Cryptografie: Van DES tot Chipknip

Gerard Tel, Informatica Instituut
email: gerard@cs.uu.nl

Dit artikel geeft een uiterst beknopt overzicht van enkele belangrijke technieken op het gebied van de cryptografie en hun toepassing in het electronische betalingsverkeer. Er wordt een belangrijk onderscheid uitgelegd, namelijk tussen *symmetrische* en *asymmetrische* (ook wel: *public key*) cryptografische algoritmen. De mogelijke toepassingen van deze twee klassen lopen sterk uiteen; het is dan ook van belang te weten, wanneer men een symmetrisch, dan wel asymmetrisch algoritme moet kiezen.

Als voorbeeld behandelen we o.a. PGP, een email-systeem waarin geschikte keuzen zijn gemaakt, en we voorspellen een debacle van de electronische betaalmiddelen Chipper en Chipknip, waarin men uit zuinigheid voor de verkeerde technieken heeft gekozen.

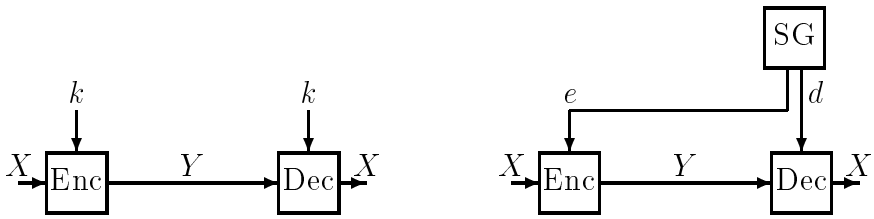
1 Cryptografische Algoritmen

Een eerste toepassing van cryptografie is het 'versleutelen', of onbegrijpelijk maken van geheime informatie, doorgaans voor verzending over een netwerk of opslag. Zoals we straks zullen zien, heeft cryptografie in ruimere zin ook te maken met het al dan niet uit kunnen voeren van bepaalde berekeningen.

Het aantal algoritmen dat voor versleuteling kan worden gebruikt is verrassend klein; slechts een handvol algoritmen hebben jarenlange pogingen tot 'kraken' kunnen doorstaan. We bespreken in deze sectie twee van die algoritmen, namelijk DES en RSA, als representant van de 'symmetrische' en 'asymmetrische' cryptografische algoritmen.

1.1 Symmetrische en Public Key Algoritmen

Bij versleuteling of encryptie denken we aan een algoritme (programma of machine) dat de informatie onleesbaar maakt door op de bits en bytes allerlei operaties uit te voeren, bijvoorbeeld verwisselen van gedeelten, of exclusive OR te nemen met een andere bitrij. De versleutelmethode is altijd parametriseerbaar: dat betekent, dat je de machine moet instellen op een bepaalde versleuteling en die instelling noemen we de sleutel of 'key' k. Bij de symmetrische algoritmen is de key steeds een willekeurige bitrij. Natuurlijk dienen we erop te letten, dat het proces *omkeerbaar* is, want uit de versleutelde informatie (de ciphertext) moet de oorspronkelijke informatie (plaintext of klare tekst) weer terug te berekenen zijn.



Figuur 1: Symmetrische en Asymmetrische Versleuteling.

Het is deze omkeerbaarheid, die bij de symmetrische en asymmetrische algoritmen op totaal verschillende manier wordt bereikt.

Bij de symmetrische algoritmen is elke bewerkingsstap *op zich omkeerbaar*; bijvoorbeeld het XORen met bitrij Y kan worden opgeheven door opnieuw met Y te XORen en een permutatie van de bits kan worden opgeheven door de inverse permutatie toe te passen. Dit betekent, dat kennis van het versleutelproces, of liever, de instelling van de versleutelmachine, voldoende informatie inhoudt om de operaties in omgekeerde volgorde weer terug te rekenen. De ontsleuteling, of decryptie, gebeurt door dezelfde sleutel k weer in de machine te stoppen en deze op 'decryptie' te zetten. Omdat encryptie en decryptie dezelfde sleutel gebruiken, noemen we deze klasse van algoritmen 'symmetrisch'.

Bij asymmetische algoritmen maken we gebruik van *onomkeerbare* operaties, waarbij je als voorbeeld mag denken aan reductie modulo een groot getal. Inderdaad kun je van een getal x best de rest uitrekenen bij deling door een getal n ($x \bmod n$ dus), maar aan die rest kun je niet meer zien wat x was en daarmee is deze stap onomkeerbaar. Dit betekent, dat je een berekening kunt zien als een *one-way* functie: kennis van het eindresultaat en het berekeningsschema is doorgaans onvoldoende om de invoer te kunnen reconstrueren.

Natuurlijk bestaat er wel een berekening die het origineel kan terugvinden, anders was versleutelde informatie voorgoed verloren. Maar hoe die berekening verloopt, is aan de versleutelingsmethode niet te zien. Anders gezegd, de machine-instelling voor *decryptie*, d, is anders dan die voor encryptie, e, en niet daaruit afleidbaar! Omdat er twee verschillende sleutels worden gebruikt noemen we deze klasse van algoritmen 'asymmetrisch'. Anders dan bij de symmetrische methoden, behoeft de encryptie-sleutel e niet geheim te worden gehouden, vandaar de benaming 'public key' cryptografie.

Omdat decryptie wel de inverse van encryptie moet zijn, moet de d-sleutel 'passen' op de e-sleutel en je kunt ze dus niet onafhankelijk van elkaar kiezen. Er is altijd een 'sleutelgeneratie' algoritme, dat een 'passend paar' sleutels kan fabriceren. Zie Figuur 1 voor een schematisch overzicht van symmetrische en asymmetrische versleuteling.

1.2 Data Encryption Standard (DES)

Een bekende symmetrische methode is de Data Encryption Standard (DES) die in 1974 door IBM is ontwikkeld. Deze methode versleutelt de data in blokken van 64 bits, en heeft

een sleutel van 56 bits; Stinson [Sti95, Hf. 3] geeft een uitvoerige technische beschrijving van de werking. De data wordt in 16 achtereenvolgende (inverteerbare!) ronden gemanipuleerd door bit-operaties zoals het permuteren van bits en het XOR-en met bitrijen.

Het ontwerp was erop gericht, een snelle implementatie in hardware mogelijk te maken en in dit opzicht is DES succesvol. Er bestaan al chips (van 10 gulden ongeveer) die 50 miljoen versleutel-operaties per seconde kunnen uitvoeren. Dit betekent, dat DES kan worden gebruikt voor de beveiliging van data-lijnen van de hoogste bandbreedte (Gigabit-lijnen).

