

Cryptografie

Olivier Tieleman en Jelmer Vernooij

2002

Inhoudsopgave

1	Inleiding	3
1.1	Wat is cryptografie?	3
1.2	Oud	3
1.3	Hedendaagse toepassing	3
1.4	Computerwetenschap	3
1.5	Opzet werkstuk	4
2	Terminologie	5
2.1	Verschillende soorten cryptografie	5
2.1.1	Symmetrische cryptografie	5
2.1.2	Asymmetrische cryptografie	6
2.2	Waarom is wiskunde zo belangrijk?	6
3	Symmetrische cryptografie	7
3.1	Het Caesar cijfer	7
3.1.1	Voorbeeld van gebruik van Caesar algoritme	7
3.2	Enkelvoudige substitutie	7
3.2.1	Voorbeeld van enkelvoudige substitutie	9
3.3	Kraken van algoritmes	9
3.3.1	Kraken van enkelvoudige substitutie	9
3.3.2	Voorbeeld van kraken van enkelvoudige substitutie	10
3.3.3	De kraakbaarheid van een algoritme	10
3.3.4	Kraken algoritme voorkomen	11
3.4	Het Vigenère systeem	11
3.4.1	Voorbeeld van het Vigenère systeem	12
3.5	Kraken van het Vigenère systeem	12
3.6	Enigma	12
3.7	Blokencryptie	14
3.7.1	DES	14
3.7.2	XOR	14
3.7.3	Feistel-netwerk	14
3.7.4	P-boxen en S-boxen	15
3.7.5	3 DES	16
4	Wiskundige principes	17
4.1	Modulair rekenen	17
4.2	Priemgetallen	17
4.3	Inverse modulo en relatieve priemgetallen	18
4.3.1	Voorbeeld inverse modulo en relatieve priemgetallen	18
4.4	Algoritme van Euclides	18
4.4.1	Bewijs	19
4.4.2	Voorbeeld gebruik algoritme van Euclides	19
4.5	Euler's phi functie	19
4.6	Euler & Fermat	19
4.7	Discrete logaritme	20
4.7.1	Voorbeeld inverse modulo	20
4.7.2	Voorbeeld inverse modulo zonder oplossing	20
4.8	Factoren vinden	21
4.8.1	Voorbeeld vinden factoren	21
4.9	Priemgeneratoren	21

4.9.1	Voorbeeld Rabin-Miller-test Priem	22
4.9.2	Voorbeeld Rabin-Miller-test Niet-Priem	22
5	Asymmetrische cryptografie	23
5.1	Public-key cryptografie	23
5.2	RSA	23
5.2.1	Werking RSA	23
5.2.2	Voorbeeld RSA	24
5.3	Veiligheid RSA	24
5.4	Veiligheid public-key cryptografie	25
5.5	Sleutels genereren	25
5.5.1	Voorbeeld van het genereren van sleutels	25
6	Toepassing van cryptografie	26
6.1	Computersystemen	26
6.1.1	Hashes	26
6.1.2	Pretty Good Privacy	26
6.1.3	Voorbeeld van een PGP handtekening	26
6.2	Mobiele Telefoon	27
6.3	Smartcards	27
6.3.1	Controle identiteit	27
6.3.2	Fraudebestendig	28
7	Politiek	29
7.1	Exportwetten in de Verenigde Staten	29
7.2	Discussie over vrijgeven sleutels door ttp's	29
7.3	DMCA	29
8	Conclusie	31
A	Logboek	33

Hoofdstuk 1

Inleiding

1.1 Wat is cryptografie?

Cryptografie komt van twee Griekse woorden: ‘κρυπτος’ en ‘γραφειν’. Deze woorden betekenen respectievelijk ‘verborgen, geheim’ en ‘schrijven’. Cryptografie betekent dus letterlijk: ‘geheimschrijven’. Hiervan kan de volgende definitie worden afgeleid:

“Cryptografie is de wetenschap die zich bezighoudt met het versleutelen en ontcijferen van al dan niet versleutelde informatie.”

Dit is een tamelijk wijd gebied, teveel om allemaal te behandelen in dit werkstuk. Daarom zal hier vooral worden ingegaan op de computertoepassingen van cryptografie en de bijbehorende technieken.

1.2 Oud

Cryptografie is heel erg oud: het bestaat vanaf het moment dat mensen berichten aan elkaar begonnen door te geven die niet voor andere oren dan die van de geadresseerde bestemd waren. Als men het erg ruim ziet, kan gesteld worden dat cryptografie is begonnen met het geven van signalen door oermensen bij de jacht: die waren alleen bestemd voor de stamgenoten, en niet voor het bejaagde wild. Dit is natuurlijk overdreven, zeker als men kijkt naar hedendaagse cryptografie: die heeft met de jacht weinig meer te maken. Daarom wordt meestal het begin van de geschiedenis van de cryptografie geplaatst bij Caesar ¹ : van hem is het oudste bekende systeem afkomstig.

1.3 Hedendaagse toepassing

In de huidige samenleving neemt het gebruik van elektronische communicatie zeer sterk toe. Voorbeelden zijn: internet, e-mail, elektronische betalen en (mobiele) telefonie. Al deze communicatiesystemen verzenden en ontvangen elektrische signalen. Dit zijn veelal signalen waarvan men liever niet heeft dat iedereen ze mee kan lezen of horen. Er zijn twee mogelijkheden om te zorgen dat iemand de betekenis van een signaal niet te weten komt: men zorgt ervoor dat hij/zij het signaal niet opmerkt, of men zorgt ervoor dat hij/zij het signaal niet begrijpt. Aangezien het eerste in de hierboven beschreven communicatie praktisch onmogelijk is, probeert men het tweede. Dat gebeurt door middel van cryptografie: deze wetenschap is, zoals hierboven gezegd, geheel op dit probleem gericht.

1.4 Computerwetenschap

Dit werkstuk zal, zoals gezegd, voornamelijk gaan over computercryptografie. Dat heeft als gevolg, dat er veel gebruik zal worden gemaakt van wiskundige bewerkingen: computers zijn immers niet veel meer dan sterk veredelde rekenmachines. Daarom zijn ze erg goed in het snel uitvoeren van wiskundige bewerkingen, maar ook niet in veel anders. Dat heeft

¹Zie ook hoofdstuk 3: ‘Symmetrische cryptografie’

tot gevolg, dat computers alle informatie zien als een getal, of een reeks getallen: alleen dan kunnen ze ermee uit de voeten.

Omdat alle informatie wordt behandeld als een getal of reeks, wordt bij het versleutelen ervan ook gewerkt met simpele wiskundige bewerkingen als optellen, aftrekken, vermenigvuldigen, delen, modulo, kwadrateren of worteltrekken². Dit heeft ook tot gevolg, dat iets dat door een computer versleuteld is, in bijna alle gevallen ook weer door een computer kan worden ontcijferd.

Daarmee zijn we aangekomen bij de belangrijkste bezigheid van de huidige cryptografische wetenschap: het verzinnen van algoritmes die niet of moeilijk te breken zijn (algoritmes die niet te breken zijn zijn bekend, maar werken niet praktisch).

1.5 Opzet werkstuk

In dit werkstuk zullen de volgende deelonderwerpen worden behandeld:

- symmetrische cryptografie (hoofdstuk 3)
- wiskundige principes waarop cryptografie gebaseerd is (hoofdstuk 4)
- asymmetrische cryptografie (hoofdstuk 5)
- enige toepassingen van cryptografie (hoofdstuk 6)
- politiek met betrekking tot cryptografie (hoofdstuk 7)

Voor lezers die minder bekend zijn met het fenomeen cryptografie zal een hoofdstuk over de terminologie worden toegevoegd, aangezien de schrijvers tijdens hun onderzoek merkten dat dit noodzakelijk is voor goed begrip van de bovengenoemde onderwerpen.

Nu rest niets dan de lezer veel plezier te wensen bij het lezen van dit werkstuk.

²Zie hoofdstuk 4: ‘Wiskundige principes’

Hoofdstuk 2

Terminologie

Cryptografie wordt gebruikt om informatie alleen leesbaar te maken voor diegenen die het mogen lezen of om de echtheid van bepaalde informatie aan te tonen. In de simpelste gevallen gaat het er om dat een *zender* een *bericht* wil sturen aan één of meerdere *ontvangers* zonder dat een ander dat bericht kan lezen.

Doel is dus om een begrijpbaar bericht (de *klare tekst* of *plaintext*) zo te veranderen dat heel erg moeilijk te herleiden is wat het oorspronkelijke bericht was. Dit proces heet *vercijferen* (Engels: *encrypt*). Na het vercijferen van een bericht houdt men de *cijfertekst* (vaak wordt het Engelse *ciphertext* gebruikt) over. Of in wiskundige notatie:

$$E(P) = C \tag{2.1}$$

Hierin is E de vercijferfunctie, P de plaintext en C de ciphertext.

De ontvanger moet vervolgens het oorspronkelijke bericht, de plaintext, kunnen herleiden uit de ciphertext. Dit proces heet *ontcijferen* (Engels: *decrypt*). In de wiskundige notatie:

$$D(C) = P \tag{2.2}$$

Hierin is D de decryptiefunctie, P de plaintext en C de ciphertext.

Een *cryptografisch algoritme* is een wiskundige functie die gebruikt wordt voor het vercijferen en ontcijferen. Zogenaamde *restricted algoritmes* zijn gebaseerd op geheimhouding van het algoritme. Wanneer de werking van zo'n algoritme bekend wordt is het plotseling mogelijk om alle met dat algoritme gecodeerde berichten te decrypten. Daarnaast zijn dit soort algoritmes niet voor gebruik op grote schaal inzetbaar — meerdere partijen zouden de werking van het algoritme moeten weten.

Daarom maken de meeste algoritmes gebruik van *sleutels*. Door het algoritme te voorzien van een sleutel naast de te vercijferen tekst is het niet mogelijk de tekst te achterhalen wanneer het algoritme bekend is. Om de ciphertext dan te kunnen decrypten moet naast het algoritme ook de sleutel bekend zijn.

2.1 Verschillende soorten cryptografie

Er zijn twee duidelijk verschillende vormen van cryptografie te onderscheiden: asymmetrisch en symmetrisch.

