Montfortcollege Rotselaar
LW6

Schooljaar 2006–2007

Cryptografie

Frederic Vleminckx

Begeleider: Mr. Olaerts

Eindwerk Wiskunde

Woord vooraf

Cryptografie is een domein dat mij al van kleins af interesseerde. Ik droomde er dan ook altijd van om detective te worden. De laatste jaren is die ambitie wat vermindert in tegenstelling tot de interesse voor cryptografie die de laatste tijd is gestegen door één van mijn hobby's namelijk programmeren. Omdat ik nu eens de kans kreeg om volledig en in detail op dit onderwerp in te gaan, heb ik het aangepakt en voìla hier is het.

Na vele slapeloze nachten om toch maar die deadline te halen van vandaag, is het mij eindelijk gelukt. Maar het was zeker niet gelukt zonder de technische hulp van Mr. Olaerts, de mentale steun van de vrienden en de correctionele steun van mijn mama. Bedankt allemaal!

Frederic Vleminckx
Herent 14 Maart 2007

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Cryptografie

Cryptografie samengesteld uit 2 Griekse woorden: 'crypto' (verborgen) en 'grafie' (schrijven), betekent letterlijk: "geheimschrijven".
De definitie luidt als volgt:

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

Dit geheimschrift heeft al een hele lange evolutie achter de rug en is daarom ook zo uitgebreid. In dit werk wil ik dan ook vooral de nadruk leggen op het overzicht van de meest gebruikte technieken, inclusief de zeer oude technieken en de wiskundige logica erachter.

Zonder deze wiskundige logica zou cryptografie er wellicht veel anders hebben uitgezien en daarom vormt wiskunde ook de basis van geheimschrift. Zo wordt de informatie, die gecodeerd moet worden met wiskundige formules, omgezet naar de gecodeerde informatie. Men streeft ernaar om wiskundige formules te vinden die niet of zeer moeilijk inverteerbaar zijn. Alleen met de sleutel zou het mogelijk moeten zijn de gecodeerde tekst als oorspronkelijke tekst terug te kunnen lezen.

Voor ik u loslaat in dit werk wil ik u toch even voorbereiden met wat extra informatie over begrippen en woorden. Dit wordt allemaal uitgelegd in de terminologie.

1.2 Terminologie

Om een stroom van informatie te hebben, is er een zender en een ontvanger nodig. De zender, die de begrijpbare boodschap (plaintekst oftewel 'klare tekst') wil versturen, moet de plaintekst vercijferen (versleutelen oftewel coderen) tot een onbegrijpbare boodschap (cijfertekst).

$$E(P) = C$$

Hier wordt de plaintekst (P) vercijferd tot de cijfertekst (C). Dit proces heet encryptie (E, Engels: encryption).

$$D(C) = P$$

Dit is net omgekeerd. Hier wordt de cijfertekst ontcijferd tot de plaintekst. Dit heet decryptie (D, Engels: decryption).

Om encryptie en decryptie handig te houden, wordt er gebruik gemaakt van sleutels (key, k) zodat je makkelijk het bericht kan vercijferen of ontcijferen.
Meer hierover volgt bij de cryptografische technieken zelf.

1.3 Hulpmiddelen

Bij het schrijven van dit werk heb ik veelvuldig gebruik gemaakt van het open-source programma 'Cryptool', ontwikkeld door Bernhard Esslinger. Met het programma is het mogelijk om verschillende encryptie en decryptie systemen te analyseren en op een efficiënte wijze uit te voeren.

Hoofdstuk 2

Wiskundige begrippen

In dit hoofdstuk gaat het voornamelijk over wiskundige logica die toch wel de basis vormt voor vele cryptografische systemen.

2.1 Priemgetallen

2.1.1 Inleiding

2, 3, 5, 7, 11, 13, 17, 19, 23, 29 zijn de eerste 10 priemgetallen. Wat hen zo speciaal maakt, volgt in dit hoofdstuk. Om beter te begrijpen wat net een priemgetal is, volgt hier de definitie:

> Een *priemgetal* (of priem) is een natuurlijk getal, verschillend van 1, dat alleen maar deelbaar is door 1 en zichzelf.

Belangrijk hierbij is dat het getal *1* niet bij de priemgetallen gerekend wordt. Hieronder volgen enkele eigenschappen en stellingen:

Stelling 2.1 Johann Gauss*: Een positief geheel getal kan op precies één manier (afgezien van de volgorde van de factoren) geschreven worden als product van priemgetallen.*

> Als voorbeeld ontbinden we het getal 420 in priemfactoren:
> $420 = 2.2.3.5.7$

Het probleem aan ontbinden in priemfactoren is dat er geen echte formule voor is. De enige mogelijkheid is diegene van trial-and-error waar je één voor één alle priemgetallen uitprobeert tot het getal ontbonden is. Daarom duurt het ook extreem lang om een groot getal te ontbinden in priemfactoren in tegenstelling tot het vermenigvuldigen van priemgetallen. Dit speelt een belangrijke rol bij codering en decodering.

Stelling 2.2 Euclides *(in ±325 vóór Chr.): Er bestaan oneindig veel priemgetallen.*

Bewijs uit het ongerijmde:

> Stel dat $p_1, p_2, ..., p_n$ de enige priemgetallen zijn.
> Het getal $p_1.p_2.p_n + 1$ is door geen van deze getallen deelbaar, en moet dus hetzij zelf een priemgetal zijn, hetzij een ander priemgetal als deler hebben, wat in tegenspraak is met onze veronderstelling.

Dus het idee dat er 'eindig' veel priemgetallen zijn leidt tot tegenspraak. De verzameling van de priemgetallen is *oneindig*.

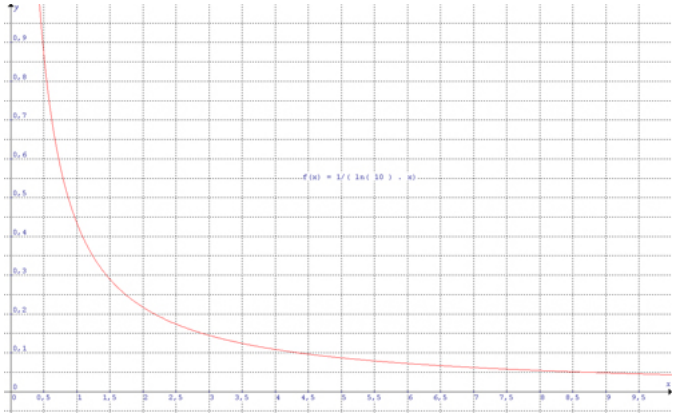
2.1.2 Distributie van priemgetallen

Hoe zit het nu met de verdeling van de priemgetallen? Hadamard (Franse) en La Vallée-Poussin (Leuvense wiskundige) bewezen in 1896 dat:

$$\lim_{x \to +\infty} \frac{\pi(x)}{\frac{x}{ln(x)}} = 1 \tag{2.1}$$

De $\pi(x)$ staat voor voor de priem-tel-functie en heeft niets met het getal π te maken. De priem-tel-functie telt het aantal priemgetallen minder dan of gelijk aan x. De kans dat een getal met n cijfers priem is, moet ongeveer gelijk zijn aan:

$$\frac{1}{ln(10^n)} = \frac{1}{n.ln(10)} = \frac{0.434...}{n} \tag{2.2}$$



Aan de hand van deze grafiek zien we duidelijk dat de kansen op priem bestaande uit weinig cijfers, zeer groot is ($n = 1$, 43,4% kans op priem). En naarmate het aantal cijfers toenemen, de kans op een priemgetal verkleint.

2.1.3 Priemgetal vinden

Om te verifiëren of een getal wel degelijk een priemgetal is, wordt een bepaald algoritme gebruikt. Men maakt onderscheid tussen de algoritmes die 100% wiskundig correct een priemgetal kunnen aangeven *(deterministische methodes)* en de overige methodes waar er altijd een bepaalde foutmarge is *(probabilistische methodes)* maar deze zijn dan ook veel sneller.

De volgende algoritmes gebruikt men vaak:

- Miller-Rabin (probabilistisch)
- Fermat (probabilistisch) (voorbeeld 2.10 op pagina 7)
- Solovay-Strassen (probabilistisch)
- AKS (deterministisch)

Bij cryptografie worden meestal priemgetallen gekozen van 512 bits (ongeveer 154 decimale cijfers) en 1024 bits (ongeveer 308 decimale cijfers), deze worden gegenereerd door een willekeurig getal tussen bepaalde grenzen te testen tegen een van de bovenstaande algoritmes. Een voorbeeld is te vinden in 2.3.2 op pagina 8. De snelste algoritmes doen over een getal van 200 cijfers enkele minuten. Dit staat wel in schril contrast met ontbinden in priemfactoren waar het miljarden jaren zou duren om een getal van 200 cijfers in priemfactoren te ontbinden. En daar ligt net de kracht van vele cryptografische technieken.

2.2 Modulo Rekenen

2.2.1 Inleiding

Modulo rekenen of klok-rekenen is een bewerking waarbij er met de rest wordt gerekend. De *"modulo"* is de rest van een getal na deling door een ander getal.

