliptische kromme. Zoals eerder opgemerkt, is het voor een groep niet van belang wat de aard van de elementen is: dit kunnen getallen zijn, maar ook objecten van heel andere aard. De elementen van een elliptische kromme zijn punten (x, y) in het platte vlak. De coëfficiënten x en y kunnen worden genomen uit $\mathbb{Q}$, $\mathbb{R}$, of $\mathbb{C}$ (rationele, reële of complexe getallen). In de cryptografie echter worden alleen eindige elliptische krommen gebruikt, waarbij in eindige getallenverzamelingen wordt gerekend. Niet alle punten zijn element van de kromme: de punten voldoen aan een definiërende derdegraadsvergelijking. Een specifieke kromme $E_{p,a,b}$ ligt vast door de keuze van p, a en b.

Definitie 15.8 *De elliptische kromme $E_{p,a,b}$ bevat de verzameling paren $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ die voldoen aan $y^2 = x^3 + a \cdot x + b$, en een speciaal punt ∞.*

Omdat y alleen als y^2 in de formule voorkomt, is de kromme symmetrisch: als $P = (x, y) \in E_{p,a,b}$, dan ook $P' = (x, -y)$. Het punt P' noemen we $-P$ of het *tegenovergestelde* van P.

Het aantal punten op de kromme bedraagt ongeveer p, hetgeen men als volgt kan inzien. Er zijn p mogelijkheden voor het kiezen van een coëfficiënt x, waarna $z = x^3 + a \cdot x + b$ wordt berekend. Als z een kwadraat is en ongelijk 0, dan zijn er twee y met $y^2 = z$ en we vinden twee punten op de kromme. Als $z = 0$ is er één punt $(x, 0)$ op $E_{p,a,b}$. Als z geen kwadraat is in $\mathbb{Z}_p$, is er geen punt (x, y) op de kromme. Omdat de helft van de getallen in $\mathbb{Z}_p^*$ een kwadraat is (stelling 5.19) verwachten we bij ongeveer de helft van de x twee punten en bij ongeveer de helft van de x geen punten.

Over het precieze aantal punten is dit bekend (Stelling van Hasse):

$$p - 2\sqrt{p} + 1 \leq \#E_{p,\,a,\,b} \leq p + 2\sqrt{p} + 1.$$

De naïeve representatie van het punt (x, y) (als twee getallen uit $\mathbb{Z}_p$) gebruikt $2 \log p$ bits. Omdat wortels modulo p kunnen worden berekend (sectie 6.1.4) is een compactere representatie in $\log p + 1$ bits mogelijk. Samen met de x wordt alleen opgeslagen of y (als getal tussen 0 en $p - 1$) even of oneven is. Van de getallen y en $-y$ is er namelijk precies één even zodat na berekening van z (als boven) de juiste y kan worden gevonden.

De groepsoperatie. De groepsoperatie neemt twee punten en beeldt deze af op een derde; de gebruikte operatie heeft een sterk meetkundige interpretatie. Omdat de definiërende vergelijking van graad drie is, kan een 'rechte lijn' (een puntverzameling van de vorm $y = c \cdot x + d$) drie punten met de kromme gemeen hebben. De twee gegeven punten definiëren de lijn door die twee punten; het derde snijpunt met de kromme wordt gezocht, en het resultaat van de operatie is het tegenovergestelde van dat punt.

Rekenkundig kan het resultaat van de operatie als volgt worden bepaald. Zij $y = c \cdot x + d$ een lijn; de snijpunten met $E_{p,\,a,\,b}$ worden bepaald door $(c \cdot x + d)^2 = x^3 + a \cdot x + b$ ofwel

$$x^3 - c^2 \cdot x^2 + (a - 2c \cdot d) \cdot x + (b - d^2) = 0.$$

Wanneer twee oplossingen x_1 en x_2 bekend zijn, is de derde oplossing bepaald door $x_1 + x_2 + x_3 = c^2$, ofwel $x_3 = c^2 - x_1 - x_2$. Het derde punt is dus eenvoudig uit te drukken in de 'richtingscoëfficiënt' c van de lijn en dan geldt $y_3 = c \cdot x_3 + d = c \cdot (x_3 - x_1) - y_1$.

Laat nu punten $P_1 = (x_1, y_1)$ en $P_2 = (x_2, y_2)$ op $E_{p,a,b}$ gegeven zijn (met $x_2 \neq x_1$). De lijn door P_1 en P_2 heeft richtingscoëfficiënt $c = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)$, waaruit x_3 en y_3 kunnen worden berekend. Het resultaat van de groepsoperatie is dan $(x_3, -y_3)$ en wordt over het algemeen geschreven als $P_1 + P_2$: voor de operatie op de elliptische kromme is namelijk de additieve schrijfwijze ingeburgerd.

Deze berekening is niet algemeen toepasbaar en er gelden dan ook enkele uitzonderingen op, waartoe onder andere het speciale punt ∞ dient. Door $P = (x_1, y_1)$ en $-P = (x_1, -y_1)$ gaat de verticale lijn bepaald door $x = x_1$, die $E_{p,a,b}$ maar in (ten hoogste) twee punten snijdt. Het punt ∞ (met $-\infty = \infty$) is opgenomen als extra (derde) snijpunt van E met elke verticale lijn. Hierdoor geldt $P + (-P) = \infty$, en ook geldt $P + \infty = P$.

Een tweede uitzondering vormt de berekening van $P + P$, immers dan geldt $x_1 = x_2$ en er mag niet door $x_1 - x_2$ worden gedeeld. In continue elliptische krommen (over $\mathbb{R}$ en $\mathbb{C}$) wordt dan de raaklijn aan de kromme genomen, dit is de lijn $y - y_1 = \left(\frac{3x_1^2 + a}{2y_1}\right) \cdot x - x_1$. Hoewel op een discrete curve het begrip 'raaklijn' niet betekenisvol is, kan deze formule worden gebruikt. De berekening van $P + P$ verschilt dus alleen in de berekening van c als $\left(\frac{3x_1^2 + a}{2y_1}\right)$ en verloopt verder analoog aan bovenstaande berekening.

Samenvattend zijn de identiteit, inverse en groepsoperatie als volgt:

1. De identiteit is ∞.

2. De inverse van ∞ is $-\infty = \infty$.
   De inverse van $P = (x_1, y_1)$ is $-P = (x_1, -y_1)$.

3. Voor alle P is $\infty + P = P + \infty = P$.

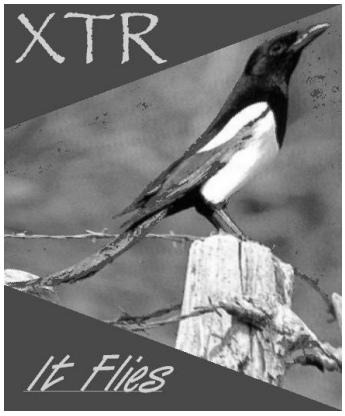
4. Voor $P_1 = (x_1, y_1)$ en $P_2 = (x_2, y_2)$ met $x_1 \neq x_2$ wordt $P_1 + P_2$ gevonden als $(x_3, -y_3)$ met
$$\begin{aligned} c &= \left(\frac{y_1 - y_2}{x_1 - x_2}\right) \\ x_3 &= c^2 - x_1 - x_2 \\ y_3 &= c \cdot (x_1 - x_3) - y_1 \end{aligned}$$

5. Voor $P = (x_1, 0)$ geldt $P + P = \infty$.

6. Voor $P = (x_1, y_1)$, $y_1 \neq 0$, wordt $P + P$ gevonden als $(x_3, -y_3)$ met
$$\begin{aligned} c &= \left(\frac{3x_1^2 + a}{2y_1}\right) \\ x_3 &= c^2 - x_1 - x_2 \\ y_3 &= c \cdot (x_1 - x_3) - y_1 \end{aligned}$$

Van de aldus gedefinieerde groepsoperatie is vrij gemakkelijk in te zien dat aan $\mathcal{G}1$, $\mathcal{G}2$, $\mathcal{G}3$ en $\mathcal{G}5$ is voldaan. Associativiteit ($\mathcal{G}4$) is af te leiden uit de ingewikkelde wiskundige achtergrond van de elliptische kromme of door het uitschrijven van $P_1 + (P_2 + P_3)$ en $(P_1 + P_2) + P_3$.

Kader 15.6: XTR: The game of the name.



XTR is in 1999 bedacht door Arjen Lenstra en Eric Verheul [LV00]. Het systeem heet *Efficient Compact Subgroup Trace Representation*, afgekort ECSTR, en nog verder afgekort tot XTR. Men spreke dit uit als het Nederlandse woord 'ekster' en de website van XTR is met een afbeelding van dit vogeltje gesierd.
Nieuwe cryptografische algoritmen maken altijd een trage start. De uitvinders hebben hun werk op conferenties gepresenteerd en hebben van een aantal mogelijke aanvallen aangegeven dat deze geen bedreiging voor XTR vormen. Ook de suggestie dat XTR niets anders is dan een vermomde elliptische kromme werd door Verheul [Ver01] weerlegd. Maar of het algoritme inderdaad duurzaam veilig is en of het op grote schaal zal worden gebruikt moet de toekomst leren.
Internet: `www.ecstr.com`.

Elliptische krommen in de cryptografische praktijk. Sinds de introductie van elliptische krommen in de cryptografie in 1985 is er veel onderzoek gedaan naar deze groepen, in het bijzonder naar het efficiënt vinden van logaritmen. De theorie achter elliptische krommen is bijzonder rijk, en er wordt een groot aantal speciale vormen onderscheiden.

Een voordeel van elliptische krommen ten opzichte van een multiplicatieve groep is dat veel kleinere elementen kunnen worden gebruikt. Een veiligheid, vergelijkbaar met 1024-bits-RSA, wordt al bereikt met krommen over een priemgetal van 200 à 240 bits. Dit resulteert in veel compactere waarden: een groepselement is circa 240 bits en een sleutel circa 160 bits. De berekeningen verlopen ook veel sneller, waardoor elliptische krommen zeer aantrekkelijk kunnen zijn voor toepassingen in smartcards of in drukke servers.

Een groot probleem dat optreedt bij cryptografische toepassing is het vinden van een geschikte kromme, dat wil zeggen, de keuze van p, a en b. De kromme moet namelijk een element hebben waarvan de orde een groot priemgetal q is en dit betekent dat $\#E_{p,a,b}$ een grote priemdeler moet hebben. De manier om geschikte parameters te vinden is verschillende keuzes uitproberen, en van de kandidaten de waarde $\#E_{p,a,b}$ te berekenen en te factoriseren. Berekening van de grootte van een kromme is echter duur, en er moeten doorgaans vrij veel keuzes worden geprobeerd. Het vinden van een kromme is daarom bewerkelijk en kan enkele uren rekentijd op een pc kosten. Om deze reden worden elliptische krommen vrijwel altijd toegepast met een gemeenschappelijke kromme voor alle gebruikers.

15.2.3 XTR

De groep XTR combineert de compacte representatie en berekening van elliptische krommen met de gemakkelijke sleutelgeneratie van de multiplicatieve groep.

Het eindig lichaam $GF(p^n)$. Berekeningen van XTR spelen zich af in een *eindig lichaam* (Engels: *finite field* of *Galois field*) $GF(p^n)$. Deze structuren zijn een zogenaamde *algebraïsche uitbreiding* van $\mathbb{Z}_p$. Het idee achter een algebraïsche uitbreiding wordt in kader 15.7 geïllustreerd. Een element van $GF(p^n)$ is een polynoom van graad $n-1$ met coëfficiënten in $\mathbb{Z}_p$, dus een uitdrukking van de vorm $p_0 + p_1 \cdot Z + \ldots + p_{n-1} \cdot Z^{n-1}$.

Kader 15.7: Algebraïsche uitbreiding: complexe getallen.

Met de reële getallen kan men optellen, aftrekken, vermenigvuldigen en delen, maar niet elke vergelijking heeft een oplossing: $x^2 + 1 = 0$ bijvoorbeeld niet. Het polynoom $Z^2 + 1$ is irreducibel over $\mathbb{R}$.
De verzameling polynomen in Z over $\mathbb{R}$ wordt genoteerd als $\mathbb{R}[Z]$ en we beschouwen deze verzameling *modulo* $Z^2 + 1$: $\mathbb{R}[Z]/(Z^2 + 1)$. Elk polynoom is equivalent met een eerstegraads polynoom, omdat een term van orde 2 of hoger kan worden vervangen door een lagere orde term, immers $a \cdot Z^{k+2} = -a \cdot Z^k$. Dus $\mathbb{R}[Z]/(Z^2+1) = \{a \cdot Z+b : a, b \in \mathbb{R}\}$. Voor optelling geldt

$$(a_1 Z + b_1) + (a_2 Z + b_2) = (a_1 + a_2)Z + (b_1 + b_2),$$

deze gebeurt dus puntsgewijs. Voor vermenigvuldiging geldt

$$\begin{aligned}(a_1 Z + b_1) \cdot (a_2 Z + b_2) &= a_1 a_2 Z^2 + (a_1 b_2 + a_2 b_1)Z + b_1 b_2 \\ &= (a_1 b_2 + a_2 b_1)Z + (b_1 b_2 - a_1 a_2)\end{aligned}$$

(het laatste wegens de reductieregel). Er kan ook worden gedeeld, want elk element $(a \cdot Z + b$, ongelijk 0) heeft een multiplicatieve inverse $\frac{-a}{a^2+b^2} Z + \frac{b}{a^2+b^2}$ (ga na!).
Onze moeite wordt beloond: In $\mathbb{R}[Z]/(Z^2 + 1)$ heeft de vergelijking $x^2 + 1 = 0$ een oplossing, namelijk $x_1 = 1 \cdot Z + 0$ en $x_2 = -1 \cdot Z + 0$. De verzameling $\mathbb{R}[Z]/(Z^2 + 1)$ heet $\mathbb{C}$ en het element Z ervan heet i.

Een element kan worden gerepresenteerd als n elementen van $\mathbb{Z}_p$ en dit kost $n \cdot \log p$ bits. Polynomen kunnen worden opgeteld en afgetrokken. Ook kunnen polynomen met elkaar worden vermenigvuldigd, maar dan ontstaat een productpolynoom van hogere graad $(2n - 2)$. Daarom wordt in $GF(p^n)$ gerekend modulo een vast polynoom M van graad n: door het product door dit polynoom te delen verkrijgen we een restpolynoom van graad $n - 1$. Hiermee is een vermenigvuldiging op $GF(p^n)$ gedefinieerd. Het vaste moduluspolynoom M moet *irreducibel* zijn: dit betekent dat er geen polynomen M_1 en M_2 van graad $\leq n-1$ bestaan met product M. Dit heeft als directe gevolg dat met de vermenigvuldiging die op $GF(p^n)$ is gedefinieerd, $GF(p^n)$ geen nuldelers kent: er kan daarom ook worden gedeeld in $GF(p^n)$.

De multiplicatieve groep van $GF(p^n)$ heeft $p^n - 1$ elementen (0 heeft geen inverse) en is cyclisch. Daarmee zijn vermenigvuldigingen in $GF(p^n)$ geschikt als groep voor cryptografische algoritmen die op de discrete logaritme zijn gebaseerd. Meer over eindige lichamen vindt men bij Garrett [Gar00] of Bach en Shallit [BS96].

Als aanduiding voor het eindig lichaam met p_n elementen komt men ook $\mathbb{F}_{p^n}$ tegen. De benaming Galois field (afkorting $GF(p^n)$) refereert aan de negentiende-eeuwse wiskundige Evariste Galois, een tijdgenoot van Niels Abel (kader 15.2). Ook Galois schreef geniale artikelen over vijfdegraadsvergelijkingen, die weinig werden gelezen en nog minder begrepen. De vroege negentiende eeuw was een ongezonde tijd voor geniale wiskundigen. Galois werd tot twintig jaar cel veroordeeld, werd in vrijheid gesteld na cholera te hebben overleefd, maar stierf op 31 mei 1832, twintig jaar oud, aan een schotwond die hij opliep in een duel (zie Guedj [Gue99]).

Subgroepen van $GF(p^6)$ en XTR. Er zijn geen efficiënte algoritmen bekend om discrete logaritmen in $GF(p^n)$ te berekenen. Het best bekende is een variant van de indexcalculus (sectie 15.4). Er wordt vermoed dat dit ten minste zo moeilijk is als het berekenen van

logaritmen in $\mathbb{Z}_{p'}$, waar p' een priemgetal is dat ongeveer zo groot is als p^n. Omdat de elementen van $GF(p^n)$ ook ongeveer zo groot zijn als elementen van $\mathbb{Z}_{p'}$ lijkt er weinig voordeel te behalen met het rekenen in een uitbreidingslichaam.

De voordelen van XTR berusten op het vinden van een speciale subgroep van $GF(p^6)$, waarmee groepsoperaties efficiënter zijn te berekenen, maar logaritmen niet. Het aantal elementen van $GF(p^6)$ is $p^6 - 1 = (p-1)(p+1)(p^2+p+1)(p^2-p+1)$ en de multiplicatieve groep heeft daarom subgroepen van orde $(p-1)$, $(p+1)$, (p^2+p+1) en (p^2-p+1). De XTR-groep is de subgroep van grootte p^2-p+1 en hiervan is te bewijzen dat hij niet is bevat in een kleiner lichaam dan $GF(p^6)$, waarmee logaritmen waarschijnlijk zo moeilijk zijn als in $GF(p^6)$.