2.1.1 Symmetrische cryptografie

Bij symmetrische algoritmes gebruikt men dezelfde sleutel voor vercijfering en voor ontcijfering. In wiskundige notatie:

$$E_k(P) = C \tag{2.3}$$

$$D_k(C) = P \tag{2.4}$$

En dus geldt ook:

$$D_k(E_k(P)) = P \tag{2.5}$$

2.1.2 Asymmetrische cryptografie

Bij asymmetrische cryptografie gebruikt men twee verschillende sleutels. De ene is “publiek” — beschikbaar voor iedereen, en de andere is “privaat” — alleen de eigenaar van de sleutel kent deze. Wanneer iemand een bericht wil sturen aan iemand anders hoeft deze alleen maar het bericht te vercijferen met de sleutel van die ander en kan alleen die ander het decrypten. In wiskundige notatie:

$$E_{k_1}(P) = C \quad (2.6)$$

$$D_{k_2}(C) = P \quad (2.7)$$

En dus geldt ook:

$$D_{k_2}(E_{k_1}(P)) = P \quad (2.8)$$

2.2 Waarom is wiskunde zo belangrijk?

Belangrijkste probleem in de cryptografie zijn de decryptie en vercijferfuncties. Het moeten functies zijn waarmee het heel moeilijk is de invoer uit de uitvoer af te leiden, terwijl het met behulp van de sleutel wel mogelijk moet blijven. Daarnaast geldt voor asymmetrische cryptografie dat een tekst die met de ene sleutel vercijferd is met een geheel andere sleutel ontcijferd kan worden, iets wat alleen dankzij de wiskunde achter de cryptografie mogelijk is.

Hoofdstuk 3

Symmetrische cryptografie

3.1 Het Caesar cijfer

Zoals in de inleiding al geschreven werd, was de Romeinse bevelhebber Caesar eigenlijk de eerste die op vrij grote schaal gebruik maakte van cryptografie. Caesar stuurde bodes op pad met versleutelde berichten, zodat, wanneer deze bodes in de handen van de vijand zouden vallen, de boodschap nog steeds niet zou weten.

Het Caesar algoritme is een van de simpelste vormen van vercijfering en werkt als volgt:

Van een bepaalde boodschap wordt elke letter k plaatsen in het alfabet opgeschoven. Bij alle letters die na die optelling 'buiten het alfabet' vallen wordt 'gemoduleerd' met 26 — de rest die overblijft na het delen van het getal door 26.

De schijf in figuur 3.1 is in 1466 gemaakt door de Italiaan Alberti en kan gebruikt worden om het Caesar algoritme toe te passen.¹

3.1.1 Voorbeeld van gebruik van Caesar algoritme

Stel dat je het bericht WISKUNDE hebt en $k = 4$

Dan verstuurt men de boodschap AMWOYRHI

Wanneer iemand de theorie achter dit algoritme te weten komt is het kraken van een boodschap triviaal, zelfs zonder een computer. $1 \leq k \leq 26$, aangezien $k = 26 + x$ hetzelfde resultaat geeft als $k = x$. Iemand die het algoritme kent hoeft dus maar 26 mogelijke oplossingen te proberen om de goede oplossing te weten te komen. In Caesar's tijd berustte de kracht van dit systeem dan ook vooral in de geheimhouding van hoe het werkte.

Om het anders te zeggen: er zijn maar 26 verschillende sleutels mogelijk, en een sleutel (een getal k) is maar 1 letter lang.

Het Caesar algoritme met $k = 13$ is een bijzondere vorm — $P = E_k(E_k(P)) = D_k(D_k(P))$. Deze versie staat ook wel bekend als *ROT13*.

3.2 Enkelvoudige substitutie

Een iets uitgebreidere variant van het Caesar algoritme is *enkelvoudige substitutie*. In dit geval wordt niet bij elke letter k opgeteld, maar wordt elke letter door een andere vervangen. Hierdoor wordt de benodigde sleutellengte al een heel stuk langer — 26 om precies te zijn, voor elke letter moet namelijk bekend zijn door welke deze vervangen moet worden.

Iemand die een boodschap wil ontcijferen, moet nu namelijk $26!$ verschillende sleutels uitproberen. Elke letter mag namelijk maar door één andere letter vervangen worden, anders wordt de versleutelde boodschap op meerdere manieren interpreteerbaar.

Een 'a' kan door 'a' tot 'z' vervangen worden, dus 26 mogelijkheden.

¹<http://www.voov.nl/284spkrenigm.html>



Figuur 3.1: Chifreerschijf

Een 'b' kan door 'a' tot 'z' vervangen worden, behalve de letter waardoor 'a' vervangen wordt. 25 mogelijkheden dus.

Een 'c' kan door 'a' tot 'z' vervangen worden, behalve door de letters waardoor 'a' en 'b' vervangen worden. 24 mogelijkheden dus.

⋮

Een 'z' kan door nog maar 1 letter vervangen worden (de andere letters uit het alfabet worden immers vervangen door de andere 25 letters).

Hieruit volgt dat er $26 \cdot 25 \cdot 24 \cdot 23 \cdot 22 \cdot 21 \cdot \dots \cdot 2 \cdot 1 = 26!$ verschillende sleutels zijn.

3.2.1 Voorbeeld van enkelvoudige substitutie

Wanneer men de boodschap DIT IS ONS PROFIELWERKSTUK EN HET GAAT OVER CRYPTOGRAFIE wil versleutelen met de volgende sleutel:

a	b	c	d	e	f	g	h	i	j	k	l	m
r	m	e	h	w	x	z	y	g	i	j	a	c
n	o	p	q	r	s	t	u	v	w	x	y	z
q	u	p	d	b	s	v	t	o	n	l	f	k

Dan gebeurt dat als volgt:

Elke letter uit de boodschap wordt vervangen door de bijbehorende letter in de tabel.

In het voorbeeld :

$$D \rightarrow H \quad (3.1)$$

$$I \rightarrow G \quad (3.2)$$

$$T \rightarrow V \quad (3.3)$$

⋮

$$I \rightarrow G \quad (3.4)$$

$$E \rightarrow W \quad (3.5)$$

Uiteindelijk wordt de versleutelde boodschap dus:

HGV GS UQS PBUXGWANWBJSVTJ WQ YWV ZRRV UOWB EBFPUVZBRXGW

3.3 Kraken van algoritmes

3.3.1 Kraken van enkelvoudige substitutie

Er zijn echter nog meer manieren dan door ‘brute-force’ een algoritme kraken. Elke natuurlijke taal bevat herhaling en ook komen sommige letters vaker voor dan andere. In het bovenstaande geval met de enkelvoudige substitutie lijkt het immers wel of we $26!$ sleutels moeten uitproberen om de uiteindelijke sleutel te krijgen, maar we kunnen door middel van taalanalyse meer te weten komen. Sommige letters komen namelijk vaker voor dan andere.

Zie tabel 3.3.1 voor een overzicht van de verhoudingen waarin letters gemiddeld voorkomen in taal. Wanneer men maar genoeg ‘ciphertext’ te pakken weet te krijgen dat gecijferd is met enkelvoudige substitutie, kan men ook van dit bericht de verhoudingen berekenen en is zo de sleutel te bepalen.

Tabel 3.1: Verhoudingen waarin letters gemiddeld in de taal voorkomen[3]

Letter	Verhouding
a	0,0804
b	0,0154
c	0,0306
d	0,0399
e	0,1251
f	0,0230
g	0,0196
h	0,0549
i	0,0726
j	0,0016
k	0,0067
l	0,0414
m	0,0253
n	0,0709
o	0,0760
p	0,0200
q	0,0011
r	0,0612
s	0,0654
t	0,0925
u	0,0271
v	0,0099
w	0,0192
x	0,0019
y	0,0173
z	0,0009

3.3.2 Voorbeeld van kraken van enkelvoudige substitutie

Om het volgende bericht te decoderen volgens de hierboven uitgelegde procedure, moeten de verhoudingen waarin de letters voorkomen bepaald worden.

Het bericht is HGV GS UQS PBUXGWANWBJSVTJ WQ YWV ZRRV UOWB EBFVUZBRXGW.

a	b	c	d	e	f	g	h	i	j
1	5	0	0	1	1	4	1	0	2
0,0208	0,1042	0	0	0,0208	0,0208	0,0833	0,0208	0	0,0416

k	l	m	n	o	p	q	r	s
0	0	0	1	1	2	2	3	3
0	0	0	0,0208	0,0208	0,0416	0,0416	0,0625	0,0625

t	u	v	w	x	y	z
1	4	5	6	2	1	2
0,0208	0,0833	0,1042	0,1250	0,0416	0,0208	0,0416

Aan de hand van deze tabel en tabel 3.3.1 is al gauw te zien dat de *W* voor een *E* staat, de *V* voor een *T*, de *B*, *G*, *S* en *T* voor een *R*, *I*, *S* of *O*.

De resultaten van een kort bericht(48 in dit geval) zijn voor deze methode nog niet zo fantastisch, maar zodra er meer ciphertext bekend is met een bepaalde sleutel komen de verhoudingen steeds dichterbij die uit tabel 3.3.1

3.3.3 De kraakbaarheid van een algoritme

Het aantal bits dat nodig is om een bepaald bericht te versturen heet de *entropie*. De entropie is te berekenen met de volgende formule:

$$H(B) = 2 \log(L)$$

Waarin H de entropie is, B het bericht en L het aantal mogelijke manieren waarop een bericht uit te leggen is. In dit geval gebruiken we ${}^2\log$ aangezien we werken met blokken van 1 bit. In een taal zou dit ${}^{26}\log$ zijn.

Immers, wanneer er bijvoorbeeld 7 mogelijke berichten zijn, zijn er 3 bits nodig om het bericht te versturen.

Daarnaast is er nog het begrip *ratio*. De ratio geeft aan hoe ‘uniek’ de gegevens in een bepaalde tekst zijn. Hoe minder mogelijke aantallen manieren van uitleggen van een bepaald bericht, hoe lager de ratio. In wiskundige notatie wordt de ratio als volgt genoteerd:

$$r = \frac{H(B)}{N}$$

Waarin r de ratio is, H de entropie en N de lengte van het bericht.

In een (voor cryptografie) perfecte taal zou de ratio van een taal maximaal zijn en dat betekent dus dat het aantal mogelijke manieren waarop een bericht uit te leggen is gelijk is aan het aantal berichten. Dat betekent dus dat de entropie gelijk moet zijn aan het aantal verstuurde bits. Deze ‘perfecte’ ratio wordt de *absolute ratio*(R) genoemd.