> Bijvoorbeeld: $27 = 5.5 + 2$, dus 2 is de rest van 27 na deling door 5.
> Dus: $27 \equiv 2 \pmod 5$

In het alledaagse leven komen we veel in contact met modulo rekenen. Namelijk het uur waar de modulus 12 is (mod 12) maar de digitale versie van het uur 24 uur is.

> Bijvoorbeeld: 16 uur $= 1.12 + 4$, dus 4 is de rest van 16 na deling door 12.
> Dus: $16 \equiv 4 \pmod{12}$

Definitie: $\forall\, n \in \mathbb{N}_0$ en $x, y \in \mathbb{Z}$.

$$\begin{aligned} &x \equiv y \pmod n \\ &x = q.n + y \text{ (q is een willekeurig natuurlijk getal.)} \end{aligned} \tag{2.3}$$

In woorden: x is congruent aan y modulo n. (Dit is een equivalentie relatie [1])

2.2.2 Bewerkingen

Hoeveel is 328.545 in mod 7?

$$\begin{aligned} 328.545 \pmod 7 &= 178760 \pmod 7 \\ \frac{178760}{7} &= 25537 + 1 \\ 328.545 &\equiv 1 \pmod 7 \end{aligned}$$

Als we grotere getallen gebruiken wordt het alsmaar moeilijker. Een manier om het te vergemakkelijken is de volgende:

$$\begin{aligned} 328 \equiv 6 \pmod 7 \qquad & 545 \equiv 6 \pmod 7 \\ 328.545 &\equiv 6.6 \pmod 7 \\ &\equiv 36 \pmod 7 \\ &\equiv 1 \pmod 7 \end{aligned}$$

Er geldt dus:

$$\begin{aligned} a \equiv b \pmod m \qquad & c \equiv d \pmod m \\ a.c &\equiv b.d \pmod m \end{aligned} \tag{2.4}$$

2.3 De stellingen van Euler en Fermat

2.3.1 Euler

Volgens de Euler stelling geldt: $\forall\ n \in \mathbb{N}_0, a \in \mathbb{Z}$ met ggd(n,a) = 1 [2]

$$a^{\varphi(n)} \equiv 1 \pmod n \tag{2.5}$$

$$\begin{aligned} \varphi(n) = &\text{ het } \textit{aantal} \text{ natuurlijke getallen} \\ &\text{die onderling ondeelbaar zijn met n. } [3] \end{aligned} \tag{2.6}$$

Deze stelling wordt gebruikt om de berekening van hoge machten mod(n) te vereenvoudigen. Als voorbeeld gebruiken we $7^{541} \pmod 8$ ($\varphi(8) = 4$).
De getallen 7 en 8 (a, b) zijn "relatief priem" omdat hun ggd = 1. Dus kunnen we de stelling van Euler (2.5) toepassen. Als n een priemgetal zou zijn geweest, zouden we de stelling van Fermat (2.10 op de pagina hierna) kunnen toepassen.

[1] Een relatie is equivalent als de relatie *reflexief, symmetrisch en transitief* is.
[2] Getallen heten "relatief priem" als ze geen gemeenschappelijk factoren bevatten. ggd(a,b) = 1
[3] Het aantal getallen dat niet groter is dan n en die relatief priem zijn met n.

$$\begin{aligned} 7^{541} &= 7^{4.135+1} \\ &\Downarrow 7^4 \equiv 1 \pmod 8 \\ (7^4)^{135}.7^1 &\equiv 1^{135}.7^1 \\ 7 &\equiv 7 \pmod 8 \end{aligned}$$

De stelling van Euler heeft ook 2 consequenties:

Als p een priemgetal is, geldt:

$$\varphi(p) = (p-1) \tag{2.7}$$

En geldt ook:

$$\varphi(p.q) = \varphi(p) \,.\, \varphi(q) \tag{2.8}$$

Als we de 2 vorige formules toepassen bij 2 priemgetallen geldt:

$$\varphi(p.q) = (p-1)(q-1) \tag{2.9}$$

Deze laatste formule is zeer belangrijk bij het RSA-systeem en volgt later nog.

2.3.2 Fermat

De *congruentie* van Fermat is een specifiek geval van de stelling van Euler (2.5). Dit geval geldt enkel wanneer $n = p$ (priemgetal).

$\forall\, p \in$ priemgetallen, $a \in \mathbb{Z}$ met $p \nmid a$ (p deelt a niet)

$$a^{p-1} \equiv 1 \pmod p \tag{2.10}$$

Uit deze stelling volgt de *kleine* stelling van Fermat:

$$\begin{aligned} a^{p-1} &\equiv 1 \pmod p \\ &\Downarrow \\ a^{p-1} &= 1 + q.p \\ &\Downarrow \text{(vermenigvuldigen met a)} \\ a^p &= a + a.q.p \\ &\Downarrow \\ a^p &\equiv a \pmod p \end{aligned} \tag{2.11}$$

Deze stelling is zeer interessant om na te gaan of een getal al dan niet een priemgetal is. [4]

We testen of het nummer 251 een priemgetal is:

$$\begin{aligned} a^{p-1} &\overset{?}{\equiv} 1 \pmod{p} \\ &\Downarrow \\ 2^{250} &\overset{?}{\equiv} 1 \pmod{251} \\ &\Downarrow \\ 2^{250} &\overset{!}{\equiv} 1 \pmod{251} \quad \Rightarrow 251 \text{ is een priemgetal} \end{aligned}$$

Nu kijken we na of het getal 265 een priemgetal is:

$$\begin{aligned} a^{p-1} &\overset{?}{\equiv} 1 \pmod{p} \\ &\Downarrow \\ 2^{264} &\overset{?}{\equiv} 1 \pmod{265} \\ &\Downarrow \\ 2^{264} &\equiv 16 \pmod{265} \quad \Rightarrow 265 \text{ is geen priemgetal} \end{aligned}$$

Deze methode op het testen van primialiteit is niet 100% nauwkeurig want de stelling van Fermat, waar deze test op steunt, geldt niet altijd in de andere richting.

Als voor zekere gehele a en k geldt dat: $a^k = a \mod k$, dan is niet noodzakelijkerwijs k een priemgetal. Een getal q dat geen priemgetal is, maar waarvoor geldt dat $a^q = a \mod q$ (voor zekere a) wordt een *'pseudo-priemgetal'* genoemd. Als q de eigenschap heeft dat het bovengenoemde geldt voor willekeurige a, dan heet q een *'Carmichael getal'*.

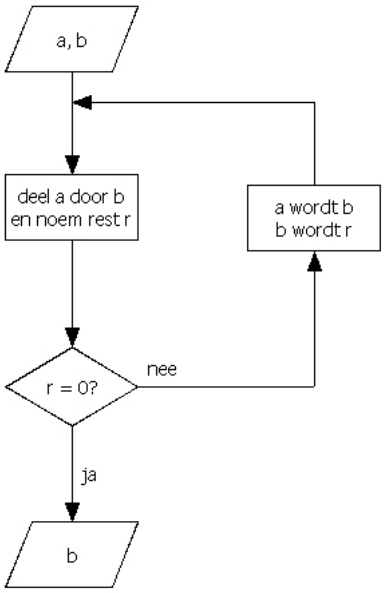
Er is bewezen dat er oneindig veel pseudo-priemgetallen bestaan. Echter, binnen de gehele getallen zijn de pseudo-priemgetallen wel 'dunner gezaaid' dan de priemgetallen.

2.4 Algoritme van Euclides

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

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

[4] In de realiteit worden er random getallen gegenereerd, vervolgens nagecheckt met een algoritme (in de meeste gevallen dat van Fermat) om snel een groot priemgetal te vinden.



Figuur 2.1: Algoritme van Euclides

In woorden ziet het er zo uit:

1. Noem het grootste van de beide getallen a, het andere b.
2. Trek b net zo vaak van a af totdat er 0 over blijft of een getal kleiner dan b.
3. Wanneer er 0 over blijft zijn we klaar, en is b de GGD.
4. Zo niet, herhaal dan het algoritme met b en wat er van a over is.

Als voorbeeld berekenen we de $GGD(654321, 123456)$:

$$\begin{aligned}
654321 &= 5.123456 + 37041 \\
123456 &= 3.37041 + 12333 \\
37041 &= 3.12333 + 42 \\
12333 &= 293.42 + 27 \\
42 &= 1.27 + 15 \\
27 &= 1.15 + 12 \\
15 &= 1.12 + 3 \\
12 &= 4.3 + 0 \\
GGD(654321, 123456) &= 3
\end{aligned}$$

Hoofdstuk 3

Cryptografische technieken

Hieronder volgt een overzicht van de cryptografische technieken die tot nu toe bekend zijn. Alles is ingedeeld volgens de vorm van de sleutel.

Om duidelijkheid te scheppen in de afkortingen gebruik ik vanaf nu voor een codering (encoding) E, voor de decodering (decrypting) D en k voor de sleutel (key).

3.1 Inleiding

In de cryptografie zijn er duidelijk 2 verschillende vormen te onderscheiden: *symmetrische* en *asymmetrische*.