Maar hoe veilig is DES? Een tegenstander die de vercijferde informatie heeft onderschept, zou kunnen proberen het origineel te achterhalen door alle sleutels uit te proberen, tot er een leesbaar resultaat ontstaat; een dergelijke computationele krachtpatserij noemen we een Brute Kracht Aanval. Zo'n aanval is in principe tegen elk cryptografisch algoritme mogelijk, en de enige bescherming is een voldoende lange sleutel. En dan blijkt de sleutellengte van 56 bits toch een beetje achterhaald: het aantal mogelijke sleutels is "slechts" 2^{56} ofwel 7×10^{16} en met de genoemde snelle DES chips is de bouw van een parallelle DES-cracker binnen de mogelijkheden. Deze zomer werd door de Electronic Frontier Foundation een dergelijk knutselwerkje aan het publiek onthuld.

Een cryptografisch algoritme wordt meestal als een mislukking beschouwd als er een of ander truukje wordt uitgevonden dat in deze aanval 'bochten kan afsnijden' door bijvoorbeeld grote verzamelingen sleutels uit te sluiten, of zonder kennis van de sleutel het origineel kan achterhalen. Een dergelijke truuk is echter voor DES nooit gevonden, en gezien het doordachte ontwerp van de methode is het niet waarschijnlijk dat zo'n truuk bestaat. Stinson besteedt aandacht aan pogingen zo'n truuk te vinden, maar die zijn alle vergeefs gebleven. Zo kunnen we dus zeggen, dat DES *algoritmisch* zeer sterk is, maar door zijn korte sleutels niet voldoende weerstand biedt tegen een brutekracht-aanval. Stinson beschrijft ook allerlei interessante aanvallen die door de korte sleutellengte tegen DES mogelijk zijn.

In de bancaire wereld is DES nog steeds erg populair, maar dan in driedubbele vorm: er wordt gewerkt met 168 bits sleutels, waarmee drie achtereenvolgende DES-vercijferingen worden uitgevoerd. Men noemt dit 'triple DES' of 3DES. Een brutekracht-aanval op een zo lange sleutel is voorlopig nog onuitvoerbaar, zelfs voor de allergrootste supercomputers. Een ander algoritme dat opgang doet is IDEA (International Data Encryption Algorithm). IDEA gebruikt veel van de ideeën van DES, maar heeft een grotere sleutel: 128 bits, voldoende om brutekrachtaanvallen te ontmoedigen. Het bewerkt de data in blokken van 64 bits tegelijk, maar de gebruikte operaties zijn niet bit-gewijs zoals bij DES, maar op 'woorden' van 16 bits. De reden is dat men niet hardware-implementaties, maar software-implementaties efficient mogelijk heeft willen maken. Een discussie over de veiligheid en het gebruik van DES en de diverse varianten vinden we bij Schneier [Sch96, Hf. 12].

In het vervolg zal ik vele malen spreken over DES of 3DES, maar het gezegde gaat doorgaans voor alle symmetrische algoritmen op.

1.3 Modulaire Exponentiaties: RSA

In 1978 ontwierpen Rivest, Shamir, en Adleman een systeem dat gebaseerd was op one-way operaties. De versleuteling gebruikt een groot getal n, en bevat reducties modulo n, zodat het 'terugdraaien' van operaties niet mogelijk is. Naast n gebruikt de encryptie een *encryptie-exponent* e, en de complete encryptie-functie is zo eenvoudig als: $y = f(x) = x^e \bmod n$. De gebruikte getallen hebben doorgaans een lengte van 512 of 1024 bits (155 of 310 cijfers) en de exponentiatie kost kubische tijd. De RSA methode is daarom lang zo snel niet als DES: de beste implementaties halen ongeveer 640 kbs, dus ruim duizendmaal minder dan de snelste DES implementaties.

Het terugrekenen is erop gebaseerd dat, net als in de continue rekenkunde, een machtsverheffing kan worden geïnverteerd door een tweede machtsverheffing. Bekijk, in de reële getallen, eens een machtverheffing, bijvoorbeeld het nemen van de zevende macht. De inverse, het trekken van de zevendemachts-wortel, is in feite ook een machtsverheffing, namelijk tot de macht $\frac{1}{7}$, zoals uitgedrukt in de formule $(x^7)^{1/7} = x$. De exponent $\frac{1}{7}$ is dus een soort 'tegen-exponent' van 7 in de zin dat exponentiëren met de machten 7 en $\frac{1}{7}$ elkaars inverse zijn.

Ook in de modulaire aritmetiek die bij RSA wordt gebruikt is er bij elke exponent e een 'tegen-exponent' d met de eigenschap dat $(x^e)^d = x$ voor elke x. Helaas voor snoodaards, maar gelukkig voor ons, gaat de vergelijking met de reële getallen verder niet op: de tegen-exponent d kan namelijk, anders dan bij de reële getallen, *niet* zomaar uit de exponent e worden berekend!!

Aan de andere kant is het *wel* heel gemakkelijk om uit een berekening een getal n, en een exponent e met zijn tegen-exponent d te laten komen. Dit heeft te maken met een ander moeilijk te inverteren probleem, dat van *factorisatie*. Een *priemgetal* is een natuurlijk getal dat behalve zichzelf en 1 geen andere delers heeft: 2, 7, 97, en 27961 zijn priemgetallen. Nu is het bekend, dat elk getal op slechts één manier kan worden geschreven als product van priemgetallen (bijvoorbeeld, $1001 = 7 \times 11 \times 13$) maar welke priemgetallen dat zijn, is (gelukkig!) heel moeilijk uit te rekenen. Zo heb ik bijvoorbeeld twee priemgetallen met elkaar vermenigvuldigd en het resultaat 39944285726269 verkregen; weet je nu welke getallen dat zijn geweest?

De modulus (n) die bij RSA wordt gebruikt is altijd het product van twee grote priemgetallen, zeg $n = p.q$. Het berekenen van de tegen-exponent van e is mogelijk voor wie de factoren van n kent, maar zonder deze kennis is het ondoenlijk. (Het uiteenzetten van de volledige getaltheoretische achtergrond gaat hier te ver, zie bijvoorbeeld [Sti95, Hf. 4].) Omdat het ondoenlijk is om uit n de factoren p en q te berekenen, is ook het berekenen van de tegenexponent, gegeven n en e, niet efficient mogelijk.

Bij de sleutelgeneratie werken we natuurlijk vanuit p en q:

1. Kies twee grote priemgetallen p en q.
2. Bereken $n = p.q$ en kies e.
3. Gebruik p en q voor het berekenen van d.
4. Gooi p en q weg, publiceer (n, e) en bewaar d geheim.

Merk op, dat elke gebruiker een eigen modulus heeft. Het resultaat: iedereen mag de

versleutel-parameters n en e weten, en kan dan informatie onleesbaar maken door tot de e-de macht te verheffen. De informatie kan weer leesbaar worden gemaakt door te verheffen tot de tegen-exponent d, maar alleen degene die de sleutels heeft gemaakt kent d.

Naast het systeem van Rivest, Shamir, en Adleman zijn er nog enkele public key systemen in gebruik, bijvoorbeeld het ElGamal systeem. Wat in het vervolg over RSA wordt gezegd, geldt mutate mutandis ook voor de andere public key systemen.