De subgroep bevat echter slechts ongeveer p^2 elementen en het blijkt ook mogelijk deze te representeren door een *trace*, een combinatie van slechts twee getallen modulo p. Verder is het mogelijk, vermenigvuldigingen op elementen van de subgroep te simuleren door berekeningen met de traces. Hiermee wordt bereikt, dat alle cryptografische functies worden uitgevoerd op elementen van slechts $2 \cdot \log p$ bits.

Het vinden van een groep. De XTR-subgroep moet een element van hoge orde q, priem, hebben. Daarom wordt gezocht naar een priemgetal q en een p zo dat $q|p^2-p+1$, beide ongeveer 170 bits. Met een q van 170 bits gebruikt een groepsalgoritme op $\langle g \rangle$ (zoals Shanks' of Pollards) zo'n 2^{85} operaties en is daarmee onhaalbaar. Met een p van 170 bits heeft $GF(p^6)$ een omvang van 1020 bits en is het berekenen van logaritmen in $GF(p^6)$ vergelijkbaar met het factoriseren van een 1020-bitsgetal.

De private exponent is een getal uit $\mathbb{Z}_q$ dus zo'n 170 bits. De publieke sleutel is de trace van de generator, en deze is zo'n 340 bits lang. De berekeningen kosten ongeveer evenveel tijd als berekeningen in elliptische krommen met 170-bitsmoduli. Het grote voordeel van XTR ten opzichte van elliptische krommen is de veel eenvoudiger generatie van de groep. Daarmee is XTR bruikbaar voor toepassingen waar elke gebruiker zijn eigen groep genereert, zelfs als dit op een smartcard moet gebeuren.

15.3 Groepsalgoritmen

In deze sectie zijn g en y en groepselementen; gevraagd wordt naar de discrete logaritme van y (met grondtal g), dit is een getal x waarvoor $g^x = y$. Wanneer y niet in $\langle g \rangle$ zit bestaat zo'n x niet.

De orde van g is q. De algoritmen van Shanks (sectie 15.3.1) en Pollard (sectie 15.3.2) vinden allebei de waarde van x in $O(\sqrt{q})$ tijd. Het algoritme van Pohlig en Hellman (sectie 15.3.3) is een uitbreiding hiervan waarin op de factoren van q afzonderlijk wordt gewerkt. Als $q = r_1 \cdot r_2 \cdot \ldots \cdot r_k$ wordt de logaritme gevonden in $O(\sqrt{r_1} + \sqrt{r_2} + \ldots + \sqrt{r_k})$ tijd.

De genoemde algoritmen zijn allemaal groepsalgoritmen: ze gebruiken alleen de standaardoperaties zoals die in alle groepen zijn gedefinieerd. In sectie 15.3.4 wordt bewezen dat er geen groepsalgoritme bestaat dat de discrete logaritme sneller kan berekenen.

15.3.1 Shanks' grote stappen, kleine stappen

De methode van Shanks [Gar00, Sec. 27.1] wordt wel de *grote stappen, kleine stappen* methode genoemd.

De methode berust erop, dat als parameters m en t worden gekozen met $mt \geq q$, dan is x, met $0 < x < q$ te schrijven als $x = x_1 \cdot t - x_2$ voor $1 \leq x_1 \leq m$ en $0 \leq x_2 < t$. Immers, neem $x_1 = \lceil x/t \rceil$ en $x_2 = x_1 \cdot t - x$. Voor de betreffende x_1 en x_2 geldt $x + x_2 = x_1 \cdot t$ en dit is equivalent met $g^x \circ g^{x_2} = (g^t)^{x_1}$ en, als $y = g^x$, met $y \circ g^{x_2} = (g^t)^{x_1}$.

Shanks' methode legt een tabel aan met de m elementen $(g^t)^{x_1}$ voor $x_1 \leq m$. Vervolgens worden de t elementen $y,\ y \circ g,\ \ldots, y \circ g^{t-1}$ gezocht in de tabel. Zodra $y \circ g^{x_2}$ wordt gevonden als $(g^t)^{x_1}$ in de tabel volgt $x = x_1 \cdot t - x_2$.

Het aanmaken van de tabel kost ongeveer m groepsoperaties (en één exponentiatie om g^t te berekenen). Omdat er hier steeds met g^t wordt vermenigvuldigd lopen we hier met 'grote stappen' door $\langle g \rangle$. Het opslaan van de tabel kost m geheugenplaatsen. Het zoeken in de tabel kost (hoogstens) t groepsoperaties, hierbij wordt steeds met g vermenigvuldigd en we lopen door $\langle g \rangle$ met 'kleine stappen'. De totale kosten zijn dus ongeveer $m + t$ operaties en m opslag. Een populaire keuze is $m \approx t \approx \sqrt{q}$: de rekentijd is dan minimaal, namelijk $O(\sqrt{q})$.

Shanks' algoritme is heel simpel te implementeren en heeft nog twee aantrekkelijke eigenschappen.

1. Als y geen element van de subgroep is, termineert het algoritme eenvoudigweg (na het maximale aantal stappen) zonder x te vinden en deze bestaat dan niet.

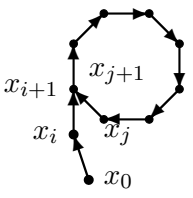
2. Het algoritme is ook te gebruiken in situaties waar weliswaar de orde van g zeer hoog is, maar een kleinere grens op x is gegeven: $x < M$ (waar $M \ll q$). Het algoritme vindt dan x in $O(\sqrt{M})$ tijd met $t \approx m \approx \sqrt{M}$.

Vaak is geheugenruimte een probleem. De parameter m kan worden aangepast aan de beschikbare hoeveelheid geheugen, waarbij natuurlijk t evenredig hoger moet worden genomen. De benodigde geheugenruimte is in een enigszins realistische situatie toch de beperkende factor in de toepassing, waarvan een kleine berekening. Ga uit van een hypothetische pc met een geheugen voor een biljoen groepselementen, dit zijn er ongeveer 2^{40}, en waarvoor een berekening van 2^{60} stappen haalbaar is. Qua rekentijd zou deze machine een groep met een q van 120 bits kunnen aanvallen, maar door de beperking $m \leq 2^{40}$ komt de machine niet verder dan 100 bits.

15.3.2 Pollards rho-methode.

Het algoritme van Pollard [Gar00, Sec. 27.2] heeft dezelfde complexiteit *in tijd* als Shanks' methode, maar zonder dat daarbij veel geheugen nodig is. De hypothetische pc uit de vorige alinea zou inderdaad logaritmen in een groep van 2^{120} elementen kunnen uitrekenen. Helaas leveren we hiervoor de drie prettige eigenschappen van Shanks' methode in. Het algoritme is wat moeilijker te begrijpen. Bovendien kan het algoritme alleen x zoeken in de complete ruimte $0 \leq x < q$ en niet sneller in een beginstuk ervan. En ten slotte, als y buiten $\langle g \rangle$ ligt termineert het algoritme niet; het is dus niet als inclusietest te gebruiken.

Pollards algoritme is gebaseerd op de ontwikkeling van een reeks elementen in H volgens een functie f, waarna de reeks gedefinieerd is door $z_{i+1} = f(z_i)$. Omdat H eindig veel elementen bevat moet er vroeg of laat een element herhaald voorkomen, dus $z_{j+1} = z_{i+1}$ terwijl $z_i \neq z_j$; we noemen dit een *botsing*. Uiteraard geldt dan ook $z_{j+2} = z_{i+2}$ enzovoort, de reeks wordt vanaf plaats i periodiek met periodelengte $j - i$. Een plaatje van zo'n reeks kan (mits men zich voorstelt vanaf z_0 'omhoog' te gaan en dan naar rechts) enige gelijkenis vertonen met de Griekse letter ρ (rho).



repeat $a := random$; $b := random$; $z := y^a \circ g^b$;
  $(t, c, d) := (z, a, b)$;
  repeat $(z, a, b) := F(z, a, b)$;
    $(t, c, d) := F(F(t, c, d))$
  until $z = t$
until $(a - c) \in \mathbb{Z}_q^*$;
return $(d - b)(a - c)^{-1}$

Algoritme 15.8: POLLARDS RHO-METHODE.

Hieraan ontleent de methode zijn naam; hij had dus evenzogoed de 6-methode, 9-methode, sigma-methode of krul-methode kunnen heten.

Daar H precies q elementen bevat, treedt een botsing na ten hoogste q stappen op. Wanneer echter f een random functie is, treedt deze al na een verwacht aantal van ongeveer $\sqrt{q}$ stappen op. Het algoritme ontwikkelt daarom een reeks op basis van een functie met voldoende random eigenschappen totdat een botsing wordt gevonden. Uit de botsing wordt $\log y$ berekend. Bij de uitwerking van dit principe moeten enkele problemen worden opgelost, waaronder:

1. De keuze van de functie f.
2. Hoe kan een botsing worden gedetecteerd zonder de noodzaak alle elementen van de reeks op te slaan?
3. Hoe kan uit een botsing $\log y$ worden berekend?

De keuze van f. De functie f moet snel te berekenen zijn omdat hij in elke iteratie wordt uitgevoerd (meermalen zelfs, zie verderop); het liefst heeft f slechts één groepsoperatie. Toch moet hij in zekere mate de eigenschappen hebben van een random functie; bij zo'n functie is de verwachte grootte van de rho voordat een botsing optreedt $\sqrt{q}$ (stelling A.6). Een geschikte keuze blijkt te zijn, z te vermenigvuldigen met óf y, óf g, óf z. Er wordt een hashfunctie $k : G \to \{1, 2, 3\}$ gekozen of verondersteld, die G op min of meer willekeurige wijze in drie delen verdeelt, en dan is

$$f(z) = \begin{cases} z \circ y & \text{als } k(z) = 1 \\ z \circ g & \text{als } k(z) = 2 \\ z \circ z & \text{als } k(z) = 3 \end{cases}$$

Botsingsdetectie. Botsingen zijn eenvoudig te detecteren door alle elementen van de reeks op te slaan en met elke z_j te zoeken onder de reeds opgeslagen elementen; dit vereist echter $O(\sqrt{q})$ geheugenplaatsen. In plaats daarvan wordt voor de botsingsdetectie een tweede reeks t ontwikkeld die dezelfde onwikkeling met f doorloopt, maar in dubbel tempo. Er geldt dus steeds $t_i = z_{2i}$ en t wordt berekend als $t_0 = z_0$ en $t_{i+1} = f(f(t_i))$.

Zij nu z_{j_0} de eerste botsing, en wel $z_{j_0} = z_{i_0}$ $(i_0 < j_0)$. De periodelengte p is $j_0 - i_0$ en nu geldt voor elke $i > i_0$ en elk geheel aantal perioden s dat $z_i = z_{i+s \cdot p}$. Neem nu voor i het eerste veelvoud van p dat ten minste gelijk aan i_0 is en vind $z_i = z_{2i} = t_i$. Het voordeel van het gelijktijdig ontwikkelen van *twee* reeksen is dat voor een botsing niet langer hoeft te worden gezocht in de complete voorgaande reeks (wat $\sqrt{q}$ geheugen zou kosten), maar slechts op één plaats tegelijk in de andere reeks.

Berekening bij een botsing. Om uit een botsing de logaritme te kunnen berekenen worden bij de reeks z twee reeksen a en b bijgehouden met de eigenschap dat $z_i = y^{a_i} \circ g^{b_i}$. Daartoe wordt z_0 gekozen als $y^{a_0} \circ g^{b_0}$ voor random a_0 en b_0, en bij de evaluatie van f kunnen deze waarden eenvoudig worden bijgewerkt:

$$F(z,\, a,\, b) = \begin{cases} (z \circ y,\ a+1,\ b) & \text{als } k(z) = 1 \\ (z \circ g,\ a,\ b+1) & \text{als } k(z) = 2 \\ (z \circ z,\ 2a,\ 2b) & \text{als } k(z) = 3 \end{cases} .$$

Voor de reeks t worden c en d bijgehouden zodat $t_i = y^{c_i} \circ g^{d_i}$. Bij een gedetecteerde botsing geldt $z_i = t_i$, dus $y^{a_i} \circ g^{b_i} = y^{c_i} \circ g^{d_i}$, oftewel $x \cdot (a_i - c_i) = (d_i - b_i)$. Als $(a_i - c_i)$ een inverse heeft modulo q is hiermee x gevonden (als $\frac{d_i - b_i}{a_i - c_i}$). Als $(a_i - c_i)$ geen inverse heeft (bijvoorbeeld 0 is) is de poging onsuccesvol en wordt met een nieuw startpunt begonnen. De kans hierop is echter heel klein.

Het algoritme van Pollard gebruikt per iteratie drie groepsoperaties tegen Shanks' algoritme één. Verder is er geen garantie op terminatie van het algoritme, zelfs als $y \in \langle g \rangle$. Als $y \notin \langle g \rangle$ ontwikkelt de reeks zich niet binnen $\langle g \rangle$ maar in een grotere verzameling (soms heel G) waardoor botsingen langer uitblijven. Bovendien leveren deze botsingen nooit succes op.

15.3.3 Pohlig-Hellman

De algoritmen van Shanks en Pollard werken in $\sqrt{q}$ tijd en gebruiken geen speciale eigenschappen van q, zoals primaliteit of een bekende factorisatie. Pohlig en Hellman hebben deze aanpak uitgebreid naar groepen van *samengestelde* orde q, waarbij de factoren bekend zijn. In het algemeen is het getal q te schrijven als $q = \prod_i r_i^{e_i}$, waarin de r_i de verschillende priemfactoren van q zijn en e_i is het aantal factoren r_i dat q bevat.

Het Pohlig-Hellman-algoritme gebruikt dan een van de vorige algoritmen (Shanks of Pollard) *per factor* van q. Er wordt namelijk e_i keer een logaritme gezocht in een subgroep met r_i elementen. Gebruikmakend van een van de vorige algoritmen resulteert een rekentijd van $\sum_i e_i \cdot \sqrt{r_i}$.

De reductie van een samengestelde orde naar een priemorde van de voortbrenger verloopt in twee stappen. We behandelen eerst het geval dat q verschillende priemfactoren bevat; dan is q te schrijven als $q = r \cdot s$, waar $\text{ggd}(r,\, s) = 1$. Door herhaalde toepassing (steeds met r gekozen als $r_i^{e_i}$) wordt het algemene geval $q = \prod_i r_i^{e_i}$ gereduceerd tot groepen met orde $r_i^{e_i}$. De aanpak in het geval de orde een macht van een priemgetal is is anders en wordt vervolgens beschreven.

Als q verschillende priemfactoren bevat. Veronderstel eerst dat $q = r \cdot s$, waar r en s geen factoren gemeen hebben. De gezochte x is een element van $\mathbb{Z}_q$, maar met de Chinese reststelling kan x worden gevonden zodra $x_r = x \bmod r$ en $x_s = x \bmod s$ bekend zijn. We concentreren ons op het vinden van x_r.

In het volgende bewijs schrijven we een exponent i van g steeds als het paar $[i_r,\, i_s]$, omdat op die manier het duidelijkst uitkomt hoe het exponentiëren uitwerkt. Het verheffen tot de s-demacht is hetzelfde als een vermenigvuldiging met s in de exponent. Die exponent behoort tot $(\mathbb{Z}_r \times \mathbb{Z}_s)$, en het tweede deel ervan wordt dan gelijk aan 0 omdat dit deel in $\mathbb{Z}_s$ wordt gerekend. Anders gezegd, de invloed van x_s verdwijnt en daarom is $(g^x)^s$ gelijk aan $(g^{x_r})^s$ ofwel $(g^s)^{x_r}$.

Lemma 15.9 $y^s = (g^s)^{x_r}$.

Bewijs. Voor y^s geldt

$$\begin{array}{rcll} y^s & = & (g^{[x_r,\,x_s]})^s & \text{uitschrijven } y \text{ als } g^x \\ & = & g^{s\cdot[x_r,\,x_s]} & \\ & = & g^{[s\cdot x_r,\,s\cdot x_s]} & \\ & = & g^{[s\cdot x_r,\,0]} & \text{tweede deel in } \mathbb{Z}_s \end{array}$$

en voor $(g^s)^{x_r}$ geldt

$$\begin{array}{rcll} (g^s)^{x_r} & = & ((g^{[1,\,1]})^s)^{x_r} & \text{uitschrijven } g \text{ als } g^1 \\ & = & (g^{[s\cdot 1,\,s\cdot 1]})^{x_r} & \\ & = & (g^{[s,\,0]})^{x_r} & \text{tweede deel in } \mathbb{Z}_s \\ & = & g^{[s\cdot x_r,\,0]} & \end{array}$$

□

Het nut hiervan is dat we x_r vinden als een logaritme met grondtal g^s, en hiervan is de orde kleiner zodat die logaritme sneller kan worden gevonden.

