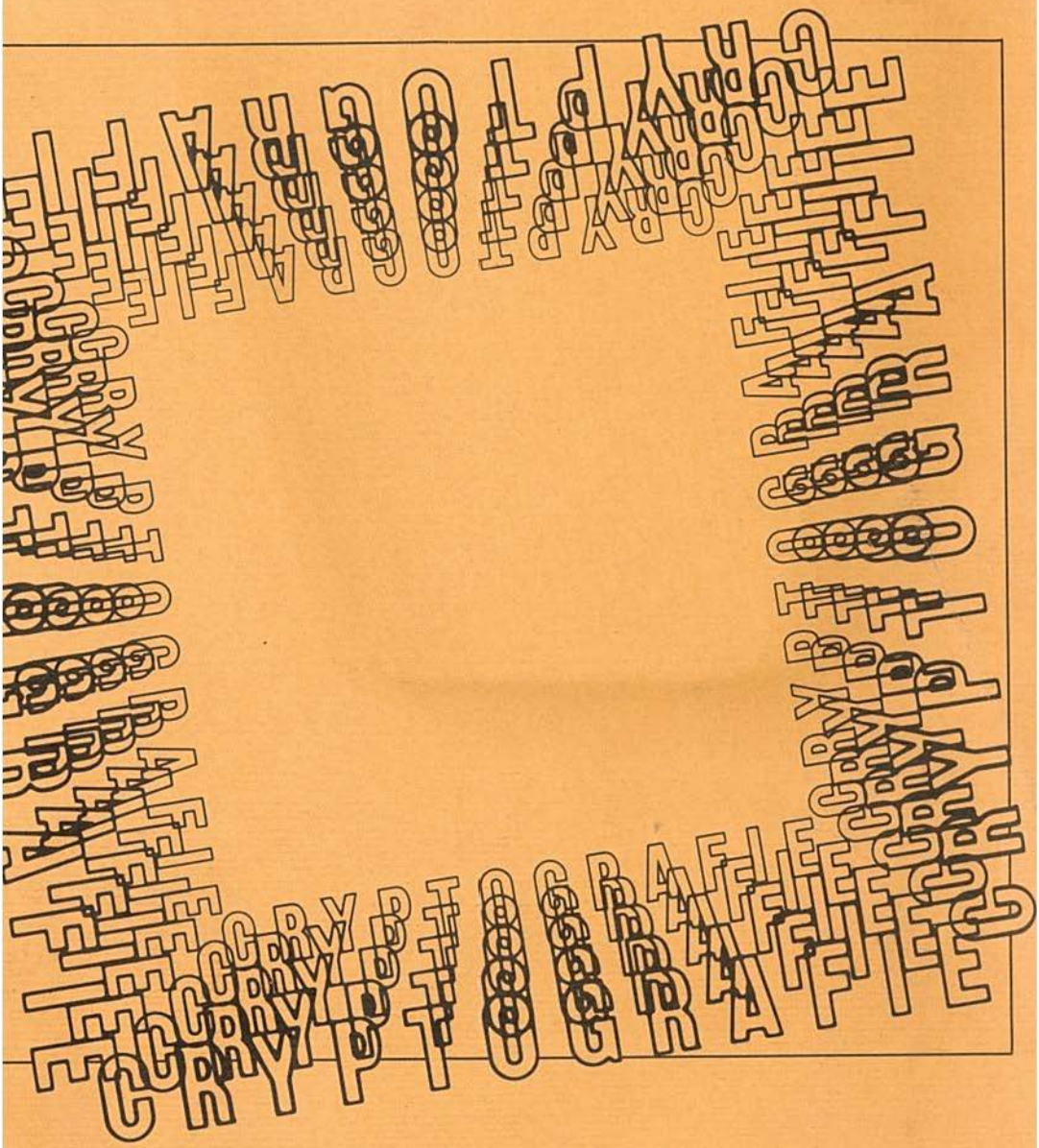


MC

CURSUS CRYPTOGRAFIE

MATHEMATISCH CENTRUM AMSTERDAM

4 - 8 OKTOBER 1982

Deze syllabus bevat de teksten van de voordrachten die worden gegeven in het kader van de cursus "CRYPTOGRAFIE", die plaatsvindt van 4 t/m 8 oktober 1982 in het Mathematisch Centrum te Amsterdam.

Opmerkingen over de hier gepresenteerde teksten kunnen rechtstreeks aan de auteur van het betreffende artikel worden toegezonden, of aan

P.J. Hoogendoorn
Mathematisch Centrum
Kruislaan 413
1098 SJ Amsterdam

ADRESSEN VAN DE AUTEURS

A.E. Brouwer,
P.J. Hoogendoorn: Mathematisch Centrum
Kruislaan 413
1098 SJ Amsterdam

M.R. Best,
H.F.A. Roefs,
R.P. de Moel: Nationaal Lucht- en Ruimtevaartlaboratorium
Postbus 153
8300 AD Emmeloord

H.C.A. van Tilborg: Onderafdeling der Wiskunde en Informatica
Technische Hogeschool Eindhoven
Postbus 513
5600 MB Eindhoven

P. van Emde Boas,
H. van der Meer: Instituut voor Toepassingen van de Wiskunde
Universiteit van Amsterdam
Roetersstraat 15
1018 WB Amsterdam

INHOUD

I. Inleiding (H.C.A. van Tilborg)

II. Shannon-theorie en cryptografie (H.F.A. Roefs)

III. Klassieke cryptografische systemen (H. van der Meer)

IV. Complexiteitstheorie (P. van Emde Boas)

V.1. Public key cryptografie (P.J. Hoogendoorn)

V.2. Voorbeelden van public key cryptosystemen (A.E. Brouwer)

VI. De Data Encryption Standard (A.E. Brouwer)

VII. Shift register sequences (H.C.A. van Tilborg)

VIII. Toepassingen van de cryptografie (M.R. Best)

IX. Security, een verzameling maatregelen of een mentaliteit (R.P. de Moel)

X. Hoe deel je een geheim (H.C.A. van Tilborg)

I. INLEIDING.

Cryptologie (van het Griekse "cryptos" en "logos") is de leer van het geheimschrift. Het is opgesplitst in:

cryptografie - het ontwerpen van geheimschriften

crypto-analyse - het ontcijferen van geheimschriften.

Het is duidelijk dat men niet goed een van deze beide takken kan beoefenen, zonder voldoende kennis van de andere tak.

Het gebruik van geheimschriften vindt men gedurende de gehele geschiedenis, van de klassieke oudheid tot nu. Een zeer lezenswaardig boek, wat dit betreft, is het boek "The Codebreakers" van D. Kahn [2], dat zeer interessant vertelt over de geschiedenis van de cryptologie door de eeuwen heen, waarbij de Tweede Wereldoorlog zeer uitvoerig behandeld wordt.

Zoals ook uit dat boek blijkt is cryptologie altijd het exclusieve domein van de overheid geweest. Het waren de militairen en de diplomatieke diensten die gebruik maakten van geheimschriften.

Zoals het woord geheimschrift suggereert houdt cryptografie zich bezig met het beveiligen van boodschappen (gegevens, data). Het is goed zich te realiseren dat deze beveiliging nodig kan zijn bij het transport van deze gegevens van een plaats naar een andere, als ook bij de opslag van gegevens (transport in de tijd).

Het woord "geheimschrift" suggereert misschien ook (en ten onrechte) dat cryptografie zich enkel bezig houdt met de beveiliging van gegevens tegen afluisteren. Cryptografische systemen worden afhankelijk van de toepassing gebruikt om redenen van

gevens

authentificatie - de ontvanger van gegevens wil zekerheid dat deze gegevens komen van wie hij denkt dat ze komen (denk hierbij ook aan access control), ook als deze deelnemer dat later ontkent (denk b.v. aan financiële transacties).

De laatste jaren is de automatische data-verwerking hand over hand toegenomen. Dat de ontwerpers en gebruikers van dit soort systemen hun gegevens willen beschermen tegen "afluisteren" en/of "vervalsen" is voor de hand liggend. Het is de cryptologie die methodes bestudeert en ontwikkelt die beveiliging geven tegen deze gevaren. Drie concrete toepassingsmogelijkheden van de cryptografie zijn:

data-transport over leidingen (directe verbindingen, netwerken) of straalzenders,

data-opslag in data-banken of files,

authentificatie van gebruikers of berichten.

De manier waarop een cryptografisch systeem werkt is door het vercijferen van de oorspronkelijke gegevens (we noemen dit de klare tekst). De aldus verkregen tekst wordt cijfertekst genoemd. Dit vercijferen gebeurt door middel van een transformatie (algorithme) T_k die afhankelijk is van een bepaalde sleutel k, gekozen uit de sleutelverzameling K. Het ontcijferen gebeurt door middel van een transformatie S_k die voor alle boodschappen m voldoet aan

(1,1) $$S_k * T_k(m) = m .$$

Hierbij zijn zowel T_k als S_k relatief snel uit te rekenen algorithmes.

In de klassieke cryptosystemen hebben beide gebruikers de sleutel k uitgewisseld op een geheime manier (of van te voren afgesproken). In de zgn. public key systemen is T_k algemeen bekend, maar kan de "vijand" hiermee toch niet het ontcijferde algorithme S_k bepalen. In latere hoofdstukken zullen beide systemen uitvoerig besproken worden.
Bij de beoordeling van de veiligheid van een cryptosysteem gaat men er meestal van uit dat de "vijand" weet voor wat voor systeem er gekozen is (vroeg of laat komt hij daar toch wel achter), maar niet de keuze van de sleutel kent. Hoe vaak je van sleutel moet wisselen om je systeem veilig te houden komt later nog aan de orde,
Het is duidelijk dat een crypto-analist met extra informatie over een voordeel beschikt. We onderscheiden de volgende drie niveau's van crypto-analyse:

<u>ciphertext only attack</u> - Alleen een stuk cijfertekst is bekend. Op grond hiervan probeert de tegenstander de klare tekst te achterhalen of beter nog de gebruikte sleutel, zodat de rest van de boodschappen (tot de eerstvolgende sleutelwisseling) gemakkelijk te ontcijferen zijn. Wel beschikt de tegenstander over de context van de data. Denk hierbij aan zaken zoals de standaard aanhef van een brief, taalkarakteristieken zoals de verschillende letterfrequenties of de standaard inrichting van de gegevens in een file.

<u>known plaintext attack</u> - De crypto-analist heeft nu bij zijn werk de beschikking over een redelijke hoeveelheid klare tekst met de corresponderende cijfertekst. Als men een systeem gebruikt dat bestand is tegen een known plaintext attack, is er geen noodzaak om ontvangen en ontcijferde berichten te

chosen plaintext attack - De crypto-analist heeft nu bij zijn werk zelfs de beschikking over een hoeveelheid door hem zelf gekozen klare tekst en de daarmee corresponderende cijfertekst. De zgn. "public key" cryptosystemen die later nog uitvoerig besproken zullen worden, moeten bestand zijn tegen een chosen plaintext attack, om de simpele reden dat bij deze systemen de manier van vercijferen openbaar is.

Om crypto-analyse op een goed niveau te kunnen verrichten zal men kennis moeten hebben van zulke uiteenlopende wiskundige disciplines als kansrekening en statistiek, lineaire algebra, abstracte algebra (vooral eindige groepen en eindige lichamen) en complexiteitstheorie.
In de literatuur komt men vaak nog de volgende begrippen tegen:

stroomcijfers - Dit zijn crypto-systemen die de binnenkomende digits een voor een vercijferen. De shift register systemen die we later zullen behandelen zijn hier een voorbeeld van.

blokcijfers - Nu worden de binnengekomen digits in blokken van vaste lengte (zeg n) behandeld. Het DES-systeem is zo'n systeem met $n = 64$.

Vanuit een strict mathematisch standpunt bezien is er geen echt verschil tussen beide systemen. Een stroomcijfer is een blokcijfer met $n = 1$ en andersom een blokcijfer is een stroomcijfer, waarbij de input uit een groter alfabet komt.
Een belangrijke vraag bij het ontwerpen van een automatisch data-ver-

werkingssysteem is voor welk cryptosysteem moet men kiezen als de gegevens beveiligd moeten worden. In zijn algemeenheid is deze vraag niet te beantwoorden. Het volgende lijstje geeft toch een idee van de diverse overwegingen, die men moet maken.

- Door de wet gestelde beperkingen.
- Taxatie van de risico's die de te beschermen data lopen in de omgeving van hun opslag en/of transport. Denk hierbij aan de hardware, het bedrijfssysteem, de transportleidingen en de gebruikersstations, het bedienings- en onderhoudspersoneel en de rechtmatige en onrechtmatige gebruikers.
- De gevoeligheid van deze gegevens. Betreft het personen? Zijn het bedrijfsgeheimen, etc? Tegen wat voor soort indringen wil je je beschermen? In dit verband maakt men onderscheid tussen:

  <u>passief indringen</u> - afluisteren van informatie

  <u>actief indringen</u> - verminken, vernietigen of toevoegen van informatie, retransmissie van (eventueel onbegrepen) boodschappen, je als een ander voordoen etc.
- Technische realiseringsmogelijkheden in soft- en/of hardware. compatibiliteitsproblemen.
- De kosten verbonden aan de invoering van het systeem, zoals aanschafkosten, implementatiekosten, lopende kosten (de kosten van het ver- en ontcijferen en van het sleutelbeheer).

De lezer die zich verder wil verdiepen in de kwetsbaarheid verwijzen we in eerste instantie naar [5].

Het is goed zich te realiseren wat de beperkingen van de cryptografie zijn.

De vercijfering beschermt de data niet tegen (on)opzettelijke veranderingen. Die verandering kan door vercijfering wel gemakkelijker ontdekt worden.

- De vercijferde data kunnen normaal gesproken niet voor onmiddelijke verwerking gereed zijn. Ze zullen eerst ontcijferd moeten worden en dienen in klare tekst vorm zorgvuldig beschermd te worden.
- Het te voeren sleutelbeheer. Wie is daar verantwoordelijk voor? Hoe transporteer je sleutels op een veilige manier? Bijhouden wie welke sleutel heeft en aan wie die sleutel doorgegeven is.

Drie standaardboeken op het gebied van de cryptologie zijn [1], [4] en [5]. Verder is [3] een goed overzichtsartikel.

REFERENTIES

[1] Beker, H. en F. Piper, Ciphersystems, the protection of communications, Northwood Books, 1982.

[2] Kohn, D., The codebreakers, the story of secret writing, Macmillan Company, New York, 1967.

[3] Lempel, A., Cryptology in transition: A survey, ACM Computing Surveys, 11, 1979, 285-303.

[4] Konheim, A., Cryptography, a primer, John Wiley & Sons, New York, 1981.

[5] Ryska, N. en S. Herda, Kryptografische Verfahren in der Datenverarbeitung, Informatik-Fachberichte 24, Springer-Verlag, Berlin etc., 1980.

SHANNON-THEORIE EN CRYPTOGRAFIE

1 Shannon-theorie

In het algemeen is het doel van communicatie om op de bestemming een verzonden boodschap zo goed mogelijk te reproduceren zelfs als de boodschap onderweg door storingen wordt verminkt. De doelstelling van cryptografie staat hier nagenoeg haaks op: men wil het reproduceren van de boodschap zo moeilijk mogelijk maken, althans voor diegene voor wie de boodschap niet bestemd is, ook al is het verzonden bericht zonder storingen ontvangen. De ideeën door C.E. Shannon ontwikkelt in het opzienbarende artikel "A mathematical theory of communication" (1948) en het vlak daarop gepubliceerde (en geclassificeerde) "Communication theory of secrecy systems" (1949) sluiten dan ook nauw op elkaar aan.
In dit hoofdstuk zullen een aantal van Shannon's inzicht-gevende ideeën worden besproken. Zij vormen een theoretische basis voor cryptografie.

1.1 Shannon's informatiemaat

Intuïtief is aan te voelen dat de hoeveelheid informatie die besloten ligt in een voorval een relatie moet hebben met de kans dat het voorval zal plaats hebben. Hoe groter de kans op een voorval hoe kleiner de onzekerheid over het voorval, of hoe kleiner de hoeveelheid informatie die met het plaatsvinden van het voorval ter beschikking komt. Men kan ook aanvoelen dat een zeker voorval als zodanig geen informatie verschaft; als twee voorvallen, die niets met elkaar te maken hebben, als één gebeurtenis worden opgevat moet de informatie vervat in die ene gebeurtenis gelijk zijn aan de som van de informatie besloten in elk der individuele voorvallen. Zo zijn een aantal voorwaarden te formuleren waaraan een informatiemaat zou moeten voldoen. Shannon's informatiemaat is gebaseerd op de negatieve logaritme van de voorvalkans. In de afgelopen 30 jaar is gebleken dat Shannon's maat uitstekend voldoet en als zodanig heeft bijgedragen tot de stormachtige ontwikkelingen in de telecommunicatie.

Neem aan dat X een stochastische variabele is die een aftelbaar aantal waarden $x_1, x_2, \ldots\ldots, x_K$ kan aannemen. Als $X = x_k$ noemen we dit een voorval. De kans op dit voorval schrijft men als $\mathrm{Prob}(X = x_k)$ of $P_X(x_k)$. De waarde van $P_X(x_k)$

dan is $x_1 = 1, x_2 = 2, \ldots\ldots, x_6 = 6$ en $p_1 = p_2 = \ldots\ldots = p_6 = \frac{1}{6}$.
Bij een valse dobbelsteen zou men echter kunnen vinden dat $p_1 = \frac{1}{2}, p_2 = p_3 = p_4 = p_5 = p_6 = \frac{1}{10}$. Intuïtief is de onzekerheid over de afloop van de worp met de zuivere dobbelsteen groter dan die met de valse dobbelsteen. In feite is de onzekerheid over de waarde die X zal aannemen het grootst als de kansen p_k, $k = 1,2,\ldots\ldots,K$ gelijk zijn.

Vooruitlopend op de formele definitie noemen we nu de gemiddelde hoeveelheid onzekerheid over de variabele X, H(X). Omdat H(X) van de kansvector $\underline{p} = (p_1, p_2, \ldots\ldots, p_K)$ afhangt is de notatie $H(\underline{p})$ nuttig ter omschrijving van enige <u>eisen</u> die aan H(X) moeten worden gesteld.

Eis 1. Als $\underline{p} = (p_1, p_2, \ldots\ldots, p_K)$ en $\underline{p}' = (p_1, p_2, \ldots\ldots, p_K, 0)$ dan moet $H(\underline{p}) = H(\underline{p}')$. Dus als X' een andere stochastische variabele is die naast $x_1, x_2, \ldots\ldots, x_K$ nog een additionele waarde x_{K+1} kan aannemen, echter met een kans 0, dan moet de onzekerheid over X' en X gelijk zijn.

Eis 2. Als $\underline{p} = (p_1, p_2, \ldots\ldots, p_K)$ en de één vector $\underline{1}_K = (1,1,\ldots\ldots,1)_K$ dan moet $H(\underline{p}) \leq H(\frac{1}{K} \cdot \underline{1}_K)$ waarbij het gelijkteken geldt als $p_k = \frac{1}{K}$ voor alle k. Ofwel, de onzekerheid over X is het grootst als de kansen op de individuele voorvallen aan elkaar gelijk zijn.

Eis 3. Hoe groter het aantal mogelijke voorvallen (met gelijke kansen) hoe groter de onzekerheid over de uitkomst. Dit vertaalt zich als
$H(\frac{1}{2} \cdot \underline{1}_2) < H(\frac{1}{3} \cdot \underline{1}_3) < H(\frac{1}{4} \cdot \underline{1}_4) < \ldots\ldots$

Eis 4. Als vector $\underline{p}$ slechts heel weinig afwijkt van $\underline{q}$ dan moet ook $H(\underline{p}) \approx H(\underline{q})$.

Eis 5. $H(p_1, p_2, \ldots\ldots, p_K) = H(p_2, p_1, \ldots\ldots, p_K) = H(p_3, p_2, p_1, \ldots\ldots, p_K) = \text{etc.}$
Verandering van de verdeling der kansen over de individuele voorvallen zal de gemiddelde onzekerheid over X niet veranderen.

Eis 6. Tenslotte stelt men een 'modificatie' eis die hier niet nader zal worden uitgewerkt maar resulteert in de voorwaarde dat

$$H(p_1, p_2, \ldots\ldots, p_{K-2}, p_{K-1} + p_K) + (p_{K-1} + p_K) H\left(\frac{p_{K-1}}{p_{K-1} + p_K}, \frac{p_K}{p_{K-1} + p_K}\right) =$$

$$H(p_1, p_2, \ldots\ldots, p_{K-1}, p_K).$$

De enige funktie die aan alle gestelde eisen blijkt te voldoen is $H(p_1,p_2,\ldots,p_K) = \beta \sum_{k=1}^{K} p_k \log p_k$, met $p_k \geq 0$, $0\log 0 \triangleq 0$ en β een willekeurige negatieve constante.

Shannon definiëert de zelfinformatie in het voorval $[X = x_k]$ als $I_X(x_k) = -{}^2\log P_X(x_k)$. De gemiddelde onzekerheid of <u>entropie</u> van de stochastische variabele X is nu

$$H(X) = -\sum_{k=1}^{K} P_X(x_k)\,{}^2\log P_X(x_k). \tag{1.1}$$

De entropie H(X) is dus de gemiddelde waarde van de zelfinformatie der individuele voorvallen. H(X) kan worden opgevat als de gemiddelde hoeveelheid informatie die nodig is om de onzekerheid over X op te heffen of om de voorvallen van X te kunnen beschrijven (in bits als 2 het grondtal van de logaritme is).

<u>Voorbeeld 2:</u> Entropie van munt

Bij het tossen van een munt met kans p op "kruis" en (dus) een kans 1 - p op "munt" is de entropie van de uitslag X gelijk aan $-p\log p - (1 - p)\log(1 - p)$. Deze funktie wordt vaak genoteerd als H(p). Figuur 1 geeft een schets van H(p) voor $0 \leq p \leq 1$.

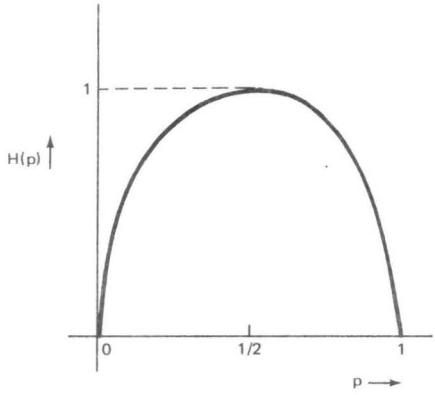


Fig. 1 $H(p) = -p\,{}^2\log p - (1 - p)\,{}^2\log (1 - p)$

over de uitslag van de toss op te heffen.
Heeft men een zuivere munt P_X('kruis') = P_X('munt') = $\frac{1}{2}$ dan is de entropie maximaal en gelijk aan 1 bit; men kan de uitslag van de toss niet korter weergeve dan met 1 bit/toss; men volstaat met het noteren van 0 bij "kruis" en een 1 bij "munt" (of omgekeerd).
Heeft met een valse munt met $p = \frac{1}{4}$ dan blijkt $H(0.25) = 0.81$ bit. De gemiddelde onzekerheid in de tossuitslag is minder dan het maximum want men verwacht meer "munt". Per uitslag heeft men slechts 0.81 bit nodig om de toss uitslagen te kunnnen weergeven. Hoe dit kan worden gedaan is hier niet van belang (in het algemeen zal men een aantal uitslagen bij elkaar nemen en representeren).

Opgave 1

Bereken de entropie van de zuivere en valse dobbelsteen in Voorbeeld 1.

Voorbeeld 3: Entropie van tekst in Nederlandse taal

Wat is de entropie van Nederlandse tekst?
Als we de 26 letters van het alphabet met binaire digits per letter willen weergeven dan zijn tenminste $^2\log 26 = 4.7$ bit per letterteken nodig (in de praktijk wordt dit dan 5 bits/letter om ook nog punten, komma's e.d. te kunnen weergeven). In de Shannon-gedachte is deze manier van representatie alleen optimaal als elke letter een gelijke kans van optreden zou hebben. Alleen dan is nl. $H(\frac{1}{26} \cdot \underline{1}_{26}) = 4.7$ bit. In onze taal is echter de letterfrequentie in tekst ongelijk. De letter e komt bijv. ongeveer 3 maal zo vaak voor als de lette t. Figuur 2 geeft een indruk van de letterstatistiek voor verschillende moderne talen [Ref. 5]. Gebruikt men deze letterstatistiek $\underline{p}$ in de Shannon informatiemaa dan vindt men $H(\underline{p}) = 4.15$ bit/letter. In de praktijk kan deze reduktie in het aantal bits per letter worden bereikt door aan veel voorkomende letters een klei aantal bits en aan weinig voorkomende een groot aantal bits toe te kennen (vgl. de zelfinformatie voor iedere letter). De Morse code is een goed voorbeeld van dit idee. Het is overigens mogelijk om met nog minder dan 4.15 bit/letter een Nederlandse tekst weer te geven. De onderlinge relaties van de letters (letterparen e.d.) dienen dan in de beschouwing te worden betrokken. De limiet ligt rond 1.5 bit/letter. Uit het grote verschil tussen de 4.7 bit/letter waarmee dit voorbeeld begon en de 1.5 bit/letter die voor Nederlandse tekst theoretisch nodig is blijkt de grote overvloedigheid (redundantie) van onze taal. Verwijderi van deze redundantie uit een boodschap kan, zoals we nog zullen zien, de cryptografische beveiliging van de informatieinhoud verbeteren.

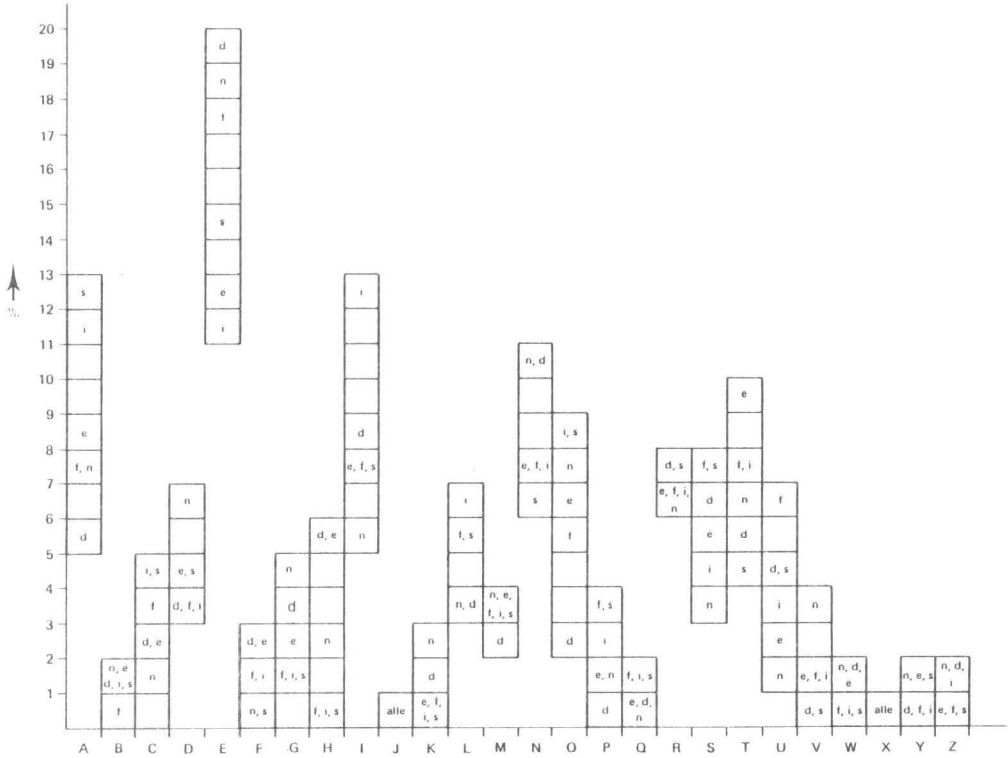


Fig. 2 Letterfrequenties nederlands (n), duits (d), engels (e), frans (f), italiaans (i) en spaans (s)

Ruwweg gesproken is wederzijdse informatie de hoeveelheid informatie over een voorval A als een voorval B plaatsvindt. In een communicatiesysteem kan men de verzonden boodschap als voorval A en de ontvangen boodschap als voorval B opvatten. In dat geval wil men de wederzijdse informatie zo groot mogelijk maken. In cryptografische toepassingen wil men echter de wederzijdse informatie tussen boodschap en cryptogram zo klein mogelijk houden zolang de 'sleutel' tot het cryptogram niet bekend is.

Neem aan dat X en Y stochastische variabelen zijn die respectievelijk de waarden $x_1, x_2, \ldots, x_K$ en $y_1, y_2, \ldots, y_M$ kunnen aannemen.
De zgn. gezamenlijke kans op het voorval $[X = x_k, Y = y_m]$ wordt geschreven als $P_{XY}(x_k, y_m)$. In het algemeen geldt

$$P_{XY}(x_k, y_m) = P_{X|Y}(x_k|y_m) P_Y(y_m), \tag{1.2}$$

met $P_{X|Y}(x_k|y_m)$ de zgn. voorwaardelijke kans dat het voorval $[X = x_k]$ plaatsvindt onder de voorwaarde dat $[Y = y_m]$ heeft plaatsgevonden. Als X en Y onafhankelijk van elkaar zijn dan reduceert $P_{X|Y}(x_k|y_m)$ tot $P_X(x_k)$. De voorwaardelijke zelf-informatie van het voorval $[X = x_k]$ gegeven dat $[Y = y_m]$ heeft plaatsgevonden is gedefiniëerd als $I_{X|Y}(x_k|y_m) = -{}^2\log P_{X|Y}(x_k|y_m)$.
De <u>voorwaardelijke entropie</u> ('equivocation') is

$$H(X|Y) = -\sum_{k=1}^{K} \sum_{m=1}^{M} P_{XY}(x_k, y_m)\, {}^2\log P_{X|Y}(x_k|y_m). \tag{1.3}$$

De voorwaardelijke entropie specificeert de gemiddelde hoeveelheid onzekerheid over X als Y bekend is.

<u>Opgave 2</u>
Ga na dat $H(X|Y) = H(X)$, als X en Y onafhankelijke variabelen zijn. Observatie van Y helpt in dat geval niet de onzekerheid over X te verminderen.

Voordat de wederzijdse informatie wordt behandeld, is het zinvol nog eerst de zelfinformatie van het paar (X,Y) te bepalen. De zelfinformatie over het voorval $[X = x_k, Y = y_m]$, opgevat als één gebeurtenis, wordt gegeven door $I_{XY}(x_k, y_m) = -{}^2\log P_{XY}(x_k, y_m)$. De <u>gezamenlijke entropie</u> of gemiddelde gezamenlijk

zelfinformatie van het paar (X,Y) is

$$H(X,Y) = -\sum_{k=1}^{K}\sum_{m=1}^{M} P_{XY}(x_k,y_m)^2 \log P_{XY}(x_k,y_m). \qquad (1.4)$$

Het is niet moeilijk om aan te tonen (doe dit!) dat de gezamenlijke entropie van het paar (X,Y) gelijk is aan de som van de entropie van X en de voorwaardelijke entropie van Y gegeven X. Als X en Y onafhankelijk zijn van elkaar reduceert dit tot $H(X,Y) = H(X) + H(Y)$.

De wederzijdse informatie is de hoeveelheid informatie die over het voorval $[X = x_k]$ wordt verschaft door het voorval $[Y = y_m]$, of omgekeerd. Men kan ook spreken over de hoeveelheid onzekerheid die door $[Y = y_m]$ over $[X = x_k]$ wordt weggenomen.
De wederzijdse informatie is gedefiniëerd als

$$I_{X;Y}(x_k;y_m) = -^2\log \frac{P_X(x_k) \cdot P_Y(y_m)}{P_{XY}(x_k,y_m)} \qquad (1.5)$$

waarbij verondersteld wordt dat de kansen P_X, P_Y en P_{XY} positief zijn.
Deze definitie kan nog worden uitgebreid met de volgende gevallen.

(i) $I_{X;Y}(x_k;y_m) = -\infty$ als $P_{XY}(x_k,y_m) = 0$ terwijl de marginale kansen P_X en P_Y positief zijn. Dit betekent dat het voorval $[X = x_k]$ niet kan plaatsvinden als $[Y = y_m]$ plaatsvindt.

(ii) $I_{X;Y}(x_k;y_m) = 0$ als $P_{XY}(x_k,y_m) = P_X(x_k) = P_Y(y_m) = 0$.

De wederzijdse informatie laat zich makkelijk interpreteren door relatie (1.2) in (1.5) te substitueren. Er volgt dan dat $I_{X;Y}(x_k,y_m) = -^2\log P_X(x_k)P^{-1}_{X|Y}(x_k|y_m)$.
Als de voorwaardelijke kans $P_{X|Y}(x_k|y_m)$ groter is dan de marginale kans $P_X(x_k)$ dan betekent dit dat de kans op voorval $[X = x_k]$ groter is geworden doordat voorval $[Y = y_m]$ heeft plaatsgevonden (is geobserveerd). Het voorval $[Y = y_m]$ levert dan een positieve bijdrage tot de informatie over het voorval $[X = x_k]$.
Als de voorwaardelijke kans gelijk is aan de marginale levert $[Y = y_m]$ geen informatie op: als de conditionele kans kleiner dan de marginale is levert $[Y = y_m]$ zelfs een negatieve bijdrage tot de informatie.

We zijn nu toe aan het belangrijkste informatiebegrip zowel voor communicatie- als cryptografische toepassingen. De <u>gemiddelde wederzijdse informatie</u> tussen

De gemiddelde wederzijdse informatie kan men opvatten als de gemiddelde hoeveelhei informatie over X die observatie van Y oplevert. Als X en Y onafhankelijke variabelen zijn volgt $I(X;Y) = 0$.
Door substitutie van relatie (1.1) en (1.3) in (1.6) verkrijgt men de belangrijke eigenschap

$$I(X;Y) = H(X) - H(X|Y). \tag{1.7}$$

Hiermee ziet men direkt dat de gemiddelde wederzijdse informatie opgevat kan worden als de gemiddelde onzekerheid over X die verdwijnt als alle voorvallen Y bekend zijn; de resterende onzekerheid over X wordt dan gegeven door de voorwaardelijke entropie $H(X|Y)$.

<u>Opgave 3</u>
Toon aan dat $I(X;Y) = H(Y) - H(Y|X)$.

Nog een kleine uitbreiding van het voorgaande is nodig om de toepassing van de geïntroduceerde begrippen op cryptografie te kunnen begrijpen. De voorwaardelijke wederzijdse informatie die over het voorval $[X = x_k]$ door voorval $[Y = y_m]$ wordt verschaft, onder de voorwaarde dat voorval $[Z = z_\ell]$ heeft plaatsgevonden, is

$$I_{X;Y}(x_k;y_m|z_\ell) = -{}^2\log \frac{P_{X|Z}(x_k|z_\ell) \cdot P_{Y|Z}(y_m|z_\ell)}{P_{XY|Z}(x_k,y_m|z_\ell)} . \tag{1.8}$$

De <u>gemiddelde conditionele wederzijdse informatie</u> tussen de stochastische variabelen X en Y gegeven Z wordt dan

$$I(X;Y|Z) = -\sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{\ell=1}^{L} P_{XYZ}(x_k,y_m,z_\ell)\ I_{X;Y|Z}(x_k;y_m|z_\ell). \tag{1.9}$$

Met een klein sprongetje vooruit kan men hierbij denken aan de gemiddelde wederzijdse informatie tussen boodschap X en cryptogram Y gegeven dat de sleutel Z tot het cryptogram bekend is.

Een <u>overzicht van de belangrijkste informatie-theoretische relaties</u> van belang voor cryptografie is hieronder gegeven.

$0 \leqq H(X) \leqq \log K$, met K het aantal waarden dat X kan aannemen.

$H(X) = \log K$ alleen als $P_X(x_k) = \frac{1}{K}$ voor $k = 1,2,\ldots,K$.

$I(X;Y) \geqq 0$, met gelijkheid als X en Y onafhankelijk zijn. (1.10)

$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = I(Y;X)$ (1.11)

$I(X;Y) = H(X) + H(Y) - H(X,Y)$ (1.12)

$I(X;Y|Z) \geqq 0$, met gelijkheid als X en Y voorwaardelijk onafhankelijk zijn gegeven Z. (1.13)

$I(X;Y|Z) = H(X|Z) - H(X|Y,Z)$ (1.14)

$I(X;Y,Z) = I(X;Y) + I(X;Z|Y)$ (1.15)

$H(X,Y) = H(X) + H(Y|X)$ (1.16)

$H(X,Y,Z) = H(X) + H(Y|X) + H(Z|X,Y)$ (1.17)

$H(X_1,X_2,\ldots,X_M) = H(X_1) + H(X_2|X_1) + \ldots + H(X_M|X_1,X_2,\ldots,X_{M-1})$ (1.18)

Notatie: De entropie van een vector $X_1,X_2,\ldots,X_M$ zal worden geschreven als $H(X^M) = H(X_1,X_2,\ldots,X_M)$.

<u>Voorbeeld 4</u>: Wederzijdse informatie over binair symmetrisch kanaal

Neem aan dat de uitslag X van het toss-experiment met een zuivere munt als een binaire 0 of 1 wordt genoteerd. Deze uitslag maken we nu bekend als Y via een apparaat dat met een kans ε, $0 \leqq \varepsilon \leqq 1$, een binaire 0 naar een 1 converteert (en omgekeerd). Figuur 3 schets de situatie. Bij iedere uitslag X wordt de schakelaar met kans ε in de stand inverteren geplaatst.

Deze configuratie wordt het binaire symmetrische kanaal genoemd omdat binaire digits aan de ingang met een bepaalde kans weer correct aan de uitgang verschijnen.

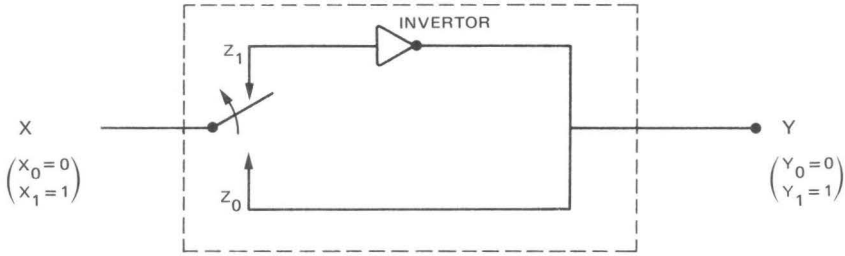


Fig. 3 Binair symmetrisch kanaal

Met $P_X(x_0) = P_X(x_1) = \frac{1}{2}$, $P_{Y|X}(y_1|x_1) = P_{Y|X}(y_0|x_0) = 1 - \varepsilon$ en $P_{Y|X}(y_1|x_0) = P_{Y|X}(y_0|x_1) = \varepsilon$ volgt middels relatie (1.2) dat $P_Y(y_0) = P_Y(y_1) = \frac{1}{2}$. Dus wordt $I(X;Y)$ na substitutie van deze kansen in (1.9) gelijk aan

$$I(X;Y) = 1 + \varepsilon\log\varepsilon + (1 - \varepsilon)\log(1 - \varepsilon) = 1 - H(\varepsilon) \qquad (1.19)$$

waarbij de tweede gelijkheid geldt naar analogie van Voorbeeld 2.
De funktie $1 - H(\varepsilon)$ is geschetst in figuur 4. Blijkbaar is $I(X;Y) = 0$ voor $\varepsilon = \frac{1}{2}$, hetgeen het geval is als de schakelaar met een kans van 50% de uitslag juist of niet juist doorgeeft.

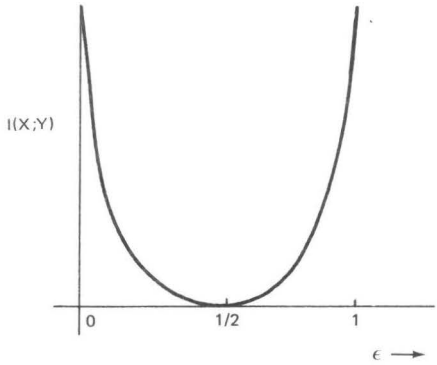


Fig. 4 $I(X;Y) = 1 - H(\varepsilon)$

$I(X;Y) = 0$ ofwel $H(X|Y) = H(X)$ betekent dat de kennis over Y niet helpt om de onzekerheid te verminderen. De gemiddelde overdracht van informatie over X via Y blijft nul: "X blijft geheim."

Stel nu dat de waarnemer van Y ook nog de toestand van de schakelaar Z zou weten dan kan hij een 'contra' apparaat zoals geschetst in figuur 5 maken, waarbij Z de schakelaar stuurt.

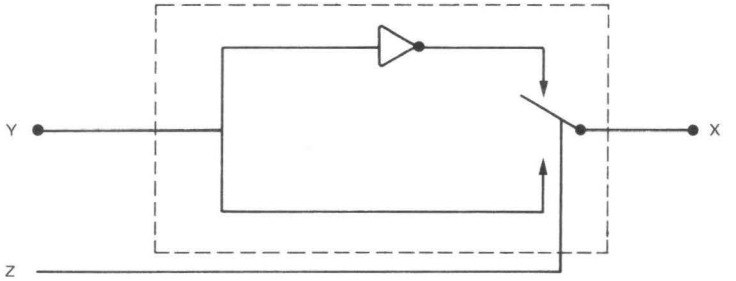


Fig. 5 Ontvanger met sleutelinformatie

De uitslag X kan nu geheel worden teruggewonnen uit de geobserveerde Y en de kennis over Z; de voorwaardelijke onzekerheid over X gegeven Y en Z is nul: $H(X|Y,Z) = 0$. Ofwel, $I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = H(X|Z) = H(X)$; de gemiddelde wederzijdse informatie tussen X en Y gegeven Z is precies voldoende om de onzekerheid over X te laten verdwijnen.

Voorbeeld 5: Redundantie Nederlandse taal

Zoals reeds vermeld in Voorbeeld 3, kan men een boodschap in de Nederlandse taal weergeven met minimaal 4.15 bit/letter als men alleen de letterfrequentie in aanmerking neemt. Relatie (1.18) geeft echter aan dat een boodschap bestaande uit M elementen, waarbij hier aan letters wordt gedacht, een entropie heeft die mede wordt bepaald door de onderlinge afhankelijkheid van die elementen (letters). Zonder onderlinge afhankelijkheid reduceert (1.18) tot $H(X^M) = H(X_1) + H(X_2) + \ldots + H(X_M) = MH(X_1)$ hetgeen neerkomt op $\frac{1}{M} \cdot MH(X_1) =$ 4.17 bit/letter. Door de frequentie van letterparen, drietallen etc. te meten heeft men gevonden dat $\frac{1}{2}H(X_1,X_2) \approx 3.57$ (bit/letter), $\frac{1}{3}H(X_1,X_2,X_3) \approx 3.23$, $\frac{1}{4}H(X_1,\ldots,X_4) \approx 2.98$ etc., tot $\lim_M \frac{1}{M} H(X_1,X_2,\ldots,X_M) \approx 1.5$ bit/letter.

men redundantie. De redundantie in een boodschap weergegeven als een binaire vector $X^M = [X_1, X_2, \ldots, X_M]$, met een entropie $H(X^M)$ is dus $M - H(X^M)$ hetgeen neerkomt op een redundantie per digit van $\rho = 1 - \frac{1}{M} H(X^M)$. Voor Nederlandse tekst is de redundantie $\rho \approx 0.7$ bij een representatie met 5 bits/letter. Geheel of gedeeltelijke verwijdering van de redundantie uit een boodschap noemt men (vervormingsloze) datacompressie. In hoeverre datacompressie praktische te verwezenlijken is valt buiten deze discussie. "Ideale" datacompressoren (tot $H(X^M)$) bestaan echter niet.

2 Shannon's cryptosysteem

In deze sectie zullen we de algemene inzichten uit de informatietheorie - zie vorige sectie - toepassen op het cryptografische probleem. Het model voor een cryptosysteem zoals voorgesteld door C.E. Shannon is weergegeven in figuur 6.

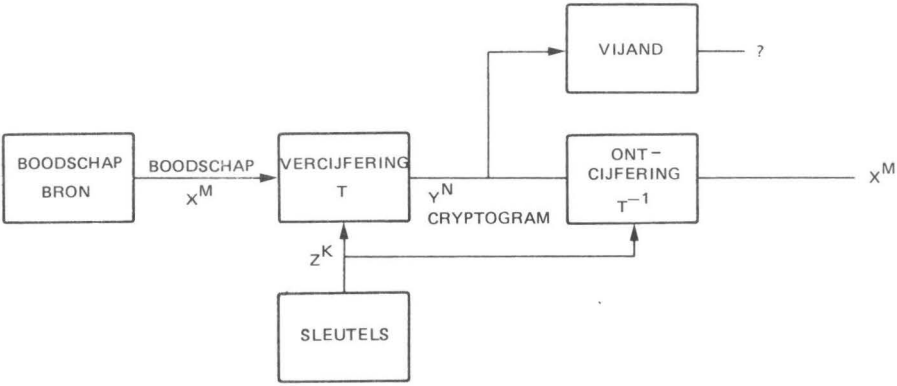


Fig. 6 Shannon's model van cryptosysteem

2.1 Absoluut-veilige cryptosystemen

Allereerst zullen we nagaan welke eisen aan een absoluut-veilig cryptosysteem moeten worden gesteld, aan de hand van figuur 6. De boodschap, het cryptogram en de gebruikte sleutel worden voorgesteld door respectievelijk de binaire vector $X^M = [X_1, X_2, \ldots, X_M]$, $Y^N = [Y_1, Y_2, \ldots, Y_N]$ en $Z^K = [Z_1, Z_2, \ldots Z_K]$. De vercijfer-operator T werkt op boodschap X^M onder afhankelijkheid van sleutel Z^K,

$$Y^N = T_{Z^K}(X^M). \tag{2.1}$$

De inverse ontcijferoperator T^{-1} haalt uit het cryptogram Y^N de originele boodschap X^M als althans Z^K bekend is. Dus

$$X^M = T^{-1}_{Z^K}(Y^N). \tag{2.2}$$

Hoewel in praktische systemen een sleutel Z^K vaak voor de vercijfering van een aantal boodschappen wordt gebruikt, zullen we in dit model aannemen dat de <u>sleutel Z^K na elke versleuteling van een boodschap X^M opnieuw wordt gekozen.</u> Daardoor kunnen we X^M, Y^N en Z^K per vercijfering als stochastische vectoren beschouwen en de op statistiek gebaseerde informatie-theoretische begrippen uitwerken.
Omdat de relatie (2.2) aangeeft dat cryptogram Y^N en sleutel Z^K op unieke wijze X^M bepalen is direkt vast te stellen dat de gemiddelde onzekerheid over de boodschap X^M gegeven het cryptogram en sleutel, nul moet zijn:

$$H(X^M|Y^N,Z^K) = 0. \tag{2.3}$$

Dit geldt in zijn algemeenheid voor cryptosystemen waarbij er hier verder vanuit wordt gegaan dat T en T^{-1} bekende (geen geheime) operatoren zijn.
Er is nog een <u>algemene relatie</u> over cryptosystemen af te leiden nl.

$$I(X^M;Y^N) \geqq H(X^M) - H(Z^K). \tag{2.4}$$

Dit bewijst men als volgt:

$$H(X^M,Z^K|Y^N) = H(Z^K|Y^N) + H(X^M|Z^K,Y^N) = H(Z^K|Y^N) \tag{2.4a}$$

vanwege relatie (1.16) en (2.3). Tevens geldt

$$H(X^M,Z^K|Y^N) = H(X^M|Y^N) + H(Z^K|X^M,Y^N) \geqq H(X^M|Y^N). \tag{2.4b}$$

Daarom is

$$H(X^M|Y^N) \leqq H(Z^K|Y^N) \leqq H(Z^K) \tag{2.4c}$$

wederzijdse informatie tussen cryptogram en boodschap: Het cryptogram levert veel informatie over de boodschap. Of het voldoende is voor ontcijfering is natuurlijk de vraag.

Een <u>absoluut-veilig</u> cryptosysteem wordt gedefinieerd als een systeem waarvoor geldt dat

$$I(X^M;Y^N) = 0. \tag{2.5}$$

Gemiddeld kan uit het cryptogram geen informatie over de boodschap worden onttrokken. Uiteraard heeft deze absolute veiligheid alleen betrekking op een analyse van het cryptogram zelf ('ciphertext-only-attack'). Met (2.4) en (2.5) volgt dat voor een absoluut-veilig cryptosysteem geldt dat

$$H(X^M) \leqq H(Z^K). \tag{2.6}$$

De entropie van de sleutel moet dus minstens zo groot zijn als de entropie van de boodschap. Ofwel de minimale lengte waarmee de sleutel kan worden gerepresenteerd (in de zin van informatieinhoud) moet dus minstens zo groot zijn als het aantal informatiebits in de boodschap. Als de sleutel willekeurig wordt gekozen uit een totaal van 2^K mogelijke sleutels met kans 2^{-K} dan moet $K \geqq H(X^M)$.

<u>Voorbeeld 6</u>: Absoluut-veilig cryptosysteem
Voor de eenvoud stellen we $M = N = K$. De boodschap X^N is de uitslag van N onafhankelijk toss-experimenten met een zuivere munt. Dus $H(X^N) = N$. Het vercijfersysteem is de schakelaar-met-invertor-configuratie zoals geschetst in figuur 3. De sleutel voor de vercijfering van X^N is de reeks van N standen van de schakelaar, met $\varepsilon = \frac{1}{2}$. Dan $H(Z^N) = N$. Het cryptogram is Y^N. Zoals reeds in Voorbeeld 4 werd aangetoond is de gemiddelde wederzijdse informatie $I(X^N;Y^N) = 1 - H(\frac{1}{2}) = 0$, dus voldoet aan de eis in (2.5). Inderdaad geldt ook (2.6), met gelijkheid: de sleutelentropie is net zo groot als de boodschapentropie. De klassieke benaming voor dit cryptosysteem is de "one-time-pad" omdat de veiligheid geheel berust op het slechts eenmaal gebruiken van een sleutel Z^N voor de boodschap X^N. Overigens is in praktische toepassingen het bezwaar niet zo zeer de kwetsbaarheid voor b.v. 'plain text attack' maar de enorme hoeveelheid

leutelinformatie die voor de vercijfering nodig is. Naast het gebruik in het ryptosysteem als zodanig moet de sleutelinformatie ook nog op een veilige wijze erspreid worden.

tel nu dat de binaire boodschap X^M een entropie $H(X^M) << M$ heeft, dus een grote edundantie heeft. Om een absoluut-veilig cryptosysteem te maken moet echter $(Z^K) \geq H(X^M)$ ofwel in het "one-time-pad" systeem is het voldoende als $K \approx H(X^M)$. e hoeveelheid sleutelinformatie in bits (K) kan dus in principe kleiner zijn dan e lengte van de boodschap (M) zolang de entropie van die boodschap maar kleiner an K is. Andersom is het ook zo dat bij een gegeven sleutelentropie de crypto-rafische veiligheid zal worden vergroot als de booodschap X^M voor de vercijfer-peratie in bits wordt gecomprimeerd tot een lengte die dichter bij $H(X^M)$ ligt. atacompressie vóór vercijferen vergroot dus de veiligheid.