1.4 Vergelijking van Symmetrische en Asymmetrische Algoritmen

Zoals reeds opgemerkt, gaat het vercijferen en ontcijferen van data met een symmetrisch systeem razendsnel, terwijl de asymmetrische systemen veel trager zijn. Aan de andere kant bieden asymmetrische systemen grote voordelen op het gebied van onder andere sleuteldistributie en toepasbaarheid in allerlei merkwaardige protocollen.

1.4.1 Sleuteldistributie

Voor berichtuitwisseling tussen twee partijen (Alice en Bob) met een symmetrisch algoritme is het nodig dat de partijen over een gemeenschappelijke sleutel beschikken en dat is nog niet zo eenvoudig. Het is immers niet mogelijk, de sleutel door één partij te laten kiezen en over het netwerk te laten versturen! Alice en Bob zullen dus een keer bij elkaar moeten komen om de sleutel af te spreken; een alternatief is, gebruik te maken van een tussenpartij die ze kan helpen bij het samen kiezen van een sleutel.

Verder is het aantal benodigde sleutels erg groot: immers, als we niet willen dat een derde (Carol) ook de berichten tussen Alice en Bob kan lezen, dan zijn er *andere* sleutels nodig voor de communicatie tussen Alice en Carol, resp. Bob en Carol. Zijn er m partijen die elk met ieder ander willen communiceren, dan zijn er $\frac{1}{2}m(m-1)$ sleutels nodig.

Bij asymmetrische systemen is de sleuteldistributie erg gemakkelijk en zijn er ook minder sleutels nodig. Alice en Bob kiezen ieder voor zich één sleutelpaar, waarvan het publieke deel aan iedereen bekend kan worden gemaakt. Dit kan gewoon over het netwerk gebeuren, want het is niet erg als afluisteraars de publieke sleutel vernemen. Om een bericht aan Alice te verzenden, kan Bob het bewerken met de publieke sleutel van Alice; dezelfde sleutel is bij Carol en alle afluisteraars bekend, maar kan niet worden gebruikt om het bericht van Bob weer leesbaar te maken.

1.4.2 PGP, een Verstandshuwelijk tussen IDEA en RSA

Dat symmetrische en asymmetrische algoritmen voor verschillende taken geschikt zijn is erkend in het ontwerp van het email beveiligingsprogramma PGP (Pretty Good Privacy) door Philip Zimmermann; zie [Sch96, Sec. 24.12]. In PGP wordt een asymmetrisch algoritme, namelijk RSA, gebruikt voor sleuteldistributie, en een symmetrisch algoritme (IDEA in dit geval) voor het feitelijke versleutelen van het email bericht.

Elke PGP gebruiker heeft een RSA sleutel, waar van het publieke deel overal kan worden aangekondigd, en de decryptie-exponent wordt natuurlijk geheim gehouden. Alice neemt (onder meer) de volgende stappen om een email M aan Bob te sturen:

1. Zij achterhaalt de publieke sleutel (n, e) van Bob.
2. Zij kiest een random string k als *sessie key*.
3. Zij versleutelt M onder IDEA met sleutel k.
4. Zij versleutelt k onder RSA met sleutel (n, e).
5. Zij stuurt de twee codes naar Bob.

Bob kan het bericht lezen door eerst k terug te vinden met zijn geheime RSA sleutel, waarna hij het eigenlijke bericht opent met IDEA.

Het is voor tegenstanders niet haalbaar de eenmaal vercijferde boodschap te ontcijferen zonder kennis van Bob's geheime sleutel. Een mogelijke aanval is, Alice een valse sleutel in handen te spelen (waarvan de tegenstander het geheime deel kant!) en haar in de waan te brengen dat dit Bob's sleutel is. PGP bevat een mechanisme, gebaseerd op echtheids-certificaten van sleutels, waarmee Alice zich van de echtheid van de sleutel kan overtuigen.

2 Protocollen

Behalve de directe beveiliging van data worden cryptografische algoritmen gebruikt in protocollen om de wel-gevormdheid van data-uitwisseling af te dwingen. De public-key systemen lenen zich beter voor protocollen dan symmetrische systemen, en dit zal in het vervolg gaan blijken bij de studie van handtekeningen en identificatie.

2.1 Privacy in de Electronic Market Place

Als Alice een key paar (n, e) en (n, d) genereert en d geheim houdt, is en blijft ze de enige die de inverse van de functie $x \to x^e \bmod n$ kan uitrekenen. In de informatie-economie is deze kennis een economisch verhandelbaar goed, dwz., Alice kan (door advertenties) een vraag naar e-demachts wortelberekeningen kweken en aanbieden deze voor geld te berekenen! Bob kan van deze dienst gebruik maken door een getal y en 10 Euro in te zenden, en ontvangt dan y^d van Alice terug. We merken op dat Bob weliswaar zelf deze berekening niet kan uitvoeren, maar hij kan wel het resultaat eenvoudig op juistheid controleren.

Het lijkt misschien wat vergezocht dat Alice, na keuze van grote getallen n en e, Bob zo gek kan krijgen geld te betalen voor het berekenen van e-demachts wortels modulo n. Als dit zo een interessant probleem is kan Bob toch zelf ook wel zo'n key-paar kiezen waarvan hij dan ook zelf de geheime exponent d kent? In de Electronic Market Place is dit echter de gewoonste zaak van de wereld zoals wij verderop zullen zien.

2.1.1 Privacy door Invoer-Blindering

Bob wil van deze dienst gebruik maken, maar zonder aan Alice te onthullen wat zijn y is; dit blijkt te kunnen door *input blindering* zoals beschreven in Protocol 2. Eigenschappen

Bob bezit y en wil een z met $z^e = y$ weten:

1. Bob kiest een random getal k en berekent $b = k^e$, de *blinder*.
2. Bob berekent $y' = y.b$, de *geblindeerde invoer*.
3. Bob stuurt 10 Euro en y' aan Alice.
4. Alice stuurt Bob $z' = y'^d$.
5. Bob deelt: $z = z'/k$.

Protocol 2: INVOER-BLINDERING.

van dit protocol:

Lemma 2.1 (Correctheid.) *Bob verkrijgt z met $z^e = y$.*

Bewijs. Er geldt $z^e = (z'/k)^e = z'^e/k^e = y'/b = y$. □

Lemma 2.2 (Perfecte verhulling.) *Alice komt niet achter y.*

Bewijs. Alice ontvangt van Bob de waarde $y' = y.b = y.k^e$; kan ze hieruit de vooralsnog onbekende y berekenen? Helaas, voor *elke mogelijke* waarde van y bestaat er een k waarvoor $y' = y.k^e$, namelijk $k = (y'/y)^d$, en Alice weet niet over *welke* k Bob beschikt. □

2.1.2 Toepassing: Geheimenhandel met Verzekerde Discretie

De eerste verschijningen van de Europese Electronic Marketplace dienen zich al aan: het is al de gewoonste zaak van de wereld om informatie aan te bieden, cq. te kopen over het Internet. Alice kan op het Internet een "geheimen-winkel" drijven; onder een geheim wordt hier niet iets verstaan dat niemand mag weten, maar enige vorm van economisch verhandelbare informatie. Meestal bestaat het aanbod uit porno, maar laten wij het zedelijk houden en aannemen dat Alice weerberichten verkoopt: haar assortiment bestaat uit eendags-weerberichten voor elk van de zeven komende dagen, en elk weerbericht kost 10 Euro.