De *redundantie* geeft aan hoeveel ‘herhaling’ er voorkomt in een bepaald stuk bericht. De *redundantie*(D) is wiskundig als volgt te noteren:

$$D = R - r$$

De herhaling die voorkomt is dus het aantal mogelijke berichten min het aantal mogelijke manieren van uitleggen van een bericht.

Bij een voor encryptie perfecte taal is de *redundantie* 0, aangezien er geen (bekende) herhaling in voorkomt. R is dan dus gelijk aan r .

3.3.4 Kraken algoritme voorkomen

Doel zal altijd zijn de *redundantie* zo laag mogelijk te houden, aangezien *redundantie* het makkelijker maakt om berichten te kraken.

Er moet dus gezorgd worden dat het gebruikte algoritme:

- niet gekraakt kan worden door alle mogelijke sleutels uit te proberen; het algoritme moet *computationeel* veilig zijn.
- niet gekraakt kan worden door analyse van de versleutelde tekst; het algoritme moet *informatietheoretisch* veilig zijn.

Een algoritme kan *computationeel* sterker gemaakt worden door de lengte van de sleutel langer te maken — en daarmee het aantal mogelijke sleutels. In theorie zou de perfecte sleutel een sleutel zijn die even lang is als de plaintext — de ciphertext is dan immers op alle mogelijke manieren uit te leggen! Wanneer er sprake is van een sleutel die even lang is als de plaintext wordt gesproken van een *one-time pad*.

Moderne computers kunnen een paar miljoen tot een paar miljard sleutels per seconde testen, vandaar dat de meeste sleutels tegenwoordig minstens 512 bit lang zijn.

3.4 Het Vigenère systeem

Een andere techniek die gebruikt wordt om het Caesar algoritme lastiger kraakbaar te maken is het Vigenère systeem. Het systeem is bedacht door Blaise de Vigenère.

In dit systeem is de k niet constant, maar laat men de k variëren volgens een bepaald sleutelwoord. Men telt dan olopend één van de letters van het sleutelwoord op bij de volgende letter in de sleuteltekst.

3.4.1 Voorbeeld van het Vigenère systeem

Een boodschap WISKUNDE moet met het Vigenère systeem versleuteld worden met de sleutel CGU. Om te vercijferen ‘tellen’ we de twee woorden bij elkaar op, de sleutel even vaak herhalend als nodig is.

W	I	S	K	U	N	D	E
C	G	U	C	G	U	C	G +
Z	P	N	N	B	I	G	L

De ontvanger van dit versleutelde bericht kan het als volgt lezen (mits de sleutel bekend is):

Z	P	N	N	B	I	G	L
C	G	U	C	G	U	C	G -
W	I	S	K	U	N	D	E

3.5 Kraken van het Vigenère systeem

Dit algoritme is al moeilijker te kraken dan enkelvoudige substitutie, maar het is nog steeds mogelijk. Hoe langer de sleutel, hoe moeilijker het is om de tekst te kraken.

Moeilijkste deel van het achterhalen van de plaintext van een met het Vigenère systeem vercijferde boodschap is het achterhalen van de lengte van de sleutel. Nadat de lengte van de sleutel bekend is, is het immers bekend welke tekens met dezelfde sleutel versleuteld zijn en kun je aan de hand daarvan met de hierboven besproken kraakprocedure achterhalen wat de sleutels en de ciphertext was.

In een taal waarin alle letters in volstrekt gelijke verhoudingen voorkomen zou de kans dat twee letters die je in een ciphertext tegenkomt gelijk zijn precies $\frac{1}{26}$ zijn. In de werkelijkheid komen bepaalde letters echter vaker voor dan andere (zie ook tabel 3.3.1). Hier volgt uit dat de kans dat twee letters in een ciphertext hetzelfde zijn gelijk is aan de kans dat ze beiden gelijk zijn aan $a +$ de kans dat ze beiden gelijk zijn aan $b +$ etc.

Wiskundig genoteerd:

$$p(a)^2 + p(b)^2 + p(c)^2 + p(d)^2 + \dots + p(z)^2 \quad (3.6)$$

De Vigenère code werd in 1863 gekraakt door Majoor Kasiski uit Pruisen. Hij bepaalde de lengte van de sleutel door te letten op combinaties van letters achter elkaar die regelmatig voorkwamen. Bepaalde letterparen (zoals ‘en’ en ‘de’ in het Nederlands) komen in een taal vaker voor dan andere. Door bij te houden met welke frequentie zulke herhalende letterparen voorkwamen kwam hij te weten wat de sleutellengte was. Vervolgens is het eerste sleutelteken te bepalen door de kraakprocedure voor enkelvoudige substitutie toe te passen op het eerste, het k -de, het $2 \cdot k$ -de, \dots , en evenzo voor het tweede, het $(k + 1)$ -de, het $2 \cdot (k + 2)$ -de, \dots , enzovoort.

3.6 Enigma

Tijdens oorlogen is het versturen van berichten zonder dat de vijand deze kan lezen erg belangrijk. In de Tweede Wereldoorlog gebruikten de Duitsers hun zogenaamde “Enigma” machine om berichten te coderen alvorens ze te versturen.

Engeland heeft veel geld en mankracht geïnvesteerd in het kraken van de enigma machine. Aanvankelijk kreeg men een aantal machines in handen die de Duitsers gebruikten voor encryptie, maar later begonnen de Duitsers andere algoritmes te gebruiken, onder andere door een extra rotor toe te voegen aan het systeem dat ze voorheen hadden gebruikt. [2]

De machine maakt gebruik van symmetrische cryptografie. Elke rotor voert enkelvoudige substitutie uit. Elke invoerpin is verbonden met een willekeurige uitvoerpin. Wanneer er bijvoorbeeld een A ingetoetst wordt wordt er stroom gezet op de bovenste invoerpin en daardoor op een bepaalde uitvoerpin, die weer verbonden is met een invoerpin van de tweede rotor, etcetera.

Na elke toetsaanslag wordt de meest rechtse rotor 1 positie gedraaid, waardoor elke letter steeds een andere betekenis krijgt.

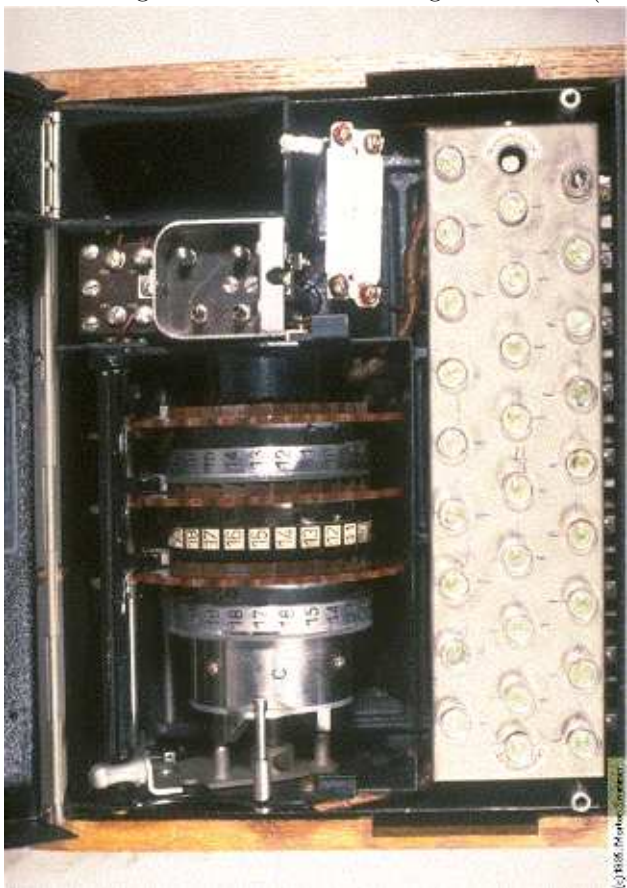
Nadat de meest rechtse rotor een keer rond gedraaid is verschuift de rotor links ervan 1 positie, en zo voort.

Omdat er 3 rotoren zijn met elk 26 letters wordt de uitvoer pas na $26^3 = 17576$ keer herhaald.

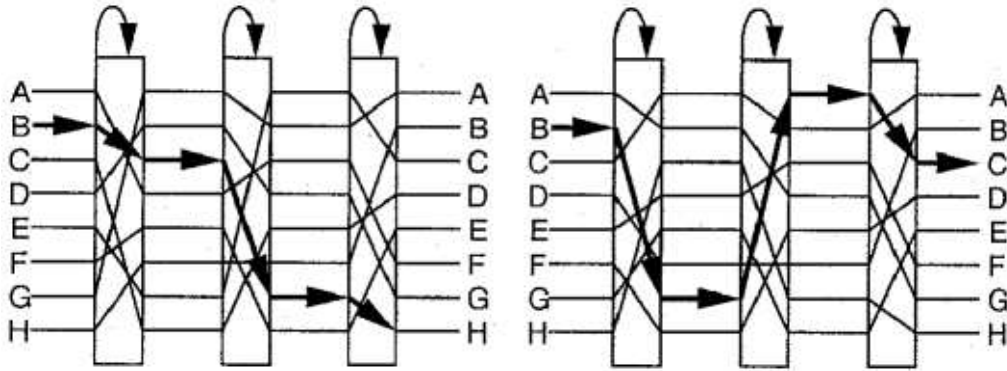
Figuur 3.2: Foto van de rotors van de enigma-machine (<http://www.math.arizona.edu/dsl/images/enigma50.gif>)



Figuur 3.3: Foto van de enigma machine (<http://www.math.arizona.edu/dsl/images/enigma34.gif>)



Figuur 3.4: De stand van de enigma bij het 2 keer intoetsen van een B
<http://studwww.rug.ac.be/kboodts/CRYPTANALYSE/crypt7.htm>



Tabel 3.2: XOR waarden

XOR	0	1
0	0	1
1	1	0

3.7 Blokcryptie

De vorige besproken algoritmes werken elk op maar 1 teken, maar er zijn ook algoritmes die werken op meerdere tekens, hele blokken data in één keer.