.2 "Unicity distance" van niet absoluut-veilige cryptosystemen

ls $I(X^M;Y^N) > 0$ spreekt men over een niet absoluut-veilig cryptosysteem. aat men weer uit van het Shannon model voor een cryptosysteem met willekeurig ekozen boodschapvector X^M en willekeurig gekozen sleutel Z^K dan zal een crypto-ysteem niet absoluut-veilig zijn als de entropie van de sleutel kleiner is dan e boodschapentropie. De omvang van de hoeveelheid sleutels is dus beperkt en men an zich dan b.v. afvragen hoeveel cryptograminformatie men nodig heeft om de ebruikte sleutel te kunnen vaststellen met een redelijke zekerheid. Het minimale antal cryptogram digits dat nodig is om het cryptogram te kunnen breken noemt ien de unicity distance* van het cryptosysteem. Om de "unicity distance" te kunnen epalen beschouwde Shannon de gemiddelde onzekerheid over de sleutel gegeven de erste n digits van het cryptogram Y^N. Zodra deze 'sleutelonzekerheid',

$$H(Z^K|Y^n) = H(Z^K|Y_1,Y_2,\ldots,Y_n) \approx 0, \quad 0 \leq n \leq N \tag{2.7}$$

an men het cryptogram als gebroken beschouwen. Hierbij is $H(Z^K|Y^n)$ een niet-tijgende funktie in n. Nu geldt ook dat

$$H(X^M,Y^N|Z^K) = H(X^M|Z^K) + H(Y^N|Z^K,X^M) = H(X^M) \tag{2.8}$$

Verder geldt ook dat

$$H(X^M,Y^N|Z^K) = H(Y^N|Z^K) + H(X^M|Y^N,Z^K) = H(Y^N|Z^K). \tag{2.9}$$

Uit (2.8) en (2.9) volgt dat

$$H(Y^N|Z^K) = H(X^M), \tag{2.10}$$

de onzekerheid in de boodschap is gelijk aan de onzekerheid in het cryptogram als de sleutel bekend is. Voor de door Shannon beschouwde systemen bleek dat, voor <u>n niet te groot</u>, de eerste n bits van een cryptogram er geheel willekeurig uitzien ofwel $H(Y_1,Y_2,\ldots,Y_n) \approx n$. Tevens zal voor typische cryptosystemen gelden dat de onzekerheid over Y^n gegeven de sleutel ook ongeveer met n zal toenemen tot $H(X^M)$ dus

$$H(Y^n|Z^K) \approx \frac{n}{N} H(X^M). \tag{2.11}$$

Met relatie (1.7) volgt nu

$$\begin{aligned} H(Y^n|Z^K) &= H(Z^K) - I(Z^K;Y^n) \\ &= H(Z^K) - H(Y^n) + H(Y^n|Z^K) \\ &= H(Z^K) - n + \frac{n}{N} H(X^M) \end{aligned}$$

dus

$$H(Y^n|Z^K) \approx H(Z^K) - \rho n \tag{2.12}$$

waarbij $\rho = 1 - \frac{1}{N} H(X^M)$ de boodschapredundantie per digit cryptogram aangeeft. De waarde van ρ wordt klein als $H(X^M)$ dichtbij het aantal cryptogram digits komt, dus als de boodschap eerst gecomprimeerd wordt.
De relatie (2.12) geeft aan dat de gemiddelde onzekerheid over de sleutel afneemt naarmate meer digits cryptogram bekend worden. Omdat (2.12) nog voor

tamelijk kleine waarden blijkt te gelden voor vele cryptosystemen kunnen we nu voor de "unicity distance" schrijven

$$\text{"unicity distance"} \approx \frac{H(Z^K)}{\rho}. \qquad (2.13)$$

Voorbeeld 7: "Unicity distance"
Voor een transpositie cryptosysteem met periode 26 (en een 'random' sleutel voor ieder boodschap) geldt een sleutelentropie $H(Z^K) = {}^2\log 26! = 88.4$ bits. Als $N = M$ volgt $\rho = 1 - \frac{1}{M} H(X^M)$, de redundantie per boodschap digit. Voorbeeld 5 gaf reeds aan dat $\rho \approx 0.7$ voor Nederlandse tekst. Dan volgt met (2.13) dat de "unicity distance" ongeveer $\frac{88.4}{0.7} \approx 126$ bits bedraagt. Na slechts de helft meer bits dan de lengte van de sleutel is het systeem in theorie gebroken omdat de oplossing van het cryptogram uniek wordt.

Als de redundantie ρ naar nul gaat, dus als een ideale datacompressor in het cryptosysteem is opgenomen, dan neemt de "unicity distance" inderdaad grote waarden aan. Dit garandeert echter niet dat het cryptosysteem dan absoluut-veilig is geworden. Een vereiste voor absolute veiligheid is nl. dat $H(X^M) \leqq H(Z^K)$ en dat is door de beperkte hoeveelheid sleutelinformatie niet het geval.
Er zullen steeds $2^{H(Z^K|Y^n)}$ verschillende oplossingen voor de sleutel zijn, hoe groot n ook wordt, en een even groot aantal geldige boodschappen die bij het cryptogram passen. Een daarvan is dan de geldige oplossing.
In figuur 7 is een schets van $H(Z^K|Y^n)$ gegeven voor Shannon's stelsel van willekeurig gekozen cryptografische vercijferingen ("random cipher model").

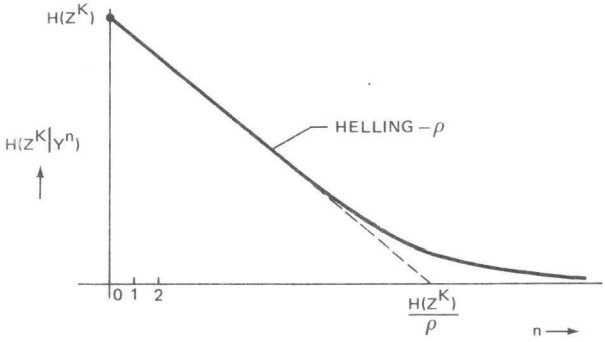


Fig. 7 Sleutelonzekerheid voor 'random cipher' model

distance" toch niet op te lossen is. Dit komt niet alleen doordat "unicity distance" zelf relatief groot kan zijn maar ook doordat het model met willekeurige sleutels ('random cipher') in de praktijk niet de beste cryptosystemen oplevert. Er zijn systemen die veel betere prestaties (in de vorm van vereiste hoeveelheid rekenwerk) leveren ondermeer door de vercijfering aan te passen aan de boodschap. Het verder niet in de beschouwing betrekken van deze hoeveelheid rekenkundige complexiteit is een fundamentele zwakte van de informatie- theoretische benadering die echter wel degelijk door Shannon werd onderkend. Hij suggereerde de term "praktische veiligheid" om de cryptografische veiligheid berustend op een grote rekenkundige complexiteit aan te duiden.
Figuur 8 geeft een schets van de hoeveelheid werk W_n die nodig is om een niet absoluut-veilig cryptosysteem te breken, als funktie van het aantal digits n die van een cryptogram bekend zijn.

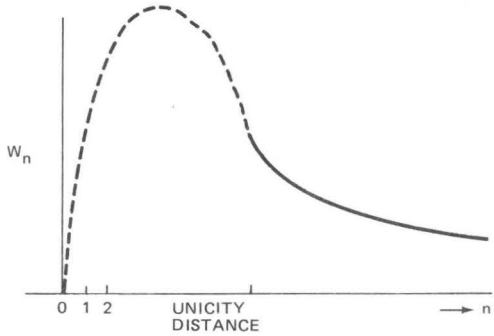


Fig. 8 Typische hoeveelheid werk om cryptogram te breken

In het gestippelde gedeelte neemt W_n eerst sterk toe omdat in dat gedeelte de oplossing van de sleutel nog niet uniek is; er zijn nog vele mogelijke oplossingen en die moeten allemaal bepaald worden. In het gedeelte voorbij de "unicity distance" is de oplossing wel uniek maar zal het nog een aanzienlijke hoeveelheid werk vergen om die oplossing te vinden. Als meer cryptografische informatie beschikbaar komt zal het steeds makkelijker worden de oplossing te vinden.

Voor de "praktische veiligheid" van cryptosystemen is het bepalen van de rekenkundige complexiteit die nodig is om het systeem te breken van groot belang. De "theoretische" benadering van de klassieke informatietheorie kan daarnaast een waardevolle bijdrage leveren tot een beter inzicht in de funktie van cryptosystemen

3 Referenties

Bij het samenstellen van dit hoofdstuk werd gebruik gemaakt van een aantal "sleutel" referenties die ook de lezer kunnen helpen bij het verder uitwerken van de geintroduceerde begrippen.

1. Shannon, C.E., "Communication Theory and Secrecy Systems", BSTJ, Vol. 28, pp. 656-715, October 1949.

2. Gallager, R.G. Information Theory and Reliable Communication. New York: John Wiley and Sons, 1968.

3. Hellmann, M.E., "An Extension of the Shannon Theory Approach to Cryptography", IEEE Transactions on Information Theory, Vol. IT-23, No. 3, May 1977.

4. Omura, J.K. and Massey, J.L., "Principles of secure digital communication: modulation, coding, spectrum spreading and cryptography", Lecture notes, Continuing Education Institute, Zurich 1981.

5. Van Soest, J.L., Informatie- en communicatietheorie, Delft: ETV, 1962.

KLASSIEKE CRYPTOGRAFISCHE SYSTEMEN

Inleiding

Dit overzicht van cryptografische systemen zal zich beperken tot systemen stammend uit het vóórcomputer tijdperk. Grofweg worden zo de systemen ontstaan in de periode tot en met de tweede wereldoorlog tot de klassieke systemen gerekend. Vanzelfsprekend kan een klassiek systeem op een computer geimplementeerd worden. Van origine zijn het echter 'potlood en papiermethoden' of middels (electro)mechanische apparatuur bewerkstelligde geheimschriften. Toepassing door middel van een computer schept wel mogelijkheden om het geheimschrift te compliceren, maar maakt het niet wezenlijk sterker.

In de loop der eeuwen zijn talloze methoden bedacht voor het verbergen van informatie. Uit deze veelheid zal noodzakelijkerwijze slechts een bescheiden greep gedaan kunnen worden. Aan exotische methoden gaan we voorbij. Het kaalscheren van het hoofd van een slaaf vervolgens daarop aanbrengen van de boodschap en na aangroei van het haar zenden aan geadresseerde, is heden ten dage niet meer zo praktisch. Onze behandeling zal zich verder beperken tot de methoden die een rij tekens, tezamen een bericht vormend, omzetten in een andere rij tekens. De ontvanger van het cryptogram kan de oorspronkelijke rij reconstrueren met behulp van de sleutel van het cryptogram. In de klassieke systemen bestaat de gebruikte tekenverzameling als regel uit de letters A t/m Z en/of de cijfers 0 t/m 9. Bij de letters wordt geen onderscheid gemaakt tussen boven- en onderkast. Spaties en leestekens laat men achterwege of schrijft ze voluit.

Een geschiedenis van geheimschriften en cryptanalyse kan men vinden bij [Kahn]. Enige inleidende teksten op het gebied van geheimschriften

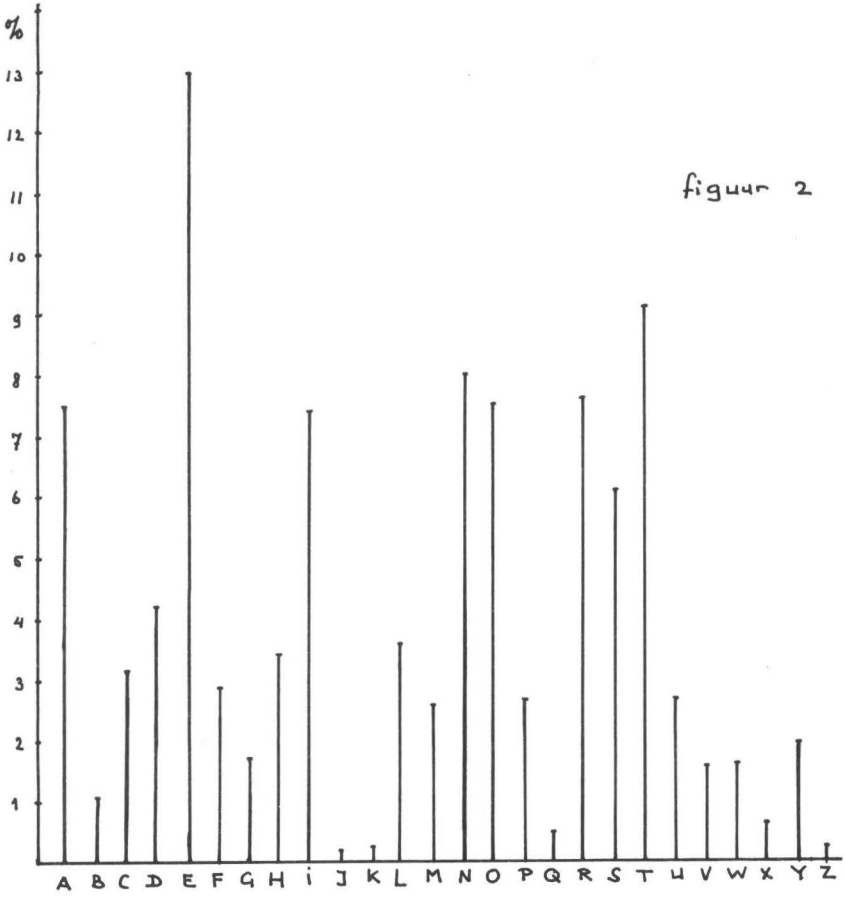
Taalkenmerken

De behandeling van een cryptografisch systeem behoort niet slechts aandacht te schenken aan de methode van vercijfering. Minstens zo belangrijk is het om een idee te krijgen omtrent de kwetsbaarheid van het systeem. Bij veel klassieke systemen is de kwetsbaarheid van het systeem gelegen in de onvolledige versluiering van de taalkarakteristieken van het vercijferde materiaal. Daarom zal allereerst de aandacht gericht worden op dat belangrijke stuk gereedschap van de cryptanalist, de kenmerken van de taal.

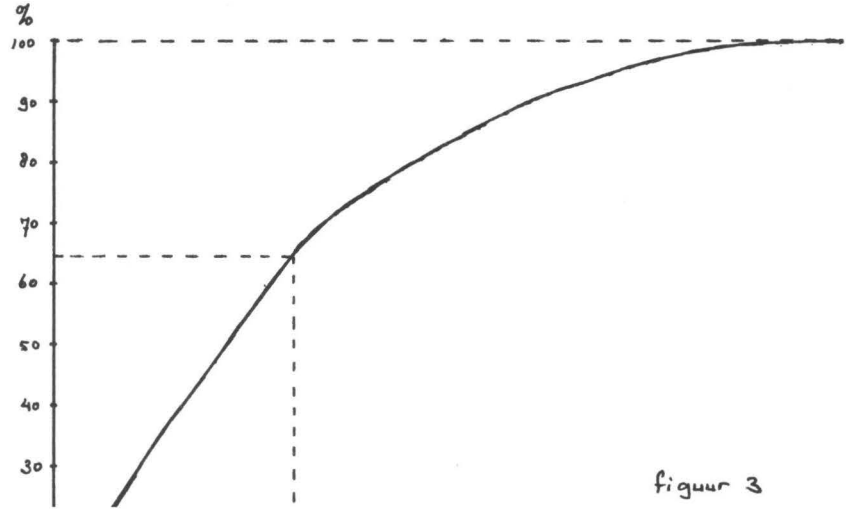
Telt men de letters in een stuk tekst dan zal blijken dat niet alle letters evenvaak voorkomen. De precieze details verschillen van taal tot taal. Er zijn ook verschillen tussen bijvoorbeeld technische artikelen en zakencorrespondentie. Gegeven echter voldoende onderliggend tekstmateriaal, is het mogelijk een redelijk nauwkeurig beeld te verkrijgen van de voor dat materiaal karakteristieke verdeling van letterfrequenties. Bij wijze van voorbeeld wordt in figuur 1 de procentuele verdeling gegeven van de letters in de engelse taal (afgeleid uit [Kullback]). In figuur 2 is deze verdeling grafisch uitgebeeld; in figuur 3 is de cumulatieve verdeling van de letterfrequenties gegeven, met daarbij de letters in volgorde van afnemende frequentie.

A	7.4	H	3.4	O	7.5	V	1.5
B	1.0	I	7.4	P	2.7	W	1.6
C	3.1	J	0.2	Q	0.4	X	0.5
D	4.2	K	0.3	R	7.6	Y	1.9
E	13.0	L	3.6	S	6.1	Z	0.1
F	2.8	M	2.5	T	9.2		
G	1.6	N	8.0	U	2.6		

figuur 1



figuur 2



figuur 3

letters. Verschillen in tekstmateriaal, spellingswijziging en dergelijke leiden hoogstens tot kleine verschuivingen in de onderlinge volgorde. Te zien is dat slechts een achttal letters (E T N R O A I S) zo'n tweederde van het tekstvolume voor zijn rekening neemt. Men kan zich voorstellen, dat identifikatie van dit achttal bij een cryptanalyse gelijkstaat met het ontcijferen van het geheimschrift. Aan de hand van de frequentieverdeling maakt men de volgende indeling van de letters (figuur 4).

klinkers	A E I O U Y	39.8%
medeklinkers veel	D N R S T	35.1%
medeklinkers midden	B C F G H L M P V W	23.8%
medeklinkers zelden	J K Q X Z	1.4%

figuur 4

De klinkers vormen tezamen ongeveer 40% van het tekstvolume. Bekijkt men de plaats van de klinkers in een tekst, dan zal snel blijken dat klinkers vaker geflankeerd worden door een medeklinker dan door een klinker. In de engelse taal bijvoorbeeld gaat in slechts 12-20% van de gevallen een klinker vooraf aan of volgt op een andere klinker. Slechts de letter die aan de U voorafgaat maakt hierop een uitzondering. Dit verschijnsel kan nader worden onderzocht door niet naar het voorkomen van enkele letters te kijken, maar door alle 676 tweelettergroepen uit te tellen. In figuur 5 wordt deze verdeling voor het engels gegeven (naar [Kullback]). Ook hier valt de ongelijke verdeling op. Van de 676 mogelijke tweelettergroepen of digrammen, komt men slechts tweederde tegen. Voor de combinatie van drie letters, de trigrammen, geldt iets soortgelijks.

Frequency distribution of digraphs—Based on 50,000 letters of Government plain-text telegrams; reduced to 5,000 digraphs

SECOND LETTER

FIRST LETTER	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	3	6	14	27	1	4	6	2	17	1	2	32	14	64	2	12		44	41	47	13	7	3		12	
B	4				18				2	1		6	1		4			2	1	1	2				7	
C	20		3	1	32	1		14	7		4	5	1	1	41			4	1	14	4		1		1	
D	32	4	4	8	33	8	2	2	27	1		3	5	4	16	5	2	12	13	15	5	3	4		1	
E	35	4	32	60	42	18	4	7	27	1		29	14	111	12	20	12	87	54	37	3	20	7	7	4	1
F	5		2	1	10	11	1		39			2	1		40	1		9	3	11	3		1		1	
G	7		2	1	14	2	1	20	5	1		2	1	3	6	2		5	3	4	2		1			
H	20	1	3	2	20	5			33			1	2	3	20	1	1	17	4	28	8		1		1	
I	8	2	22	6	13	10	19				2	23	9	75	41	7		27	35	27		25		15		2
J	1				2										2						2					
K	1		1		6				2			1		1					1							
L	28	3	3	9	37	3	1	1	20			27	2	1	13	3		2	6	8	2	2	2		10	
M	36	6	3	1	26	1		1	9				13		10	8		2	4	2	2				2	
N	26	2	19	52	57	9	27	4	30	1	2	5	5	8	18	3	1	4	24	82	7	3	3		5	
O	7	4	8	12	3	25	2	3	5	1	2	19	25	77	6	25		64	14	19	37	7	8	1	2	
P	14	1	1	1	23	2		3	6			13	4	1	17	11		18	6	8	3	1	1		1	
Q													1					1			15					
R	39	2	9	17	98	6	7	3	30	1	1	5	9	7	28	13		11	31	42	5	5	4		9	
S	24	3	13	5	49	12	2	26	34		1	2	3	4	15	10		5	19	63	11	1	4		1	
T	28	3	6	6	71	7	1	78	45			5	6	7	50	2	1	17	19	19	5		36		41	1
U	5	3	3	3	11	1	8		5			6	5	21	1	2		31	12	12		1				
V	6				57				12						1					1						
W	12				22			4	13			1		2	19			1	1						1	
X	2		2	1	1	1		1	2					1	1	2		1	1	7						
Y	6	2	4	4	9	11	1	1	3			2	2	6	10	3		4	11	15	1		1			
Z	1				2				1																	

uit: S.Kullback. Statistical Methods in Cryptanalysis.
Aegean Park Press

De eerste ondervedeling die kan worden aangebracht in de klassieke geheimschriften, is die tussen transpositie- en substitutiemethoden. Een transpositie laat de tekens van het bericht ongemoeid, maar verandert hun volgorde. Een substitutie substitueert andere tekens voor de oorspronkelijke, maar laat de volgorde ongewijzigd. Het onderscheid tussen beide methoden leidt al direct tot een eerste toepassing van de letterfrequenties. Een transpositie vertoont de normale letterfrequenties van de gebezigde taal, een substitutie geeft een geheel ander beeld. Gevallen waarin een transpositie met een substitutie gecombineerd wordt, laten we voor het gemak buiten beschouwing.

Route Transpositie

De eenvoudigste transpositiemethode is de Route Transpositie. Hierbij schrijft men de te vercijferen tekst in een rechthoek. Vervolgens neemt men de letters er in een vaste, maar andere volgorde weer uit. Als voorbeeld nemen we de tekst

EEN CURSUS VAN HET MATHEMATISCH CENTRUM XX

De twee X'en aan het eind zijn bedoeld om het aantal letters aan te vullen tot het totaal van de gekozen rechthoek. Voor deze rechthoek is hier een 6x6 vierkant gekozen. Figuur 6 geeft de ingeschreven tekst.

E	E	N	C	U	R
S	U	S	V	A	N
H	E	T	M	A	T
H	E	M	A	T	I
S	C	H	C	E	N
T	R	U	M	X	X

figuur 6

De tekst van het geheimschrift kan nu op allerlei manieren uit figuur 6 gehaald worden. Bijvoorbeeld verticaal te beginnen in de

geschiedt door het met behulp van de sleutel weer in de goede volgorde plaatsen van de kolommen.

De sleutel is een rij getallen die aangeeft welke kolommen achtereenvolgens moeten worden genomen. Bij zes kolommen kan dat zijn 4-1-6-3-2-5; dwz neem eerst de tweede kolom, dan de vijfde, enz. Om deze sleutel gemakkelijk te kunnen onthouden wordt ze afgeleid uit een eenvoudig te memoriseren sleutelwoord. De alfabetische volgorde van de letters uit het sleutelwoord bepaalt de volgorde van de kolommen. Zij het sleutelwoord bijvoorbeeld MASKER. De letter A op de tweede plaats is van alle letters uit het sleutelwoord degene die het meest vooraan staat in het alfabet. De tweede kolom wordt daarom het eerst afgenomen. De eerstvolgende letter in het alfabet, die ook in het sleutelwoord voorkomt, is de E op plaats vijf: de vijfde kolom wordt als tweede afgenomen, etc. Op deze wijze ontstaat de sleutel 4-1-6-3-2-5. In figuur 7 wordt deze methode toegepast op de voorbeeldzin.

sleutel:	M	A	S	K	E	R
volgorde:	4	1	6	3	2	5
tekst:	E	E	N	C	U	R
	S	U	S	V	A	N
	H	E	T	M	A	T
	H	E	M	A	T	I
	S	C	H	C	E	N
	T	R	U	M	X	X

cryptogram: EUEEC RUAAT EXCVM ACMES HHSTR NTINX NSTMH U

figuur 7

De cryptanalyse van dit geheimschrift vergt twee stappen. In de eerste plaats moet bepaald worden welke rechthoek gediend heeft voor het afnemen van de kolommen. Immers elk product n x m = l met l het aantal letters van het cryptogram, kan voor de vercijfering hebben dienst gedaan. We gaan dan nog voorbij aan gemakkelijk te realiseren

rechterondertoek, afwisselend van beneden naar boven en omgekeerd.

XNITN RUAAT EXMCA MVCNS TMHUR CEEUE ESHHS T

Zoals gebruikelijk in de cryptografie is de vercijferde tekst in groepen van vijf letters uitgeschreven. Nog twee voorbeelden van het afnemen van een tekst. Eerst diagonaalsgewijs steeds van links onder naar rechts boven.

ESEHU NHESC SETVU TCMMA RRHAA NUCTT MEIXN X

Deze spiraalvormig vanuit het midden, draaiend tegen de wijzers van de klok in:

MAMTE ECHCE TAAVS USHHS TRUMX XNITN RUCNE E

Andere mogelijkheden zijn het gebruik van magische vierkanten, paardsprongen, etc. Het totaal aantal mogelijke routes is echter vrij beperkt in aantal.

Cryptanalyse van een route transpositie is niet echt moeilijk. Uitgaande van een aantal standaardroutes en de mogelijke verdelingen van de tekst in rechthoeken, kan men gaan proberen. Zelfs al zit de toegepaste route niet in het standaardpakket, dan nog zullen vaak brokjes leesbare tekst verschijnen. Met deze brokjes als gids wordt alras de gevolgde route duidelijk.

De route transpositie geeft op zich dus een zeer geringe bescherming. Ze biedt echter wel een gemakkelijke manier om in een meerstapsproces een met een andere methode vercijferd bericht verder te compliceren.

Kolom Transpositie

In het geval van de route transpositie ligt het patroon bij het afnemen van het cryptogram geheel vast. Bijvoorbeeld verticaal afnemen betekent: eerst kolom 1, dan kolom 2, enzovoorts. Dit vaste patroon leidt tot het betrekkelijk geringe aantal routes, die gemakkelijk uitputtend afgezocht kunnen worden. Het is echter ook mogelijk om de volgorde waarin de kolommen afgenomen worden te varieren. Deze volgorde vormt de sleutel tot de transpositie. Ontcijfering

In het voorbeeld bestaat het cryptogram uit 36 letters. De mogelijke basisrechthoeken zijn derhalve 2 x 18, 3 x 12, 4 x 9, 6 x 6, 9 x 4, 12 x 3 en 18 x 2. Een verdeling 9 x 4 verschilt niet erg van 4 x 9; het komt neer op kijken naar rijen in plaats van naar kolommen. Bij de bepaling van de juiste verdeling, cq het uitschakelen van onwaarschijnlijke verdelingen, komen de taalkenmerken te hulp. We zagen reeds dat klinkers zich graag met medeklinkers omringen. Dit impliceert dat de klinkers vrij regelmatig over de tekst verspreid zitten. Daarom is die verdeling in kolommen het meest waarschijnlijk, die de regelmatigste klinkerverdeling geeft. Nota bene: bij zeer korte tekstfragmenten zijn uitzonderingen te verwachten.

In de volgende figuur zijn voor de verdelingen 4 x 9 en 6 x 6 in elke rij en kolom de aantallen klinkers geteld. De verdelingen 2 x 18 en 3 x 12 zijn a priori niet zo waarschijnlijk.

4 x 9

E	U	E	E	C	R	U	A	A	7
T	E	X	C	V	M	A	C	M	2
E	S	H	H	S	T	R	N	T	1
I	N	X	N	S	T	M	H	U	2
3	2	1	1	0	0	2	1	2	

6 x 6

E	U	E	E	C	R	4
U	A	A	T	E	X	4
C	V	M	A	C	M	1
E	S	H	H	S	T	1
R	N	T	I	N	X	1
N	S	T	M	H	U	1
3	2	2	3	1	1	

figuur 8

De kolommen uit de 6 x 6 verdeling geven de meest gelijkmatige klinkerverdeling. De zes letters van deze kolommen stammen dus waarschijnlijk uit zes opvolgende letters van de oorspronkelijke tekst. De rijen uit figuur 8 zijn dan de verwisselde kolommen van het geheimschrift. Het loont de moeite om met deze indeling de volgende stap aan te vangen: het in de juiste volgorde leggen van de kolommen.

Bij het in de juiste volgorde brengen van de kolommen uit figuur 9 bewijst een compilatie van digramfrequenties zoals in figuur 5 goede diensten. Kolommen worden samengelegd die een groot aantal veel voorkomende digrammen opleveren. Bepaalde letters, zoals de Q die bijna uitsluitend door de U gevolgd wordt, kunnen daarbij als gidsletter dienen. Ook kan worden gezocht naar veel voorkomende voor- en achtervoegsels en lidwoorden (-ion, the in het engels). In figuur 9 springt het lidwoord EEN in de bovenste rij snel in het oog als een mogelijke combinatie. Bij proberen van de beide mogelijkheden ziet de rechter er het meest gezond uit. Uitbreiden van de zichtbare woordfragmenten met de overblijvende kolommen is dan een koud kunstje.

E	U	C	E	R	N	E	E	N	E	E	N
U	A	V	S	N	S	U	S	S	S	U	S
E	A	M	H	T	T	E	H	T	H	E	T
E	T	A	H	I	M	E	H	M	H	E	M
C	E	C	S	N	H	C	S	H	S	C	H
R	X	M	T	X	U	R	T	U	T	R	U

figuur 9

Kolom transpositie met ongelijke kolommen

Het oplossen van een kolom transpositie wordt gemakkelijk gemaakt door het feit dat alle kolommen van gelijke lengte zijn. Hierdoor is, gegeven de afmetingen van de gebruikte rechthoek, precies bekend waar iedere kolom begint en eindigt. Door nu met een aantal kolommen van ongelijke lengte te werken, en deze in willekeurige volgorde achter elkaar te zetten, wordt extra onzekerheid geintroduceerd omtrent begin en eind van elke kolom. De kolomtranspositie met ongelijke kolommen levert daarom een sterker geheimschrift op, dan een met kolommen van dezelfde lengte.

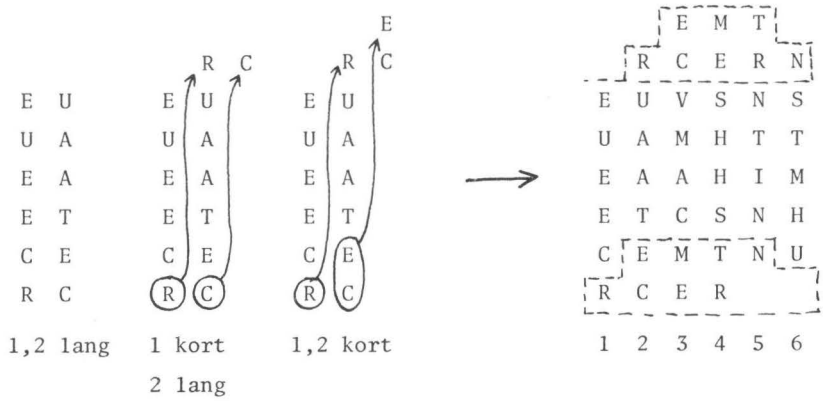
Ter illustratie wederom dezelfde tekst, nu echter zonder de twee aanvullende X'en. De tekst wordt opnieuw in een 6 x 6 rechthoek uitgeschreven en met dezelfde sleutel afgenomen.

sleutel:	4	1	6	3	2	5
tekst:	E	E	N	C	U	R
	S	U	S	V	A	N
	H	E	T	M	A	T
	H	E	M	A	T	I
	S	C	H	C	E	N
	T	R	U	M		

cryptogram: EUEEC RUAAT ECVMA CMESH HSTRN TINNS TMHU

figuur 10

De cryptanalist heeft nu rekening te houden met de onzekerheid in kolomlengte. Gegeven de rechthoekafmetingen 6 x 6, volgt uit de lengte van het cryptogram (34 letters) een verdeling in vier lange kolommen van zes letters en twee korte kolommen van elk vijf letters. In figuur 11 is deze onzekerheid uitgebeeld door middel van streepjeslijnen. Als de eerste kolom een korte kolom zou blijken, moet de letter onder verhuizen naar de top van de volgende kolom. Daarop verhuist vanzelf de onderste letter van de tweede kolom naar de top van de derde, enz.



figuur 11

Wanneer kolommen bijeen gesprokkeld worden op zoek naar de

de kolommen 3 en 2. In de combinaties V en VI gaat een korte kolom vooraf aan een lange. Deze situatie doet zich voor als de korte kolom uiterst rechts in de rechthoek stond en de lange kolom uiterst links. In dat geval moet de korte kolom één plaats zakken om op de juiste plaats aan te sluiten. Selectie van de juiste mogelijkheid geschiedt door te zoeken naar een combinatie met een hoog percentage aan veel voorkomende digrammen. Gezocht wordt naar gidsletters, achtervoegsels, lidwoorden en andere waarschijnlijke woorden. Een extra houvast wordt geboden door het feit dat van de eerste kolom uit het cryptogram het begin vaststaat en van de laatste het eind.

	I		II		III		IV		V		VI		VII	
	3	2	3	2	3	2	3	2	3	2	3	2	3	2
	E	R	C	U	C	U	C	R		R		U	V	U
	C	U	V	A	V	A	V	U	C	U	V	A	M	A
	V	A	M	A	M	A	M	A	V	A	M	A	A	A
	M	A	A	T	A	T	A	A	M	A	A	T	C	T
	A	T	C	E	C	E	C	T	A	T	C	E	M	E
	C		M				M	E	C	E	M	C	E	C
kort:	1,2		2		2,3		1		1,3		3		-	
lang:	3		1,3		1		2,3		2		1,2		1,2,3	

figuur 12

Tweevoudige kolomtranspositie

Een formidabele transpositie ontstaat wanneer het proces van de kolomtranspositie met ongelijke kolommen twee keer achterelkaar wordt toegepast. In figuur 13 wordt dit geillustreerd; ditmaal met een 4 x 9 rechthoek als basis. Cryptanalyse van dit geheimschrift komt neer op een anagram van de letters tot begrijpelijke tekst ontstaat. Zo'n proces leidt niet gemakkelijk of snel tot succes. Een zwakke plek verschijnt echter wanneer verschillende boodschappen van dezelfde lengte en met dezelfde sleutel vercijferd worden gevonden. In deze verschillende boodschappen zijn de letters op dezelfde manier

van plaats verwisseld. De anagram techniek kan dus tegelijkertijd op verschillende teksten worden losgelaten. De kans op succes stijgt hiermee enorm.

sleutel:	H A R I N G T O N		H A R I N G T O N
volgorde:	3 1 8 4 5 2 9 7 6		3 1 8 4 5 2 9 7 6
tekst:	E E N C U R S U S		E A E E R T I U E
	V A N H E T M A T	→	V H C C H A T U E
	H E M A T I S C H		T R S T H U A C N
	C E N T R U M		N M N S M S M

cryptogram: AHRMT AUSEV TNECT SRHHM EENUU CECSN ITAM

figuur 13

Substitutie Methoden

In geheimschriften gebaseerd op substitutie worden de tekens van het bericht vervangen door andere tekens. Des te beter de substitutie slaagt in het versluieren van de taalkenmerken, des te sterker in de regel het geheimschrift. De gesubstitueerde eenheid kan één, twee of meer tekens tegelijk vervangen. Worden gehele lettergrepen, woorden en zinsdelen als een geheel vervangen, dan spreekt men van een code.

De systemen die teken voor teken vervangen, kunnen worden beschreven met een substitutie alfabet. Dwz elke letter uit de te vercijferen tekst wordt vervangen door de overeenkomstige letter uit het substitutie alfabet (zie figuur 14). De eenvoudigste substitutiemethoden hebben slechts één substitutiealfabet. Ingewikkelder systemen maken gebruik van verscheidene tot zeer vele van deze alfabetten.

normale alfabet: $[\ a_1\ ,\ a_2\ ,\ a_3\ ,\ \ldots\ ,\ a_{26}\ ]$
substitutie alfabet: $[\ b_1\ ,\ b_2\ ,\ b_3\ ,\ \ldots\ ,\ b_{26}\ ]$

De eenvoudigste substitutie is de Caesar substitutie, genoemd naar Julius Caesar. Het substitutie alfabet bestaat uit een verschoven normaal alfabet. In figuur 15 is een verschuiving over 5 plaatsen toegepast. Men noteert deze substitutie wel als

$$b_t = C^i \, a_t$$

Hierin is a_t de t^e-letter uit de te vercijferen tekst, b_t de t^e-letter van het resulterend cryptogram, C het symbool voor de Caesar substitutie en i het getal dat de verschuiving van het substitutie- tov het gewone alfabet aangeeft.

```
normaal:   A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Caesar 5:  F G H I J K L M N O P Q R S T U V W X Y Z A B C D E

tekst:        EENCU RSUSV ANHET MATHE MATIS CHCEN TRUM
cryptogram:   JJSHZ WXZXA FSMJY RFYMJ RFYNX HMHJS YWZR
```

figuur 15

Cryptanalyse van dit geheimschrift is kinderlijk eenvoudig. Aangezien er maar 26 verschillende substitutie alfabetten zijn, staat niets een uitputtende zoekpartij in de weg. In figuur 16 is dit gedaan voor de eerste 10 letters van het cryptogram van figuur 15. De regel met de oorspronkelijke tekst springt er vanzelf uit.

J	J	S	H	Z	W	X	Z	X	A
K	K	T	I	A	X	Y	A	Y	B
L	L	U	J	B	Y	Z	B	Z	C
M	M	V	K	C	Z	A	C	A	D
N	N	W	L	D	A	B	D	B	E
O	O	X	M	E	B	C	E	C	F
P	P	Y	N	F	C	D	F	D	G
Q	Q	Z	O	G	D	E	G	E	H
R	R	A	P	H	E	F	H	F	I
S	S	B	Q	I	F	G	I	G	J
T	T	C	R	J	G	H	J	H	K
U	U	D	S	K	H	I	K	I	L
V	V	E	T	L	I	J	L	J	M

W	W	F	U	M	J	K	M	K	N
X	X	G	V	N	K	L	N	L	O
Y	Y	H	W	O	L	M	O	M	P
Z	Z	I	X	P	M	N	P	N	Q
A	A	J	Y	Q	N	O	Q	O	R
B	B	K	Z	R	O	P	R	P	S
C	C	L	A	S	P	Q	S	Q	T
D	D	M	B	T	Q	R	T	R	U
E	E	N	C	U	R	S	U	S	V
F	F	O	D	V	S	T	V	T	W
G	G	P	E	W	T	U	W	U	X
H	H	Q	F	X	U	V	X	V	Y
I	I	R	G	Y	V	W	Y	W	Z

figuur 16

Monoalfabetische Substitutie met willekeurig alfabet

Neemt men in plaats van een verschoven normaal alfabet een alfabet waarin de letters in willekeurige volgorde staan, dan komen er ineens erg veel mogelijkheden bij. Kende de Caesarsubstitutie een sleutelruimte van 26, bij een willekeurige alfabetkeuze heeft deze ruimte de afmeting 26! . Uitputtend afzoeken is er nu niet meer bij (wel uitputtend in letterlijke zin). Het geheimschrift schijnt veel sterker te zijn geworden. Uit het vervolg zal echter blijken dat ook hier de schijn bedriegt.

Het substitutiealfabet kan willekeurig gekozen worden. In dat geval is uit het hoofd leren van de volgorde niet zo eenvoudig. Men baseert het substitutiealfabet daarom veelal op een sleuteltekst. Telkens als een letter voor de eerste maal in de sleuteltekst voorkomt, wordt deze aan het substitutiealfabet toegevoegd. Als de sleuteltekst is uitgeput, worden de nog ontbrekende letters in normale volgorde aangevuld. In figuur 17 is een voorbeeld van dit systeem gegeven.

sleutel:	DIT IS TE MOOI OM WAAR TE ZYN
substitutie:	D I T S E M O W A R Z Y N B C F G H J K L P Q U V X
normaal:	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
tekst:	EENCU RSUSV ANHET MATHE MATIS CHCEN TRUM
cryptogram:	EEBTL HJLJP DBWEK NDKWE NDKAJ TWTEB KHLN

figuur 17

Notatie voor deze methode:

$$b_t = X\ a_t$$

waarin X de transformatie door het substitutiealfabet voorstelt.

De cryptanalyse van dit geheimschrift trekt zich weinig aan van de 26! mogelijke sleutels. De kenmerken van de taal zorgen voor een flinke redundantie. Reeds opvallen wordt en het feit dat een klein aantal letters

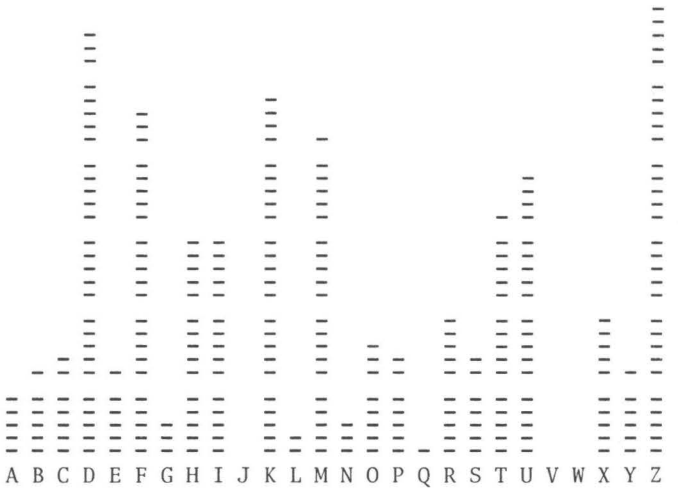
engels beslaan de acht letters E T N R O A I S ruim 65% van de tekst. Identificatie van deze acht letters betekent in de praktijk oplossing van het cryptogram.

Op eenvoudige wijze kan een ruwe schatting gemaakt worden van het effect van dit taalkenmerk op het af te zoeken deel van de sleutelruimte. Na tellen van de letterfrequenties kan veilig worden aangenomen dat de vier meest voorkomende letters uit de groep van acht stammen. De overige vier zijn onder de rest te zoeken. Het bestaan van de weinig voorkomende medeklinkers doet tenminste nog eens vijf letters van die overige groep afvallen. Voor de vier moet een greep gedaan worden uit ten hoogste zeventien letters. Alle verwisselingen onder de uitgekozen acht meetellend komt men tot plm 10^8 mogelijkheden; heel wat minder dan 26! (plm 4.10^{26}). Men bedenke verder dat de letter E zich meestal duidelijk genoeg manifesteert voor direkte identificatie. Contactinformatie (de frequentie van digrammen) ondersteunt de verdere selectie. Het zal duidelijk zijn, dat de monoalfabetische substitutie geen noemenswaardige bescherming biedt.

Ter illustratie zal opgave 1 uit hoofdstuk 9 van Computer Networks door A.S. Tanenbaum gemaakt worden. Gegeven is dat een monoalfabetische substitutie werd toegepast op een passage uit een gedicht van Lewis Carroll. In figuur 18 is de opgave op de gebruikelijke wijze in groepen van vijf letters weergegeven. In figuur 19 staat de frequentietelling; ook zonder het geven van het systeem zou de monoalfabetischesubstitutie direkt uit deze telling gebleken zijn.

KFDKT BDFZM EUBDK FDPZY IOMMZ TXKUK ZYGUR BZHAK
FTHCM URMFU DMZHX MFTNM ZHXMD ZYTHC PZQUR EZSSZ
CDMZH XGTHC MZHXP FAKFD MDZTM SUTYT HCFUK ZHXPF
DKFDI NTCMF ZLDPT HCMSO KPZTK ZSTKK FDUAM KDIME
ITDXS DRUID PDFZL DUOIE FZKRU IMUBD UROMZ IDUOK
URSID ZKFZH XZYYU ROMZI DRZKH UFOII AMZTX KFDEZ
INDHK DIKFD AKFZH GDXFT BBOEF RUIKF ZK

figuur 18



figuur 19

De meest voorkomende letters zijn D F H I K M T U Z. Ze komen 15-30 maal voor op een totaal van 272 letters. In deze groep zal het merendeel van de letters e t n r o a i s moeten worden gezocht, deze zijn immers de meest voorkomende letters in het engels.

Als eerste stap zal worden geprobeerd de klinkers van de medeklinkers te onderscheiden. Daarvoor kan gebruik gemaakt worden van de voorkeur die klinkers hebben voor contacten met medeklinkers. Tot 80 à 90 % van de klinkercontacten ter linker- en rechterzijde in het engels, hebben betrekking op een medeklinker. Het is daarom zaak een aantal geheide medeklinkers uit te selecteren en de contacten daarvan te bestuderen. De letters die zowel ter linker- als ter rechterzijde met deze in contact staan zijn waarschijnlijk klinkers, de letters met een voorkeur voor links of rechts zijn waarschijnlijk medeklinkers.

Een betrouwbare selectie van waarschijnlijke medeklinkers voor dit doel kan worden verkregen uit de groep letters die weinig voorkomen en daarbij liefst niet teveel variatie in contacten vertonen. In figuur 20 zijn hiervoor de letters Q J N X C P G B uitgezocht. In de figuur zijn

medeklinkers.

```
      Q L N Y G P C B
-----------------------
          C :
        DDD : DDDDD DDDD
            : FFF
            : G
      HHHHH : H
         II : I
          K :
            : MMMMM
            : O
            : P
          R :
      TTTTT : TTTTT
         UU : UU
        XXX :
          Y :
  ZZZ ZZZZZ : ZZZZZ
```

figuur 20

Een overzicht van de lettercontacten toont nog iets anders. De digram KF komt niet minder dan elfmaal voor, de digram FK geen enkele maal. Dit gedrag is karakteristiek voor de digram th. Ondersteuning wordt gevonden bij het feit dat F reeds bij de medeklinkers werd ingedeeld.

Uit de hoge frequentie van de geindiceerde klinkers Z (30) en D (28) volgt dat onder hen in ieder geval de e zal moeten worden gezocht. Dankzij de identificatie van K met h en de relatieve voorkomens van de digrammen ha, he, hi, ho kan als tweede kandidaat de letter a worden gekozen. Het paar i,o valt dan toe aan T,U.

Van de veel voorkomende letters resten nog H I M, waarvoor de n s r het meest in aanmerking komen. Hier helpt het feit dat in het engels de n zich in 88% van de gevallen door een klinker laat voorafgaan, de r in 69% en de s in 55%. Tellen van de contacten met onze klinkers D T U Z geeft: H 14 uit 15, I 9 uit 15, M 5 uit 21. Dus wordt de H geidentificeerd als n en het paar I,M met r,s, waarbij de voorkeur uitgaat naar I is r.

Men merke op dat in dit stadium reeds negen letters in vergaande mate geidentificeerd zijn. Alle conclusies werden getrokken op grond van algemene kenmerken. Nog geen beroep is gedaan op een interpretatie van de tekst zelf. In figuur 21 is een deel van de tekst weergegeven met daaronder de voorlopige toewijzingen. Hier en daar is een interpretatie aangegeven, die nieuwe letters oplevert. Invullen hiervan leidt tot nieuwe vondsten. Een sneeuwbaleffect ontstaat dat snel tot volledige oplossing van het cryptogram leidt.

```
      D     F     H     I     K     M     T     U     Z
     a,e    h     n    r,s    t    s,r   i,o   o,i   e,a

K F D K T B D F Z M E U B D K F D P Z Y I O M M Z T X K U K
t h a e t i o . a e h e s a r . o i . a e t h a e . e a . r s . s r s e r a i o . t o i t
t h e/t i m e/h a s/       t h e             s a i d/t o

Z Y G U R B Z H A K F T H C M U R M F U D M Z H X M F T N M
e . . o i . . e a . n . t h o i n . s r i o . s r h o i a s e e r a n . s r h i o . s r
                t h i n g s     s h o e s/a n d/s h i p s
```

figuur 21

De oorspronkelijke opgave liet de verdeling in woorden zien (figuur 22). Hiermee is het oplossen van het cryptogram echter triviaal. De combinatie KFD adverteert zichzelf als the. Voor ZHX valt and te proberen, vooral gezien de hoge frequentie van Z en het voorkomen van het éénletter woord Z. De vele kleine woorden van twee en drie letters bieden voldoende probeermateriaal om snel verder te komen.

KFD KTBD FZM EUBD KFD PZYIOM MZTX KU KZYG UR BZHA KFTHCM
UR MFUDM ZHX MFTNM ZHX MDZYTHC PZQ UR EZSSZCDM ZHX GTHCM
ZHX PFA KFD MDZ TM SUTYTHC FUK ZHX PFDKFDI NTCM FZLD PTHCM
SOK PZTK Z STK KFD UAMKDIM EITDX SDRUID PD FZLD UOI EFZK
RUI MUBD UR OM ZID UOK UR SIDZKF ZHX ZYY UR OM ZID RZK
HU FOIIA MZTX KFD EZINDHKDI KFDA KFZHGDX FTB BOEF RUI KFZK

In het voorafgaande is de kwetsbaarheid van de monoalfabetische substitutie gebleken. De kenmerken van de taal schijnen er te duidelijk doorheen. Wanneer echter de tekst met meer dan één substitutiealfabet wordt vercijferd, raken de taalkenmerken verspreid over meer dan één alfabet. De waarde van de taalkenmerken voor de cryptanalyse vermindert daardoor en het geheimschrift wordt sterker.

De meest bekende polyalfabetische substitutie is de Vigenère, stammend uit de 16e eeuw. In dit systeem wordt op de achtereenvolgende letters van de te vercijferen tekst steeds een andere Caesar substitutie toegepast. Dus bijvoorbeeld een verschuiving over 25 posities voor de eerste letter, over 14 voor de tweede, 19 voor de derde, en dan weer van voren afaan. De opvolgende verschuivingen vormen de sleutel tot de substitutie. Is de lengte van de sleutel kort in verhouding tot de lengte van het bericht, dan spreekt men van periodieke polyalfabetische substitutie. De Vigenère behoort tot dit type.

In formule uitgedrukt bestaat de Vigenère uit de volgende transformatie:

$$b_t = C^{p(t)} a_t \quad \text{met} \quad p(t) = (t-1) \bmod k$$

Hierin is p(t) een periodieke functie bepaald door de sleutel $s_0\ s_1 \ldots s_{k-1}$ met lengte k. De Caesar substitutie $C^{p(t)}$ toe te passen op de te letter, wordt gegeven door het element van de sleutel s met een nummer gelijk aan de rest bij deling van (t-1) doorde lengte van de sleutel k.

Het onthouden van de sleutel wordt vergemakkelijkt wanneer deze niet in de vorm van een rij getallen, maar als een sleutelwoord of sleutelzin gespecificeerd is. Hierin heeft de sleutelletter A de betekenis: neem Caesar substitutie C^0, B betekent neem C^1, enz. In het spraakgebruik bezigt men de uitdrukking: vercijferen met het X-alfabet. Gemakkelijk valt in te zien, dat hoe langer de sleutel, hoe beter de bescherming. Het onthouden van een zin, kost niet veel meer moeite dan het onthouden van een woord.

Diverse hulpmiddelen zijn bedacht om het vercijferen te vergemakkelijken (vermindert de kans op fouten!). Een veelgebruikt hulpmiddel is het Vigenère-tableau van figuur 23. De substitutie C^D wordt bewerkstelligd door de te vercijferen letter op te zoeken in de bovenste rij en de daardoor aangegeven kolom te snijden met de rij die met D begint (het D-alfabet). De letter op het snijpunt van rij en kolom geeft de vercijferde letter. Bij het decoderen van een bericht zoekt men in de rij aangegeven door de sleutel naar de te ontcijferen letter. De letter aan de top van de desbetreffende kolom geeft de bijbehorende letter van de klare tekst. In figuur 24 is de gebruikelijke voorbeeldzin met het Vigenère systeem vercijferd.

Vigenere Tableau

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

figuur 23

tekst:	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
sleutel:	STOEI	POESS	TOEIP	OESST	OEIPO	ESSTO	EIPO
cryptogram:	WXBGC	GGYKN	TBLMI	AELZX	AEBXG	GZUXB	XZJA
coincidenties:	--	--		--	--	---	--
	1	3		2	2	3	1

figuur 24

Lange tijd, ongeveer tot de jaren 1860, heeft men gemeend dat de Vigenère veilig was. Toen echter ontwikkelde de duitse cryptanalist Kasiski een methode die het belangrijkste geheim, de lengte van de sleutel, aan het cryptogram ontfutselt. Zodra de sleutellengte k bekend is, kan het cryptogram worden opgesplitst in k afzonderlijke monoalfabetische substituties. Daar alle substituties plaatsvinden met hetzelfde, slechts onderling verschoven, alfabet kunnen indentificaties in de ene set gecorreleerd met die in andere sets worden.

Voor het principe achter de Kasiski methode richte men de aandacht in figuur 24 op de regel 'coincidenties'. De balkjes 1 markeren twee plaatsen waar eenzelfde stuk klare tekst met hetzelfde deel van de sleutel bewerkt is. De afstand van deze coincidentie is dan vanzelfsprekend een geheel aantal malen de sleutellengte. Wanneer het bericht lang is ten opzichte van de sleutel, neemt de kans op dit gebeuren toe. Men haalt de gemeenschappelijke factor uit de afstanden van de waargenomen coincidenties en wint aldus de sleutellengte.

In de praktijk zijn er echter een paar storende factoren aanwezig. In figuur 24 duiden de blokjes 2 een coincidentie aan, die ontstaan is door het samenvallen van een herhaling in de tekst met een herhaling in de sleutel. De afstand hier is geen veelvoud van de sleutellengte. Coincidenties kunnen ook louter toevallig zijn. Dan is uiteraard geen enkel verband met de sleutellengte. Balkjes 3 in figuur 24 geven zo'n toevallige coincidentie aan. Alle hier getoonde coincidenties zijn slechts twee letters lang. Naarmate de coincidentie zich over meer letters uitstrekt, wordt de geindiceerde periode betrouwbaarder. Immers de kans op toevallige herhaling van veel letters is gering.

Een instruktief voorbeeld van de cryptanalyse van een Vigenère wordt ons aangereikt door Edgar Allen Poe (zie ook Cryptologia, vol 1 (1977) 93-96, 318-325). In 1839 had Poe een kolom in een te Philadelphia uitgegeven blad, Alexander's Weekly Messenger. Daarin had hij zijn lezers uitgedaagd hem boodschappen in geheimschrift te zenden. Een zekere G.W. Kulp uit Lewiston, Pennsylvania deed dit. Poe slaagde er niet in de boodschap te ontcijferen en publiceerde dit negatieve resultaat in de veronderstelling dat er sprake was van bedrog. Het blijkt echter een eenvoudige Vigenère, gecompliceerd door een vijftiental fouten, waarschijnlijk voor een belangrijk deel te wijten aan een moeilijk leesbaar handschrift. Figuur 25 geeft de boodschap van Mr Kulp.