Bob wil een weerbericht kopen voor een bepaalde dag, maar uit privacy-oogpunt houdt hij liever voor zich welke dag dat is. Voor Alice maakt dit eigenlijk niet uit, want alle weerberichten zijn even duur; Alice neemt de privacy van haar cliënten hoog op en het kan haar niet schelen *wat* Bob krijgt, als hij maar betaalt. De overdracht van een geheim van Alice, waarbij ze zelf niet weet welk geheim ze overdraagt, heet in de literatuur *oblivious transfer* en verloopt volgens Protocol 3.

Alice plaats haar geheimen op haar web server, elk versleuteld onder een andere sleutel; een symmetrisch algoritme is hier geschikt. De s sleutels codeert ze met haar RSA-key (n, e) en zet ze ook op de server. De versleutelde geheimen zijn natuurlijk waardeloos zonder de bijbehorende keys, en daarom kan Alice de afrekening van de geheimen baseren

Eerste fase: voorbereiding.

1. Alice genereert verhandelbare geheimen G_1 tot G_s.
2. Alice kiest encryptiesleutels k_1 tot k_s en berekent $P_i = E_{k_i}(G_i)$.
3. Alice berekent $y_i = (k_i)^e$.
4. Alice openbaart de G_i en y_i op haar web server.

Tweede fase: Bob wil geheim j kopen:

1. Bob haalt G_j en y_j van Alice' server.
2. Bob betaalt Alice 10 Euro om de wortel uit y_j te berekenen.
3. Bob gebruikt de verkregen x_j als sleutel om G_j te decrypten.

Protocol 3: Oblivious Transfer.

op het ontsleutelen van de y_i. Bob betaalt in stap 2 een bedrag om één van de s sleutels te laten openmaken, dwz., hij betaalt 10 Euro voor het trekken van een wortel. Hij maakt hierbij gebruik van invoer-blindering (volgens Prot. 2) als hij niet wil dat Alice weet welk geheim hij koopt.

2.2 Digitale handtekeningen

Een digitale handtekening is een toevoeging of cryptografische bewerking van een bericht waardoor de herkomst met zekerheid kan worden vastgesteld. Anders dan bij een klassieke handtekening (die door het papier onlosmakelijk is verbonden met de mededeling waaronder hij gezet is) moeten we speciaal letten op de binding van de handtekening aan het betreffende bericht. Daarom is de handtekening altijd een functie van het bericht.

2.2.1 Symmetrische Handtekeningen

In een situatie van symmetrische cryptografie beschikken Alice en Bob over dezelfde sleutel k. Alice kan een bericht voor Bob ondertekenen door het met sleutel k te vercijferen; omdat Alice, naast Bob, de enige is die over k beschikt, verkrijgt Bob de zekerheid dat het ontvangen bericht van Alice afkomstig is.

Helaas is bij een symmetrisch systeem de informatie die Bob nodig heeft om deze handtekening te *verifiëren*, dezelfde als die Alice gebruikt heeft om de handtekening te *produceren*. Dit betekent, dat Bob nimmer een derde partij van zijn gelijk zal kunnen overtuigen: ten eerste al, omdat Bob zelf het vercijferde bericht had kunnen produceren met zijn kennis van k; ten tweede vereist arbitrage door een derde partij dat Bob de sleutel k aan deze partij overdraagt, wat natuurlijk onwenselijk is.

De conclusie is, dat symmetrische cryptografie *geen* geschikt voertuig is voor de vervaardiging van handtekeningen.

2.2.2 Public Key handtekeningen

Gebruiker Alice genereert een RSA sleutelpaar (n, e) met geheim deel d. Alice is nu de enige die een e-demachts wortel modulo n kan trekken (namelijk door exponentiätie met d), en deze berekening is *wel* verifieerbaar maar *niet* uitvoerbaar met behulp van e! De digitale handtekening van Alice onder bericht m is het getal $s = m^d$. Het volstaat, dit getal s aan Bob te zenden want Bob kan dan met de publieke informatie (n, e) de waarde van m vinden. Bob verkrijgt bericht m met de handtekening s die hem de zekerheid geeft dat het bericht door Alice is opgemaakt. Bob kan de handtekening aan een rechter overleggen, die met de *publieke* sleutel van Alice kan concluderen dat het bericht van Alice afkomstig is. Bob, noch de rechter, beschikken over de informatie die nodig is om de handtekening te *produceren*.

We zien een zeer belangrijk onderscheid opduiken tussen de *produceerbaarheid* en de *verifieerbaarheid* van een bepaalde uitkomst. Dit onderscheid kan met symmetrische algoritmen niet worden gemaakt. Oplettende lezers zullen misschien tegenwerpen dat dit verschil *niet kan bestaan* indien $P = NP$, en dat de ongelijkheid $P \neq NP$ (nog?) niet is aangetoond. Inderdaad zal de cryptografie als een kaartenhuis inelkaar storten als zou blijken dat $P = NP$, maar dit is onwaarschijnlijk.

2.3 Identificatie

We zullen nu bekijken, hoe men zekerheid kan verkrijgen over de identiteit van een partij waarmee men zaken doet. In de informatie-wereld herkent men zijn partner niet aan stem of uiterlijk, maar (uiteraard) aan een informatie-kenmerk, namelijk kennis van een specifieke informatie. Vergelijk dit met het inloggen op een computersysteem, waarbij men zich bekend maakt met een login-naam, en vervolgens zijn identiteit *bewijst* door het intypen van een geheim password.

In het verdere stellen we ons voor dat Bob de vorm aanneemt van een terminal waarop Alice wil inloggen, of een Betaalautomaat of Flappentap waar Alice geld wil opnemen; voor het gemak heten deze installaties in de bancaire wereld ook terminals.

2.3.1 Malafide Terminals

Een voor de hand liggende manier voor Alice om zich aan Bob te identificeren is, haar geheim te overleggen (door het intypen van haar password of PIN); het is tevens een gevaarlijke manier. Een populaire tijdsbesteding van de maffia is het nabouwen van terminals; zo verschenen er op luchthavens apparaten, waaruit men cigaretten kon halen tegen 'betaling' met de PIN-betaalpas. De apparaten waren echter niet op de banken aangesloten, maar copiëerden alle pas-gegevens en registreerden de ingetypte PINs, waarna passen werden nagemaakt en rekeningen leeggeplukt.

We zien dat het voor Alice riskant is zich te identificeren door het overleggen van haar geheim aan Bob; als Bob het geheim heeft gehoord, kan hij zich bij andere terminals als Alice impersonifiëren. We merken direct op, dat, indien hetzelfde password bij meerdere

Aanname: Bob en Alice kennen beide een sleutel k.