3.7.1 DES

DES is het meestgebruikte blokalgoritme. Dat komt vooral omdat het vrijwel het enige algoritme is dat vrij in en uit de VS ge-exporteerd mag worden. Hoewel het algoritme al enige tijd geleden gekraakt is wordt het nog steeds op vrij grote schaal toegepast in allerlei producten. DES is gebaseerd op het in de jaren '70 ontwikkelde algoritme "LUCIFER" van IBM en ontwikkeld door het NIST, een Amerikaans technologie instituut.

DES werkt met sleutels van 64 bits, maar elke 8^e bit wordt gebruikt ter controle. Dat betekent dus dat er $2^{64-8} = 2^{56} = 72057594037927936$ mogelijke sleutels zijn.

DES werkt, in tegenstelling tot de voorgaande besproken algoritmes niet door de sleutel 'op te tellen' bij de plaintext.

DES werkt op blokken van 64 bits.

3.7.2 XOR

DES werkt niet met optellen of aftrekken, maar met XOR(\otimes). de XOR van twee waarden is 1 als precies 1 van beiden gelijk is aan 1. Wanneer beiden gelijk zijn aan 0 of beiden gelijk zijn aan 1 is de XOR van deze twee waarden 0. Zie de tabel "XOR-waarden".

3.7.3 Feistel-netwerk

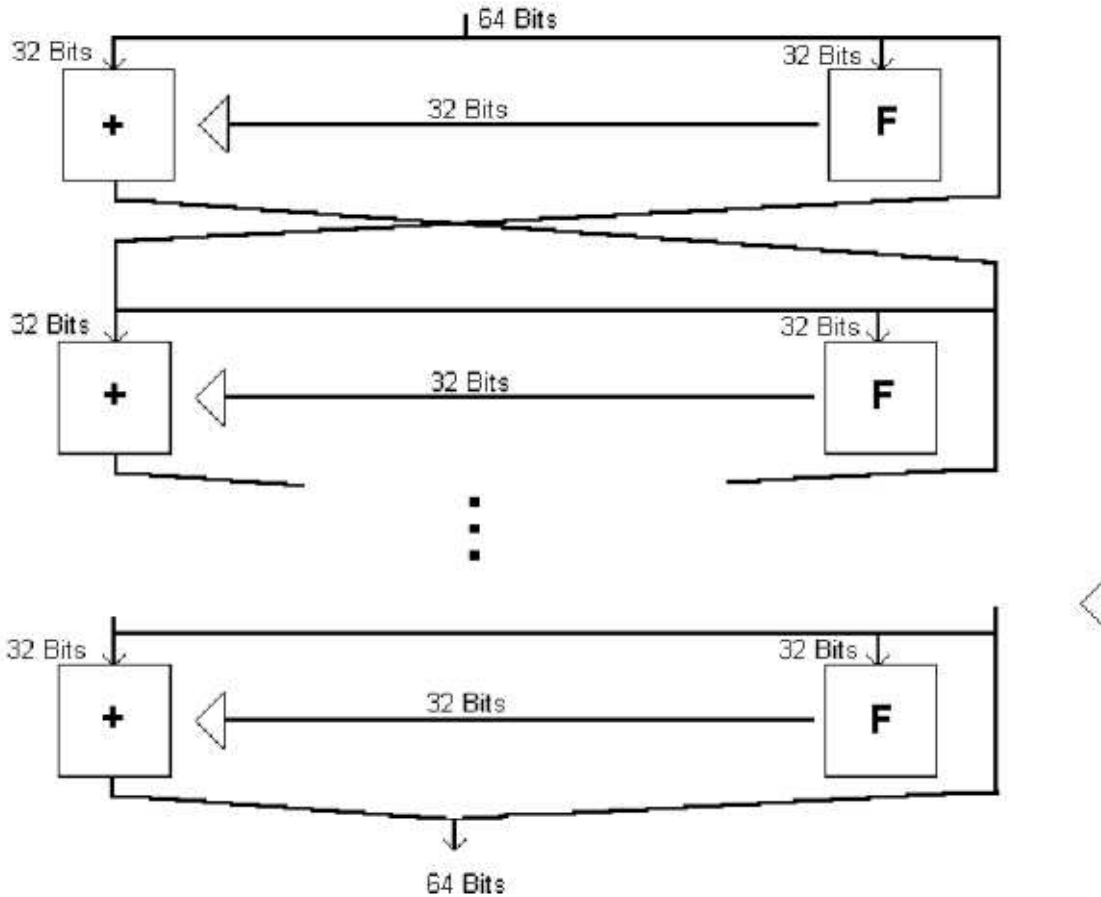
DES is een algoritme dat gebruik maakt van een zogenaamd "Feistel Netwerk".

DES werkt in zestien rondes waarbij in elke ronde steeds de beide helften van het blok verwisseld wordt en waarbij op de linker encryptie toegepast wordt:

$$L_i = R_{i-1} \tag{3.7}$$

Voor de rechterkant gebruikt men een functie van een 'Feistel-netwerk', deze gebruikt een sleutel en een deel van het blok zelf!:

Figuur 3.5: Uitvoeren van DES



Tabel 3.3: Opsplitsing datareeks DES

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

$$R_i = L_i \otimes f(R_{i-1}, K_i) \tag{3.8}$$

Decrypten van dit gedeelte kan ook weer door middel van XOR gebeuren, aangezien $a \otimes b \otimes b = a$. Wat f dus ook voor functie is, decrypten kan altijd met dezelfde functie als encrypten (zolang de sleutel bekend is...).

3.7.4 P-boxen en S-boxen

De functie f uit het bovenstaande stuk krijgt een 32-bits lange datareeks (R_{i-1}) en een 56-bits lange sleutelreeks (K_i) mee.

De 32 bits uit de datareeks worden opgesplitst in blokjes van 8 blokjes van 6 bits, met overlap. De eerste 2 bits zijn steeds gemeenschappelijk met het voorgaande blokje en de laatste 2 zijn steeds gemeenschappelijk met het nakomende blokje.

Zie de tabel “Opsplitsing datareeks DES”.

De 48-bits lange reeks die op deze manier ontstaat wordt ge-XOR-t met de 48 bits uit de 56-bits lange sleutel.

Nu wordt er gebruik gemaakt van de zogenaamde “S-boxen”: 8 verschillende tabellen waarin de 1^e en de 6^e bit uitstaan tegen de 2^e tot en met de 5^e bit. Wiskundig genoteerd: $S_1(b_1b_6, b_2b_3b_4b_5)$. Voor elk van bovenstaande blokjes wordt een andere tabel gebruikt.

De 32 bits die uit de acht S-boxen komen worden uiteindelijk in de volgende volgorde gezet:

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

Deze tabel is de zogenaamde “P-box”.

3.7.5 3 DES

Aangezien de sleutel van DES niet zo heel erg lang is, zijn er veel bedrijven die tegenwoordig gebruik maken van “triple DES” — DES wordt drie keer toegepast in plaats van 1 keer. Hiervoor zijn 2 sleutels nodig, en de procedure werkt als volgt:

1. Versleutel met k_1
2. Ontcijfer met k_2
3. Versleutel met k_1

Bij het ontcijferen geldt precies de omgekeerde procedure:

1. Ontcijfer met k_1
2. Versleutel met k_2
3. Ontcijfer met k_1

Hoofdstuk 4

Wiskundige principes

Aangezien cryptografie een wiskundige wetenschap is, moeten de wiskundige principes die eraan ten grondslag liggen in een werkstuk als dit besproken worden. Opvallend (en logisch) is dat alle wiskunde die met cryptografie te maken heeft met integers werkt: er zijn immers geen ‘gebroken’ boodschappen.

Symmetrische cryptografie maakt gebruik van zeer basale en eenvoudige wiskunde: optellen en aftrekken (het Caesar cijfer, substitutie, Vigenère) en het verschuiven van bits (DES).

Asymmetrische cryptografie maakt gebruik van modulair rekenen en priemgetallen. Deze twee gebieden zullen hier worden behandeld, voor zover van toepassing op cryptografie.

4.1 Modulair rekenen

Modulair rekenen gaat over het rekenen met resten bij deling. Het basisidee is het volgende:

$$a \equiv c \pmod{b} \tag{4.1}$$

Dit kan ook worden opgeschreven als:

$$a = k \cdot b + c \tag{4.2}$$

voor k = een willekeurig natuurlijk getal

Het voordeel voor cryptografen van modulair rekenen is dat het niet direct omkeerbaar is: dat is te zien aan de bovenstaande formule. k kan immers alle natuurlijke getallen zijn, en is dus niet te vinden als men alleen beschikt over c .

Natuurlijk is het wel mogelijk om een modulo-bewerking om te keren, anders was het immers onbruikbaar voor cryptografie aangezien decryptie dan onmogelijk zou zijn. Hoe dit kan wordt later uitgelegd.

4.2 Priemgetallen

Een priemgetal is een getal groter dan 1, en alleen deelbaar door 1 en zichzelf. Er zijn oneindig veel priemgetallen, en het is tegenwoordig een soort sport onder wiskundigen om steeds grotere priemgetallen te vinden. In de cryptografie worden priemgetallen van 512 bits (ongeveer 154 decimale cijfers) en 1024 bits (ongeveer 308 decimale cijfers) gebruikt.

4.3 Inverse modulo en relatieve priemgetallen

De inverse van een modulo wordt als volgt omschreven:

Voor inverse i van a , modulo m , geldt:

$$(a \cdot i) \equiv 1 \pmod{m} \quad (4.3)$$

Deze formule kan ook als volgt worden omschreven:

$$a \cdot i = m \cdot k + 1 \quad (4.4)$$

voor $k =$ een willekeurig natuurlijk getal

Dit wordt ook wel genoteerd als:

$$a^{-1} \equiv i \pmod{m} \quad (4.5)$$

Er is niet altijd een oplossing voor dit soort problemen: een voorwaarde is dat i en m relatief priem zijn. Dit betekent dat ze geen gemeenschappelijke priemfactoren hebben, ofwel dat hun GGD(grootste gemene deler) gelijk is aan 1.

De inverse van een modulo kan worden berekend met het zogenoemde ‘uitgebreide Euclidische algoritme’, hetgeen een uitbreiding is van het Euclidische algoritme waarmee GGD’s berekend worden.