Ge Jeasgdxv,
Zij gl mw, laam, xzy zmlwhfzek ejlvdxw
kwkw tx lbr atgh lbmx aanu bai Vsmukks pwn
vlwk agh gnumk wdlnzweg jnbxvv oaeg enwb
zwmgy mo mlw wnbx mw al pnfdcfpkh wzkex
hssf xkiyahul. Mk num yexdm wbxy sbc hv
wyx Phwkgnamcuk?

figuur 25

```
 2=  ----- ----- ----- ----- ----- ----- ----
 4=  ----- ----- ----- ----- -
     ----- ----- ----- -----
     ----- --
 6=  ----- ----- ----- --
     ----- --
 8=  ----- ----- ---
     ----- -----
10=  ----- -
     -
12=  ----- ----- --
     ----- --
14=  -
     --
16=  --
18=  ----- ----
20=  --
     -
22=  -
24=  ----- ----
     --
```

figuur 26

letters. Er zijn vier van zulke coincidenties (MLW, NBX, NUM, WKW). Verder zijn er een dertigtal digrammen die een of meermalen herhaald in de tekst voorkomen. De afstanden van al deze coincidenties worden in factoren ontbonden. In figuur 26 zijn deze factoren tot en met 25 geturfd. Opvallend is hierin het sterk naarvoren komen van de zesvouden 6, 12, 18 en 24. Een periode van 6 wordt hierdoor sterk gesuggereerd.

Geholpen door de woordindeling van het cryptogram, kan nu geprobeerd worden de sleutel terug te winnen. Veronderstel eens dat alle woorden van drie letters het engelse lidwoord 'the' voorstellen. De eerste kandidaat is de groep ZIJ beginnend op de 11e plaats in het cryptogram. De t leidt naar de Z in het G-alfabet, de h naar de I in het B-alfabet en de e naar de J in het F-alfabet. De bijbehorende sleutelletters zijn dan GBF. Hun plaats in het sleutelwoord kan worden gehaald uit de positie van het gekozen woord in het cryptogram in combinatie met de vermoedelijke lengte van de sleutel. In figuur 27 is dit gedaan voor alle woorden van drie letters uit het cryptogram. Het zo gevonden sleutelwoord NITESU ziet er niet aantrekkelijk uit. Anderzijds zijn de aanwijzingen voor dit sleutelwoord nogal sterk: 5 van de 10 woorden zijn er mee consistent! Daarom is in figuur 27 met behulp van dit sleutelwoord een deel van de tekst uitgewerkt.

Hoewel het resultaat op het eerste gezicht teleurstellend lijkt, zijn er toch brokjes goede tekst in te vinden. Woorden als 'how', 'arrives', 'at the', 'with the' en 'other' duiden erop dat de oplossing dichtbij is. Nadere overweging brengt het idee dat de afwisselende stukjes tekst en onzin hun oorzaak zouden kunnen hebben in een verdubbeling van de sleutellengte. De sleutel wordt dan twaalf letters lang, niet direct in tegenspraak met het beeld van figuur 26.

Op dit punt aangeland moet, geleid door de herkenbare tekst, een gooi gedaan worden naar een woord dat overlapt met het ontbrekende deel van de sleutel. Met in het achterhoofd de naam Alexander's Weekly Messenger is het niet moeilijk om 'the messplmwr' te lezen als

'the messenger'. Zoeken van het sleutelwoord geeft de sleutel UNITEDSTATES. Dan wordt ook duidelijk waar de sterke voorkeur voor een periode van zes in figuur 26 zijn oorsprong vindt. In het sleutelwoord bevindt zich tweemaal de groep TE, precies op afstand zes.

positie in de sleutel		s_0	s_1	s_2	s_3	s_4	s_5
trigram met positie							
ZIJ	[11]	F				G	B
XZY	[22]				E	S	U
LBR	[47]	N				S	U
BAI	[62]		I	T	E		
PWN	[73]	W	P	Y			
AGH	[80]		H	Z	D		
MLW	[117]			T	E	S	
NUM	[156]	N	I				U
SBC	[168]	U	Y				Z
WYX	[173]	T				D	R
veronderstelde sleutel=		N	I	T	E	S	U

NI TESUNITE SUN IT ES UNIT ESU NITESUNIT
GE JEASGDXV ZIJ GL MW LAAM XZY ZMLWHFZEK
tw qaiytver how ys ie rnst the messplmwr

ESUNITE SUNI TE SUN ITES UNIT ESUN ITE
EJLVDXW KWKW TX LBR ATGH LBMX AANU BAI
arrives scxo at the sacp roee with the

SUNITESU NIT ESUN ITE SUNIT ESUNITES
VSMUKKSS PWN VLWK AGH GNUMK WDLNZWEG
dyzmrgay cou rtcx snd other slrardao

figuur 27

Er zijn vele variaties op het Vigenère thema. Het systeem met een cijfersleutel en slechts de eerste tien alfabetten heet Gronsfeld. Ook het anders hanteren van de Vigenère tabel leidt tot variaties. Wanneer in de rij die begint met de tekstletter de sleutelletter wordt opgezocht en vandaaruit aan de top van de kolom de vercijferde letter, ontstaat de Beaufort. Bij het ontcijferen in dit systeem kan

gecodeerde bericht. Het aardige van de Beaufort is, dat deze equivalent is met de gewone Vigenère mits daarin de alfabetten in omgekeerde volgorde worden uitgeschreven. De Variant-Beaufort ontstaat als men de Vigenère procedure omdraait. Dwz coderen in Vigenère is decoderen in de Variant, decoderen in Vigenère is coderen in de Variant. In figuur 28 is de relatie tussen deze methode weergegeven. Tenslotte kan men al deze methoden toepassen met een willekeurig alfabet in plaats van het normale alfabet.

	Vigenere	Beaufort		Variant
	ABC ... Z	ABC ... Z =	ABC ... Z	ABC ... Z
	ABC ... Z	ABC ... Z	AZY ... B	ABC ... Z
	BCD ... A	BCD ... A	BAZ ... C	BCD ... A
	. .	. .	. .	. .
	ZAB ... Y	ZAB ... Y	ZYX ... A	ZAB ... Y
coderen	K	C	K	C
	↓	↑	↓	↑
	S → C	K → S	S → C	S → K
decoderen	K	K	K	C
	↑	↑	↑	↓
	S → C	C → S	S → C	S → K

S = sleutel letter K = klare tekst letter C = code letter

figuur 28

Multiplex systemen

Voor verdere uitbouw van de polyalfabetische substitutie staan wegen open. Men kan de sleutel langer maken, hetgeen leidt tot de in het vervolg te bespreken rotorsystemen. Men kan echter ook werken met een aantal van elkaar verschillende substitutie alfabetten. Daarmee ontstaan de multiplex of strip systemen. Deze vinden hun oorsprong bij Thomas Jefferson (eind 18e eeuw) en bij Etienne Bazeries (eind 19e eeuw). In de dertiger jaren kwam zo'n systeem in gebruik bij het U.S. State Department. Het U.S. Army Signal Corps maakte vanaf 1922 geruime tijd gebruik van de M-94, een aan het veldwerk aangepaste versie van het stripsysteem.

Het systeem kan in zijn algemeenheid als volgt beschreven worden. Gegeven is een groot aantal (bijvoorbeeld 50) permutaties van het alfabet. Deze permutaties worden uitgeschreven op afzonderlijke strips, zodat ze onderling verwisseld en ten opzichte van elkaar verschoven kunnen worden. Uit deze voorraad strips neemt men volgens een of andere sleutel een geschikte selectie (als regel 15 à 30 stuks). De genomen strips worden in de door de sleutel aangegeven volgorde gelegd. De strips worden onderling nu zo geschoven dat de te coderen tekst op een verticale lijn te lezen valt. De gecodeerde tekst kan nu worden afgelezen van een willekeurig gekozen lijn uit de 25 andere verticalen. In figuur 29 is de gang van zaken afgebeeld. Coderen en decoderen geschiedt in groepen, elke groep is zolang als het aantal geselecteerde strips. Decoderen komt neer op leggen van de strips en in de verticale lijn zoeken naar die van de 26 kolommen die begrijpelijke tekst levert.

figuur 29

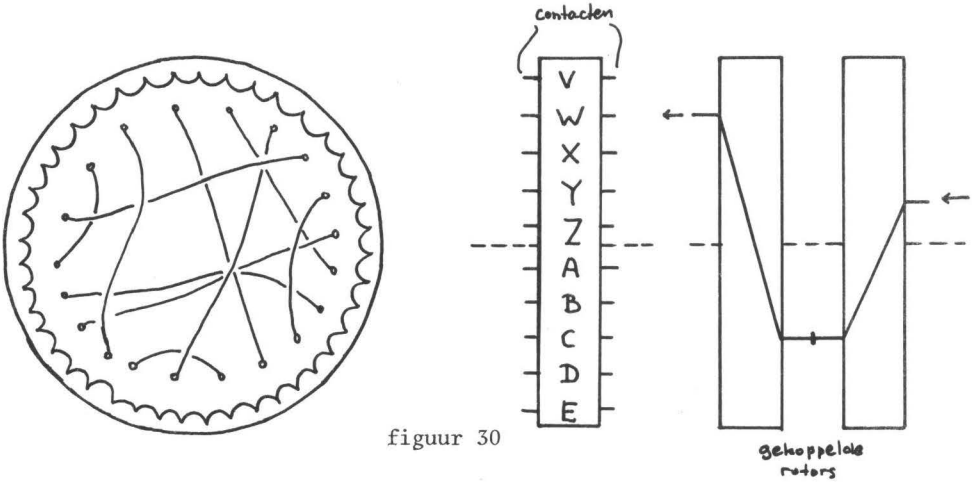
Cryptanalyse van een systeem als dit is niet langer een triviale ondernemening. Bij de tot op dit punt behandelde systemen kon een beperkt bericht in kort bestek worden geanalyseerd en tot oplossing gebracht. Dergelijke eenvoudige illustraties zijn er niet bij de strips. Analyse van een stripsysteem uitsluitend op grond van codetekst behelst de reconstructie van de strips; uit aanzienlijke hoeveelheden materiaal, met dezelfde sleutel vercijferd, moeten de stukken worden geïsoleerd die met dezelfde vertieaal zijn gecodeerd. Voor elk van de

verticalen voert tenslotte tot complete reconstructie van de strips. Een uitvoerig verslag van de analyse van een stripsysteem vindt men bij [Rohrbach].

Rotorsystemen

De rotorsystemen ontlenen hun kracht aan een zeer lange sleutel voor een polyalfabetische substitutie. De rotormachines werden ontwikkeld in de periode aan het eind van de eerste wereldoorlog. In de tweede wereldoorlog zijn ze op uitgebreide schaal gebruikt.

Een rotor bestaat uit een ring die op een as kan worden geschoven (figuur 30). Op de rand van de ring staat een alfabet dat dient om de stand van de rotor te kunnen instellen. Een tandwiel zorgt voor de draaiing van de rotor en de koppeling met de andere rotors in het systeem. Aan beide zijkanten bevindt zich een rij contacten, die willekeurig met elkaar verbonden zijn, steeds van de ene zijkant naar de andere, twee aan twee. Een electrische stroom die bij een bepaald contact binnenkomt, verschijnt aan de andere zijde op een geheel andere plaats. Op deze manier bewerkstelligt de rotor een monoalfabetische substitutie.



figuur 30

Door verscheidene rotors met onderling verschillende verbindingen aan elkaar te koppelen, kan achtereenvolgens een reeks substituties worden uitgevoerd. De lange sleutel van de rotormachines ontstaat nu

doordat na elke vercijferde letter de onderlinge stand van de rotors door draaiing wordt gewijzigd. De eerste rotor kan bijvoorbeeld bij elke complete omwenteling de tweede een stap meenemen, enz, analoog aan de telwielen van een gasmeter. Vanzelfsprekend kunnen ook minder voor de hand liggende draaipatronen geimplementeerd worden. De sleutel van een rotorsubstitutie bestaat uit de volgende elementen: de keuze en onderlinge positie van de rotors, de positie van de rotorcontacten ten opzichte van de koppeling tussen de beweging van de rotors en de startpositie van de rotors aan het begin van de codering.

Algebraisch kan het effect van een rotor beschreven worden met de formule

$$b_t = C^t \, R \, C^{-t} \, a_t$$

Hierin is R de door de rotor bewerkstelligde substitutie (bijvoorbeeld R = ABC... → YKA...), C de Caesar substitutie. De formule kan als volgt geinterpreteerd worden. Bij het vercijferen van de t^e-letter brengt Caesar C^{-t} de rotor eerst terug in de beginstand, daarin kan R worden toegepast en vervolgens brengt C^t de rotor weer in de huidige stand. Bij combinatie van rotors en ingewikkelder bewegingen wordt de t in de Caesar substituties vervangen door een of andere functie van t.

Hebern Cryptograaf

De oudste machine gebaseerd op het rotor-principe is de Hebern cryptograaf. De rotors van deze machine kunnen in willekeurige volgorde worden samengevoegd. Van deze rotors zijn er een aantal beweeglijk en een aantal vast. Bijvoorbeeld de eerste rotor draait een stap bij elke letter, de vijde rotor een stap bij elke omwenteling van de eerste en de derde rotor draait eens per omwenteling van de vijfde. De weg door de rotors wordt voorafgegaan door een monoalfabetische substitutie afkomstig van het toetsenbord. De QWERTY... van het normale schrijfmachine toetsenbord wordt afgebeeld op de ABCDE... van de eerste rotor.

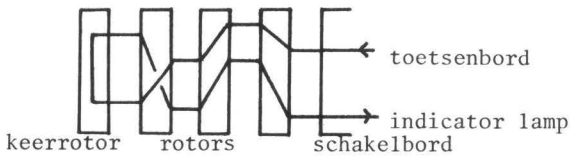
Een zwakte van deze machines is gelegen in de betrekkelijke regel-

de eerste rotor. De rest van de machine bewerkstelligt daarom gedurende die 26 vercijferingen slechts een constante monoalfabetische substitutie. De machine kan dan worden beschreven als $b_t = X\ C^t\ R_1\ C^{-t}\ a_t$ waarin X het effect van de stilstaande rotors. Evenzo is het effect van de derde en volgende rotors constant wanneer niet meer dan 676 letters vercijferd worden. Deze zwakte kan worden aangegrepen bij het bepalen van de rotor-substituties.

Een hulpmiddel bij de cryptanalyse van rotor-geheimschriften zijn de zogenaamde isomorfen. Een isomorf is een stuk tekst dat door een monoalfabetische substitutie uit een ander stuk tekst kan ontstaan. Bijvoorbeeld de groep AXQJPAJ is isomorf met ZOQMJZM. Het duidelijkst constateert men de isomorfie aan het voorkomen van herhaalde letters. Echter FITGH is isomorf met ZSKUH zonder dat van herhaalde letters sprake is. Lange isomorfen kunnen ontstaan door te vercijferen met dezelfde stukken tekst bij dezelfde startpositie van de eerste rotor. De monoalfabetische substitutie van de som der overige rotors geeft dan juist de relatie tussen de beide vercijferde tekstdelen. Men kan ook, indien de eerste rotor bekend is, waarschijnlijk voorkomende woorden omzetten met behulp van die rotor en vervolgens naar isomorfen van het resultaat gaan zoeken. Langs deze weg penetreert men gestaag de geheimen van de machine en analyseert rotor voor rotor.

Enigma

Waarschijnlijk de bekendste rotormachine is de Enigma. Oorspronkelijk bedacht door Arthur Scherbius (plm 1923) werd de machine verder ontwikkeld tot de standaard codemachine van de duitse legers in de tweede wereldoorlog. In de Wehrmacht uitvoering omstreeks 1940 zag de machine er als volgt uit (figuur 31).



figuur 31

Aanslaan van een letter op het toetsenbord doet een stroom lopen. Eerst gaat deze door een schakelbord. Op dit schakelbord bevindt zich voor elke letter een contact. Zes tot tien van deze contacten worden door stekers met elkaar verbonden; ongeveer op de wijze van een ouderwetse telefooncentrale. Het schakelbord zorgt voor een eerste, gemakkelijk te veranderen, monoalfabetische substitutie. Daarna vloeit de stroom door de drie rotors, gekozen uit een set van vijf. Elke rotor heeft een instelmogelijkheid voor de onderlinge positie van de contacten en het voortbewegingsmechanisme. De snelste rotor draait een stap bij elke letter, de volgende een stap bij elke omwenteling van de eerste - op één plaats echter 2 stappen tegelijk - en de derde rotor draait een stap per omwenteling van de tweede. Vervolgens gaat de stroom door de keerrotor, waarin alle contacten twee aan twee zijn verbonden. De stroom gaat nu in omgekeerde richting opnieuw door de drie rotors en het schakelbord. De oplichtende lamp geeft de codeletter aan. Het proces is volledig omkeerbaar, zodat ontcijferen op dezelfde manier gaat. Men behandelt de gecodeerde tekst als was het te coderen materiaal.

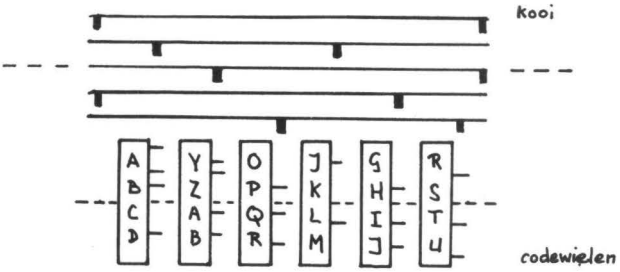
In formule luidt de enigma:

$$b_t = S^{-1}\ V^{-1}\ X^{-1}\ Y^{-1}\ Z^{-1}\ K\ Z\ Y\ X\ V\ S\ a_t$$

In de formule zijn X, Y en Z het effect van de draaiende rotors, K is de keerrotor, S het schakelbord en V de verbinding tussen de rotors en het schakelbord. X kan uiteraard nader worden gepreciseerd dmv de uitdrukking $X = C^{-f(t)}\ X'\ C^{f(t)}$.

Alhoewel de duitsers de enigma behoorlijk veilig achtten, is vele jaren na de oorlog gebleken dat de ermee vercijferde berichten veelvuldig werden gedecodeerd door de engelsen. De eerste, belangrijke stappen daartoe werden gezet in de periode voorafgaande aan de tweede wereldoorlog door drie Polen, Rejewski, Rózycki en Zygalski. Men kan het verhaal van deze eerste grote doorbraak lezen in [Rejewski].

Een verwante, maar op een wat andere manier geconstrueerde, machine is de Hagelin Cryptograaf. Deze machine werd in de dertiger jaren uitgevonden door de Zweed Boris Hagelin. Onder de naam M-209 heeft dit apparaat tijdens de tweede wereldoorlog dienst gedaan in het amerikaanse leger.



figuur 32

De Hagelin in de M-209 versie (figuur 32) bestaat uit een kooi met 27 staven. Op elk van deze staven zitten twee verplaatsbare 'tanden'. Voor elk paar op een staaf zijn acht posities beschikbaar. Twee van deze, die uiterst links en uiterst rechts, maken de tand inaktief. Met ieder van de zes overige posities correspondeert een codewiel. Op deze codewielen zitten respectievelijk 26, 25, 23, 21, 19 en 17 pinnen. Deze pinnen hebben twee posities, een aktieve en een inaktieve. Pinnen en tanden vertonen een onderlinge interaktie. Bij het vercijferen van een letter draait de kooi rond. Stel op de contactplaats tussen kooi en codewiel is op het codewiel een aktieve pin aanwezig. Nu correspondeert bij een rondgang van de kooi het aantal tanden op deze plaats met een bijbehorende substitutie. Zijn er meer codewielen aktief (pin op contactplaats), dan telt het aantal daardoor aangeslagen tanden op. Echter twee tanden op een en dezelfde staaf (overlap) tellen

maar voor één. Na de vercijferoperatie draaien de codewielen elk één stap, zodat een nieuwe combinatie van pinnen op de contactplaats komt. Het totaal aantal geactiveerde tanden bij de rondgang van de kooi, bepaalt welk alfabet uit de Beaufort tabel wordt gebruikt voor de vercijfering. In figuur 33 is de Beaufort tabel gegeven met aan de linkerrand het met het aantal actieve tanden (exclusief overlap) overeenkomende getal.

Beaufort tabel voor Hagelin C-48 (M-209)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0,26	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A
1,27	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B
2	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C
3	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D
4	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E
5	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F
6	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G
7	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H
8	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I
9	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J
10	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K
11	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L
12	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M
13	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O	N
14	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P	O
15	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q	P
16	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R	Q
17	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S	R
18	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T	S
19	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U	T
20	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V	U
21	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W	V
22	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X	W
23	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y	X
24	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z	Y
25	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A	Z

figuur 33

In figuur 34 is een zin gecodeerd met een voorbeeldinstelling, waarbij slechts drie codewielen actief zijn. De actieve codewielen zijn die met 17, 19 en 21 pinnen en respectievelijk 4, 7 en 5 tanden

inactieve pin op het codewiel dan. Het voorbeeld laat eveneens zien hoe bij de Hagelin de letter Z gebruikt wordt voor het aanduiden van een spatie. Bij het decoderen onderdrukt de machine het afdrukken van de letter Z, zodat gedecodeerde tekst met woordverdelingen ontstaat. Bij het coderen verzorgt de Hagelin automatisch het indelen in groepen van vijf letters.

	EENZC	URSUS	ZVANZ	HETZM	ATHEM	ATISC	HZCEN	TRUMZ
#1	00044	04400	04044	04000	44044	00040	44040	00440
#2	77007	07000	07770	00077	70070	70000	77700	00777
#3	05000	05000	05000	05000	50500	00500	00500	00500
totaal	1 1	1	1 1		1 1		111	11
	72040	05400	05704	09077	54504	70540	00240	00507
cryptogram	CHMEH	FXLFH	ATGWE	SEGHU	OKXFR	GGWLX	CKJZM	GIUXH

figuur 34

Een uitvoerige verhandeling over de cryptanalyse van de Hagelin cryptograaf vindt men bij [Barker].

Willekeurige rijen

De zeer lange sleutel, zoals deze bij de rotormachines door draaiende codewielen wordt gevormd, kan ook op andere wijze gemaakt worden. Men kan een formule voor het genereren van een pseudo-random rij hanteren. Een eenvoudig voorbeeld is

$$X_{i+1} = a\ X_i + b \mod p$$

Elk element van de rij wordt afgeleid uit het vorige; voor p neemt men een priemgetal, a en b zijn constanten. De elementen X_i van de rij gebruikt men voor de cryptografische transformatie. Van groot belang is de kwaliteit van de rij X_i. De periode van deze rij moet groot zijn en er mogen geen regelmatigheden in voorkomen.

Een werkelijke randomrij kan worden verkregen door het waarnemen van willekeurige processen als het uiteenvallen van een radioaktief element. Gebruikt men een dergelijke willekeurige rij slechts éénmaal

dan bereikt men daarbij de hoogst mogelijke beveiliging. In dit geval spreekt men van de methode van de 'one time pad'.

Playfair

Een andere richting wordt ingeslagen, indien men tegelijkertijd twee of meer tekens vervangt door andere, al dan niet gelijk in aantal. De eenvoudigste vervanging is die van telkens een tweetal letters door een ander tweetal. In principe is het mogelijk een vervangingstabel op te stellen voor alle 676 digrammen en daarmee te werken op precies dezelfde wijze als bij een monoalfabetische substitutie. In de praktijk echter heeft men meer aan methoden waarvan de details gemakkelijker zijn te onthouden.

De populairste digramsubstitutie is ongetwijfeld de Playfair. De methode werkt met een sleutelvierkant waaruit de voor elk digram toe te passen substitutie wordt afgeleid. Het sleutelvierkant heeft de afmetingen 5 x 5 en herbergt 25 letters van het alfabet; de I en J worden voor het geamk samengenomen. Het beste is het, het sleutelvierkant volkomen willekeurig met deze 25 letters te vullen. Vaak echter wordt een sleutelwoord gebruikt voor de constructie van het vierkant. De verschillende letters van het sleutelwoord komen achtereenvolgens in de eerste vakjes van het vierkant, de rest van het alfabet wordt in de standaardvolgorde ingevuld. In figuur 35 is een sleutelvierkant geconstrueerd uitgaande van de sleutel ZONDERLING.

Voor het codeerproces van telkens twee letters zijn nu de volgende gevallen te onderscheiden:

(1) De twee letters staan op de tegenovergestelde hoekpunten van een vierkant. Het substitutie equivalent wordt gevormd door de beide andere hoekpunten, de rijen nemend in dezelfde volgorde als in het te coderen digram. Voorbeeld: UE wordt YZ en EU wordt ZY in de figuur 35.

(2) De letters van het digram staan op dezelfde rij. Nu worden de twee

uit (2) wordt nu op de kolom toegepast. Digram UZ wordt dus ZR, men neemt steeds de letter onder de te coderen letter.

(4) De letters van het digram zijn aan elkaar gelijk. Het digram wordt met behulp van een vrij te kiezen hulpletter opgesplitst.

Z	O	N	D	E
R	L	I	G	A
B	C	F	H	K
M	P	Q	S	T
U	V	W	X	Y

tekst:	EEN CURSUS VAN HET MATHEMATISCH CENTRUM
opgesplitst:	EX EN CU RS US VA NH ET MA TH EM AT IS CH CE NT RU MX
omgezet:	DY ZD BV GM XM YL DF AY TR SK ZT KY GQ FK KO EQ BZ SU
cryptogram:	DYZDB VGMXM YLDFA YTRSK ZTKYG QFKKO EQBZS U

figuur 35

De Playfair bezit een aantal zwakke plekken. Een direkte aanval is mogelijk als voldoende tekst ter beschikking staat voor een analyse van de digram frequenties. Omkering van digrammen als TH en HT levert ook gecodeerd een omkering. Wanneer de betrokken digrammen een sterk verschillend gedrag vertonen (zoals th en ht in het engels), vertonen ze dit gedrag ook in het cryptogram en verraden zich daardoor. Alhoewel de Playfair een digramsubstitutie is, worden de karakteristieken van elke letter toch over slechts vijf letters (de vier in dezelfde rij en de letter in de kolom eronder) uitgesmeerd.

Cryptanalyse van een Playfair geschiedt meestal door reconstructie van het sleutelvierkant. In dit verband kan worden opgemerkt dat het construeren van het vierkant met behulp van een sleutel een behoorlijke zwakte betekent. Gebruik van een sleutelwoord zorgt er doorgaans voor dat van de letters VWXYZ het merendeel probleemloos in de laatste hokjes van het vierkant geplaatst kunnen worden. De alfabetische opvulling van de restletters biedt eveneens aanknopingspunten waar een deel van het vierkant is gereconstrueerd. Blijkt bijvoorbeeld

een rij eruit te zien als M P Q ? T, dan komen slechts R en S in aanmerking voor de open plaats.

Liefhebbers van het genre kunnen hun krachten beproeven op de brief uit de detective roman 'Have His Carcase' van Dorothy Sayers. Het valt wel aan te raden langs algemene weg te werken en de goed gerichte gok van Lord Peter Wimsey niet te gebruiken. Dat ook een Playfair van slechts 30 letters met succes ontrafeld kan worden leze men bij [Mongé].

Code

Het vervangen van complete lettergrepen, woorden en zinnen door groepen letters of cijfers voert tot een code. Een codeboek bevat een lijst met deze woorden en uitdrukkingen vergezeld door het code-equivalent. Een tweedelige code bezit een tweede lijst, geordend naar de coderingen, zodat daarin de betekenis van de codegroepen kan worden teruggezocht. Bij een ééndelige code volstaat men met een lijst voor beide operaties. Het nadeel van de eendelige code is dat ten behoeve van het terugzoeken ook de codequivalenten een ordening moeten vertonen. Dit maakt het bij de analyse van de code gemakkelijk om nieuwe groepen te plaatsen in een reeds gedeeltelijk ontrafelde code. Bij een tweedelige code is deze ordening niet nodig.

Ondanks de schijnbaar hoge bescherming die het gebruik van een code biedt, verkijke men zich niet op de mogelijkheden tot cryptanalyse. Alleen al de vaak zeer grote hoeveelheden materiaal die in code overgebracht moeten worden, vormen een zwakte. Vele berichten hebben een stereotyp begin of einde; herkenning daarvan zorgt voor een eerste inbraak in de code. Naarmate meer groepen geidentificeerd worden, neemt de kans op herkennen van verdere groepen toe. De code raakt vrij snel gecompromitteerd. Om deze reden wordt een code doorgaans niet zonder meer gebruikt. Een ander geheimschrift wordt er als een extra beveiligingslaag omheen gelegd.

Literatuur

[Barker]
W.G.Barker. Cryptanalysis of the Hagelin Cryptograph.
Aegean Park Press 1977.
[Friedman]
W.F.Friedman. Elements of Cryptanalysis. Aegean ParkPress.
[Gaines]
H.F.Gaines. Cryptanalysis. Dover Publications 1957.
[Kahn]
D.Kahn. The Codebreakers. MacMillan Publ.Co. 1967.
[Kullback]
S.Kullback. Statistical Methods in Cryptanalysis. Aegean Park Press.
[Monge]
A Tribute to Alf Monge. Cryptologia vol 2 (1978) 178-185. Herdruk
uit Signal Corps Bulletin No 93, NovDec 1936.
[Rejewski]
M.Rejewski. How Polish Mathematicians Deciphered the Enigma.
Annals of the History of Computing. vol 3 (1981) 213-234.
[Rohrbach]
H.Rohrbach. Report on the Decipherment of the American Strip Cipher 0-2
by the German Foreign Office. Cryptologia vol 3 (1979) 16-26.
[Sacco]
L.Sacco. Manual of Cryptography. Aegean Park Press.

CRYPTOGRAFIE EN COMPLEXITEIT

door

PETER van EMDE BOAS, IIW/FVI, UVA.

1. INLEIDING

Deze voordracht gaat over de begrippen *makkelijk* en *moeilijk*. Wellicht zult U zich afvragen of deze, inherent subjectieve, begrippen zich lenen voor een wiskundige bestudering. U zult het onderwerp dan ook niet snel aantreffen tussen de klassieke onderwerpen als algebra en meetkunde die wellicht de hoekstenen vormen van het U bekende deel der wiskunde. Dat de begrippen relevant zijn voor het onderwerp van deze cursus, te weten de cryptografie, behoeft amper toelichting. Het dient bij een goed cryptosysteem tenslotte zo te zijn dat de legitime gebruikers ervan in staat moeten zijn hun berichten volgens het systeem te coderen en te decoderen, en dit proces dient bij voorkeur *makkelijk* te zijn. Tegelijkertijd dient het de vijand *moeilijk* zo niet *onmogelijk* te zijn om een ontvangen boodschap te ontcijferen.

Verplaats U in de positie van het bevoegde gezag waaraan een nieuw cryptografisch systeem wordt aangeboden. U zult uzelf ervan willen overtuigen of, en zoja, in welke maten dit systeem voldoet aan de eisen hierboven gesteld. Het blijkt dan alras dat U met het huis,tuin en keuken begrip van *makkelijk* en *moeilijk* niet uitkomt. Veelal zat de claim dat een voorgesteld systeem *moeilijk* te decoderen is berusten op een argumentatie als: *hier zijn een heleboel ingewikkelde dingen gebeurd en we zien geen kans het te ontcijferen en we kennen niemand anders die het wel kan.* Op basis van een dergelijk argument zou uw leverancier ook een tijd lang kunnen volhouden dat het oplossen van Rubik's Kubus een moeilijk probleem is, totdat U ontdekt dat uw zoontje inmiddels enkele truckjes blijkt te beheersen waarmee hij in een minuut tijd deze puzzel vanuit een willekeurige beginstand feilloos in de correcte staat kan brengen.

In het verleden bestond het vak cryptografie dan ook uit een gestage strijd tussen de ontwerpers van systemen die op zoek waren naar codes die *moeilijk* te ontcijferen zouden moeten zijn, en codebrekers

door te laten zien dat deze codes toch *makkelijk* te breken vielen. De thans herlevende belangstelling voor dit onderwerp is mede ingegeven door de omstandigheid dat, als uitvloeisel van ontwikkelingen in de informatica, het thans mogelijk is aan de begrippen *makkelijk* en *moeilijk* een welgedefinieerde inhoud te geven die het vooreerst mogelijk maakt in principe van een systeem te bewijzen dat het *moeilijk* is (in deze welgedefinieerde technische zin). Helaas volgt hieruit in het geheel niet dat het systeem voldoet aan de in aanhef gestelde eisen. Het is zelfs zo gesteld dat de theorie in het geheel niet kan garanderen dat systemen als gezocht bestaan. Populair gezegd wordt dit onvermogen van de theorie veroorzaakt door de volgende omstandigheden: Hoe komt het dat een legitime gebruiker van een systeem berichten die volgens dit systeem verzonden zijn gemakkelijk kan decoderen ? Dit kan hij omdat hij in het bezit is van een beperkte hoeveelheid geheime informatie *(een sleutel oid.)* die de vijand niet heeft. Nu kan de vijand uiteraard proberen via list en verraad achter deze sleutel te komen maar dit wordt verondersteld *moeilijk* te zijn. De vijand kan eveneens proberen de boodschap te ontcijferen door achtereenvolgens alle mogelijke sleutels te proberen, maar in ieder zich zelf respecterend systeem is er inmiddels voor gezorgd dat het aantal mogelijke sleutels zo groot is dat dit een *moeilijke* zo niet *practisch onmogelijke* taak is. De vijand zou echter kunnen proberen de juiste sleutel te achterhalen door deze te raden, dan wel door erom te dobbelen. De kans is groot dat de vijand verliest: de gerade sleutel is niet de juiste en gebruik bij decodering levert alleen maar onzin op, maar deze methode heeft het voordeel dat zij, met niet geheel te ontkennen kans tot een goede oplossing voert (ook al zal uw commandant hier geen genoegen mee nemen), en (hetgeen voor ons de cruciale eigenschap is) in ieder geval *gemakkelijk* uitvoerbaar is.

De vraag naar het bestaan van een goed cryptografisch systeem voert ons aldus naar de vraag naar het bestaan van problemen die *moeilijk* zijn maar die met gebruik van een *gok-en-verifieer methode* gemakkelijk oplosbaar zijn. Het betreft nu precies de existentie van dit soort problemen waar het in de complexiteitstheorie beruchte P = NP ? vraagstuk over handelt, en tot nogtoe is de wetenschap onmachtig gebleken om dit probleem op te lossen door het bestaan van dit type problemen aan te tonen.

Hoewel de hedendaagse theorie dus tekort schiet om de mogelijkheid van cryptosystemen te garanderen, levert zij wel hulpmiddelen om van een gegeven systeem te bewijzen dat het *moeilijk* is, aannemende dat cryptografische systemen kunnen bestaan (een situatie die zich voordoet als $P \neq NP$). Nochthans is het maar zeer de vraag of U met een dergelijk certificaat van gegarandeerde moeilijkheid gelukkig zou zijn, aangezien dit certificaat in het algemeen slechts zal garanderen dat *sommige* volgens uw systeem gecodeerde cryptogrammen *moeilijk* te ontcijferen zijn, terwijl U, omwille van de practische bruikbaarheid van het systeem deze eis zoudt willen opleggen aan *vrijwel alle* cryptogrammen. Dit laatste vereist een ander begrip voor *moeilijkheid* dan het technische begrip van NP-volledigheid dat de hedendaagse complexiteitstheorie U levert.

In het vervolg van deze voordracht wil ik een conceptueel overzicht geven van de wiskundige theorie waaraan ik in het voorafgaande heb gerefereerd. Deze theorie is gedurende de afgelopen vijftien jaar ontstaan als uitvloeisel van de theorie der berekenbaarheid die zelf ontstaan is in de dertiger jaren van deze eeuw. We zullen de klassen P en NP invoeren en uitleggen hoe men heeft getracht te bewijzen dat deze twee klassen verschillend zijn en de rol die daarbij gespeeld wordt door de zogenaamd NP-volledige problemen. Aan het einde van deze verhandeling zult U hoplijk in staat zijn te zien hoe een en ander past in de hierboven gegeven beschouwingen.

2. THEORIE DER BEREKENBAARHEID.

Wellicht herinnert U zich uit uw schoolmeetkunde het probleem van de driedeling van de hoek. Het betreft hier het probleem om een constructie aan te geven met passer en lineaal om een willekeurige gegeven hoek in drie gelijke stukken te verdelen. Het probleem was reeds aan de oude Grieken bekend maar het heeft tot het begin van de 19e eeuw geduurd voordat de wiskunde er in geslaagd is aan te tonen dat een dergelijke constructie niet bestaat. Hierbij doet zich het verschijnsel voor dat er een groot verschil bestaat tussen het geven van een constructie en het geven van een onmogelijkheidsbewijs: van een gegeven recept kunt U snel inzien of dit al dan niet een legitieme constructie met passer en lineaal is, terwijl een onmogelijkheidsbewijs vereist dat er een mathematische theorie wordt ontwikkeld die iets verteld over alle mogelijke constructies, of deze nu in de literatuur zijn beschreven of niet. In het onderhavige geval werd deze theorie gevormd door de algebra; men beschouwde de algebraische eigenschappen van de grootheden die zich laten construeren met passer en lineaal, en plaatste deze naast de eigenschappen van de grootheden waar het bij de driedeling van de hoek om ging, en kon vaststellen dat deze eigenschappen op een wezenlijk punt uiteenliepen.

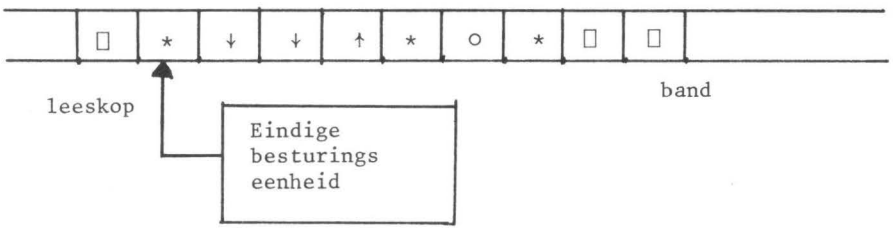
De hedendaagse theorie der berekenbaarheid is mede geinspireerd door een vergelijkbare vraag naar een onmogelijkheidsbewijs. Naast het probleem van de driedeling van de hoek blijkt de grote belangstelling van amateur wiskundigen uit te gaan naar het probleem van de grote stelling van Fermat, waarin gezocht wordt naar een niet triviale geheeltallige oplossing van de vergelijking $x^n + y^n = z^n$ voor $n \geq 3$. Het vermoeden bestaat dat dergelijke oplossingen met $xyz \neq 0$ niet bestaan, en voor vele vaste waarden van n is dit ook reeds bewezen, maar voor algemene n is dit probleem totop de huidige dag open. Een groot deel van de hedendaagse algebra en getaltheorie is onstaan uit pogingen dit probleem op te lossen.

De vergelijking waar het bij Fermat over gaat is een speciaal geval (voor vaste waarde van n althans) van een zogenaamde Diophantische vergelijking. In zijn beroemde voordracht op het Internationale Wiskunde Congres te Parijs in 1900 vroeg de Duitse wiskundige David Hilbert naar een algemene methode om voor een gegeven Diophantische vergelijking aan te tonen of deze al dan niet oplosbaar is.

Het onderzoek naar dit probleem van Hilbert heeft in 1970 tot het negatieve resultaat geleid dat een dergelijke algemene methode niet kan bestaan. De laatste stap in het bewijs werd in 1970 geleverd door de Soviet wiskundige Yu. Matijasevic, maar het conceptueel veel belangrijkere deel van het bewijs, waarmede de nonexistentie van een algemene beslissingsmethode voor een ruimere klasse van vergelijkingen reeds kon worden aangetoond, dateerde reeds van de vijftiger jaren van onze eeuw, en het is dit deel van het bewijs dat vraagt om een mathematische beschrijving van het begrip *Berekenbaarheid*. Want ook hier geldt: van een recept voor een berekeningsmethode kan men gemakkelijk nagaan of dit een legitiem recept is, maar een nonexistentiebewijs vraagt om een concept dat betrekking heeft op alle mogelijke berekeningsmethoden, gepubliceerd of niet.

De oudste aanzetten tot de benodigde mathematische beschrijving van het begrip berekenbaarheid zijn gegeven in de twintiger jaren van deze eeuw door wiskundigen als Post, Church en Kleene, maar het thans meest populaire model is in 1936 beschreven door de Engelse wiskundige Alan M. Turing. Het is dit model dat ten grondslag ligt aan de berekenbaarheidstheorie in de hedendaagse informatica en daarmede ook aan de complexiteitstheorie.

In de beschrijving van Turing worden wij geconfronteerd met een geidealiseerde rekenmachine die alles moet kunnen wat een wiskundige met potlood en papier ook kan. Deze machine bestaat uit een *Eindige besturings eenheid* en een (potentieel) *oneindige band*. De band is opgesplitst in *cellen* die netjes naast elkaar op de band liggen, waarbij op iedere cel een *symbool* uit een eindig *Alphabet* kan worden geschreven. De machine voert een *programma* uit waarbij symbolen op de band kunnen worden gelezen en zonodig kunnen worden overschreven door nieuwe symbolen, en waarbij de band naar behoefte naar links of naar rechts kan worden verplaatst, zodat de naastliggende symbolen kunnen worden gelezen. Het idee van de combinatie machine-band is aangegeven in de onderstaande figuur.



figuur 1. een Turingmachine

Het programma van de machine is opgebouwd uit instructies die ieder bestaan uit een vijftal van de vorm (q,s,q',s',m) . Hierbij zijn q en q' zogenaamde *toestanden* van de machine, afkomstig uit een eindige verzameling van toestanden die wij zullen aanduiden met het symbool K . De tekens s en s' geven *symbolen* aan uit het alphabet S van bandsymbolen. Het teken m dat de waarde L, Ø of R kan aannemen geeft een beweging aan. De betekenis van bovenstaande instructie luidt: Als de machine in toestand q het symbool s op de band ziet staan wordt dit symbool overschreven met symbool s' , voert de machine beweging m uit (L = één veld naar links met de leeskop, Ø = blijf waar je bent, en R = één veld naar rechts), en gaat verder in toestand q' . Door op deze wijze de in het programma aanwezige instructies uit te voeren kan de machine een *berekening* volbrengen. Uiteraard moet een dergelijke berekening beginnen en eindigen, en hiervoor zijn nog enkele details die nader geregeld moeten worden.

Allereerst valt het op dat de band oneindig is terwijl wij bij berekeningen denken aan bewerkingen op eindige hoeveelheden gegevens. Hiertoe is een speciaal bandsymbool (*blanco* in de figuur aangeduid met □) opgenomen in het bandalphabet, en we veronderstellen dat de gehele band op eindig vele cellen na beschreven is met het symbool blanco. Om een berekening te beginnen wijzen we bovendien een startoestand q_0 aan in de verzameling K. Verder moet de positie van de leeskop aan het begin worden vastgelegd, en hierbij geldt de afspraak dat aan het begin van de berekening er een blok van niet blanco symbolen is gegeven (de *invoer* van de berekening), waarbij in de begintoestand de leeskop staat op het meest linker symbool van de invoer. Een berekening moet ook kunnen stoppen, en dit gebeurt doordat de machine terecht komt in een configuratie waarbij een toestand q optreedt en een symbool s wordt gelezen zonder dat er voor deze combinatie q,s een instructie aanwezig is in het programma van de machine. De machine stopt de berekening in dit geval bij gebrek aan opdracht. Hieruit volgt meteen dat het maken van een volledig programma dat rekening houdt met ieder combinatie van toestand en bandsymbool voor dit model niet lonend is - de machine met een dergelijk volledig programma zou immers nooit meer tot stilstand komen.

Resteert de vraag wat de betekenis van de zojuist verklaarde berekening is. Men kan hier verschillende kanten mee uit. Het is allereerst zo dat de machine de invoer op de band kan hebben overgevoerd in een mogelijk afwijkende

rij symbolen op de band. De aldus gedefinieerde afbeelding van symbolenrij bij invoer naar symbolenrij bij uitvoer kan men beschouwen als de *functie* berekend door de machine. Maar is dit wel een functie zoals de wiskunde die kent ? Hiervoor dienen we het traditionele functiebegrip enig geweld aan te doen. We kunnen immers niet garanderen dat iedere berekening eindigt, en het is derhalve denkbaar dat voor zekere waarden van het argument de waarde van de functie niet gedefinieerd is. Een andere storende bijkomstigheid is dat de waarde van een berekening niet eenduidig hoeft vast te liggen. Ik heb namelijk bij mijn beschrijving van de machine achterwege gelaten te vereisen dat voor iedere combinatie van toestand en gelezen symbool niet meer dan één instructie aanwezig mag zijn, en het is dus denkbaar dat er in zekere configuraties meerdere legitieme voortzettingen van de berekening mogelijk zijn die niet noodzakelijkerwijs tot eenzelfde resultaat hoeven te leiden. Wiskundig gezien is de berekende functie derhalve een *relatie*.

Het geval waarbij voor iedere combinatie van symbool en toestand hoogstens één instructie aanwezig is is overigens belangrijk genoeg om appart te worden vermeld. Men spreekt in dit geval van een *Deterministische* machine terwijl in het algemene geval de machine *Nondeterministisch* wordt genoemd.

Komen wij op een tweede mogelijk gebruik van de machine. Hierbij wordt in de verzameling toestanden een tweedeling aangebracht tussen *Accepterende* en *Verwerpende* toestanden, en we zeggen dat een berekening zijn *invoer accepteert* indien de toestand waarin de machine tot stilstand komt accepterend is. Voor een deterministische machine is het zo dat een invoer hetzij geaccepteerd, hetzij *verworpen* wordt (bij stoppen in een verwerpende toestand), hetzij aanleidig geeft tot een oneindige berekening. In het nondeterministische geval kan één enkele invoer aanleiding geven tot berekeningen van al deze types tegelijk. De afspraak is dat in dit geval de invoer wordt geaccepteerd indien er minstens één accepterende berekening voor deze invoer bestaat, en zonee dan wordt de invoer verworpen. Een speciaal geval ontstaat door alle toestanden accepterend te verklaren - in dit geval wordt iedere invoer waarop een eindige *(terminerende)* berekening mogelijk is geaccepteerd.

Alvorens U te belasten met verdere definities is het wellicht nuttig een voorbeeld te geven. Beschouw een machine met toestanden $K = \{q_0, q_1, q_2\}$, en bandalphabet $S = \{0, 1, \square\}$. Het programma bestaat uit de volgende

collectie vijftallen:

$(q_0,0,q_0,0,R)$
$(q_0,1,q_0,1,R)$
$(q_0,\square,q_1,\square,L)$
$(q_1,0,q_2,1,\emptyset)$
$(q_1,1,q_1,0,L)$
$(q_1,\square,q_2,1,\emptyset)$

Uit het feit dat voor ieder paar (q,s) hoogstens één instructie in het programma is opgenomen zien we dat we hier te maken hebben met een Deterministische machine. Beschouwen wij vervolgens een typische berekening die ik noteer door volgens een zichzelve verklarende notatie de opeenvolgende configuraties die tijdens de berekening optreden op te schrijven:

$q_0 1\ 0\ 1\ 1\ \square$
$1\ q_0 0\ 1\ 1\ \square$
$1\ 0\ q_0 1\ 1\ \square$
$1\ 0\ 1\ q_0 1\ \square$
$1\ 0\ 1\ 1\ q_0 \square$
$1\ 0\ 1\ q_1 1\ \square$
$1\ 0\ q_1 1\ 0\ \square$
$1\ q_1 0\ 0\ 0\ \square$
$1\ q_2 1\ 0\ 0\ \square$

Merk op dat de toestand q_0 kennelijk bestemd is om het einde van de invoer op te zoeken. Vervolgens springt de machine naar toestand q_1 en loopt vervolgens naar links. Hierbij worden symbolen 1 vervangen door het symbool 0 totdat een symbool 0 of een blanco symbool wordt aangetroffen; dit laatste symbool wordt dan overschreven door een symbool 1 waarna de machine springt naar toestand q_2 die kennelijk geen ander doel heeft dan de machine te laten stoppen. Merk op dat , als we de inhoud van de band zouden interpreteren als een binair geschreven getal, de werking van de machine het effect heeft dat de waarde van dit getal met één wordt vermeerderd.

Wat tevens opvalt is de regelmatige structuur die optreedt in het schema waarin de berekening is aangegeven. Alle regels zijn evenlang (iets waartoe het in het algemeen nodig is om aan het begin van de berekening voldoende veel blanco symbolen van de band te noteren). Iedere regel is vrijwel gelijk aan de regel erboven, met uitzondering van de drie symbolen in de directe omgeving van het unieke optreden van een symbool in de regel die een toestand in K aangeeft. De herschrijving op deze plaats is conform een opdracht in het programma.

We hebben gezien dat de machine rekent met symbolenrijen en niet met getallen. In principe is dit echter geen beperking want symbolenrijen kunnen worden gebruikt om getallen op te schrijven. In een echte computer gebeurt dit immers ook. Een vak als de cryptografie kent overigens uit zich zelf reeds aan symbolenrijen een belangrijke betekenis toe. Het is derhalve nuttig enige notatie in te voeren. Gegeven het alphabet S noteren we de verzameling van alle rijtjes symbolen uit S met S^*, en de elementen van deze verzameling noemen we *woorden*. Op woorden kennen we de bewerking die twee woorden achter elkaar schrijft en aldus een nieuw woord vormt; deze bewerking heet *concatenatie* en wordt genoteerd als een vermenigvuldiging : het woord vw is het woord v gevlogd door het woord w. Merk op dat wel geldt $(vw)x = v(wx)$ maar niet in het algemeen $vw = wv$. De concatenatie gedraagt zich dus slechts ten dele als een gewone vermenigvuldiging. Er bestaat een speciaal woord te weten het lege rijtje symbolen. Omdat dit zich slecht laat opschrijven gebruiken wij hiervoor de notatie ε. Merk op dat $w\varepsilon = \varepsilon w = w$. Het aantal symbolen dat in een woord optreedt heet de *lengte* van het woord, notatie $|w|$. Een verzameling woorden (een deelverzameling van S^* dus) heet een *taal*.

Zonder beperking van de algemeenheid mogen wij veronderstellen dat alle verder te beschouwen Turingmachines gebruik maken van eenzelfde bandalphabet S. Men kan immers altijd nieuwe symbolen invoeren door een rijtje symbolen uit S op te vatten als een gecodeerd nieuw symbool : op een echte computer gebeurt dit immers ook - daar worden alle symbolen gecodeerd in bits (nullen en enen). Gemakshalve zullen wij S gelijk nemen aan $S = \{0,1,\square\}$.

Een Turingmachine is nu verder geheel bepaald door zijn programma. Om eenheid te brengen in de mogelijke verzameling toestanden veronderstellen we dat alle toestanden afkomstig zijn uit een oneindige verzameling toestanden $\{q_0,q_1,q_2,\ldots\ldots\}$ zodat alleen het nummer van de toestand relevant is. Iedere machine zal slechts eindig vele toestanden gebruiken. Een programma kunnen wij nu geheel beschreven achten door een eindige deelverzameling van het Cartesisch product $\mathbb{N} \times S \times \mathbb{N} \times S \times \{L,R,\emptyset\}$. Een dergelijk programma laat zich zelf weer coderen door een rij symbolen uit S. Het blijkt nu mogelijk te zijn deze codering zo te kiezen dat een Turingmachine kan worden ontworpen die een dergelijk gecodeert programma, gevolgd door een eveneens gecodeerde invoer ontvangt, en

die vervolgens de met het programma beoogde berekening op deze invoer stap voor stap *simuleert*. Dit laat zich aangeven door de volgende formule:

$$M_u(i,x) = M_i(x)$$

Hierbij stelt M_i de machine met programma i voor, terwijl het paar (i,x) de gecodeerde combinatie van programma i en invoer x is.
De hierbij optredende machine M_u noemt men een *Universele Turingmachine* .
Het = teken heeft hier de volgende betekenis: als een van beide berekeningen termineert dan termineert ook de andere berekening en hebben beide berekeningen (modulo het effect van de codering die ertussen zit) dezelfde uitkomst.

Ik heb enkele paginas terug het gehad over de verschillende gebruiksmogelijkheden die een Turingmachine heeft. We kunnen de machine gebruiken om een relatie in $S^* \times S^*$ te berekenen, maar dit is voor ons een weinig zinvolle bezigheid. In het algemeen zullen wij ons beperken tot het geval waarbij deze relatie een echte functie is, bijvoorbeeld omdat de machine deterministisch is en voor iedere invoer stopt. Een op deze wijze verkregen functie noemt men een *Recursieve functie*. Laat men de eis vallen dat de functie voor iedere invoer is gedefinieerd dan spreekt men van een *Partieel Recursieve functie*.