1. Een dame meldt zich bij Bob en beweert Alice te zijn.
2. Bob kiest een random x en berekent $y = E_k(x)$.
3. Bob geeft x aan de dame en vraagt, deze te versleutelen.
4. Alice berekent $z = E_k(x)$ en geeft z aan Bob.
5. Bob vergelijkt zijn y met de z van de dame.

Protocol 4: Symmetrische Identificatie.

terminals geldig is, er ook voor echte terminals de mogelijkheid is zich te gaan misdragen en zich bij andere terminals als Alice voor te doen.

2.3.2 Symmetrische Identificatie

We laten Alice niet haar geheim aan Bob geven, maar we laten haar een berekening uitvoeren met haar geheim, die door Bob kan worden gecontroleerd indien hij eveneens over het geheim beschikt; de berekening is een versleuteling van een random bitrij; zie Prot. 4. Het is voor Alice niet schadelijk, voor een nep-terminal een random bitrij te versleutelen.

Dit protocol wordt in de praktijk gebruikt, maar de problemen die aan symmetrische identificatie zijn verbonden, zijn enorm.

1. Elke terminal kent de sleutel k van Alice, dus terminals (danwel, de terminalhouders) kunnen zich bij andere terminals als Alice voordoen.

2. Hoe kent de terminal de sleutels van alle mogelijke gebruikers? Het is natuurlijk niet mogelijk, Alice deze sleutel aan de terminal te laten verstrekken, omdat ze hem daardoor aan een nep-terminal zou kunnen onthullen.

2.3.3 Public Key Identificatie

Al deze problemen worden met public key cryptografie elegant opgelost: we behandelen nu het identificatie-protocol van Gilliou en Quisquater; zie [Sti95, Sec. 9.4]. Dit is een voorbeeld van een zogenaamd *Zero-knowledge Proof* techniek, waarmee Alice *bewijst* over zekere informatie te beschikken, zonder ook maar iets van deze informatie prijs te geven.

Protocol 4 is *niet* zero-knowledge: een nep-terminal verkrijgt weliswaar niet de begeerde geheime sleutel van Alice, maar verkrijgt wel een berekeningsresultaat (namelijk $z = E_k(x)$) dat zonder hulp van Alice niet berekend had kunnen worden. De paradox die we gaan oplossen is, dat (1) Alice kennis van haar geheim gaat demonstreren door iets te berekenen, wat je alleen met dat geheim kunt uitrekenen, (2) Alice geeft aan Bob geen enkele informatie die je zonder dat geheim niet zou kunnen berekenen!!

Het gehele systeem maakt gebruik van een *enkele* RSA sleutel (n, e) waarvan het geheime deel uitsluitend bij de systeembeheerder (bank) berust. Gebruiker Alice krijgt van

Aanname: Bob kent het *publieke* wachtwoord a van Alice en $(n,\ e)$.

1. Alice meldt zich bij Bob en beweert het bij a passende geheim te kennen.
2. Alice kiest een *random* r en stuurt $s = r^e$ aan Bob.
   Ze claimt hiermee de wortels uit s en $s.a$ te kennen.
3. Bob gooit een munt en kiest voor 0 (s) of 1 ($s.a$).
4. Alice geeft Bob r (ingeval van 0) of $r.k$ (ingeval van 1).
5. Bob verifiëert dat het antwoord klopt.
6. Dit wordt 20 keer herhaald.

Protocol 5: Gilliou en Quisquater Identificatie.

de bank een publiek wachtwoord, bijvoorbeeld haar naam `Alice` als getal a gerepresenteerd, en een geheim wachtwoord $k = a^d$; merk op dat het publieke wachtwoord (bekend bij de terminals) verschilt van het geheime (dat Alice alleen zelf kent). Alice identificeert zich bij Bob door te bewijzen een getal k te kennen waarvoor geldt $k^e = a$ en dit doet zij volgens Protocol 5.

Uiteraard kan Alice haar claim niet bewijzen door k te overleggen, hiermee zou zij immers haar unieke geheim prijsgeven. Alice stuurt Bob een getal s en beweert, de e-demachts wortels uit *zowel* s als $s.a$ te kennen. Nu is het niet zo bizonder dat ze de wortel uit s kent, want ze heeft net s berekend als de macht van een random getal; het kennen van beide wortels is echter voorbehouden aan Alice (of beter, bezitters van het geheim k).

Lemma 2.3 *Wie y_0 en y_1 kent, waarvoor geldt dat $y_0^e = s$ en $y_1^e = s.a$, kent k.*

Bewijs. Zo iemand kan altijd y_1/y_0 berekenen, en omdat $(y_1/y_0)^e = y_1^e/y_0^e = s.a/s = a$, is het resultaat van de deling k. □

Alice moet dus oppassen, en niet de beide antwoorden aan Bob opsturen, maar het is volstrekt ongevaarlijk voor haar, en heeft geen waarde voor Bob, om één van de antwoorden op te sturen.

Lemma 2.4 *Iedereen kan, ook zonder kennis van k, (0) een s en y produceren zdd $y^e = s$, OF (1) een s en y produceren zdd $y^e = s.a$.*

Bewijs. Een paar $s,\ y$ als onder (0) produceer je door een random y te nemen, en $s = y^e$. Een paar $s,\ y$ als onder (1) produceer je door een random y te nemen, en $s = y^e/a$. □

De muntworp van Bob is een uitdaging aan Alice om te bewijzen dat ze beide antwoorden kent door er slechts een te vertellen. Immers, Alice stuurt s aan Bob *voordat* ze weet, of ze de wortel van s of die van $s.a$ moet onthullen, en ze moet dus over *beide* antwoorden beschikken om met 100% kans het antwoord op de door Bob gekozen som te kunnen geven. Alice kan, spelend volgens het protocol, twintig maal het goede antwoord aan Bob overleggen. Iemand die een s opstuurt waarbij hij slechts één van de twee antwoorden kent, heeft

slechts 50% kans dat Bob dat antwoord vraagt. Iemand die k niet kent, heeft een kans van minder dan 1 op een miljoen om twintig maal het goede antwoord te kunnen geven.

We bekijken nu de problemen die verbonden waren met het symmetrische identificatieprotocol, als eerste het bedrog door malafide terminals. Terminal Bob beschikt voor de identificatie en na de identificatie van Alice over niets anders dan de *publieke* sleutel a van Alice, terwijl kennis van de *geheime* sleutel k nodig is om het protocol met succes als Alice te kunnen doorlopen. De kennis van een terminal is *wel* voldoende om Alice te kunnen herkennen, maar *niet* om Alice te kunnen impersonifiëren.

Vervolgens kijken we naar de sleuteldistributie en merken op dat herkenning slechts de publieke sleutel vereist, en dat misbruik hiervan niet mogelijk is: kortom, de publieke sleutel kan vrijelijk worden verspreid, en zelfs door Alice aan Bob worden verstrekt *bij de identificatie zelf*! Hierbij meldt Alice zich bij Bob met de mededeling: "Ik ben Alice en mijn publieke sleutel is a."