Lemma 15.10 *De orde van g^s is r.*

Bewijs. Omdat $g^q = e$ is $(g^s)^r = e$. Omdat er geen $q' < q$ is waarvoor $g^{q'} = e$ is er geen $r' < r$ waarvoor $(g^s)^{r'} = e$. De orde van (g^s) is dus r. □

Stelling 15.11 *De waarde x_r is gelijk aan ${}^{g'}\log y'$, waar $g' = g^s$ een element van orde r is en $y' = y^s$.*

Voor het algemene geval $q = \prod_{i=1}^{v} r_i^{e_i}$ wordt de voorgaande stelling v keer toegepast, steeds op $r = r_i^{e_i}$ en $s = q/(r_i^{e_i})$. Men vindt dan, voor $i = 1, \ldots, v$:

$$y_i = y^{q\,/\,r_i^{e_i}} \qquad g_i = g^{q\,/\,r_i^{e_i}} \qquad x_i = {}^{g_i}\log y_i$$

Uit x_1 tot en met x_v wordt met de Chinese reststelling x gevonden.

Als q een priemmacht is. We beschouwen nu het geval $q = r^e$; merk allereerst op dat $\mathrm{ord}(g^{q/r}) = r$. Het berekenen van de logaritme met grondtal g wordt gesplitst in e logaritmen met grondtal $g^{q/r}$.

Net als in het vorige geval profiteren we van een andere schrijfwijze van getallen in $\mathbb{Z}_q$. Daar gebruikten we de Chinese reststelling, hier wordt een getal geschreven als een e-cijferig getal in de radix-r notatie. Anders gezegd, we kunnen $i \in \mathbb{Z}_q$ schrijven als $i_0 + i_1 \cdot r + \ldots + i_{e-1} \cdot r^{e-1}$, voor $i_0, \ldots, i_{e-1} < r$, en noteren $[i_0, .., i_{e-1}]_r$.

Het vinden van het eerste 'cijfer' x_0 gaat analoog aan het vorige geval. Door y en g te verheffen tot de macht q/r (dit is r^{e-1}) verdwijnt de invloed van hogere cijfers.

Lemma 15.12 $y^{q/r} = (g^{q/r})^{x_0}$.

Bewijs. Omdat de exponent in het r-tallig stelsel is uitgeschreven doet vermenigvuldigen met een macht van r de cijfers van de exponent naar rechts opschuiven.

$$\begin{array}{rcll} y^{(r^{e-1})} & = & (g^{[x_0,\ldots,x_{e-1}]_r})^{(r^{e-1})} & \text{uitschrijven } y = g^x \\ & = & g^{r^{e-1}\cdot[x_0,\ldots,x_{e-1}]_r} & \\ & = & g^{[0,..,0,x_0,\ldots,x_{e-1}]_r} & \text{exponent begint met } e-1 \text{ nullen} \\ & & & \text{LET OP! Is nu } 2e-1 \text{ cijfers lang.} \\ & = & g^{[0,..,0,x_0]_r} & \text{alleen de eerste } e \text{ cijfers} \end{array}$$

Omdat de exponent modulo $q = r^e$ wordt gerekend, kunnen alle cijfers behalve de eerste e verwaarloosd worden.

$$\begin{aligned}(g^{(r^{e-1})})^{x_0} &= ((g^{[1,0,..,0]_r})^{(r^{e-1})})^{x_0} \quad \text{uitschrijven } g = g^1\\ &= (g^{r^{e-1}\cdot[1,0,..,0]_r})^{x_0}\\ &= (g^{[0,..,0,1]_r})^{x_0}\\ &= g^{[0,..,0,x_0]_r}\end{aligned}$$

□

De verdere cijfers van x worden op soortgelijke manier geïsoleerd, maar er is een extra stap nodig. Om cijfer x_j te isoleren kan men y verheffen tot de macht r^{e-j-1} om de cijfers van *hogere* orde (dus x_{j+1} t/m x_{e-1}) te laten verdwijnen. De cijfers van lagere orde blijven dan echter aanwezig, als in:

$$r^{e-j-1} \cdot [x_0, x_1, .., x_j, ..x_{e-1}]_r = [0, ..., 0, x_0, x_1, .., x_j]_r$$

Daarom wordt x_j pas bepaald als x_0 tot en met x_{j-1} al bekend zijn, waarna we deze cijfers uit de exponent kunnen laten verdwijnen door y te delen door $g^{[x_0,..,x_{j-1},0,..,0]_r}$. Voor de berekende $y' = y \circ (g^{[x_0,..,x_{j-1},0,..,0]_r})^{-1}$ geldt

$$\begin{aligned}y' &= y \circ (g^{[x_0,..,x_{j-1},0,..,0]_r})^{-1}\\ &= (g^{[x_0,x_1,..,x_j,..x_{e-1}]_r}) \circ (g^{[x_0,..,x_{j-1},0,..,0]_r})^{-1}\\ &= g^{[x_0,x_1,..,x_j,..x_{e-1}]_r - [x_0,..,x_{j-1},0,..,0]_r}\\ &= g^{[0,..,0,x_j,..x_{e-1}]_r}\end{aligned}$$

Vervolgens wordt berekend $y'' = y'^{r^{e-j-1}}$ en voor de berekende y'' geldt

$$\begin{aligned}y'' &= y'^{r^{e-j-1}}\\ &= (g^{[0,..,0,x_j,..x_{e-1}]_r})^{r^{e-j-1}}\\ &= g^{(r^{e-j-1}\cdot[0,..,0,x_j,..x_{e-1}]_r)}\\ &= g^{[0,..,0,..,0,x_j]_r}\\ &= (g^{q/r})^{x_j}\end{aligned}$$

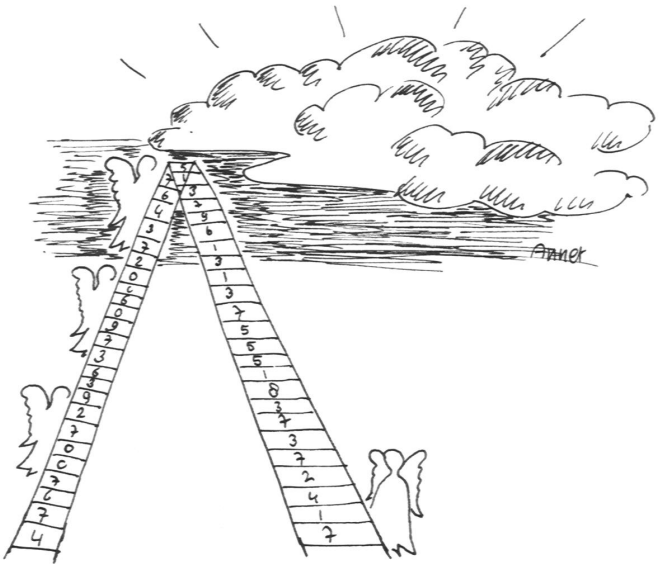
Hieruit kan het volgende cijfer x_j worden berekend als logaritme met grondtal $(g^{q/r})$, dus in een groep van grootte r.

Consequenties voor sleutelkeuze. Bij het kiezen van een cryptografisch algoritme en de parameters daarvoor moet men er natuurlijk voor zorgen dat alle bekende aanvallen ondoenlijk zijn. De algoritme voor discrete log zoals behandeld in deze sectie zijn daarom van direct belang voor de ontwikkeling van de cryptografie.

Met Pohlig-Hellman wordt de logaritme in een groep van orde $q = \prod_i r_i^{e_i}$ uitgesplitst in logaritmen in groepen van orde r_i. Bepalend voor de moeilijkheid van logaritmen in $\langle g \rangle$ is dus alleen de *grootste priemfactor* van q.

Subgroep-ElGamal. De eerste versie van ElGamals cryptosysteem gebruikte een generator van $\mathbb{Z}_p^*$, dus een element van orde $p-1$. Al snel werd duidelijk dat $p-1$ ten minste één grote priemfactor q moest bevatten als bescherming tegen aanvallen met Pohlig-Hellman. Dus werd p gekozen als $q \cdot k + 1$, waarbij k kleine factoren mocht bevatten. Is dan a een geheime sleutel in $\mathbb{Z}_{p-1}$ en b de publieke sleutel $b = g^a \in \mathbb{Z}_p^*$, dan is a mod k snel te vinden met Pohlig-Hellman. De omvang van het priemgetal q weerhoudt de aanvaller alleen van het vinden van a mod q. Dit leidt tot een besparing op sleutellengte en berekening: in plaats van een echte generator van $\mathbb{Z}_p^*$ kiest men een element van orde q en een geheime sleutel in $\mathbb{Z}_q^*$.

Kader 15.9: De Klima-Rosa-aanval op ElGamal in PGP



De geheime sleutels in PGP zijn goed beschermd tegen uitlezen door buitenstaanders, maar minder goed beschermd tegen manipulatie. Vlastimil Klima en Tomas Rosa [KR01] uit Praag ontdekten dat het mogelijk is sommige parameters te wijzigen.
Bij het ondertekenen van een bericht (sectie 7.2.3) wordt voor een random k het getal $c = g^k \pmod p$ in de handtekening opgenomen. De aanval van Klima en Rosa bestaat uit het vervangen van g en p door getallen g' en p' waarvoor logaritmen kunnen worden berekend! Getal p' is $167 \cdot 2^{151} + 1$, zodat $p' - 1$ en de orde van g' alleen kleine factoren bevatten, waardoor de logaritme met Pohlig-Hellman te berekenen is. Als k eenmaal bekend is, kan uit de handtekening ook de geheime sleutel worden berekend. Hierna worden ongemerkt de oude g' en p' weer hersteld.

Aanval op subgroep-ElGamal. Oscar kan met een gekozen-cijfertekstaanval de macht u^a van een door hem gekozen getal verkrijgen. Bij een gekozen-cijfertekstaanval stuurt Oscar Alice een bericht (u, v) en verkrijgt de ontsleutelde waarde x, in dit geval $x = v/u^a$, waaruit $z = u^a$ te berekenen is als v/x. Bij een normaal bericht is u een macht van g en heeft dus orde q, waarmee het berekenen van a uit z niet mogelijk is. Echter, Oscar kan een u kiezen waarvan de orde groter is dan q, maar alleen uit kleine factoren bestaat (mits $k = (p-1)/q$ voldoende veel kleine factoren heeft). De logaritme met grondtal u is dan met Pohlig-Hellman te berekenen.

15.3.4 Ondergrens

De algoritmen van Shanks en Pollard zijn met $O(\sqrt{q})$ rekentijd de snelst bekende groepsalgoritmen voor de logaritme in een groep van priem grootte q. In 1997 bewees Victor Shoup [Sho97] dat er geen beter groepsalgoritme kan bestaan: als een groepsalgoritme m stappen doet op invoer g en y en een getal x uitvoert, is de kans dat $g^x = y$ begrensd door m^2/q. Met andere woorden, voor een succeskans ϵ zijn ten minste $m = \sqrt{q \cdot \epsilon}$ stappen nodig.

De redeneringen in het bewijs zijn niet heel gecompliceerd, maar er is een goede modellering nodig van de werking van een algoritme A op een *willekeurige* groep. Als groepsoperaties staan we toe: het berekenen van $a \circ b$ of $a \circ b^{-1}$ voor bekende a en b, en het testen op gelijkheid $a = b$. Verder staan we toe het trekken van een random element uit $\langle g \rangle$; hiervan heeft geen van de tot nu toe behandelde algoritmen gebruikgemaakt.

Om af te dwingen dat het algoritme voor *elke* groep van orde q werkt, hoe deze ook gedefinieerd is, wordt de groep beschreven als een *random codering* van $\mathbb{Z}_q$. De codering is niets anders dan een injectieve functie $\sigma : \mathbb{Z}_q \to \mathbb{B}^+$, dat wil zeggen, een toekenning van verschillende bitrijen aan de getallen uit $\mathbb{Z}_q$. De groep S bestaat uit de verzameling voorkomende bitrijen, dus $S = \sigma(\mathbb{Z}_q)$ en de operatie $\circ$ is gedefinieerd door $\sigma(i) \circ \sigma(j) = \sigma(i + j)$.

Merk op, dat in alle daadwerkelijk gebruikte systemen de codering helemaal geen random functie is. De codering is juist heel structureel zodat $\circ$ effectief kan worden berekend. Bijvoorbeeld in $\mathbb{Z}_p^*$ is $\sigma(i)$ precies de binaire beschrijving van het getal g^i en het getal g^{i+j} is (snel) te berekenen uit de getallen g^i en g^j. Maar een groepsalgoritme mag van de structurele eigenschappen van de codering geen gebruikmaken. Omdat over de codering niets is gegeven dan dat het een injectie is, is op de groepselementen alleen een test op gelijkheid zinvol, immers $\sigma(i) = \sigma(j) \iff i = j$.

Natuurlijk moet A wel de groepsoperatie kunnen evalueren maar hiervoor nemen we aan dat van een tabel gebruik wordt gemaakt, waarin, gegeven a, $b \in S$, de waarde $a \circ b$ kan worden opgezocht. (In de informatica wordt zo'n tabel een *orakel* genoemd: het antwoord is altijd juist, maar hoe het orakel er aankomt blijft altijd raadselachtig.)

Een algoritme A dat na m groepsoperaties een uitvoer geeft, kunnen we nu als volgt modelleren. Er wordt een rij $a_1, a_2, \ldots, a_m$ van m groepselementen gevormd, telkens door toepassing van een van de toegestane groepsoperaties. In het begin beschikt A over de identiteit e, de generator g en de invoer y en dat zijn de eerste drie bekende groepselementen:

$$a_1 = e, \quad a_2 = g, \quad a_3 = y.$$

Er wordt pas echt gerekend vanaf stap 4, en a_k kan worden gevormd als $a_i \circ a_j$ of $a_i \circ a_j^{-1}$ voor $i, j < k$ of door random trekking uit G. Welke stap wordt uitgevoerd, kan A laten afhangen van de eerder geziene waarden $a_1, \ldots, a_{k-1}$. Omdat nu testen op gelijkheid de enige operatie is die A op de bitrijen kan uitvoeren hangt elke volgende stap dus alleen af van eventuele gelijkheden tussen die eerdere waarden.

Het bewijs kijkt nu eerst naar de werking van A ingeval *alle elementen a_i verschillend zijn.* De kans op een goed antwoord $x = \log y$ is dan uiterst gering, namelijk $1/q$, voor een random invoer y. Redelijke kans op succes voor A is er dus alleen in executies waarin er gelijkheden $a_i = a_j$ voorkomen, en de kans daarop kan als (m^2/q) worden afgeschat voor een random codering σ.

Lineaire expressies. We willen A confronteren met een reeks elementen waarin geen botsingen voorkomen, maar dit is, afhankelijk van wat A precies doet, niet altijd mogelijk. Een voorbeeld hiervan; veronderstel dat A de volgende berekeningen uitvoert:

	Stappen van A
36	$h_1 := \mathit{random}$
37	$h_2 := h_1 \circ y^{-1}$
38	$h_3 := g \circ y$
39	$h_4 := g \circ h_1$
40	$h_5 := h_2 \circ h_3$

De waarde van h_5 *moet* gelijk zijn aan h_4, wat men als volgt kan inzien. Element h_1 wordt random gekozen, maar is dan gelijk aan g^{x_2} voor zeker getal $x_2 \in \mathbb{Z}_q$. De invoer y is gelijk aan g^{x_1} (voor zekere x_1 die A probeert uit te rekenen). Dan is $h_2 = g^{x_2 - x_1}$, $h_3 = g^{1+x_1}$, $h_4 = g^{1+x_2}$ en $h_5 = g^{(x_2-x_1)+(1+x_1)} = g^{x_2+1}$. De exponenten van h_4 en h_5 zijn dus gelijk, *ongeacht de keuze van y en h_1.*

We gaan nu voor elke rekenstap van A de exponent van het nieuwe element uitdrukken in exponenten van de random gekozen getallen. Samen met de eerste drie (ons al bekende) stappen van A ziet bovenstaand voorbeeld er dan zo uit:

Stap	1	2	3	... ...	36	37	38	39	40	... ...
a_i	e	g	y	... ...	h_1	h_2	h_3	h_4	h_5	... ...
F_i	0	1	X_1	... ...	X_2	$X_2 - X_1$	$X_1 + 1$	$X_2 + 1$	$X_2 + 1$	... ...

De uitdrukkingen F_i noemen we een *lineaire expressie* over $\mathbb{Z}_q$; deze hebben allemaal de vorm $a_0 + \sum_j a_j \cdot X_j$, waar X_j een formele variabele is. Ze worden als volgt opgebouwd. De eerste drie elementen die A ziet zijn e, g en y, en hierbij horen de expressies 0, 1 en X_1. Als a_k random wordt gekozen hoort hierbij de expressie X_j, waar X_j de (laagstgenummerde) variabele is die in de voorgaande F_i nog niet voorkomt. Als a_k wordt berekend als $a_i \circ a_j$ hoort hierbij de expressie $F_i + F_j$, merk op dat dit weer een *lineaire* expressie is. Bij $a_k := a_i \circ a_j^{-1}$ hoort uiteraard $F_k = F_i - F_j$.