Voor het gebruik van de machine om invoerwoorden te accepteren kunnen wij gemakshalve veronderstellen dat alleen de toestand met het hoogste nummer die in het programma van de machine optreedt accepterend is terwijl alle andere toestanden verwerpen zijn. Dit spaart ons de zorg om de verzameling van accepterende toestanden nader te specificeren. In het geval dat de machine voor iedere invoer stopt ongeacht de loop van de berekening (in het geval van nondeterminisme) kunnen we de collectie woorden vormen waarvoor een accepterende berekening bestaat: de aldus bepaalde taal heet de taal *herkend door* de machine en een op deze wijze herkende taal heet *Recursief*.

Tenslotte was er het gebruik waarbij de machine accepteert door te stoppen. De taal bestaande uit die invoerrijen waarvoor de machine kan stoppen heet de taal *geaccepteerd door* de machine en een aldus bepaalde taal heet *Recursief opsombaar*.

Het is niet moeilijk in te zien dat een machine bestemd om woorden te herkennen kan worden veranderd in een machine die woorden accepteert door te stoppen, eenvoudig door iedere verwerpende toestand waarin de berekening zou kunnen stoppen te voorzien van een "doe niets" instructie waardoor de machine in een oneindige berekening geraakt. Hieruit volgt dat alle Recursieve

talen tevens recursief opsombaar zijn. De omkering hiervan geldt niet. Een voorbeeld van een taal die wel recursief opsombaar is maar niet recursief wordt gegeven door het zogenaamde *stopprobleem* : deze taal laat zich definieren (modulo de nodige coderingsafspraken) door:

halt := $\{i \mid M_i(i)$ termineert $\}$.

Terminatie wil hier zeggen terminatie in een mogelijke berekening. Dat halt recursief opsombaar is laat zich bewijzen door, gebruik makende van de universele machine een programma te ontwerpen waarmee halt zich laat accepteren. Dat halt niet recursief is volgt door het onderstaande *diagonalisatie* argument. Veronderstel dat halt wordt herkend door een machine M_h . Het is uiteraard denkbaar dat M_h een nondeterministische machine is, maar het blijkt mogelijk te zijn in dat geval een nieuwe machine te ontwerpen die alle mogelijke berekeningen van M_h nazoekt om te kijken of er soms een accepterende berekening bij zit. We mogen daarom veronderstellen dat M_h deterministisch is. Koppeling van M_h met de universele machine en wat eenvoudig sleutelwerk zou ons vervolgens een machine M_d opleveren die de volgende taal zou moeten herkennen:

diag := $\{i \mid M_i$ is deterministisch en $M_i(i)$ stopt en verwerpt$\}$

Tenslotte is de test of M_i deterministisch is een eenvoudige controle op het programma dat in i gecodeerd staat, voor de test of $M_i(i)$ stopt hebben we de machine M_h (die gegarandeerd met een antwoord komt) en de universele machine vertelt ons, in het nog te analyseren geval dat $M_i(i)$ stopt, na eindig veel stappen of deze berekening al dan niet accepterend is (twijfel is niet mogelijk want M_i was immers deterministisch). Helaas blijkt dat de keuze $i = d$ tot een tegenspraak voert: volgens de constructie zal de berekenig van $M_d(d)$ die geheel deterministisch is beschreven altijd stoppen en accepteren dan en slechts dan als zij verwerpt en omgekeerd.

De oplettende lezer zal opmerken dat in het bovenstaande bewijs nog al wat details onder de tafel zijn verdwenen. Het nondeterminisme is gemakshalve terzijde gesteld (daarover later meer) en er is voortdurend een beroep gedaan op mogelijke koppelingen van machines waarvan U intuitief kunt inzien dat dit een berekenbaar proces beschrijft, maar waarvoor een formele verificatie binnen het zojuist ontwikkelde formalisme achterwege blijft.

Wij komen hiermee op de vraag inhoeverre het door ons geschetste model adequaat is om te worden gebruikt als model voor het algemene begrip "berekenbaarheid". In de practijk leert de theorie ons dat in principe ieder recept waarvan men heeft kunnen inzien dat het een effectieve berekeningsmethode voorstelt, zich met het nodige handen en voeten werk laat uitprogrammeren voor een Turingmachine, maar er is niemand die dit in de practijk ook echt zal doen. Wat wel is aangetoond is dat het Turingmachine model precies dezelfde klasse van berekenbare functies en relaties en dezelfde klassen van recursieve of recursief opsombare talen oplevert als alle ander abstracte modellen die in de loop de tijden zijn voorgesteld. Deze equivalenties worden aangetoond door een machinemodel te simuleren in het andere formalisme en omgekeerd, en de practijk met deze simulaties opgedaan levert de overtuiging dat we inderdaat te maken hebben met een welbeschreven klasse van berekenbare functies etc. . Een en ander is verwoord in de zogenaamde *these van Church:* Iedere berekenbare functie kan worden berekend met een Turingmachine.

Ter afsluiting van dit overzicht over de berekenbaarheidstheorie wil ik nog noemen het begrip *reductie*. Een reductie f is een berekenbare (dus recursieve) afbeelding van S^* naar S^* . We zeggen dat f de taal A reduceert tot de taal B (notatie $A \leq B$ via f) indien geldt $f(x) \in B$ dan en slechts dan als $x \in A$. In deze situatie geldt dat recursiviteit van B impliceert dat A eveneens recursief is, aangezien koppeling van de afbeelding f met de machine die B herkent een mechanisme oplevert dat A herkent. Weten wij derhalve reeds dat A niet recursief is, bijvoorbeeld omdat $A =$ <u>halt</u> , dan volgt hieruit dat B al evenmin recursief kan zijn. Op deze wijze worden reducties gebruikt om de *onbeslisbaarheid*, dat is de niet recursiviteit van zekere talen aan te tonen. Hiermee komen wij terug bij het in aanhef van deze paragraaf genoemde probleem van Hilbert. De vraag naar het bestaan van een methode om vast te stellen of een willekeurige Diophantische vergelijking oplosbaar is, laat zich alsvolgt formaliseren: geef een codering voor alle Diophantische vergelijkingen en vorm een taal D bestaande uit alle coderingen van oplosbare vergelijkingen. Kan deze taal D recursief zijn ? Het antwoord luidt ontkennend. Het blijkt immers mogelijk te zijn een reductie te beschrijven die een Turingmachineprogramma i vertaalt in een Diophantische vergelijking (althans de code ervan) V_i , zodanig dat V_i oplosbaar is dan en slechts dan als $M_i(i)$ stopt. Dit levert ons een reductie van <u>halt</u> tot D en daarom is D niet recursief.

3. COMPLEXITEIT.

Het voorafgaande overzicht over de berekenbaarheidstheorie heeft een antwoord gegeven op de vraag wat principieel *doenlijk* of *ondoenlijk* is, maar de vraag naar een formalisatie van de in de aanhef genoemde begrippen *makkelijk* en *moeilijk* is nog niet beantwoord. In deze paragraaf zullen we echter zien dat het Turingmachine model ons in staat stelt aan berekeningen grootheden als rekentijd en geheugengebruik toe te kennen, die vervolgens kunnen worden gebruikt om de gevraagde definities te geven.

Gegeven een berekening van een Turingmachine. De *rekentijd* van deze berekening is het aantal keren dat tijdens de berekening een instructie wordt uitgevoerd. Gebruiken we een schema als aangegeven in paragraaf 2 dan is dit aantal één minder dan het aantal regels in het schema voor de berekening. Het *geheugengebruik* is het aantal bandsymbolen dat tijdens de berekening ooit door de leeskop is gelezen.

In het geval van een nondeterministische machine hangt het geheugengebruik en de rekentijd uiteraard af van de gekozen berekening. De afspraak is dat hierbij maatgevend is de rekentijd resp. het geheugengebruik, van de zuinigste accepterende (voor het geval van recursief opsombare verzamelingen terminerende) berekening.

Rekentijd en geheugengebruik voor een gegeven machine hangen uiteraard af van de beschouwde invoer. In de Complexiteitstheorie is het gewoonte deze grootheden te begrenzen door middel van een functie in de lengte van de invoer in plaats van door een functie in de invoer zelf. Dit leidt tot de volgende definities:

Zij $f(n)$ een functie van $\mathbb{N}$ naar $\mathbb{N}$. De machine M_i accepteert (herkent) een taal A in tijd $f(n)$ (geheugengebruik $f(n)$), indien voor iedere invoer $x \in A$ geldt dat x wordt geaccepteerd (herkend) door M_i in tijd (geheugengebruik) $\leq f(|x|)$, terwijl iedere invoer buiten A wordt verworpen door de machine M_i.

Merk op dat in de bovenstaande definitie in feite vier definities tegelijkertijd worden gegeven. We hebben nog niet vertelt wat het betekent om een functie te berekenen in tijd of geheugen $f(n)$, maar deze uitbreiding zult U inmiddels zelf wel kunnen aangeven. Merk tevens op dat de definitie geen eis oplegt aan de complexiteit van verwerpende berekeningen. In de practijk is het zo dat met een kleine wijziging van het machineprogramma

bewerkstelligd kan worden dat alle berekeningen in rekentijd (geheugengebruik) begrensd worden door $f(n)$, mits de functie $f(n)$ voldoende netjes is, en de grenzen waarnaar wij in het algemeen zullen kijken, zoals $f(n) = n , n^2 , \ldots , n^k , 2^n , 3^n , \ldots , 2^{2^n} , \ldots, n.\log(n), ..$ hebben deze eigenschap. Het opleggen van eisen aan verwerpende berekeningen heeft overigens alleen maar zin in het geval van het herkennen van talen; de mogelijkheid deze extra eis op te leggen levert overigens meteen een bewijs dat een taal die in begrensde tijd of geheugengebruik wordt herkend of geaccepteerd automatisch recursief is.

Tenslotte de complexiteit van een taal A zelve. Het hierboven gedefinieerde begrip is gekoppeld aan een machine die A herkent, en het zou derhalve voor de hand liggen te kijken naar de minimale tijd (geheugengebruik) waarbinnen de taal A door een machine M_i wordt herkend, doch niemand garandeert dat een dergelijk minimum van een collectie functies bestaat. Integendeel - in het algemeen bestaat zo'n minimum niet. We kunnen echter wel vaststellen dat een taal herkend wordt binnen een zekere grens indien zij door een machine herkend wordt binnen die grens. Om te ontkennen dat een taal binnen de grens $f(n)$ wordt herkend dienen wij derhalve aan te tonen dat iedere machine die A herkent voor zekere invoer $x \in A$ meer dan $f(|x|)$ rekentijd resp. geheugengebruik vergt. Merk op dat opnieuw gevraagd wordt iets aan te tonen voor alle mogelijke machines die A herkennen, of deze nu bekend zijn of niet. Het moge duidelijk zijn dat het voldoen aan deze conditie niet gemakkelijk zal zijn, en het aantonen van goede ondergrenzen is dan ook een van de lastige onopgeloste problemen in de complexiteitstheorie.

Het is een redelijke veronderstelling dat bij iedere berekening de gehele invoer gelezen wordt, alvorens wordt vastgesteld of deze al dan niet tot de taal behoort. Dit betekent dat automatisch een lineaire ondergrens geldt voor rekentijd en geheugengebruik. Voor wat betreft de rekentijd is dit acceptabel, maar voor het geheugengebruik ware het wenselijk te kunnen werken met sublineair geheugengebruik (grenzen als 0 (geen geheugengebruik), 1,2,3, ... (constant geheugengebruik), $\log\log(n)$, $\log(n)$, $\log^2(n)$, ... etc. Hiertoe blijkt het zinnig het model van de Turingmachine wat uit te breiden. De machine wordt versterkt door deze uit te rusten met meerdere banden, waarbij een van de banden wordt gebruikt om de invoer op te schrijven. Deze invoerband kan wel worden gelezen maar niet worden beschreven, en dit levert een excuus om het geheugengebruik op

de invoerband niet mee te tellen. Men kan, in het geval dat men een functie of een relatie wil berekenen, ook een speciale band aanwijzen als uitvoerband; dit is dan een band waarop alleen maar geschreven wordt, en waarvan nooit gelezen wordt, en ook het geheugengebruik op deze band telt niet mee. De overige banden kunnen worden gelezen en beschreven; wij noemen dit de werkbanden, en alleen het geheugengebruik op deze werkbanden wordt gemeten.

Een volgende versterking is dat toegestaan wordt dat meerdere koppen op een werkband aanwezig zijn. Deze koppen kunnen voelen of zij hetzelfde veld van de band aan het lezen zijn, zodat het programma op dit type configuraties kan reageren. Men kan in plaats van lineaire banden kijken naa twee- of hoger dimensionale opslagmedia, met een corresponderende uitbreiding van het bewegingsrepertoire voor de leeskoppen. Tenslotte kan men instructies toelaten waarbij de leeskoppen in één stap naar elkaars positie toespringen. Variaties van het Turingmachine model bestaan er dus in overvloed.

In paragraaf 2 vermelde ik reeds dat alle machinemodellen die bekend zijn in principe dezelfde klasse van functies kunnen berekenen en dezelfde klasse talen kunnen herkennen en accepteren. Dit geldt in het bijzonder voor de hierboven genoemde varianten van Turingmachines. Het blijkt echter wel zo te zijn dat de benodigde rekentijd en geheugengebruik van model tot model verschillend kan zijn. Zo is het bijvoorbeeld zo dat het herkennen van de taal der palindromen (dat zijn woorden die achterstevoren geschreven zichzelf opleveren, zoals het woord "PARTERRETRAP") op een Turingmachine met één band rekentijd n^2 vraagt, terwijl een machine met meerdere banden (of meerdere koppen op één band) dit in lineaire tijd kan klaren.

Een onaangename situatie dreigt hier: de complexiteit van een probleem zou aldus wezenlijk af kunnen hangen van het gekozen machinemodel. Gelukkig blijkt deze afhankelijkheid in de practijk mee te vallen. De verschillende modellen blijken elkaar te kunnen simuleren op een voor de hand liggende wijze waarbij de rekentijd met nooit meer dan een polynomiale factor wordt vermeerderd: als de gesimuleerde machine tijd $f(n)$ vraagt, gebruikt het simulerende programma tijd $(f(n))^k$ voor een getal k dat in het algemeen niet groter dan 2 is. Tegelijkertijd blijkt het geheugengebruik voor deze simulatie op zijn hoogst te worden vermenigvuldigd met een constante factor. Deze eigenschap blijft behouden indien we kijken naar alternatieve modellen voor berekenbaarheid, waarbij de, op een echte computer geinspireerde *registermachines* zeker genoemd moeten worden. Ook deze machines kunnen Turingmachines simuleren, dan wel door Turingmachines worden gesimuleerd

met een overhead in rekentijd die polynomiaal begrensd is, en op straffe van een constante factor in het geheugengebruik. Een en ander geldt slechts voorzoverre rekentijd en geheugengebruik van deze registermachines op een correcte wijze wordt gemeten, en deze machines niet worden uitgerust met te machtige instructies, een problematiek waarop ik hier verder niet kan ingaan. Laat ik volstaan met de mededeling dat de Turingmachine aldus een klasse van machinemodellen representeert, die als practisch realistisch, wordt ervaren. Modulo een polynomiale overhead in tijd, resp. een constante factor in geheugengebruik, stelt dit model ons in staat uitspraken te doen over in de practijk bruikbare methoden om concrete problemen op te lossen.

We zijn dan nu in staat de gevraagde formalisatie van de begrippen *makkelijk* en *moeilijk* te geven. We stellen vast dat een taal A makkelijk is indien zij herkend wordt op een DETERMINISTISCHE Turingmachine binnen tijd $f(n) = k.n^k$ voor een zekere waarde van k. De klasse van talen die volgens deze definitie gemakkelijk zijn noemen wij P (staat voor herkenbaar in polynomiale tijd). Let op het optreden van het woord Determinisme in deze definitie. In de voorafgaande paragraaf heb ik tenslotte verteld dat we bij het gebruik als herkenningsmechanisme het zonder nondeterminisme kunnen stellen, doch ampele overweging leert dat de aldaar gesuggereerde simulatiemethode van het nalopen van alle mogelijke berekeningen in het algemeen de rekentijd op exponentiele wijze opblaast.

Niets belet ons echter om een nieuwe verzameling te definieren van die talen die makkelijk worden als we nondeterminisme toestaan. Dit is de verzameling NP bestaande uit die talen die herkend worden binnen een tijd $f(n) = k.n^k$ voor zekere k door een niet noodzakelijk deterministische Turingmachine.

Merk op dat beide bovenstaande verzamelingen goed gedefinieerd zijn, inzoverre dat het exacte model van de gebruikte Turingmachine er niets toe doet, bij gratie van de polynomiale overhead waarmee de modellen elkaar simuleren. Om aan te tonen dat een taal tot P behoort mogen wij gebruik maken van Turingmachines uitgerust met alle wenselijke toeters en bellen, terwijl wij voor een bewijs dat A niet tot P behoort uit mogen gaan van een machine die A herkent op één band.

De klassen P en NP zijn niet de enige klassen die in de literatuur optreden. Het aanleggen van begrenzingen op het geheugengebruik geeft aanleiding tot de klassen LOGSPACE resp. NLOGSPACE en PSPACE (geheugengebruik begrensd door $k.\log(n)$ resp. $k.n^k$ voor zekere waarde van k). Daarnaast kennen wij de klassen EXPTIME en NEXPTIME.

Deze klassen zijn in elkaar bevat volgens het schema :

LOGSPACE $\subset$ NLOGSPACE $\subset$ P $\subset$ NP $\subset$ PSPACE $\subset$ EXPTIME $\subset$ NEXPTIME

U zult zich wellicht afvragen waarom er in dit rijtje geen klasse NPSPACE is opgenomen. De verklaring voor dit feit wordt gegeven door een stelling in 1970 bewezen door de Amerikaanse informaticus Walter Savage die aantoont dat nondeterminisme gesimuleerd kan worden met een kwadratische overhead in geheugengebruik, en deze stelling impliceert dat de klasse NPSPACE gelijk zou zijn aan de klasse PSPACE. De overige inclusies laten zich gemakkelijk bewijzen op grond van de volgende triviale argumenten:

- het geheugengebruik wordt begrensd door de rekentijd
- in een berekening is het zinloos twee keer eenzelfde configuratie op te laten treden
- het aantal verschillende configuraties wordt begrensd door het product van de lengte van de invoer en een exponentiele functie in het geheugengebruik

De enige inclusie die hierna nog problemen geeft is NLOGSPACE $\subset$ P . Deze wordt echter bewezen door de (polynomiaal begrensde) lijst van configuraties op te vatten als een doolhof waarbinnen een schat (de accepterende configuratie) gezocht moet worden, en te bedenken dat er algorithmen bestaan die een doolhof in polynomiaal begrensde tijd kunnen doorzoeken.

De complexiteitstheorie laat het echter afweten (tot heden toe althans) als het erom gaat de vraag te beantwoorden of de bovengenoemde inclusies echte inclusies zijn. Is ieder klasse echt groter dan de eraan voorafgaande in het rijtje ? We weten weinig meer dan dat in het rijtje echte inclusies moeten voorkomen; er geldt immers NLOGSPACE $\neq$ PSPACE , P $\neq$ EXPTIME , en NP $\neq$ NEXPTIME . De meest bekende van deze open problemen is de vraag of de gelijkheid P = NP al dan niet geldt. Dit is precies de vraag of een probleem dat met gebruik van nondeterministische hulpmiddelen makkelijk opgelost kan worden al dan niet automatisch makkelijk is, en zoals ik in de inleiding van dit verhaal heb gesuggereerd is deze vraag van belang voor de cryptografie.

In onze formalisatie noemen we alle talen die niet makkelijk zijn moeilijk. Vaak noemt men dit *exponentiele problemen*, maar dit is een misleidende naamgeving. Het is immers denkbaar dat een moeilijke taal herkenbaar is in tijd $n^{\log\log\log(n)}$, een grens die harder groeit dan ieder polynoom, maar minder hard dan een exponentiele functie.

4. NP-VOLLEDIGHEID.

In het voorafgaande gedeelte hebben we het probleem gezien of de gelijkheid $P = NP$ al dan niet geldig is. Dit is in feite de vraag naar het bestaan van een taal A die zich makkelijk laat herkennen met gebruik van nondeterminisme en die desalniettemin moeilijk is, in de zin dat iedere deterministische machine die de taal herkent daar (voor zekere invoer) meer dan $k.n^k$ rekenstappen voor nodig heeft, hoe groot k ook gekozen wordt (waarbij k uiteraard niet van de gekozen invoer mag afhangen). Om dit probleem nader te kunnen apprecieren dienen wij een indruk te krijgen van het type talen dat tot NP behoort. Het blijkt dat een groot aantal practijk-problemen uit de combinatorische optimalisering zich laten herformuleren tot het herkenningsprobleem voor een geschikte taal, waarbij dit herkennings-probleem tot de klasse NP blijkt te behoren.

Een representatief voorbeeld van een dergelijk probleem uit de combinatorische optimalisering is het zogenaamde *Knapsack* probleem. Veronderstel dat U beschikt over een bedrag van N florijnen, waarvoor U een aantal geschenken voor uw dame kan aanschaffen. De door U benaderde leverancier beschikt over de objecten $a_1, a_2, \ldots, a_m$ die voorzien zijn van prijskaartjes voor de respectiefelijke bedragen van $n_1, n_2, \ldots n_m$ florijnen. Om een goede sier te maken bij Uw dame bent U er op uit om een zo groot mogelijk bedrag uit te geven, een en ander echter binnen het beschikbare budget, want de winkelier geeft geen crediet, om over kortingen maar te zwijgen. Wat is in dit geval de beste oplossing ?

In deze vorm hebben wij te maken met een optimaliseringsprobleem. Wij verkrijgen hieruit een *beslissingsprobleem* door aan de gegevens van het probleem een streefbedrag $N' \leq N$ toe te voegen en vervolgens de vraag te stellen: is het mogelijk exact het bedrag van N' florijnen te besteden ? Vervolgens kunnen wij uit dit beslissingsprobleem een taal destileren, doordat we alle gegevens van het probleem in gecodeerde vorm achterelkaar schrijven. Iedere *Instantie* van het probleem wordt daarbij een woord, en de taal bestaat uit al die woorden die afkomstig zijn van instanties waarvoor de gecodeerde vraag met "Ja" kan worden beantwoord. Het domein S^* wordt als het ware opgesplitst in drie delen: de collectie gecodeerde instanties die *oplosbaar* zijn, de collectie gecodeerde instanties die *niet oplosbaar* zijn, en tenslotte die tekenrijen die niet de codering van een instantie van het onderhavige probleem zijn. Deze laatste klasse is in het algemeen gemakkelijk herkenbaar op grond van overtredingen tegen de regels der gebruikte codering.

Over deze codering dienen wij een belangrijke opmerking te maken. De bedragen uit het Knapsack probleem zijn uiteraard gehele getallen, en die moeten worden opgeschreven. In het algemeen zal daarvoor een vorm van *binaire* of *decimale* schrijfwijze worden gebruikt, waarbij de lengte van de tekenrij nodig om het getal n te representeren evenredig is met $^2\log(n)+1$ (afgezien van het getal 0 waarvoor deze formule ongedefinieerd is, maar dat zeker één teken vergt om het op te schrijven). Nu kent de klassieke recursietheorie die we in paragraaf 2 hebben besproken een alternatieve schrijfwijze voor getallen, te weten de *unaire* codering, waarbij het getal n gecodeerd wordt door een rij van $n+1$ optredens van het teken "1" . Voor de formulering van het wiskundige probleem doet het er niets toe of we decimale, binaire of unaire codering gebruiken, maar voor de complexiteit van de met het herkenningsprobleem corresponderende taal is dit van groot belang. Immers, door gebruik van unaire codering wordt de invoer exponentieel langer zodat er exponentieel langer kan worden gerekend zonder dat het probleem ophoudt makkelijk te zijn. Zo leert de practijk dat het genoemde Knapsack probleem bij gebruik van unaire codering makkelijk is; er bestaat een *dynamische programmering algorithme* waarin als het ware een lijst wordt gemaakt van alle totaalbedragen die kunnen worden uitgegeven, en deze berekeningsmethode vergt een rekentijd die polynomiaal is in N . Of het probleem ook makkelijk is bij gebruik van binaire codering is echter een open vraag; merk op dat de waarde van het getal N niet polynomiaal begrensd wordt door de lengte benodigd om het getal N op te schrijven, want die is immers van de orde $\log(N)$.

Als wij dus in het vervolg spreken over een probleem dat al dan niet tot een klasse als P of NP behoort, dan zullen wij gemakshalve volstaan met de formulering van een wiskundige vraag, en alle aspecten van de vertaling van het beslissingsprobleem in een taal onvermeld laten. We gaan er hierbij wel van uit dat eventueel optredende getallen binair worden gecodeerd tenzij expliciet wordt vermeld dat dit niet zo is.

De formulering van het probleem kan er uitzien als gesuggereerd door het voorbeeld:

KNAPSACK.

INSTANTIE: een lijst getallen $n_1, n_2, \ldots, n_m$, een getal N'
VRAAG: Bestaat er een oplossing van de vgl. $\sum_{i=1}^{m} x_i n_i = N'$ met $x_i \in \{0,1\}$?

Het moge U duidelijk zijn dat het hierboven als voorbeeld genoemde Knapsack probleem tot de klasse NP behoort. De oplosbaarheid van een instantie laat zich immers vaststellen door de juiste objecten bij elkaar te kiezen en via een simpele optelling te verifieren dat de totale kosten exact uitkomen op N' florijnen. Ons machinemodel zou niet deugen als het uitvoeren van deze optelling niet makkelijk is. Merk hierbij op dat, indien de gegeven instantie oplosbaar is, wij via gokken achter de goede oplossing zouden kunnen komen, en dit is nu precies de extra faciliteit die het nondeterministische machinemodel ons biedt.

We hebben dus tevens een candidaat in handen om te bewijzen dat $P \neq NP$; het is immers voldoende aan te tonen dat KNAPSACK niet tot de klasse P behoort, dwz. dat ieder algorithme voor het Knapsack probleem voor sommige instanties meer dan polynomiaal begrensde rekentijd vergt. De complexiteitstheorie heeft helaas tot de huidige dag niet een dergelijk bewijs kunnen leveren. Wat zij wel heeft aangetoond is dat wij kunnen volstaan met een onderzoek van het Knapsack probleem. Mocht, geheel tegen de verwachting in blijken dat het Knapsack probleem wel makkelijk is, dan kunnen wij verdere pogingen te zoeken naar moeilijke problemen in NP wel achterwege laten want dan geldt $P = NP$ (en dan is het afgelopen met de cryptografie). Deze speciale positie dankt het Knapsack probleem aan zijn eigenschap *NP-volledig* te zijn. In het vervolg van deze paragraaf wil ik uitleggen wat deze eigenschap inhoudt, en hoe voor een gegeven probleem kan worden aangetoont dat het de eigenschap NP-volledigheid bezit.

In paragraaf 2 heb ik gesproken over het begrip *reductie* . Veronderstel dat wij een taal A reduceren tot een taal B met een afbeelding f , en veronderstel dat deze afbeelding f zich *makkelijk* (dwz. in polynomiale tijd) laat berekenen. Dan geldt dat als het probleem B makkelijk is (dus B behoort tot de klasse P) dan is A eveneens makkelijk. Om instanties van A op te lossen gebruiken wij de vertaling f om vervolgens de gemakkelijk veronderstelde oplossingsmethode voor probleem B aan te roepen, en deze samenstelling van makkelijke methoden is weer makkelijk. Op basis van dezelfde redenering volgt dat B moeilijk is indien A moeilijk is .

Stel nu dat taal B in NP de eigenschap bezit dat voor iedere taal A in NP er een in polynomiale tijd berekenbare reductie f van A naar B bestaat. Een taal B met deze eigenschap noemen we *NP-volledig* . Als B makkelijk is volgt dat iedere taal in NP makkelijk is (dus $P = NP$), terwijl aanwezigheid van een moeilijke taal in NP impliceert dat B moeilijk is.

Merk op dat de eigenschap NP-volledigheid op basis van onze huidige kennis niet impliceert dat B moeilijk is, want dit zou immers aantonen dat $P \neq NP$. We leiden er echter uit af dat B zo ongeveer het lastigste probleem is dat tot de klasse NP behoort, want door de vertaling zijn alle andere problemen in NP tot het probleem B te herleiden. Dit laatste wil niet zeggen dat de andere problemen minder rekentijd zouden vergen, want bij de vertaling kan de lengte van een instantie met een polynomiale factor worden vergroot, en dit speelt een rol bij de bepaling van de feitelijke complexiteit van een probleem.

Als het probleem A polynomiaal zich laat reduceren tot probleem B en B polynomiaal gereduceerd kan worden tot een probleem C , dan is de samenstelling van deze twee reducties weer een polynomiale reductie. Deze transitiviteit van het reductiebegrip heeft tot gevolg dat de volgende bewering geldt:

Als probleem A polynomiaal reduceerbaar is tot probleem B in NP, en A is NP-volledig , dan is ook B NP-volledig .

Ieder probleem C in NP kan immers worden gereduceerd tot A en via de transitive eigenschap van de reducties daarmee tot B.

Deze laatste eigenschap laat zien hoe we in de practijk kunnen aantonen dat een probleem B in NP de eigenschap NP-volledigheid heeft. Men neme een probleem A waarvan men reeds weet dat het NP-volledig is, en men tone aan dat A zich polynomiaal tot B laat reduceren. Dit in de literatuur met zeer veel succes toegepaste recept vergt uiteraard een NP-volledig probleem om mee te beginnen. Het bestaan van een dergelijk probleem is in 1971 aangetoond door de Amerikaanse informaticus S. Cook , en ik wil U in de hiernavolgende paragraaf een indruk geven van zijn bewijsmethode.

Voordien echter enkele voorbeelden van polynomiale reducties. Beschouw het volgende probleem:

BOEDELSCHEIDING.

INSTANTIE: een lijst getallen $n_1, n_2, \ldots, n_m$
VRAAG: Bestaat er een deelverzameling $J \subset \{1,2,\ldots,m\}$ zó dat $\sum_{i \in J} n_i = \sum_{i \notin J} n_i$?

Een instantie van BOEDELSCHEIDING laat zich op natuurlijke wijze opvatten als een instantie van KNAPSACK door als streefbedrag $(\sum_{i=1}^{m} n_i)/2$ te kiezen. Deze simpele vertaling is een voorbeeld van een polynomiale reductie.

Omgekeerd is het mogelijk een willekeurige instantie van het Knapsack probleem te vertalen in een ermee equivalente instantie van het probleem BOEDELSCHEIDING. Beschouw een instantie van KNAPSACK met lijst getallen $n_1, n_2, \ldots, n_m$ en streefbedrag N'. Zij $M = \sum_{i=1}^{m} n_i$. Wij vormen nu een instantie van BOEDELSCHEIDING door een boedel te vormen bestaande uit de gegeven objecten van de Knapsack instantie $n_1, n_2, \ldots, n_m$ en een tweetal nieuwe objecten met prijs $2M-N'$ resp. $M+N'$. De totale waarde van deze boedel bedraagt $4M$ zodat iedere helft de waarde $2M$ moet krijgen.

Stel nu dat de gegeven instantie van het KNAPSACK probleem oplosbaar is. Dan bestaat er een deelverzameling $J \subset \{1,2,\ldots,n\}$ zódat $\sum_{i \in J} n_i = N'$. De boedel laat zich eerlijk verdelen door in de eerste helft alle objecten in J te stoppen tezamen met het nieuwe object met waarde $2M-N'$.

Omgekeerd kan men aantonen dat iedere eerlijke verdeling van de boedel van de hierboven gesuggereerde vorm is, dwz. afkomstig van een oplossing van de Knapsack instantie. Het is immers onmogelijk een van beide delen beide nieuwe objecten toe te wijzen aangezien de som van de twee nieuwe objecten reeds de waarde $3M$ heeft, hetgeen reeds drie kwart van de totale waarde is. De helft die het object met waarde $2M-N'$ omvat dient derhalve te worden aangevuld met "goedkope" objecten tot een totaal van N', en de aldus gevormde collectie "goedkope" objecten levert ons een oplossing van de gegeven Knapsack instantie.

De bovenstaande beschrijving levert ons nog geen polynomiale reductie in de formele zin; we hebben slechts geformuleerd in termen van het onderliggende wiskundige probleem, terwijl de reductie moet werken op tekenrijen die de gecodeerde instanties van het probleem voorstellen. Ons vertrouwen in onze programmeerkunde, en de macht van het machinemodel is echter zo groot dat wij gemakshalve aan deze technische details voorbijgaan. De reductie hoeft immers niet veel meer te doen dan uit de codering een lijst getallen destileren en enkele simpele optellingen en aftrekkingen uit te voeren, om vervolgens een nieuwe lijst getallen in gecodeerde vorm te produceren. Met onze nalatigheid bevinden wij ons overigens in goed gezelschap want in de gehele literatuur van de complexiteitstheorie worden dit soort details onvermeld gelaten.

Het netto resultaat van de bovenvermelde beschouwing is dat de problemen KNAPSACK en BOEDELSCHEIDING gelijkwaardig zijn in de zin dat zij hetzij geen van beiden, hetzij beiden NP-volledig zijn. De theorie leert dat de laatstgenoemde situatie zich voordoet.

5. DE STELLING VAN COOK.

In dit gedeelte wil ik U een indruk geven van het bewijs van de stelling van Cook die uitspreekt dat er NP-volledige problemen bestaan. In feite toont deze stelling de NP-volledigheid aan van een specifiek probleem dat bekend staat onder de naam SATISFIABILITY, maar de formulering van dit probleem vergt een minimale voorkennis over het vak propositielogica, en ik wil er niet van uitgaan dat mijn huidige gehoor deze voorkennis bezit. Bovendien vereist het volgen van de standaardliteratuur dat, na het bewijs van de NP-volledigheid van dit probleem een keten van reducties wordt beschreven die uiteindelijk leidt tot de NP-volledigheid van het Knapsack probleem. In plaats hiervan wil ik U een schets geven van een rechtstreeks bewijs van de laatsgenoemde NP-volledigheid waarbij slechts gebruik gemaakt wordt van een tussenprobleem dat wij zullen aanduiden met de naam LEGPUZZEL.

In de paragraaf over de berekenbaarheidstheorie heb ik laten zien hoe het mogelijk was de opeenvolgende configuraties die optreden tijdens de berekening van een Turingmachine op te schrijven in een regelmatig rechthoekig schema. De opeenvolgende regels van dit schema ontstaan uit elkaar door vrijwel de gehele regel te copieëren afgezien van de directe omgeving van de leeskop, alwaar een instructie uit het programma van de Turingmachine moet worden verwerkt.

Veronderstel nu dat wij een taal A in NP hebben. Er bestaat nu een Turingmachine M_j die de taal A herkent in een rekentijd begrensd door $k.n^k$ voor geschikt gekozen k, waarbij n de lengte van de invoer voorstelt. Wij mogen over deze Turingmachine nog enkele extra aannamen plegen, zonder dat dit van wezenlijke invloed is op de rekentijd. Zo mogen wij veronderstellen dat de machine nooit in het bandgedeelte links van de invoer komt. De berekening op invoer x speelt zich dus geheel af op het bandgedeelte van lengte $k.|x|^k$ beginnende met de plaats waar de invoer bij het begin van de berekening staat geschreven.

Voorts mogen wij veronderstellen dat de machine, nadat zij heeft besloten de invoer te accepteren de band in nette toestand achterlaat: de leeskop beweegt zich naar het linkeruiteinde van de band, nadat de gehele bandinhoud is uitgeveegd. Het is hierbij nuttig aan het begin van de berekening het linker uiteinde van de band te hebben "gemarkeerd" met een speciaal teken.

Omwille van de correctheid van het vervolg dienen wij ook te eisen dat de machine in een toestand q niet zowel naar links als naar rechts kan bewegen.

De grens $k.n^k$ op de lengte van een berekening is per definitie een bovengrens, en het is voor de navolgende constructie gemakkelijk zeker te weten dat de berekening exact zo lang duurt. Daartoe vervangen wij de accepterende toestand door een toestand waarin een instructie wordt uitgevoerd die niets doet. Volgens de formele definitie stopt de machine nu niet meer, maar het herkennen van de invoer is nu herkenbaar geworden aan het feit dat de machine na $k.n^k$ stappen verkeert in een uniek bepaalde (zichzelf reproducerende) configuratie.

Voor het vervolg blijkt het tevens handig te zijn de toestand in de beschrijving van een configuratie niet langer links van het gelezen symbool te schrijven, maar als het ware "bovenop" het gelezen symbool door het paar (q,s) te beschouwen als één nieuw symbool.

Beschouw nu het tableau dat een accepterende berekening van de machine M_j weergeeft op invoer x. Stel $n = |x|$ en $M = k.n^k$. Op grond van de bovenstaande aannamen weten wij exact hoe dit tableau er uit zou moeten zien. De bovenste rand bestaat uit de, geheel door de invoer x bepaalde beginconfiguratie, terwijl de onderrand bestaat uit de unieke accepterende configuratie. Lengte en breedte van het tableau zijn M. Alleen de invulling van de tussenliggende regels dient nog te worden gevonden, maar het oplossen van deze legpuzzel is precies het aangeven van de accpeterende berekening. Ik gebruik hier de term "legpuzzel" omdat de mogelijke vulling van een regel inderdaad afhankelijk is van de bovenliggende regel (dan wel de regel eronder indien we de puzzel van beneden naar boven willen volleggen).

Het blijkt mogelijk te zijn het volleggen van dit tableau te zien als een legpuzzel probleem. Hiertoe definieren we een aantal puzzelstukken die gebruikt mogen worden om het tableau te vullen. Ieder stukje is een vierkant van lengte één, waarvan de randen zijn gekleurd, waarbij de kleur een symbool of een paar symbolen representeert. Wij kunnen dit aangeven door het vierkant langs de diagonalen in vier stukken te verdelen, en de kleur van de rand te zien als kleur van de aangrenzende driehoek. De stukken mogen niet worden verdraaid - ze moeten in het tableau worden geplaatst in de orientatie waarin ze zijn gegeven. Stukken die elkaar raken langs een rand moeten eenzelfde kleur langs deze rand hebben. Er is een kleur die de afwezigheid van een symbool representeert; deze kleur zullen we "wit" noemen, en noteren door niets te schrijven.

5. DE STELLING VAN COOK.

In dit gedeelte wil ik U een indruk geven van het bewijs van de stelling van Cook die uitspreekt dat er NP-volledige problemen bestaan. In feite toont deze stelling de NP-volledigheid aan van een specifiek probleem dat bekend staat onder de naam SATISFIABILITY, maar de formulering van dit probleem vergt een minimale voorkennis over het vak propositielogica, en ik wil er niet van uitgaan dat mijn huidige gehoor deze voorkennis bezit. Bovendien vereist het volgen van de standaardliteratuur dat, na het bewijs van de NP-volledigheid van dit probleem een keten van reducties wordt beschreven die uiteindelijk leidt tot de NP-volledigheid van het Knapsack probleem. In plaats hiervan wil ik U een schets geven van een rechtstreeks bewijs van de laatsgenoemde NP-volledigheid waarbij slechts gebruik gemaakt wordt van een tussenprobleem dat wij zullen aanduiden met de naam LEGPUZZEL.

In de paragraaf over de berekenbaarheidstheorie heb ik laten zien hoe het mogelijk was de opeenvolgende configuraties die optreden tijdens de berekening van een Turingmachine op te schrijven in een regelmatig rechthoekig schema. De opeenvolgende regels van dit schema ontstaan uit elkaar door vrijwel de gehele regel te copieëren afgezien van de directe omgeving van de leeskop, alwaar een instructie uit het programma van de Turingmachine moet worden verwerkt.

Veronderstel nu dat wij een taal A in NP hebben. Er bestaat nu een Turingmachine M_j die de taal A herkent in een rekentijd begrensd door $k.n^k$ voor geschikt gekozen k, waarbij n de lengte van de invoer voorstelt. Wij mogen over deze Turingmachine nog enkele extra aannamen plegen, zonder dat dit van wezenlijke invloed is op de rekentijd. Zo mogen wij veronderstellen dat de machine nooit in het bandgedeelte links van de invoer komt. De berekening op invoer x speelt zich dus geheel af op het bandgedeelte van lengte $k.|x|^k$ beginnende met de plaats waar de invoer bij het begin van de berekening staat geschreven.

Voorts mogen wij veronderstellen dat de machine, nadat zij heeft besloten de invoer te accepteren de band in nette toestand achterlaat: de leeskop beweegt zich naar het linkeruiteinde van de band, nadat de gehele bandinhoud is uitgeveegd. Het is hierbij nuttig aan het begin van de berekening het linker uiteinde van de band te hebben "gemarkeerd" met een speciaal teken.

Omwille van de correctheid van het vervolg dienen wij ook te eisen dat de machine in een toestand q niet zowel naar links als naar rechts kan bewegen.

De grens $k.n^k$ op de lengte van een berekening is per definitie een bovengrens, en het is voor de navolgende constructie gemakkelijk zeker te weten dat de berekening exact zo lang duurt. Daartoe vervangen wij de accepterende toestand door een toestand waarin een instructie wordt uitgevoerd die niets doet. Volgens de formele definitie stopt de machine nu niet meer, maar het herkennen van de invoer is nu herkenbaar geworden aan het feit dat de machine na $k.n^k$ stappen verkeert in een uniek bepaalde (zichzelf reproducerende) configuratie.

Voor het vervolg blijkt het tevens handig te zijn de toestand in de beschrijving van een configuratie niet langer links van het gelezen symbool te schrijven, maar als het ware "bovenop" het gelezen symbool door het paar (q,s) te beschouwen als één nieuw symbool.

Beschouw nu het tableau dat een accepterende berekening van de machine M_j weergeeft op invoer x. Stel $n = |x|$ en $M = k.n^k$. Op grond van de bovenstaande aannamen weten wij exact hoe dit tableau er uit zou moeten zien. De bovenste rand bestaat uit de, geheel door de invoer x bepaalde beginconfiguratie, terwijl de onderrand bestaat uit de unieke accepterende configuratie. Lengte en breedte van het tableau zijn M. Alleen de invulling van de tussenliggende regels dient nog te worden gevonden, maar het oplossen van deze legpuzzel is precies het aangeven van de accpeterende berekening. Ik gebruik hier de term "legpuzzel" omdat de mogelijke vulling van een regel inderdaad afhankelijk is van de bovenliggende regel (dan wel de regel eronder indien we de puzzel van beneden naar boven willen volleggen).

Het blijkt mogelijk te zijn het volleggen van dit tableau te zien als een legpuzzel probleem. Hiertoe definieren we een aantal puzzelstukken die gebruikt mogen worden om het tableau te vullen. Ieder stukje is een vierkant van lengte één, waarvan de randen zijn gekleurd, waarbij de kleur een symbool of een paar symbolen representeert. Wij kunnen dit aangeven door het vierkant langs de diagonalen in vier stukken te verdelen, en de kleur van de rand te zien als kleur van de aangrenzende driehoek. De stukken mogen niet worden verdraaid - ze moeten in het tableau worden geplaatst in de orientatie waarin ze zijn gegeven. Stukken die elkaar raken langs een rand moeten eenzelfde kleur langs deze rand hebben. Er is een kleur die de afwezigheid van een symbool representeert; deze kleur zullen we "wit" noemen, en noteren door niets te schrijven.

Na deze uiteenzettingen kan ik de omschrijving geven van het probleem:

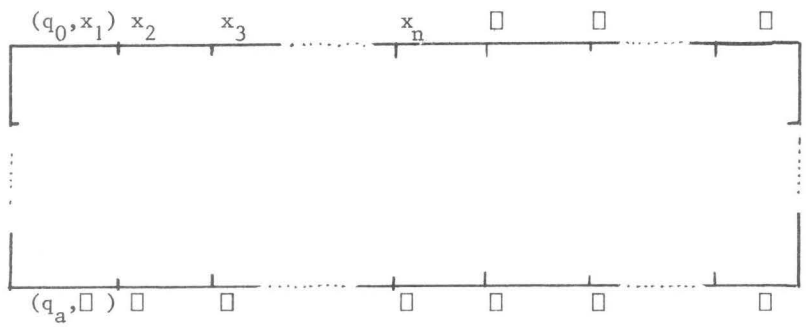
LEGPUZZEL.

INSTANTIE: een lijst van puzzelstukkjes (georienteerde vierkanten met gekleurde randen), en een M bij M vierkant, waarvan de randen reeds van een kleuring zijn voorzien.

VRAAG: Is het mogelijk het gegeven vierkant geheel te vullen met copieen van de gegeven puzzelstukjes, zó dat de kleuren aan de rand van het grote vierkant passen ?

Dat dit een probleem in NP is is vanzelfsprekend; een nondeterministische machine hoeft niets anders te doen dan naar willekeur het grote vierkant vol te leggen met stukjes en achteraf te controleren dat hij het op correcte wijze heeft gevuld. NP-volledigheid van dit probleem kunnen we aantonen door in aansluiting op ons eerdere betoog voor de genoemde Turingmachine M_j een lijst van puzzelstukjes te definieren, en voor de gegeven invoer x een M bij M vierkant met gekleurde rand, zodanig dat een oplossing van de legpuzzel zich laat lezen als een accepterende berekening, en omgekeerd, iedere accepterende berekening een oplossing van de legpuzzel geeft.

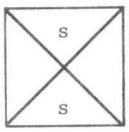
Allereerst de rand van het M bij M vierkant. Deze kleuren wij als aangegeven in het onderstaande diagram:



linker en rechter rand van het vierkant zijn wit gekleurd. Boven zien we de unieke beginconfiguratie op invoer $x = x_1x_2...x_n$, terwijl de onderrand de unieke accepterende eindconfiguratie aangeeft.

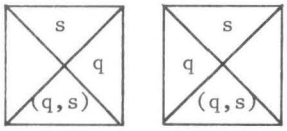
De puzzelstukjes hebben de functie de kleur van hun bovenrand ongewijzigd door te geven , dan wel, in het geval van aanwezigheid van de leeskop een rekenstap te simuleren, of om met behoud van kleur, de leeskop "op te vangen" .

Een stukje dat het bandsymbool s als kleur van de bovenrand ongewijzigd doorgeeft naar de onderrand ziet er alsvolgt uit:

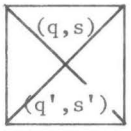


Voor ieder symbool s in het bandalphabet van de Turingmachine is een dergelijk stukje aanwezig.

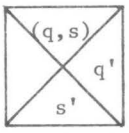
Het stukje dat aangeeft hoe een bandveld dat beschreven is met symbool s de leeskop opvangt in toestand q komt in twee vormen voor afhankelijk of de leeskop van links of van rechts aan komt zetten. Deze stukjes hebben de volgende vorm (slechts één van beide zal optreden !):



De overige stukjes die rekenstappen beschrijven zijn gekoppeld aan instructies in het programma. Voor iedere instructie (q,s,q',s',∅) vormen wij het stukje:



Instructies waarbij de kop beweegt dienen zó gevormd te zijn dat zij afdwingen dat ernaast een stukje gelegt wordt dat de leeskop opvangt. Voor iedere instructie (q,s,q',s',R) vormen we het stukje



en een analoog stukje wordt geformeerd voor instructies waarbij de kop naar links beweegt.

Hiermee is de beschrijving van de puzzelstukjes voltooid.

Samenvattend luidt de argumentatie, op grond waarvan we LEGPUZZEL NP-volledig hebben verklaard als volgt: Stel A in NP; dan wordt A geaccepteerd door M_j in tijd $k.n^k$ hetgeen wil zeggen: x behoort tot A desda. M_j accepteert x in tijd M desda. er bestaat een tableau van afmetingen M bij M voorstellende een accepterende berekening van M_j desda. de geconstrueerde instantie van het probleem LEGPUZZEL is oplosbaar.

Wellicht zult U opmerken dat het werken met legpuzzels met gekleurde randen niet aansluit op het type puzzels dat U in de winkel pleegt aan te schaffen. Het is echter eenvoudig om de puzzels van het besproken type te herleiden tot puzzels met kromme randen die in elkaar moeten passen. Voor iedere kleur ontwerpe men een fraaie krul, waarbij een optreden van de kleur aan de linkerrand (resp. rechterrand) wordt gecodeerd door deze krul als uitstulping resp. gat in de rand te zagen. Op deze wijze passen alleen die stukken in elkaar die afkomstig zijn van overeenkomstige kleuren. De eis dat de stukjes niet mogen worden gedraaid laat zich opvangen door ervooor te zorgen dat de krullen voor boven- resp. onderkanten niet passen in die voor linker- resp. rechterkanten, weshalve het roteren van stukjes zinloos is. De vulling van de rand kan worden afgedwongen door de randstukjes zo uit te zagen dat de rand zich slechts op één manier laat neerleggen, waarbij het binnengebied een krullenpatroon vertoont dat overeenstemt met de gegeven beginkleuring van de rand die we hebben gebruikt bij de reductie. Aldus kunnen wij ons artificiele probleem LEGPUZZEL reduceren tot de welbekende Jig Saw puzzels, en weet U meteen waar U mee begint als U een dergelijke puzzel in huis haalt.

Voor de ontwikkeling van onze theorie is het veel belangrijker in te zien hoe het probleem LEGPUZZEL zich laat reduceren tot een probleem als KNAPSACK. Hiertoe gaan wij als het ware de gegeven puzzel digitaliseren: de puzzelstukjes worden gecodeerd in de vorm van getallen, en de eis dat met de gegeven stukjes een gegeven vierkant wordt gevuld wordt hierbij vertaald in de eis om met gebruik van de gegeven getaalen een streefbedrag exact vol te maken.

Om deze digitalisering te beschrijven doe ik een beroep op uw inzicht in het rekenen in een getalstelsel met een van 10 of 2 verschillend grondtal. In het getalstelsel met *grondtal* (*radix)* g kent men symbolen (*cijfers*) $0,1,2,3,\ldots,g-1$. Een tekenrij $d_m d_{m-1} \ldots d_0$ stelt het getal $\sum_{i=0}^{m} d_i . g^i$ voor. Getallen worden opgeteld door de cijferrijtjes op de juiste wijze onder elkaar te schrijven, en cijfergewijs de som te vormen. Als hierbij een som optreedt die groter is dan g treedt een zogenaamde *carry* op: de cijfersom wordt met g verminderd, terwijl in de kolom links ervan de cijfersom met één wordt vermeerderd.

Als U nu een lijst met m getallen heeft, waarbij het grootste optredende cijfer d is , en het grondtal g voldoet aan $m.d < g$ dan ziet U gemakkelijk in dat bij iedere optelling die onstaat door een deelverzameling van deze getallen op te tellen, er nooit een carry kan optreden; per kolom kunt U de som vormen en deze kolomsommen als cijfers naast elkaar geschreven vormen het resultaat. Omgekeerd, als U probeert een gegeven streefbedrag te schrijven als som van een deelverzameling dient U een lijst van getallen samen te stellen waarvoor op iedere plaats de cijfersom het gevraagde cijfer oplevert, want bij gebrek aan carries vind er geen beinvloeding van de resultaten over en weer plaats. Wij zullen dit verschijnsel aanwenden om een grote waslijst van condities samen te bundelen tot één Knapsack conditie.

Ik beschrijf nu de reductie van het probleem LEGPUZZEL tot het Knapsack probleem. Zij gegeven een instantie van LEGPUZZEL. De optredende kleuren noemen we $k_1, k_2, k_3, \ldots, k_s$. Het te beleggen veld heeft omvang M bij M . We nemen aan dat de kleuren $k_1, \ldots, k_t$ alleen maar voorkomen langs horizontale randen, terwijl de kleuren $k_{t+1}, \ldots, k_s$ alleen maar langs verticale randen optreden. Merk op dat de in hetvoorafgaande geconstrueerde instanties deze eigenschap bezitten.

Laat de gegeven collectie puzzelstukjes bestaan uit K exemplaren. Kies een grondtal $g > K.M^2+1$. Zij $M' = M+1$ en $L = M.M'.s$. De getallen die we gaan vormen zullen in het g-tallig stelsel worden beschreven door een rij cijfers die ieder op zich de waarde 0 of 1 aangeven, terwijl het aantal cijfers gelijk L is (waarbij mogelijkerwijze niet-significante nullen aan het begin kunnen optreden) . Een getal x_0 gebruiken we om de gegeven rand van de legpuzzel te beschrijven, terwijl de resterende $K.M^2$ getallen worden gebruikt om de mogelijkheid te coderen een van de gegeven K stukjes op een van de te vullen M^2 plaatsen neer te leggen. Het streefgetal N' kiezen we gelijk aan $\sum_{i=0}^{L-1} g^i$, dat wil zeggen het getal dat zich in het g-tallige systeem laat opschrijven als een rij van L optredens van het cijfer "1" .