Voorkomen moet worden, dat iemand de verkeerde publieke sleutel overlegt: een schurk kiest een random k', berekent $a' = k'^e$, en beweert Alice te zijn en overlegt a'! Dit wordt eenvoudig uitgesloten door elke gebruiker een 'digitaal paspoort' ofwel sleutel-certificaat te geven. Dit is een door de centrale autoriteit (bank, systeembeheer) uitgegeven en getekende verklaring van de vorm "De publieke sleutel van Alice is a". De terminal behoeft nu slechts de publieke sleutel van deze autoriteit te kennen om de echtheid van de aangeboden publieke sleutel te kunnen controleren.

3 Electronisch Betaalverkeer

Betalen door middel van cryptografisch beveiligde informatieoverdracht is een zeer interessante toepassing van de behandelde protocollen, zowel uit technisch oogpunt als vanwege de recente invoering van electronische betaalmiddelen (Chipper en Chipknip) in het Nederlandse monetaire verkeer. Na een korte definiërende uiteenzetting van geld en betalen (Subsec. 3.1) beschrijven we een elegant systeem gebaseerd op Public Key cryptografie, en het Chipper/Chipknip systeem. Uitgebreider materiaal vindt men in de scriptie van Marjan de Vries [Vri96].

3.1 Geld en Betalen

Om de discussie te vergemakkelijken zullen we stellen dat de meest elementaire vorm van geld de *girale* is, dwz., dat geld bestaat uit een saldo dat de bezitter ervan heeft bij de bank. Ter vereenvoudiging nemen we aan dat er slechts één bank is, waarbij zowel de koper (betaler) als de verkoper (ontvanger) een rekening hebben. Een betaling van een bedrag x van Alice aan Bob bestaat dus hieruit, dat het saldo van Alice met x verminderd, en dat van Bob met x vermeerderd wordt. Het eerste gedeelte is de *debetering* van Alice, het tweede de *creditering* van Bob.

Voor het doen plaatsvinden van de betaling worden in Nederland twee modellen op grote schaal gebruikt.

Het debet/credit model. Er wordt een opdracht aan de bank verstrekt om de debetering en creditering tesamen uit te voeren, bijvoorbeeld in de vorm van een overschijvingsopdracht, een PINpas-transactie of een girobetaalkaart of cheque. Bij de overschijvingsopdracht, girobetaalkaart of cheque vindt de betaling niet gelijk plaats met de aankoop of dienstverlening, maar dat is geen probleem (o.a. wegens garantie van betaalcheques); let wel op dat debetering en creditering onlosmakelijk met elkaar verbonden zijn.

Het Saldo-certificaten model. Alice kan een gedeelte van haar geld *overdraagbaar* maken door saldo te converteren in niet-persoonsgebonden saldo-certificaten. Tegen afboeking van saldo (debetering dus) ontvangt Alice enkele certificaten, en zij kan een of meer van deze certificaten ter betaling aan Bob geven. De certificaten zijn niet-persoonsgebonden, dus Bob kan ze bij de bank weer converteren in saldo: creditering tegen inlevering van certificaten dus.

Debetering en creditering zijn onafhankelijk en niet met elkaar in verbinding te brengen. Bij het opnemen van saldo-certificaten zal de besteding ervan doorgaans nog niet eens bekend zijn. De meeste gebruikers houden een voorraadje certificaten in voorraad om kleine uitgaven snel te kunnen betalen zonder dat het banksaldo moet worden aangesproken. De certificaten hebben de vorm van rechthoekige bedrukte papieren die *bankbiljetten* worden genoemd, of ronde metalen schijfjes die als *munten* bekend staan. Hun overdraagbaarheid is zo groot, dat Bob ze, in plaats van ze in geld te converteren bij de bank, best aan een derde kan overdragen.

Een derde betaalvorm, de *credit card*, waarbij creditering *voor* debetering plaats vindt, kunnen we als Amerikaans wanproduct in onze beter georganiseerde Nederlandse betaaleconomie buiten beschouwing laten. Bij het bestuderen van al dan niet electronische betaalsystemen zijn de volgende onderwerpen van belang.

Identificatie. In diverse stadia van het betalen is het noodzakelijk dat Alice, Bob, of de bank zich ondubbelzinnig identificeert. In het huidige model is dit nodig bij betaling met een PINpas, of bij opname van saldo certificaten bij een geldautomaat. Bij deze handelingen wordt Alice immers gedebeteerd.

In de huidige betaalsystemen is identificatie in feite het enige cruciale probleem. Men identificeert zich in de praktijk door overlegging van de PINpas, en het intypen van de bijbehorende PIN. De copiëerbaarheid van de pas, gecombineerd met deze erg primitieve methode om kennis van de PIN te bewijzen, maken voortzetting van het systeem in huidige vorm onmogelijk. Ondanks alle beweringen van de banken dat het systeem veilig is, wordt de fraude door allerlei vormen van misbruik onbetaalbaar voor de banken [And94].

Anonimiteit. De thans gebruikte papieren en metalen saldo-certificaten zijn niet persoonsgebonden in de zin dat ze door iedereen in saldo kunnen worden geconverteerd en door iedereen kunnen worden overgedragen. De metalen certificaten zijn daarnaast anoniem: ze zijn namelijk van elkaar ononderscheidbaar, zodat het niet mogelijk is aan de hand van

een betaling met munten na te gaan, wie de betaler is geweest of waar de certificaten vandaan kwamen. Menige gangster die meende dat ook bankbiljetten deze anonimiteit boden overdenkt zijn vergissing nu in detentie.

Sommige burgerrechtenorganisaties betogen, dat electronische betaalsystemen een volledige anonimiteit van de betaler zouden moeten bieden om de persoonlijke levenssfeer van de consument te beschermen. Zij krijgen steun vanuit de georganiseerde prostitutie en sectoren waar veel zwart geld wordt omgezet, omdat men voor omzetderving vreest wanneer de (op zich legale) transacties niet langer anoniem kunnen worden afgerekend. Justitie is minder enthousiast over de mogelijkheid ook illegale transacties (drugs en wapens) met het geld van de toekomst anoniem te verrekenen, of anoniem losgeld te innen. Het is daarom onwaarschijnlijk dat anonieme betaalprotocollen op grote schaal gebruikt zullen mogen worden, en de privacy van de burger wordt dan slechts beschermd door de convenanten van geheimhouding ondertekend door de banken. Zie [Sch96, Sec. 6.4] voor een discussie over anonimiteit in betaalsystemen.

Hardware of Software. Het uitvoeren van de cryptografische protocollen omvat bijvoorbeeld exponentiaties van 1000 bits getallen en gaat de hoofdrekencapaciteit van de gemiddelde Nederlander te boven. De bank zal daarom doorgaans aan clienten (consumenten en winkeliers) protocol-software verstrekken, of zelfs voorgeprogrammeerde rekencapaciteit beschikbaar stellen in de vorm van smartcards die de protocollen uitvoeren.