Nu kan tijdens de berekening van A gelijkheid van twee elementen a_i en a_j op twee manieren voorkomen. (1) Als $F_j = F_i$ doet A een rekenstap die *in alle gevallen*, dus bij alle invoeren, onafhankelijk van de random gekozen elementen, leidt tot $a_j = a_i$. Gelijkheid volgt hier uit de eigenschappen van de groepsoperator. (2) Als $F_i \neq F_j$ zal gelijkheid van a_i en a_j *niet noodzakelijk* in alle executies voorkomen, maar afhankelijk van de random gekozen waarden kan het wel. In bovenstaand voorbeeld zou h_3 gelijk zijn aan h_2 als de invoer $y = g^{x_1}$ en de random gekozen $h_1 = g^{x_2}$ zouden voldoen aan $x_2 - x_1 = x_1 + 1$, ofwel $x_2 = 2 \cdot x_1 + 1$, ofwel $h_1 = y^2 \circ g$. De kans hierop is uiteraard $\frac{1}{q}$ en deze kans op gelijkheid geldt voor alle i, j met ongelijke F_i en F_j (lemma 15.13).

We gebruiken de hoofdletter als symbolische variabele X in de lineaire expressies. Voor een specifieke toekenning x_1, x_2, ... schrijven we $F(x)$ voor het getal in $\mathbb{Z}_q$ dat ontstaat door substitutie van het getal x_k voor de variabele X_k.

Lemma 15.13 *Als F_i en F_j* ongelijke *lineaire expressies zijn is, bij een random toekenning x_1, x_2, ...: $Pr(F_i(x) = F_j(x)) \leq 1/q$.*

Bewijs. Zij F de expressie $F_i - F_j$. Als F de nulexpressie is, is uiteraard $F_i(x) = F_j(x)$ voor elke toekenning, maar omdat $F_i \neq F_j$ is F ongelijk nul. Als F een constante expressie is ($F = a_0$ voor $a_0 \in \mathbb{Z}_q$ ongelijk 0), dan is $F_i(x)$ zeker ongelijk $F_j(x)$ voor alle toekenningen, dus de kans op gelijkheid is nul.

Zij ten slotte F een niet-constante expressie, dat wil zeggen, er zit ten minste één symbolische variabele in en F is te schrijven als $a_k \cdot X_k + F'$. Dan is $F(x)$ gelijk aan 0 in precies die toekenningen waarin $x_k = -F'(x) \cdot a_k^{-1}$, en dat is precies $\frac{1}{q}$ deel van alle toekenningen. De kans op $F_i(x) = F_j(x)$ is dus exact $\frac{1}{q}$. □

Basisexecutie en kans op correctheid. Beschouw nu als basisexecutie van A de reeks R van stappen die A doet als er geen gelijke elementen worden gevonden (behalve daar, waar gelijkheid volgt uit de groepseigenschap). Reeks R kan worden gedefinieerd door de tabel met de groepsoperatie *tijdens* de executie op te bouwen. Steeds als A een element a_k berekent, kijken we eerst naar de expressie F_k. Is F_k gelijk aan een eerdere expressie F_i, dan krijgt A de string a_i terug als resultaat van de stap. Ontstaat er een nieuwe expressie F_k, dan krijgt A als a_k een nieuwe string a_k, ongelijk aan a_1, ..., a_{k-1}. Na m stappen levert A in reeks R een getal x_R op.

Nu bekijken we de reeks R_y van stappen waar A wordt uitgevoerd op een random invoer y en bij elke random trekking inderdaad een random element g^{x_k} krijgt. De groepscodering σ is willekeurig. We onderscheiden deze drie mogelijkheden, aan de hand waarvan de kans op een correct antwoord kan worden afgeschat:

1. In de reeks R_y doet A dezelfde stappen als in R (omdat er dezelfde gelijkheden in voorkomen) en het antwoord x_R is correct.

2. In de reeks R_y doet A andere stappen dan in R omdat er gelijkheden $a_j = a_i$ optreden waar in reeks R geldt $a_j \neq a_i$.

3. In de reeks R_y doet A dezelfde stappen als in R (omdat er dezelfde gelijkheden in voorkomen) maar het antwoord x_R is niet gelijk aan $x = \log y$.

De kans op gebeurtenis (1) is hoogstens $1/q$ omdat y een random groepselement is en dus is $Pr(\log y = x_R) = 1/q$. De kans op gebeurtenis (2) is kleiner dan $\frac{1/2 \cdot m(m-1)}{q}$ omdat voor elk paar (i, j) de kans op een botsing tussen $a_i = g^{F_i(x)}$ en $a_j = g^{F_j(x)}$ ten hoogste $1/q$ is (lemma 15.13). In geval van gebeurtenis (3) is het antwoord van A fout. De conclusie is, dat voor een random invoer de kans dat A een juist antwoord geeft begrensd is door $\frac{1/2 \cdot m(m-1)+1}{q}$.

15.4 De indexcalculus

De snelst bekende algoritmen voor discrete logaritmen zijn de indexcalculus en algoritmen die hiervan weer een verbetering zijn, zoals de *Number Field Sieve*. We beschrijven hier kort het principe van de indexcalculus zoals toegepast op logaritmen in $\mathbb{Z}_p^*$ waarbij g een generator is (dus $\text{ord}(g) = p - 1$).

Rekentijd en geheugengebruik zijn van dezelfde grootteorde als de overeenkomstige algoritmen voor factoriseren. Rond het jaar 2000 lag de grens van toepassing van deze techniek op ongeveer 512 bits. Verbetering van technieken en beschikbaarheid van veel rekenkracht leidden tot deze situatie in 2015 [ABD+15]: (1) Voor een 512-bit modulus kun je met eenvoudige middelen in een week een tabel berekenen voor modulus p, waarmee een logaritme in ongeveer een minuut op te zoeken is. (2) Een goed geëquipeerd academisch team of criminele bende kan een tabel opstellen en daarna snel logaritmen uitrekenen voor een 768-bit modulus. (3) Een sterke natie (bv. China of de VS) kan voldoende rekenkracht mobiliseren om 1024-bits moduli aan te vallen, en heeft dit waarschijnlijk al gedaan.

15.4.1 Relaties

De indexcalculus vat willekeurige getallen in $\mathbb{Z}_p$ op als elementen van $\mathbb{Z}$ en gebruikt hier de mogelijkheid getallen uniek te ontbinden in priemfactoren. Het is niet de bedoeling dat alle willekeurige getallen worden ontbonden; immers, het vinden van de volledige ontbinding van een willekeurig getal is zelf een zeer moeilijk probleem.

Het algoritme maakt gebruik van een *factor basis* B van 'kleine' priemgetallen: $B = \{p_1, \ldots, p_b\}$, de eerste b priemgetallen, en zo dat $p_b \ll p$. Een typische keuze, bij p een getal van 512 bits, is om b ongeveer vier à acht miljoen te nemen en B bevat dan alle priemgetallen kleiner dan ongeveer honderd miljoen.

Een random element $z \in \mathbb{Z}_p$ wordt afgebeeld op het getal $z \in \mathbb{Z}$, $0 \leq z < p$, en dan wordt gekeken of dit getal z volledig factoriseert over B. Uiteraard is dat niet voor elke z het geval, want z kan priemfactoren $> p_b$ hebben. Als z wel volledig factoriseert, is z te schrijven als $z = \prod_j p_j^{e_j}$ en zo'n geslaagde factorisatie noemen we een *relatie*.

Een belangrijke kwestie bij de indexcalculus is de vraag, hoeveel willekeurige z men naar verwachting zou moeten proberen om een relatie te verkrijgen. Een getal z heet *r-glad* (Engels: *r-smooth*) als alle priemdelers van z kleiner dan r of gelijk aan r zijn. De factorisatie van z

over B slaagt als z p_b-glad is. Het aantal getallen z tot een zekere grens X dat r-glad is wordt genoteerd als $\Psi(X, r)$. Bach en Shallit [BS96, Stelling 8.8.11] geven een afschatting van het aantal r-gladde getallen.

Feit 15.14 $\Psi(X, r) \geq X^{1-\frac{\log\log X}{\log r}}$.

Voor een random gekozen $z < p$ is de kans dat de factorisatie slaagt dan $\frac{\Psi(p,p_b)}{p}$, en het verwachte aantal pogingen voor één factorisatie is (volgens stelling A.5) $\frac{p}{\Psi(p,p_b)}$. Voor p een 512-bitsgetal en p_b ongeveer honderd miljoen is het aantal pogingen ongeveer tien miljoen.

Een factorisatie kan sneller worden gevonden naarmate b (en dus p_b) groter wordt gekozen, er zijn dan immers meer getallen die factoriseren over B.

15.4.2 Tabel van logaritmen van de basis

Om relaties te gebruiken om $\log y$ te berekenen veronderstellen we eerst dat we de beschikking hebben over een logaritmentabel voor de basis. Dus, bij de priemgetallen p_1 tot en met p_b zijn getallen t_1 tot en met t_b bekend, zo dat $p_i = g^{t_i}$. De logaritme van y is te berekenen uit één relatie, en wel als volgt.

Een random getal z wordt gekozen door $z = y \circ g^k$, waar k een random getal is; merk op dat nu geldt $z = g^{x+k}$ voor de onbekende (maar begeerde) x en een bekende k. Veronderstel dat z factoriseert, dwz. $z = \prod_j p_j^{e_j}$. Omdat $p_j = g^{t_j}$ betekent dit $g^{x+k} = g^{\sum_j e_j \cdot t_j}$ en dus $x + k = \sum_j e_j \cdot t_j$ waaruit $x = \sum_j e_j \cdot t_j - k$ direct kan worden berekend.

Het verwachte aantal getallen $z = y \circ g^k$ dat men moet proberen om een relatie te vinden is in de grootteorde van tienmiljoen (zie boven). Met een snelle pc is dit goed te realiseren.

15.4.3 Berekening van de basistabel

De basistabel met logaritmen wordt berekend met relaties van de vorm $z = g^k$ voor willekeurige k. Immers, een factorisatie van g^k in $\prod_j p_j^{e_j}$ geeft een lineaire vergelijking in t_j, namelijk $k = \sum_j e_j \cdot t_j$. Hadden we bij het vinden van $\log y$ slechts één onbekende (namelijk x, omdat de t_j daar bekend werden verondersteld), nu zijn er b onbekenden, namelijk de logaritmen t_1 t/m t_b.

Om de t_j op te kunnen lossen moeten we daarom over ten minste b relaties beschikken. Een complicatie waar we hier verder niet diep op ingaan is dat het stelsel moet worden opgelost in $\mathbb{Z}_{p-1}$. Dit is alleen mogelijk als de determinant van het stelsel vergelijkingen inverteerbaar is in $\mathbb{Z}_{p-1}$. Daarom wordt het zoeken naar relaties nog iets verder voortgezet, bijvoorbeeld tot er $b + 20$ gevonden zijn. Door twintig willekeurige vergelijkingen weg te laten ontstaat een $b \times b$ matrix, en de kans is erg groot dat minstens één van deze stelsels oplosbaar is.

Het oplossen van een lineair stelsel lijkt een routineklus, maar gezien het aantal variabelen (enige miljoenen) is dat het niet. In de praktijk kost het zoeken naar relaties het meeste werk, maar dit is heel goed te parallelliseren. Het oplossen van het stelsel is een rekenintensieve klus die op een supercomputer wordt uitgevoerd.

15.4.4 Rekenvoorbeeld voor indexcalculus

Als voorbeeld nemen we vrij kleine getallen, namelijk $p = 83$ en $g = 8$. De basis B bestaat uit de vier priemgetallen 2, 3, 5 en 7, met vooralsnog onbekende logaritmen t_1, t_2, t_3 en t_4. We zullen de logaritme van $y = 47$ berekenen.

Berekening van de basistabel. Voor het berekenen van de logaritmentabel zoeken we naar relaties van de vorm $z = g^k$. Voor random k (eerste kolom) wordt steeds 8^k berekend (tweede kolom) en als een relatie wordt gevonden (derde kolom) levert dit een lineaire vergelijking in t_1 tot en met t_4 (vierde kolom).

k	g^k	Relatie	Vergelijking
1	8	2^3	$3 \cdot t_1 = 1$
17	55	Geen	
38	77	Geen	
47	53	Geen	
39	35	$5 \cdot 7$	$t_3 + t_4 = 39$
11	71	Geen	
45	54	$2 \cdot 3^3$	$t_1 + 3 \cdot t_2 = 45$
7	74	Geen	
63	22	Geen	
9	5	5	$t_3 = 9$

Na tien pogingen beschikken we over vier relaties en daarmee over vier lineaire vergelijken in t_1 tot en met t_4, in matrixvorm geschreven als

$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 39 \\ 45 \\ 9 \end{pmatrix}$$

De determinant van deze matrix is 9 en $9 \in \mathbb{Z}_{82}^*$ zodat de matrix inverteerbaar is over $\mathbb{Z}_{82}$. Het stelsel is dus oplosbaar en de oplossing geeft de volgende logaritmentabel:

i	1	2	3	4
p_i	2	3	5	7
t_i	55	24	9	30

Het berekenen van een logaritme. Om $\log y$ voor $y = 47$ te berekenen zoeken we relaties van de vorm $z = y \cdot g^k$. Weer geeft de eerste kolom k, de tweede geeft $47 \cdot g^k$ en de derde geeft een factorisering van z. In dit geval kunnen we stoppen zodra we over de eerste relatie beschikken.

k	$y \cdot g^k$	Relatie	Vergelijking
3	77	Geen	
8	19	Geen	
10	54	$2 \cdot 3^3$	$x + 10 = t_1 + 3 \cdot t_2$
			$\Rightarrow x = 55 + 3 \cdot 24 - 10 = 35$

Omdat voor het berekenen van een logaritme slechts één, maar voor het opzetten van de tabel ruim b relaties nodig zijn, is het berekenen van de tabel verreweg het duurste deel van de berekening. Het vinden van de vele relaties is veel werk maar goed parallelliseerbaar, het inverteren van de matrix is in de praktijk de lastigste klus. Wel kan de tabel, als hij eenmaal is berekend, voor altijd worden gebruikt om alle logaritmen in $\mathbb{Z}_p^*$ te berekenen.

15.4.5 Relaties voor factoriseren

Relaties spelen een belangrijke rol in moderne factorisatiealgoritmen. We willen nu kort aangeven hoe relaties voor factoriseren kunnen worden gebruikt, en hiermee verklaren waarom logaritmen in $\mathbb{Z}_p^*$ ongeveer zo moeilijk zijn is als het factoriseren van een getal met evenveel bits als p.

Een getal n kan worden gefactoriseerd zodra er twee getallen x en $y \neq \pm x$ zijn gevonden met $x^2 = y^2$ (zie sectie 5.5.2). Geprobeerd wordt relaties te verkrijgen van de vorm $z = k^2$, waar k een random getal is, wat dan leidt tot een uitdrukking van de vorm $k^2 = \prod_j p_j^{e_j}$.

Een uitdrukking van deze vorm is bijzonder bruikbaar wanneer alle e_j even zijn, zeg $e_j = 2e'_j$. In dat geval schrijft men namelijk:

$$k^2 = \prod_j p_j^{2e'_j} = \left(\prod_j p_j^{e'_j} \right)^2$$

Helaas is de kans dat in een gevonden relatie alle exponenten even zijn bijzonder klein.

Nu kunnen relaties met elkaar worden vermenigvuldigd: als z_1 en z_2 gefactoriseerd zijn over B, heeft het product $z_1 \cdot z_2$ ook een factorisatie over B. De exponenten worden gevonden door de overeenkomende exponenten in de twee factorisaties op te tellen. De kunst is nu om uit een flink aantal relaties een deelverzameling $k_1^2, \ldots, k_r^2$ te kiezen zo, dat als de exponenten van de factorisaties over B worden opgeteld, alle exponenten even zijn. Er geldt dan een relatie

$$(k_1 \cdot \ldots \cdot k_r)^2 = \prod_j p_j^{2e'_j} = \left(\prod_j p_j^{e'_j} \right)^2$$

Het vinden van zo'n deelverzameling komt neer op het inverteren van de matrix met vergelijkingen over $\mathbb{Z}_2$ en om een goede kans op succes te hebben moet het aantal relaties weer ruim b zijn. Net als het berekenen van een logaritmetabel vereist factorisatie dus het vinden van ruim b relaties en een matrixberekening. Belangrijk verschil is, dat de matrixberekening hier over $\mathbb{Z}_2$ is, en voor de indexcalculus over $\mathbb{Z}_{p-1}$.

15.4.6 Uitbreiding en verbetering van de indexcalculus

Deze sectie beschrijft de basis van de indexcalculus, maar om daadwerkelijk voor 512-bits-moduli bruikbaar te zijn moeten er nog diverse verbeteringen worden doorgevoerd, waarvan we er kort nog enkele noemen.

De getallen z waarvoor een factorisering wordt geprobeerd worden niet helemaal willekeurig gekozen, maar juist op zo'n manier dat ze klein zijn ten opzichte van p. Onder kleinere getallen is een grotere fractie p_b-glad, zodat deze verbetering het verwachte aantal pogingen voor het vinden van een relatie omlaag brengt.

Verder worden de achtereenvolgende z niet onafhankelijk gekozen, maar als elementen van een rij. Hierdoor kan het werk voor het factoriseren worden verminderd.