Merk op dat uit de tot hiertoe gegeven informatie reeds volgt dat een oplossing van de Knapsack instantie die we aan het maken zijn klaarblijkelijk moet bestaan uit een lijst van getallen met de eigenschap dat er voor iedere positie $j = 0,1,\ldots,L-1$ er exact één getal in de lijst van gekozen getallen moet zijn dat op die positie een "1" heeft, terwijl alle andere getallen op die plaats een "0" hebben.

Omdat we te maken hebben met een vierkant van afmetingen M bij M dat gevuld moet worden treden in het probleem M' horizontale en M' verticale lijnen op waarop de puzzelstukjes hun randen hebben liggen. In het bijzonder zal een puzzelstuk gelegen op de positie met rijnummer i en kolomnummer j voorzien zijn van een linker(rechter) rand in verticale lijn j resp. j+1 en een boven(beneden) rand op horizontale lijn i resp. i+1 .

Beschouw nu een getal, bestaande uit nullen en eenen van lengte L. wij denken deze rij cijfers verdeeld in blokken ter lengte s . De eerste t cijfers in het blok van s cijfers beginnende op plaats ((i-1).M+j-1).s hangen samen met het optreden van kleuren langs de doorsnede van de horizontale lijn met nummer i en de verticale strook vierkantjes met nummer j ; de overige cijferposities worden gebruikt voor het optreden van kleuren langs de verticale lijn met nummer i in de horizontale strook met nummer j . Een optreden van een kleur in een linker- resp. bovenrand van een vierkantje wordt aangegeven door in het corresponderende "venster" in het getal een "0" te plaatsen terwijl de overige kleuren een "1" krijgen toebedeeld. Voor rechter- resp. onderranden geldt het omgekeerde: de aanwezige kleur krijgt een "1" toegewezen en juist de afwezige kleuren krijgen een "0" . Op deze wijze volgt na ampele overweging dat de enige methode om bij het optellen van een aantal van de op deze wijze geconstrueerde getallen binnen een venster een resultaat te krijgen dat bestaat uit een blok eenen eruit bestaat een gelijkgekleurd optreden van dit venster als linker- en rechterrand (resp. onder- en bovenrand) op te nemen, en vervolgens nooit meer een getal op te nemen dat iets met dit venster te maken heeft. Op deze wijze dwingt het voldoen aan de Knapsack conditie af dat we een oplossing van de puzzel vormen.

Na deze informele uitleg van de constructie resteert het cijferwerk om de beoogde getallen exact te definieren. We definieren de volgende hulpgetallen:

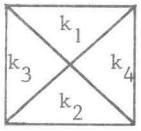
$a_j = g^j$, $j= 0,1,\ldots,t-1$; $b_j = g^j$, $j=t,\ldots,s-1$.

$\bar{a}_j = 1+g+g^2+ \ldots + g^{t-1} - g^j$, $j= 0,1,\ldots,t-1$.

$\bar{b}_j = g^t+g^{t+1}+ \ldots + g^{s-1} - g^j$, $j=t,\ldots,s-1$.

$G = g^s$.

Het getal x_h met $h = ((i-1).M + j).K + k$ geeft de codering aan van de bezetting met kleuren die optreedt als we het vierkantje met nummer k plaatsen op het veld met positie rijnummer i, kolomnummer j. Laat het vierkantje met nummer k er als volgt uitzien:



Dan krijgt x_h de waarde:

$$x_h = G^{((i-1).M+j-1)}.\bar{a}_{k_1} + G^{((j-1).M+i-1)}.\bar{b}_{k_3} + G^{(i.M+j-1)}.a_{k_2} + G^{(j.M+i-1)}.b_{k_4}$$

Merk op dat de machten van G de "bitpatronen" van de a_j, $\bar{a}_j$ etc. verschuiven tot in de correcte vensters.

Rest te definieren wat het getal x_0 is dat met de rand correspondeert. Dit moet uiteraard de volledige kleuring van de rand beschrijven: Laat de kleur van veld j van de linker- (rechter-,boven- resp. onderrand) gelijk zijn aan $l(j)$ ($r(j)$,$b(j)$, resp $o(j)$). dan definieren we x_0 door

$$x_0 = \sum_{j=1}^{M} (G^j.a_{b(j)} + G^j.b_{l(j)} + G^{M^2+j}.\bar{a}_{o(j)} + G^{M^2+j}.\bar{b}_{r(j)}) .$$

Hiermee is de beschrijving van de reductie van LEGPUZZEL tot KNAPSACK in principe voltooid. Ook voor deze reductie geldt dat er nog veel werk te verzetten zal zijn alvorens wij haar kunnen laten uitvoeren door een in polynomiaal rekenende Turingmachine, maar dit laatste werk zal wel nooit worden verzet.

Ik wil nog wel even wijzen op een ander combinatorisch probleem dat we in feite onderweg zijn tegen gekomen. De geconstrueerde Knapsack instantie vraagt een getal dat in het g-tallig stelsel geschreven wordt als een rij eenen te schrijven als een som van getallen die allen worden geschreven als rijen bestaande uit nullen en eenen, waarbij geen gebruik gemaakt wordt van carries. In feite vragen wij hier een verzameling te schrijven als vereniging van deelverzamelingen uit een gegeven lijst van deelverzamelingen, zonder dat een element meer dan eens mag optreden. Dit laatste probleem is een bekend probleem uit de combinatorische optimalisering dat bekend staat onder de naam EXACT COVER ; nodeloos te vermelden dat het NP-volledig is, want wij hebben dit in feite in het voorafgaande bewijs aangetoond.

6. EEN NP-VOLLEDIG CRYPTOSYSTEEM.

Ik wil thans een voorbeeld bespreken van een cryptosysteem waarvan bewezen kan worden dat het *moeilijk* is in de technische zin waar het in deze voordracht over gaat. Dit wil geenszins impliceren dat we hiermee een bruikbaar systeem beschrijven, want zoals de ontwerpers van het systeem, S. Even & Y. Yacobi reeds hebben opgemerkt is het systeem in de practijk redelijk makkelijk te kraken. Dit feit laat tevens op navrante wijze zien dat het begrip NP-volledigheid nog lang niet die graad van moeilijkheid beschrijft waar we in de cryptografie behoefte aan hebben.

We zullen bij deze behandeling overigens te maken krijgen met een afwijkend begrip van een reductie. Bij de tot nu toe besproken voorbeelden van reducties gingen we uit van een instantie van probleem A en vertaalden die in een enkele instantie van het probleem B . We kunnen ons echter voorstellen dat instanties van A worden opgelost door meerdere instanties van B te produceren, en deze achereenvolgens op te lossen, waarbij het is toegestaan dat de loop van de berekening beinvloed wordt door het resultaat van een aanroep van de "subroutine" waarmee de instanties van probleem B worden opgelost. Zolang dit proces, de tijd benodigd voor het oplossen van de instanties van probleem B niet meegerekend, in polynomiaal begrensde tijd verloopt, stelt een dergelijke reductie ons in staat de cruciale conclusie: "als B tot de klasse P behoort dan behoort ook A tot P" te trekken.

In het verdere vervolg van deze paragraaf zullen wij een cryptografisch systeem beschrijven, en het codekraak probleem voor dit systeem definieren. Vervolgens zullen we laten zien hoe het mogelijk is een gegeven instantie van KNAPSACK op te lossen door een rij instanties van dit codekraak probleem op te lossen.

Het bedoelde systeem is een implementatie van een pseudo one-time scratch pad systeem. Boodschappen M_t van lengte m worden versleuteld door er een "willekeurige" sleutel K_t bij op te tellen in de bitsgewijze optelling mod. 2:

$$C_t = M_t \oplus K_t \quad \text{, weshalve} \quad M_t = C_t \oplus K_t \quad .$$

De sleutel K_t is tijdsafhankelijk, en dient bekend te zijn zowel aan de zender als aan de ontvanger. Om K_t te bepalen wordt gebruik gemaakt van een rij "random" woorden R_t, die in onversluierde vorm met de boodschappen worden meeverzonden, en waarvan men mag veronderstellen dat de vijand deze eveneens kent, en een geheime sleutel X die alleen bekend is aan verzender en ontvanger van het systeem. Het proces waarmee de sleutel K_t wordt aangemaakt laat zich alsvolgt beschrijven:

Er is een vaste lijst gegeven van n getallen $a_1,\ldots,a_n$. de geheime sleutel X is een bitvector van lengte n, en ook de random informatie R_t is een tijdsafhankelijke bitstring van lengte n. Men vormt de bitstring $G_t = g_1 g_2 \ldots g_n$ van lengte n via $G_t = X \oplus R_t$, en tenslotte kiest men $K_t = g_1 . a_1 + g_2 . a_2 + \ldots + g_n . a_n$. Dit getal K_t vat men weer op als een bitstring van lengte m, waarbij het getal m voldoet aan de gelijkheid $m = \lceil {}^2\log(\sum_{i=1}^{m} a_i + 1) \rceil$; $\lceil x \rceil$ geeft hierbij aan het kleinste gehele getal groter of gelijk x.

De vijand mag worden veronderstel te beschikken over de kennis van de getallen a_i en daarmee tevens over de lengtes n en m. Verder weet hij hoe het systeem werkt. Voor het ontcijferen is het gelijkwaardig om de boodschap M_t of de sleutel K_t te ontrafelen, want het cryptogram C_t is bekende informatie. Kennis van de sleutel X compromiteert het systeem natuurlijk direct, maar het is denkbaar dat het systeem wordt gekraakt zonder dat X bekend is. Er is zelfs geen garantie dat uit een gebruik van het systeem een unieke waarde van X valt te reconstrueren.

Wij komen derhalve tot de volgende formulering van het codekraak probleem:

CODEKRAAK.

INSTANTIE: Een gebruiksgeschiedenis, bestaande uit k paren (K_1,R_1), $\ldots,(K_k,R_k)$; een nieuwe random string R, en de lijst $a_1,\ldots,a_n$.

VRAAG: Geef een sleutel K zó dat het paar (K,R) gekoppeld aan deze gebruiksgeschiedenis een *consistente* geschiedenis oplevert, gesteld dat zo een geschiedenis bestaat.

Consistentie wil hier zeggen: er bestaat een geheime sleutel X zó dat voor $i = 1,\ldots,k$ de sleutel K_t op de beschreven wijze bepaald is door R_i en X, terwijl deze zelfde sleutel X met de string R de sleutel K bepaalt.

Uit de formulering blijkt al direct dat we hier niet te maken hebben met een beslissingsprobleem , en alleen daardoor valt dit probleem buiten het patroon dat in de voorafgaande twee paragrafen is gevolgd; ook dit is een reden om gebruik te maken van een meer algemeen reductiebegrip.

We zullen nu laten zien hoe het mogelijk is instanties van het Knapsack probleem op te lossen als wij kunnen beschikken over een hypothetische machine die voor ons instanties van het probleem CODEKRAAK in één stap oplost.

Zij gegeven een instantie van KNAPSACK met de lijst getallen $a_1, a_2, \ldots, a_n$ en streefbedrag N' . Wij beschouwen een implementatie van het genoemde cryptosysteem op basis van deze zelfde lijst getallen $a_1, a_2, \ldots, a_n$, waarmee tevens de bloklengte m is vastgelegd.

De oplosbaarheid van de gegeven instantie van KNAPSACK zullen wij vaststellen door te proberen een oplossing $x_1, x_2, \ldots, x_n$ van $\sum_{i=1}^{n} x_i . a_i = N'$ met $x_i = 0$ of 1 te fabriceren.

Beschouw het paar $(N', 0)$ als element van een gebruiksgeschiedenis, waarbij 0 hier staat voor de "random" bitstring bestaande uit n nullen, em N' staat voor het getal N' uitgedrukt als een bitstring van lengte m . Omdat $X \oplus 0 = X$ kan dit paar alleen maar in een consistente geschiedenis optreden als $X = x_1 x_2 \ldots x_n$ met $x_1 . a_1 + \ldots + x_n . a_n = N'$. Stel nu dat we een geschiedenis, bestaande uit dit ene paar $(N', 0)$ aanbieden tezamen met de random string $R = 10000\ldots0$. Vervolgens kijken wij naar het resultaat dat onze hypothetische machine voor het probleem CODEKRAAK ons oplevert. Dit resultaat zal bestaan uit een sleutel K met de eigenschap dat $(N', 0)$, $(K, 10\ldots0)$ een consistente geschiedenis is, gesteld dat het paar $(N', 0)$ zelf mogelijk is.

Er kunnen zich nu twee gevallen voordoen. Het is denkbaar dat de gegeven instantie van KNAPSACK onoplosbaar is, hetgeen zou impliceren dat het paar $(N', 0)$ nooit element van welke consistente geschiedenis dan ook kan zijn. In dit geval is de opgave gesteld aan onze machine voor CODEKRAAK onoplosbaar, en moeten wij niet verbaasd zijn als de machine onzin produceert. In het geval dat de gegeven KNAPSACK instantie echter wel oplosbaar is weten we dat er passende sleutels X kunnen bestaan, en is het derhalve mogelijk een sleutel K te produceren die past in een consistente geschiedenis. Merk nu op dat de waarde van K gelijk zou moeten zijn aan $K = (1-x_1).a_1 + x_2.a_2 + \ldots + x_n.a_n$; uit

het feit of K al dan niet groter dan N' is kunnen wij hierdoor bepalen of voor de gebruikte oplossing X van de KNAPSACK instantie geldt $x_1 = 0$ of $x_1 = 1$. Als de instantie oplosbaar was hebben wij hiermee de eerste component x_1 van een mogelijke oplossing bepaald, en tevens een nieuwe instantie van KNAPSACK bepaald, bestaande uit de getallen $a_2,\ldots,a_n$ en de streefwaarde $N'-x_1.a_1$ die opnieuw oplosbaar is. Gebruikmakende van dezelfde techniek kunnen we nu de component x_2 van een oplossing achterhalen via een aanroep van onze hypothetische CODEKRAAK machine, etc..... .

Merk op dat achteraf kan worden gecontroleerd of de geproduceerde oplossing $x_1,\ldots,x_n$ correct is. Zo ja, dan zijn wij tevreden, en zo nee, dan weten wij dat de KNAPSACK instantie kennelijk niet oplosbaar was, want anders hadden we een correcte oplossing gevonden.

Ik hoop U te hebben overtuigd dat met deze beschouwing de formele moeilijkheid van het probleem CODEKRAAK is aangetoond. Helaas leert de practijk ons dat vele instanties van het probleem zich gemakkelijk laten oplossen. Bij ons bewijs gebruiken wij die instanties waarbij de geschiedenis extreem kort is : één voorafgaand woord van lengte m dat door het systeem is gecodeerd. (met lege voorgeschiedenis is het probleem triviaal, want iedere sleutel X is dan consistent met het voorafgaande). Naar mate een groter gebruik van het systeem beschikbaar is kan de vijand meer informatie over de mogelijke sleutels X achterhalen. Hierbij wreekt zich het feit dat het systeem gebruik maakt van lineiare berekeningsmethoden, en lineaire methoden geven zelden aanleiding tot een veilig systeem.

Bij nadere analyse blijkt de bepaling van de sleutel X te herleiden te zijn tot het oplossen van een stelsel van lineaire vergelijkingen in de n onbekenden $x_1,x_2,\ldots,x_n$, waarbij het aantal vergelijkingen gelijk is aan de lengte van de voorgeschiedenis. Beschouw hiertoe opnieuw de definitie van de waarde van K_t : er goldt $K_t = \sum_{i=1}^{n} g_i.a_i$ waarbij $g_1\ldots g_n = X \oplus R_t$. Stel $X = x_1x_2\ldots x_n$ en $R_t = r_1r_2\ldots r_n$ (waarbij de r_i uiteraard van t afhangen). Dan geldt $g_i = x_i + r_i-2.x_i.r_i$. Dit betekent dat geldt:

$$K_t = \sum_{i=1}^{n}(x_i + r_i-2.x_i.r_i).a_i \quad \text{oftewel} \quad \sum_{i=1}^{n}(1-2r_i).a_i.x_i = K_t - \sum_{i=1}^{n} r_i.a_i$$

Ieder paar in de voorgeschiedenis levert derhalve een lineaire vergelijking in de x_i op. Indien onder deze vergelijkingen er n zijn die lineair onafhankelijk zijn is het systeem derhalve makkelijk te

kraken, en de kans dat onder k willekeurige vergelijkingen er n zijn die lineair onafhankelijk zijn neemt snel toe tot 1 naarmate k-n groter wordt.

In het vervolg van hun artikel laten de auteurs zien hoe een alternatieve methode om de sleutels K_t te destileren uit X en R_t die, om uit te rekenen, net zo goedkoop is, maar die niet berust op lineaire methodieken, leidt tot een cryptosysteem waarvan geen kraakmethode bekend is aan de auteurs; helaas kunnen zij van dit systeem al evenmin aantonen dat het "veilig" is, hoewel ook van dit systeem wordt bewezen dat het moeilijk is in de zin van de NP-volledigheidstheorie.

7. RELEVANTIE.

Aan het einde van deze beschouwingen doen wij er goed aan terug te keren tot ons uitgangspunt, de kennelijk tegenstrijdige eisen van *gemakkelijkheid* en *moeilijkheid* die een cryptosysteem ons oplegt, en ons afvragen wat de hierboven ontwikkelde NP-volledigheids theorie ons met betrekking tot deze vraagstelling heeft te vertellen.

Onze eerste conclusie lijkt vrij negatief. De conditie dat een cryptosysteem hanteerbaar moet zijn voor de legitieme gebruikers voerde ons tot herkenningsproblemen in NP, terwijl de eis dat de vijand het systeem niet zou kunnen kraken leerde dat deze problemen liever niet tot de klasse P dienden te behoren. Aangezien de vraag P = NP ? totop heden open is, en er de schijn van heeft voorlopig onopgelost te blijven, levert de theorie ons dus het onbevredigende antwoord dat zij niet bij machte is ons het bestaan van cryptosystemen te garanderen.

Deze laatste conclusie zou zelfs nog in sterkere mate geformuleerd kunnen worden. Zelfs indien $P \neq NP$ (hetgeen in het algemeen als gangbare werkhypothese wordt aanvaard) dan nog hebben we kunnen zien dat de graad van moeilijkheid in NP-problemen begrensd is - er zijn geen problemen in NP die wezenlijk moeilijker zijn dan een NP-volledig probleem. In de practijk blijkt een grote klasse van NP-problemen zeer wel aan te vatten te zijn. Hierbij spelen verschillende keuzen die wij hebben gemaakt bij de formulering van onze theorie een rol.

Allereers hebben wij voortdurend te maken gehad met zogeheten *worst case* complexiteit; de moeilijkheid van een probleem wordt gemeten aan de graad van moeilijkheid van zijn lastigste instanties.

In het algemeen is het aantal woorden in een taal ter lengte n een aantal dat exponentieel toeneemt met de lengte n. Zelfs indien van dit enorme aantal slechts een verwaarloosbare fractie woorden bestaat die voor een herkenningsmethode lastig te herkennen zijn, blijft het probleem moeilijk. Het blijkt in de practijk ook dikwijls zo te zijn dat voor diverse NP-volledige problemen een groot deel van de instanties efficient kunnen worden opgelost.

Een ander aspect dat hierbij een rol speelt is het gebruik van *benaderings methoden*. Bij het oplossen van een combinatorisch optimaliseringsprobleem komt het in de practijk zeer vaak voor dat men kan bewijzen dat het bepalen van de optimale oplossing NP-volledig is, terwijl daarnaast polynomiale algorithmen bestaan die redelijke tot goede benaderingen van de oplossing geven. Hierbij blijken de verschillende problemen zich afwijkend te gedragen. Van een probleem als het Knapsack probleem is bekend dat het zich in polynomiale tijd willekeurig goed laat benaderen; de exacte formulering hiervan is nogal gecompliceerd, en ik verwijs de lezer hiervoor naar de literatuur. Van andere problemen is echter bekend dat er een "mysterieuze" drempelwaarde bestaat tot waar toe polynomiale benaderingen mogelijk zijn - ook het vereenvoudigde probleem om een benadering te geven die qua kwaliteit deze drempelwaarde overschreidt in gunstige zin is bewijsbaar NP-volledig.

Een ander verschil dat tussen diverse NP-volledige problemen kan optreden betreft de rol van de binaire of decimale schrijfwijze voor de erin optredende getallen. Ik merkte reeds eerder op dat voor het Knapsack probleem een algorithme bestaat die dit probleem oplost in een tijd begrensd door een polynoom in termen van de waarde van de erin optredende getallen. KNAPSACK is derhalve lastig omdat we in staat zijn in weinig ruimte wezenloos grote getallen op te schrijven. Er zijn andere NP-volledige problemen waarbij dit aspect echter geen rol speelt, hetzij omdat er geen getallen in de formulering van het probleem optreden (zoals bij het probleem LEGPUZZEL), dan wel dat het probleem NP-volledig blijft zelfs indien de getallen in het unaire stelsel worden gerepresenteerd. Een voorbeeld van een probleem met deze complexiteitsstatus is het hier niet besproken probleem 3-PARTITION , waarin gevraagd wordt een collectie van $3m$ getallen op te splitsen in m drietallen die allen dezelfde som hebben.

Ieder van de bovengenoemde aspecten kan veroorzaken dat een voorgesteld cryptosysteem tegelijkertijd NP-volledig, en in de practijk makkelijk oplosbaar kan zijn.

We moeten hierbij tevens rekening houden met het feit dat we gebruik hebben gemaakt van een wel zeer grof hulpmiddel bij het verdelen van de wereld in makkelijke en moeilijke problemen. Het middel van de polynomiaal begrensde reductie stelt ons in staat problemen die zich oppervlakkig gezien totaal verschillend gedragen toch tot elkaar te vertalen. Dit leidt tot de bevreemdende situatie dat alle totop heden bekende NP-volledige problemen in elkaar kunnen worden vertaald, waarbij bovendien kan worden verondersteld dat de vertalende reductie een bijectieve afbeelding is van S^* naar S^*. Alle bekende NP-volledige problemen blijken dus *P-isomorf* te zijn.

De theorie leert daarnaast dat er, in het geval dat $P \neq NP$ problemen in NP moeten bestaan die niet in P zitten maar ook niet NP-volledig zijn. De enige manier om dit soort problemen te maken levert als product een volkomen kunstmatig diagonalisatie gedrocht op waarvan men direct kan inzien dat dit geen practisch zinvol probleem is. Er bestaan wel diverse problemen in NP waarvan totop heden de complexiteitsstatus open is. Het bekendste daarvan is het probleem om vast te stellen of een getal al dan niet een priemgetal is. Bekend is een polynomiale non-deterministische algorithme die met kans $\frac{3}{4}$ een deelbaar getal als zodanig ontmaskert (zonder echter hierbij een echte factor aan te wijzen); deze algorithme berust op een test die vergelijkbaar is met een criterium voortkomende uit de kleine stelling van Fermat: $a^{p-1} \equiv 1 \mod. p$ voor p priem en a onderling ondeelbaar met p. Een dergelijke test kan worden herhaald, waardoor in principe in polynomiaal begrensde tijd kan worden vastgesteld of een getal priem is, waarbij de kans dat de uitslag van deze test incorrect is willekeurig klein kan worden gemaakt. Een dergelijke methode moge dan wiskundig gezien zijn bezwaren hebben, maar practisch bruikbaar is zij wel, zeker in een context als de cryptografie. Men kan ook een variant van deze methode aangeven die geheel deterministisch is en die in polynomiale tijd loopt, maar hiervan weten we weer niet of de methode correct is: het bewijs maakt gebruik van de zogenaamde gegeneraliseerde Riemann hypothese, en dit is een onbewezen vermoeden uit de wiskunde. De beste bekende wiskundig correcte methode om priemgetallen te testen vraagt tijd in de orde van $n^{\log\log(n)}$.

Een andere klasse van problemen in NP waarvan de status open ligt zijn de problemen die samenhangen met de vraag vast te stellen of twee gegeven grafen al dan niet isomorf zijn; ik voorzie hier vooralsnog geen zinvol verband met de cryptografie.

De technieken ontwikkeld om problemen NP-volledig te bewijzen zijn overigens ook toegepast om problemen te analyseren die nog lastiger zijn. Denk terug aan het besproken probleem LEGPUZZEL . Dit probleem behoorde tot de klasse NP omdat de lengte van het te vullen vierkant werd begrensd door de lengte van de formulering van een instantie. Bij het bewijs dat dit probleem NP-volledig was hebben we echter gebruik gemaakt van instanties met een erg regelmatige structuur : onder en bovenrand vrijwel geheel gevuld met blanco symbolen, en linker en rechter zijde wit. Men kan zich een notatiesysteem voorstellen waarbij herhaalde symbolen langs de rand worden aangegeven door het symbool te noemen, en, in decimale notatie, het aantal keren dat dit symbool langs de rand optreedt. Dit maakt het mogelijk een vierkant van zijde M te specificeren in ruimte van de orde log(M) . Gebruik van dezelfde bewijsmethode als toegepast in paragraaf 5 levert ons dan een voorbeeld van een probleem dat volledig is voor de klasse NEXPTIME . Beschouwen we een conventie waarbij boven en benedenrand exact worden gespecificeerd, terwijl van de zijranden slechts wordt vereist dat zij wit gekleurd zijn, waarbij de lengte in het midden wordt gelaten, dan levert onze methode een probleem dat volledig is voor PSPACE . Uitgaande van dit soort problemen kan men van practische problemen uit de wiskunde of uit andere gebieden aantonen dat zij volledig zijn voor PSPACE of NEXPTIME . Zo is bijvoorbeeld bekend van een groot aantal bordspelen zoals Hex of Go-Moku dat een geschikt gekozen wiskundige generalisatie een spel oplevert waarvan de eindspelanalyse PSPACE volledig is. Een spel als schaken blijkt in dit opzicht nog een extra lastig te zijn: de generalisatie hiervan blijkt volledig te zijn voor EXPTIME .

Men kan zich echter afvragen of deze klassen van problemen die nog lastiger zijn dan NP nog wel relevant zijn voor de cryptografie, aangezien het weinig zin heeft een cryptosysteem te ontwerpen dat de vijand niet kan kraken maar dat ook voor de legitieme gebruiker tot onhanteerbare rekentijden aanleiding geeft.

8. LITERATUUR.

Een standaardverwijzing voor de theorie over NP-volledigheid is de in 1979 uitgekomen monografie: *Computers and Intractability, A Guide to the theory of NP-completeness*, van de hand van de auteurs

Michael R. Garey & David S. Johnson, uitgave Freeman, San Francisco. Het werk bevat een uitgebreide lijst van enkele honderden bekende NP-volledige problemen. De ontwikkelingen op dit gebied zijn echter zo snel dat reeds na twee jaar blijkt dat het boek aan een herziening toe is. Van de opgenomen open problemen blijken in twee jaar tijd de helft te zijn opgelost. De auteurs hebben echter besloten de aanpassing aan de nieuwe ontwikkelingen niet de vorm te geven van een tweede editie (die ongetwijfeld een zelfde lot beschoren zal zijn) maar via een driemaandelijkse kroniek van recente ontwikkelingen die is opgenomen in het tijdschrift Journal of Algorithms, waarvan inmiddels drie afleveringen zijn verschenen.

Een klassieke inleiding tot de theorie der berekenbaarheid is het handboek: *Introduction to Metamathematics*, van de hand van S.C. Kleene, uitgegeven bij North Holland Publ. Cie in 1952. Een modernere inleiding tot de automatentheorie, en de theorie der formele talen, waarin tevens is opgenomen een inleiding tot de complexiteitstheorie is het handboek: *Introduction to Automata theory, Languages and Computation,* geschreven door J.E. Hopcroft & J.D. Ullman, uitgegeven in 1979 bij Addison Wesley, Reading Mass.

Het in paragraaf 6 besproken NP-volledige cryptosysteem is beschreven in: *Cryptocomplexity and NP-completeness* by S. Even and Y. Yacobi; Dit artikel is opgenomen in de proceedings van het 7e colloquium over Automaten Talen en Programmering (ICALP 7), gehouden in Jul 1980 te Noordwijkerhout. Deze proceedings zijn uitgegeven onder redactie van J.W. de Bakker & J. van Leeuwen en opgenomen in de reeks Springer Lecture Notes in Computer Science als aflevering 85 .

De meeste onderwerpen die ik besproken heb in de laatste paragraaf worden ook aangegroerd in Garey en Johnson (zie hierboven) met uitzondering van het resultaat over het schaken, dat gepubliceerd is in de proceedings van de 8e ICALP gehouden in 1981 te Akko: *Computing a perfect Strategy for $n \times n$ Chess requires time exponential in n,* van de hand van A.S. Fraenkel en D. Lichtenstein; deze proceedings zijn uigegeven door S. Even & O. Kariv, en verschenen als aflevering 115 in de hierboven genoemde reeks Lecture Notes in Conputer Science van Springer.

PUBLIC KEY CRYPTOGRAFIE
door
Pieter J. Hoogendoorn

1. Inleiding

In de cryptografie gaat het in hoofdzaak om het oplossen van twee soorten veiligheidsproblemen, te weten *privacy* en *authenticiteit*. Een systeem dat privacy waarborgt garandeert de legale gebruikers van dat systeem dat de berichten, die via openbare kanalen worden verstuurd, alleen door de bedoelde ontvangers(s) kunnen worden gelezen; afluisteren is zinloos. Een systeem dat authenciteit waarborgt garandeert de gebruikers dat de berichten die zij ontvangen werkelijk zijn verstuurd door de "ondergetekenden"; figureren of illegaal berichten injecteren is niet mogelijk.

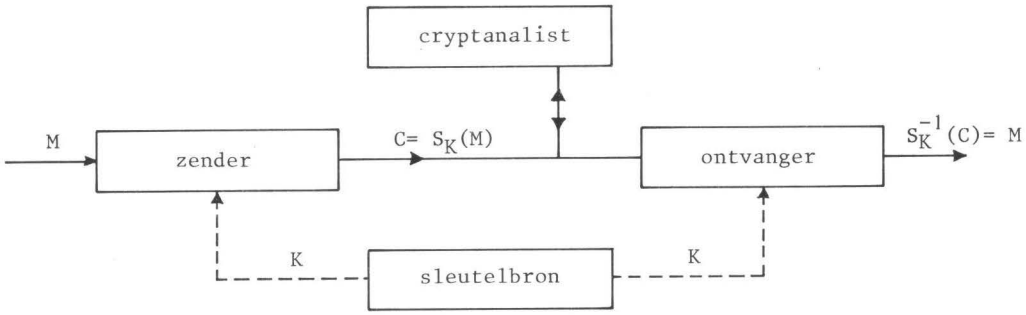


Fig. 1

Voor vertrouwelijke communicatie wordt al sinds jaar en dag gebruik gemaakt van wat wij thans *aonventionele* cryptografische systemen noemen (zie fig. 1). Er zijn drie partijen: een zender, een ontvanger en een afluisteraar (cryptanalist). De zender genereert een vertrouwelijk bericht M, dat over een openbaar kanaal verzonden moet worden naar een legitieme ontvanger. Om te voorkomen dat de afluisteraar de inhoud van M leert kennen verandert (vercijfert) de zender M d.m.v. een (omkeerbare) transformatie S_K in het cryptogram $C= S_K(M)$. De sleutel K is -via een betrouwbaar kanaal- alleen aan de legitieme ontvanger bekend gemaakt. Wanneer nu de ontvanger C binnen krijgt, ontcijfert hij dit cryptogram door er de inverse transformatie S_K^{-1} op toe te passen:

$$S_K^{-1}(C)= S_K^{-1}(S_K(M))= M.$$

De transformatie S_K wordt gekozen uit een algemene verzameling transformaties, die in wezen het cryptografisch systeem vormt. De sleutel is de parameter die de specifieke transformatie bepaalt. M.a.w. een conventioneel cryptografisch systeem is een verzameling $\{S_K\}_{K\in\mathcal{K}}$ van

omkeerbare transformaties S_K: $M \rightarrow Q$ van een ruimte M van klare berichten naar een ruimte Q van cryptogrammen.
De parameterverzameling K wordt wel de sleutelruimte genoemd.
De kunst is nu zodanige cryptosystemen te ontwerpen, dat voor iedere keuze van K de transformaties S_K en S_K^{-1} "goedkope" operaties zijn, d.w.z. S_K en S_K^{-1} zijn snelle algoritmen, terwijl iedere succesvolle actie van een cryptanalist, hetzij om berichten te onderscheppen dan wel om (valse) berichten te injecteren, te "duur" wordt, d.w.z. meer rekentijd en/of geheugenruimte vergt dan economisch verantwoord is. Dergelijke systemen zullen wij *praktisch betrouwbaar* noemen, de cryptanalitische acties om deze systemen te breken: *in de praktijk ondoenlijk*.
Wij merken nog op dat in een conventioneel cryptosysteem vercijfering en ontcijfering niet te scheiden functies zijn: ieder die een sleutel heeft om te vercijferen (zenden) kan ook ontcijferen (ontvangen).

Twee aspecten die het gebruik van conventionele cryptoystemen op grote schaal hebben beperkt zijn:

a. sleutelbeheer en -distributie
b. de onmogelijkheid berichten te ondertekenen.

Zoals figuur 1 laat zien dient elk tweetal gebruikers dat onderling wil communiceren over een sleutel "voor hen alleen" te beschikken, en om veiligheidsredenen wordt veelal per sessie een nieuwe sleutel gebruikt. In de praktijk komt dit erop neer dat door een betrouwbare centrale instantie aan ieder paar legitieme gebruikers dat onderling wil communiceren min of meer regelmatig sleutels wordt verstrekt. Bij een netwerk met N gebruikers zijn er N(N-1)/2 paren die in principe alle onderling moeten kunnen communiceren. Derhalve wordt bij grote netwerken de toepassing van conventionele cryptografische technieken alleen al i.v.m. het beheer en de distributie van sleutels een zeer gecompliceerde zaak. En, ofschoon conventionele cryptosystemen bescherming bieden tegen vervalsing van informatie, zijn zij in feite niet geschikt voor de hedendaagse commerciële praktijk, waarin de geldigheid van een overeenkomst wordt bepaald door een handtekening onder een document. Het kenmerkende van de handtekening is dat deze (in principe) slechts door één persoon kan worden gegenereerd, maar door iedereen kan worden herkend. De ontwikkeling van cryptografische systemen die de mogelijkheid van "digitale ondertekening" van berichten bieden is dus noodzakelijk.

Bovengenoemde aspecten zijn in belangrijke mate drijfveren geweest achter de ontwikkeling van de zgn. *public key cryptografie*, de ontwikkeling van systemen waarin ieder tweetal onderling veilig kan communiceren, zonder de noodzakelijke aanwezigheid van een centrale autoriteit die sleutels beheert en distribueert. In deze ontwikkeling zijn twee trends te onderscheiden: public key distributiesystemen en public key cryptosystemen. In een *public key distributiesysteem* kunnen de zender en ontvanger zonder tussenkomst van enige andere partij,

via het openbare kanaal een sleutel overeenkomen voor gebruik in een conventioneel cryptosysteem.
In een *public key cryptosysteem* worden verschillende sleutels gebruikt voor vercijfering en ontcijfering; deze sleutels zijn niet verwisselbaar, d.w.z. dat de sleutel voor vercijfering niet te gebruiken is voor ontcijfering en daarom ook niet geheim behoeft te worden gehouden.

In deze bijdrage wordt overwegend aandacht geschonken aan de ontwikkeling van public key cryptosystemen; een voorbeeld van een public key distributie systeem staat in sectie 3.

2. Public key cryptosysteem

In een public key cryptosysteem genereren zowel de zender als de ontvanger elk (eenmalig) twee verschillende sleutels: een vercijfersleutel, die dient om het vercijfer-algoritme van het systeem toe te kunnen passen, en een ontcijfer-sleutel, die dient om het ontcijfer-algoritme toe te kunnen passen. Zowel het vercijfer- als het ontcijfer-algoritme is bekend. Verder maakt iedere gebruiker zijn vercijfer-sleutel bekend, maar houdt zijn ontcijfer-sleutel geheim. De sleutels zijn zodanig gerelateerd, dat zij inverse transformaties specificeren: toepassing van het vercijfer-algoritme gevolgd door toepassing van het ontcijfer-algoritme levert het oorspronkelijke bericht op (en bij sommige systemen ook omgekeerd).

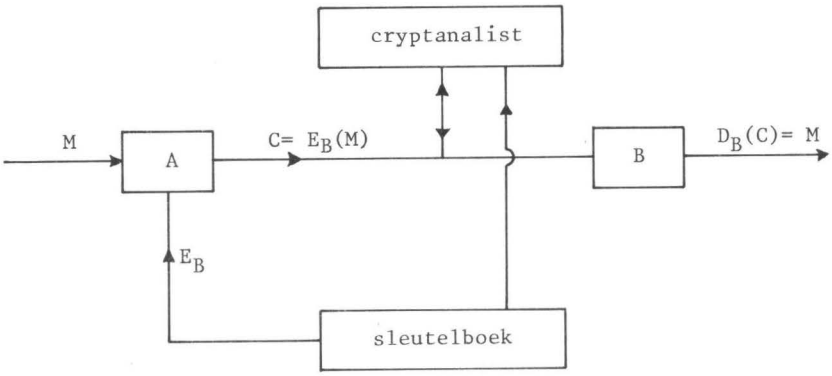


Fig. 2

systeem. Als partij A een bericht M wil zenden aan partij B, dan past A m.b.v. B's vercijfer-sleutel het vercijfer-algoritme E_B toe op M en stuurt hem het cryptogram $C= E_B(M)$. B ontcijfert dit cryptogram m.b.v. zijn (geheime) ontcijfer-sleutel door toepassing van het ontcijfer-algoritme D_B:

$D_B(C) = D_B(E_B(M)) = M.$

Formeel definiëren wij een public key cryptosysteem als een triple $(\mathcal{K}, \{E_K\}_{K\in\mathcal{K}}, \{D_K\}_{K\in\mathcal{K}})$ waarin $\mathcal{K}$ een eindige verzameling (van sleutels) is en $\{E_K\}_{K\in\mathcal{K}}$ en $\{D_K\}_{K\in\mathcal{K}}$ twee verzamelingen van transformaties (algoritmen)

$$E_K: \mathcal{M} \to \mathcal{M}; \quad D_K: \mathcal{M} \to \mathcal{M}$$

van een eindige verzameling $\mathcal{M}$ (van berichten), zo dat

(i) $\forall_{\mathcal{K}} K \; \forall_{\mathcal{M}} M : D_K(E_K(M)) = M$

(ii) $\forall_{\mathcal{K}} K$ zijn E_K en D_K snelle algoritmen

(iii) $\forall E_K$ is het in de praktijk ondoenlijk om uitgaande van E_K een transformatie van $\mathcal{M}$ te vinden die equivalent is met D_K.

De derde eigenschap brengt dus tot uiting dat de gebruikers van het systeem hun specifieke vercijfer-algoritme bekend kunnen maken zonder hun geheime ontcijfer-algoritme in gevaar te brengen.

Bovenstaande eigenschappen kunnen worden samengevat in de volgende (enigszins slordige) definitie:

Een public key cryptosysteem is een verzameling valluiken.

In deze context is een *valluik* een *trap-door one-way function*. Een funktie f is een one-way function als voor ieder argument x in het domein van f de funktiewaarde f(x) eenvoudig (d.w.z. snel) te berekenen is, terwijl het in de praktijk ondoenlijk is om gegeven een waarde y in het bereik van f een argument x te vinden zo dat f(x)= y; f is een trap-door one way function als met wat extra kennis van f (trap-door information) de vergelijking y= f(x) voor gegeven y wel eenvoudig is op te lossen. Wanneer f bovendien een bijectie is spreken we van een *one-way trap-door permutation*.

Intermezzo
Een public key cryptosysteem dat gebaseerd is op een one-way trap-door permutation heeft de extra eigenschap

(iv) $\forall_{\mathcal{K}} K \; \forall_{\mathcal{M}} M: E_K(D_K(M)) = M.$

M.b.v. zo´n systeem kan zowel het probleem van de privacy als van de authenticiteit worden opgelost. Om dit in te zien merken wij op, dat een systeem dat de eigenschappen ii, iii, en iv heeft, kan worden gebruikt om het probleem van de authenciteit op te lossen. Als A een door hem gewaarmerkt (ondertekend) bericht M naar B wil zenden past hij op M zijn ontcijfer-algoritme D_A toe, en zendt B het signaal

$S = D_A(M)$.

B kan vervolgens S ontcijferen door daarop het bekende vercijfer-algoritme E_A van A toe te passen:

$E_A(S) = E_A(D_A(M)) = M$.

Iedere gebruiker kan dus vaststellen dat A, en alleen A (immers alleen A kent D_A) M heeft verzonden.
Stel nu dat het systeem ook nog aan voorwaarde i voldoet. Dan kan A zich de nodige privacy verschaffen door op S -alvorens dit signaal te verzenden- B's bekende vercijfer-algoritme toe te passen: $E_B(S)$. Alleen B kent D_B, dus alleen B kan $E_B(S)$ ontcijferen (en vervolgens M bepalen). In sectie 4 wordt een voorbeeld gegeven van een public key cryptosysteem dat gebaseerd is op een trap-door one-way permutation.
Keren wij thans nog even terug naar de valluiken.

Zoals wij zagen is een public key cryptosysteem te beschouwen als een verzameling valluiken. Het ontwikkelen van een public key cryptosysteem kan dus beginnen bij het zoeken naar geschikte one-way functions (of permutations), waarin een trap-door kan worden "ingebouwd". In de praktijk heeft men dit ook gedaan. Hierbij is de klasse van "NP-problemen" een bron van kandidaat-valluiken gebleken (populair gezegd is de complexiteitsklasse NP de verzameling van problemen die praktisch onoplosbaar zijn, maar overigens de eigenschap hebben dat iedere oplossing die wordt gesuggereerd eenvoudig is te verifiëren; voor een juiste definitie van de klasse NP en meer complexiteitstheorie wordt de lezer verwezen naar het hoofdstuk *Cryptografie en complexiteit*, door P. van Emde Boas).
Enkele voorbeelden van public key cryptosystemen zijn te vinden in sectie 4.

3. Een public key distributiesysteem

Zoals wij reeds zagen is één van de problemen met conventionele cryptosystemen de distributie van sleutels. Diffie en Hellman [1] hebben daar het volgende op gevonden:
Zij q een priemgetal en a een primitief element van het lichaam GF(q). Zij verder

$y = a^x \bmod q, \quad 0<x<q,$

dan heet x de a-log van y modulo q:

$x = \log_a y \bmod q, \quad 0<y<q.$

Voor gegeven x is y eenvoudig (snel) te berekenen; volgens de methode "herhaald vermenigvuldigen en kwadrateren" kost deze berekening ten

hoogste $2.\log_2 q$ vermenigvuldigingen (modulo q). Bijvoorbeeld, voor x= 18 krijgen we

$$y= a^{18}= (((a^2)^2)^2)^2.a^2.$$

Maar om, gegeven y, x te berekenen is in het algemeen veel en veel moeilijker; voor geschikt gekozen waarden van q kost dit zelfs met het tot dusverre snelste algoritme, in de orde van $q^{\frac{1}{2}}$ operaties en geheugenruimte (zie [4]). En hiervan hebben Diffie en Hellman als volgt gebruik gemaakt: Iedere gebruiker genereert een willekeurig geheel getal x_i tussen 0 en q (q is van tevoren vastgesteld en vervolgens is een a bepaald). Hij houdt x_i geheim, maar maakt

$$y_i= a^{x_i} \bmod q$$

bekend. Wanneer de gebruikers i en j willen corresponderen gebruiken zij

$$K_{ij}= a^{x_i x_j} \bmod q$$

als hun sleutel. Gebruiker i berekent K_{ij} eenvoudig m.b.v. y_j:

$$\begin{aligned} K_{ij} &= y_j^{x_i} \bmod q \\ &= (a^{x_j})^{x_i} \bmod q \\ &= a^{x_j x_i} \bmod q \\ &= a^{x_i x_j} \bmod q. \end{aligned}$$

Gebruiker j berekent K_{ij} op soortgelijke wijze, nl.

$$K_{ij}= y_i^{x_j} \bmod q.$$

Iedere andere gebruiker die K_{ij} wil weten heeft slechts de beschikking over y_i en y_j, waarmee K_{ij} in principe als volgt kan worden bepaald:

$$K_{ij}= y_i^{(\log_a y_j)} \bmod q.$$

We zien dus dat dit systeem kan worden gebroken zodra men snel logaritmen in een eindig lichaam kan berekenen. Ofschoon tot op heden niet is bewezen dat het systeem veilig is zolang logaritmen in GF(q) moeilijk te berekenen zijn, is tot op heden geen methode gevonden om uitgaande van y_i en y_j de sleutel K_{ij} te berekenen zonder eerst x_i of x_j te bepalen.

4. Enige public key cryptosystemen

Wij zullen drie voorbeelden geven van public key cryptosystemen:

1. het RSA systeem
2. het knapsack systeem
3. het Goppa-code systeem.

Deze systemen zijn gebaseerd op one-way functions (het RSA systeem op een one-way permutation) die gerelateerd zijn aan NP-volledige problemen (zie *Cryptografie en complexiteit*, deze syllabus). De genoemde systemen worden tot in detail behandeld door A.E. Brouwer in diens bijdrage *Voorbeelden van public key cryptosystemen* aan deze syllabus.

4.1 *Het RSA systeem*

Het RSA systeem is ontworpen door Rivest, Shamir en Adleman [5]. Zij maken gebruik van het feit dat grote priemgetallen (van zeg 100 cijfers) eenvoudig zijn te vinden, maar dat het factoriseren van het product van twee van die getallen in de praktijk ondoenlijk is.
Iedere gebruiker kiest twee priemgetallen p en q (van tenminste 100 cijfers), vermenigvuldigt ze en maakt het resultaat

$$n = p.q$$

bekend (p en q houdt hij geheim). M.b.v. p en q berekent hij -in dit geval heel eenvoudig- de Euler totiëntfunktie Φ in n. ($\Phi(n)$ is het aantal gehele getallen kleiner dan n, welke onderling ondeelbaar zijn met n). Er geldt

$$\Phi(n) = \Phi(pq) = \Phi(p)\ \Phi(q) = (p-1)(q-1).$$

Vervolgens kiest hij een geheel getal e tussen 1 en $\Phi(n)$ zodanig dat ggd $(e, \Phi(n)) = 1$, dat hij ook bekend maakt. De public key is nu het paar (n,e).
Gegeven e bestaat er een getal d zo dat

$$ed = 1 \mod \Phi(n),$$

dus $ed = k\Phi(n)+1$, voor zekere k.
Voorts weten we uit de getaltheorie dat voor alle x tussen 0 en n-1 en voor alle k geldt dat

$$x^{k\Phi(n)+1} = x \mod n.$$

Dus geldt voor alle x (ook voor x= 0)

$$x^{ed} = x \mod n.$$

d is eenvoudig te berekenen, d is de secret key.

Een bericht wordt voorgesteld als een rij gehele getallen $M_1, M_2, \ldots$ waarbij iedere M tussen 0 en n ligt. Vercijfering vindt plaats door op ieder "blok" M de volgende transformatie toe te passen:

$$C = E(M) = M^e \bmod n.$$

Merk op dat $0<C<n$. Het cryptogram C wordt ontcijferd door het tot de macht d (modulo n) te verheffen:

$$\begin{aligned} D(C) &= C^d \bmod n \\ &= (M^e)^d \bmod n \\ &= M^{ed} \bmod n \\ &= M \bmod n. \end{aligned}$$

Wanneer nu de gebruikers overeenkomen hun berichten zodanig in blokken te hakken dat elk blok M kleiner is dan n_0, waarbij

$$n_0 < \min\{n_i \mid i \text{ gebruiker}\},$$

dan kan het RSA systeem ook worden gebruikt voor digitale ondertekening. Dit gaat als volgt:
Stel gebruiker i wil gebruiker j een getekend bericht M zenden (we nemen gemakshalve aan dat $M \leq n_0$). Het eerste wat i doet is nagaan of $n_i < n_j$ dan wel of $n_j < n_i$. Stel $n_i < n_j$, dan stuurt i naar j het cryptogram

$$\begin{aligned} S &= E_j(D_i(M)) \\ &= E_j(M^{d_i} \bmod n_i) \\ &= (M^{d_i} \bmod n_i)^{e_j} \bmod n_j \\ &= (M^{d_i})^{e_j} \bmod n_j. \end{aligned}$$

De lezer ga zelf na hoe j het cryptogram S ontcijfert (wat te doen als $n_j < n_i$?). Verder verifiëre hij de eigenschappen i t/m iv uit sectie 2.

4.2 *Het knapsack systeem*

Het knapsack systeem (ontwikkeld door Merkle en Hellman [3]) is gebaseerd op een bekend probleem uit de combinatoriek, het zgn. knapzakprobleem:

gegeven een vector $\underline{a} = (a_1, a_2, \ldots, a_n)$ van positieve gehele getallen en een positief geheel getal S, bepaal een deelverzameling van

$\{a_i\}_{i=1, \ldots, n}$

zo dat de som van de elementen van die deelverzameling gelijk is aan S. M.a.w. gegeven $\underline{a}$ en S, vind een binaire vector

$\underline{x} = (x_1, x_2, \ldots, x_n)$ zo dat $\underline{a}.\underline{x} = S$.

Van het knapzak-probleem wordt aangenomen dat het *moeilijk* is (zie *Cryptografie en complexiteit*, deze syllabus), maar eenvoudige versies zijn bekend. Hiervan hebben Merkle en Hellman gebruik gemaakt; zij beginnen met een eenvoudige versie en veranderen die in een ingewikkelde.
De vector $\underline{a}$ kan worden gebruikt om berichten te vercijferen door elk bericht in blokken $\underline{x}_i$ van n bits te hakken en per blok het inwendig product $S_i = \underline{a}.\underline{x}_i$ te berekenen. De S_i vormen dan het cryptogram. Gegeven $\underline{a}$ en S_i moet dus een knapzak-probleem worden opgelost om $\underline{x}_i$ terug te vinden, hetgeen in de praktijk wel eens ondoenlijk kan blijken te zijn. Om dit te voorkomen wordt $\underline{a}$ zodanig gekozen dat iedere a_i groter is dan de som van zijn voorgangers (de rij $a_1, a_2, \ldots$ heet een *super stijgende rij*.) Een voorbeeld:

Stel $\underline{a}' = (171, 197, 459, 1191, 2410)$ en $S' = 3798$.
Nu moet x_5 gelijk zijn aan 1. Immers als $x_5 = 0$ en $x_1 = x_2 = x_3 = x_4 = 1$, dan nog zou $\underline{a}'.\underline{x} < S'$ $(x_5' > x_1' + x_2' + x_3' + x_4')$.
Dus moet $S' - a_5' = 1388$ de som zijn van een deelverzameling van de eerste vier elementen van $\underline{a}'$. Daar $1388 > 1191$ volgt dat $x_4 = 1$ en tenslotte $S - a_5' - a_4' = 197 = a_2'$ dus $x_3 = 0$, $x_2 = 1$ en $x_1 = 0$.

Het is duidelijk dat de eenvoudige knapzakvector $\underline{a}'$ niet te gebruiken is als public key. Wat de gebruiker (A) doet is het volgende: naast de eenvoudige knapzakvector $\underline{a}'$ (met zeg 100 elementen) genereert hij een groot getal m, zo dat $m > \Sigma a_i'$ en een getal w dat onderling ondeelbaar is met m (dan bestaat er ook een getal w^{-1} zo dat $ww^{-1} = 1$ mod m); $\underline{a}'$, m, w en w^{-1} worden geheim gehouden. Vervolgens construeert A de public knapsack vector $\underline{a} = w\underline{a}'$ mod m, d.w.z.

$a_i = wa_i'$ mod m, $i = 1, 2, \ldots, n$.

Door deze operatie wordt de super stijgende structuur van $\underline{a}'$ verhuld. Wanneer nu een andere gebruiker aan A een vertrouwelijk bericht $\underline{x}$ wil sturen, dan berekent hij m.b.v. A's vercijfer-sleutel $\underline{a}$ het cryptogram

$S = \underline{a}.\underline{x}$

en stuurt dit naar A. Om S te ontcijferen berekent A

$$
\begin{aligned}
S' &= w^{-1}S \mod m \\
&= w^{-1}\underline{a}.\underline{x} \mod m \\
&= w^{-1}\Sigma a_i x_i \mod m \\
&= w^{-1}\Sigma(wa'_i \mod m)x_i \mod m \\
&= \Sigma(w^{-1}wa'_i \mod m)x_i \mod m \\
&= \Sigma\, a'_i x_i \mod m \\
&= \underline{a}'.\underline{x},
\end{aligned}
$$

want $m > \Sigma\, a'_i$. En dit is een eenvoudig knapzak-probleem.