4.3.1 Voorbeeld inverse modulo en relatieve priemgetallen

Bereken: $6^{-1} \pmod{13}$:

$$6 \cdot 1 \equiv 6 \pmod{13} \quad (4.6)$$

$$6 \cdot 2 \equiv 12 \pmod{13} \quad (4.7)$$

$$6 \cdot 3 \equiv 5 \pmod{13} \quad (4.8)$$

$$6 \cdot 4 \equiv 11 \pmod{13} \quad (4.9)$$

$$6 \cdot 5 \equiv 4 \pmod{13} \quad (4.10)$$

$$6 \cdot 6 \equiv 10 \pmod{13} \quad (4.11)$$

$$6 \cdot 7 \equiv 3 \pmod{13} \quad (4.12)$$

$$6 \cdot 8 \equiv 9 \pmod{13} \quad (4.13)$$

$$6 \cdot 9 \equiv 2 \pmod{13} \quad (4.14)$$

$$6 \cdot 10 \equiv 8 \pmod{13} \quad (4.15)$$

$$6 \cdot 11 \equiv 1 \pmod{13} \quad (4.16)$$

$$(6 \cdot 12 \equiv 7 \pmod{13}) \quad (4.17)$$

Er geldt dus: $6^{-1} \equiv 11 \pmod{13}$ Dit voldoet aan de voorwaarde dat i en m relatief priem moeten zijn: 11 en 13 hebben geen gemeenschappelijke delers behalve 1.

4.4 Algoritme van Euclides

Het ‘standaard’ algoritme van Euclides wordt gebruikt om de grootste gemene deler te berekenen van twee getallen. Het werkt als volgt:

Neem twee getallen a en b waarvan de GGD uitgerekend moet worden en waarvoor geldt: $a > 0$ en $a \geq b \geq 0$.

Schrijf a als $k \cdot b + c$

Schrijf b als $l \cdot c + d$

Schrijf c als $m \cdot d + e$

...

Ga door tot $e = 0$, dan geldt: $GGD(a, b) = d$

4.4.1 Bewijs

Als $e = 0$, dan geldt:

$$c = m \cdot d \tag{4.18}$$

$$b = (l \cdot m + 1) \cdot d \tag{4.19}$$

Dus $GGD(b, c) = d$, omdat m en $l \cdot m + 1$ geen gemeenschappelijke factoren hebben.

Nu kunnen a en b worden geschreven als:

$$a = k \cdot (l \cdot m + 1) \cdot d + m \cdot d = d \cdot (k \cdot (l \cdot m + 1) + m) \tag{4.20}$$

$$b = d \cdot (l \cdot m + 1) \tag{4.21}$$

Omdat $(l \cdot m + 1)$ en $(k \cdot (l \cdot m + 1) + m)$ geen gemeenschappelijke factoren hebben, geldt ook dat $GGD(a, b) = d$. Dit kan net zo vaak herhaald worden als nodig.

4.4.2 Voorbeeld gebruik algoritme van Euclides

Bereken met het algoritme van Euclides $GGD(654321, 123456)$.

$$654321 = 5 \cdot 123456 + 37041 \tag{4.22}$$

$$123456 = 3 \cdot 37041 + 12333 \tag{4.23}$$

$$37041 = 3 \cdot 12333 + 42 \tag{4.24}$$

$$12333 = 293 \cdot 42 + 27 \tag{4.25}$$

$$42 = 1 \cdot 27 + 15 \tag{4.26}$$

$$27 = 1 \cdot 15 + 12 \tag{4.27}$$

$$15 = 1 \cdot 12 + 3 \tag{4.28}$$

$$12 = 4 \cdot 3 + 0 \tag{4.29}$$

$$GGD(654321, 123456) = 3$$

4.5 Euler's phi functie

Een waarde waarmee gemoduleerd wordt, heet modulus. Bij elke modulus is er een *restklasse*: de mogelijke waarden van $k \pmod{m}$ waarbij k een natuurlijk getal is en m de modulus. Behalve de gewone restklasse bestaat er ook een *gereduceerde restklasse*: de leden van de restklasse die relatief priem zijn ten opzichte van de modulus. Een voorbeeld: de restklasse van 12 is $0, 1, 2, \dots, 10, 11$; de gereduceerde restklasse van 12 is $1, 5, 7, 11$.

De 'Euler phi functie' (geschreven als $\phi(n)$) is het aantal elementen in de gereduceerde restklasse. Bij 12 is dat dus 4, bij priemgetallen is dat altijd $n - 1$. Een bijzonder geval hiervan is als p en q priem zijn, en $n = p \cdot q$: dan geldt $\phi(n) = (p - 1)(q - 1)$. Dit is belangrijk voor asymmetrische cryptografie: vrijwel alle algoritmes maken er gebruik van.

4.6 Euler & Fermat

De kleine stelling van Fermat luidt:

Als m een priemgetal is en a niet deelbaar door m , dan geldt:

$$a^{m-1} = 1 \pmod{m} \tag{4.30}$$

Volgens 'Euler's generalisatie van de kleine stelling van Fermat' geldt:

$$a^{\phi(n)} = 1 \pmod{n} \tag{4.31}$$

4.7 Discrete logaritme

Een andere berekening, die enige gelijkenissen vertoont met het berekenen van de factoren van een priemgetal, is het berekenen van een discrete logaritme. Een discrete logaritme is de omgekeerde bewerking van een modulaire machtsverheffing. De meest opvallende eigenschap van de discrete logaritme, die ook al in de naam zit, is het feit dat de uitkomst altijd een geheel getal is. Bij grote getallen is dit, net als grote priemfactoren, moeilijk te vinden. Daarom wordt er ook van discrete logaritmen gebruik gemaakt in de asymmetrische cryptografie.

De volgende waarde is gemakkelijk te berekenen:

$$a^x \pmod{n} \tag{4.32}$$

De inverse-bewerking is echter moeilijker: vind een $|x|$ waarvoor geldt:

$$a^x = b \pmod{n} \tag{4.33}$$

waarbij i , n en b bekend zijn.

4.7.1 Voorbeeld inverse modulo

Vind een x waarvoor geldt:

$$3^x \equiv 15 \pmod{17} \tag{4.34}$$

$$3^1 \equiv 3 \pmod{17} \tag{4.35}$$

$$3^2 \equiv 9 \pmod{17} \tag{4.36}$$

$$3^3 \equiv 10 \pmod{17} \tag{4.37}$$

$$3^4 \equiv 13 \pmod{17} \tag{4.38}$$

$$3^5 \equiv 5 \pmod{17} \tag{4.39}$$

$$3^6 \equiv 15 \pmod{17} \tag{4.40}$$

In dit geval is de oplossing ($x = 6$) nog betrekkelijk eenvoudig te vinden, maar als het gaat om 1024-bits getallen, is het gemakkelijk in te zien dat het een stuk moeilijker wordt.

Een andere eigenschap van discrete logaritme is dat er niet voor alle vergelijkingen een oplossing bestaat, omdat de enige goede oplossingen integers zijn.

4.7.2 Voorbeeld inverse modulo zonder oplossing

$$3^x = 7 \pmod{13} \tag{4.41}$$

heeft geen oplossing.

$$3^1 \equiv 3 \pmod{13} \tag{4.42}$$

$$3^2 \equiv 9 \pmod{13} \tag{4.43}$$

$$3^3 \equiv 1 \pmod{13} \tag{4.44}$$

$$3^4 \equiv 3 \pmod{13} \tag{4.45}$$

enz. enz. enz.: dit gaat altijd zo door, dus komt er nergens 7 uit.

De veiligheid van veel public-key algoritmes is gebaseerd op de moeilijkheid van het vinden van een discrete logaritme. Deze is vergelijkbaar met die van RSA: er zijn, net als bij RSA, ongeveer

$$e^{(\ln(n))^{\frac{1}{2}} (\ln(\ln n))^{\frac{1}{2}}} \tag{4.46}$$

bewerkingen nodig om een discrete logaritme te vinden van n . Dit houdt in dat als priemgetal n de modulus is, het vinden van een discrete logaritme praktisch even moeilijk als het vinden van de factoren van een integer p van ongeveer dezelfde lengte als n , als p het product is van twee ongeveer even lange priemfactoren.

4.8 Factoren vinden

Een van de meestgebruikte public-key algoritmes is RSA. De veiligheid van RSA is gebaseerd op de moeilijkheid van het vinden van de factoren van grote getallen die het product zijn van twee grote priemgetallen.

Factoren zijn de getallen die met elkaar vermenigvuldigd moeten worden om een bepaald getal te krijgen.

4.8.1 Voorbeeld vinden factoren

$$2 \cdot 5 = 10 \quad (4.47)$$

$$2 \cdot 2 \cdot 3 \cdot 5 = 60 \quad (4.48)$$

$$3391 \cdot 23279 \cdot 65993 \cdot 1868569 \cdot 1066818132868207 = 2^{113} - 1 \quad (4.49)$$

De simpelste manier om de factoren van een getal te vinden is: deel het getal door alle priemgetallen die kleiner zijn dan de wortel van dat getal. Als het getal deelbaar blijkt door een van die priemgetallen, deel het er dan door, en ga op die manier verder met het overgebleven quotiënt.

Tegenwoordig zijn er snellere algoritmes om de factoren van een getal te vinden, maar ze hebben nog steeds allemaal de eigenschap dat ze n voor n getallen testen. Dit maakt het erg moeilijk om van grote getallen die als factoren twee ongeveer even grote priemgetallen hebben de factoren te vinden. Dit vergt ongeveer

$$e^{(\ln(n))^{\frac{1}{2}} \ln(\ln(n))^{\frac{1}{2}}} \quad (4.50)$$

bewerkingen. Dat betekent dat het vinden van de factoren van een getal van 200 decimale cijfers (tegenwoordig worden meestal getallen van 300 cijfers gebruikt in de cryptografie) 10^{23} bewerkingen vereist. Op een computer die per seconde 10 miljard (10 keer zoveel als nu gebruikelijk voor PC's) bewerkingen kan uitvoeren duurt dit 370 000 jaar.

4.9 Priemgeneratoren

Voor RSA zijn er vaak grote priemgetallen nodig. Daarom moet ook enige aandacht worden besteed aan de vraag hoe zo'n priemgetal wordt gegenereerd. De vraag is namelijk: als het vinden van de factoren van grote getallen zo moeilijk is, hoe moeilijk is het dan om grote priemgetallen te vinden?