Het principe van de indexcalculus kan ook in andere groepen worden toegepast. De indexcalculus gebruikt dat getallen $z \in \mathbb{Z}_p^*$ kunnen worden geïnterpreteerd als getallen in $\mathbb{Z}$, waar een unieke factorisatie over priemgetallen geldt. Een eindig lichaam $GF(p^n)$ bestaat uit polynomen over $\mathbb{Z}_p$, gerekend modulo een vast irreducibel polynoom. Omdat polynomen een unieke factorisatie hebben, kan de indexcalculus worden toegepast in alle eindige lichamen $GF(p^n)$.

Op elliptische krommen of XTR is de indexcalculus niet toepasbaar, en dit verklaart waarom deze systemen met kortere getallen kunnen werken en toch hetzelfde niveau van veiligheid bieden.

In 2015 wezen Adrian e.a. [ABD+15] op een gevaar vanwege het delen van moduli in Elgamal en Diffie-Hellman. Weliswaar is het berekenen van *een* (Diffie-Hellman) logaritme duurder dan het factoriseren van *een* (RSA) modulus, vanwege de matrix-inversie over $\mathbb{Z}_{p-1}$ ipv $\mathbb{Z}_2$. Maar bij RSA heeft elke gebruiker zijn eigen modulus, terwijl bij Diffie-Hellman de g en p worden gedeeld (zie bv de Oakley Group 2, een populaire 1024-bit modulus). Na de initiële kosten voor het berekenen van de tabel, kunnen dus meerdere gebruikers tegen lage kosten worden aangevallen. Geschat wordt dat voor een modulus van 1024 bits, met een hardware investering in de orde van honderden miljoenen dollars, na een tabelberekening in de orde van een à twee jaar, een logaritme in ongeveer een maand kan worden berekend.

Overigens heeft het NIST al rond 2011 de aanbeveling gedaan om geen 1024-bits moduli meer te gebruiken. RSA sleutels moeten minstens 2048 bits lang zijn, en Diffie-Hellman moet ook gebaseerd zijn op 2048-bits moduli, of op elliptische krommen.

Samenvatting en conclusies

Er bestaat een verscheidenheid aan cryptografische technieken die zijn gebaseerd op het *discrete logaritme* probleem. Naast het ElGamal-algoritme en andere technieken die werken met exponentiatie in $\mathbb{Z}_p^*$ zijn dit bijvoorbeeld de elliptische-krommencryptografie en het recente XTR. Dit hoofdstuk laat met de wiskundige abstractie van een *groep* zien wat de algemene achterliggende structuur van deze systemen is.

Vervolgens is gekeken naar algoritmen die de discrete logaritme kunnen uitrekenen. Deze algoritmen kunnen voor een aanval op de genoemde cryptografische technieken worden gebruikt. Men moet er bij de keuze van parameters en de implementatie van het systeem dus altijd voor zorgen dat deze algoritmen niet in realiseerbare tijd door een aanvaller kunnen worden toegepast.

Bij deze algoritmen onderscheiden we *groepsalgoritmen* en *speciale algoritmen*.

Groepsalgoritmen. Deze gebruiken alleen de gemeenschappelijke groepsstructuur van de cryptografische systemen. Hun kracht is daarom, dat ze tegen alle cryptografische algoritmen kunnen worden ingezet, ongeacht de gebruikte onderliggende groep. Gelukkig gebruikt het best mogelijke groepsalgoritme $\sqrt{q}$ tijd, als q de grootte van de groep is en bovendien priem. Een 80-bitsveiligheidsniveau tegen aanvallen met groepsalgoritmen wordt dus verkregen door q als 160-bitspriemgetal te kiezen.

Speciale algoritmen. Deze gebruiken niet alleen de groepsstructuur maar ook verdere eigenschappen van de gebruikte getallen. Een groepsalgoritme kan daarom slechts tegen één systeem of een beperkte klasse van systemen worden ingezet. Het meest succesvolle is de indexcalculus met zijn opvolger, de Number Field Sieve, waarmee anno 2015, of snel daarna, moduli van ongeveer 1024 bits konden worden aangevallen.

Opgaven bij hoofdstuk 15

Opgave 15.1 *Wat is de kleinste groep? Wat is de kleinste niet-commutatieve groep?*

Opgave 15.2 *Zij $F = \{A, B, C, D\}$ met operatie $\bullet$ volgens deze tabel.*

(a) Controleer of dit een groep is.
(b) Wat is de inverse van elk element?
(c) Wat is de orde van elk element?
(d) Is de groep cyclisch?
(f) Welke subgroep(en) van grootte 2 heeft F?
(g) Bewijs dat $F \cong \mathbb{Z}_4$.

$\bullet$	A	B	C	D
A	A	B	C	D
B	B	A	D	C
C	C	D	B	A
D	D	C	A	B

Opgave 15.3 *Bewijs dat in een groep voor elk element b geldt: $(b^{-1})^{-1} = b$.*

Opgave 15.4 *Geef een efficiënt groepsalgoritme dat van een gegeven element a de orde bepaalt. Je algoritme mag kennis van $\#G$ en de factorisatie ervan gebruiken.*
*(***Hint:** *doorzoek op een systematische manier alle delers van $\#G$.)*
Kun je het ook zonder kennis van $\#G$?

Opgave 15.5 *Het RSA-algoritme rekent in een multiplicatieve groep $\mathbb{Z}_n^*$. Kun je het algoritme uit opgave 15.4 gebruiken om de orde van een bericht x in die groep uit te rekenen? Waarom wel/niet?*

Opgave 15.6 De Discrete Log. *(Uit het tentamen van januari 1997.)*
Leg uit wat p, α, en β zijn in de uitdrukking "$\mathrm{ind}_\alpha(\beta)$ in $\mathbb{Z}_p$". Ken je methoden om de index te berekenen en hoe efficient zijn die?

Iemand beweert dat je kunt factoriseren als je discrete logaritmen kunt uitrekenen, immers het vinden van wortels leidt tot factorisatie. Zij n een groot getal en neem een willekeurige α en x. Neem $y = x^2$, $l = \mathrm{ind}_\alpha(y)$ en $z = \alpha^{l/2}$. Nu is $x^2 = z^2$, en met 50% kans is $\mathrm{ggd}(n, x + z)$ een factor van n. Is dit zo, en waarom wel/niet?

Opgave 15.7 *Algoritme 15.5 begint aan een nieuwe iteratie als de gekozen a voldoet aan $a^r = 1$. Bewijs dat $\{a \in G : a^r = 1\}$ een subgroep van G is. Bewijs dat het verwachte aantal iteraties ten hoogste 2 is.*
*(***Hint:** *voor een subgroep H van G geldt $\#H|\#G$.)*

Opgave 15.8 *Wat kan er misgaan in algoritme 15.5 als $q|r$?*

Opgave 15.9 *Bewijs dat een cyclische groep G precies $\phi(\#G)$ voortbrengers heeft.*

Opgave 15.10 *Welke lineaire expressies F_i (in de notatie van sectie 15.3.4) worden gevormd door het algoritme van Shanks? Laat zien dat er geen botsing optreedt tussen de F_i (dat wil zeggen, er is zijn geen i, j met $F_i = F_j$), maar dat er, gegeven een toekenning x aan X_1, wel een botsing tussen de a_i optreedt binnen een begrensd aantal stappen.*

Opgave 15.11 *Breid de redenering in sectie 15.3.4 uit voor de aanname dat A in één stap een exponentiatie kan uitvoeren, dat wil zeggen, voor a_k een element a_i^l kan vragen. Hoe wordt dan de expressie F_k opgebouwd? Bewijs dat exponentiëren niet helpt om de logaritme sneller te kunnen berekenen.*

Opgave 15.12 *Breid de redenering in sectie 15.3.4 uit voor de aanname dat A beschikt over een* Diffie-Hellman-orakel*, dat wil zeggen, in één stap kan vragen om een a_k waarvoor $DifHel(a_i, a_j, a_k)$. Hoe wordt de expressie F_k opgebouwd? Kun je nog steeds bewijzen dat $\sqrt{q}$ stappen nodig zijn?*

Bijlage A

Kansrekening

Deze bijlage geeft een kort overzicht van enkele van de belangrijkste resultaten uit de kansrekening die in dit boek worden gebruikt.

A.1 Experimenten en verwachting

Een handeling die een willekeurige keuze uit een aantal mogelijke uitkomsten bepaalt heet in de kansrekening een *experiment*. Voorbeelden van experimenten zijn het werpen met een munt (uitkomsten: 'kop' en 'munt') of een dobbelsteen (uitkomsten: 1, 2, 3, 4, 5 en 6). Ook het schudden van kaarten of n andere objecten wordt als experiment beschouwd, waarbij elk van de 52!, respectievelijk $n!$ permutaties een mogelijke uitkomst is. In de cryptografie is het vaak nodig een random sleutel of random getal te trekken en ook deze trekking is een experiment.

Van deze elementaire experimenten, waarbij het aantal mogelijke uitkomsten eindig is, wordt vaak aangenomen dat ze *uniform* zijn, dat wil zeggen, dat elk van de uitkomsten gelijke kans heeft om voor te komen. Voor de dobbelsteen betekent dit, dat elk van de zes uitkomsten een kans van $\frac{1}{6}$ heeft om voor te komen.

De kansrekening spreekt van een *samengesteld experiment* wanneer de uitkomst bepaald wordt door een aantal deelexperimenten. Hierbij wordt steeds aangenomen dat de uitkomst van een deelexperiment niet van invloed is op de uitkomst van een ander experiment. Voorbeelden van samengestelde experimenten zijn het meermalen werpen van een munt of het gooien met meerdere dobbelstenen.

Waarom gooit men met twee dobbelstenen vaker 7 dan 9? De uitkomst 7 kan op zes manieren worden gevormd (namelijk, 1-6, 2-5, 3-4, 4-3, 5-2, 6-1) en de uitkomst 9 kan op vier manieren worden gevormd (namelijk 3-6, 4-5, 5-4, 6-3). Daarom is de kans op 7 gelijk aan $\frac{6}{36}$ en de kans op 9 gelijk aan $\frac{4}{36}$.

De kansverdeling op uitkomsten van een samengesteld experiment kan ingewikkeld zijn wanneer de uitkomst van deelexperimenten bepalend is voor het vervolg van het experiment. Men kan zich bijvoorbeeld het volgende experiment voorstellen: eerst wordt met een munt geworpen, en is de uitkomst 'munt' (M) dan wordt nogmaals met de munt geworpen, maar bij uitkomst 'kop' wordt met een dobbelsteen geworpen. Het samengestelde experiment heeft acht uitkomsten, te noteren als MM, MK, K1, K2, K3, K4, K5 en K6, maar de kansverdeling is niet uniform over deze uitkomsten. De kans dat de eerste muntworp M geeft is exact $\frac{1}{2}$, en in dat geval geeft de volgende muntworp weer even grote kansen op M en K, daarom is

worp	mogelijkheden	kans
2	1-1	$\frac{1}{36}$
3	1-2, 2-1	$\frac{2}{36}$
4	1-3, 2-2, 3-1	$\frac{3}{36}$
5	1-4, 2-3, 3-2, 4-1	$\frac{4}{36}$
6	1-5, 2-4, 3-3, 4-2, 5-1	$\frac{5}{36}$
7	1-6, 2-5, 3-4, 4-3, 5-2, 6-1	$\frac{6}{36}$
8	2-6, 3-5, 4-4, 5-3, 6-2	$\frac{5}{36}$
9	3-6, 4-5, 5-4, 6-3	$\frac{4}{36}$
10	4-6, 5-5, 6-4	$\frac{3}{36}$
11	5-6, 6-5	$\frac{2}{36}$
12	6-6	$\frac{1}{36}$

Tabel A.1: Kansverdeling bij het werpen van twee dobbelstenen

$\Pr[\text{MM}] = \Pr[\text{MK}] = \frac{1}{4}$. Verder is $\Pr[\text{K1}] = \Pr[\text{K2}] = \Pr[\text{K3}] = \Pr[\text{K4}] = \Pr[\text{K5}] = \Pr[\text{K6}] = \frac{1}{12}$.

Vaak is aan de uitkomst van een experiment een getalswaarde verbonden, bijvoorbeeld de kosten of juist opbrengst die met de uitkomst is gemoeid. Zo kan men bij een worp met een dobbelsteen kijken naar het aantal ogen, of bij twee dobbelstenen naar het totaal aantal ogen. Formeel is een *stochastische variabele*, of *stochast*, een afbeelding van de verzameling uitkomsten naar getallen. We kijken alleen naar de situatie waarin de stochast alleen gehele waarden (uit $\mathbb{Z}$) aanneemt.

Definitie A.1 *De* verwachte waarde *van een stochast X, notatie* $\mathrm{E}[X]$*, is* $\sum_{i\in\mathbb{Z}} i \cdot \Pr[X = i]$.

De verwachte waarde van een stochast is te zien als een 'afkoopwaarde' van een experiment. Stel dat Alice een lot heeft, waarbij zij eenmaal met een dobbelsteen mag gooien en dan het aantal ogen in euro uitbetaald krijgt. Bob vraagt het lot van Alice te mogen kopen voor 2,50 euro; is het verstandig voor Alice om hierop in te gaan? Om deze vraag te beantwoorden bepalen we eerst de verwachte opbrengst van het experiment. Met een kans van elk eenzesde wint Alice 1, 2, 3, 4, 5 of 6 euro en volgens de definitie is de verwachte opbrengst dan

$$1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} = 3\frac{1}{2}.$$

Merk allereerst op, dat de verwachting van een geheeltallige stochast zelf niet geheeltallig hoeft te zijn. Als Alice het experiment een groot aantal malen zou herhalen, zou haar opbrengst gemiddeld ongeveer 3,50 euro per experiment bedragen. Men moet bedenken, dat de opbrengst van een enkel experiment heel goed hoger of lager dan de verwachting kan zijn. Alice' beslissing om het lot te verkopen zal er sterk van afhangen of Alice graag gokt of juist van zekerheid houdt; maar als zij verkoopt is 3,50 euro een redelijke prijs.

Het volgende voorbeeld illustreert dat verwachting *additief* is: als een stochast X is te schrijven als $X = X_1 + X_2$, dan is $\mathrm{E}[X] = \mathrm{E}[X_1] + \mathrm{E}[X_2]$. We werpen met een rode en een

blauwe dobbelsteen en de stochast X is het totale aantal ogen. De kansverdeling van X is weergegeven in tabel A.1 en het verwachte aantal ogen wordt berekend als

$$\mathrm{E}[X] = 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + \ldots 11 \cdot \frac{2}{36} + 12 \cdot \frac{1}{36} = 7.$$

Nu schrijven we X_1, respectievelijk X_2, voor het aantal ogen van de rode, respectievelijk bauwe dobbelsteen; er geldt $X = X_1 + X_2$. De verwachting van het aantal ogen van één dobbelsteen werd in de vorige alinea berekend en is $3\frac{1}{2}$; er geldt dus $\mathrm{E}[X_1] = \mathrm{E}[X_2] = 3\frac{1}{2}$. Er geldt hier dus $\mathrm{E}[X] = \mathrm{E}[X_1] + \mathrm{E}[X_2]$.

Verwachting is additief, ook als X de som is van meer dan twee stochasten, en het is niet vereist dat de stochasten onafhankelijk van elkaar zijn.

Feit A.2 *Als* $X = \sum_i X_i$ *dan is* $\mathrm{E}[X] = \sum_i \mathrm{E}[X_i]$.

A.2 Bernoullireeksen

We kijken naar een elementair experiment, nu poging genaamd, met twee uitkomsten, namelijk 'succes' (S) en 'mislukking' (M), waarbij de kans op succes p, en de kans op mislukking $1-p$ is. Hieruit is een samengesteld experiment af te leiden, namelijk het doen van meerdere pogingen; dit samengestelde experiment heet een *Bernoullireeks* (met succeskans p). De volgende drie vragen worden in deze sectie behandeld: (1) wat is de kans op minstens een succes in een bepaald aantal pogingen; (2) wat is het verwachte aantal successen in een bepaald aantal pogingen; (3) hoeveel pogingen zijn verwacht nodig tot het eerste succes?

Voor een enkele poging definiëren we de stochast X_0 met waarde 1 bij uitkomst succes en waarde 0 bij uitkomst mislukking. De verwachting van X_0 is $0 \cdot (1-p) + 1 \cdot p = p$.

Voor de eerste twee stellingen nemen we aan dat het aantal pogingen vast is, dus niet afhangt van het resultaat van pogingen. Hoe groot is de kans op ten minste één succes?

Stelling A.3 *De kans op ten minste één succes in een Bernoullireeks met succeskans* p *en lengte* k *is* $1 - (1-p)^k$.

Bewijs. Er is sprake van ten minste een succes als de reeks niet uit alleen mislukkingen bestaat. Omdat de kans op mislukking van een poging $1 - p$ is en de achtereenvolgende pogingen onafhankelijk zijn, is de kans op k achtereenvolgende mislukkingen $(1-p)^k$. De kans dat een reeks van lengte k niet bestaat uit k achtereenvolgende mislukkingen is dus $1-(1-p)^k$. □

Stelling A.4 *Het verwachte aantal successen in een Bernoullireeks met succeskans* p *en lengte* k *is* $k \cdot p$.