Het knapsack systeem voldoet i.h.a. niet aan voorwaarde iv van sectie 2 en is i.h.a. dus niet te gebruiken voor authenticatie doeleinden. De reden hiervoor is, dat gegeven $\underline{a}$, de meeste kandidaat-cryptogrammen S, met $0 \leq S \leq \Sigma a_i$, geen origineel (originele boodschap) $\underline{x}$ hebben (waarvoor dus geldt $\underline{a}.\underline{x} = S$).

4.3 *Het Goppa-code systeem*

Het Goppa-code systeem, geïntroduceerd door R.J. McEliece [2], is gebaseerd op het feit dat er voor sommige lineaire codes, i.h.b. voor de zgn. Goppa codes, snelle decodeer-algoritmen bestaan, terwijl het decoderen van een algemene lineaire code te boek staat als een NP-volledig probleem.

Een lineaire code is een deelruimte van een lineaire ruimte over een eindig lichaam. Codewoorden zijn dus vectoren; het gewicht $w(\underline{x})$ van een woord $\underline{x} = (x_1, x_2, \ldots, x_n)$ is het aantal $x_i \neq 0$. Een lineaire code wordt voortgebracht door een *generator matrix* ; de hoogte/breedte van de generator matrix zijn respectievelijk de dimensie en de woordlengte van de code.
Een t-fouten-verbeterende code is een code die de eigenschap heeft dat, wanneer tijdens het transport een codewoord (vector) op t (of minder) plaatsen wordt verminkt, het resultaat toch nog goed gedecodeerd kan worden.
Zoals gezegd is een Goppa-code een lineaire code waarvoor een snel decodeer-algoritme bestaat. Om het decodeer-algoritme toe te passen moet je evenwel de generator matrix van de code kennen. Dit feit wordt als volgt uitgebuit.

Zij voor gegeven t en n (n= woordlengte) G de k x n generator matrix van een Goppa code; (er geldt $k \geq n - t\log_2 n$). Zij S een niet-singuliere k x k matrix en P een willekeurige n x n permutatie-matrix. Dan genereert de matrix

$G' = SGP$

een k-dimensionale t-fouten-verbeterende code met woordlengte n. Gegeven S en P zijn S^{-1} en P^{-1} eenvoudig te berekenen. De gebruiker (A) maakt G' bekend, maar G, S^{-1} en P^{-1} houdt hij geheim. Wanneer een andere gebruiker een vertrouwelijk bericht M naar A wil zenden, dan gaat hij als volgt te werk:
Het bericht M wordt in blokken gehakt van k bits. Stel $\underline{u}$ is zo'n blok, dan verstuurt hij de vector

$$\underline{x} = \underline{u}G' + \underline{z},$$

waarbij $\underline{z}$ een willekeurige fout-vector van gewicht $w(\underline{z}) \leq t$ is.
A ontcijfert $\underline{x}$ als volgt: eerst berekent hij

$$\begin{aligned} \underline{x} &= \underline{x}P^{-1} \\ &= (\underline{u}G' + \underline{z})P^{-1} \\ &= \underline{u}SG + \underline{z}P^{-1}. \end{aligned}$$

$\underline{x}'$ kan nu m.b.v. het decodeer-algoritme snel worden gedecodeerd, immers $w(\underline{z}P^{-1}) \leq t$. Dit levert $\underline{u}S$, waaruit tenslotte $\underline{u}$ volgt door vermenigvuldiging met S^{-1}.

De lezer verifiëre zelf de voorwaarden i,ii en iii uit sectie 2.

Aangezien, zolang $w(\underline{z}) \leq t$, vele verschillende fout-vectoren $\underline{z}$ kunnen worden opgeteld bij $\underline{u}G'$ zonder enige invloed op het uiteindelijke resultaat van de ontcijfering, is het Goppa-code systeem niet geschikt voor authenticatiedoeleinden.

5. Referenties

[1] Diffie, W. en M.E. Hellman, New directions in cryptography, IEEE Trans. Inf. Theory, Vol. IT-22, No. 6, Nov. 1976.

[2] McEliece, R.J., A public key cryptosystem based on algebraic coding theory, JPL DSN Progress Report 42-44, Jan.-Febr. 1978.

[3] Merkle, R.C. en M.E. Hellman, Hiding information and signatures in trap door knapsacks, IEEE Trans. Inf. Theory, Vol. IT-24, No. 5, Sept. 1978.

[4] Pohlig, S.C. en M.E. Hellman, An improved algorithm for computing logarithms in GF(p) and its cryptographic significance, IEEE Trans. Inf. Theory, Vol. IT-24, No. 1 , Jan. 1978.

[5] Rivest, R.L., A. Shamir en L. Adleman, On digital signatures and public key cryptosystems, Commun. ACM, Vol. 21, No. 2, Febr. 1978.

V.2 Voorbeelden van Public Key cryptosystemen (Meer over het kapzaksysteem)

door A.E. Brouwer

Zoals boven uiteengezet (zie "Public Key Cryptography", H V.1) wordt bij het knapzak systeem door degene die boodschappen wil ontvangen een vector (rij) van getallen $a_1,\ldots,a_N$ gepubliceerd. Degene die hem een boodschap wil sturen hakt deze in mootjes van N bits, en een blok van N bits $x_1,\ldots,x_N$ (elke x_i is 0 of 1) wordt verstuurd als $\sum_{i=1}^{N} a_i x_i$, d.w.z. de a_i's die horen bij een $x_i = 1$ worden opgeteld.

Voor N neemt men gewoonlijk een getal in de grootteorde van 100; de a_i's zijn zelf getallen van 50 à 100 cijfers elk.

Van alle tot nu toe voorgestelde openbare sleutel cryptosystemen is dit systeem verreweg het eenvoudigste, en met een hardware implementatie zijn hoge data rates (10M bits/sec. [Henry]) mogelijk.

Echter, het systeem is lineair, en in de cryptografie heeft men lineaire systemen leren wantrouwen: ze zijn stukken eenvoudiger te kraken dan niet-lineaire systemen.

Enige voorzichtigheid is hier dan ook geboden.

<u>Voorbeeld</u>

Stel dat een hoofdkwartier een zekere boodschap naar een aantal buitenposten stuurt, en elke buitenpost B_k heeft zijn eigen knapzak systeem $(a_{ik})_i$, dan wordt dus $M_k = \sum_{i=1}^{N} a_{ik} x_i$ verstuurd voor $1 \leq k \leq A$, met A het aantal buitenposten. De a_{ik} zijn bekend en de M_k kunnen afgeluisterd worden. Als nu $A \geq N$ en de knapzak vectoren zijn onafhankelijk gekozen (om precies te zijn: als rang $(a_{ik}) \geq N$), dan kan de spion eenvoudig A vergelijkingen met N onbekenden oplossen en zo de boodschap $x_1,\ldots, x_N$ vinden. Uit veiligheidsoverwegingen wil men niet alle buitenposten dezelfde knapzak geven; dit betekent dus dat het verboden is om dezelfde boodschap aan veel correspondenten te versturen.

Mogelijke oplossingen zijn:

- Het toepassen van dubbele codering zodat de boodschap al ontvanger-afhankelijk is gecodeerd voor hij de knpazak ingaat.
- De ontvangers publiceren elk hun modulus M zodat niet $\sum_{i=1}^{N} a_i x_i$ maar $\sum_{i=1}^{N} a_i x_i \pmod M$ verstuurd wordt. Dit heeft echter het nadeel dat het nu veel makkelijker wordt om de knapzak te kraken (zie [Shamir & Zip-

pel]).

- Houd de knapzakvectoren geheim: in een eerste contact (via een of ander openbare sleutel cryptosysteem) komen beide partijen de knapzakvectoren overeen die ze voortaan zullen gebruiken. Nu wordt de knapzak dus gebruikt als klassiek (d.w.z. geheime sleutel) systeem en blijft alleen het voordeel dat hij makkelijk te implementeren is in hardware of software.

[Opmerking. Het probleem van herhaalde transmissie van dezelfde boodschap doet zich ook bij andere systemen voor. Vooral in een context waar boodschappen door een computer gegenereerd worden is het zeer waarschijnlijk dat deze stereotiep van aard zijn. Stel bijvoorbeeld dat we het Goppa-code systeem gebruiken (zie "Public key cryptografie" H V.1) en iemand stuurt ons meermalen dezelfde boodschap. In de notatie van H V.1 wordt dus telkens $\underline{u}G'+\underline{z}^{(j)}$ verstuurd, waarbij de $\underline{z}^{(j)}$ willekeurige foutvectoren van gewicht hooguit t zijn. Maar na drie of vier van zulke boodschappen onderschept te hebben is het de spion duidelijk wat de $\underline{z}^{(j)}$'s geweest zijn zodat hij $\underline{u}G'$ en daarna ook $\underline{u}$ kent, d.w.z. de boodschap is ontcijferd.]

Voorbeeld 2

Stel dat de spion de getallen W en M kan vinden zodanig dat als $b_i = Wa_i$ (mod M) (onder a (mod b) zal ik de rest verstaan bij deling van a door b: 666 (mod 7)= 1) dan $\sum_{i=1}^{N} b_i < M$ en $\sum_{i=2}^{N} b_i < b_1$. (Voorbeeld: als a_1 oneven is en alle andere a_i zijn even kies dan W= 1 en M= 2.) Nu kan hij al 1 bit van ieder binnengekomen blok van N bits lezen: bij afluisteren van $\sum_{i=1}^{N} a_i x_i = \sum_{x_i=1} a_i$ vermenigvuldigt hij dit met W en reduceert mod M en vindt $\sum_{x_i=1} b_i$. Is dit groter of gelijk b, dan was $x_1 = 1$ en anders was $x_1 = 0$. Om de resterende bits te vinden moet nu een iets kleinere knapzak (met N-1 getallen) gekraakt worden.

(Deze observatie is de basis van het (overigens waardeloze) artikel van Desmedt, Vanderwalle en Govaerts [ ]. Hij was al eerder gemaakt in het (bijna even slechte) artikel van Herlestam [ ]. Ingemarsson [ ] vindt knapzakken waarop deze aanval niet van toepassing is.)

Knapzak varianten

De ontvanger moet de knapzakvectoren $a_1,\ldots,a_N$ natuurlijk zo kiezen dat hijzelf uit $\sum_{i=1}^{N} a_i x_i$ weer de x_i terug kan vinden.
Het oorspronkelijke voorstel ([Merkle & Hellman]) was om uit te gaan van een superstijgende rij getallen a_i' (d.w.z. zo, dat $a_{i+1}' > \sum_{j=1}^{i} a_j'$) en getallen W', M met ggd $(W',M)= 1$ en $M > \sum_{j=1}^{N} a_j'$ en dan $a_i = W'a_j'$ (mod M) te publiceren. Als W zo is dat $WW' = 1$ (mod M) (zo'n W bestaat omdat ggd $(W',M)= 1$) dan is $W \Sigma a_i x_i = \Sigma a_i' x_i$ (mod M) en hieruit zijn eenvoudig de x_i af te lezen (zie H V.1).
Dit systeem moet echter als gekraakt beschouwd worden ([Shamir], zie onder).

Een variatie op dit idee is afkomstig van Graham & Shamir (zie [Shamir & Zippel]). Hierbij hebben de a_i' de speciale vorm $a_i' = Kop_i | b_i' | Staart_i$ waarbij '|' concatenatie (achterelkaar schrijven) van bitstrings voorstelt, Kop_i en $Staart_i$ willekeurig gekozen bitstrings zijn (maar wel met dezelfde lengte voor alle i) en b_i' een superstijgende rij is; de enige extra voorwaarde is dat bij het optellen van alle staarten geen overdracht in het gebied van de b_i' mag komen - hiervoor kan men zorgen door de staarten met $\lceil 2 \log N \rceil$ nullen te laten beginnen. (Weer wordt natuurlijk $a_i = W'a_i'$ (mod M) gepubliceerd.) Het voordeel van deze constructie is dat hij de meest opvallende zwakte van het gebruik van superstijgende rijen, namelijk dat het kleinste getal vele ordes van grootte kleiner is dan de modulus, vermijdt: alle a_i' zijn even groot, en het superstijgende gedeelte zit binnenin verstopt.

Als speciaal geval van dit schema kan men $b_i = 0\ldots010\ldots0$ kiezen, met i-1 nullen voor de 1 en N-i nullen na de 1. Nu is decoderen voor de ontvanger wel zeer eenvoudig: vermenigvuldig met W, reduceer modulo M, gooi de kop en de staart weg, en daar staat de boodschap.

Benjamin Arazi([ ]) merkt op dat het gewenst is eenzelfde boodschap telkens op een andere manier te coderen, en stelt het volgende systeem voor: De ontvanger publiceert N knapzakgetallen a_i en een getal B_{max}. De zender codeert nu een boodschap B die voldoet aan $0 \leq B \leq B_{max}$ als

$$B + \sum_{i \in J} a_i$$

d.w.z. hij telt een willekeurig gekozen stel a_i's bij zijn boodschap op.

Gezien het grote aantal mogelijke keuzen voor J (zeg $\binom{100}{50} \approx 10^{29}$) is het voor de spion ondoenlijk om alle mogelijke sommen af te trekken teneinde B terug te vinden. (Bovendien, misschien is het niet mogelijk B te herkennen als goede boodschap.) Inderdaad bereiken we zo dat dezelfde boodschap op vele verschillende manieren wordt gecodeerd.

De ontvanger had de a_i als volgt geconstrueerd:

Begin met een superstijgende rij $c_0', c_1', \ldots, c_N'$, $c_{N+1}' = M$. Kies willekeurige gehele getallen b_i' (mogelijk negatief) en schrijf

$d_i' = Mb_i' + c_i'$ $(i = 1, \ldots, N)$.

Kies een willekeurig getal p en resten r_i' met $p > r_i' \geq 0$.

Publiceer nu $a_i = pd_i' + r_i'$ en $B_{max} = 2^h$ met $h = \lfloor {}^2\log p.(c_0' - n + 1) \rfloor$.

Inderdaad, bij ontvangst van $B + \sum_{i \in J} a_i$ delen we eerst door p; dit geeft quotiënt

$\sum_{i \in J} d_i' + R$ met $R \leq c_0'$. Reduceer nu modulo M; dit geeft

$\sum_{i \in J} c_i' + R$ en hieruit zijn J en R eenvoudig af te lezen.

Nu volgt $B = (B + \sum_{i \in J} a_i) - \sum_{i \in J} a_i$.

Zoals gezegd heeft dit schema het voordeel dat elk bericht op veel manieren gecodeerd kan worden. Een bijkomend voordeel is dat de zender sommen $S_J = \sum_{i \in J} a_i$ in de leeglooptijd van zijn machine kan berekenen en opslaan, en op het moment dat er iets verstuurd moet worden slechts een enkele optelling $B + S_J$ hoeft uit te voeren. Aan de andere kant is de decodering gecompliceerder dan in het Graham-Shamir schema.

Shamir [Shamir 1979] heeft een 'Graham-Shamir'-achtige variant van het schema van Arazi aangegeven: weer wordt een boodschap B gecodeerd als

$B + \sum_{i \in J} a_i$ voor een willekeurig gekozen $J \subset \{1, 2, \ldots, N\}$.

Deze keer worden de a_i's als volgt gemaakt:

Kies $a_i' = Kop_i | nullen | Staart_i$ met willekeurig gekozen Kop_i en $Staart_i$ bitstrings; laat alle strings $Staart_i$ s bits lang zijn en zo gekozen zijn dat $\Sigma Staart_i < 2^s$, en laat alle strings met nullen g+1 bits lang zijn.

Kies een getal M en een getal W met ggd (W,M)= 1, zodat er een W' bestaat met $WW' = 1 \pmod M$. Laat W s+1 bits lang zijn (d.w.z. kies $2^s < W < 2^{s+1}$).

Publiceer nu $a_i = W'a_i' \pmod M$ en een bovengrens $B_{max} = 2^g$ voor B.

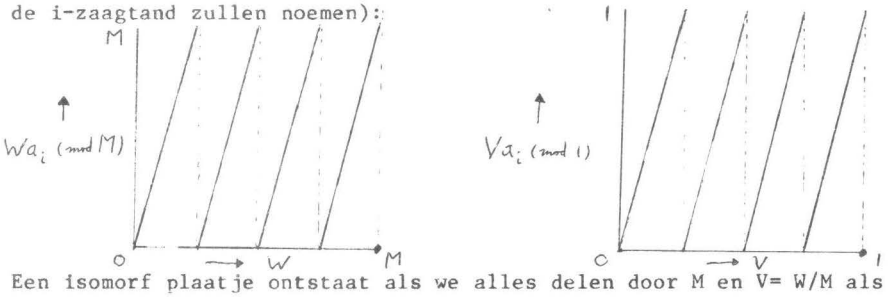
Het decoderen gaat als volgt: ontvangt men $B + \sum_{i \in J} a_i$ dan geeft vermenigvuldigen met W en reduceren mod M het getal $BW + \sum_{i \in J} a_i'$. Na weggooien van kop en staart van dit getal vindt men de eerste g+1 bits van het product BW, en

een deling door W levert nu de g bits van B.
Dit systeem heeft dezelfde voordelen als het Arazi systeem, en het decoderen is wat eenvoudiger. Shamir claimt dat dit de veiligst bekende variant op de knapzak is.

Hoe Shamir de Merkle-Hellman knapzak kraakt

Hieronder een (zeer ruwe) schets van Shamir´s knapzakprocedure. Het resultaat is dat met grote kans een Merkle-Hellman knapzak in polynomiale tijd gekraakt wordt. (Dit heeft Shamir een $ 100,- check van Merkle opgeleverd.)
Bij een Merkle-Hellman knapzak heb ik N getallen a_i zodanig dat Wa_i (mod M) een superstijgende rij vormt. Neem aan dat het i-de getal van de superstijgende rij n+i-1 bits heeft, zodat het kleinste getal kleinder is dan 2^n, het grootste kleiner dan 2^{n+N-1}, en neem aan dat $2^{n+N-1} < M < 2^{n+N}$.
(In hun oorspronkelijke artikel bevelen Merkle & Hellman dit schema aan met n=N=100.)
De cryptanalyst kent de a_i maar helaas niet W en M, en beschouwt W en M dus maar als variabelen. De W en M gekozen door de ontwerper van de knapzak zullen we W_0 en M_0 noemen. Het is echter denkbaar dat de ontwerper voor publikatie de a_i´s gepermuteerd heeft, zodat a_1 niet noodzakelijk overeenkomt met het kleinste element uit de superstijgende rij.

Als we de grafiek van Wa_i (mod M) (met W als onafhankelijk variabele) tekenen (en alle reële waarden van W toelaten) krijgen we een zaagtand (die we de i-zaagtand zullen noemen):



Een isomorf plaatje ontstaat als we alles delen door M en V= W/M als variabele kiezen. Dit nieuwe plaatje kennen we echt - we kennen immers a_i.

Als we aannemen dat er niet gepermuteerd is, zodat $W_0 a_1 \pmod{M_0}$ het kleinste element uit de superstijgende rij is, dan geldt (met $V_0 = W_0/M_0$) dat

$V_0 a_1 \pmod 1 < 2^n/2^{n+N-1} = 2^{-N+1}$ zodat V_0 vlakbij een van de nulpunten van de 1-zaagtand ligt. In feite is de richtingscoëfficiënt van de 1-zaagtand a_1, en a_1 is een getal van dezelfde orde van grootte als M zodat V_0 van het nulpunt van de 1-zaagtand minder dan $2^{-N+1}/a_1 < K.2^{-n-2N}$ afligt - hierbij is K een kleine constante, bijvoorbeeld K= 10.

Iets dergelijks geldt voor elke zaagtand: ook bij a_2, a_3, a_4 enz. vinden we dat V_0 dichtbij een nulpunt van de bijbehorende zaagtand ligt.

Leggen we de grafieken van al die zaagtanden op elkaar dan moet er dus een plaats zijn waar veel nulpunten vlakbij elkaar liggen. De nulpunten van de i-zaagtand zijn c_i/a_i met $0 \leq c_i \leq a_i$, c_i een geheel getal. Als we c_1/a_1 vast kiezen en kijken naar de kans dat punten c_i/a_i voor i= 2,...,k willekeurig gekozen in het interval $[\frac{c_1+\frac{1}{2}}{a_1}, \frac{c_1-\frac{1}{2}}{a_1}$ niet verder dan $K.2^{-n-2N}$ van c_1/a_1 afliggen (voor een of andere constante K , bijvoorbeeld K= 250) dan is dat iets in de orde van

$$\left(\frac{2K.2^{-n-2N}}{2^{-n-N}}\right)^{k-1} = \left(\frac{2K}{2^N}\right)^{k-1}$$

Maar c_1 kan vrij gekozen worden, dus de kans dat voor een of andere c_1 dit gebeurt is niet meer dan

$$a_1 \cdot \left(\frac{2K}{2^N}\right)^{k-1} < (2K)^{k-1} . 2^{n-(k-2)N}$$

Conclusie: kiezen we k zo dat (k-2)N veel groter is dan n en is 2^n veel groter dan 2K dan is de kans op een toevallige opeenhoping van k nulpunten van zaagtandkrommen bijzonder klein.

In het geval N=n=100 is het voldoende om k= 4 te nemen: het is uiterst onwaarschijnlijk dat bij superpositie van de i-zaagtanden voor i= 1,2,3,4 een nulpuntencluster zoals vereist optreedt.

Echter, we weten dat er zo´n ophopingspunt bestaat, namelijk V_0, dit volgt uit de constructie van de a_i; we kunnen nu veilig aannemen dat V_0 het enige ophopingspunt is (of in elk geval dat er slechts weinig zijn zodat we ze een voor een kunnen proberen).

Dit ophopingspunt kunnen we vinden met H.W. Lenstra jr.'s geheeltallige lineaire programmeringsalgoritme: we moeten de ongelijkheden

$$1 \leq c_i \leq a_i - 1 \qquad (i = 1,2,3,4),$$

$$\left| \frac{c_1}{a_1} - \frac{c_i}{a_i} \right| \leq K.2^{-n-2N} \qquad (i = 2,3,4).$$

met geheeltallige onbekenden c_i (i= 1,2,3,4) oplossen, hetgeen mogelijk is in een tijd polynomiaal in max (n,N).
Hebben we deze cluster van nulpunten eenmaal gevonden dan weten we dat V_0 groter is dan elk van hen, maar zo weinig groter dat $V_0 a_i$ (mod 1) nog kleiner is dan 2^{-N+i} (i= 1,2,...,N). Deze restricties leveren ons een klein intervalletje $[\alpha,\alpha+\varepsilon)$ waarbinnen V_0 moet liggen en waarbinnen bovendien alle N zaagtandkrommen continu zijn (ze blijven immers kleiner dan 1).

Op dit moment kunnen we ons even zorgen maken over de mogelijkheid dat de a_i's voor publikatie gepermuteerd zouden kunnen zijn. In dat geval zijn we verplicht alle mogelijke keuzen voor $a_1,\ldots,a_k$ te proberen, hetgeen het werk een factor N^k langer laat duren. Echter, de totale rekentijd blijft polynomiaal in max (n,N). Hebben we eenmaal $a_1,\ldots,a_k$ goed gegokt dan eisen we dat $V_0 a_1$ (mod i) kleiner is dan 2^{-N+i} voor i= 1,2,...,k en kleiner dan 1 voor alle i.
Dit levert weer het intervalletje $[\alpha,\alpha+\varepsilon)$ waarbinnen V_0 moet liggen en waarbinnen alle zaagtandkrommen continu zijn.

In het intervalletje $[\alpha,\alpha+\varepsilon)$ snijden de zaagtandlijnen elkaar in niet meer dan N^2 punten, zodat we het intervalletje kunnen onderverdelen in ten hoogste N^2 deelintervallen waarbinnen deze lijnen een constante verticale ordening hebben (dit bepaalt de permutatie π die gebruikt is).
Binnen elk deelinterval kan ik de voorwaarden voor het superstijgend zijn van de rij $Va_{\pi(i)}$ (mod 1) opschrijven: dit zijn lineaire ongelijkheden in V, d.w.z. deze voorwaarden definiëren een subdeelinterval (l_j,r_j) van het beschouwde deelinterval. Tenminste een van de subdeelintervallen (l_j,r_j) is niet leeg (want V_0 ligt in zo'n subdeelinterval), en elk rationaal getal W/M in dit subdeelinterval verandert onze knapzak in een superstijgende knapzak en stelt ons in staat de knapzak op te lossen.

Referenties

Benjamin Arazi, A trapdoor multiple mapping, IEEE Trans. on Inf.Th. 26 (1980), 100-102.

Y. Desmedt, J. Vandewalle & R. Govaerts, A critical analysis of the security of knapzak public key algorithms, preprint (submitted to IEEE Trans. on Inf.Th.)

Tore Herlestam, Critical remarks on some public-key cryptosystems, BIT 18 (1978), 493-496.

P.S. Henry, Fast decryption algorithm for the knapsack cryptographic system, Bell System Techn. J. 60 (1981), 767-773. See also: Computers & Security 1 (1982), 80-83.

Ingemar Ingemarsson, Knapsacks which are not partly solvable after multiplication modulo q, IBM research report RC 8515 (#37034) Okt. 10,1980.

R.C. Merkle & M.E. Hellman, Hiding information and signatures in trapdoor knapsacks, IEEE Trans. on Inf.Th. 24 (1978), 524-530.

Adi Shamir & Richard E. Zippel, On the security of the Merkle-Hellman Cryptographic Scheme, IEEE Trans. on Inf.Th. 26 (1980), 339-340.

Adi Shamir, On the cryptocomplexity of knapsack systems, Laboratory for Computer Science, MIT report MIT/LCS/TM-129, April 1979 = Proc. 11th ACM Symp. on the Theory of Computing, Atlanta, Georgia 1979, pp. 118-129.

Adi Shamir, A polynomial time algorithm for breaking the Merkle-Hellman Cryptosystem, Research announcement, April 20th 1982.

H.VI De ´Data encryption standard´

A.E. Brouwer

In 1973 en 1974 stelde het Amerikaanse National Bureau of Standards publiekelijk de vraag naar een cryptografisch systeem dat geschikt was om als standaard voor de overheidsdiensten gebruikt te worden. Het door de IBM ontwikkelde ´Lucifer-algoritme´ (zie bijv. [Smith]) is (gespecialiseerd en gekortwiekt) in 1977 ondanks alle daartegen gerezen protesten als US encryption standard aangenomen [NBS].

DES is een ´klassiek´ systeem (d.w.z. geen public key cryptosystem): encryptie en decryptie vinden plaats met dezelfde sleutel.
Diagram:

```
                |                        |
                | Sleutel S              | Sleutel S
                |                        |
                v                        V
64 bits       -------  64 bits         -------  64 bits
input I       |     |  cijfertekst     |     |  klare tekst I
----------->| DES |----->...------->| DES |----->
              |     |                  |     |
              -------                  -------

          Versluiering            Ontsluiering
```

DES is een gehaktmolen die de input bits op een gecompliceerde manier door elkaar roert en verminkt, waarbij de precieze operaties gestuurd worden door bits uit de sleutel. Voor een exacte beschrijving van het algoritme zie de appendix.

Gebruik van DES.

De meest voor de hand liggende manier om DES te gebruiken is een sleutel S te kiezen en vervolgens de te versluieren tekst in blokken van 8 bytes (64 bits) te hakken (eventueel aan het eind enige vulsymbolen toevoegend) en elk te versleutelen met de sleutel S.
Dit wordt wel ´Electronic code book mode´ genoemd: een gegeven data blok wordt telkens op dezelfde manier gecodeerd, alsof het in een gigantisch code boek werd opgezocht. (Met andere woorden, we hebben hier niets anders dan een monalfabetische substitutie met een alfabet van 2 ** 64 letters.) Een nadeel van deze methode is dat struktuur van de tekst nog in de code terug te vinden is; ook zou een spion een lijst kunnen gaan aanleggen met overeenkomstige klare tekst - cijfertekst paren, en zo misschien fragmenten van de boodschap lezen. Zo´n lijst behoeft niet eens zo lang te worden, want het is bekend dat in Engelse tekst niet 2 ** 64 (zeg 20000000000000000000) maar slechts ongeveer 2 ** 12 (zeg 4000) groepen van 64 bits met niet verwaarloosbaar kleine frequentie voorkomen [Ingemarsson].

Een aanzienlijk veiliger gebruiksmethode is de zogenaamde "Cipher Block Chaining Mode".
Hierbij versleutelt men niet de klare tekst, maar de som van de klare tekst en de het laatst verzonden cijfertekst:

```
                 |                        |
                 | S                      | S
                 |                        |
                 v                        v
klare         -------                  -------        klare
tekst ---     |     |   cijfertekst    |     |   ---  tekst
----->|+|-->| DES |----->...------->| DES |-->|+|------->
      ---     |     |   |         |    |     |   ---
       ^      -------   |         |    -------    ^
       |                |         |               |
       \--------------/           \---------------/
        vorige output             vorige cijfertekst

        Versluiering              Ontsluiering
```

(Voor de allereerste versluiering/ontsluiering is er geen ´vorige cijfertekst´ en moet men een vaste waarde afspreken - de ´initial chaining value´ - die in plaats van de ´vorige cijfertekst´ gebruikt wordt.)
Hoewel bij deze manier van versluieren elke output afhangt van alles wat eerder verzonden is, is het niet zo dat als een transmissiefout een bit cijfertekst verminkt heeft alle verdere tekst onleesbaar is geworden - integendeel, men controleert eenvoudig dat 1 fout invloed heeft op de ontcijfering van slechts 2 blokken informatie (16 bytes).

Chaining kan ook gebruikt worden om vul-symbolen te vermijden: voor sommige toepassingen is het belangrijk dat klare tekst en cijfertekst precies evenveel ruimte in beslag nemen; als de klare tekst nu niet precies een lengte die een 8-voud is heeft, kan men het restje R (met een lengte van m bits) versluieren als

R + linker m bits van DES(S, vorige output).

Is het hele bericht korter dan 8 symbolen dan versturen we

R + linker m bits van DES(S, initial chaining value).

Wil men geregeld berichten versturen die korter zijn dan 8 symbolen (bijvoorbeeld telkens 1 letter) dan is laatstgenoemde methode niet zo succesvol: in feite telt men telkens een constante vector op bij de te versturen boodschap, een uiterst primitief coderingssysteem!
Nu kan men de zogeheten "cipher feedback mode" overwegen; bij deze methode wordt uitgaande van een chaining value CV telkens

Z = R + linker m bits van DES(S, CV)

verstuurd, waarna CV vervangen wordt door de rechter 64-m bits van de oude CV gevolgd door de m cijferbits van Z.
Weer wordt een initial chaining value afgesproken.

Merk op dat als m klein is (bijv. m=1), de chaining value CV langzaam naar links schuift, zodat er hoge correlatie bestaat tussen de opeenvolgend gebruikte waarden van CV. Voorts, dat als er een fout bij transmissie optreedt alle volgende berichten niet meer te lezen zijn.
Bij een kanaal met ruis maakt men dan ook wel gebruik van de zogenaamde ´stream encipherment´. Hierbij wordt net als boven

Z = R + linker m bits van DES(S, CV)

verstuurd, maar de nieuwe chaining value wordt DES(S,CV) en hangt dus niet van R of Z af. Een fout bij de transmissie beinvloedt nu maar een boodschap, en in het speciale geval m=1 dus slechts 1 bit.

In alle bovenbeschreven voorbeelden is de sleutel S altijd vast geweest. Men kan natuurlijk ook schema´s verzinnen waarbij de sleutel verandert afhankelijk van het verzonden bericht.

One-way functions.

In sommige toepassingen is het nuttig om een ´scrambling´ functie f te hebben: een functie zodanig dat gegeven f(x) het ondoenlijk is om x terug te vinden. Bijvoorbeeld zijn in ouderwetse computersystemen de passwords die nodig zijn om toegang te krijgen tot zekere faciliteiten (login password, file protection password etc.) ergens in het systeem opgeborgen, zodat het systeem bij voorkomende gelegenheden kan controleren of het opgegeven password juist is. Dit heeft echter tot gevolg dat een operateur of privileged user of systeemprogrammeur vaak de mogelijkheid heeft achter deze passwords te komen. Nieuwere bedrijfssystemen gebruiken een ´one-way function´ f en slaan niet het password x maar f(x) op. Kennis van f(x) helpt niet om x te vinden, zodat het niet nodig is om f(x) geheim te houden. Voor het controleren van een aangeboden password y berekent het systeem f(y), vergelijkt dit met f(x) en accepteert precies dan wanneer f(x) = f(y).

Een van de manieren om een one-way function f te construeren is met behulp van DES: gebruik x als sleutel om een bekende (vaste) boodschap, bijvoorbeeld "CONSTANT" te versluieren, en noem het resultaat f(x). DES is juist zo ontworpen dat kennis van een paar (klare tekst, cijfertekst) het niet makkelijk maakt om de sleutel te vinden.

[Waarschuwing: bij de public key cryptosystems treft men ook one-way functions aan, maar van een speciaal soort: de ´trapdoor one-way functions´. Hierbij kan men gegeven f(x) niet x terugvinden tenzij men zekere extra ´trapdoor´ informatie kent. Als we de IBM mogen geloven zou DES een echte one-way function zjn, zonder valluik.]

Kritiek op DES.

Hoewel publicatie van het voorstel DES als standaard te accepteren tot een storm van kritiek leidde (vgl. [Diffie & Hellman], [Morris e.a.]), is het voorstel ongewijzigd aangenomen.
De twee belangrijkste punten van kritiek betreffen de sleutelgrootte en het niet openbaar maken van de ontwerpcriteria.

1. Sleutelgrootte.

In het oorspronkelijke ´Lucifer´ algoritme had de sleutel een lengte van 128 bits. In DES heeft de sleutel een lengte van slechts 56 bits (namelijk 64 bits waarbij 8 bits alleen controlebits zijn en niet door DES gebruikt worden).
Gegeven een paar (klare tekst, cijfertekst) kan men nu de sleutel proberen te vinden door uitputtend alle mogelijkheden te proberen. Weliswaar is het aantal mogelijkheden zeer groot (2 ** 56, ruim 70000000000000000) zodat bij het testen van een sleutel per microseconde men waarschijnlijk meer dan 1000 jaar op antwoord wacht, maar bij bouw van een special purpose computer met een miljoen chips die parallel sleutels testen heeft men gemiddeld binnen een halve dag de sleutel gevonden. Diffie en Hellman schatten dat de constructie van zo´n machine $20.000.000 zou kosten (zodat bij continu bedrijf het kraken per sleutel een paar duizend dollar zou kosten), de IBM schat dat het apparaat eerder tien keer zo duur zou uitvallen. Hoe dit ook zij, exhaustive search is nu bijna doenlijk (doenlijk voor de NSA, de National Security Agency, maar niet voor de meeste particulieren) en gezien de snelle voortgang van de technische ontwikkeling zal DES over een tiental jaren werkelijk niet meer tegen exhaustive search bestand zijn.

Een sleutellengte van 128 bits zou daarentegen volledig afdoende zijn, ook nog over een paar honderd jaar. (Opmerkelijk is dat de NSA weigert exportvergunning te verlenen aan cryptosystemen waarbij een sleutel van meer dan 64 bits kan worden gebruikt.)

2. De S-dozen.

Het DES algoritme bestaat uit een 16-voudige iteratie van permutatie gevolgd door niet-lineaire substitutie (zie de appendix). Deze substituties worden beschreven door de zgn. ´S-boxes´. De criteria bij de keuze van de S-dozen zijn geheim.
We citeren Morris e.a.:

> The method by which the S-boxes were chosen is at present classified. A representative from IBM claimed that this choice of the S-boxes actually increased the security of the cipher. Since the design is classified he was unable to offer any proof of this. (His words were: "You must trust us, we are all good Boy Scouts".) While there is no particular reason to doubt this, some doubt remains. There are enough examples of innocent looking algorithms which later turned out to contain so-called ´trapdoors´ or ´Trojan horses´ to indicate caution. The only serious cryptographic study of the DES that has been made public, a study by Lexar Corp. [], pointed out a number of regularities in the structure of the S-boxes.

Ook Hellman e.a. [] stelden met behulp van statistische tests vast dat de S-dozen niet toevallig gekozen zijn, maar een (onbekende) structuur hebben.
Op deze kritiek reageerde de NSA met de volgende (grotendeels al bekende) informatie:
- geen S-doos is een lineaire of affiene functie
- bij verandering van 1 input bit van een S-doos veranderen ten minste 2 output bits
- de S-dozen zijn zo gekozen dat wanneer een vast input bit constant wordt gehouden het verschil tussen het aantal nullen en enen in de output geminimaliseerd wordt.

Voorts verdedigt men DES door op te merken dat 5 jaar na publicatie er nog geen ontcijfering gevonden is (althans niet gepubliceerd is).

REFERENTIES

W. Diffie & M.E. Hellman,
A critique of the proposed Data Encryption Standard,
Communications ACM 19 (1976) 164-165.

W. Diffie & M.E. Hellman,
Cryptanalysis of the NBS Data Encryption Standard,
NTIS, May 1976 (Report No. Mem-76-2).

M.E. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie,
S. Pohlig & P. Schneider,
Results of an initial attempt to cryptanalyze the NBS
Data Encryption Standard,
Stanford Univ., Nov. 1976, SEI 76-042.

M.E. Hellman,
DES will be totally insecure within ten years,
IEEE Spectrum 16 (1979) 32-39.

I. Ingemarsson,
Encryption in data networks with applications to teletex,
Linköping Univ., 1978, Internskrift LiTH-ISY-0235.

Lexar Corp.,
An evaluation of the NBS Data Encryption Standard,
Report Lexar Corp., 11611 San Vicente Blvd., Los Angeles 1976.

Robert Morris, N.J.A. Sloane & A.D. Wyner,
Assessment of the National Bureau of Standards Proposed
Federal Data Encryption Standard,
Cryptologia 1 (1977) 281-306.

National Bureau of Standards,
Notice of a proposed Federal Information Processing Data Encryption
Standard, Federal Register 40 (No. 12607, Aug 1, 1975).

J.L. Smith,
The design of Lucifer, a cryptographic device for data communications,
Research Report RC-3326, IBM, 1971.

J.L. Smith,
Recirculating block cipher cryptographic system,
U.S. Patent No. 3796830, March 12, 1974.

$

APPENDIX I

PROPOSED

FEDERAL INFORMATION PROCESSING DATA ENCRYPTION STANDARD

AS PUBLISHED IN

FEDERAL REGISTER

OF

AUGUST 1, 1975

NOTICE OF A PROPOSED FEDERAL INFORMATION PROCESSING DATA ENCRYPTION STANDARD

Under the provisions of Public Law 89-306 and Executive Order 11717, the Secretary of Commerce is authorized to establish uniform Federal ADP Standards. A proposed standard for computer data encryption is being recommended for Federal use. This proposed standard specifies a mathematical algorithm for encrypting (enciphering) and decrypting (deciphering) binary coded information. Encrypting converts data to an unintelligible form called cipher. Decrypting converts the cipher back to the original data.

Because certain communicated and stored data can have significant value or sensitivity, the need for adequate protection of these data from theft and misuse has become a national issue. It is generally recognized that cryptography is an effective means of protecting data, provided that encryption techniques of adequate strength are devised, validated and integrated into a system. In order to insure compatibility of cryptographically protected data, it is necessary to establish a Data Encryption Standard and develop guidelines for its implementation and use.

Solicitations for computer data encryption algorithms were published by NBS in the Federal Register issues of May 15, 1973 (38FR12763) and of August 27, 1974 (39FR30961). An algorithm was received in response to these submissions that satisfies the primary technical requirements for the algorithm of a Data Encryption Standard. This algorithm was published for comment in the Federal Register issue of March 17, 1975 (40FR12134) and is contained in the specification section of this proposed standard.

In order to ensure that all parties have a full opportunity to present their views, NBS is soliciting comments on the following Data Encryption Standard. Readers should be aware that cryptographic devices and technical data relating to them may come under the export controls of Title 22, Code of Federal Regulations, Parts 121 through 128. Readers should also be aware that certain U.S. and foreign patents contain claims which may cover implementation and use of this algorithm. In this connection, the reader should note the references in the proposed standard.

The proposed Federal Information Processing Standard contains two basic sections: (1) an announcement section which provides information concerning the applicability, implementation, and maintenance of the standard; and (2) a specification section which deals with the technical requirements of the standard. Both sections are provided in their entirety in this notice.

Interested parties may submit comments to the Associate Director for ADP Standards, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D.C. 20234, within 90 days after publication of this notice in the Federal Register.

Federal Information
Processing Standards Publication

Date _____

ANNOUNCING THE

DATA ENCRYPTION STANDARD

Federal Information Processing Standards Publications are issued by the National Bureau of Standards pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat 1127) as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973), and Part 6 of Title 15 CFR (Code of Federal Regulations).

Name of Standard. Data Encryption Standard (DES)

Category of Standard. ADP Operations, Computer Security.

Explanation: This Data Encryption Standard specifies an algorithm for the cryptographic protection of computer data. This publication provides a complete description of a mathematical algorithm for encrypting (enciphering) and decrypting (deciphering) binary coded information. Encrypting converts data to an unintelligible form called cipher. Decrypting converts the cipher back to the original data. Both of these mathematical transformations are based on a single binary variable called the key.

Data may be protected against unauthorized disclosure by generating a random key and issuing it to the authorized users of the data. The cipher that has been produced by performing the steps of the encryption algorithm on data using a particular key can only be returned to the original data by use of the decryption algorithm using the identical key. Unauthorized recipients of the cipher who may have the algorithm but do not have this key cannot derive the original data. A standard algorithm based on a user-generated key thus provides a basis for compatible cryptographic protection of computer data while preventing unauthorized use of the data in cipher form.

Approving Authority. Secretary of Commerce.

Maintenance Agency. Institute for Computer Sciences and Technology, National Bureau of Standards.

Applicability. The Data Encryption Standard will be used by Federal agencies for protecting unclassified computer data when the responsible authority for the data or the computer systems of that agency has stipulated that cryptographic protection is required. Data that is considered sensitive by the responsible authority or data which has a high value or represents a high value should be cryptographically protected if it is vulnerable to unauthorized disclosure or undetected modification during transmission or in dormant storage. During transmission data may be encrypted at a terminal and the resulting cipher

transmitted. Data may also be encrypted before it is written as cipher onto a storage device (magnetic tape, removable disk pack, etc.) which may be removed and read by unauthorized personnel. However, cipher must be decrypted before it can be processed. Data stored in cipher form can only be read if the key used to encrypt it is stored until the data is to be read and used. This standard is not applicable for the cryptographic protection of computer data that is classified according to the National Security Act of 1947 or the Atomic Energy Act of 1954, as amended. Provisions of these Acts and their implementing regulations specify the means for protecting classified data.

Implementation. This standard becomes effective six months from the date of its publication following approval by the Secretary of Commerce. As new ADP systems and networks are developed and current systems are improved, Federal agencies, based upon their specific data protection requirements, should develop and implement regulations for the use of this standard. These regulations should specify when and where data encryption should be used and include administrative procedures for using it in a computer system or network. Instructions for procuring data processing equipment utilizing the DES will be provided by the General Services Administration.

The algorithm specified in this Data Encryption Standard is to be implemented in special purpose hardware when used by Federal agencies. An electronic device which performs the mathematical steps of the algorithm may comprise one or more Large Scale Integration (LSI) "chips" in a single electronic package. An alternate implementation may consist of many Medium Scale Integration (MSI) electronic packages. Developing technologies may allow the effective and efficient performance of the algorithm in other electronic devices (e.g., micro-computers with Read Only Memories) which are dedicated to performing the operations of the algorithm. Only hardware implementations of the algorithm which can be tested and certified as being accurate will be considered as complying with the standard. Such devices must also conform to the export controls of Title 22, Code of Federal Regulations, Parts 121 through 128. These regulations specify that cryptographic devices or cryptographic information are controlled if intended for export.

Cryptographic devices implementing this standard may be covered by U. S. and foreign patents held by the International Business Machines Corporation, which has agreed to grant nonexclusive, royalty-free licenses under the patents to make, use and sell apparatus which complies with the standard. The terms, conditions and scope of the licenses are set out in a notice published in the May 13, 1975 issue of the Official Gazette of the United States Patent and Trademark Office (934 O. G. 452).

Specifications. Federal Information Processing Standard (FIPS) Data Encryption Standard (affixed).

Cross Index:

a. FIPS PUB 31, "Guidelines to ADP Physical Security and Risk Management"

b. FIPS PUB 41, "Computer Security Guidelines for Implementing the Privacy Act of 1974"

Qualifications. This Data Encryption Standard specifies an algorithm which may be utilized in many applications and environments. A device which performs the algorithm may be used as a fundamental building block in applications areas where cryptographic protection is needed. Implementation of a cryptographic system comprising many cryptographic devices located in computer terminals, computer "front end" communications processors and computer storage device data channels is a complex task. Guidelines for implementing and using data encyrption devices will be provided by NBS. A series of technical notes describing alternative ways of using data encryption devices will be produced. For example, the algorithm may be used both directly as an encryptor of blocks of data and indirectly as a binary stream generator which may be combined with the data to produce the cipher. The cipher produced with the latter technique has the same high degree of cryptographic protection as the cipher produced if the data were entered directly into the data encryption device. In either case, the cryptographic device must be properly interfaced to the other system components in the application area. When properly implemented and used, Government agencies may rely on the DES to provide a high level of cryptographic protection to valuable and sensitive information. NBS, supported by the technical assistance of appropriate Government agencies, has determined that the algorithm in this Data Encryption Standard can provide this level of protection beyond the normal life cycle of its associated ADP equipment.

Comments and suggestions regarding the use of this standard are welcomed and should be addressed to the Associate Director for ADP Standards, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D. C. 20234.

Waiver Procedure. The head of a Federal agency may waive the provisions of this FIPS PUB upon proper justification and upon coordination with the National Bureau of Standards. A waiver is not necessary unless cryptographic protection is required for unclassified computer data and either a different encryption algorithm is to be used or a software implementation of this algorithm is needed. Letters describing the nature of, and reasons for, the waiver should be addressed to the Associate Director for ADP Standards, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D. C. 20234.

Sixty days should be allowed for review and response by NBS. The waiver is not to be made until a reply from NBS is received; however, the final decision for granting the waiver is the responsibility of the agency head.

Where to Obtain Copies of the Standard.

a. Copies of this publication are for sale by the Superintendent of Documents, U. S. Government Printing Office, Washington, D. C. 20402 (___ per copy; SD Catalog Number C _____). There is a 25 percent discount on quantities of 100 or more. When ordering, specify document

number, title, and SD Catalog Number. Payment may be made by check, money order, coupons, or deposit account.

b. Microfiche copies of this publication are available from the National Technical Information Service, U. S. Department of Commerce, Springfield, Virginia 22161. When ordering, refer to Report Number NBS-FIPS-PUB-____ and title. The cost is ____ per copy and payment may be made by check, money order, coupons or deposit account.

Federal Information
Processing Standards Publication

Date _____

SPECIFICATIONS FOR THE
DATA ENCRYPTION STANDARD

The Data Encryption Standard (DES) shall consist of the following Data Encryption Algorithm implemented in a special purpose electronic device. This device shall be designed in such a way that it may be embedded in a computer system or network and provide cryptographic protection to binary coded data. The method of implementation, the control of the cryptographic device and the interface of the device to its associated equipment will depend on the application and environment. The device shall be designed and implemented in such a way that it may be tested and validated as accurately performing the transformations specified in the following algorithm. Certification of compliance with this standard is the responsibility of the designer and manufacturer of the device.

DATA ENCRYPTION ALGORITHM

Introduction

The algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64 bit key. Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process. A block to be enciphered is subjected to an initial permutation IP, then to a complex key-dependent computation and finally to a permutation which is the inverse of the initial permutation IP^{-1}. The key-dependent computation can be simply defined in terms of a function f, called the cipher function, and a function KS, called the key schedule. A description of the computation is given first, along with details as to how the algorithm is used for encipherment. Next, the use of the algorithm for decipherment is described. Finally, a definition of the cipher function f is given in terms of primitive functions which are called the selection functions S_i and the permutation function P. S_i, P and KS of the algorithm are contained in the Appendix.

The following notation is convenient: Given two blocks L and R of bits, LR denotes the block consisting of the bits of L followed by the bits of R. Since concatenation is associative

$B_1B_2...B_8$, for example, denotes the block consisting of the bits of B_1 followed by the bits of B_2... followed by the bits of B_8.

Enciphering

A sketch of the enciphering computation is given in Figure 1.

The 64 bits of the input block to be enciphered are first subjected to the following permutation, called the initial permutation IP:

IP

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

That is the permuted input has bit 58 of the input as its first bit, bit 50 as its second bit, and so on with bit 7 as its last bit. The permuted input block is then the input to a complex key-dependent computation described below. The output of that computation, called the preoutput, is then subjected to the following permutation which is the inverse of the initial permutation:

ENCIPHERING COMPUTATION

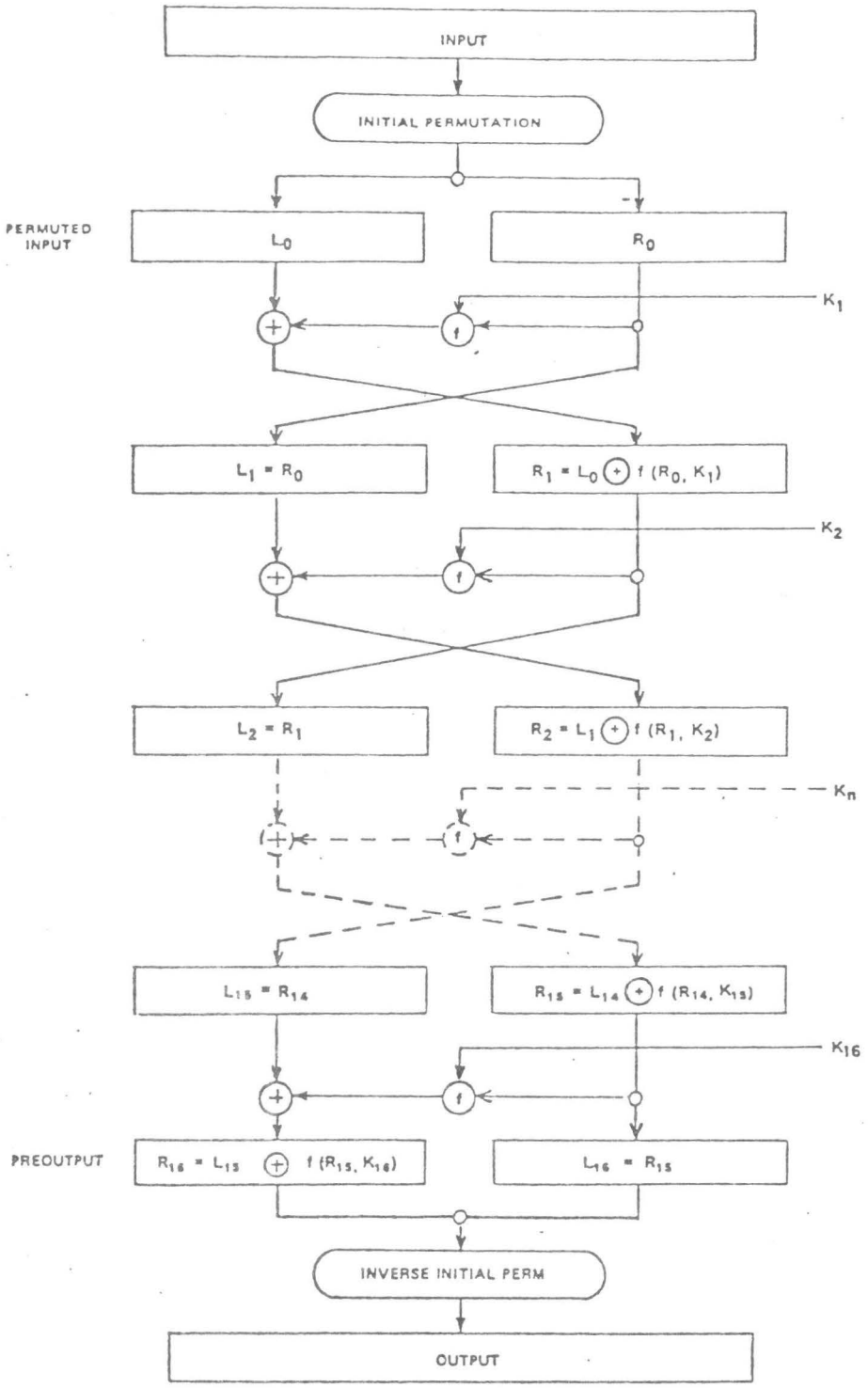


FIGURE 1

IP^{-1}

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

That is, the output of the algorithm has bit 40 of the preoutput block as its first bit, bit 8 as its second bit, and so on, until bit 25 of the preoutput block is the last bit of the output.