Het antwoord is: moeilijk. Dat wil zeggen, als men gewoon willekeurig grote getallen kiest en dan probeert de factoren te vinden. Daarom zijn er een aantal tests bedacht, waarmee kan worden getest of een getal waarvan wordt vermoed dat het priem is ook werkelijk priem is.

Er zijn meerdere van dit soort tests, hier zal er een worden besproken om het principe te verduidelijken.

Rabin-Miller-test voor priemgetallen:

1. Kies een getal p waarvan je vermoedt dat het priem is.
2. Bereken b , waarbij geldt dat 2^b de grootste macht van 2 is die $p - 1$ deelt.
3. Bereken m , waarbij geldt dat $2^b \cdot m = p - 1$.
4. Kies een willkeurig getal a , zodat $a < p$
5. Stel $j = 0$ en stel $z = a^m \bmod p$
6. Als $z = 1$ of $z = p - 1$ dan kan p priem zijn; ga door met de test
7. Als $j > 0$ en $z = 1$ dan is p geen priem

8. Stel $j = j + 1$. Als $j < b$ en $z \neq p - 1$ stel dan $z = z^2 \pmod p$ en ga terug naar (4)

9. Als $z \neq p - 1$ niet voorkomt voordat $j = b$ dan is p geen priem

4.9.1 Voorbeeld Rabin-Miller-test Priem

Is 127 priem?

$$p = 127, b = 1, m = 63, a = 5$$

$$j = 0, z \equiv a^m \equiv 126 \pmod{p}$$

$z = p - 1, j = 0$ dus kleiner dan b , dus 127 is priem.

4.9.2 Voorbeeld Rabin-Miller-test Niet-Priem

Is 129 priem?

$$p = 129, b = 7, m = 1, a = 5$$

$$j = 0, z = a^m \equiv 5 \pmod{p}$$

$$j = 1, z = z^2 \equiv 25 \pmod{p}$$

$$j = 2, z = z^2 \equiv 109 \pmod{p}$$

$$j = 3, z = z^2 \equiv 13 \pmod{p}$$

$$j = 4, z = z^2 \equiv 40 \pmod{p}$$

$$j = 5, z = z^2 \equiv 52 \pmod{p}$$

$$j = 6, z = z^2 \equiv 124 \pmod{p}$$

$$j = 7, z = z^2 \equiv 25 \pmod{p}$$

$j = b$, maar $z = p - 1$ is niet voorgekomen, dus 129 is geen priem.

Door deze test een keer uit te voeren weet je met 75% zekerheid dat je een priemgetal hebt; met andere woorden: deze test produceert in een van de vier keren een niet-priemgetal. Door deze test dus 10 keer op een getal los te laten, heb je een kans van $\frac{1}{4}^{10}$ dat je geen priemgetal hebt. Omdat de test relatief gemakkelijk uit te voeren is door een computer, kan op deze manier een zekerheid van praktisch 100% bereikt worden dat je een priemgetal hebt.

Hoofdstuk 5

Asymmetrische cryptografie

Asymmetrische cryptografie is gebaseerd op het principe dat de encryptiesleutel verschilt van de decryptiesleutel. Dat ziet er in formule als volgt uit:

$$C = E_{k_e}(P) \quad (5.1)$$

$$P = D_{k_d}(C) \quad (5.2)$$

De functies voor encryptie en decryptie zijn dus niet gelijk, daarom heet het asymmetrische cryptografie.

5.1 Public-key cryptografie

De vorm van asymmetrische cryptografie die verreweg het meest voorkomt is de zogeheten ‘public-key cryptografie’: de encryptiesleutel is bekend, de decryptiesleutel is geheim. Dit systeem heeft het voordeel boven symmetrische cryptografie dat iedereen een boodschap die alleen voor persoon A bedoeld is kan versleutelen op zo’n manier dat alleen A hem ook weer kan ontcijferen. Bij symmetrische cryptografie is daarvoor de sleutel nodig die zowel en- als decryptiesleutel is. Deze zou dus algemeen bekend moeten zijn als iedereen een boodschap voor A zou moeten kunnen versleutelen, hetgeen het voordeel van de encryptie totaal teniet doet.

Er zijn vele public-key algoritmes, waarvan er veel onveilig zijn. Van de veilige algoritmes zijn er veel onpraktisch: als bijvoorbeeld de ciphertext veel groter is dan de plaintext, werkt dat niet prettig. Toch zijn er nog een aantal zowel veilig als praktisch. Deze zullen hier niet allemaal behandeld worden: we zullen ons hier beperken tot de bekendste: RSA.

5.2 RSA

RSA is genoemd naar de bedenkers: Rivest, Shamir en Adleman. Het is gebaseerd op de moeilijkheid van het vinden van de factoren van een getal dat het product is van twee zeer grote (100 of meer decimale cijfers) priemgetallen. Hier volgt de werking:

5.2.1 Werking RSA

Om de sleutels te verkrijgen, zijn twee grote priemgetallen p en q nodig.

Bereken $n = p \cdot q$

Kies een willekeurige encryptiesleutel e , zodat e en $(p - 1) \cdot (q - 1)$ relatief priem zijn.

Bereken tenslotte met het algoritme van Euclides d zodat

$$e \cdot d \equiv 1 \pmod{(p - 1) \cdot (q - 1)} \quad (5.3)$$

dit betekent dat d de modulaire inverse is van e :

$$e^{-1} \equiv d \pmod{(p-1) \cdot (q-1)} \quad (5.4)$$

Het betekent ook dat n en d relatief priem zijn.

De getallen p en q zijn nu niet meer nodig, maar moeten wel geheim blijven.

Om nu een bericht b te versleutelen moet het bericht eerst worden opgedeeld in blokken zodat elk blok modulo n uniek is. Dit betekent dat als p en q allebei 100 cijfers hebben, en n dus net geen 200, de lengte van die blokken ook net geen 200 tekens is.

Al deze blokken moeten nu worden versleuteld volgens de formule:

$$c_i \equiv b_i^e \pmod{n} \quad (5.5)$$

Alle blokken worden nu weer achter elkaar gezet, en vormen op die manier de ciphertext c . Om het bericht te ontcijferen moet met elk blok de volgende bewerking worden uitgevoerd:

$$b_i \equiv c_i^d \pmod{n} \quad (5.6)$$

Nu geldt:

$$c_i^d \equiv (b_i^e)^d \equiv b_i^{ed} \pmod{n} \quad (5.7)$$

Omdat $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$ geldt

$$b_i^{ed} = b_i^{k(p-1)(q-1)+1} = b_i \cdot b_i^{k(p-1)(q-1)} \equiv b_i \cdot 1 \equiv b_i \pmod{n} \quad (5.8)$$

voor k is een willekeurig natuurlijk getal.

5.2.2 Voorbeeld RSA

Codeer en decodeer het bericht '123456' met RSA.

$p = 37, q = 41, e = 77, d = 1253$

$$n = 37 \cdot 41 = 1517 \quad (5.9)$$

$$b_1 = 123 \quad (5.10)$$

$$b_2 = 456 \quad (5.11)$$

$$c_1 \equiv b_1^e \pmod{n} = 451 \quad (5.12)$$

$$c_2 \equiv b_2^e \pmod{n} = 636 \quad (5.13)$$

$$d_1 \equiv c_1^d \pmod{n} = 123 = b_1 \quad (5.14)$$

$$d_2 \equiv c_2^d \pmod{n} = 456 = b_2 \quad (5.15)$$

5.3 Veiligheid RSA

Om een RSA-sleutel te kraken is de simpelste manier het vinden van de factoren p en q van n . Dan kan immers met behulp van de openbare encryptiesleutel e de decryptiesleutel d worden berekend. De veiligheid van RSA hangt dus af van de moeilijkheid van het vinden van de factoren van n . Op dit onderwerp wordt verder ingegaan in het hoofdstuk over wiskundige principes.

5.4 Veiligheid public-key cryptografie

De veiligheid van public-key cryptografie is beperkt als het aantal mogelijke plaintexts beperkt is. Omdat een aanvaller beschikking heeft over de openbare sleutel, kan hij simpelweg alle mogelijke berichten encrypten en kijken of er toevallig dezelfde ciphertext uitkomt. Dit heet een gekozen-plaintext aanval.

Als het aantal mogelijke plaintexts te groot is om allemaal uit te proberen, hetgeen in de meeste gevallen zo zal zijn (berichten zijn in het algemeen veel te lang om alle mogelijkheden uit te proberen), moet rekening worden gehouden met de ‘gekozen-ciphertext aanval’. Dit is de zwakste plek van public-key algoritmes. Een gekozen-ciphertext aanval vereist dat de aanvaller een van aantal berichten de ciphertext en de ontcijferde plaintext weet. Dat kan alleen gebeuren in geval van onvoorzichtigheid aan de kant van de ontvanger (die is immers de enige die de berichten kan ontcijferen), maar dit is nou eenmaal geen ondenkbaar scenario, dus moet er rekening mee worden gehouden.

5.5 Sleutels genereren

Een manier om op een onveilig communicatiekanaal te zorgen dat meerdere personen beschikking hebben over dezelfde sleutel staat bekend als het Diffie-Hellman systeem. Het werkt als volgt:

5.5.1 Voorbeeld van het genereren van sleutels

1. Personen A en B nemen twee grote integers m en g zodat $1 < g < m$. Deze twee hoeven niet geheim te zijn.
2. Persoon A kiest een grote integer x_1 en berekent $X_1 = g^{x_1} \pmod{m}$.
3. Persoon B kiest een grote integer x_2 en berekent $X_2 = g^{x_2} \pmod{m}$.
4. Persoon A stuurt X_1 naar persoon B, en persoon B stuurt X_2 naar persoon A. x_1 en x_2 blijven geheim.
5. Persoon A berekent $k = X_2^{x_1} \pmod{m}$
6. Persoon B berekent $k' = X_1^{x_2} \pmod{m}$

k en k' zijn allebei gelijk aan $g^{x_1 x_2} \pmod{m}$. Persoon A en B hebben nu dus allebei beschikking over dezelfde sleutel, terwijl voor de rest van de wereld slechts m , g , X_1 en X_2 bekend zijn. Een buitenstaander kan k niet berekenen, tenzij hij de discrete logaritme kan berekenen en x of y ontdekken. Dat dit moeilijk is, is uitgelegd in het hoofdstuk over wiskundige principes.