Bij **een symmetrische versleuteling** is de sleutel om te coderen dezelfde als om te decoderen.

$$\begin{aligned} E_k(P) &= C \\ D_k(C) &= P \end{aligned} \tag{3.1}$$

Er geldt dus ook:

$$D_k(E_k(P)) = P$$

Daarentegen bij de **asymmetrische versleuteling** gebruikt men 2 verschillende sleutels: ééntje is publiek en kan door iedereen gebruikt worden, de andere is een privé sleutel die alleen de eigenaar kent. Als deze eigenaar nu een gecodeerd bericht wil sturen, moet hij de publieke sleutel van de ontvanger gebruiken om het bericht te coderen en te versturen. Dit gecodeerd bericht wordt door deze ontvanger met zijn privé sleutel gedecodeerd tot de oorspronkelijke tekst die de eigenaar verstuurd had.

$$\begin{aligned} E_{k1}(P) &= C \\ D_{k2}(C) &= P \end{aligned} \tag{3.2}$$

Er geldt dus ook:

$$D_{k2}(E_{k1}(P)) = P$$

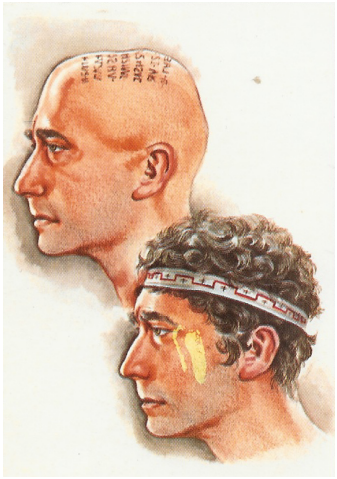
Hieruit blijkt dat asymmetrische versleutelingen veel complexer zijn dan symmetrische en daarom is de rekenkracht om een asymmetrische versleuteling te berekenen veel groter.

3.2 Pré cryptografie

In het pré-cryptoriaans tijdperk dat terug gaat tot 600 vóór Chr. werd vooral gewerkt om met de vorm en de manier van een bericht geheim over te dragen van verzender op ontvanger. Deze geheime uitwisseling noemt men *steganografie*. Steganós (Gr.) betekent bedekken, grafein (Gr.) schrijven, dus bedekt schrijven.

3.2.1 Slaven

De Griekse geschiedschrijver Herodotus vermeldt dat een zekere Histaeius een geheim bericht had verzonden door het haar van één van zijn slaven te scheren en vervolgens een boodschap op zijn hoofdhuid te tatoeëren. Ten slotte wachtte hij tot zijn haar weer aangroeide en transporteerde dan het bericht.



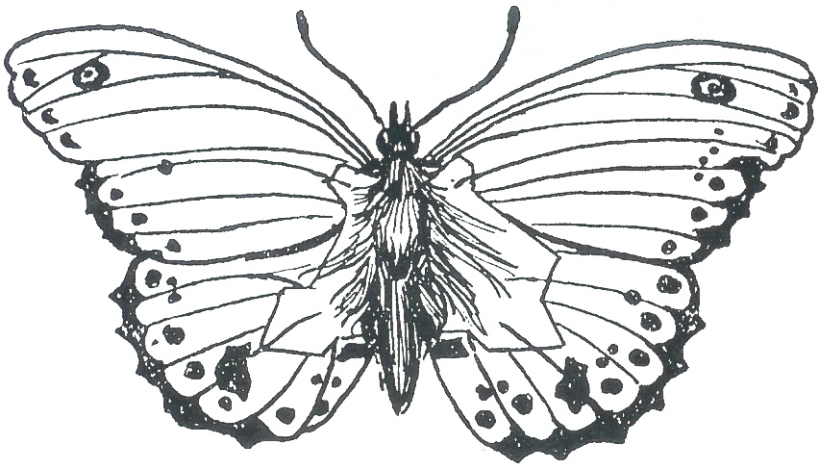
Figuur 3.1: Slaven

3.2.2 Perkamentrollen

De oude Grieken en Romeinen stuurden brieven in de vorm van perkamentrollen die om een stok (de skutalè) gewikkeld werden. Alexander de Grote bedacht een manier om geheime boodschappen over te brengen door zijn woorden met zorg te kiezen. De methode is duidelijk te zien op de afbeelding.

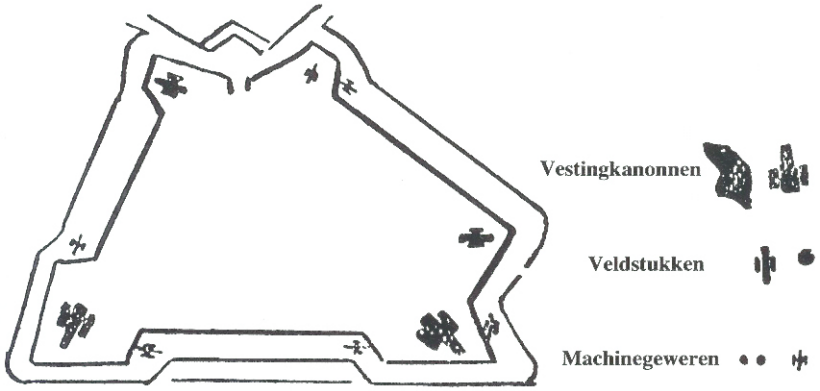
Figuur 3.2: Perkamentrollen

3.2.3 Tekeningen



Figuur 3.3: Vlinder

Een andere mogelijkheid is om je bericht te verwerken in tekeningen. Lord Baden Powell, de oprichter van de padvinderijbeweging in Engeland, was spion tijdens de boerenoorlog (1899-1902). Hij deed zich voor als vlinderverzamelaar en deed alsof hij zeldzame vlinders tekende. Maar in werkelijkheid schetste hij de vijandelijke versterkingen.



Figuur 3.4: Vijandelijke versterkingen

De vlekken op de vleugels tussen de lijnen betekenden niets, maar die op de lijnen geven de aard en grootte van de bewapening aan. De positie van elk wapen is op de plaats binnen de omtrek van het fort op de vlinder waar de lijn met de vlek eindigt. De kop van de vlinder wijst naar het noorden.

Deze techniek wordt zelfs nog vandaag de dag gebruikt. De geheime tekst wordt dan gewoon aan een digitaal bestand gevoegd zoals een foto of een audio bestand. Omdat men niets merkt aan de kwaliteit van het bestand, vermoedt men ook niet dat er een geheime tekst in verborgen zit.

3.2.4 Onzichtbare inkt

Onzichtbare inkt zelf maken is een fluitje van een cent. Je gebruikt citroensap, melk of een suikeroplossing om je boodschap te schrijven en je laat dit drogen. Als je de verstopte boodschap wil lezen, moet je enkel nog het blad verhitten. Vroeger gebruikte men als geheime inkt voornamelijk cobaltchloride. Bij wijze van voorbeeld heb ik zelf de test gedaan met melk, zie hieronder.

Figuur 3.5: Onzichtbare inkt met melk

3.3 Symmetrische key cryptografie

3.3.1 Het Caesar cijfer

Inleiding

Gaius Julius Caesar is de grondlegger van de symmetrische versleuteling. Omstreeks 50 vóór Chr. wou Caesar komaf maken met het probleem dat zijn boodschappers door de vijand gevangen genomen werden waardoor deze het bericht te weten kwamen. Hij wou het bericht onleesbaar maken voor de vijanden die het in handen konden krijgen maar wel leesbaar voor de ontvanger. Hij bedacht het Caesar Cijfer (Caesar Cipher) waar de letters van het alfabet met k plaatsen opgeschoven werden. De enige sleutel die dan nodig is om de boodschap te ontcijferen, is deze k.
Wiskundig gaat het als volgt (k is key, P is plaintekst, C is cijfertekst):

$$C \equiv (P + k) \pmod{26} \tag{3.3}$$

Figuur 3.6: Caesar schijf

Voorbeeld

Als voorbeeld nemen we $k = 3$ en als plaintekst: 'cryptografie is leuk'.
Het alfabet wordt verschoven met 3 en zo ziet het er dan uit:

Plain-alfabet:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cijfer-alfabet:	DEFGHIJKLMNOPQRSTUVWXYZABC

De boodschap wordt verschoven volgens het nieuwe alfabet:

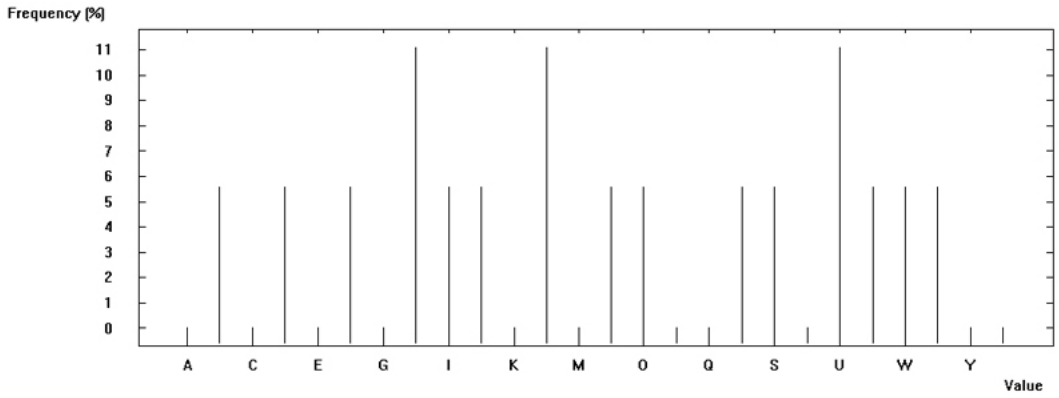
Plaintekst:	CRYPTOGRAFIE IS LEUK
Cijfertekst:	FUBSWRJUDILH LV OHXN

Kraakbaarheid

Theoretisch is deze vorm van coderen niet te kraken. Maar omdat k hoogstens 26 kan zijn, zijn er ook maar 26 sleutels mogelijk. Door het alfabet te vergroten, met cijfers en leestekens, kun je het aantal mogelijke sleutels vergroten. Omdat de plaintekst bij een Caesar versleuteling even veel karakters bevat als de cijfertekst en opgesteld is in een taal (Nederlands) is er nog een mogelijkheid om in plaats van een *brute-force-attack (zie 5.4 op pagina 38)*, aan de hand van een letter-frequentietabel de k af te leiden uit het gecodeerde bericht.