The computation which uses the permuted input block as its input to produce the preoutput block consists, but for a final interchange of blocks, of 16 iterations of a calculation that is described below in terms of the cipher function f which operates on two blocks, one of 32 bits and one of 48 bits, and produces a block of 32 bits.

Let the 64 bits of the input block to an iteration consist of a 32 bit block L followed by a 32 bit block R. Using the notation defined in the introduction, the input block is then LR.

Let K be a block of 48 bits chosen from the 64 bit key. Then the output L'R' of an iteration with input LR is defined by:

$$(1) \qquad L' = R$$
$$R' = L \oplus f(R,K)$$

where $\oplus$ denotes bit-by-bit addition modulo 2.

As remarked before, the input of the first iteration of the calculation is the permuted input block. If L'R' is the output of the 16th iteration then R'L' is the preoutput block. At each iteration a different block K of key bits is chosen from the 64 bit key designated by KEY.

With more notation we can describe the iterations of the computation in more detail. Let KS be a function which takes an integer n in the range from 1 to 16 and a 64 bit block KEY as input and yields as output a 48 bit block K_n which is a permuted selection of bits from KEY. That is

$$(2) \qquad K_n = KS(n,KEY)$$

with K_n determined by the bits in 48 distinct bit positions of KEY. KS is called the key schedule because the block K used in the n'th iteration of (1) is the block K_n determined by (2).

As before, let the permuted input block be LR. Finally, let L_o and R_o be respectively L and R and let L_n and R_n be respectively L' and R' of (1) when L and R are respectively L_{n-1} and R_{n-1} and K is K_n; that is, when n is in the range from 1 to 16,

$$(3) \qquad L_n = R_{n-1}$$
$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

The preoutput block is then $R_{16}L_{16}$.

The key schedule KS of the algorithm is described in detail in the Appendix. The key schedule produces the 16 K_n which are required for the algorithm.

Deciphering

The permutation IP^{-1} applied to the preoutput block is the inverse of the initial permutation IP applied to the input. Further, from (1) it follows that:

$$(4) \qquad R = L'$$
$$L = R' \oplus f(L', K)$$

Consequently, to decipher it is only necessary to apply the very same algorithm to an enciphered message block, taking care that at each iteration of the computation the same block of key bits K is used during decipherment as was used during the encipherment of the block. Using the notation of the previous section, this can be expressed by the equations:

$$(5) \qquad \begin{aligned} R_{n-1} &= L_n \\ L_{n-1} &= R_n \oplus f(L_n, K_n) \end{aligned}$$

where now $R_{16}L_{16}$ is the permuted input block for the deciphering calculation and L_0R_0 is the preoutput block. That is, for the decipherment calculation with $R_{16}L_{16}$ as the permuted input, K_{16} is used in the first iteration, K_{15} in the second, and so on, with K_1 used in the 16th iteration.

The Cipher Function f

A sketch of the calculation of f(R,K) is given in Figure 2.

CALCULATION OF f(R,K)

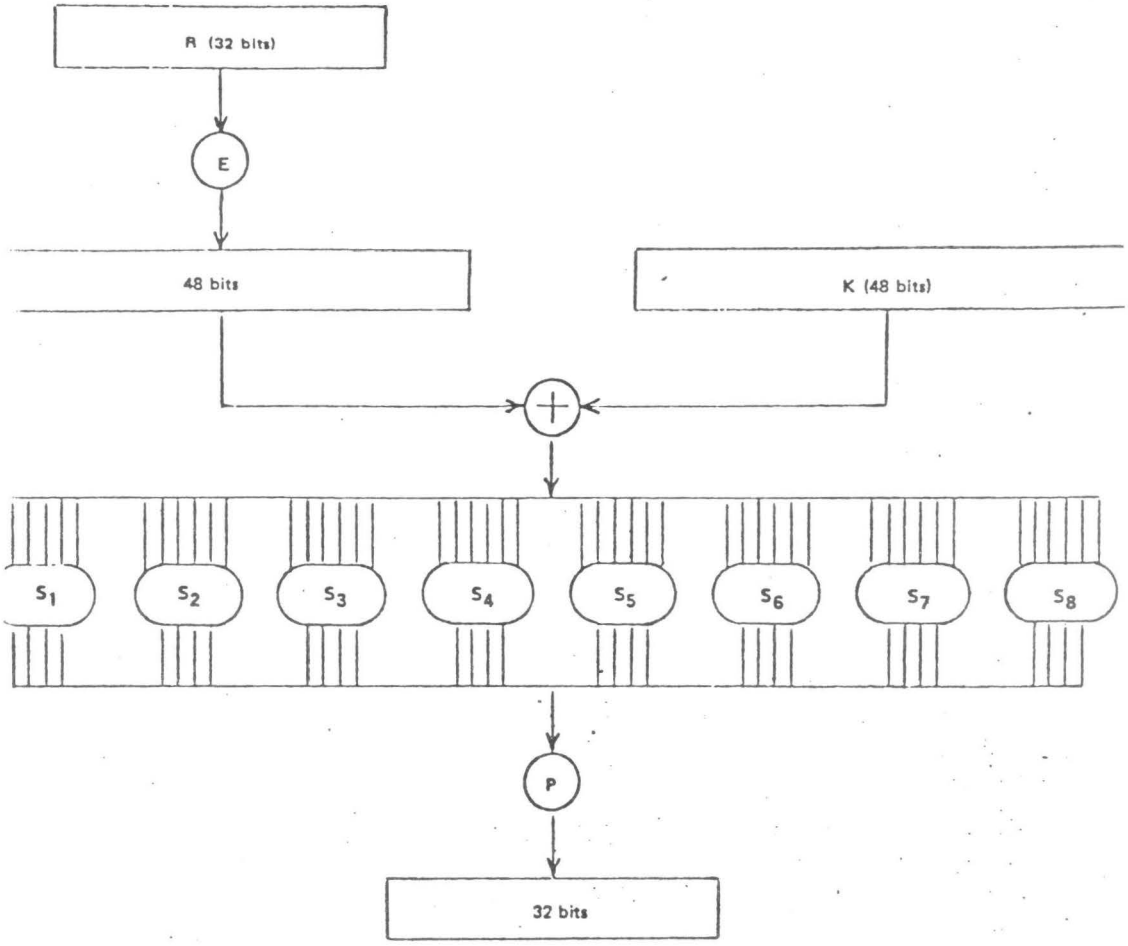


Figure 2

Let E denote a function which takes a block of 32 bits as input and yields a block of 48 bits as output. Let E be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

E BIT-SELECTION TABLE

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

Thus the first three bits of E(R) are the bits in positions 32, 1 and 2 of R while the last 2 bits of E(R) are the bits in positions 32 and 1.

Each of the unique selection functions S_1, S_2, ..., S_8, takes a 6 bit block as input and yields a 4 bit block as output and is illustrated by using a table containing the recommended S_1:

S_1

Column Number

Row No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

If S_1 is the function defined in this table and B is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of B represent in base 2 a number in the range 0 to 3. Let that number be i. The middle 4 bits of B represent in base 2 a number in the range 0 to 15. Let that number be j. Look up in the table the number in the i'th row and j'th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of S_1 for the input B. For example, for input 011011 the row is 01, that is row 1, and the column is determined by 1101, that is column 13. In row 1 column 13 appears 5 so that the output is 0101. Selection functions S_1, $S_2,\ldots,S_8$ of the algorithm appear in the Appendix.

The permutation function P yields a 32 bit output from a 32 bit input by permuting the bits of the input block. Such a function is defined by the following table:

P

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

The output P(L) for the function P defined by this table is obtained from the input L by taking the 16th bit of L as the first bit of P(L), the 7th bit as the second bit of P(L), and so on until the 25th bit of L is taken as the 32nd bit of P(L). The permutation function P of the algorithm is repeated in the Appendix.

Now let $S_1,\ldots,S_8$ be eight distinct selection functions, let P be the permutation function and let E be the function defined above.

To define f(R,K) we first define $B_1,\ldots,B_8$ to be blocks of 6 bits each for which

$$(6) \qquad B_1B_2\ldots B_8 = K \oplus E(R)$$

The block f(R,K) is then defined to be

$$(7) \qquad P(S_1(B_1)S_2(B_2)\ldots S_8(B_8))$$

Thus $K \oplus E(R)$ is first divided into the 8 blocks as indicated in (6). Then each B_i is taken as an input to S_i and the 8 blocks $S_1(B_1), S_2(B_2), \ldots, S_8(B_8)$ of 4 bits each are consolidated into a single block of 32 bits which forms the input to P. The output (7) is then the output of the function f for the inputs R and K.

APPENDIX

PRIMITIVE FUNCTIONS FOR THE DATA ENCRYPTION ALGORITHM

The choice of the primitive functions KS, S_1, ..., S_8 and P is critical to the strength of an encipherment resulting from the algorithm. Specified below is the recommended set of functions, describing S_1, ..., S_8 and P in the same way they are described in the algorithm. For the interpretation of the tables describing these functions, see the discussion in the body of the algorithm.

The primitive functions S_1,...,S_8, are:

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The primitive function P is:

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

Recall that K_n, for $1 \leq n \leq 16$, is the block of 48 bits in (2) of the algorithm. Hence, to describe KS, it is sufficient to describe the calculation of K_n from KEY for n = 1, 2, ..., 16. That calculation is illustrated in Figure 3. To complete the definition of KS it is therefore sufficient to describe the two permuted choices, as well as the schedule of left shifts. One bit in each eight-bit byte of the KEY may be utilized for error detection in key generation, distribution and storage. Bits 8, 16, ..., 64 are for use in assuring that each byte is of odd parity.

KEY SCHEDULE CALCULATION

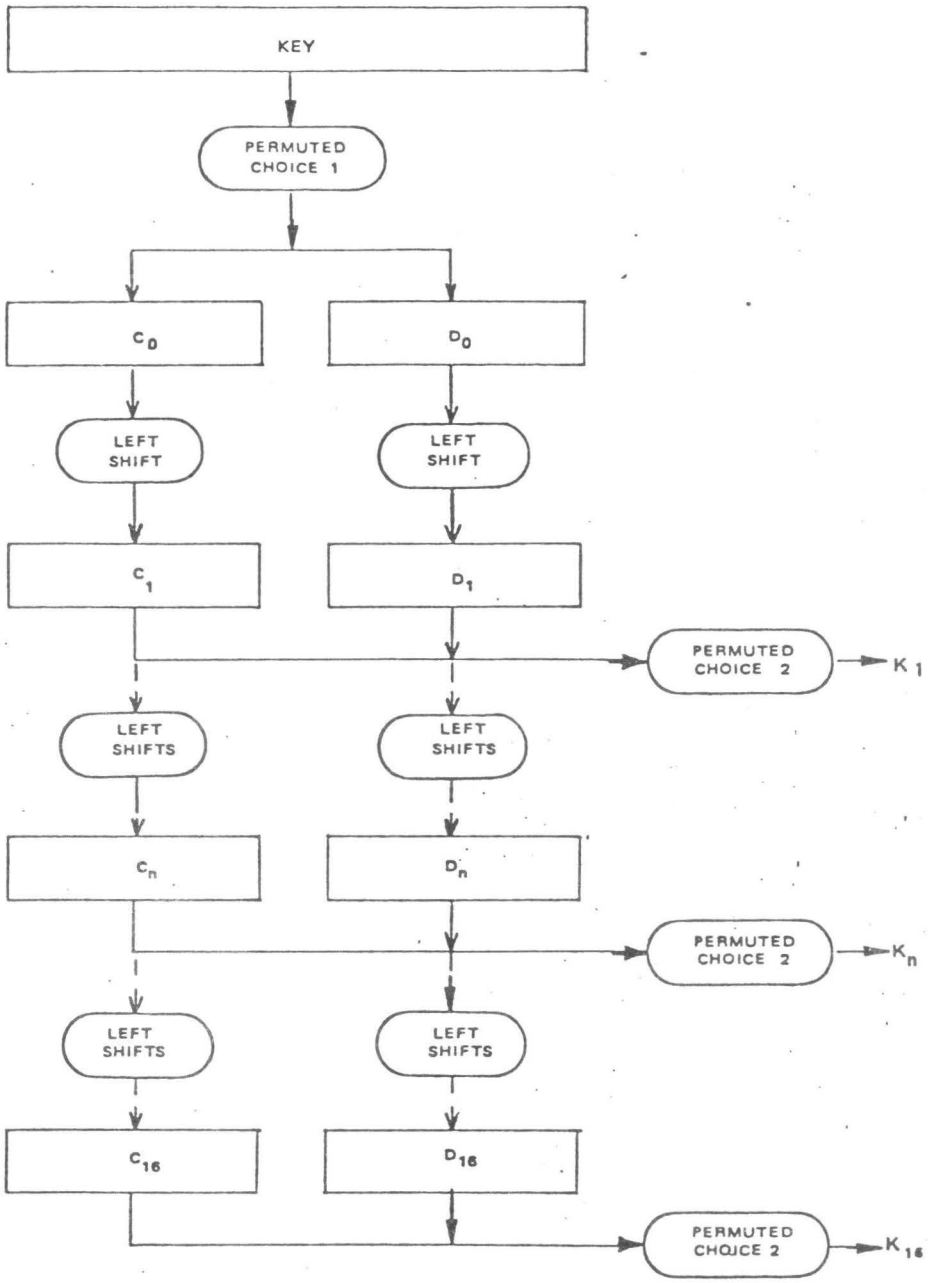


Figure 3

Permuted choice 1 is determined by the following table:

PC-1

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

The table has been divided into two parts, with the first part determining how the bits of C_o are chosen, and the second part determining how the bits of D_o are chosen. The bits of KEY are numbered 1 through 64. The bits of C_o are respectively bits 57, 49, 41, ..., 44 and 36 of KEY, with the bits of D_o being bits 63, 55, 47, ..., 12 and 4 of KEY.

With C_o and D_o defined, we now define how the blocks C_n and D_n are obtained from the blocks C_{n-1} and D_{n-1}, respectively, for $n = 1, 2, ..., 16$. That is accomplished by adhering to the following schedule of left shifts of the individual blocks:

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

For example, C_3 and D_3 are obtained from C_2 and D_2, respectively, by two left shifts, and C_{16} and D_{16} are obtained from C_{15} and D_{15}, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3, ..., 28, 1.

Permuted choice 2 is determined by the following table:

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

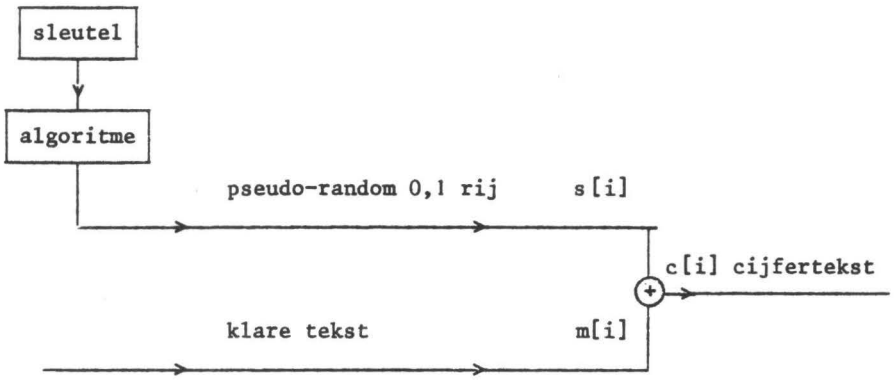
Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on with the 47th bit the 29th, and the 48th bit the 32nd.

VII. SHIFT REGISTER SEQUENCES

7.1. Inleiding

In een vorig hoofdstuk hebben we gezien hoe het z.g. "one time pad" absolute veiligheid geeft, maar in vele situaties voor practische doeleinden ongeschikt is. Het "one time pad" kan worden gezien als een Vigenère cijfer met een sleutel die oneindig lang is en die tevens op een volstrekt willekeurige manier is samengesteld.
Het is dan ook niet zo verwonderlijk dat men cijfersystemen heeft ontwikkeld van het Vigenère type met een sleutel die een zeer lange periode heeft, gemakkelijk te maken is en er redelijk "random" uitziet [2], [4].

Vanwege het voornaamste toepassingsgebied - electronische informatieoverdracht - zullen we ons beperken tot het alfabet $\{0,1\}$. Dit geeft aanleiding tot het volgende diagram



Het is nodig dat de lezer zich realiseert dat de s[i] rij op den duur periodiek is. Hij wordt immers voortgebracht door een algoritme, dat werkt op een eindige geheugenruimte. Vroeg of laat zal het algoritme dezelfde

toestand van de geheugenruimte zien. Vanaf dat ogenblik treedt periodiciteit op.

DEFINITIE 7.1. De <u>periode</u> van een rij s[i] is het kleinste getal p zodat er een getal d is (delay geheten) met de eigenschap dat s[i] = s[i+p] voor alle i ≥ d.
Als d > 0 noemen we de rij <u>op den duur periodiek</u>, terwijl als d = 0 de rij <u>periodiek</u> heet.

Intuïtief willen we dat de rij s[i] er zo random mogelijk uitziet. In 1967 formeerde Golomb [2] drie eisen, waaraan een pseudo-random rij moet voldoen om zo te mogen heten. Om deze eisen te kunnen formuleren moeten we eerst enige nieuwe begrippen introduceren.

DEFINITIE 7.2. Een <u>run</u> van lengte k is een serie van k opeenvolgende enen (nullen) voorafgegaan en gevolgd door een nul (resp. een).

DEFINITIE 7.3. De <u>autocorrelatie</u> van een periodieke rij s[i] met periode p, genoteerd met AC(k), is (A-D)/p, waarbij A (en D) het aantal keren is dat s[i] en s i+k hetzelfde (resp. verschillend) zijn voor 0 ≤ i < p.
Als k niet deelbaar is door p spreekt men van "out-of-phase autocorrelation".

We kunnen nu Golomb's eisen formuleren.
RANDOMNESS POSTULATEN (Golomb)
R1 Het aantal nullen en enen per periode is zo gelijk mogelijk.
R2 Per periode heeft de helft van het aantal runs lengte 1, een kwart heeft lengte 2, een achtste heeft lengte 3, etc.

R3 AC(k) is constant (en dus laag) voor $0 < k < p$.

In de literatuur ([2],[3]) kan men allerlei testen vinden, die met een bepaalde nauwkeurigheid aangeven of een gegeven periodieke rij voldoet aan de postulaten.

Voor onze cryptografische doeleinden moet de pseudo-random rij s[i] ook nog voldoen aan drie andere eisen.

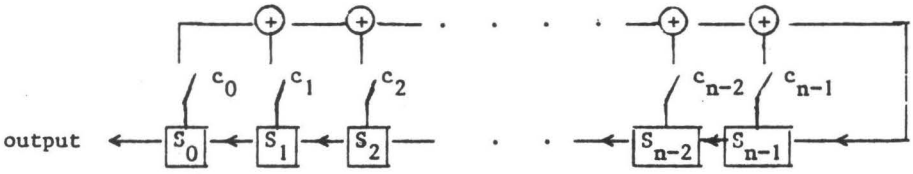
C1 De periode moet lang zijn (b.v. 10^{50}).

C2 De rij s[i] moet tamelijk simpel te genereren zijn.

C3 Kennis van een beperkt stuk klare tekst met de corresponderende cij-cijfertekst mag de cryptanalist niet in staat stellen de gehele s[i] rij te bepalen.

7.2. Lineaire shift-registers

Bij het genereren van pseudo-random rijen wordt vaak gebruik gemaakt van lineaire shift-registers. Dit komt omdat dit soort registers gemakkelijk te maken zijn en omdat de door hun gegenereerde rijen goed te analyseren zijn.



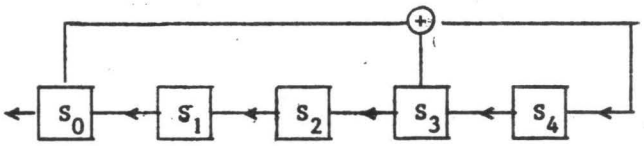
lineair shift-register

In bovenstaande figuur stelt iedere S_i een geheugenelement voor. Deze n geheugenelementen bepalen samen de toestand van het shift-register.

De schakelaars c_i zijn open als $c_i = 0$ en gesloten als $c_i = 1$. Op bepaalde momenten, aangegeven door een moederklok, verschuift de inhoud van de geheugenelementen zoals in de figuur aangegeven. Om preciezer te zijn laat s_i de inhoud zijn van S_i op een bepaald tijdstip, i.e. het register is in toestand $(s_0, s_1, s_2, \ldots, s_{n-2}, s_{n-1})$ dan is de toestand op het volgende opgenblik $(s_1, s_2, s_3, \ldots, s_{n-1}, s_n)$ met output s_0, waarbij

(7.1) $$s_n = c_0 * s_0 + c_1 * s_1 + \ldots + c_{n-1} * s_{n-1} \bmod 2.$$

Ter verduidelijking van het bovenstaande geven we een voorbeeld.



Voorbeeld van een lineair shift-register

t=0	01010	t=8	01100	t=16	11100	t=24	10010
1	10101	9	11000	17	11001	25	00100
2	01011	10	10001	18	10011	26	01000
3	10111	11	00011	19	00110	27	10000
4	01110	12	00111	10	01101	28	00001
5	11101	13	01111	21	11010	29	00010
6	11011	14	11111	22	10100	30	00101
7	10110	15	11110	23	01001	31	01010

De eerste 32 toestanden van bovenstaand voorbeeld.

Als $c_0 = 0$ komt ons shif-register met n geheugenelementen overeen met

een shift-register die maar n-1 geheugenelementen heeft (en tevens de output met een tijdseenheid vertraagt). Aangezien dat niet erg zinvol is nemen we vanaf nu aan dat altijd $c_0 = 1$. Met behulp van (7.1) is nu de volgende stelling gemakkelijk in te zien.

STELLING 7.4. Iedere toestand van een lineair shift-register heeft een unieke opvolger en een unieke voorganger.

Op grond van deze stelling kunnen we concluderen dat de toestanden van het shift register elkaar cyclisch opvolgen. Aangezien de (0,0,...,0) toestand in zichzelf overgaat en er 2^n verschillende toestanden zijn, kunnen we concluderen dat de periode van elke cyclus hooguit 2^n-1 is.

STELLING 7.5. De uitvoer van een lineair shift register met n geheugenelementen is een periodieke rij met periode kleiner of gelijk 2^n-1.

DEFINITIE 7.6. De uitvoerrij van een lineair shift register met n geheugenelementen heet een <u>pseudo-noise rij</u> (PN-rij) als hij periode 2^n-1 heeft.

Duidelijk is dat bij het genereren van een PN-rij het shift-register alle niet-nul toestanden cyclisch doorloopt en andersom dat elke niet-nul begintoestand een PN-rij voortbrengt die een verschuiving is van de oorspronkelijke PN-rij.
Zoals de naam suggereert en zoals we later zullen aantonen voldoen PN-rijen vrij goed aan de eisen R1 - R3, Het is dus van belang te bepalen wanneer ze voorkomen. Uit (7.1) blijkt dat de uitvoerrij (s_i) voldoet aan een lineaire betrekking. Met deze lineaire betrekking is een z.g.

karakteristiek polynoom geassocieerd nl.

$$f(x) = c_0 + c_1x + c_2x^2 + \ldots + c_{n-1}x^{n-1} + x^n .$$

Vanaf nu rekenen we enkel met polynomen die coëfficiënten in het lichaam $\{0,1\}$ hebben (binaire polynomen geheten).

DEFINITIE 7.7. Een polynoom f heet ontbindbaar als het het product is van twee polynomen met positieve graad. Zo niet, dan heet f onontbindbaar.

DEFINITIE 7.8. De exponent van een polynoom f met $f(0) \neq 0$ is het kleinste getal e met f(x) deelt x^e-1.

Dat zo'n getal e bestaat volgt vrij simpel als men opeenvolgende machten van x modulo f(x) beschouwt. Daar $f(0) \neq 0$ zal x^i modulo f(x) nooit 0 opleveren. Hieruit volgt dat de exponent van een binair polynoom ten hoogste 2^n-1 is.

DEFINITIE 7.9. Een binair polynoom van de graad n heet primitief als het onontbindbaar is en exponent 2^n-1 heeft.

Jammer genoeg voert het hier te ver om de fraaie theorie van lineaire shift-registers uitvoeriger te behandelen. Zonder bewijs poneren we dan ook de volgende stelling, die aangeeft op welke manier men alle PN-rijen kan voortbrengen.

STELLING 7.10. Zij f het karakteristieke polynoom van een lineair shift-register en zij (s_i) een niet-nul uitvoerrij. Dan geldt dat (s_i) een PN-rij is dan en slechts dan als f een primitief polynoom is.

In de literatuur kan men gemakkelijk preciese formules vinden voor het aantal primitieve polynomen van de graad n. Het is nu voldoende te weten dat dit aantal ongeveer $2^n/n$ is. Er is dus keuze genoeg!

We zullen nu nagaan in welke mate PN-rijen aan de voorwaarden R1 - R3 voldoen.

LEMMA 7.11. PN-rijen bevatten 2^{n-1} enen en $2^{n-1}-1$ nullen per periode.
Bewijs. Per periode komen al de mogelijke toestanden precies eenmaal voor behalve de nultoestand. Precies 2^{n-1} hiervan beginnen met 1. De andere beginnen met een 0. □

LEMMA 7.12. Het aantal runs van lengte r is 2^{n-r-1} voor $1 \leq r \leq n-2$ en 1 voor $r = n-1$ en $r = n$.
Bewijs. Deze beweringen volgen allen uit het feit dat alle niet-nul toestanden voorkomen en dat een run van lengte r correspondeert met één van de toestanden 011...10**..* of 100..01**..* . Verder moet men zich realiseren dat de toestand 011...1 gevolgd moet worden door de toestand 11...1 . □

LEMMA 7.13. De autocorrelatie AC(k) met $0 < k < 2^n-1$ is $-1/(2^n-1)$.
Bewijs. De somrij $(s_i + s_{i+k})$ van de twee rijen (s_i) en (s_{i+k}) voldoet ook aan (7.1) en is dus ook een PN-rij. Een 1 (resp. 0) hierin correspondeert met een positie waar de rijen (s_i) en (s_{i+k}) verschillen (resp. overeenstemmen). De bewering volgt nu uit Lemma 7.11. □

Hiermee voldoen PN-rijen in voldoende mate aan R1 - R3. Duidelijk is dat ze ook voldoen aan C1 en C2. Echter aan C3 voldoen ze niet! Immers, als we voldoende klare tekst hebben met de corresponderende cijfertekst, kunnen we deze modulo 2 optellen en vinden we het corresponderende stuk van de PN-rij . We kunnen daarna de volgende stelling toepassen.

STELLING 7.14. Zij (s_i) een niet-nul PN-rij voortgebracht door een lineair shift-register met karakteristiek polynoom f van de graad n en neem aan dat s_i bekend is voor 2n opeenvolgende woorden van i, zeg $r \leq i \leq r+2n-1$. Dan is f uniek bepaald door een matrixvergelijking (en de rest van de (s_i) rij is nu ook gemakkelijk te bepalen).

<u>Bewijs</u>. Laat $f = c_0 + c_1x + c_2x^2 + \ldots + c_{n-2}x^{n-2} + x^{n-1}$. Dan geldt vanwege (7.1) de matrixvergelijking

(7.2) $\quad \underline{s}_{r+n} = \underline{c}S$

waarbij $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ en $\underline{s}_i = (s_i, s_{i+1}, \ldots, s_{i+n-1})$ en

(7.3) $$S = \left(\underline{s}_r^t, s_{r+1}^t, \ldots, \underline{s}_{r+n-1}^t\right) .$$

Aangezien elke toestand van het shift register een lineaire combinatie is van de rijen van S geldt dat de rijen lineair onafhankelijk zijn. Met andere woorden S is een reguliere matrix die dus een inverse heeft. We kunnen dus $\underline{c}$ heel eenvoudig met (7.3) bepalen. De rest van de (s_i) rij volgt nu uit (7.1). □

Uit het bovenstaande volgt dat we PN-rijen niet zonder meer voor cryptografische doeleinden gebruiken. Aangezien ze echter zo simpel te maken en zo goed te analyseren zijn, worden ze echter wel vaak als bouwstenen

gebruikt in ingewikkelder schema's.

7.3. Een andere pseudo-random {0,1} generator

In de vorige paragraaf hebben we geconstateerd dat het bezwaar van de lineaire shift-registers voor de cryptografie juist hun lineariteit was. We zullen dus op een niet-lineaire manier getallen moeten gaan genereren. Van de andere kant moet men zich realiseren dat elke periodieke rij met periode p voortgebracht kan worden door een lineair shift-register, nl. met $f = 1 + x^p$. Daarom introduceren we nu een begrip dat in zeker opzicht een maat is voor de complexiteit van een periodieke rij.

DEFINITIE 7.15. Het lineaire equivalent van een periodieke rij is de lengte van het kortste lineair shift-register dat die rij kan voortbrengen.

In dit korte bestek is het niet mogelijk om diverse pseudo-random {0,1} generators te bespreken op hun toepasbaarheid in de cryptografie. We zullen hier het idee van "multiplexing" bespreken zoals men dat kan vinden in [1].
Hiervoor gebruiken we simultaan twee lineaire shift-registers die beide een primitief polynoom als karakteristiek hebben. Laat de geheugenplaatsen $A_0, A_1, \ldots, A_{m-1}$ resp. $B_0, B_1, \ldots, B_{n-1}$ van deze registers gevuld zijn op tijdstip t met $a_i(t)$ resp. $b_j(t)$. Om dit systeem te laten werken kiezen we eerst een h met $1 \leq h \leq m$ die voldoet aan

$$2^h \leq n \text{ als } h < m \text{ en } 2^h - 1 \leq n \text{ als } h = m.$$

Vervolgens kiezen we h getallen $0 \leq i_1 < i_2 < \ldots < i_h < m$.

Op tijdstip t beelden we de toestand van het eerste register of op het getal

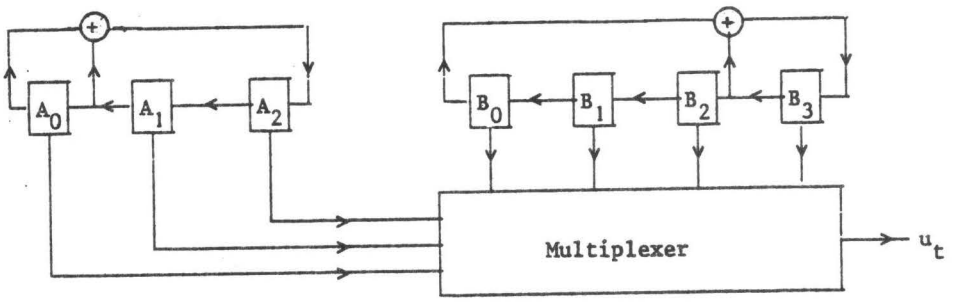
$$N_t = a_{i_1}(t) + a_{i_2}(t) * 2 + \ldots + a_{i_h}(t) * 2^{h-1} .$$

Tenslotte moeten we nog een afbeelding f kiezen die de getallen $\{0,1,\ldots,2^h - 1\}$ als $h < m$ of de getallen $\{1,2,\ldots,2^h - 1\}$ als $h=m$ afbeeldt in de verzameling $\{0,1,\ldots,n-1\}$.

De nieuwe rij (u_t) is nu gedefinieerd door

$$(7.4) \qquad u_t = b_{f(N_t)}(t) .$$

De inhoud van het A-register bepaalt dus welke geheugenplaats van het B-register uitgelezen wordt. Merk op dat we het geval $h = m$ apart behandelen omdat de nul-toestand in het A-register niet voorkomt.



Voorbeeld van een multiplexer

In dit voorbeeld kiezen we $h = 2$, $i_1 = 0$, $i_2 = 1$ en de afbeelding f gegeven $f(0) = 2$, $f(1) = 3$, $f(2) = 0$, $f(3) = 1$.

t	LSR 1	LSR 2	$a_{i_1}(t)$	$a_{i_2}(2)$	N_t	$f(N_t)$	u_t
0	1 0 0	1 0 0 0	1	0	2	0	1
1	0 0 1	0 0 0 1	0	0	0	2	0
2	0 1 0	0 0 1 1	0	1	1	3	1
3	1 0 1	0 1 1 1	1	0	2	0	0
4	0 1 1	1 1 1 1	0	1	1	3	1
5	1 1 1	1 1 1 0	1	1	3	1	1
6	1 1 0	1 1 0 1	1	1	3	1	1
7	1 0 0	1 0 1 0	1	0	2	0	1

etc.

Voorbeeld van een multiplexer (vervolg)

Jammer genoeg voert het nu te ver om de eigenschappen R1 - R3 en C1 - C3 hier uitvoerig te bespreken. Hoofdzaak is dat men 2 LSR's op een zodanige wijze heeft gecombineerd dat een analyse mogelijk blijft. We vermelden voor de volledigheid de volgende stellingen.

STELLING 7.16. Als n niet m deelt en $ggd(2^m - 1, (2^n-1)/(2^{ggd(m,n)} - 1)) = 1$, dan is de periode van de uitvoerrij van de multiplexer gelijk aan

$$kgv(2^m - 1, 2^n - 1).$$

I.h.b. als $ggd(m,n) = 1$, dan is de periode gelijk aan $(2^m - 1) * (2^n - 1)$.

STELLING 7.17. Als $ggd(m,n) = 1$ dan voldoet het lineaire equivalent d van de multiplexed rij aan:

1) $d = n(1+m)$ als $h = 1$,

2) $d = n\{1 + m + \binom{m}{2}\}$ als $h = 2 < m$,

3) $d \leq n \sum_{i=1}^{h} \binom{m}{i}$ als $2 < h < m-1$, met gelijkheid als de h geheugenplaatsen op gelijke afstanden staan,

4) $d = n(2^m - 1)$ als $h = m-1$ of $h = m$.

REFERENTIES

[1] Beker, H. en F. Piper, Cipher systems, the protection of communications, Northwood Books,1982.

[2] Golomb, S., Shift register sequences, Holder-Day, 1967.

[3] Knuth, D.E., The art of computer programming, Vol I en II, Addison-Wesley, 1969, 1973.

[4] Selmer, F.S., Linear recurrence relations over finite fields University of Bergen, 1966.

VIII Toepassingen van Cryptografie

1 Inleiding

In dit hoofdstuk gaan we in op de problemen die ontstaan als cryptografie toegepast gaat worden in een communicatiesysteem. Als de communicatie zich afspeelt tussen twee partijen die direct met elkaar communiceren, die elkaar volledig vertrouwen, die van elkaars identiteit overtuigd zijn, en die hun sleutel(s) nooit verliezen of vergeten, dan zijn er weinig moeilijkheden. In de praktijk doet zo'n ideale situatie zich zelden of nooit voor. Vaak zijn er meer gebruikers, vindt de communicatie plaats via een (computer-)netwerk waarin snoodaards zouden kunnen proberen zich te vermommen als een ander, worden boodschappen vervalst of achtergehouden, willen partijen later ontkennen een boodschap verstuurd of ontvangen te hebben, enzovoorts. Tegen al dit soort problemen zijn oplossingen bedacht met behulp van technieken uit de cryptografie.

We zullen proberen deze technieken aan de hand van een aantal voorbeelden toe te lichten. Vanzelfsprekend kunnen in dit overzichtsverhaal niet alle details besproken worden. Daartoe wordt verwezen naar de aangehaalde literatuur.

In de volgende paragrafen zullen we het cryptogram dat ontstaat door de boodschap M te vercijferen met sleutel K aangeven met K(M). We identificeren dus de vercijfer-operator met de sleutel, waarbij de achterliggende algoritme, veelal DES of een public key system, gegeven wordt verondersteld. Omgekeerd wordt de boodschap die ontstaat door het cryptogram C te ontcijferen met de sleutel K aangegeven met $K^{-1}(C)$. Tot slot staat $\oplus$ voor bitsgewijs optellen modulo 2, d.w.z. bitsgewijs XOR.

2 Opslag van passwords

In een computer systeem heeft het gebruik van passwords voor log-in procedures of toegang tot gegevensbestanden of programma's algemeen ingang gevonden. Passwords moeten echter bewaard blijven in het computer geheugen. Dat moet dan wel in een beveiligd gebied gebeuren, om niet het risico te lopen dat de password-tabel bij een (al dan niet opzettelijke) geheugen-dump openbaar gemaakt wordt. Nu is het nagenoeg onmogelijk een tabel met enige honderden of wellicht duizenden passwords

hier effectief tegen te beveiligen.

Het is echter relatief eenvoudig de tabel van passwords te vercijferen met behulp van een enkele sleutel. Deze ene sleutel wordt zorgvuldig bewaard. Direct na het intoetsen van het password door de gebruiker wordt dit m.b.v. de sleutel vercijferd, en het resultaat wordt vergeleken met het betreffende element uit de tabel. Op deze manier bevindt het niet-vercijferde password zich maar zeer kort in het computer systeem.

De vercijfering kan gebeuren met een conventioneel systeem zoals DES. Een fraaiere oplossing is de vercijfering te doen met een one-way function. Gezien nergens ontcijferd hoeft te worden, is een trapdoor hier niet nodig. In dit geval mag de sleutel rustig openbaar gemaakt worden, en behoeft er dus helemaal niets fysiek beveiligd te worden.

Als one-way function kan bijv. een random knapsack worden gekozen. Ook is het mogelijk DES te gebruiken. De afbeeldingen

$$\text{PASSWORD} \rightarrow \text{KEY(PASSWORD)} \oplus \text{PASSWORD}$$

en

$$\text{PASSWORD} \rightarrow \text{PASSWORD(KEY)}$$

zijn one-way functions zonder trap-door. Hierin is KEY een vast bitpatroon, dat publiekelijk bekend mag zijn.

3 Terminal-to-host verbinding

Indien vele terminals met een centrale computer (host) zijn verbonden, en alle lijnverbindingen beveiligd moeten (kunnen) worden met cryptografische sleutels, dan moeten deze sleutels in de host veilig opgeborgen worden. Hetzelfde probleem als met de passwords doet zich hier voor. De oplossing die hieronder wordt geschetst is bekend als het IBM Cryptografie Subsystem (zie [1] of [3]).

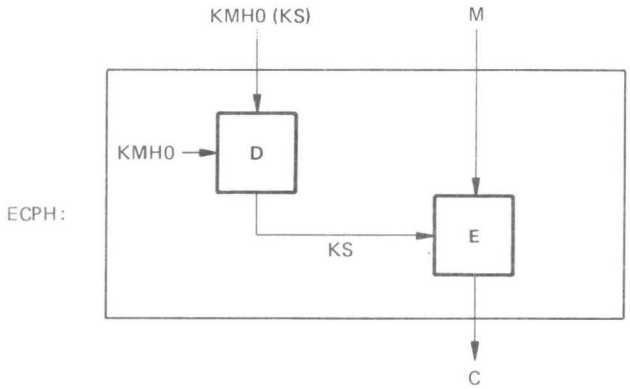
De sleutels worden vercijferd m.b.v. een master-key. Deze master-key (KMHO) is alleen bekend binnen de host, en wordt daar zorgvuidig bewaakt. In feite bevindt KMHO zich, tesamen met de DES-chip, binnen een "cryptografic facility", die slechts een beperkt aantal opdrachten kan verrichten.

De niet-vercijferde sleutel waarmee de conversatie met de terminal plaats vindt, KS, moet vanzelfsprekend ook aan de terminal bekend zijn. Hoe dat gebeurt komt later.

Wil de host een boodschap M overzenden naar een terminal T, en moet dit gebeuren onder bescherming van sleutel KS, dan zoekt hij de vercijferde sleutel KMHO(KS)= RN op, ontcijfert dit m.b.v. de master-key KMHO tot KS, en vercijfert de boodschap M m.b.v. KS. Deze twee laatste operaties gebeuren binnen de cryptografic facility, teneinde de sleutel KS niet te openbaren. In formulevorm gebeurt daar:

$$C = ECPH(KMHO(KS), M) = KMHO^{-1}(KMHO(KS))(M).$$

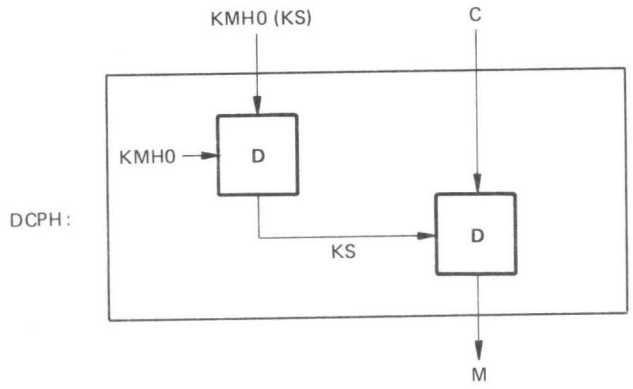
Schematisch:



Wil de host omgekeerd het cryptogram C dat ontvangen wordt van de terminal ontcijferen, dan berekent hij weer eerst KS als boven, en ontcijfert nu C met behulp van KS tot de boodschap M:

$$M = DCPH(KMHO(KS), C) = (KMHO^{-1}(KMHO(KS)))^{-1}(C)$$

ofwel



Aan de andere kant van de lijnverbinding kan de terminal direct vercijferen en ontcijferen met de sleutel KS. Natuurlijk kan ook hier KS beveiligd worden m.b.v. een master-key die alleen bekend is binnen de terminal, maar meestal zal dit niet nodig zijn aangezien de meeste terminals maar met één host communiceren. Is de terminal zelf een host, zodat we met een host-to-host verbinding te maken hebben, dan zal deze beveiliging wel degelijk toegepast moeten worden.

Dit protocol functioneert redelijk, zolang de sleutel KS geheim blijft. Worden echter veel data overgestuurd, dan zou dit de sleutel kunnen comprimiteren: een chosen plaintext attack wordt zelfs mogelijk. Een tweede probleem is hoe terminal en host de sleutel uitwisselen.

Het eerste probleem wordt opgelost door de sleutel KS (de "session key") alleen maar geldig te laten zijn tijdens een zitting. Het probleem van de verspreiding van de sleutel wordt in een klassiek cryptografisch systeem opgelost (of liever: verschoven) door de terminal uit te rusten met een vaste "device key" KMT die bekend is aan zowel host als terminal. We hebben nu het oude probleem terug hoe al die device keys op te bergen in de host, en dit gebeurt met een tweede master-key, KMH1. Het uitwisselen van de session key KS gebeurt nu als volgt;

1. De host genereert een willekeurig rij bits RN (bijv. door de tijd van de dag herhaald te vercijferen met de master-key), en bewaart deze. RN wordt opgevat als de vercijfering van sleutel KS.

2. Vervolgens wordt RN m.b.v. KMH0 ontcijferd tot KS.
3. Parallel hiermee wordt de vercijferde device-key, KMH1(KMT) ontcijferd m.b.v. sleutel KMH1 tot KMT.
4. Nu wordt KS vercijferd m.b.v. KMT en naar de terminal gezonden.
5. In de terminal wordt deze vercijferde sleutel weer ontcijferd m.b.v. KMT.

De host berekent dus:

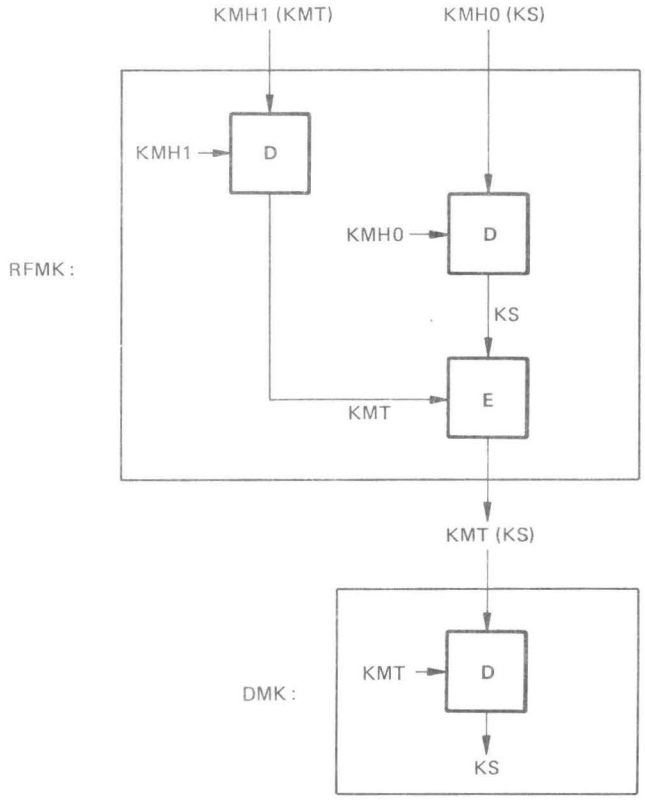
$$KMT(KS) = RFMK(KMH1(KMT), KMH0(KS)) =$$
$$= KMH1^{-1}(KMH1(KMT))(KMH0^{-1}(KMH0(KS))).$$

(RFMK betekent: "reencipher from master-key".)

De terminal berekent:

$$KS = DMK(KMT, KMT(KS)) = KMT^{-1}(KMT(KS)).$$

(DMK betekent: "decipher under master-key".)



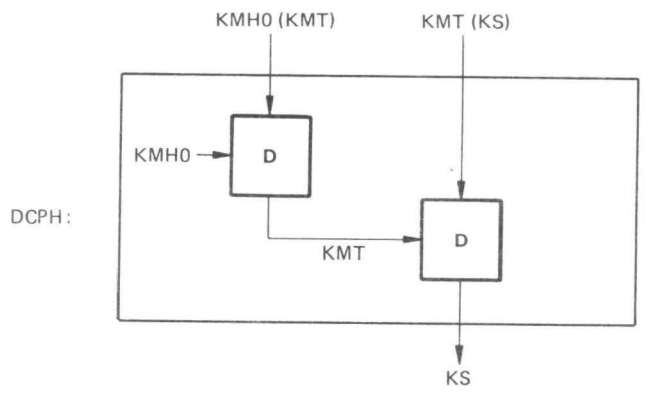
Het probleem van de verspreiding van de session key KS is hiermee verschoven naar de verspreiding van de device key KMT. Dit laatste is echter een eenmalige actie, die niet oplosbaar is binnen een conventioneel cryptografisch systeem. De enige oplossing binnen de cryptografie is een public key system te gebruiken, zie § 5.

Tot slot nog een opmerking over de master keys KMH0 en KMH1 in de host. Men kan zich afvragen waarom de device key KMT niet wordt opgeborgen onder bescherming van KMH0. In dat geval zou de volgende stategie voor een indringer bestaan om achter de session key KS te komen:

1. Tap KMT(KS) af tijdens de uitwisseling van de session key.
2. Roep DCPH aan met argumenten KMH0(KMT) en KMT(KS).
   De uitkomst is:

$$\begin{aligned} &DCPH(KMH0(KMT),\ KMT(KS)) = \\ &= (KMH0^{-1}(KMH0(KMT)))^{-1}(KMT(KS)) = \\ &= KMT^{-1}(KMT(KS)) = KS. \end{aligned}$$

Hiermee zou de session key gereconstrueerd kunnen worden. Schematisch:



In de IBM benadering volgen KMH0 en KMH1 uit elkaar door bepaalde bits te complementeren. Men had ook KMH1 = $KMH0^{-1}$ kunnen kiezen, m.a.w. de ontcijferde device keys i.p.v. de vercijferde kunnen bewaren.

4 Cryptografie in een netwerk

Er zijn in wezen twee methoden om cryptografie in een communicatie netwerk in te passen:

- end-to-end encryption: de boodschap wordt vercijferd in de terminal van de zender, en pas ontcijferd in de terminal van de ontvanger. Onderweg wordt de boodschap nergens ontcijferd.
- node-by-node encryption: hier wordt iedere lijnverbinding tussen twee knooppunten in het netwerk beveiligd door een aparte sleutel. Een boodschap wordt bij ieder knooppunt vertaald van de ene sleutel naar de volgende. Dit gebeurt in een fysiek beveiligd module zodat de boodschap zelf niet openbaar gemaakt wordt.

Een voordeel van de eerste methode is dat maar op één plaats ontcijferd hoeft te worden, en dat de eindgebruikers de zekerheid hebben dat de boodschap nergens anders ontcijferd kan worden dan bij de ontvanger. Er is dus geen risico van - al dan niet opzettelijke - aflevering op de verkeerde plaats (misrouting). Een probleem is echter de uitwisseling van de sleutel, en het feit dat de routing informatie niet vercijferd kan worden, waardoor een indringer weliswaar niet weet wat er verstuurd wordt, maar wel wie met wie communiceert. Deze "traffic analysis" kan in bepaalde gevallen een serieuse aantasting van de privacy betekenen. De fraaiste oplossing is beide vormen van encryptie te combineren. Dat wil zeggen, dat op (tenminste) twee niveaus in het netwerk-protocol vercijfering en ontcijfering mogelijk moet zijn.

Het is duidelijk, dat de sleutel in een cryptografisch systeem nimmer met end-to-end encryption uitgewisseld kan worden (tenzij bij iedere terminal een complete lijst van alle device keys van de andere terminals beschikbaar is, maar dat zou de veiligheid van die sleutels niet verhogen!). Een oplossing is dat de (een) host een session key KS bedenkt (zoals in §3), en deze volgens node-by-node encryption meedeelt aan de terminals van de beide eindgebruikers.

5 Sleutel-verspreiding met een public key system

In de vorige paragrafen zijn we verscheidene keren met de onmogelijkheid geconfronteerd om binnen een conventioneel cryptografisch systeem een sleutel uit te

wisselen over een niet-beveiligd kanaal. Met een public key system kunnen we hier wat aan doen. Volgende protocol maakt gebruik van de praktisch-onmogelijkheid om de logarithme van een groot getal modulo een groot priemgetal te bepalen.

Twee gebruikers, A en B die een sleutel willen uitwisselen over een niet-beveiligd kanaal gaan als volgt te werk.

- Eerst komen ze (openbaar) een groot priemgetal p en ander getal q overeen.
- Vervolgens bedenkt gebruiker A een groot getal a, en zendt q^a <u>mod</u> p over naar gebruiker B.
- Parallel hiermee bedenkt B een groot getal b, en zendt q^b <u>mod</u> p over naar A.
- A berekent $K = (a^b)^a$ <u>mod</u> p.
- B berekent $K = (q^a)^b$ <u>mod</u> p.
- K is de overeengekomen sleutel.

Een afluisteraar kent p, q, q^a <u>mod</u> p, en q^b <u>mod</u> p. Gezien de moeilijkheid van het logarithme-probleem kan hij hieruit noch a, noch b berekenen, en ook niet $K = q^{ab}$.

Dit protocol voorkomt niet dat een indringer S zich voordoet als een van de gebruikers, zeg B. Zelfs kan deze snoodaard zich tegenover A voordoen als B, en tegelijk tegenover B als A. A en B denken dan met elkaar te communiceren, terwijl in feite S alle boodschappen opvangt, ontcijfert, eventueel wijzigt, opnieuw vercijfert, en doorstuurt. A en B kunnen het S wel èrg moeilijk maken door via een fysiek ander kanaal, bijv. per brief, nogmaals q^a <u>mod</u> p en q^b <u>mod</u> p uit te wisselen.

Een alternatief is de getallen q^a <u>mod</u> p en q^b <u>mod</u> p op te slaan in een "public key directory". De enige gebruiker die b kent is de echte B, en S kan daar niet tussen komen. Dit levert echter het probleem op hoe en waar de public key directory opgeslagen moet worden. Slaat men deze op één plaats op, dan zou S ervoor kunnen zorgen dat A of B de verkeerde public key ophaalt. Verspreidt men de directory als een telefoonboek, dan is wijziging van een gecompromiteerde sleutel moeilijk, en moet tijdens de verspreiding geen vervalsing mogelijk zijn.

De les hieruit is dat public key systems de verspreiding van een sleutel over en niet-beveiligd kanaal mogelijk maken, maar dat deze methode niet bestand is tegen stelselmatige vervalsing van alle communicatie. Tegen een dergelijk machtige vijand is geen enkel cryptografisch systeem opgewassen.

Zodra het mogelijk is tussen twee partijen een sleutel uit te wisselen, is het niet moeilijk een beveiligde conferentie te beleggen waaraan een willekeurig aantal (n) partijen deelnemen. De oplossing is eenvoudig: partij A ("de voorzitter" bedenkt een conferentiesleutel K, en zendt deze, beveiligd volgens de hierboven beschreven methode, over naar alle andere deelnemers. Na deze n-1 transmissies kan de conferentie aanvangen, waarbij iedere boodschap vercijferd wordt m.b.v. sleutel K.

6. Legitimatie

Legitimatie ("user authentication") is een proces waarin een partij A tegenover een partij B bewijst dat hij inderdaad degene is voor wie hij zich uitgeeft. Normaal gebeurt dit door een handtekening, een paspoort, een wachtwoord, een duimafdruk, of iets dergelijks. Essentiëel hiervoor is, dat

- A iets kan, weet of heeft, dat niemand anders kan, weet of heeft;
- dat iets niet na te maken is;
- B het wel kan controleren.