Dit systeem werkt ook voor meer dan twee partijen: dan worden er extra rondes ingebouwd, en is $k = g^{x_1 x_2 \dots x_n} \pmod{m}$.

Hoofdstuk 6

Toepassing van cryptografie

6.1 Computersystemen

6.1.1 Hashes

De plaintext van wachtwoorden in computersystemen wordt nooit rechtstreeks opgeslagen, maar in plaats daarvan wordt de hash van het wachtwoord opgeslagen. Wanneer een gebruiker probeert in te loggen genereert de computer een hash van het ingevoerde wachtwoord en vergelijkt dat met het opgeslagen wachtwoord. Is dat wachtwoord correct, dan wordt de gebruiker toegestaan in te loggen, anders wordt inloggen geweigerd.

Groot voordeel van dit systeem is het feit dat het wachtwoord nergens in plaintext opgeslagen hoeft te worden. Dat betekent dat wanneer een ‘hacker’ toegang krijgt tot de computer en tot de opgeslagen wachtwoorden, deze wachtwoorden niet leesbaar zijn!

6.1.2 Pretty Good Privacy

Pretty Good Privacy (PGP) is een systeem dat veel gebruikt wordt voor digitale handtekeningen en om e-mail te versleutelen. Het systeem werkt met asymmetrische cryptografie. Elke gebruiker heeft een private sleutel die alleen op zijn of haar eigen computer staat en een sleutel die opgeslagen wordt op een globale server.

Gebruikers kunnen elkaars sleutels ‘ondertekenen’ met hun eigen sleutel — dat wil zeggen dat ze garant staan dat degene van wie ze de sleutel ondertekenen echt degene is voor wie deze zich uitgeeft. Over het algemeen gebeurt dit ondertekenen door controle van paspoorten. Wanneer een gebruiker de sleutel van een andere gebruiker ondertekent heeft stuurt hij de ondertekening naar de globale server.

PGP wordt gebruikt om handtekeningen te zetten onder e-mails en om e-mail te versleutelen¹. Wanneer een gebruiker *A* een bericht ondertekent en naar *B* stuurt, kan *B* controleren of *A* is wie hij zegt te zijn door te kijken of er iemand is die hij vertrouwt die op zijn beurt weer (indirect) *A* vertrouwt.

6.1.3 Voorbeeld van een PGP handtekening

Dit is het voorbeeld van een handtekening van een met PGP ondertekend document.

```
-----BEGIN PGP SIGNATURE-----  
Version: GnuPG v1.2.1 (GNU/Linux)  
  
iD8DBQE+M+I6Pa9Uoh7vUnYRAoUCAJ0dcXZ1G9E6NnEnm85Hn7ArsZ9mXQCgi+nQ  
BjhX25/nQ2yKKDkNfaYleSw=  
=lxuc  
-----END PGP SIGNATURE-----
```

¹zie het hoofdstuk over asymmetrische cryptografie voor details

Aangezien de kans bestaat dat iemand alle mogelijke publieke en private sleutels door gaat rekenen, is het zaak om regelmatig van sleutel te veranderen.

PGP gebruiker A stuurt een bericht aan gebruiker B , ondertekend met zijn PGP sleutel. B kijkt nu of er iemand is die hij vertrouwt die op zijn beurt weer (indirect) A vertrouwt. B blijkt de sleutel van C ondertekend te hebben die op zijn beurt de identiteit A geverifieerd heeft en A 's sleutel heeft getekend.

6.2 Mobiele Telefontie

Ook mobiele telefonie maakt gebruik van encryptie — de ether waar alle mobiele communicatie door heen gaat is immers door iedereen afluisterbaar. Oudere — analoge — mobiele telefoons die geen gebruik maakten van encryptie zijn met de juiste apparatuur zonder problemen af te luisteren.

De huidige mobiele telefoons (GSM's) maken gebruik van het zogenaamde A5/1-algoritme. Dit algoritme is door de fabrikanten geheimgehouden (de zogenaamde “security by obscurity”) maar is langzamerhand uitgelekt en in 1999 door Briceno, Goldberg en Wagner na analyse van een GSM-telefoon gekraakt. Het algoritme bleek gevoelig te zijn voor aanvallen en was met meer openheid waarschijnlijk veiliger ontworpen geweest.

Universal Mobile Telecommunication System (UMTS) is het systeem dat in de toekomst de GSM moet gaan vervangen. Het algoritme dat hier gebruikt wordt is wel meteen openbaar gemaakt. Het gebruikte algoritme wordt het *Kasumi* algoritme genoemd. Dit algoritme is een blokalgoritme dat met 128-bit sleutels werkt op blokken van 64 bits. Het algoritme wordt op elk blok 8 keer toegepast. De overeenkomsten met DES zijn groot, al is het voor UMTS gebruikte algoritme een stuk moeilijker te breken, aangezien er om precies te zijn $2^{128-56} = 2^{72} = 4722366482869645213696$ mogelijke sleutels zijn.

6.3 Smartcards

Smartcards(ook wel chipkaarten genoemd) zijn kleine plastic kaarten met een microchip erop. Ze worden gebruikt voor identificatie en zijn betrouwbaarder dan de magneetstrippen die momenteel voor bijvoorbeeld bankpassen gebruikt worden.

De smartcard bevat een geheugen waarin een geheime sleutel k opgeslagen is. Deze sleutel is alleen door de microchip te lezen. Verder is er nog een identificatienummer i op een toegankelijke plaats in het geheugen opgeslagen. i is bijvoorbeeld het rekeningnummer van de eigenaar van de smartcard. De microchip is bruikbaar voor vercijfering, in de meeste gevallen maken de chipcards gebruik van DES.

Zodra een chipkaart in een kaartlezer (bijvoorbeeld een pin- of een chipknipapparaat) wordt gestopt, zullen de kaart en het apparaat moeten verifiëren of de identiteit van de ander klopt.

Wanneer de identiteit van een kaart niet gecontroleerd zou worden zou het mogelijk zijn dat een vervalser een rekeningnummer van iemand anders op een kaart zet en er vervolgens geld mee opneemt.

Wanneer de identiteit van de kaartlezer niet klopt zou het kunnen dat de chipkaart in een apparaat is gestopt dat niet door de bank erkend is en misschien wel meer geld van een rekening afschrijft dan op het display komt te staan.

6.3.1 Controle identiteit

Alle kaartlezers die een bepaalde kaart kunnen lezen bevatten een zeer geheime sleutel K (wanneer deze sleutel bekend wordt kan het hele systeem misbruikt worden).

De sleutel k die op elke kaart veilig is opgeslagen is het resultaat van het uitvoeren van DES op de meestersleutel K en identiteitsnummer i :

$$k = DES(K, i) \tag{6.1}$$

De kaartlezer kan dus altijd k berekenen aangezien hij K en i weet. De kaart zal de k echter niet vrij willen geven, zolang niet zeker is dat de kaartlezer te vertrouwen is.

Wanneer een kaartlezer de identiteit van een kaart wil controleren stuurt deze een 64 willekeurige bits(r) naar de kaart. De kaart voert

$$c = DES(k, r) \tag{6.2}$$

uit. Als dit klopt met wat de kaartlezer zelf uitgerekend heeft, is de kaart echt.

Wanneer de kaart de identiteit van de kaartlezer wil controleren werkt het precies andersom — de kaart genereert een willekeurige reeks bits(r) en stuurt die naar de kaartlezer. De kaartlezer voert DES uit met de k van de kaart die hij uitgerekend heeft en r . Klopt deze uitkomst met die van de kaart, dan is de kaartlezer in orde.

6.3.2 Fraudebestendig

De truc die oplichters de afgelopen tijd regelmatig toegepast hebben om illegaal geld op te kunnen nemen zou met smartcards niet werken. Een magneetstriplezer werd in een pinautomaat geplaatst zodat de magneetstrip van een pinpas die in de pinautomaat gebruikt werd gelezen werd. Vervolgens schreven ze de magneetstrip op een nieuwe pinpas en namen er vervolgens geld mee op.

Met een smartcard zou zoiets onmogelijk zijn — de persoonlijke sleutel van de gebruiker is alleen door de microchip erop te lezen en de challenge is telkens anders.

Toch zijn er ook bij smartcards mogelijke gevaren. Wanneer de sleutel van de kaartlezer eenmaal bekend is is het meteen mogelijk alle kaarten te misbruiken.

Hoofdstuk 7

Politiek

7.1 Exportwetten in de Verenigde Staten

Volgens de Amerikaanse wet is encryptie een vorm van munitie en daarom gelden er dezelfde wetten voor als munitie: alleen onder bepaalde voorwaarden en alleen bepaalde (zwakkere) vormen mag cryptografische techniek geëxporteerd worden. Hierdoor is het niet mogelijk om bepaalde software vanuit de Verenigde Staten te importeren naar Europa.

Er zit echter een gat in de Amerikaanse wetgeving: alleen het exporteren van de broncode in elektronische vorm is verboden, maar het programma mag nog wel in boekvorm het land uit! Dit is dan ook wat er met de meeste bedrijven doen die hun software uit willen brengen op buitenlandse markten: de broncode wordt uitgeprint, getransporteerd en vervolgens opnieuw ingetypt.

In 2000 zijn deze wetten aangepast¹. Export van producten die sterke cryptografie (meer dan 64 bits) bevatten moet nog wel gemeld worden bij de overheid en export naar “schurkenstaten” is verboden, maar er hoeft niet meer om toestemming gevraagd te worden.

7.2 Discussie over vrijgeven sleutels door ttp's

De laatste tijd is er in Nederland een discussie ontstaan over in hoeverre TTP's (*Trusted Third Parties*) verplicht moeten worden om privé-sleutels te bewaren zodat die eventueel later door de overheid opgevraagd kunnen worden in geval van criminele of terroristische activiteiten.

Een aantal belangenorganisaties zoals *Bits of freedom*² en EFF³, die opkomen voor digitale burgerrechten en vrijheid van meningsuiting, menen dat alle vormen van cryptografie aan iedereen beschikbaar zouden moeten zijn. Overheden zijn vaak voor het verbieden van bepaalde vormen van cryptografie — zo wordt het bijvoorbeeld onmogelijk voor de politie om verdachten af te luisteren wanneer zij gebruik maken van versleutelingstechnieken. Daarnaast zijn er een aantal overheden (zoals die in de VS⁴) die de export verbieden omdat ze bang zijn dat vijandige overheden of terroristische organisaties er gebruik van zullen kunnen maken.