Figuur 3.7: Frequentie tabel (Nederlands)

Als we de gecodeerde tekst analyseren merken we dit:



Figuur 3.8: Analyse 'FUBSWRJUDILH LV OHXN'

De hoogste pieken van de analysetabel zijn te vinden bij 'H', 'L', 'U'. We kunnen nu deze analyse verschuiven tot we op het punt zijn waar de freqentietabel de analysetabel evenaart. Als we de analysetabel 3 letters terugschuiven dan zien we dat ze overeenkomt met de frequentietabel. k is dus 3 in dit geval.

ROT13

Wanneer in het Caesar Cijfer de k gelijk is aan 13 (oftewel de helft van het gebruikte alfabet) dan gebruikt men hiervoor de naam *ROT13*. Er is een kleine eigenschap verbonden aan deze heel onveilige versleuteling:

$$E_{13}(E_{13}(P)) = P \tag{3.4}$$

Men gebruikt deze codering vooral op fora en websites waar veel spoilers (ontknopingen van films of spannende elementen) neergeschreven staan. Zo kan de nietsvermoedende lezer niet geconfronteerd worden met een spoiler en kan de geïnteresseerde lezer eenvoudigweg de decodering uitlezen.

3.3.2 Enkelvoudige substitutie

Dit encryptiesysteem zoals de naam zelf zegt, vervangt (substitueert) een enkelvoudig teken door een ander. Wat dus een eenvoudigere manier is dan de Caesar Cijfer methode. Nu is er niet 1 key die bepaalt hoeveel plaatsen het alfabet opschuift, maar een hele translatietabel waarmee de tekst gecodeerd moet worden.

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

Plaintekst:	cryptografie is leuk
Cijfertekst:	ekbhmgukayot ol stwq

Om de tekst te decoderen is de hele tabel die als sleutel fungeert, nodig.

Net als bij het Caesar Cijfer is het mogelijk om de key snel te kraken met een brute-force-attack of met de letter-frequentietabel.

3.3.3 Het Vigenère systeem

Inleiding

Het Vigenère systeem is een uitbreiding op het Caesar Cijfer en is ontwikkeld door Blaise de Vigenère die in 1585 een boek publiceerde met het 'Vigenère tableau' of zoals we het nu noemen het 'Vigenère vierkant'. Dit Vigenère vierkant bestaat uit 26 alfabetten die telkens een plaats naar links opschuiven zoals op de tekening.

	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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 3.9: Vigenère vierkant

Voorbeeld

Je kiest een sleutelwoord (key) en een plaintekst die je wil versleutelen.

Key (k):	SECRET
Plaintekst (P):	DAS IST GEHEIM

Vervolgens moet de key verlengd worden tot de lengte van de plaintekst.

SECRETSECRET
DASISTGEHEIM

De cryptograaf volgt het alfabet aan de bovenkant van de tabel tot hij de eerste letter van het ongecodeerde bericht bereikt, dan gaat hij in deze kolom naar beneden tot hij de lijn vindt die begint met de eerste letter van het sleutelwoord. Op dit kruispunt vindt hij de eerste letter van de code. Dit proces wordt herhaald tot het volledige bericht is omgezet in code.

Er is ook een wiskundige manier om de cijfertekst (C) te berekenen:

$$C \equiv (P + k) \pmod{26} \tag{3.5}$$

Elk apart karakter van het woord wordt genummerd volgens het alfabet.
Als voorbeeld bereken ik de eerste letter van de cijfertekst. $3 + 18 \pmod{26} \equiv 21 \equiv V$ Als we dit herhalen krijg je uiteindelijk:

Cijfertekst: VEUZWMYIJVMF

Kraakbaarheid

Lang geloofde men dat dit cryptografische systeem onkraakbaar was omdat het voor dezelfde letter een andere gecodeerde letter kon zijn. Dus het kraken met de hulp van een frequentietabel was niet meer te doen. Maar in 1854 werd het Vigenère systeem gekraakt door de Britse cryptograaf Charles Babbage. Hij bemerkte dat er overeenkomsten waren in de herhaling van de plaintekst en die van de cijfertekst.

De sterkte van het Vigenère systeem is dat voor éénzelfde letter verschillende gecodeerde letters mogelijk zijn afhankelijk van het sleutelwoord. In onderstaande tekening is duidelijk dat er voor de codering van de letter 'e' met sleutelwoord 'key' meerdere coderingen zijn. De mogelijkheden zijn 'I O C', 3 in dit geval omdat het sleutelwoord 3 letters bevat.

	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

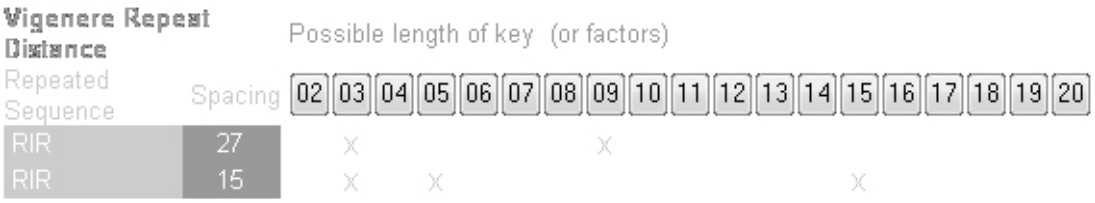
Figuur 3.10: Vigenère vierkant van I, O, C

Sleutelwoord:	k	e	y	k	e	y	k	e	y	k	e	y	k	e	y	k	e	y	k	e	y	k	e	y	k	e	y
Plaintekst:	h	e	t	m	e	i	s	j	e	i	n	h	e	t	b	a	d	e	n	i	n	h	e	t	b	e	d
Cijfertekst:	r	i	r	w	i	g	c	n	c	s	r	f	o	x	z	k	h	c	x	m	l	r	i	r	l	i	b

Het woord 'het' wordt als 'r i r' gecodeerd en vervolgens als 'f o x' en 'r i r'. Met de lengte tussen deze herhalingen van 'r i r' kun je de lengte van het sleutelwoord achterhalen. In het volgende voorbeeld kraken we de sleutel volledig.

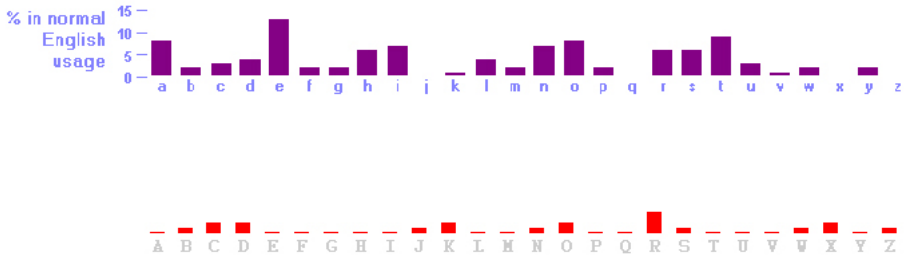
Sleutelwoord:	k e y k e y k e y k e y k e y k e y k e y k e y k e y k e y k e y k e y k e y k e y k e y k e y k e
Plaintekst:	h e t m e i s j e z e i t e g e n h e t k i n d d a t h e t h a a r p o p w a s e n h e t n i e t v a n h a a r w a s
Cijfertekst:	r i r w i g c n c j i g d i e o r f o x i s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

Eerst wordt er gezocht naar herhalende delen in de cijfertekst.



Figuur 3.11: Analyse van de cijfertekst

Het herhalende deeltje is 'R I R' met telkens de factor van de lengte tussen de deeltjes. Bij factor 3 is te zien dat beide 'R I R'-en overeenkomen qua lengte tussen de sequenties. De lengte van de sleutel is dus 3 en nu we dit weten, kunnen we verder gaan met het kraken van het sleutelwoord. We noemen de letters van het sleutelwoord voorlopig x, y en z. De letter x bepaalt een rij in het Vigenère vierkant en deze rij bepaalt dan een enkelvoudige substitutie (precies als het Caesar Cijfer) voor de eerste letter van de plaintekst, voor de 4, 7, 10, 13,... Door middel van de frequentietabel kunnen we de k voor deze ene rij die staat voor x, bepalen.