Worden de protocollen in software aangeboden en de gegevens op de gebruiker's disk opgeslagen, dan kan de gebruiker natuurlijk de programmatuur of de gegevens wijzigen. We spreken van een *software* systeem als de gebruiker van dergelijke manipulaties geen voordeel kan hebben. In sommige systemen is echter bedrog mogelijk als de gebruiker de gegevens of procedures in zijn smartcard vrijelijk kan wijzigen. De veiligheid van een dergelijk systeem vereist *onschendbaarheid* van de gebruikte hardware en we spreken dan van een *hardware* systeem.

Double Spender probleem. Saldo-certificaten die als een fysiek object (van papier of metaal) bestaan worden door hun natuur *overgedragen* en zijn na overdracht van Alice aan Bob niet langer in bezit van Alice. De fysieke vorm is doorgaans zo gekozen dat namaken of genereren van certificaten zeer moeilijk is; deze activiteit wordt daarnaast zeer zwaar bestraft.

Informatie-certificaten, echter, worden door hun natuur bij overdracht *gecopiëerd* en moeten dus bij de overdracht expliciet door Alice worden vernietigd of onbruikbaar gemaakt. Doet zij dit niet, dan kan zij met hetzelfde certificaat een tweede maal betalen en dit is het *double spender* probleem. De diverse betaalprotocollen kennen zeer uiteenlopende oplossingen voor dit probleem.

Fase een: Opname van een saldo-certificaten.

1. Alice identificeert zich bij de bank.
   Zij verzoekt opname van een b-certificaat.
2. De bank kiest een random r, en genereert en ondertekent $\langle\, \mathbf{sald}, b, r \,\rangle$.
3. Alice wordt gedebeteerd en ontvangt het certificaat y.
   Zij controleert dat y^e van de vorm $\langle\, \mathbf{sald}, b, r \,\rangle$ is.

Fase twee: Betaling aan Bob met het certificaat.

1. Alice koopt in Bob's winkel iets met prijs b.
2. Alice identificeert zich en wil betalen met getal y.
3. Bob controleert dat y^e van de vorm $\langle\, \mathbf{sald}, b, r \,\rangle$ is.
4. Bob genereert een random string u en geeft u aan Alice.
5. Alice draagt het certificaat aan Bob over door ondertekening van $\langle\, \mathbf{over}, r, u, Bob \rangle$.
6. Bob controleert de handtekening en accepteert de betaling.

Fase drie: Verzilvering.

1. Bob overlegt y aan de bank.
2. De bank controleert het certificaat, en gaat na of r al is verzilverd.
3. Bob wordt gecrediteerd.

Protocol 6: Een Elementair Betalingssysteem.

3.2 Digitaal Geld en Overdracht

Een eenvoudig, maar goed functionerend systeem maakt gebruik van genummerde saldo-certificaten en ontmoedigt double spenders door bij overdracht van een saldo-certificaat een overdrachts-certificaat te laten ondertekenen; zie Protocol 6.

De bank heeft een RSA sleutelpaar, waarvan $(n,\ e)$ het publieke deel is en dit is aan alle partijen bekend. Daarnaast heeft elke gebruiker een eigen RSA sleutelpaar (en een certificaat voor het publieke deel daarvan) voor handtekeningen en identificatie. Het saldo-certificaat is de bank's handtekening over het tuple $\langle\, \mathbf{sald}, b, r \,\rangle$, waar b de waarde van het certificaat en r het serienummer is. Het overdrachts-certificaat is Alice' handtekening over $\langle\, \mathbf{over}, r, u, X \,\rangle$, waar r het serienummer, u een unieke (random) string, en X de identiteit van de ontvanger is.

In dit systeem is double spending mogelijk, zodat we al snel een Netwerk of Nova uitzending zullen kunnen aanschouwen waarin "slimmerikken" laten zien hoe je een keer een tientje kunt opnemen en er twee keer mee kunt betalen. Wat men dan waarschijnlijk niet laat zien is hoe deze zaak door de bank wordt afgewikkeld.

Het saldo-certificaat wordt tweemaal ter verzilvering aangeboden, en in dat geval wordt van elke aanbieder het overdrachts-certificaat opgevraagd waarmee de aanbieder het eigendom van het certificaat heeft verkregen. Uiteindelijk stuit men op de persoon die het éénmaal heeft ontvangen maar tweemaal heeft uitgegeven. De twee ondertekende overdrachtscertificaten vormen een ondubbelzinnige schuldbekentenis van valsheid in geschrift

of valsemunterij en doen de dader snel in een penitentiaire inrichting belanden. Het double spenden is dus een eenvoudig traceerbare handeling, goed te vergelijken met het uitschrijven van ongedekte girobetaalkaarten, waarvan het risiko voor de banken acceptabel is. We moeten ons realiseren, dat men ook na doorgifte van een saldo-certificaat, het bijbehorende overdrachts-certificaat moet bewaren; het is immers een bewijs van onschuld in geval de betreffende munt later dubbel wordt verzilverd.

Verdere ontwikkelingen en vooruitzichten. Het Amsterdamse bedrijf DigiCash heeft een betaalsysteem op basis van saldo-certificaten geïmlementeerd en het product, Ecash, wordt al door diverse banken en Cybershops gebruikt [Dig96]. Ecash verschilt van het hier beschreven systeem doordat het anonimiteit biedt; een gebruiker die certificaten opneemt, maakt het serienummer zelf en laat het "ongezien" door de bank tekenen. Double spending is onmogelijk doordat betaling on-line is, waarbij tijdens de betaling wordt geverifieerd of het certificaat al eerder voor betaling is gebruikt. Een toekomstige implementatie zal anonieme betalingen *off-line* mogelijk maken.

De huidige implementaties worden gebruikt om betalingen over het Internet met PC's en servers te verrichten; smartcards zijn op het moment nog iets te beperkt voor certificaatsystemen, zoals uit een kleine berekening mag blijken. Voor voldoende veiligheid moeten RSA-moduli van tenminste 1024 bits worden gebruikt, waarmee ook de omvang van elk certificaat 1024 bits is. Een chipkaart met 16 kB geheugen kan dus ongeveer honderd certificaten (saldo èn overdrachts) bevatten, en het aantal betalingen dat kan worden gedaan totdat herladen met munten en het wegschrijven van overdrachtscertificaten nodig is, is dan te klein.

Verder moeten we bedenken, dat de banken een administratie moeten bijhouden van alle uitgegeven en geretourneerde saldo-certificaten. Bij invoering van de Euro als Europees betaalmiddel zullen *470 miljard* munten en biljetten in omloop worden gebracht; electronische implementatie met een bank-administratie van 0.5 kB per munt geeft een databank met een omvang van 235.000 GB. De grootste nu in gebruik zijnde databank heeft een omvang van ongeveer 20.000 GB.