Op 17 December 2002 heeft het kabinet besloten dat TTP's in Nederland geen sleutels hoeven te bewaren⁵.

7.3 DMCA

De DMCA *Digital Millenium Copyright Act* is een wet die eind 1998 is aangenomen door het Amerikaanse congres. Doel van de wet was het tegengaan van de omzeiling van copyright-beschermings mechanismes. Grootste probleem dat echter optreedt is het feit dat analyse van de encryptie die gebruikt wordt om de copyrights te beschermen verboden is

¹<http://www.cdt.org/crypto/admin/>

²<http://www.bof.nl/cryptografie.html>

³www.eff.org

⁴De NSA (National Security Agency) in de VS moet apart toestemming geven voor elk product dat encryptie bevat en ge-exporteerd gaat worden

⁵<http://www.hccnet.nl/404/nieuws.cfm?id=7564>

geworden. Zo is er een Rus opgepakt die de cryptografie van e-books (electronische boeken) gebroken had, terwijl er slechts sprake was van ROT13! ⁶

Tegenstanders van de wet menen dat het verbieden van de analyse van crypto-algoritmes deze producten juist minder beschermen. Het is namelijk voor wetenschappers ook moeilijker om valkuilen in het algoritme op te sporen.

Ook in Europa zijn er initiatieven tot het invoeren van een DMCA-achtige wet, de EUCD⁷.

⁶Zie de paragraaf “Het Caesar Algoritme” in het hoofdstuk “Symmetrische Cryptografie”

⁷<http://wiki.ael.be/index.php/EUCD-Status>

Hoofdstuk 8

Conclusie

Op de hoofdvraag, (“Wat is cryptografie?”) is door het hele werkstuk heen antwoord gegeven. Hier zal een korte samenvatting komen, een antwoord op de vraag zoals de schrijvers die hebben opgevat.

Cryptografie is de wetenschap die zich bezighoudt met het versleutelen en ontcijferen van informatie.

In de cryptografie noteert men de informatie als P , de versleutelde informatie als C , de versleutelfunctie als E en de ontcijferfunctie als D . Deze functie heeft een parameter: de sleutel k . De versleutelfunctie is dus E_k , de ontcijferfunctie D_k .

Aangezien cryptografie tegenwoordig een computerwetenschap is, wordt de informatie gezien als binaire getallen. Dit heeft het voordeel dat er wiskundige berekeningen mee kunnen worden uitgevoerd: dat is in feite alles wat een computer kan.

Er zijn twee soorten van veiligheid waarmee wordt gewerkt: informatietheoretische en computationele veiligheid. Informatietheoretische veiligheid betekent dat de *cyphertext* geen informatie geeft over de *plaintext*. Dit wordt weinig toegepast, vanwege de grote sleutels die ervoor nodig zijn. Computationele veiligheid betekent dat de *cyphertext* wel enige informatie geeft over de *plaintext*, maar dat het zoveel rekenwerk kost om de sleutel te kraken dat het nutteloos is.

Technisch gezien zijn er twee soorten cryptografie: symmetrische en asymmetrische cryptografie.

Symmetrische cryptografie is de variant waarbij dezelfde sleutel wordt gehanteerd bij het versleutelen (encrypten) en ontcijferen (decrypten). Er geldt dus:

$$P = D_k(E_k(P)) \quad (8.1)$$

Bij symmetrische cryptografie gaat men ervan uit dat het algoritme bekend is, maar de sleutel niet. Als de sleutel ontdekt wordt, is het immers gemakkelijk een andere te kiezen, maar een ander algoritme vinden is moeilijker.

Symmetrische cryptografie werkt met het verschuiven, substitueren en vervangen van tekens. Dit kan ook nog in combinatie met blok-encryptie worden toegepast: per blok van een bepaald aantal tekens wordt een andere sleutel of een ander algoritme toegepast. Bekende voorbeelden van symmetrische cryptografie met blok-encryptie zijn de Duitse Enigma uit de WOII en het DES-algoritme dat sinds de jaren '70 veel wordt gebruikt.

Asymmetrische cryptografie is de variant waarbij er voor het encrypten en decrypten een verschillende sleutel wordt gebruikt:

$$P = D_{k_d}(E_{k_e}(P)) \quad (8.2)$$

Het voordeel hiervan is, dat het encryptie-algoritme en de encryptiesleutel bekend mogen zijn, alleen de decryptiesleutel moet geheim blijven.

Asymmetrische cryptografie werkt met het moduleren met priemgetallen van (machten van) de waarden die gecodeerd moeten worden. Hieraan is meteen te zien dat er verschillende sleutels moeten worden gebruikt om de informatie te encrypten en decrypten: een modulobewerking kan niet rechtstreeks worden omgekeerd.¹

Vrijwel alle cryptografie die tegenwoordig wordt toegepast is asymmetrisch. Tegenwoordig worden er sleutels gebruikt tot 1024 bits, vanwege de toegenomen beschikbare rekenkracht voor aanvallers, mensen die de codering willen kraken.

¹Het is indirect wel mogelijk; zie hoofdstuk 4: Wiskundige Principes

Cryptografie is in het huidige elektronische tijdperk hard nodig om gevoelige informatie te beschermen tegen buitenstaanders. Gevoelige informatie wordt namelijk steeds meer per computer verwerkt, verzonden en opgeslagen. Wat voor informatie dat is varieert sterk: pincodes, vertrouwelijke teksten, informatie over strategische doelen voor mogelijke vijanden of terroristen: het kan van alles zijn.

Op dit moment worden er nog niet alle mogelijkheden die cryptografie biedt gebruikt, maar het is te voorzien dat dit in de toekomst wel het geval zal zijn. Een goed voorbeeld van een toepassing die nog niet algemeen wordt gebruikt is de digitale handtekening: iedereen die iets over het internet verstuurt codeert en ondertekent dat met zijn/haar eigen 'handtekening'. Die handtekening is een groot getal dat alleen aan de eigenaar bekend is, waarmee deze het bericht versleutelt. De decryptiesleutel is een ander groot getal, waarmee het kan worden ontcijferd. Deze decryptiesleutel is bekend aan de ontvanger, die op deze manier kan controleren of de afzender wel is wie hij zegt te zijn. Die handtekening is een toepassing van RSA.

Zo'n digitale handtekening zal waarschijnlijk in de toekomst net zo algemeen worden als een geschreven handtekening nu: het biedt bijvoorbeeld zekerheid bij e-commerce en dergelijke toepassingen.

Een goede vraag voor ethici met betrekking tot cryptografie tot slot: hoever mag een overheid gaan in het controleren van (het gebruik van) cryptografische kennis? Enige controle moet er vanzelfsprekend zijn, net als dat de politie in sommige gevallen telefoontaps mag plaatsen. Men kan echter te ver gaan: veel mensen vinden dat de Amerikaanse regering dat doet, door bijvoorbeeld encryptie van e-mail te verbieden, omdat die anders niet gemakkelijk genoeg kan worden gelezen door de FBI en CIA. Een nieuwe vraag is dus: hoe ver kan men gaan?

De cryptografie is nog steeds niet uitontwikkeld. Er worden steeds nieuwe algoritmes verzonden, oude worden gekraakt. Quantumcomputers zullen het in de toekomst mogelijk maken de cryptografie die we vandaag de dag gebruiken in luttele seconden te breken.

Bijlage A

Logboek

Datum	Omschrijving
20-25 September 2002	Vorbereiding en opzoeken documentatie over onderwerp
30 September 2002	Werkplan ingeleverd en overleg met Niels de Bruin over werkplan
1 Oktober 2002	Eerste opzet document
29 Oktober en 31 Oktober 2002	Naar Universiteitsbibliotheek Utrecht geweest
25-30 November 2002	Gewerkt aan eerste hoofdstukken
4-14 December 2002	Gewerkt aan Politiek en Toepassingen
16 December 2002	Bij Olivier de hele dag gewerkt aan begrijpen van modulo-rekenen, asymmetrische cryptografie en DES
17-19 December 2002	Gewerkt aan DES en asymmetrische cryptografie
19 December 2002	Aanpassingen layout
20 December 2002	Overleg met Niels de Bruin over vooruitgang
20 December-3 Januari	Afronden hoofdstukken wiskundige principes
3 Januari 2003	Bij Jelmer thuis gewerkt aan asymmetrische cryptografie en DES
21 Januari 2003	Bij Olivier thuis gewerkt aan wiskundige principes
27 Januari 2003	Bespreking met Niels de Bruin
10-22 Februari 2003	Verbeteringen en toelichtingen gemaakt

Bibliografie

- [1] Data encryption standard (des). *Blehtley Park*. <http://www.bletchleypark.net/encrypt/des.html>.
- [2] Pete Azzole. Ultra: The silver bullet. "*CRYPTOLOG*", *the Journal of the U.S. Naval Cryptologic Veterans Association*, 1996. <http://www.cl.cam.ac.uk/Research/Security/Historical/azzole1.html>.
- [3] Faculteit Wiskunde en Informatica. Profielwerkstuk kraken. *TU Eindhoven*, 2001. <http://www.win.tue.nl/jessers/aansluiting/profielwerkstukkraken.htm>.
- [4] Bert-Jaap Koops. Crypto law survey. , 2002,2003. <http://rechten.kub.nl/koops/cryptolaw/>.
- [5] majoor Detlev Simons. Van spijkerschrift tot enigma. *Vereniging van Onderofficieren en oud- Onderofficieren van het Wapen van de Verbindingsdienst*, . <http://www.voov.nl/284spkrenigm.html>.
- [6] University of Arizona. The enigma machine. <http://www.math.arizona.edu/dsl/enigma.htm>.
- [7] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc, 1994.
- [8] Claus Schnleber. *Verschlusungsverfahren fr PC-Daten*. Franzis' Online-Book. ISBN 3-7723-8853-1.
- [9] Gerard Tel. *Cryptografie: Beveiliging van de digitale maatschappij*. Pearson Education, 2002.
- [10] Lorraine C. Williams. A discussion of the importance of key length in symmetric and asymmetric cryptography. 2001. http://www.sans.org/rr/encryption/key_length.php.