Figuur 3.12: Frequentietabel van analyse en taal

Men maakt een frequentieanalyse van de 1^e letter, 4^e, 7^e,... en vergelijkt dit met de frequentietabel van de desbetreffende taal (hier engels) en men verschuift de frequentieanalyse totdat ze de frequentietabel van de taal benadert. Het aantal keren dat de frequentietabel heeft

moeten opschuiven (k), staat gelijk met de x. Dit proces moet steeds herhaald worden totdat het gehele sleutelwoord achterhaald is en dan is deze code pas gekraakt.

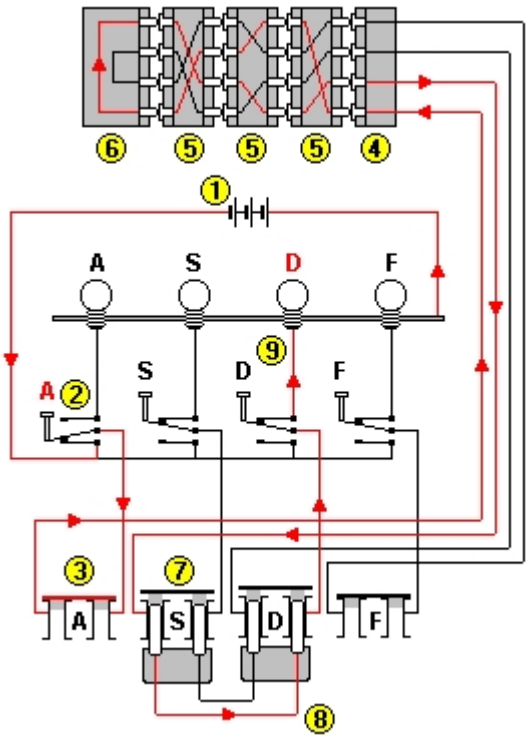
3.3.4 Enigma

Inleiding

In 1918 ontwikkelde de Duitser Scherbius de Enigma, één van de eerste draagbare codeermachines. De Enigma wordt bij de familie van de electromechanische rotor-machines gerekend en is zeer bekend geworden in de Tweede Wereldoorlog wanneer de Duitsers deze machine in grote oplage produceerde en in gebruik namen. Zelf noemde ze het de 'Wehrmacht Enigma'.

Werking

Het apparaat bestaat uit een toetsenbord om de plaintekst in te geven, een encryptie deel met rotors en een paneel met lichtjes om de cijfertekst af te lezen. De Enigma bestaat uit een elektrisch gedeelte en een mechanisch gedeelte. Het mechanische gedeelte, het encryptiedeel met de rotors (scramblers) bestaat uit 4 schijven die instaan voor het sleutelwoord waarmee gecodeerd wordt. In zo een schijf zitten 26 verbindingen die voor de connectie van de 26 letters zorgen dus een simpele substitutie versleuteling maar na elke letter die doorgevoerd was, versprong de eerste rotor één plaats. Wanneer de éérste rotor 26 keer versprongen was, versprong de tweede rotor één plaats, enzo verder.



Figuur 3.13: Voorbeeld van werking Enigma met enkel 4 letters.

Wat de Enigma nog speciaal maakt was de *reflector* (5^{de} rotor) die zorgde dat de encryptie gelijk was aan de decryptie waardoor na de codering van een bericht geen andere handelingen uitgevoerd moesten worden dan tijdens de decodering. De plaintekst en cijfertekst zijn gespiegeld tegenover elkaar.

Als we al de elementen die bijdragen tot een complexere code: het schakelbord zelf, ordening rotoren, beginpositie rotoren, beweegware ring en de reflector optellen dan komen we aan 3 283 883 513 796 974 198 700 882 069 882 752 878 379 955 261 095 623 685 444 055 315 226 006 433 615 627 409 666 933 182 371 154 802 769 920 000 000 000 mogelijkheden.

$$E_k(E_k(P)) = P \tag{3.6}$$

Kraakbaarheid

De Enigma code is zeer moeilijk te kraken omdat de cijfertekst van eenzelfde letter altijd anders is door de draaiende rotors. Wat wel een zwakte betekent in deze machine is dat een letter nooit gecodeerd diezelfde letter kan zijn. Een belangrijke fout in het ontwerp van de Enigma was dat een letter nooit in zichzelf vercijferd werd, wat het speurwerk van de codebrekers beperkte. Een andere belangrijke techniek was het zoeken naar 'cribs'. Dit was het zoeken naar de juiste positie van vercijferde tekst binnen een bericht, waarvan men de

klare tekst vermoedde. Door de strikte uniformiteit in de Duitse berichten kon men dikwijls voorspellen waar welk stukje tekst voor zou komen. Eens zo'n crib gelokaliseerd, zocht men met een Bombe (machine die alle mogelijkheden onderzocht) de sleutelinstellingen die erbij hoorden, om vervolgens het bericht te ontcijferen.

3.3.5 XOR

Inleiding

XOR is een techniek maar kan op zichzelf ook als cryptografisch systeem gebruikt worden. Een tekst wordt binair omgezet en ge-XOR'ed met een sleuteltekst (binair).

De volgende rekenregels worden gebruikt:

$$\begin{aligned} XOR(1,1) &= 0 \\ XOR(0,0) &= 0 \\ XOR(1,0) &= 1 \\ XOR(0,1) &= 1 \end{aligned} \tag{3.7}$$

Voorbeeld:

P	k	C
0	0	1
0	1	0
1	0	0
1	1	1

Toepassing

XOR heeft een zeer mooie toepassing bij databeheer. Harde schijven kunnen op verschillende manieren opgesteld worden en een manier hiervan is de RAID-3. Om dit systeem te kunnen installeren zijn er 3 schijven nodig. De 1^e schijf en de 2^e schijf worden gebruikt om de data op te schrijven en de 3^e als parity schijf (de repair schijf). Als er data worden weggeschreven, wordt er afwisselend op de ene schijf geschreven en dan op de andere. De XOR van deze data wordt dan weer geschreven op de parity disk.
Voorbeeld:

1	2	3 (parity)
0	0	1
0	1	0
1	0	0
1	1	1

Omdat het lezen van data op 2 schijven tegelijk sneller gaat als lezen op 1 enkele schijf, wordt deze techniek toegepast. Maar omdat de kans dat je data verliest 2 maal zo groot is omdat de data gespreid weggeschreven worden, wordt er een parity disk aangemaakt. Als disk 2 wegvalt is het zeer eenvoudig om samen met de parity disk de data van de disk 2 terug te halen met de hulp van de omgekeerde XOR.

3.3.6 DES

Inleiding

DES (Data Encryption Standard) is een vorm van blokkencryptie die beschouwd wordt als een moderne symmetrische versleuteling in tegenstelling tot de voorgaand besproken systemen die als klassiek worden gedefinieerd. Bij dit systeem wordt er niet letter per letter gewerkt maar met een hele blok tekst (in binaire vorm) van 64 bit. DES is ontwikkeld door IBM, en is in 1977 tot standaard verheven. Deze blokken van 64 bit worden één voor één door 19 verschillende stappen gehaald. Bij deze stappen wordt voornamelijk de XOR techniek gebruikt (zie 3.3.5).
DES werkt met sleutels van 64 bits, maar elke 8^{ste} bit wordt gebruikt ter controle. Dit betekent dus dat er $2^{64-8} = 2^{56} = 72057594037927936$ mogelijke sleutels zijn.

Een plaintekst kan met DES maar op één manier gecodeerd worden, bijgevolg zijn de cijferteksten van éénzelfde plaintekst dezelfde.

3 DES

3 DES of triple-DES is hetzelfde als een DES-encryptie maar ze wordt 3 keer afzonderlijk uitgevoerd met 2 verschillende sleutels. De stappen zien er dan zo uit:

1. Versleutel met k1
2. Ontcijfer met k2
3. Versleutel met k1

Bij het decoderen:

1. Ontcijfer met k1
2. Versleutel met k2
3. Ontcijfer met k1

Kraakbaarheid

Tot op de dag van vandaag zijn er nog geen zwakheden gevonden in het DES-systeem zelf. Maar dit systeem wordt toch als kraakbaar gezien omdat de sleutels te kort zijn (56 bits) en bijgevolg dus snel te kraken met een brute-force-attack.

3.3.7 Rijndael

Rijndael is de opvolger van DES. Joan Daemen (Leuvense) heeft *Rijndael* ontwikkeld en won er een wedstrijd mee om het oude DES systeem te verbeteren. Vervolgens kreeg het de officiële naam AES, Advanced Encryption Standard waaronder het beter bekend is. Zo werd zijn systeem als veiligst, prestatievolst, efficiëntst, eenvoudigst en flexibelst gequoteerd en noemde zijn Rijndael vanaf toen officieel AES.

De grootste verbetering waren de vergroting van blokgrootten en sleutels die een veelvoud konden zijn van 32-bit met een minimum van 128-bit en een maximum van 256-bit.