3.3 Chipknip en Chipper

Alle kwaliteit kost geld: een smartcard die RSA berekeningen kan uitvoeren kost ongeveer 20 gulden, terwijl men voor 10 gulden al aan smartcards kan komen die DES versleutelingen kunnen uitvoeren en een geheugen van circa 8 kB hebben. De banken hebben, uit zuinigheid en de overweging "dat DES immers een sterk cryptografisch algoritme is", bij het ontwerp van het nieuwe electronische betaalsysteem gekozen voor een implementatie zonder certificaten en op basis van symmetrische cryptografie.

Implementatie van betaalfuncties. De plastic chipper/chipknip bevat een processor met geheugen die een oppervlakte van slechts enkele millimeters heeft en zich geheel onder de goudkleurige contacten bevindt. Voor het opnemen van saldo en het verrichten van

betalingen is identificatie vereist; dit gebeurt op basis van het symmetrische protocol 4. Het geld in de chip heeft niet de vorm van identificeerbare objecten (certificaten) maar is gerepresenteerd door een saldo-bedrag dat in het geheugen is opgeslagen.

Voor opname van geld is vereist, dat de kaart zich tegenover de opname-terminal identificeert, en dat de terminal zich tegenover de kaart identificeert als zijnde een echte terminal. Na identificatie wordt een bedrag van de bankrekening afgeschreven, waarna de kaart toestemming krijgt van de terminal om het interne saldo te verhogen. Bij betaling identificeert de kaart zich tegenover de betaalautomaat als een echte kaart, waarna het te betalen bedrag wordt afgeboekt van het saldo op de kaart, en bijgeboekt op het saldo in de betaalterminal.

Iedereen beheert zijn eigen chipknip; in feite betekent dit dat iedereen zijn eigen saldo bijhoudt, en de deur staat wijd open voor mogelijk misbruik. Het saldo wordt bijgehouden in de kaart, en de primaire beveiliging tegen het smokkelen met het saldo is dan ook niet cryptografisch, maar wordt verkregen door de (vermeende) fysieke onschendbaarheid van de kaart. Handige knutselaars maken hun kaart open, modificeren de chip zo dat geen betalingen meer worden afgeboekt, en bouwen de chip weer in in een plastic omhulling [AK96]. Men verkrijgt zo een kaart die altijd blijft betalen en nooit geladen hoeft te worden. Wel identificeert de kaart zich bij elke betaling, en uit de afschriften van deze betalingen zal de bank kunnen opmaken dat de kaart gemodificeerd is.

Implementatie van de Identificatie. Om deze fraude ontraceerbaar uit te voeren moet je de kaart ook zo maken dat hij zich valselijk als een andere kaart kan identificeren; hiertoe is ook de identificatie van een lamentabel veiligheidsniveau gemaakt. Immers, voor identificatie moeten de terminal en de kaart over een gemeenschappelijk geheim password beschikken, maar geen terminal kan alle passwords bevatten.

Daarom is er een master-key M, waarmee de geheime sleutel k van pas nummer x wordt berekend als $k = E_M(x)$; de kaart bevat de master-key zelf niet, maar is bij fabricage van x en de bijpassende k voorzien. De terminals bevatten de master-key wel. Een kaart meldt zich met het nummer x, waarna de terminal het bijbehorende geheime password k uitrekent met de master-key. Een wederzijdse identificatie (volgens Prot. 4) op basis van de geheime k levert dus (1) voor de kaart het bewijs dat zijn tegenpartij M kent, en dus een echte terminal is; en (2) voor de terminal het bewijs dat de kaart niet alleen het getal x kent, maar ook de bijbehorende k en dus een echte kaart is die gemaakt is met behulp van M.

Dit mechanisme biedt de mogelijkheid "echte" kaarten te maken zodra men over M beschikt en dit strategisch cruciale geheim bevindt zich in elke terminal! Zou men nu nog hopen dat het voor gangsters moeilijk is een terminal te stelen (een doorgaans omvangrijk apparaat immers), helaas. Het geheim M bevindt zich "om veiligheidsredenen" op een kleine smartcard die in de terminal moet worden gestoken, en er dus ook snel door dieven uit is te verwijderen. Een beetje crimineel heeft trouwens zelf een "respectabele" onderneming zoals een restaurant of café en zal de terminals dus erg gemakkelijk van de banken krijgen.

De apparatuur die nodig is om M uit de terminals te lezen en chipkaarten te herinrichten om zich als een andere kaart valselijk te identificeren staat niet in elke huiskamer. Er

is een investering van enkele tonnen voor nodig, maar in Nederland bevinden zich een tiental universitaire en industriële laboratoria en werkplaatsen waar deze chip-chirurgie uitvoerbaar is.

4 Slotopmerkingen

We hebben een korte studie gemaakt van symmetrische en asymmetrische cryptografische algoritmen en hun toepassingen in protocollen, met name in electronisch betaalverkeer.

Onder de thans wijd gebruikte algoritmen bevinden zich geen slechte algoritmen: DES, IDEA, 3DES, RSA en ElGamal zijn cryptografisch zeer sterk. Het verschil tussen symmetrische en asymmetrische algoritmen is echter zo groot, dan men moet stellen dat zij eigenlijk verschillende problemen te lijf kunnen. Symmetrische algoritmen zijn erg geschikt voor het beveiligen van data tijdens transport en opslag. Asymmetrische algoritmen kunnen goed worden toegepast in cryptografische protocollen, bijvoorbeeld voor sleutel-distributie, handtekeningen, idententificatie. Een weldoordacht systeem herkent deze verschillende functies en gebruikt de cryptografische algoritmen die voor elke functie het geschiktst zijn, bijvoorbeeld PGP.

Electronische betaalsystemen bestaan al voor toepassing op PCs en betaling over het Internet, bijvoorbeeld Ecash. Voor grootschalige toepassing op smartcards is het nog net iets te vroeg: de benodigde chips met RSA-functies zijn nog iets te duur en hebben nog iets te weinig opslagcapaciteit, een situatie die binnen enkele jaren zal veranderen. Het thans ingevoerde chipper/chipknip systeem is gebaseerd op de foute veronderstelling dat symmetrische cryptografie een goedkope variant van de asymmetrische is. Het systeem staat nu al onder kritiek en zal binnen enkele jaren weer verdwijnen, overigens zonder dat de banken de onveiligheid ervan toegeven, want dat doen ze nooit.

Referenties

[AK96] Anderson, R. and Kuhn, M. Tamper resistance – a cautionary note. 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.

[Dig96] DigiCash. An introduction to ecash. URL `http://www.digicash.nl/publish/ecash_intro/ecash_intro.html`, 1996.

[Sch96] Schneier, B. *Applied Cryptography*, 2nd edition. Wiley, 1996.

[Sti95] Stinson, D. R. *Cryptography: Theory and Practice*. CRC Press, 1995.

[Vri96] Vries, M. de. Cryptography and electronic payment systems. Scriptie INF/SCR-96-23, Dept Computer Science, Utrecht University, The Netherlands, 1996.

Inhoudsopgave