Bewijs. Het resultaat is af te leiden uit de additiviteit van verwachting (feit A.2). Zij A de stochast die het aantal successen aangeeft. Definieer de stochast X_i als 1 indien poging i slaagt en 0 als poging i mislukt; dan is de stochast A gelijk aan $\sum_{i=1}^{k} X_i$. De verwachting van X_i is hierboven berekend als $\mathrm{E}[X_i] = p$, dus is

$$\mathrm{E}[A] = \mathrm{E}[\sum_{i=1}^{k} X_i] = \sum_{i=1}^{k} \mathrm{E}[X_i] = \sum_{i=1}^{k} p = k \cdot p.$$

De verwachting kan ook rechtstreeks berekend worden als de kans op j successen bekend is. De kans op een bepaalde reeks, bijvoorbeeld $SSMSMMM$, is te vinden door per poging de kans op succes, dan wel mislukking, te nemen en te vermenigvuldigen:

$$\Pr[SSMSMMM] = p \cdot p \cdot (1-p) \cdot p \cdot (1-p) \cdot (1-p) \cdot (1-p) = p^3 \cdot (1-p)^4.$$

De kans op een specifieke reeks met j successen en $k-j$ mislukkingen is dus gelijk aan $p^j \cdot (1-p)^{k-j}$. Nu zijn er meerdere reeksen met dit aantal successen, namelijk, voor elke keuze van j plaatsen uit k is er precies één zo'n reeks. Het aantal reeksen is dus $\left(\begin{array}{c} k \\ j \end{array} \right)$, en omdat elk van deze reeksen kans $p^j \cdot (1-p)^{k-j}$ heeft om voor te komen is

$$\Pr[X = j] = \left(\begin{array}{c} k \\ j \end{array} \right) \cdot p^j \cdot (1-p)^{k-j}.$$

Nu kan de verwachting van A volgens definitie A.1 worden uitgedrukt als

$$\mathrm{E}[A] = \sum_{j=0}^{k} \; j \cdot \left(\begin{array}{c} k \\ j \end{array} \right) \cdot p^j \cdot (1-p)^{k-j}.$$

Omdat $\left(\begin{array}{c} k \\ j \end{array} \right) = \frac{k}{j} \left(\begin{array}{c} k-1 \\ j-1 \end{array} \right)$ geldt nu

$$\begin{array}{rcll}
\mathrm{E}[A] & = & \displaystyle\sum_{j=1}^{k} \; j \cdot \left(\begin{array}{c} k \\ j \end{array} \right) \cdot p^j \cdot (1-p)^{k-j} & \text{term } j=0 \text{ draagt niet bij} \\
& = & \displaystyle\sum_{j=1}^{k} \; j \cdot \frac{k}{j} \left(\begin{array}{c} k-1 \\ j-1 \end{array} \right) \cdot p \cdot p^{j-1} \cdot (1-p)^{k-j} & \text{factoren apart zetten} \\
& = & \displaystyle k \cdot p \cdot \sum_{j=1}^{k} \left(\begin{array}{c} k-1 \\ j-1 \end{array} \right) \cdot p^{j-1} \cdot (1-p)^{k-j} & \text{haal } k \cdot p \text{ buiten de som} \\
& = & \displaystyle k \cdot p \sum_{j'=0}^{k-1} \left(\begin{array}{c} k-1 \\ j' \end{array} \right) \cdot p^{j'} \cdot (1-p)^{k-1-j'} & \text{substitueer } j' \text{ voor } j-1 \\
& = & k \cdot p \cdot (p + (1-p))^{k-1} & \text{Newtons binomiumregel} \\
& = & k \cdot p &
\end{array}$$

□

Vaak ligt in een Bernoullireeks het aantal pogingen niet vantevoren vast, maar worden de pogingen net zo lang herhaald tot succes is behaald (zie bijvoorbeeld kader 14.5).

Stelling A.5 *De verwachte lengte van een Bernoullireeks met succeskans p die na het eerste succes wordt beëindigd is $1/p$.*

Bewijs. De stochast A is de lengte van de reeks. We bepalen eerst $\Pr[A = j]$. De reeks heeft lengte j als er eerst $(j-1)$-maal een mislukking optreedt en dan succes. De kans hierop is

$(1-p)^{j-1} \cdot p$. Vervolgens geldt

$$\begin{aligned}
\mathrm{E}[A] &= \sum_{j=1}^{\infty} j \cdot \Pr[A=j] && \text{gebruik definitie A.1} \\
&= \sum_{j=1}^{\infty} j \cdot (1-p)^{j-1} \cdot p && \text{invullen van de kans} \\
&= \tfrac{p}{1-p} \cdot \sum_{j=1}^{\infty} j \cdot (1-p)^{j} && \text{breng factor } p/(1-p) \text{ naar buiten} \\
&= \tfrac{p}{1-p} \cdot \tfrac{1-p}{p^2} && \text{uitkomst van reeks} \\
&= \tfrac{1}{p}
\end{aligned}$$

De vierde stap gebruikt de gelijkheid $\sum_{j=1}^{\infty} j \cdot x^j = \frac{x}{(1-x)^2}$. □

A.3 Botsingen en de verjaardagsstelling

Ook in deze sectie worden reeksen van herhaalde experimenten bekeken. In de vorige sectie werd gewacht op het voorkomen van een bepaalde uitkomst (daar 'succes' genoemd), nu wordt gekeken naar het herhaald voorkomen van een uitkomst, een *botsing*. Het elementaire experiment is een uniforme trekking uit n uitkomsten.

De eerste verjaardagsstelling vormt één reeks van experimenten en berekent het aantal botsingen binnen die reeks.

Stelling A.6 (Eerste verjaardagsstelling) *Zij B het aantal botsingen in een reeks van k uniforme trekkingen uit n uitkomsten; dan is*

$$\mathrm{E}[B] = \frac{k(k-1)}{2 \cdot n} \approx \frac{k^2}{2n}.$$

Bewijs. Noem de uitkomst van het i-de experiment u_i en definieer voor $i < j$ de stochast A_{ij} als 1 als $u_i = u_j$ en 0 anders. Er geldt $B = \sum_{i,j} A_{ij}$ en dus $\mathrm{E}[B] = \sum_{i,j} \mathrm{E}[A_{ij}]$. Bij het j-de experiment is elk van de n uitkomsten even waarschijnlijk en onafhankelijk van de eerdere uitkomsten, dus $\Pr[u_i = u_j] = \frac{1}{n}$, en dus is $\mathrm{E}[A_{ij}] = \frac{1}{n}$.

Het aantal paren (i, j) dat voldoet aan $1 \leq i < j \leq k$ is $\frac{1}{2}k(k-1)$, en daarom is $\mathrm{E}[B] = \frac{k(k-1)}{2 \cdot n}$. □

Volgens deze stelling is het verwachte aantal botsingen groter dan één als k ongeveer $\sqrt{2n}$ of meer bedraagt. De kans op ten minste één botsing is groter dan een half als de lengte van de reeks $1 + 1,18\sqrt{n}$ is. De kans dat twee personen in een groep op dezelfde dag jarig zijn is groter dan een half wanneer de groep uit 23 of meer personen bestaat. Aan de analyse van dit verjaardagsprobleem ontlenen de stellingen uit deze sectie hun naam.

De tweede verjaardagsstelling betreft het aantal botsingen tussen twee afzonderlijk ontwikkelde reeksen.

Stelling A.7 (Tweede verjaardagsstelling) *Zij C het aantal botsingen tussen een reeks van k, en een reeks van l uniforme trekkingen uit n uitkomsten; dan is* $\mathrm{E}[C] = \frac{k \cdot l}{n}$.

Bewijs. Noem de uitkomst van het i-de experiment van de eerste reeks u_i, en van het j-de experiment van de tweede reeks v_j en definieer de stochast A_{ij} als 1 als $u_i = v_j$. Weer is de kans op deze botsing $\frac{1}{n}$ en dus $\mathrm{E}[A_{ij}] = \frac{1}{n}$. Het aantal paren (i, j) waarvoor $1 \leq i \leq k$ en $1 \leq j \leq l$ is $k \cdot l$ en omdat $C = \sum_{i,j} A_{ij}$ geldt $\mathrm{E}[C] = \frac{k \cdot l}{n}$. □

Het voorkomen van botsingen. Botsingen komen in een reeks (of twee reeksen) experimenten veel sneller voor dan een van te voren bepaalde uitkomst. Beschouwen we een bepaalde uitkomst als succes, dan is de kans daarop $\frac{1}{n}$ en de waarde komt verwacht eenmaal voor in een reeks van n experimenten; een botsing wordt al na ruwweg $\sqrt{n}$ experimenten verwacht. De verklaring hiervoor is, dat een reeks van k experimenten een veel groter aantal mogelijkheden voor botsingen creëert, en elk van die potentiële botsingen vindt plaats met kans $\frac{1}{n}$.

Bijlage B

Verantwoording van afbeeldingen

Auteur en uitgever hebben zich alle moeite getroost om de rechten voor de gebruikte afbeeldingen af te handelen. Zij die menen alsnog rechten te kunnen doen gelden, wordt verzocht contact op te nemen met de uitgever.

De volgende afbeeldingen zijn gemaakt door de auteur: 1.1 (in Piešt'any, Slowakije; de afgebeelde personen zijn (v.l.n.r.) Peter van Emde Boas, Vaughan R. Pratt, Jiří Wiedermann, Martin Beran), 1.2, 1.3, 5.1, 5.4, 6.4, 6.5, 7.5, 8.2 (token ter beschikking gesteld door SnS bank), 8.4 (de hand is van Sandra Verdonk), 11.4, 12.1.

Overige afbeeldingen:

2.6 Zie `http://www.eff.org/freespeech.html`, EFF website.

2.7 Zie `http://www.cryptography.com/des/despictures/index.html`, *Paul and the machine*.

2.10 Van de NSA-website `http://www.nsa.gov/about_nsa/`, *NSA Seal and Eagle*.

4.1 *Fort Knox kaart*, van PrimaCryption website `www.primacryption.com`.

4.5 Door Lieke Feddema.

4.6 Het Hodshon Huis in Haarlem, door Xander Verrijn Stuart.

5.6 Foto door Karen Aardal.

5.11 Door Lieke Feddema.

5.13 *Veilig uitgaan, een bekeken zaak*, Voorlichting van de Utrechtse politie, 2001.

6.1 Hellman: `http://www-ee.stanford.edu/ee/bitmaps/Hellman_Martin.gif`. Diffie: `http://www.sun.com/research/features/encryption/`. Fotograaf: Dennis Hayes, copyright Sun Microsystems.
6.6 Van `http://www.i.cz/onas/`, foto door Jan Polašek.
6.7 Van `http://web.mit.edu/prz/photos.shtml`: 'Phil R. Zimmermann in Red Square', met dank aan Computerworld Russia.
7.2 Van `www.identrus.com`.
7.4 Jim Davis, Garfield-pocket 31, Loeb uitgevers.
8.1 Bob Evers Serie deel 6, *Sensatie op een Engelse Vrachtboot*, Willy van der Heide (omslag).
8.6 Omslag van [DL98].
8.11 Overgenomen van Köhler [Köh99].
8.12 Het Wombat Midas toetsenbord, van `www.toetsenborden.nl`.
9.1 De foto, voorstellende de auteur, is gemaakt door René van Oostrum en bewerkt met een programma van Dirk Wegener.
9.2 Zie `http://jamesbond007.net/Ope/ope18.html`.
9.4 Zie `http://hotwired.lycos.com/netizen/96/43/index3a.html`. Uit *Crypto Imerialism* in *The Netizen*.
9.10 Door Branislav Rovan: de auteur stemt voor de SOFSEM-verkiezing in Tsjechië.
10.6 Zie `www.nokia.com`, press photo archive.
11.1 Ontleend aan `http://www.jjtc.com/Steganography/`.
11.2 Ontleend aan `http://theory.lcs.mit.edu/~rivest/`.
12.2 Zie `www.semiconductors.philips.com`, Philips SmartXA Family fact sheet.
12.3 Door Annet Roodenburg.
12.4 Overgenomen van Kocher e.a. [KJJ99]; copyright berust bij Springer-Verlag.
12.5 Bron: ADSR Ltd., Engeland.
13.1 Door Annet Roodenburg.
13.2 Uit *Philips 100*, uitgegeven bij het 100-jarig bestaan van Philips.
13.5 Van `http://www.mediamatic.nl/whoiswho/chaum/index.html`.
13.10 Bron: ADSR Ltd., Engeland.
14.2 Van `www.eucybervote.org`.
14.3 Zie `www.votehere.net`.
14.4 Persbericht van de gemeente Voorburg, uit de Metro van 17 oktober 2001.
14.5 Door Annet Roodenburg.
15.2 Zie `www-history.mcs.st-andrews.ac.uk/history/PictDisplay/Abel.html`. Fotograaf onbekend.
15.6 Van `www.xtr.com`, Fotograaf onbekend.
15.9 Door Annet Roodenburg.

Bibliografie

[ABD+15] Adrian, D., Bhargavan, K., Durumeric, Z., Gaudrey, P., Green, M., Halderman, J. A., Heninger, N., Springall, D., Thomé, E., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Béguelin, S., and Zimmermann, P. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In proc. *CCS15 (22nd ACM SIGSAC Conf. on Computer and Communications Security)* (2015), pp. 5–17. DOI: dx.doi.org/10.1145/2810103.2813707.

[AK96] Anderson, R. and Kuhn, M. Tamper resistance – a cautionary note. Zie URL `http://www.cl.cam.ac.uk/users/rja14/tamper.html`, 1996.

[And94] Anderson, R. J. Why cryptosystems fail. *Commun. ACM* **37**, 11 (1994), 32–40.

[BAN90] Burrows, M., Abadi, M., and Needham, R. A logic of authentication. *ACM Trans. on Computer Systems* **8**, 1 (1990), 18–36.

[Bau97] Bauer, F. L. *Decrypted Secrets: Methods and Maxims of Cryptology.* Springer-Verlag, 1997 (448 pp.).

[BB06] Barkan, E. and Biham, E. Slow adjacent control channel. In proc. *Selected Ares in Cryptography* (Berlin, 2006), B. Preneel and S. Tavares (eds.), vol. 3897 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–19.

[BBB+07] Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. *Recommendation for Key Management - Part 1: General (Revised).* NIST, 2007.

[BDK+09] Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., and Shamir, A. *Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds.* IACR, 2009.

[BDL97] Boneh, D., DeMillo, R. A., and Lipton, R. J. On the importance of checking cryptographic protocols for faults (extended abstract). In proc. *Advances in Cryptology - EuroCrypt '97* (Berlin, 1997), W. Fumy (ed.), Springer-Verlag, pp. 37–51. Lecture Notes in Computer Science Volume 1233.

[Beu98] Beukel, A. van den. *Geen beter leven dan een goed leven.* Ten Have, 1998.

[BGW99] Briceno, M., Goldberg, I., and Wagner, D. A pedagogical implementation of A5/1, 1999. `http://www.scard.org`.

[BKN09] Biryukov, A., Khovratovich, D., and Nikolic, I. *Distinguisher and Related-Key Attack on the Full AES-256 (Extended Version).* IACR, 2009.

[BL96] Boneh, D. and Lipton, R. J. Algorithms for black-box fields and their application to cryptography. In proc. *Advances in Cryptology - Crypto '96* (Berlin, 1996), N. Koblitz (ed.), Springer-Verlag, pp. 283–297. Lecture Notes in Computer Science Volume 1109.

[BM97] Bunten-Mines, E. Announcing request for candidate algorithm nominations for the advanced encryption standard. *Federal Register* **62**, 177 (1997). `http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm`.

[Boe99] Boer, J. de. Veilig telewerken met behulp van smartcards. Scriptie INF/SCR-99-03, Informatica Instituut, Universiteit Utrecht, 1999.

[BS90] Biham, E. and Shamir, A. Differential cryptanalysis of DES-like cryptosystems (extended abstract). In proc. *Advances in Cryptology - Crypto '90* (Berlin, 1990), A. J. Menezes and S. A. Vanstone (eds.), Springer-Verlag, pp. 2–21. Lecture Notes in Computer Science Volume 537.

[BS96] Bach, E. and Shallit, J. *Algorithmic Number Theory (Vol. 1: Efficient Algorithms)*. MIT Press, 1996.

[BS98] Biham, E. and Shulman, A. User-defined divisibility of ecash and a practical implementation. In Quisquater and Schneier [QS98], pp. 1–18.

[BSW00] Biryukov, A., Shamir, A., and Wagner, D. Real time cryptananlysis of A5/1 on a PC. In proc. *Fast Software Encryption* (Berlin, 2000), B. Schneier (ed.), vol. 1978 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–18.

[Cam99] Campbell, D. Interception capabilities 2000 (report to the director general for research of the european parliament), 1999. Zie de URL `http://www.cyber-rights.org`.

[CDD+01] Canetti, R., Darmgård, I., Dziembowski, S., Ishai, Y., and Malkin, T. On adaptive versus non-adaptive security of multiparty protocols. In Pfitzmann [Pfi01], pp. 262–279.