3.4 Asymmetrische key cryptografie

3.4.1 RSA

Inleiding

Dit algoritme heeft zijn naam te danken aan 3 personen die dit systeem in 1978 ontworpen hebben: *R*ivest, *S*hamir en *A*dleman. Bij een asymmetrische versleuteling zijn 2 sleutels nodig: een publieke- en een privésleutel. Het steunt op de nog niet ontdekte techniek om snel een getal in priemgetallen te ontbinden.

Werking

Allereerst om de 2 sleutels te creëren zijn er 2 priemgetallen nodig: p en q die ongeveer elk 150 cijfers lang zijn en $p \neq q$.
Dan wordt $n = p.q$ berekend, wat dus meer dan 300 cijfers zal bevatten.
Vervolgens is er nog een encryptiesleutel e nodig zodat $GGD(e, \varphi(n)) = 1$. Nu hebben we uiteindelijk de publieke sleutel die bestaat uit n en e.

Om te *coderen* wordt de plaintekst omgezet van tekst naar getallen door bijvoorbeeld ASCII-code of zoals we voorheen gedaan hebben volgens de plaats in het alfabet ($a = 1$, $b = 2$,...). Deze nieuwe omgezette plaintekst is vanaf nu M en de cijfertekst N. Met volgende bewerking wordt de plaintekst gecodeerd.

$$N \equiv M^e \pmod{n} \tag{3.8}$$

Om te *decoderen* hebben we de d nodig die vanaf het berekenen van de publieke sleutels ook berekend was. Via het algoritme van Euclides werd d zo berekend:

$$\begin{aligned} ed &\equiv 1 \pmod{\varphi(n)} \\ &\Downarrow \varphi(p.q) = (p-1)(q-1) \\ ed &\equiv 1 \pmod{(p-1)(q-1)} \end{aligned}$$

Hieruit volgt ook dat n en d relatief priem zijn.
Met deze d kunnen we de cijfertekst als volgt decoderen:

$$\begin{aligned} N^d \pmod{n} &= (M^e \pmod{n})^d \pmod{n} \\ &= (M^e)^d \pmod{n} \\ &= M^{ed} \pmod{n} \\ &\Downarrow \text{stelling van Euler} \\ &= M \pmod{n} \\ &= M \end{aligned}$$

Voorbeeld

Als voorbeeld wordt er een situatie gecreëerd waarbij A de zender is en B de ontvanger. Persoon B creëert een publieke- en een privésleutel met volgende parameters.

Hij genereert eerst 3 priemgetallen met een bepaalt algoritme (Fermat) en krijgt:
$p = 37$, $q = 41$, $e = 77$
Hij berekent zo $n = 1517$, $\varphi(n) = 1440$ en $d = 1253$.

Persoon B publiceert zijn publieke sleutel n en e op het i-net. Persoon A wil het bericht 'CRYPTO' versturen naar persoon B en leest de n en e van persoon B op het i-net.

Eerst zet persoon A zijn tekst om in cijfers volgens plaats in het alfabet.
Dan wordt 'CRYPTO':

C	R	Y	P	T	O
03	18	25	16	20	15

Nu past hij op elk blok: $N \equiv M^e \pmod{n}$

N wordt dan:	0243	0570	0400	0625	1017	1208

Persoon A stuurt de gecodeerde boodschap N door naar persoon B.

Persoon B past dan het volgende toe op elk blok van N:
$N^d \pmod{n} = M$

M wordt dan:	0003	0018	0025	0016	0020	0015
	C	R	Y	P	T	O

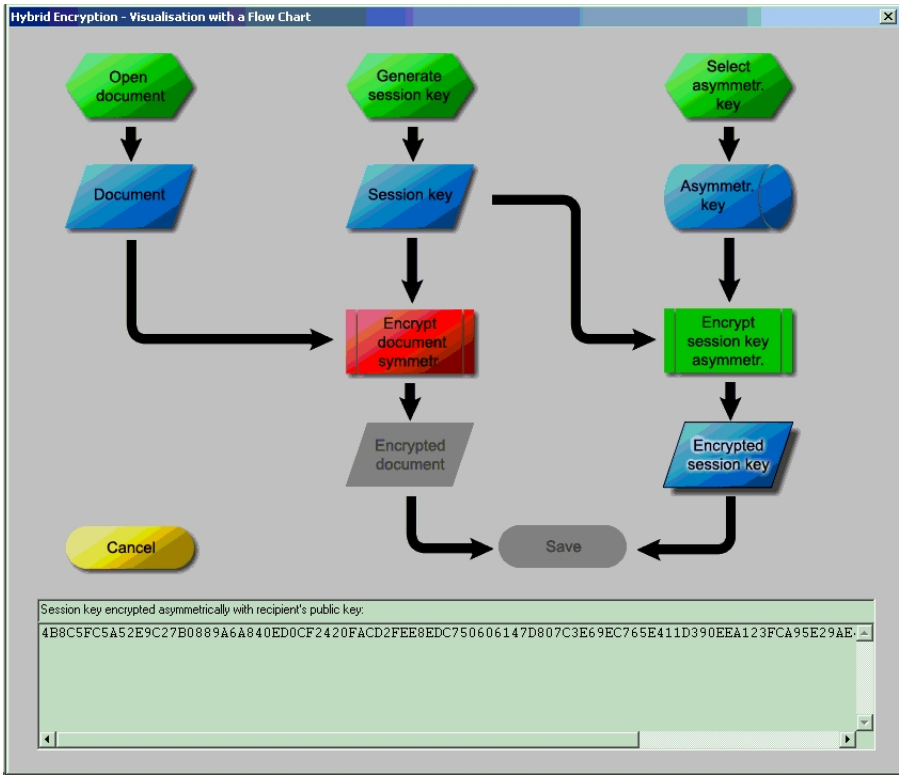
3.5 Hybride key cryptografie

Hybride key cryptografie is zoals de titel al doet vermoeden een kruising tussen de asymmetrische- en symmetrische versleuteling. Bij een juiste keuze van versleuteling kunnen de 2 voordelen van beide technieken gecombineerd worden en net daarom is deze manier één van de populairste.

De hybride key cryptografie werkt als volgt: het bericht wordt symmetrisch versleuteld, de sleutel hiervan wordt asymmetrisch versleuteld en doorgegeven aan de ontvanger. Hierbij wordt de snelheid van de symmetrische versleuteling gecombineerd met de moeilijk kraakbare asymmetrische versleuteling van de sleutel.

3.5.1 RSA-AES

RSA-AES is de combinatie van RSA en AES. De werking ervan wordt geïllustreerd aan de hand van onderstaande figuur.



Figuur 3.14: RSA-AES

Een session-key wordt willekeurig aangemaakt waarmee de plaintekst volgens het AES-principe symmetrisch versleuteld wordt. Vervolgens wordt deze session-key versleuteld volgens de asymmetrische RSA-methode. De versleutelde session-key wordt samen met het versleutelde

bericht doorgegeven aan de ontvanger die de sleutel van de versleutelde session-key heeft. Om het bericht te ontcijferen dient de ontvanger eerst de asymmetrische versleutelde session-key te decrypten om vervolgens de cijfertekst te decoderen tot de plaintekst.

Hoofdstuk 4

Toepassingen

4.1 Rekeningnummer

Bankrekeningnummers lijken helemaal willekeurig gekozen maar dit is maar schijn. Er is namelijk een logica in de keuze van de cijfers wat misbruik van rekeningnummers kan tegen gaan.
Een rekeningnummer bestaat uit 12 cijfers: 3 groepjes van cijfers gescheiden door een ' - ' waarvan het 1^e groepje een kenteken van de bank is en het laatste groepje de 2 controle cijfers bevatten. De controle cijfers worden gevormd door van de eerste 10 cijfers de modulus 97 te nemen. Het getal 97 is tevens ook het grootste priemgetal van 2 cijfers.

Een voorbeeld:

> Het rekeningnummer is 235-0351345-23.
> Er zou dus moeten gelden dat $2350351345 = 23 \bmod 97$.

4.2 Smartcards

4.2.1 Inleiding

Figuur 4.1: Smartcard

Tegenwoordig zijn de smartcards niet meer weg te denken en zitten ze zelfs op onze identiteitskaart. Op deze kaarten zit een kleine chip die bestaat uit een soort ROM-geheugen, een CPU en contactpunten. Dit geheugen is verdeeld in 2 zones: de publieke zone en een privé zone die alleen leesbaar is door de CPU van de chip zelf. In de publieke zone zit een identificatie nummer i, bij een bankkaart is dit het rekeningnummer. In de privé zone zit een geheime sleutel k. Zo is het bij de meeste smartcards dat DES als versleuteling wordt gebruikt.

De smartcard-lezer zelf bezit K dat een super geheime sleutel is. Eenmaal deze sleutel geweten is, is het systeem makkelijk te kraken.

Dit gehele gedoe met sleutels werd gebruikt om de identiteit van de gebruiker in het oog te houden. Anders zouden er vervalsers een andere i hebben kunnen programmeren zodat ze geld van iemand anders zijn rekening kunnen afhalen of een vervalser die met een smartcard-lezer geld van rekeningen kan afhalen.

4.2.2 Identiteit controleren

Op het geheugen van de chip staat de k, die als volgt op de chip werd gezet:

$$k = DES(K, i) \tag{4.1}$$

De smartcard-lezer zelf kan ook de k berekenen omdat hij de i en de K kent. Maar de smartcard zelf zal nooit de k blootgeven.