Voor datacommunicatie komt alleen een digitale boodschap in aanmerking die

- iets bevat dat alleen aan A bekend is;
- iets éénmaligs, dus tijdsafhankelijks heeft;
- wel controleerbaar is.

In een public key system is dit te bereiken door A een aan B bekende, niet eerdere opgetreden, boodschap te vercijferen m.b.v. de geheime sleutel van A. B controleert of ontcijfering m.b.v. de openbare sleutel van A de afgesproken boodschap oplevert. In een conventioneel systeem lijkt legitimatie en stuk moeilijker, omdat een geheime sleutel pas uitgewisseld kan worden nadat de identiteit van beide partijen vaststaat. We gaan hier nu niet verder op in, omdat hetzelfde

probleem terugkomt in § 10 en § 11 bij de verificatie van de authenticiteit van een boodschap.

7 Authenticering met een public key system

Hoewel reeds eerder (Hoofdstuk V) behandeld is hoe een boodschap gesigneerd kan worden in een public key system, willen we dit hier kort herhalen, en enige praktische problemen aanstippen. We gaan ervan uit dat géén beveiliging nodig is tegen afluisteren. Mocht dit wel nodig zijn, dan kan dit bereikt worden door eerder besproken cryptografische technieken.

In een public key system vercijfert partij A de boodschap M met behulp van zijn geheime sleutel D_A tot $C = D_A(M)$. De ontvanger B ontcijfert dit met behulp van de openbare sleutel $E_A = D_A^{-1}$ tot $E_A(C) = M$. Hiervoor is vereist dat $E_A D_A$ de identiteit is, m.a.w. dat D_A één-éénduidig is. Uit het feit dat M zinvolle informatie bevat concludeert B dat A de zender moet zijn geweest, want niemand anders kent D_A, en kan C produceren, ook B zelf niet. Hiertoe moet M echter redundantie bevatten (anders zou iedere boodschap "zinvol" zijn, en B niets wijzer worden). A en B spreken dus af dat een deel van de boodschap bekende informatie bevat, bijv. de namen van A en B, en een volgnummer.
De boodschap M ziet er dan uit als:

$$M = (A,B,i,T),$$

waarin T de eigenlijke informatie bevat. Als B na ontcijferen een boodschap van deze aard terugvindt, dan weet hij dat

- A de zender is (want D_A is gebruikt),
- de boodschap onverminkt is overgekomen.

Het nut van de naam van A is niet zo groot: dit had ook een rij nullen kunnen zijn. De naam van B dient er toe dat B later kan bewijzen dat hij de boodschap niet afgeluisterd heeft. Het volgnummer i is van belang opdat een boodschap niet afgeluisterd kan worden, en later opnieuw verzonden. (Voor elektronisch geldverkeer zou dat uiterst ongelukkig zijn!) De redundantie in (A,B,i) moet groot

genoeg zijn om trial-and-error door B of een indringer te voorkomen: anders zou deze at random cryptogrammen kunnen genereren, deze ontcijferen met E_A, net zolang totdat de ontcijferde boodschap "per ongeluk" met (A,B,i) begint.

B kan later met recht beweren dat de boodschap (A,B,i,T) van A afkomstig is. Hij bewaart daartoe het cryptogram $D_A(A,B,i,T)$. Een arbiter kan verfiëren dat $E_A(D_A(A,B,i,T))$ met A,B,i begint. Aangezien A de enige is die D_A kent, moet hij de afzender zijn geweest.

De redundantie A,B,i kan een bezwaar zijn, omdat deze in ieder blok dat vercijferd wordt herhaald moet worden: blokken zonder deze redundantie kunnen ongemerkt gewijzigd worden. Dit geeft een behoorlijke overhead. Er bestaat echter een methode om blokken "aan elkaar te rijgen" tot een groot geheel. Dit proces, geheten "block chaining", is niet specifiek voor public key systems, maar geldt evengoed voor conventionele systemen.

8 Block chaining

Stel dat een boodschap M bestaat uit een rij blokken $(M_0,M_1,\ldots,M_{n-1})$ waar de vercijfer-algoritme op werkt. Als we nu vercijferen volgens

$$C_i = K(M_i) \qquad (i = 0,1,\ldots,n-1),$$

dan beïnvloedt een wijziging in het i^{de} blok niet de overige blokken. Soms is dit een voordeel bijv. een fout tijdens transmissie tast slechts één blok aan, maar soms is het een nadeel, immers een indringer kan C_i wijzigen zonder dat de ontvanger dat merkt (gesteld dat M_i geen bekende informatie bevat).

Een remedie is de blokken "aan elkaar te rijgen" door middel van z.g. "plaintext-ciphertext feedback". Hiertoe definiëren we:

$$C_i = K(M_i \oplus M_{i-1} \oplus C_{i-1}) \qquad (i = 0,1,\ldots,n-1),$$

$M_{-1} = 0$, en $C_{-1} = 0$. Ontcijferen gebeurt volgens

$$M_i = K^{-1}(C_i) \oplus M_{i-1} \oplus C_{i-1} \qquad (i = 0,1,\ldots,n-1).$$

Nu veroorzaakt verminking van C_i tijdens transmissie -hetzij door een indringer hetzij door storing- een verandering in alle volgende blokken van de ontcijferde boodschap. Door nu in het laatste blok M_{n-1} redundantie op te nemen (bijv. $M_{n-1} = 0$), weet de ontvanger onmiddellijk of er tijdens transmissie iets misgegaan is. In feite werkt deze vorm van block chaining als een fouten-detecterende code.

Voor de volledigheid zij hier opgemerkt dat er nog een andere,eenvoudiger, vorm van block chaining bestaat, de "ciphertext feedback". Hiervoor geldt:

$$C_i = K(M_i \oplus C_{i-1}) \qquad (i = 0,1,\ldots,n-1),$$

en omgekeerd:

$$M_i = K^{-1}(C_i) \oplus C_{i-1} \qquad (i = 0,1,\ldots,n-1).$$

Deze vorm van block chaining veroorzaakt geen -al dan niet gewenste- foutvoortplanting, en is dan ook niet geschikt voor foutendetectie. De techniek wordt gebruikt om te voorkomen dat herhaald optredende structuren in de boodschap, bijv. lange rijen spaties, niet terug te vinden zijn in het cryptogram.

9 Authenticering met een public key system gecombineerd met block chaining

Met de techniek van plaintext-ciphertext feedback uit de vorige paragraaf werd de ontvanger in de gelegenheid gesteld de authenticiteit van een boodschap te verifiëren. Als bewijs kan hij het echter alleen gebruiken, als hij de sleutel K zelf niet kent, want anders had hij de boodschap met het bijbehorende cryptogram net zo goed zelf kunnen maken. Dit idee kan in een public key system worden toegepast. Daartoe wordt de boodschap $(M_0,M_1,\ldots M_{n-1})$ vercijferd tot $(C_0,C_1,\ldots,C_n)$ met

$$C_i = D_A(M_i \oplus M_{i-1} \oplus C_{i-1}) \qquad (i = 0,1,\ldots,n),$$

waarin $M_{-1} = C_{-1} = 0$, $M_n = 0$. De ontvanger ontcijfert volgens

$$M_i = E_A(C_i) \oplus M_{i-1} \oplus C_{i-1} \qquad (i = 0,1,\ldots,n),$$

en verifiëert dat $M_n = 0$. Stel dat een indringer blok C_i (en eventueel een aantal blokken met hoger indices) opzettelijk verandert in C_i', etc..
Dan blijven $C_0,C_1,\ldots,C_{i-1}$ en $M_0,M_1,\ldots,M_{i-1}$ ongewijzigd, dus in de ontcijferde boodschap verandert M_i in M_i':

$$M_i' = E_A(C_i') \oplus M_{i-1} \oplus C_{i-1}.$$

Het is ook niet mogelijk dat C_i' zodanig gekozen wordt dat M_i' een gewenste waarde aanneemt, want dat vereist kennis van D_A, de geheime sleutel van de zender. Nu is $M_i' \oplus C_i'$ bekend is. C_{i+1}' kan nu vrij gekozen worden, maar M_{i+1} volgt uit:

$$M_{i+1}' = E_A(C_{i+1}') \oplus M_i' \oplus C_i'.$$

Wederom kan C_{i+1}' niet zodanig gekozen worden dat M_{i+1}' een gewenste waarde aanneemt, bijv. M_{i+1}. Dit proces gaat zo door totdat M_n onbeïnvloedbaar wordt gewijzigd.

Het bovenstaande bewijst dat een indringer niet in staat is het cryptogram zodanig te vervalsen dat M_n behouden blijft. Dit bewijst de authenticiteit van de boodschap.

In het geval van een conflict tussen de zender A en de ontvanger B moet B kunnen bewijzen dat alleen A het bericht heeft kunnen versturen. Daartoe zou B het cryptogram $C_0,C_1,\ldots,C_n$ kunnen bewaren. Hierboven is aangetoond dat noch B, noch een indringer in staat is een legitiem cryptogram te fabriceren, d.w.z. horende bij een boodschap met $M_n = 0$. B overlegt aan de arbiter het cryptogram $C_0,C_1,\ldots,C_n$ en de boodschap $M_0,M_1,\ldots,M_{n-1}$. De arbiter controleert of deze bij elkaar horen, d.w.z. of

$$M_i = E_A(C_i) \oplus M_{i-1} \oplus C_{i-1} \qquad (i = 0,1,\ldots,n),$$

waarin $M_{-1} = C_{-1} = 0$, en i.h.b. of $M_n = 0$. Klopt dit, dan moet A de zender zijn; klopt dit niet, dan heeft B vervalsing gepleegd.

Hiermee is aangetoond dat het voor B voldoende is $C_0, C_1, \ldots, C_n$ te bewaren. Nodig is dit echter niet. B kan volstaan met het bewaren van het laatste blok C_n van het cryptogram. Mocht het ooit tot een arbitrage komen, dan kan B of de arbiter de rest van het cryptogram reconstrueren volgens

$$C_{i-1} = M_i \oplus M_{i-1} \oplus E_A(C_i) \qquad (i = n, n-1, \ldots, 0).$$

Het laatste blok C_n heet wel de handtekening van A.

Nog een stap verder gaand, kan men besluiten dat A net zo goed $(M_0, M_1, \ldots, M_{n-1}, C_n)$ kan oversturen naar B, want M_{i-1} en C_{i-1} volgen uit elkaar zodra M_i en C_i bekend zijn.

Samenvattend gaat de bewijsvoering als volgt: B verschaft de arbiter de boodschap $M_0, M_1, \ldots, M_{n-1}$ en de handtekening C_n. De arbiter reconstrueert het totale cryptogram volgens

$$C_{i-1} = M_i \oplus M_{i-1} \oplus E_A(C_i) \qquad (i = n, n-1, \ldots, 0),$$

met $M_n = M_{-1} = 0$. Blijkt $C_{-1} = 0$, dan is A de zender; is $C_{-1} \neq 0$, dan heeft B de boel opgelicht.

De geschetste methode voorkomt niet dat een boodschap afgeluisterd en opnieuw verstuurd wordt. Evenmin wordt voorkomen dat B de boodschap gestolen heeft. Het is echter eenvoudig in de boodschap, bijv. in M_0, de informatie (A,B,i) op te nemen.

10 Authenticering met een conventioneel systeem - met arbiter

De essentie van het authenticeren van een boodschap in een data-communicatie-systeem is dat de zender een geheim kent dat niet bekend is aan enig ander, ook (zeker) niet aan de ontvanger. Als de ontvanger, of een indringer, dezelfde kennis heeft als A, dan kan niemand hem weerhouden zich te gedragen als ware hij A. In een cryptografisch systeem waar zender en ontvanger dezelfde sleutels bezitten,

is authenticering dus principiëel onmogelijk: de ontvanger kan iedere boodschap naar believen vervalsen.

Binnen een conventioneel systeem wordt de oplossing gevonden door inschakeling van een arbiter - ook wel authentication server genoemd - wiens integriteit boven verdenking staat. Als A de boodschap M wil zenden naar B, dan stuurt hij deze naar de arbiter, vercijferd volgens een sleutel K_A die alleen hij en de arbiter kennen. Dit gebeurt volledig analoog aan de methode zoals beschreven in de vorige paragraaf, dus $M = (M_0, M_1, \ldots, M_{n-1})$ wordt vercijferd tot $(C_0, C_1, \ldots, C_n)$ volgens

$$C_i = K_A(M_i \oplus M_{i-1} \oplus C_{i-1}) \qquad (i = 0,1,\ldots,n),$$

waarin $M_{-1} = C_{-1} = M_n = 0$. Het enige verschil is dat de ontvanger (de arbiter) hier de sleutel K_A kent. Gezien een eventuele indringer K_A niet kent kan de arbiter de authenticiteit van de boodschap verifiëren. Hij is zelf de enige die door boodschap zou kunnen vervalsen, maar volgens aanname doet hij dat niet.

Vervolgens stuurt de arbiter aan B de boodschap $M' = (M, C_n)$ door, vercijferd volgens de sleutel K_B die alleen hij en B kennen, wederom met plaintext-ciphertext feedback, en voorzien van zijn handtekening:

$$C'_i = K_B(M'_i \oplus M'_{i-1} \oplus C'_{i-1}) \qquad (i = 0,1,\ldots,n+1),$$

waarin $M'_{-1} = C'_{-1} = M'_{n+1} = 0$, $M'_n = C_n$, en $M'_i = M_i$ voor $i = 0,1,\ldots,n-1$.

B ontcijfert de boodschap M' en controleert of $M'_{n+1} = 0$, daarmee de authenticiteit van de boodschap voor zichzelf bewijzend. Vervolgens bewaart hij de volledige boodschap $M' = (M, C_n)$. (Het bewaren van de handtekening C'_{n+1} van de arbiter is zinloos, want die kan B zo namaken.) B kent nu de handtekening C_n van A m.b.t. de boodschap M, zonder dat hij de gebruikte sleutel, K_A, kent. Als in het vorige voorbeeld is dit voldoende om in een eventueel dispuut te kunnen bewijzen - zij het met medewerking van de arbiter - dat A de afzender is.

Ook hier wordt aangenomen dat M de informatie (A,B,i) bevat, ter voorkoming van afluisteren door B of retransmissie door een indringer.

11 Authenticering met een conventioneel systeem - zonder arbiter

Soms is het onmogelijk, of ongewenst, om ten tijde van de verzending van een boodschap de hulp in te roepen van een arbiter. Het is dan toch nog mogelijk gebruik te maken van een conventioneel systeem, gesteld dat tevoren reeds bepaalde geauthenticeerde informatie is verzonden. Een oplossing is aangegeven door M.O. Rabin [7].

De zender A bedenkt een lijst van blokken van ieder 2r sleutels. Hoe groot r is zal hieronder gespecificeerd worden. Om latere verificatie mogelijk te maken, signeert hij een standaard boodschap met ieder van deze sleutels, en doet deze lijst van handtekeningen, het "contract", aan de ontvanger B toekomen. Dit gebeurt op een beveiligde methode, bijv. via een arbiter (zie § 10), via een "normaal" ondertekend geschrift, of - als het niet de eerste keer is - via de hieronder beschreven methode.

Zodra A de boodschap M wil verzenden aan B signeert hij M m.b.v. alle 2r sleutels $K_1, K_2, \ldots, K_{2r}$ uit blok i van de lijst. B wil natuurlijk verifiëren of de boodschap onverminkt is overgekomen, en of A inderdaad de afzender is. Daartoe vraagt hij de afzender hem r sleutels $K_{j_1}, K_{j_2}, \ldots, K_{j_r}$ te onthullen, waarbij $j_1, j_2, \ldots, j_r$ door B gekozen worden uit de getallen $1, 2, \ldots, 2r$. B verifiëert of

- deze sleutels kloppen met de handtekeningen uit het contract - dit bewijst dat A de afzender is;
- deze sleutels kloppen met de handtekeningen onder de boodschap M - dit bewijst de authenticiteit van de boodschap.

In het geval dat A later ontkent de afzender te zijn overlegt B aan de arbiter de boodschap M, de lijst met 2r handtekeningen onder standaard boodschap, en blok i van het contract. De arbiter vraagt van A de sleutels $K_1, K_2, \ldots, K_{2r}$. De arbiter verifieert nu of

- deze sleutels kloppen met de handtekeningen uit het contract - is dit niet zo dan heeft A de zaak kennelijk bedrogen;
- deze sleutels kloppen met de handtekeningen onder de boodschap M - als tenminste r+1 handtekeningen correct zijn, dan wordt A verondersteld de boodschap verstuurd te hebben; anders wordt B verondersteld de boodschap verzonnen te hebben.

Hoe kan A succesvol ontkennen een boodschap verzonden te hebben die hij wel degelijk verzonden heeft? Daartoe moeten maximaal r handtekeningen correct zijn (anders gelooft de arbiter zijn ontkenning niet), en moet B precies deze handtekeningen opgevraagd hebben.
De kans op dit laatste is $P_1 = 1/\binom{2r}{r}$.

Hoe kan B de boodschap succesvol aan A toeschrijven die deze nooit verstuurd heeft? Daartoe moet hij, behalve de r handtekeningen gemaakt met $K_{j_1}, K_{j_2}, \ldots, K_{j_r}$ nog tenminste één handtekening goed gegokt hebben. De kans hierop is $P_2 \approx r.2^{-k}$, waarin k het aantal bits in de handtekening is.

Willen P_1 en P_2 van vergelijkbare grootte zijn, dan moet r iets minder dan $\frac{1}{2}k$ zijn. Bijv. voor k = 64 en r = 31 is $P_1 \approx P_2 \approx 2.10^{-18}$.

12 Beveiliging van bestanden

Wil men een gegevensbestand beveiligen tegen ongeoorloofd uitlezen of wijzigen, dan kan dat heel goed gebeuren door het te vercijferen met een geheime sleutel. De communicatie met zo'n vercijferd bestand lijkt sterk op de communicatie tussen twee gebruikers in een netwerk. Het grootste verschil is dat de sleutel langere tijd bewaard moet blijven: bij een communicatie tussen twee gebruikers hoeft de session key maar bewaard te blijven gedurende één zitting, terwijl een gegevensbestand meestal een veel langer leven beschoren is. Een ander probleem is dat een bestand vaak door meer gebruikers gelezen moet kunnen worden.

De eenvoudigste methode is het bestand te beveiligen m.b.v. een file key KF (te vergelijken met een session key), en deze te laten bewaren door de gebruiker zelf. Er zijn methoden aangegeven [1] om de file key in het systeem of bij het bestand op te slaan, maar uiteindelijk moet toch de gebruiker een sleutel bewaren om bij deze sleutel te komen. In het Information Protection System (IPS) van IBM wordt er dan ook vanuit gegaan dat de gebruiker verantwoordelijk is voor het bewaren van de sleutel.

Het bestand zelf zal vercijferd worden met behulp van een vorm van block chaining,

teneinde herhaalde patronen niet te laten doorwerken in het cryptogram, en (eventueel) de authenticiteit van het bestand te kunnen controleren. Voorts zal aan het begin van een vercijferd bestand het cryptogram van een bekende boodschap worden opgenomen, zodat het mogelijk is een foutieve sleutel onmiddellijk te detecteren.

IPS levert in de eerste plaats een programma IEBCODE. Dit programma stelt de gebruiker in staat een vercijferde of ontcijferde copie te maken van een geheel bestand met behulp van een door hem te kiezen sleutel. Normaal gesproken zal dit alleen worden gebruikt om een reeds bestaand bestand in een beveiligd systeem op te nemen, of bij wijziging van een sleutel.

Belangrijker zijn de IPS subroutines. Deze kunnen vanuit een programmeertaal worden aangeroepen, en schrijven de data meteen in vercijferde vorm in het bestand, dan wel lezen en ontcijferen de vercijferde data uit het bestand. Allerlei opties, zoals het selectief vercijferen van bepaalde delen van een bestand (bijv. namen of geldbedragen), eventueel met verschillende sleutels (teneinde bepaalde gebruikers maar beperkt toegang te geven), zijn mogelijk. Het lijkt niet zinvol hier op de details van deze subroutines in te gaan.

13 Het genereren van een sleutel

In verscheidene van de in dit hoofdstuk behandelde protocollen is het nodig ergens een (tijdelijke) sleutel te bedenken. Dit moet op een niet-voorspelbare wijze gebeuren. Eén methode is het herhaald werpen met een zuivere munt. Dit wordt inderdaad voorgesteld bij het genereren van de master key in de host [5].
De random generator van een computer is duidelijk niet bruikbaar, want die is voorspelbaar, danwel vereist een onvoorspelbaar seed. Wel kan een vercijfer algoritme (bijv. DES) worden gebruikt waarvan de sleutel geheim is. Zo kan een session key worden gegenereerd door een éénmalige (niet-geheime) boodschap - bijv. een combinatie van datum en tijd - te vercijferen m.b.v. de master key.

Bij sleutels voor persoonlijk gebruik, zoals passwords, speelt hiernaast nog dat deze enerzijds niet voorspelbaar moeten zijn, anderzijds wel te onthouden. Een

oplossing hiervoor staat bekend onder de naam "key crunching". De gebruiker bedenkt een uitdrukking (in letters) die veel langer mag zijn dan de sleutellengte. Deze uitdrukking wordt vervolgens vercijferd m.b.v. een master key met block chaining, en de "handtekening" wordt gebruikt als echte sleutel.

14 Mental Poker

Aan het eind van dit hoofdstuk vermelden we een wat ludieke toepassing van cryptografie. Twee spelers willen een kaartspel, bijv. poker, spelen over de telefoon. De vraag is een sluitend protocol te ontwerpen, waarmee de kaarten gedeeld kunnen worden zonder tussenkomst van een derde. Men mag ervan uitgaan dat de telefoonverbinding veilig is, maar over de betrouwbaarheid van beide spelers mag niets dan kwaads worden verondersteld. Achteraf moet voor beide spelers controleerbaar zijn dat zijn tegenspeler niet vals heeft gespeeld.

We laten dit probleem graag over als vermakelijke opgave aan de lezer. In feite is het een probleem van authenticering, waarbij zelfs de zender niet mag weten wat hij verzendt. Alle cryptografische ingrediënten voor een oplossing zijn te vinden in dit hoofdstuk. Een volledige oplossing (èn een bewijs dat het niet kan) is te vinden in [8].

15 Tot slot

In dit hoofdstuk hebben we geprobeerd aan te geven dat het gebruik van cryptografie in een communicatie systeem meer is dan het ontwerpen van een goede algoritme, of het kopen van een chip.

Veel meer dan een paar illustratieve voorbeelden konden in dit hoofdstuk niet worden gegeven. Een kant-en-klare eenheids-oplossing bestaat niet. Veel hangt af van de specifieke wensen van de gebruiker, van de gewenste mate van beveiliging, van de structuur van de organisatie, etc.

Uit de voorbeelden is het hopelijk duidelijk geworden dat conventionele cryptografische systemen (DES) en public key systemen beide hun voorbeelden hebben, en elkaar aanvullen. Ook "klassieke" beveiligingsmethoden zoals handgetekende contracten en fysieke sloten zullen hun plaats in een systeem blijven innemen. Maar ook menselijke acties, als het intoetsen en bewaren van passwords en sleutels, zullen nooit gemist kunnen worden. Enerzijds betekent dit een zwak punt in ieder systeem anderzijds is het een hele geruststelling!

16 Referenties

1. Ehrsam, W.F., Matyas, S.M., Meyer, C.H. and Tuchman, W.L., "A cryptographic key management scheme for implementing the Data Encryption Standard", IBM Systems Journal, Vol. 17, No. 2, pp. 106-125, 1978.

2. Fernandez, E.B., Summers, R.C. and Wood, C., Database security and integrity, Addison-Wesley, 1980.

3. Konheim, A.G., Cryptography: a primer, John Wiley and Sons, 1981.

4. Matyas, S.M., "Digital signatures - an overview", Comp. Networks, Vol. 3, No. 2, pp. 87-94, 1979.

5. Matyas, S.M. and Meyer, C.H., "Generation, distribution, and installation of cryptographic keys", IBM Systems Journal, Vol. 17, No. 2, pp. 126-137, 1978.

6. Needham, R.M. and Schroeder, M.D., "Using encryption for authentication in large networks of computers", Comm. ACM, Vol. 21, No. 12, pp. 993-999, 1978.

7. Rabin, M.O., "Digitalized signatures", in: De Millo, R.A., Dobkin, D.P., Jones, A.K. and Lipton, R.J., Eds., Foundations of secure computation, Academic Press, 1978, pp. 155-168.

8. Shamir, A., Rivest, R.L. and Adleman, L.M., "Mental Poker" in: D.Klarner, Ed., The mathematical gardner, Prindle, Weber & Schmidt, 1981, pp. 37-43.

SECURITY,
EEN VERZAMELING MAATREGELEN
OF
EEN MENTALITEIT

door

R.P. DE MOEL
Gegevens Beveiligings Inspecteur
van het
Nationaal Lucht- en Ruimtevaartlaboratorium

INDELING

1. Inleiding 2

2. Begripsbepaling en gebiedsafbakening 3

3. Bijzondere aandacht voor computerbeveiliging 5

4. Classificatie van informatie 6

5. Afweermechanismen 8

6. Algemene richtlijnen voor beveiliging 10

7. Beleidsoverwegingen met betrekking tot beveiliging 11

8. Slotopmerking. 13

1. INLEIDING

Cryptografie is de kunst om zodanig te schrijven, dat alleen zij het kunnen lezen, die de sleutel tot het gehanteerde vercijferingssysteem bezitten. In deze definitie ligt de eerste aanleiding besloten om als motto van de uiteenzetting "Security, een verzameling maatregelen of een mentaliteit" te kiezen. Cryptografie wordt niet alleen aangewend om zo te vercijferen, dat alleen de geadresseerde het bericht kan lezen, maar ook om te trachten in het bezit van de vercijferingssleutel te komen. Als vercijfering een maatregel genoemd mag worden in het kader van gegevensbeveiliging, dan mag het onbevoegd trachten te ontcijferen van voor anderen bedoelde berichten een tegenmaatregel genoemd worden, die de gegevensbeveiliging ernstig in gevaar brengt. Vanuit het standpunt van de deskundigen op het gebied van cryptografie kan de strijd tussen maatregelen en tegenmaatregelen als een interessante technische uitdaging gezien worden.

Een ander belangrijk aspekt van het gekozen motto is, dat niet alleen vercijfering als beveiligingsmaatregel aan de orde is, maar de gehele verzameling maatregelen, die onder het begrip "security" kan worden samengevat. Tegelijk komt hierbij ter sprake of dit "security" begrip alleen maar staat voor een verzameling maatregelen, of dat er meer achter moet worden gezocht. Hiermede wordt bedoeld, zoals ook het motto al suggereert, dat security in eerste instantie dient te worden gezien als een grondslag, een wijze van benadering, een mentaliteit, van waaruit een verzameling maatregelen tot stand komt.

De grondslag van de uiteenzetting is het belang van informatie voor een onderneming of instituut. Het begrip "informatie" wordt niet in de balans van een onderneming opgenomen en is zelden als zodanig terug te vinden in een jaarverslag. Toch zijn organisaties als geheel en de individuen binnen die organisaties dikwijls sterk afhankelijk van de beschikbare informatie en kan verlies of uitlekken van informatie de oorzaak zijn van aanzienlijke moeilijkheden of zelfs van de ondergang. Tegen deze risico's moeten de beveiligingsmaatregelen worden afgewogen. Beveiligingsmaatregelen roepen in het algemeen weerstand op, omdat daardoor de wijze van werken kan worden bemoeilijkt. Veronachtzaming van beveiligingsmaatregelen is gevaarlijk voor een onderneming en het is goed zich te realiseren dat informatie niet op de balans staat, omdat de werkelijke waarde onschatbaar is.

2. BEGRIPSBEPALING EN GEBIEDSAFBAKENING

De beveiligingsmaatregelen kunnen worden verdeeld naar een viertal gebieden:
- beveiliging m.b.t. personeel; "screenen" van (toekomstige) medewerkers is hier een belangrijk element van.
- toegangsbeveiliging, waardoor ongeautoriseerde toegang tot ruimten wordt voorkomen.
- computeromgevingsbeveiliging; hierbij is sprake van maatregelen die betrekking hebben op zaken als stroomvoorziening, luchtconditioneringsinstallaties, maar ook op maatregelen ter voorkoming van (compromiterende) elektromagnetische straling.
- computerbeveiliging; dit is het gebied van maatregelen, dat betrekking heeft op de beveiliging in de gegevensverwerkende apparatuur, meer algemeen aangeduid als EDP-faciliteiten.

In deze uiteenzetting wordt de aandacht vooral gericht op het laatste gebied, de computerbeveiliging.

Onder computerbeveiliging wordt verstaan het geheel van maatregelen dat gericht is op:

A. Het beschermen van de geautomatiseerde informatieverzorging tegen incidenten, die de continuïteit van dit proces kunnen verstoren; voor het merendeel zijn dit organisatorische maatregelen, die tot de computeromgevingsbeveiliging behoren, maar ook interne "recovery"-procedures worden hiertoe gerekend.

B. Het beschermen van gegevensverzamelingen en computerprogramma's tegen verminking; hierbij gaat het om voorkomen van verandering van de inhoud van de informatie en de programmatuur, al dan niet bedoeld. Bij opzettelijke inhoudelijke verandering is er sprake van computerfraude, bij onopzettelijke verandering van aantasting van de integriteit.

C. Het beschermen van gegevensverzamelingen en computerprogramma's tegen ongeautoriseerde kennisname; ook hierbij moet een onderscheid gemaakt worden tussen onbedoelde ongeautoriseerde kennisname, veelal als gevolg van storing, en opzettelijke ongeautoriseerde kennisname.

Het geheel van maatregelen houdt zich niet alleen bezig met maatregelen ter voorkoming van schending van beveiliging, maar ook met maatregelen ter beperking van de schade, indien deze niet (geheel) kon worden voorkomen.
In deze uiteenzetting wordt de meeste aandacht besteed aan de maatregelen ter voorkoming van ongeautoriseerde kennisname. Het is mogelijk, dat zaken, die daarbij aan de orde komen, tevens betrekking hebben op de andere gebieden van computerbeveiliging.

De belangrijkste zes elementen van EDP-faciliteiten, die kwetsbaar zijn voor schending van beveiliging, zijn: 1. processoren, 2. opslagfaciliteiten, 3. communicatiefaciliteiten, 4. terminals (op afstand), 5. gebruikers, 6. bedienend en onderhoudspersoneel. Onderstaande tabel geeft een overzicht van deze kwetsbare punten.

EDP-faciliteiten / kwetsbaar voor	processor	opslag-fac.	communicatie fac.	terminals (op afstand)	gebruikers	systeem-personeel
storingen in beschermingscircuits	X	X				
verkeerde adressering	X	X				
misbruik van speciale instrukties	X					
passeren van beschermings- en toegangsmaatregelen		X				
diefstal van afneembare gegevensdragers		X				
compromiterende straling		X	X	X		
overspraak			X			
technische inbraak			X	X		
ongecontroleerde copieën				X		
identificatie/autorisatiemisbruik					X	
gaten in het beveiligingsnetwerk					X	
toegang tot het gehele systeem						X

Overzicht 1: Kwetsbare punten in EDP-faciliteiten v.w.b. computerbeveiliging

In dit hoofdstuk is een indruk gegeven, wat security in het algemeen en in het bijzonder computersecurity inhoudt. Ook indien het gebied zorgvuldig wordt afgebakend, is er nog steeds sprake van een uitermate complexe materie, waarvoor een netwerk van maatregelen nodig is om zo op het eerste gezicht een redelijk afdoende beveiliging te bieden. Helaas moet echter worden gekonstateerd, dat het beveiligingsnetwerk veel overeenstemming vertoond met iets als de wetten, het beschermingsnetwerk van de maatschappij: er zitten altijd mazen in.

3. BIJZONDERE AANDACHT VOOR COMPUTERBEVEILIGING

Er is een aantal redenen op te noemen, waarom beveiliging van informatie in EDP-faciliteiten meer aandacht verdient dan op conventionele wijze opgeslagen informatie. Deze redenen zijn: de dichtheid van de informatie, de verborgenheid van de informatie, de toegankelijkheid, de vervalsingsmogelijkheden en de vasthoudendheid van de EDP-informatiedragers.

De dichtheid van de informatie in EDP-faciliteiten is veel hoger dan de dichtheid van informatie in bijv. drukwerken, zoals boeken. Dit wordt enerzijds veroorzaakt door de mogelijkheden van de EDP-informatiedragers om veel meer op te slaan dan in druktwerken mogelijk is, anderzijds doordat in EDP-faciliteiten veelal uitsluitend de belangrijkste informatie wordt opgeslagen: er wordt gekozen voor een gecondenseerde vorm.

De verborgenheid van de informatie komt het duidelijkst tot uiting door het ontbreken van de mogelijkheid om de opgeslagen informatie te onderwerpen aan een direkte visuele inspektie. Op de eerste plaats zijn er speciale programma's nodig om de informatie zichtbaar te maken en op de tweede plaats is de hoeveelheid informatie dikwijls zo groot, dat van serieuze menselijke inspektie niet te veel verwacht mag worden.

Aan de toegankelijkheid van EDP-faciliteiten, vooral van multi-user installaties, worden hoge eisen gesteld. Terminals op afstand worden noodzakelijk geacht voor een redelijke taakuitvoering in de diverse onderdelen van een bedrijf. Gezamenlijke informatie-opslag met toegankelijkheid voor iedereen brengt extra risico's met zich mee.

De vervalsingsmogelijkheden van in EDP-faciliteiten opgeslagen informatie zijn groter dan v.w.b. conventioneel opgeslagen informatie. EDP-vervalsingen zijn moeilijk te ontdekken, omdat de informatie op zich er nog heel plausibel uitziet. Aan de informatie is niet te zien, of een wijziging geautoriseerd of ongeautoriseerd is aangebracht.

EDP-informatiedragers zijn zeer vasthoudend. Ook wanneer de informatie is gewist, blijven er nog bitpatronen van voorheen opgeslagen informatie op achter. Eenmalig destruktief dumpen is in de regel niet voldoende om de informatie volledig te verwijderen.

4. CLASSIFICATIE VAN INFORMATIE

Maatregelen ter bescherming van informatie leggen beperkingen op aan de toegang er toe en het gebruik er van en hebben bovendien de eigenschap kostbaar te zijn. De maatregelen moeten daarom worden afgestemd op het belang van de informatie. Dit betekent dat informatie naar belangrijkheid moet worden geclassificeerd. Zowel de overheid als particuliere bedrijven kennen deze classificaties, die ook wel rubriceringen worden genoemd.

De overheid onderscheidt de volgende classificaties:

- zeer geheim (top secret), indien kennisneming der gegevens door onbevoegden zeer ernstige schade aan de veiligheid of het belang van de staat of van zijn bondgenoten kan veroorzaken.
- geheim (secret), indien kennisneming der gegevens door onbevoegden ernstige schade aan de veiligheid of het belang van de staat op zijn bondgenoten kan veroorzaken.
- vertrouwelijk of confidentieel (confidential), indien kennisneming der gegevens door onbevoegden nadeel aan de veiligheid of het belang van de staat of zijn bondgenoten kan veroorzaken.
- dienstgeheim (restricted), indien kennisneming der gegevens door onbevoegden, hoewel geen nadeel aan de veiligheid of het belang van de staat kunnen veroorzaken, uit beleidsoverwegingen ongewenst moet worden geacht.
- ongeclassificeerd (unclassified), indien geen bezwaar bestaat tegen kennisneming der gegevens door derden.

Particuliere bedrijven kennen overeenkomstige classificaties, met daarbij aangepaste motiveringen:

- bedrijfs-vertrouwelijk, speciaal toezicht (company confidential, special-control); onbevoegde kennisname leidt tot een getaxeerd verlies van omstreeks 10% van de jaarlijkse winst.
- bedrijfs-vertrouwelijk (company confidential); onbevoegde kennisname leidt tot een getaxeerd verlies van omstreeks 1% van de jaarlijkse winst.
- persoonlijk-vertrouwelijk (private confidential); onbevoegde kennisname kan een individu in moeilijkheden brengen of de publieke opinie t.o.v. het bedrijf (ongunstig) beïnvloeden.
- alleen voor intern gebruik (internal use only); dit is een verzameling informatie, die niet gemakkelijk past in een van de drie genoemde bedrijfsclassificaties, omdat deze informatie voor classificatie te licht is en niet voor algemene publikatie geschikt.
- beperkte verspreiding (limited distribution); is hier sprake van informatie, waarvoor het bedrijf een voorbehoud maakt t.a.v. verspreiding: het bedrijf wil de mogelijkheid hebben per voorkomende gelegenheid te beslissen.
- ongeclassificeerd (unclassified): vrij voor publikatie.

Opgemerkt moet worden dat de classificaties van het bedrijfsleven in vergelijking tot die van de overheid wel een duidelijke definitie hebben, maar dat kennelijk het bedrijfsleven zich daar minder van bewust is. De indruk bestaat, dat met name de classificatie "company confidential" nog wel eens voor lager geclassificeerd materiaal gebruikt wordt. Daar staat tegenover, dat de definities/motiveringen van de overheid zodanig zijn, dat er ruimte is voor eigen interpretatie.

In onderstaand overzicht zijn de overeenkomstige classificaties van overheid en bedrijfsleven naast elkaar opgenomen met hun Nederlandse en Engelse benaming en de bijbehorende motivering van beide instanties.

Overheid		Bedrijfsleven	
Motief	Benaming	Motief	Benaming
zeer ernstige schade aan landsbelang	zeer geheim (top secret)	bedrijfsvertrouwelijk, speciaal toezicht (comp.conf., special control)	schade ca. 10% jaarlijkse winst
ernstige schade aan landsbelang	geheim (secret)	bedrijfsvertrouwelijk (company confidential)	schade ca. 1% jaarlijkse winst
nadeel voor het landsbelang	vertrouwelijk, confidentieel (confidential)	persoonlijk-vertrouwelijk (private conf.)	pers. moeilijkheden of problemen met de publieke opinie
uit beleidsoverwegingen publikatie ongewenst	dienstgeheim (restricted)	intern gebruik (internal use only)	niet voor algemene publikatie geschikt
		beperkte verspreiding (limited distribution)	voorbehoud bij verspreiding
vrije publikatie	ongeclassificeerd	ongeclassificeerd	vrije publikatie

Overzicht 2: Vergelijking classificaties overheid v.s. bedrijfsleven

Informatie dient niet alleen geclassificeerd te worden, maar tevens dient te worden aangegeven, voor wie de informatie bestemd is. Dit komt voort uit het m.b.t. geclassificeerde informatie algemeen geldende "need-to-know" principe; indien informatie geclassificeerd is, dient het uitsluitend ter kennis te komen van diegenen, voor wie de informatie bestemd is.

5. AFWEERMECHANISMEN

Er zijn zeven afweermechanismen beschikbaar in het kader van de beveiliging van EDP-faciliteiten:

A. Materiële beveiliging heeft te maken met toegangscontrole tot ruimten en alle andere vormen van omgevingsbescherming.

B. Personeelsbeveiliging betreft selektie, screening, clearance van toezicht van medwerkers.

C. Vercijfering van gevoelige informatie wordt gehanteerd ter bescherming van de data-communicatie door de data onbegrijpelijk te maken voor iedereen, behalve de bevoegde geadresseerde.

D. Technische inspektie houdt zich bezig met de afweer tegen apparaten, die bedoeld zijn voor wederrechtelijke toeëigening van informatie en die geplaatst kunnen zijn in verder beveiligde ruimten.

E. Onderdrukking van compromiterende straling is wellicht nodig, omdat EDP-apparatuur akoestische en elektro-magnetische uitstraling produceert, die kan worden opgevangen en geanaliseerd.

F. Technische communicatie-inspektie betreft het onderzoek naar wederrechtelijke aansluitingen op verbindingen tussen de verschillende onderdelen van de EDP-faciliteiten.

G. Interne EDP-beveiliging is bedoeld als bescherming van de data in het systeem tegen onbevoegde toegang door het oprichten van hindernissen in het EDP-systeem.

Met het opsommen van bovengenoemde mechanismen wordt aangegeven, wat er gedaan moet worden om EDP-faciliteiten te beveiligen. Het is onmogelijk om de in deze mechanismen op te nemen maatregelen op te sommen, en indien het al mogelijk zou zijn, dan zou het onjuist zijn. De onmogelijkheid komt voort uit interpretatieverschillen omtrent het belang van de informatie en omtrent de betekenis van afweermaatregelen en uit verschillen in EDP-installaties en het gebruik ervan. De onjuistheid komt voort uit het beveiligingsbelang, om exacte maatregelen niet te publiceren.

Om enige indruk te geven, welke afweermechanismen voor welke classificatieniveaus effekt kunnen hebben m.b.t. beveiliging en ook welke afweermechanismen noodzakelijk zouden kunnen zijn bij de verschillende classificatieniveaus, bevat onderstaande tabel een overzicht, waarin deze relaties worden aangegeven. Met nadruk moet echter worden opgemerkt, dat het hypothetische relaties zijn, waarvan bij de uitwerking voor een specifieke toepassing kan blijken, dat ze niet met de werkelijkheid overeenkomen.

Beveiligingsmechanisme	Classificatie			
	zeer geheim	geheim	confidentiëel	dienstgeheim
Materiële beveiliging	X	X	X	X
Personeelsbeveiliging	X	X	X	X
Vercijfering	X	X		
Technische inspektie	X			
Onderdrukking van compromiterende straling	X			
Technische communicatie inspektie			X	
Interne EDP-beveiliging	X	X	X	X

Overzicht 3: Mogelijke relaties classificatie-afweermechanisme

6. ALGEMENE RICHTLIJNEN VOOR BEVEILIGING

Er is een twintigtal richtlijnen, betrekking hebbend op beveiliging in het algemeen, die op overeenkomstige wijze duiden op aandachtsgebieden binnen de computerbeveiliging. Tezamen geven deze richtlijnen een goed beeld van wat beveiliging inhoudt. De richtlijnen vragen aandacht voor:

a. regeling van bevoegdheden.
b. loyaliteit en betrouwbaarheid van de medewerkers.
c. hulpmiddelen, waardoor autorisaties éénduidig als geldig kunnen worden herkend.
d. identificatie van objekten, die bescherming verdienen.
e. vastlegging van beschermde objekten.
f. concentratie van waardevolle objekten, zodat deze beter kunnen worden beschermd.
g. inrichting van speciale gebieden rond de beschermde objekten.

h. verdediging van deze voor bescherming bedoelde speciale gebieden.
i. inspektiefaciliteiten v.w.b. de beschermde objekten.
j. toegangsregeling tot de beschermde objekten.
k. beperking van kennis m.b.t. de beschermde objekten en de beveiligingsmechanismen.
l. vermindering van het zicht op de beschermde objekten.
m. beperking van voorrechten m.b.t. de beschermde objekten.
n. vastlegging van de verantwoordelijkheid voor beschermde objekten.
o. registratie van aktiviteiten die te maken hebben met beschermde objekten.
p. uitvoering van dubbele controles op alle aktiviteiten m.b.t. beschermde objekten.
q. analyse van de registratie van de aktiviteiten m.b.t. beschermde objekten.
r. onderzoek van alle tegenstrijdigheden.
s. afstraffing van de afwijkingen.
t. voorbereiding tot terugval op back-up mogelijkheden.

7. BELEIDSOVERWEGINGEN MET BETREKKING TOT BEVEILIGING

In vergelijking met de richtlijnen in het voorgaande hoofdstuk hebben beleidsoverwegingen een wat algemener karakter en dienen als uitgangspunten voor de uit te werken regels. Enige gedachten worden gewijd aan overclassificatie, nieuwgierigheid, evenwicht van maatregelen, het "nooit-alleen" principe, de beperkte tijdsduur bij het bekleden van een aan beveiliging gerelateerde funktie, en het principe van scheiding van taken.

Overclassificatie van informatie is een gangbare methode bij twijfel aan de hoogte van het classificatieniveau. In eerste instantie schijnt het ook een goede methode, omdat de informatie in ieder geval veilig is. Bij nader inzien blijkt, dat overclassificatie het gevaar in zich bergt, dat deskundige gebruikers nonchalant met de veiligheidsmaatregelen gaan omspringen, omdat zij de verleende classificatie belachelijk vinden. Van de andere kant geldt, dat wanneer bij overclassificatie een ieder zich strikt aan de regels houdt, er gemakkelijk een onhandelbaar informatiesysteem ontstaat.

Onderclassificatie van informatie wordt soms toegepast, om de toegang tot de informatie gemakkelijker te maken en daardoor beter en sneller te kunnen werken. De aan zo'n handelwijze verbonden risico's zijn niet gering, omdat de veiligheid van de informatie niet meer gewaarborgd wordt. Zowel overclassificatie als onderclassificatie zijn daarom gevaarlijk en er moet dus veel aandacht besteed worden aan de juiste classificatie.

Nieuwsgierigheid is een onprettige eigenschap, wanneer die uitgaat naar de kennis van informatie, waarvoor de bevoegdheid niet aanwezig is. Nieuwsgierigheid kan leiden tot situaties, die rechtstreeks strijdig zijn met het "need-to-know"principe en dat betekent, dat akties die in het kader van nieuwsgierigheid plaatsvinden, kunnen worden beschouwd als opzettelijke schending van de beveiliging.

Evenwicht van maatregelen m.b.t. beveiliging is even belangrijk als evenwicht in alle andere omstandigheden. Hiervoor gelden twee argumenten. Ten eerste geeft evenwicht van maatregelen een keten met schakels van gelijke sterkte, en zo'n keten is in het algemeen moeilijk te breken. Ten tweede dwingt zo'n keten bij gebruikers respekt af, waardoor deze gebruikers sneller geneigd zijn zich aan de beveiligingsmaatregelen te houden. Om evenwicht van beveiligingsmaatregelen te bewerkstelligen, is het nodig om deskundigen in te schakelen. Het is om deze reden, dat security-analisten een goede toekomst voorspeld wordt.

Het "nooit-alleen" principe is van belang, omdat toepassing daarvan de kans op misdragingen m.b.t. geclassificeerde informatie aanzienlijk verkleint. In conventionele beveiligingsomgeving wordt dit principe algemeen toegepast. In de computeromgeving wordt er nog al eens van afgeweken vanwege gebrek aan personeel en/of om het werk niet "onnodig" op te houden. Het uitbetalen van een bedrag van ƒ 10,- is soms met betere garanties omkleed dan het manipuleren met de kostbaarste informatie van een bedrijf.

Beperkte tijdsduur voor het bekleden van funkties, waarin computer-medewerkers kennis (kunnen) nemen van geclassificeerde informatie heeft aantrekkelijke kanten. Niemand zou zo lang zo'n funktie mogen bekleden, dat hij denkt dat hij onaantastbaar is of zijn taken volledig voorspelbaar zijn. De mate waarin zo'n funktie kan c.q. moet worden gewisseld, hangt af van de beschikbaarheid van personeel c.q. van de dreiging die uitgaat van de mogelijke misdragingen van een medewerker, die zo'n funktie bekleedt.

De laatste en ook zeer belangrijke beleidsoverweging is de scheiding van taken. Het is duidelijk dat wanneer één medewerker alle taken binnen een rekencentrum vervult, de kans op moeilijkheden het grootst is. Daar waar scheiding van taken in een computercentrum niet tot de mogelijkheden behoort, dient een zorgvuldige analyse te worden gemaakt van de risico's, die daardoor gelopen worden. Extra controle-maatregelen zijn daarbij aanbevelenswaardig.

8. SLOTOPMERKING

Het gebied van beveiligingsmaatregelen vormt een netwerk en een netwerk heeft mazen. De ervaring leert, dat er altijd lieden zijn met als doelstelling de mazen te benutten. Maatregelen zijn bovendien altijd gebaseerd op een bepaalde situatie, en in de computeromgeving kunnen situaties snel wijzigen. Wijzigingen kunnen ontstaan door veranderingen in (een van) de toepassing(en), door veranderingen in de EDP-faciliteiten of door veranderingen in het gebruik ervan. De beveiligingsmaatregelen moeten met de gewijzigde situatie mee veranderen.

Daarom is voortdurende waakzaamheid geboden. Waakzaamheid vraagt continuïteit en een alerte houding van alle medewerkers, die het belang van de informatie en daarmee het belang van de onderneming ter harte gaat. Van een pakket maatregelen gaat een in slaap sussende werking uit, die in snel veranderende situaties als gevaarlijk voor de onderneming moet worden gekenschetst. Maatregelen zijn nodig, om er in de praktijk van alle dag mee te kunnen werken, maar deze maatregelen blijven alleen zinvol, als ze continu worden getoetst aan de voorkomende problemen en de optredende wijzigingen. Dat kan alleen indien alle belanghebbenden deze waakzaamheid betrachten.

X. HOE DEEL JE EEN GEHEIM?

Terwijl we ons in de andere hoofdstukken beziggehouden hebben met cryptografische systemen in de stricte zin, zullen we nu een aanverwant onderwerp behandelen. De geïnteresseerde lezer verwijzen we naar [1] en [3].

In [2] komt men het volgende probleempje tegen:

> Elf wetenschappers werken samen aan een geheim project. Zij wensen de documenten op te sluiten in een kabinet op een zodanige manier dat het kabinet geopend kan worden dan en slechts dan als er zes of meer wetenschappers aanwezig zijn. Wat is het kleinste aantal sloten dat nodig is? Met hoeveel sleutels voor deze sloten moet elke wetenschapper rond lopen?

Het is niet zo moeilijk in te zien dat er $\binom{11}{5} = 462$ sloten nodig zijn en dat elke wetenschapper met $\binom{10}{5} = 252$ sleutels moet rondlopen. Zowaar, een onpractische situatie!

We zullen het probleem nu enigszins generaliseren.

De geheime informatie D (b.v. de sleutelcombinatie) moet op de een of andere manier opgesplitst worden in stukjes $D_1,\ldots,D_n$ over n deelnemers zodanig dat:

1) kennis van k of meer van de stukjes D_i D gemakkelijk berekenbaar maakt.
2) kennis van k - 1 of minder van de stukjes D_i D compleet onbepaald laat, in de zin dat alle mogelijke waarden even waarschijnlijk zijn.

Zo'n schema wordt een (k,n) <u>treshold schema</u> genoemd.

Een mogelijke toepassing is de situatie waarbij D de sleutel van een cryptografisch systeem is. D wil je niet op één plaats opbergen, maar in stukjes

op meerdere plaatsen.

De volgende constructie van zo'n treshold scheme komt uit [3].

Kies een polynoom

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1} ,$$

waarbij $a_0 = D$ en waarbij de andere a_i's willekeurig gekozen zijn. Bereken vervolgens de stukjes deelinformatie $D_i = f(i)$. $1 \leq i \leq n$.

Dit schema voldoet aan de eisen 1) en 2). Immers stel dat de stukjes D_i bekend zijn voor $i = i_1, \ldots, i_k$ (allen verschillend), dan is $f(x)$ eenvoudig te berekenen met de Lagrange interpolatieformule

$$\sum_{j=1}^{k} D_{i_j} \prod_{\ell=i}^{k} \frac{(x - i_\ell)}{(i_j - i_\ell)} .$$

Inderdaad $f(x)$ en bovenstaand polynoom hebben beide graad $k - 1$ en zijn identiek in de punten $i_1, \ldots, i_k$. Bijgevolg zijn het dezelfde polynomen.

De waarde D is nu gemakkelijk uit te rekenen, immers $D = f(0)$.

Als slechts $k - 1$ waardes D_i bekend zijn kan er nog elke waarde D^* uitkomen. Om dit in te zien past men bovenstaande interpolatieformule toe met de bekende D_j, $i = i_1, \ldots, i_{k-1}$ en $i_k = 0$, $D_0 = D^*$.

In [2] kan men een generalisatie van dit schema vinden, die je in staat stelt D te vinden als sommige van de D_i's (met opzet eventueel) verkeerd zijn doorgegeven. Natuurlijk mogen niet te veel D_i's verkeerd zijn en moeten we in principe meer dan k D_i's kennen. Meer precies als we D_i kennen voor s verschillende waarden van i en t hiervan zijn fout dan kan men D toch nog bepalen als voldaan is aan $2t \leq s - k$. Merk op dat als $t = 0$ en $s = k$ we in het vorige geval terug zijn.

Aangezien deze constructie gebruik maakt van algebraische coderingstheorie, voert het te ver om dit systeem hier te behandelen.

REFERENTIES

[1] Liu, C.L., Introduction to combinatorial mathematics, McGraw-Hill, 1968.

[2] McEliece, R.J. en D.V. Sarwate, On sharing secrets and Reed-Solomon codes, Comm. of the ACM, 24, 1981, 583-584.

[3] Shamir, A., How to share a secret, Comm. of the ACM, 22, 1979, 612-613.