[CDN01] Cramer, R., Darmgård, I., and Nielsen, J. B. Multiparty computation from threshold homomorphic encryption. In Pfitzmann [Pfi01], pp. 280–300.

[CGS97] Cramer, R., Gennaro, R., and Schoenmakers, B. A secure and optimally efficient multi-authority election scheme. In proc. *Advances in Cryptology - EuroCrypt '97* (Berlin, 1997), W. Fumy (ed.), Springer-Verlag, pp. 103–118. Lecture Notes in Computer Science Volume 1233.

[Cha81] Chaum, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **24**, 2 (1981), 84–88.

[Cha82] Chaum, D. Verification by anonymous monitors (also known as silo watching). In proc. *Advances in Cryptography* (Santa Barbara, California, USA, 1982), A. Gersho (ed.), University of California, Santa Barbara, pp. 138–139.

[Cha88] Chaum, D. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In proc. *Advances in Cryptology - EuroCrypt '88* (Berlin, 1988), C. G. Günther (ed.), Springer-Verlag, pp. 177–182. Lecture Notes in Computer Science Volume 330.

[CHP91] Chaum, D., Heijst, E. van, and Pfitzmann, B. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In proc. *Advances in Cryptology - Crypto '91* (Berlin, 1991), J. Feigenbaum (ed.), Springer-Verlag, pp. 470–484. Lecture Notes in Computer Science Volume 576.

[CK93] Crépeau, C. and Kilian, J. Discreet solitary games. In proc. *Advances in Cryptology - Crypto '93* (Berlin, 1993), D. R. Stinson (ed.), Springer-Verlag, pp. 319–330. Lecture Notes in Computer Science Volume 773.

[CNS03] CNSS. National policy on the use of the advanced encryption standard (AES) to protect national security systems and national security information, 2003. `http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf`.

[Cop94] Coppersmith, D. The data encryption standard (DES) and its strength against attacks. *Journal of Research and Development* **38**, 3 (1994), 243–250. `http://www.research.ibm.com/journal/rd/383/coppersmith.pdf`.

[Cra99] Cramer, R. Introduction to secure computing. In proc. *Modern Cryptology in Theory and Practice* (Berlin, 1999), I. Darmgård (ed.), vol. 1561 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 16–62.

[CW92] Campbell, K. W. and Wiener, M. J. DES is not a group. In proc. *Advances in Cryptology - Crypto '92* (Berlin, 1992), E. F. Brickell (ed.), Springer-Verlag, pp. 512–520. Lecture Notes in Computer Science Volume 740.

[DeL84] DeLaurentis, J. M. A further weakness in the common modulus protocol for the RSA cryptosystem. *Cryptologia* **8** (1984), 253–259.

[DF89] Desmedt, Y. and Frankel, Y. Threshold cryptosystems. In proc. *Advances in Cryptology - Crypto '89* (Berlin, 1989), G. Brassard (ed.), Springer-Verlag, pp. 307–315. Lecture Notes in Computer Science Volume 435.

[DH76] Diffie, W. and Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **IT-22** (1976), 644–654.

[DH77a] Diffie, W. and Hellman, M. Exhaustive cryptanalysis and the NBS data ancryption standard. *Computer* (1977), 74–84.

[DH77b] Diffie, W. and Hellman, M. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer* **10**, 6 (1977), 74–84. `http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=1646525`.

[DKL+98] Dhem, J.-F., Koeune, F., Leroux, P.-A., Mestré, P., Quisquater, J.-J., and Willems, J.-L. A practical implementation of the timing attack. In Quisquater and Schneier [QS98], pp. 167–182.

[DL98] Diffie, W. and Landau, S. *Privacy on the Line: The Politics of Wiretapping and Encryption.* MIT Press, 1998 (346 pp.).

[DR99] Daemen, J. and Rijmen, V. AES proposal: Rijndael, 1999. Zie URL `http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf`.

[DR01] Daemen, J. and Rijmen, V. The wide trail design strategy. *Lecture Notes In Computer Science* **2260** (2001), 222–238.

[DR02] Daemen, J. and Rijmen, V. *The design of Rijndael: AES - the advanced encryption standard*. Springer-Verlag Telos, 2002.

[DRB02] Daemen, J., Rijmen, V., and Barreto, P. Rijndael: beyond the AES. *CryptoWorld* (2002).
`http://www.larc.usp.br/~pbarreto/BeyondAES.pdf`.

[Dwo01] Dworkin, M. Recommendation for block cipher modes of operation. NIST Special Publication 800-38A, Computer Security Resource Center, 2001.
`http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf`.

[EGL82] Even, S., Goldreich, O., and Lempel, A. A randomized protocol for signing contracts (extended abstract). In proc. *Advances in Cryptology: Proceedings of Crypto '82* (New York, USA, 1982), D. Chaum, R. L. Rivest, and A. T. Sherman (eds.), Plenum Publishing, pp. 205–210.

[EL83] Eier, R. and Lagger, H. Trapdoors in knapsack cryptosystems. In proc. *Cryptography - Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany* (Berlin, 1983), T. Beth (ed.), Springer-Verlag, pp. 316–324. Lecture Notes in Computer Science Volume 149.

[Ele98] Electronic Frontier Foundation. *Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design*. O'Reilly, 1998.

[ElG85] ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In proc. *Advances in Cryptology: Proceedings of Crypto '84* (Berlin, 1985), G. R. Blakley and D. Chaum (eds.), Springer-Verlag, pp. 10–18. Lecture Notes in Computer Science Volume 196.

[FFS88] Feige, U., Fiat, A., and Shamir, A. Zero knowledge proofs of identity. *Journal of Cryptology* **1** (1988), 77–94.

[FS86] Fiat, A. and Shamir, A. How to prove yourself: practical solutions to identification and signature problems. In proc. *Advances in Cryptology - Crypto '86* (Berlin, 1986), A. M. Odlyzko (ed.), Springer-Verlag, pp. 186–194. Lecture Notes in Computer Science Volume 263.

[FS03] Ferguson, N. and Schneier, B. *Practical Cryptography*. John Wiley & Sons, 2003 (56-57 pp.).

[Gar00] Garrett, P. *Making, Breaking Codes: An Introduction to Cryptology*. Prentice-Hall, 2000.

[GHP13] Gluck, Y., Harris, N., and Prado, A. BREACH: Reviving the CRIME attack. In proc. *Blackhat* (2013).

[GJ79] Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GP09] Gilbert, H. and Peyrin, T. Super-sbox cryptanalysis: Improved attacks for AES-like permutations. IACR, 2009.
`http://eprint.iacr.org/2009/531.pdf`.

[GQ88] Guillou, L. C. and Quisquater, J. J. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In proc. *Advances in Cryptology - EuroCrypt '88* (Berlin, 1988), C. G. Günther (ed.), Springer-Verlag, pp. 123–128. Lecture Notes in Computer Science Volume 330.

[Gue99] Guedj, D. *De Stelling van de Papegaai: Roman over de Geschiedenis van de Wiskunde.* Ambo, 1999.

[Gue08] Gueron, S. *Advanced Encryption Standard (AES) Instructions Set.* Intel Corporation, 2008.
`http://softwarecommunity.intel.com/isn/downloads/intelavx/AES-Instructions-Set_WP`

[Hel80] Hellman, M. E. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory* **IT-26**, 4 (1980), 401–406.

[HP98] Handschuh, H. and Paillier, P. Smart card crypto-coprocessors for public-key cryptography. In Quisquater and Schneier [QS98], pp. 372–379.

[Joh98] Johnson, T. R. *American Cryptology during the Cold War, 1945-1989. Book III: Retrenchment and Reform, 1972-1980.* 1998 (232 pp.).

[KJ98] Koops, B.-J. and Jong, H. de. De risico's van data recovery voor overheid en gebruikers. *Computerrecht*, 5 (1998), 222–227.

[KJJ99] Kocher, P., Jaffe, J., and Jun, B. Differential power analysis. In proc. *Advances in Cryptology – CRYPTO* (Berlin, 1999), M. Wiener (ed.), vol. 1666 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 388–397.

[KK74] Korteweg and Keesing. *Het Moderne Geldwezen, Deel II.* Noord-Hollandsch Uitgevers Maatschappij, 1974.

[KK99] Kömmerling, O. and Kuhn, M. G. Design principles for tamper-resistant smartcard processors. In proc. *USENIX Workshop on Smartcard Technology* (Chicago, 1999), S. Guthery and P. Honeyman (eds.).

[Knu98] Knuth, D. E. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms.* Addison-Wesley, 1998.

[Koe01] Koelman, K. J. De derde laag: Bescherming van technische voorzieningen. *Auteurs & Media* **1** (2001), 82–89.

[Koh04] Kohno, T. Attacking and repairing the winzip encryption scheme. In proc. *Proceedings of the 11th ACM conference on Computer and communications security* (2004), pp. 72–81.

[Köh99] Köhler, W. Kloppende vinger: Vingerafdrukherkenning is ideale biometrische identificatie. *NRC Handelsblad (W & O)* (20 november 1999).

[Koo98] Koops, B.-J. *The Crypto Controversy. A Key Conflict in the Information Society.* Kluwer Law International, 1998.

[Koo01] Koops, B.-J. Crypto law survey: Overview per country. Zie URL `cwis.kub.nl/ frw/people/koops/`, 2001.

[KR01] Klíma, V. and Rosa, T. Attack on private signature keys of the OpenPGP format, PGP programs and other applications compatible with OpenPGP. `http://www.i.cz/en/pdf/openPGP_attack_ENGvktr.pdf`, 2001.

[Kru02] Kruijswijk, L. B. Elektronisch stemmen is strijdig met de grondwet. *Automatiseringsgids* (2002), 12.

[Küh01] Kühn, U. Cryptanalysis of reduced-round MISTY. In Pfitzmann [Pfi01], pp. 325–339.

[Lan88] Landau, S. Zero knowledge and the department of defense. *Notices of the American Mathematical Society* **35**, 1 (1988), 5–12.

[LV00] Lenstra, A. K. and Verheul, E. R. The XTR public key system. In proc. *Advances in Cryptology – CRYPTO 2000* (2000), M. Bellare (ed.), vol. 1880 of *Lecture Notes in Computer Science*, pp. 1–19.

[MH81] Merkle, R. and Hellman, M. On the security of multiple encryption. *Commun. ACM* **24**, 7 (1981), 465–467.

[Mic92] Micali, S. Fair public-key cryptosystems. In proc. *Advances in Cryptology - Crypto '92* (Berlin, 1992), E. F. Brickell (ed.), Springer-Verlag, pp. 113–138. Lecture Notes in Computer Science Volume 740.

[Mil86] Miller, V. S. Use of elliptic curves in cryptography. In proc. *Advances in Cryptology - Crypto '85* (Berlin, 1986), H. C. Williams (ed.), Springer-Verlag, pp. 417–428. Lecture Notes in Computer Science Volume 218.

[MW96] Maurer, U. M. and Wolf, S. Diffie-Hellman oracles. In proc. *Advances in Cryptology - Crypto '96* (Berlin, 1996), N. Koblitz (ed.), Springer-Verlag, pp. 268–282. Lecture Notes in Computer Science Volume 1109.

[NBB+] Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., and Roback, E. Report on the development of the advanced encryption standard (AES). Tech. rep., National Institute of Standards and Technology. `http://csrc.nist.gov/archive/aes/round2/r2report.pdf`.

[NBD+] Nechvatal, J., Barker, E., Dodson, D., Dworkin, M., Foti, J., and Roback, E. Status report on the first round of the development of the advanced encryption standard. Tech. rep., National Institute of Standards and Technology. `http://csrc.nist.gov/archive/aes/round1/r1report.pdf`.

[NIS01] NIST. Specification for the advanced encryption standard (AES). NIST, 2001. `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[Ott01] Otterloo, S. van. A security analysis of Pretty Good Privacy. Master's thesis, Institute for Information and Computation Science, 2001.

[Ped91] Pedersen, T. P. Distributed provers with applications to undeniable signatures. In proc. *Advances in Cryptology - EuroCrypt '91* (Berlin, 1991), D. W. Davies (ed.), Springer-Verlag, pp. 221–242. Lecture Notes in Computer Science Volume 547.

[Pfi01] Pfitzmann, B. (ed.). *Advances in Cryptology – EUROCRYPT 2001* (Berlin, 2001), vol. 2045 of *Lecture Notes in Computer Science*, Springer-Verlag.

[QS98] Quisquater, J.-J. and Schneier, B. (eds.). *Smart Cart Research and Applications* (Berlin, 1998), vol. 1820 of *Lecture Notes in Computer Science*, Springer-Verlag.

[Riv98] Rivest, R. L. The case against regulating encryption technology. *Scientific American* (1998), 88–89.

[RSA78] Rivest, R., Shamir, A., and Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21** (1978), 120–126.

[SASR01] Sanchez-Avila, C. and Sanchez-Reillol, R. The rijndael block cipher (AES proposal) :a comparison with DES. In proc. *IEEE 35th International Carnahan Conference* (2001), pp. 229–234.

[Sch89] Schnorr, C. P. Efficient identification and signatures for smart cards. In proc. *Advances in Cryptology - Crypto '89* (Berlin, 1989), G. Brassard (ed.), Springer-Verlag, pp. 239–252. Lecture Notes in Computer Science Volume 435.

[Sch96] Schneier, B. *Applied Cryptography*, 2nd edition. Wiley, 1996.

[Sch97] Schoenmakers, B. Basic security of the ecash payment system. In proc. *State of the Art in Applied Cryptography, Course on Computer Security and Industrial Cryptography* (Berlin, 1997), B. Preneel and V. Rijmen (eds.), vol. 1528 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 338–352.

[Sch02] Schalekamp, R. Elektronisch stemmen is onafwendbaar. *Automatiseringsgids* (2002), 13.

[Sha46] Shannon, C. E. Communication theory of secrecy systems. Tech. rep., Bell Labs, 1946.
`http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf`.

[Sha78] Shanks, D. *Solved and Unsolved Problems in Number Theory*, 2nd edition. Chelsea Publishing Company, 1978.

[Sha79] Shamir, A. How to share a secret. *Commun. ACM* **22** (1979), 612–613.

[Sha00] Shamir, A. Protecting smart cards from passive power analysis with detached power supplies. In proc. *Cryptographic Hardware and Embedded Systems – CHES* (Berlin, 2000), Çetin K. Koç and C. Paar (eds.), vol. 1965 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 71–77.

[Sho97] Shoup, V. Lower bounds for discrete logarithms and related problems. In proc. *Advances in Cryptology - EuroCrypt '97* (Berlin, 1997), W. Fumy (ed.), Springer-Verlag, pp. 256–266. Lecture Notes in Computer Science Volume 1233.

[Sim84] Simmons, G. J. The subliminal channel and digital signatures. In proc. *Advances in Cryptology: Proceedings of EuroCrypt '84* (Berlin, 1984), T. Beth, N. Cot, and I. Ingemarsson (eds.), Springer-Verlag, pp. 364–378. Lecture Notes in Computer Science Volume 209.

[SL00] Sterbenz, A. and Lipp, P. Performance of the AES candidate algorithms in java. In proc. *AES Candidate Conference* (2000), pp. 161–165.

[SMR09] Saha, D., Mukhopadhyay, D., and RoyChowdhury, D. A diagonal fault attack on the advanced encryption standard. IACR, 2009. `http://eprint.iacr.org/2009/581.pdf`.

[Ste67] Stein, J. Computational problems associated with Racah algebra. *J. Comp. Phys.* **1** (1967), 397–405.

[Sti95] Stinson, D. R. *Cryptography: Theory and Practice.* CRC Press, 1995.

[Tan90] Tanenbaum, A. S. *Computernetwerken.* Prentice Hall/Academic Service, 1990.

[UMTSa] 3rd Generation Partnership Project. Specification of the 3gpp confidentiality and integrity algorithms: Document 2, KASUMI specification. Tech. rep., ETSI/Sage, 1999.

[UMTSb] 3rd Generation Partnership Project. Security threads and requirements. Tech. rep. 3G TS 21.133, 3GPP support office, Valbonne (Fr), 1999.

[UMTSc] 3rd Generation Partnership Project. Specification of the 3gpp confidentiality and integrity algorithms: Document 1, f8 and f9 specification. Tech. rep., ETSI/Sage, 2000.

[UMTSd] 3rd Generation Partnership Project. A guide to 3rd generation security. Tech. rep. 3G TR 33.900, 3GPP support office, Valbonne (Fr), 2000.

[UMTSe] 3rd Generation Partnership Project. Cryptographic algorithm requirements. Tech. rep. 3GPP TS 33.105, 3GPP support office, Valbonne (Fr), 2001.

[Ver01] Verheul, E. R. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In proc. *Advances in Cryptology – EUROCRYPT 2001* (2001), B. Pfitzmann (ed.), vol. 2045 of *Lecture Notes in Computer Science*, pp. 195–210.

[VT97] Verheul, E. R. and Tilborg, H. C. van. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In proc. *Advances in Cryptology – EuroCrypt 1997* (1997), W. Fumy (ed.), vol. 1233 of *Lecture Notes in Computer Science*, pp. 119–133.