Om nu de identiteit vast te stellen stuurt de smartcard-lezer 64 willekeurige bits (r) naar de kaart die het volgende berekent:

$$c = DES(k, r) \tag{4.2}$$

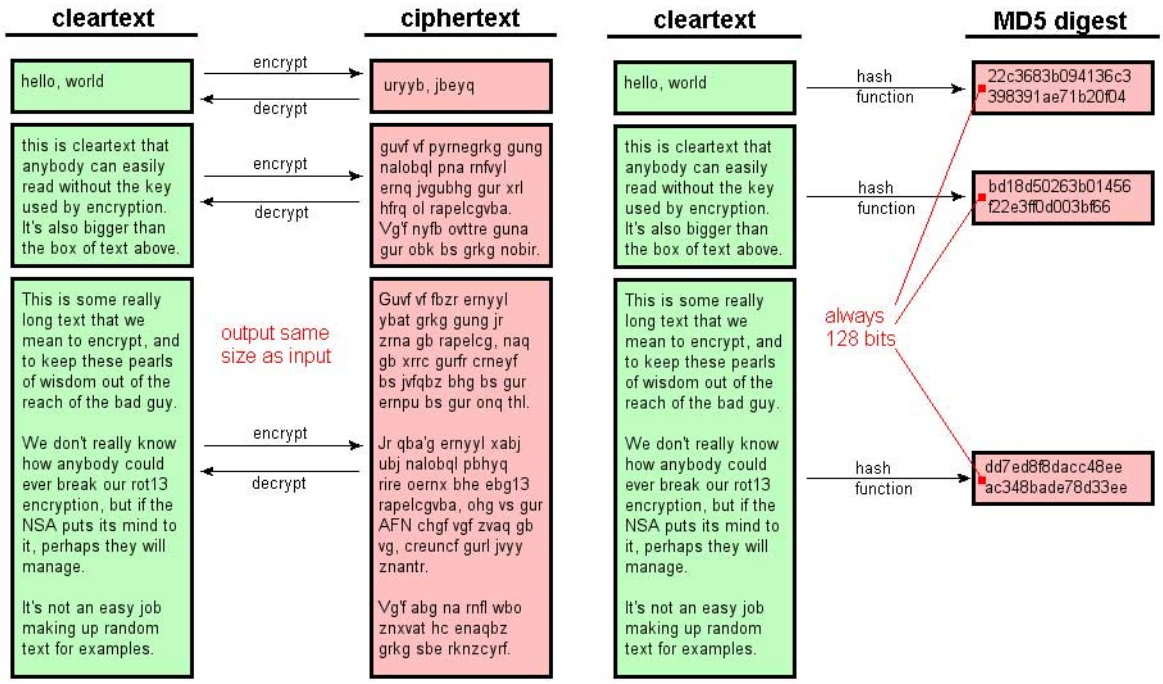
Omdat de smartcard-lezer zelf de k ook kan berekenen, kan hij de uitkomst voorspellen. Als de voorspelde uitkomst niet overeen komt met de berekende sleutel dan is de identiteit vervalst. De berekende waarde c staat voor de challenge omdat ze telkens weer verandert.

Net andersom kan de smartcard ook de identiteit van de smartcard-lezer controleren door 64 willekeurige bits (r) naar de smartcard-lezer te sturen.

4.3 Hashes

4.3.1 Inleiding

'Hashes' hebben weinig met encryptie te maken maar ze gaan wel vaak samen. Dit omdat ze vaak gebruikt worden in beveiligingssystemen.
De hash-functie of ook wel 'digest' genoemd, berekent van een plaintekst een unieke code (de hash). Het is te vergelijken met een veiligheidszegel op een medicijn: als het potje open wordt gedaan, is het zegel verbroken. Dus als je de plaintekst maar miniem zou aanpassen zou de hash zelf al geheel anders zijn.
Anders als bij encryptie wordt de plaintekst niet vertaald naar hetzelfde aantal karakters maar steeds naar een bepaald aantal bits afhankelijk van de gebruikte hash-functie. Op onderstaande figuur is dit duidelijk te zien en in dit voorbeeld werd er gebruik gemaakt van de md5 hash-functie.



Figuur 4.2: Verschil: hash-encryptie

Een hash-functie is uitsluitend een 'one-way' operatie waardoor uit een hash zelf de plaintekst niet gereconstrueerd kan worden dit maakt dat de hash-functie niet bij cryptografie gerekend kan worden. Bij het volgende voorbeeld wordt duidelijk dat bij een kleine verandering het lawine-effect in werking treedt en de gehele hash anders is als de originele. De hoofdletter 'T' wordt veranderd in 't'.

file1:
This is a very small file with a few characters

file2:
this is a very small file with a few characters

md5-hash van de teksten:
75cdbfeb70a06d42210938da88c42991 file1
6fbe37f1eea0f802bd792ea885cd03e2 file2

4.3.2 Gebruik

Hashes worden om verschillende nuttige redenen gebruikt:

Bestand authenticiteit

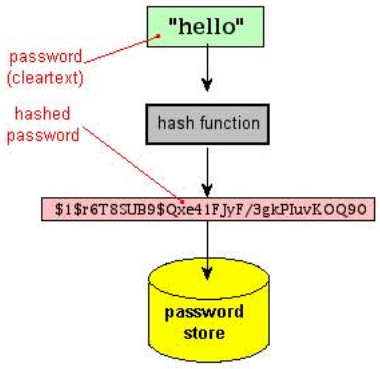
De meest voor de hand liggende redenen is om te verifiëren of een gedownload programma wel degelijk echt van de 'makers' komt en niet eerst aangepast is geweest door een cracker die er een virus in heeft verwerkt. Op de meeste sites van freeware software wordt steeds de md5-hash van het programma uitgestald zodat je nadien makkelijk je eigen programma kan vergelijken met de authentieke md5-hash.
Als voorbeeld nemen we het programma 'Cryptool' waarvan in dit eindwerk gebruik werd van gemaakt:

> Op de officiële site staat de volgende info over de Engelse versie 1.4:
> md5-hash: b997e263e685a5ef5f90cf0fbcefe1ba *CrypTool.exe
>
> De md-5 hash wordt berekend van het gedownloade Cryptool.exe
> md5-hash: b997e263e685a5ef5f90cf0fbcefe1ba
>
> Net dezelfde, dus is het bestand authentiek.

Paswoorden hashen

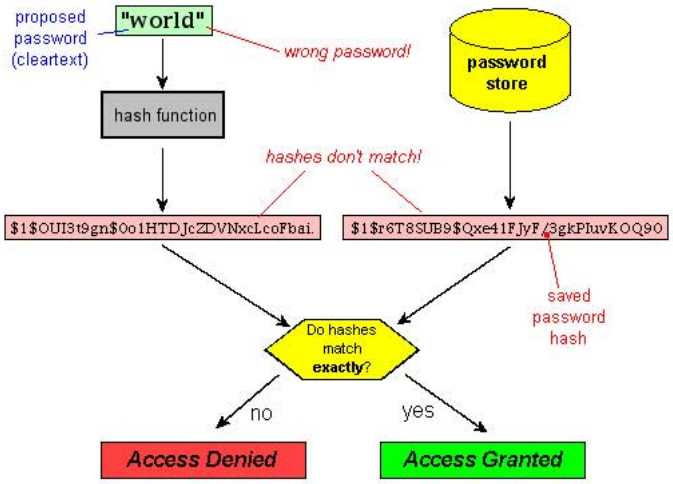
Deze toepassing wordt bijna altijd gebruikt en zonder dat de meesten het zelfs weten. Aan de hand van een voorbeeld is het makkelijk te begrijpen.

Stel je voor dat je jezelf registreert op een site waar er een sectie met login is. Je zal een formulier krijgen waar je verplicht je gegevens zal moeten invullen ook zal er gevraagd worden naar een paswoord. Van zodra je het gehele formulier hebt ingevuld, zullen afhankelijk van de programmastructuur al je gegevens in een database worden gestoken. Maar je paswoord zal eerst met een hash functie worden bewerkt vooraleer het in de database gestopt wordt. Zo kunnen hackers en systeembeheerders die toevallig of met opzet in de database geraken, de paswoorden niet gebruiken.