[Wer92] Wernsdorf, R. The one-round functions of the DES generate the alternating group. In proc. *Advances in Cryptology - EuroCrypt '92* (Berlin, 1992), R. A. Rueppel (ed.), Springer-Verlag, pp. 99–112. Lecture Notes in Computer Science Volume 658.

[Yao82] Yao, A. C. Protocols for secure computations (extended abstract). In proc. *23 Foundations of Computation Theory* (1982), pp. 160–164.

Index

$GF(p^n)$, 302
$\mathbb{B}$, bitwaarden, 50
$\mathbb{C}$, 303
$\mathbb{N}$, natuurlijke getallen, 71
$\mathbb{Z}$, gehele getallen, 72
$\mathbb{Z}_m$, modulaire groep, 77
$\mathbb{Z}_m^*$, multiplicatieve groep, 79
2DES, 53
3DES, 28, 53, 146

A5/1-algoritme, 6
aandeel, 166
aandeelhouder, 166
aanval, 4
    brutekrachtaanval, 5
    cijfertekstaanval, 4
    gekozen-cijfertekst-, 108, 114
    gekozen-klaretekst-, 25, 94
    handtekeningschema, 121
    identificatie, 141
    Klima-Rosa, 111, 310
    op A5/1, 57
abacus, 98
Abel, Niels K., 293
Abelse groep, 292
acceptatie (identificatie), 148, 155
actieve aanval (identificatie), 141, 144
Advanced Encryption Standard (AES), 30, 65
    stroomaanval, 239
    tijdaanval, 238
affiene systeem, 3
aftappen, 6, 219
Aibo, 229
algebraïsche uitbreiding, 302
    $GF(p^n)$, 302
    $\mathbb{C}$ (voorbeeld), 303
algoritme
    ElGamal, 106
    Euclides, 72
    Indiaas machtsverheffen, 97
    Pohlig-Hellman, 307
    Pollard (rho), 305
    RSA, 108
    Shanks, 304
    Stein (ggd), 95
anonimiteit, 251
asymmetrische algoritmen, 2
authenticatie, 140, 214
    client (Kerberos), 193
    server (Kerberos), 194
    SSL, 211
    UMTS, 210
authenticatiecode, 125
authenticatieserver, 194
authenticiteit, 120, 130

BAN-logica, 198
BankGiroCentrale, 248
bastaardsleutel, 10
berekenbaar, 93
Bernoullireeks, 321
binair ggd-algoritme, 95
binding (blob), 133
bit, 3
blinde ontsleuteling, 137
    RSA, 137
blinde overdracht, 136, 271, 282
    keuze, 282
    random-keuze, 282
    random-succes, 282
blob, 132, 153, 156
block cipher, 5
blokfunctie, 131
blokversleuteling, 5
Bluetooth, 202
Blum-integer, 113
Bob Evers, 142
Bond, James Bond, 170
botsing, 323
botsingsvrij (hashfunctie), 130, 131
Brute Force Attack, 5
brutekrachtaanval, 5
    op DES, 22

Caesariaans systeem, 3
Capstone, 129

Carmichael-getal, 99
cellulair (telefonie), 201
certainty level (identificatie), 155
certificaat, 122
Certificate Authority, 122, 224
Certificate Revocation List, 122
challenge, 146, 149
    rol, 157
chartaal geld, 247, 248
Chaum, David, 254
    hashfunctie, 132
Chinese reststelling, 86, 103, 111, 113, 127, 299, 307
    foutinjectie, 239
chipknip, 233, 260
    anoniem, 252
    authenticatie, 146, 147
    sleutelbeheer, 62
Chosen Ciphertext Attack, 108
Chosen Plaintext Attack, 25
cijfertekst, 1
cijfertekstaanval, 4
    op DES, 24
cipher block chaining, 55
ciphertext, 1
Ciphertext-Only Attack, 4
client-serverarchitectuur, 194
Clipper, 129
coëfficiënt, 185
coalitie, 141
COCOM, 218
commoditeitsgeld, 245
commutatieve groep, 292
compressie, 11
computationele verhulling, 11, 167
computationele verifieerbaarheid, 168
concealing (blob), 133
confidentiality, 120
contractprobleem, 283
credit-debetbetaling, 249
creditbetaling, 249
creditcard, 248
creditering, 249
cryptanalyse, 218
cryptografisch sterk, 102
    encryptie, 114
    hashfunctie, 132
    verdeelschema, 177
    versleuteling, 113
cubic-rootaanval, 48
cut-and-choose, 135, 257
CyberVote, 276
cyclisch (groep), 82

Daemen, Joan, 65
Data Encryption Standard (DES), 15
    3DES, 53
    brutekrachtaanval, 22
    differentiële aanval, 25
    EFF-kraker, 22
    expansiefunctie, 18
    geschiedenis, 28
    iteratiesleutel, 17
    opvolger AES, 65
    S-box, 19, 20, 30
debetbetaling, 249
debitering, 249
decryptie, 1
Deep Crack, 22
DeLaurentis, 110
delen, 166
    zonder deler, 181, 189
deling, deler, 72
denial-of-service, 204
differentiële cryptanalyse, 25
differentiële stroomaanval, 238
differential power attack, 239
Diffie, Whitfield, 94
Diffie-Hellman
    beslissing, 102
    completering, 60, 102, 107
    exponent, 102, 108
    relatie, 102, 180, 273, 296
    sleutelprotocol, 60, 107
        gegeneraliseerd, 296
DigiCash, 244, 253
digitaal paspoort, 122
Digital Milennium Copyright Act, 229
Digital Signature Standard (DSS), 129
digitale handtekening, 120
Direct Betalen (Rabo), 249
directe ruilhandel, 245
Dirichlet, Lejeune, 77
discrete geheimenwinkel, 137, 282
discrete log, 85, 101
drempelschema, 168
driekleuring (graaf), 105, 156
dubbel-DES (2DES), 53

Ecash, 253, 254
Echelon, 220
Ecolex IV, 8
eerlijke muntworp, 133
effectief berekenbaar, 93
Egyptisch vermenigvuldigen, 98
Electronic Frontier Foundation (EFF), 23
elektronisch geld, 125, 250
elektronisch stemmen, 184, 276
Elektronische Handtekeningen, Wet, 228
ElGamal-algoritme, 106, 148, 179

binding, 227
gegeneraliseerd, 296
handtekening, 126
multiplicatief, 274
subgroep, 107, 309
elliptische kromme, 234, 300
$E_{p,a,b}$, 300
groepsstructuur, 301
sleutelgeneratie, 302
encryptie, 1
enkele stroomaanval, 238
entropie, 9
Euclides-algoritme, 72, 95
Euler-criterium, 85, 97
Euler-functie, 79, 110
existentiële vervalsing, 121, 176
experiment (kansrekening), 319
samengesteld, 319
uniform, 319
expliciet geheim, 177
exponentiële codering, 274

f8, 207
f9, 208, 215
factorisering, 93, 101
False Acception Rate (FAR), 160
False Rejection Rate (FRR), 160
Feige, Fiat, Shamir, 148
foutinjectie, 239
Feistel-netwerk, 16, 206
Fermat, kleine stelling, 81, 99
Fermat-pseudopriem, 99
Fermat-test, 99
Fibonacci-getallen, 95
fiduciair geld, 247
fingerprint, 129
forgery, 121
foutinjectie, 111, 239

Galois, Evariste, 303
Gauss (priemgetalstelling), 100
cyclische groep, 84
geheime sleutel, 93
geheime verkiezingen, 276
gehele getallen, 72
gekozen-cijfertekstaanval, 114
subgroep-ElGamal, 108, 310
gekozen-klaretekstaanval, 25, 48, 94, 147
geld, 244
commoditeit, 245
fiduciair, 247
representatief, 246
geldschepping, 251
geldverdubbeling, 251
generator, 82, 295

ggd (grootste gemene deler), 72
giraal geld, 247, 248
glad, 313
globale consistentie, 177
Gouden Standaard, 247
graafkleuring, 105, 156
groep, 292
orde, 294
groepsalgoritme, 295, 317
groepscryptografie, 179, 187, 272
grondrecht, 217
grondwet, 217
grootste gemene deler, 72, 95
grote stappen, kleine stappen, 274, 304
gsm-telefonie, 6
aanval, 57
encryptie, 56
sleutelbeheer, 61
Guillou, Quisquater (ident.), 153

haltingprobleem, 104
hand-over, 202
handtekening, 140
ElGamal, 126, 297
RSA, 126
handtekeningschema, 158
hashfunctie, 128
Chaum, 132, 297
Secure Hash Algorithm, 132
Heide, Willy van der, 142
hellingschema, 167
Hellman, Martin E., 94
cubic-rootaanval, 48
herverdelen, 182, 189
Hill-systeem, 6
Home Environment, 203
homomorf verdeelschema, 269
homomorfe encryptie, 118, 272
hoofdstelling van rekenkunde, 72

IDEA, 29
ideaal protocol, 267
idempotent, 52
identificatie, 140, 214, 249
aanval, 141
Fiat-Feige-Shamir, 148
Guillou-Quisquater, 153
Schnorr, 151, 297
Identikey, 143
impersonificatie, 141
impliciet geheim, 177
index, 84
indexcalculus, 313
$GF(p^n)$, 316
Indiaas machtsverheffen, 97

inpakken (blob), 132
integerfactorisering, 101
integriteit, 120, 214
    SSL, 212
    UMTS, 211
interne consistentie, 177
Interpay, 261
intrekking, 213
invoerblindering, 90, 134
irisherkenning, 161
ITsec, Haarlem, 60, 115

James Bond, 170

kaartspel, 136
Kasumi, 59, 206
Kerberos, 193
Kerckhoffs' principe, 4, 58
keuzeoverdracht, 282
key, 1
key escrow, 129, 173, 223
key management, 59
key recovery, 223
klare tekst, 1
klaretekstaanval, 5
    op DES, 24
kleine stelling van Fermat, 81
Klima-Rosa-aanval (PGP), 111, 310
knapsacksysteem, 111
knoeibestendig, 235
Known Plaintext Attack, 5
Kocher, Paul, 23
kritieke lengte, 10
    DES, 22
kwadraatstelling
    priemmodulus, 85
    samengestelde modulus, 87, 103
kwantumcryptografie, 63

Lagrange, stelling, 80, 98
laptop, 2
Lenstra, Arjan, 291
Linear Feedback Shift Register, 56
lootjes trekken, 286, 289
Lucifer, 28

Maastricht, Verdrag van, 217
malafide deelnemer, 268
man-in-the-middle (UMTS), 204
MD5, 132
meervoudige encryptie, 51
meet-in-the-middleaanval, 52
Message Authentication Code (MAC), 125, 224
    Secure Socket Layer, 212
Micali's vetoschema, 177
Miller-Rabin-pseudopriem, 100
Miller-Rabin-test, 100
minutiae, 160
Misty, 206
modulaire aritmetiek, 76
modulus, 76
Monte-Carlo-algoritme, 100
multipartyberekening, 170, 266
multiplicatief, 273
multiplicatieve inverse, 79
muntworp, 133

National Security Agency (NSA), 30, 218
natuurlijke getallen, 71
network of trust, 124
niet-interactief zero-knowledge proof, 157
nieuwsgierige deelnemer, 268
non-deterministisch polynomiaal, 103
non-repudiation, 120
NP, 103
NP-compleet, 105, 156
nuldeler, 82
nulpolynoom, 185
nulpunt, 185

oblivious transfer, 136, 271, 282
offline (betaling), 252
onderbewust kanaal, 127
one-time pad, 8, 55
one-wayfunctie, 49, 130, 145, 148
online (betaling), 252
onloochenbaarheid, 120, 214
ontcijferen, 1
ontsleutelen, 1
openen (blob), 132
Operatie Donder, 170
orde (element), 80
orde (groep), 79, 294

passieve aanval (identificatie), 141, 144
password, 143
    snooping, 145
perfect veilig, 7, 135
perfecte verhulling, 7, 167
    additief schema, 175
    polynoomschema, 185
perfecte verifieerbaarheid, 168
Philips SmartXA, 234
pinpas, spookopname, 144
plaintext, 1
Pohlig-Hellman, algoritme, 307
Pollard, ρ-algoritme, 305
polyalfabetische substitutie, 6
polynomiaal, 94
polynoom, 83

definitie, 184
polynoomschema (Shamir), 184
power attack, 66
Pretty Good Privacy (PGP), 115, 224
Additional Decryption Key, 173, 226
compressie, 11
encryptie-exponent, 111
hashfunctie, 132
key sharing, 188
Klima-Rosa-aanval, 111, 310
sleutelbeheer, 61, 124
priemgetal, 72
dichtheid, 100
genereren, 99
sterk, 102
priemgetalstelling, 100
priemtest, 99
PrimaCryption, 54, 233
probabilistische encryptie, 273
productversleuteling, 51
pseudopriem, 99
public-keycryptografie, 2, 94
public-keycryptosysteem, 93
ElGamal, 106
knapsack, 111
ontstaan, 94
Rabin, 113
RSA, 108
public-keyinfrastructuur (PKI), 122, 213
publiek aandeel, 168, 175, 297
publiek geheim, 177
publieke sleutel, 93

quorumfactor, 186

Rabin, algoritme, 113, 148
ramp scheme, 167
random-keuzeoverdracht, 282
random-succesoverdracht, 282
Rechten van de Mens, Verklaring, 217
reconstructie, 166
reconstrueerbaarheid, 166
reductiefunctie (Hellman), 50
redundantie, 9
rekenkunde, hoofdstelling, 72
relatie (indexcalculus), 313
representatief geld, 246
respons, 146
restklasse, 77
revocation, 213
rho-algoritme (Pollard), 305
Rijmen, Vincent, 65
Rijndael, 65, 238, 239
ring, 78
roaming, 201
rotormachine, 7
RSA Challenge, 291
RSA-algoritme, 108, 148
foutinjectie, 239
handtekening, 126
kleine encryptie-exponent, 110
multiplicatief, 273
tijdaanval, 237
Russische boerenmethode, 98

S-box (DES), 19
S-key, 145
samengesteld, 72
samengesteld experiment, 319
samengesteldheidstest, 99
Schnorr (identificatie), 151, 297
foutinjectie, 243
schrijfgedrag (biometrie), 161
schuifsysteem, 3
secrecy order, 150
secret splitting, 167
secure device, 235
Secure Electronic Transactions (SET), 250
Secure Hash Algorithm (SHA), 132
Secure Socket Layer (SSL), 211, 250
Senderik, Ralph, 226
Shanks, Daniel, 304
share, 166
shareholder, 166
shift system, 3
Sig, signature, 121
sikkel, 245
simple power attack, 238
Sinterklaas, 286
Skipjack, 129
sleutel, 1
publiek, 93
sleutelafgifte, 173
sleutelbeheer, 59
groepscryptografie, 181, 189
sleutelgeneratie
DES, 17
Rijndael, 66
verdeeld, 181, 189
smart gun, 161
smartcard, 64
SmartXA (Philips), 234
smooth, 313
snooping, password, 145
speciaal algoritme, 295, 317
speciaal getal (fact.), 102
spookopname (pinpas), 144
steganografie, 221, 222
Steins algoritme, 95
stelling

Chinese rest-, 86, 103, 113, 127, 239
Fermat, kleine, 81
Gauss, 84
Lagrange, 80, 98
verjaardags-, 323
sterk botsingsvrij (hashfunctie), 130
sterk priemgetal, 102
stochastische variabele, 320
stroomaanval, 66, 238
stroomversleuteling, 55
integriteit, 215
subgroep, 294
subgroep-ElGamal, 309
subliminal channel, 127
subset-sumprobleem, 105, 111
substitutiesysteem, 4
superstijgende rij, 112
symmetrische cryptografie, 1

tamper proof, 235
telefoonkaart, 232
telewerkprotocol, 60
template (biometrie), 159
Thunderball, 170
ticketserver, 194
tijdaanval, 66, 237
timing attack, 66, 237
transpositiesysteem, 6
traveling salesperson (TSP), 105
triple-DES, 28, 53
Trusted Third Party (TTP), 123, 193, 214, 267

uitgebreid Euclides-algoritme, 73
UMTS, 200
uniform (experiment), 319
universele vervalsing, 121
UNIX, 145

Vasco Identikey, 143
veilig platform, 235
vercijferen, 1
verdeelschema, 165, 166
additief schema, 174
homomorf, 269
polynoomschema, 184
verhulling, 120, 167
verifieerbaar, 168
verify, signature, 121
verjaardagsaanval, 130
verjaardagsstelling, 323
versleutelen, 1
vertrouwelijkheid, 194
vervalsing, 121
hashfunctie, 129
verwachting (stochast), 320
vetoschema, 167
additief schema, 174
Micali, 177
Vigenère-systeem, 5
villabewoners, 170, 269, 274
vingerafdruk, 160
visuele cryptografie, 166, 175
voortbrenger (generator), 295
VoteHere, 279

Wassenaar, akkoord, 116, 218
weigering (identificatie), 148
wettig betaalmiddel, 247
Willem Alexander, 77
wortel (polynoom), 83
worteltrekken (modulair), 97

X.509, 213
XTR, 234, 302

zero-knowledge proof, 136, 150, 175, 236, 257
niet-interactief, 157
Zimmermann, Philip, 116
zwak botsingsvrij (hashfunctie), 130