Figuur 4.3: Paswoord hashen: opslagen paswoord

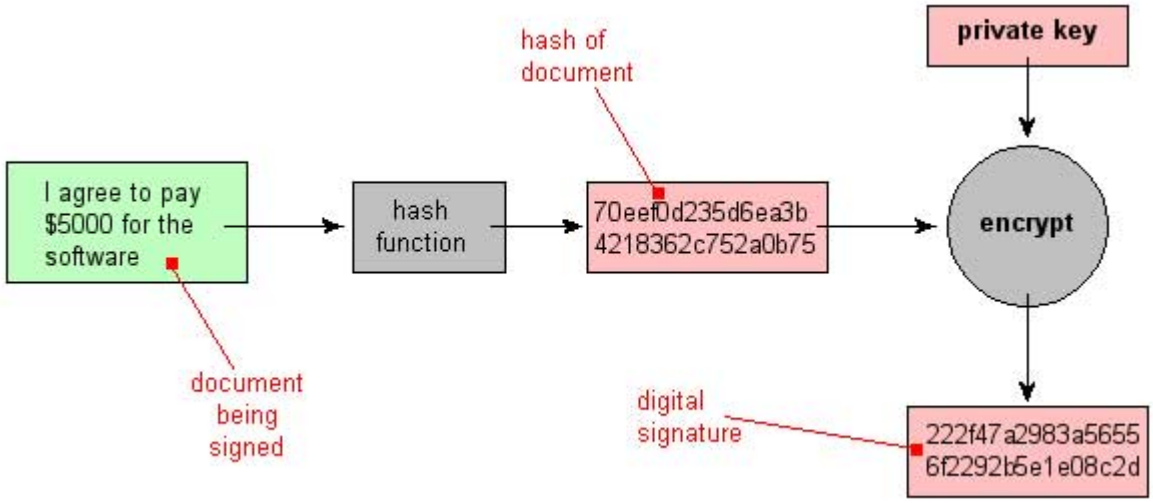
Wanneer je dan wil inloggen, wordt er van je net ingegeven paswoord een hash berekend en vergeleken met de hash van in de database. Als de 2 dezelfde zijn, wil dit zeggen dat ook de plainteksten dezelfde waren en log je in. Als de 2 niet dezelfde zijn, betekent dit dat je het foute paswoord hebt gebruikt.



Figuur 4.4: Paswoord hashen: inloggen

Digitale handtekeningen

Bij deze methode komt cryptografie eindelijk te pas. De procedure om aan een tekst een handtekening mee te geven, is mooi geïllustreerd op de volgende figuur.



Figuur 4.5: Digitale handtekening

Eerst wordt er van de tekst de hash berekend. Vervolgens wordt deze hash gecodeerd volgens

het asymmetrische RSA principe, maar de hash wordt deze keer versleuteld met de privé-sleutel van de zender. Als de ontvanger nu wil verifiëren of het bericht wel degelijk ongewijzigd verzonden is door de zender, moet hij de publieke-sleutel gebruiken om de handtekening te ontcijferen en dit vergelijken met de eigen gemaakte hash van de tekst.

4.3.3 Collisions

Collisions zijn 2 plainteksten die dezelfde hash krijgen. De ene hash-functie is er gevoeliger voor als de andere. Het brengt een groot gevaar met zich mee als de plaintekst gemanipuleerd kan worden zodat de hash hetzelfde blijft. Hieronder is een voorbeeld opgesteld waar duidelijk te zien is dat collisions een groot probleem kan opleveren qua veiligheid.

Het volgende bericht wordt verstuurd en wordt onderschept op weg naar de bank.

> Beste bank,
>
> Ik wil 10 euro van mijn rekening halen.
>
> Groeten
> Vleminckx Frederic

De tekst wordt zodanig aangepast dat hij lijkt op de volgende:

> Beste bank,
>
> Ik wil 10000 euro van mijn rekening halen en verzenden naar godelindestraat.
>
> Groeten
> Vleminckx Frederic

tekst	md2 hash
Originele:	E3 05 71 AE FC 55 87 D1 FC AC 38 F3 8C 85 7C BE
Gemanipuleerde:	E4 A1 D5 0A B5 A3 AB 6F B3 6B C8 13 DA D6 78 0D

Beiden hebben een andere hash met md2. Met behulp van een snel zoekend collision algoritme berekende ik ,met de instellingen dat alleen de eerste 32 bits hetzelfde moesten zijn omdat het algoritme dan anders dagen nodig had om een collision te vinden, de collision van de 2 teksten.

De originele tekst:

Beste bank,

Ik wil 10 euro van mijn rekening halen.

Groeten
Vleminckx Frederic
AAABCDBCDADADADCDB

En de gemanipuleerde tekst:

Beste bank,

Ik wil 10000 euro van mijn rekening halen en verzenden naar godelindestraat.

Groeten
Vleminckx Frederic
AACCBCDACBBDCBCACD

tekst	md2 hash
Nieuwe Originele:	0F 4B 48 1A 79 9E BC AA DF 9B 34 A5 ED 24 D6 EB
Nieuwe Gemanipuleerde:	0F 4B 48 1A 3B AA 71 4F FB 08 7C 58 79 9C A4 F1

De 2 aangepaste berichten geven dus duidelijk dezelfde hash-waarde (eerste 32 bits). Nu is het duidelijk dat dit bericht is aangepast maar in plaats van met letters de collision te doen kloppen kan er ook gebruik gemaakt worden van onzichtbare tekens zoals witregels, spaties,....

4.3.4 Hash-functies

Natuurlijk zijn er verschillende soorten hash-functies en die worden allemaal onderverdeeld in families.

MD-familie

De MD-familie, *M*essage-*D*igest bestaat momenteel uit md2, md4 en md5 waarvan md5 de bekendste en meest gebruikte is. De md van de naam staat voor de familie waarvan ze afstammen nl. Message-Digest en het cijfer voor het algoritme zo is md5 dus Message-Digest algorithm 5. De ontwikkelaar is Ronald Rivest die in 1989 begon met md2, is ook de grote man achter de RSA-cryptografie.

De md2 tot en met de md5 zijn niet meer zo veilig als gedacht werd. In deze hashes zou redelijk snel een collision kunnen berekend worden.

SHA-familie

De SHA-familie, *S*ecure *H*ash *A*lgorithm bestaat momenteel uit de SHA-0, SHA-1 en SHA-2. SHA-2 is de verzamelnaam van SHA-224, SHA-256, SHA-384 en SHA-512. Dit algoritme werd geschreven door de NSA (National Security Agency van Amerika).

Enkel bij SHA-2 is er nog geen collision gevonden voor de overige 2: SHA-0 en SHA-1 wel.

LM-familie

De LM hash oftewel de LAN Manager hash, bestaat uit 2 hash-functies. De eerste is LM-hash en de andere is NTLM. Ze zijn beiden door Microsoft ontworpen en zijn te vinden in Windows XP waar het gebruikt wordt om de login-paswoorden in op te slaan. Een zeer onverstandige keuze omdat LM een zeer zwakke hash-functie is, NTLM daarentegen is redelijk veilig.

De LM-hash is zeer onveilig omdat het ten eerste: als de plaintekst langer is als 7 karakters, de plaintekst opgesplitst worden in 2 delen waarvan afzonderlijk een hash berekend wordt. Ten tweede omdat al de karakters automatisch omgezet worden naar hoofdletters dus in plaats van 26 hoofdletters én 26 kleine letters én 10 cijfers, wat 52 combinaties geeft, worden er enkel 26 hoofdletters én 10 cijfers, wat maar 36 cominaties geeft, gebruikt. Dit maakt de LM-hash extreem gevoelig aan brute-force-attacks.

Hoofdstuk 5

Kraken

In dit hoofdstuk gaat het over de 5 basis manieren om in het geval van een cryptografische-functie een versleuteld bericht te ontcijferen zonder de sleutel te kennen, en in het geval van een hash-functie de originele plaintekst te weten te komen vanuit de hash zelf.

5.1 Zwakte in algoritme

Als een systeem gebruik maakt van een zwakke crypto- of hash-functie dan is het mogelijk om de plaintekst terug te vinden. De decodering met behulp van de gevonden zwakheid kan ook heel lang duren maar als de decryptie tijd onder de tijd die nodig is om de plaintekst te vinden, ligt; dan wordt het aanzien voor een succesvolle uitbuiting van de zwakheid in het algoritme.

Zo een zwak algoritme is 'LM' dat in 4.3.4 op de pagina hiervoor wordt uitgelegd. Het is net zo zwak omdat het enkel maar hoofdletters gebruikt en toegepast wordt enkel op 7 karakters, als de plaintekst langer is, wordt hij opgesplitst en apart gehashed.

5.2 Gokken

Omdat mensen meestal de zwakke factor zijn in cryptografie steunt deze methode daar net op. Veel mensen gebruiken altijd dezelfde combinaties en de meest gebruikte vermeld ik hieronder.

- Gewoon 'niets' als paswoord: blanc
- De standaard ingestelde paswoorden: 'paswoord', 'admin', 'demo', 'test' en al de vormen daarvan
- De acountnaam zelf of de naam van de persoon

- De naam van hun geliefd persoon, man of vrouw of zelfs hun moeders naam.
- De geboorteplaats, geboortedatum.
- Naam van huisdier
- Een cominaties van cijfers die elkaar opvolgen zoals '1234' en al de vormen daar rond.
- Net zoals het vorige maar dan met letters zoals bv een hele rij op een toetsenbord bv: 'azerty'
- Een simpele verandering aan bovenstaande voorbeelden door middel van de orde van de letters te verwisselen of nummers erin de verwerken bv: 'oli4'

Het is eenvoudig om een computerprogramma te maken dat al deze dingen overloopt en variaties berekent en test.

5.3 Dictionary-attack

Dictionary-attacks zijn in feite een uitwerking van het vorige. Een dictionary-attack is een programma dat als een woordenboek elke woord versleutelt of hasht en vergelijkt met de oorspronkelijke cijfertekst of hash. In dat woordenboek staan al de vorig vermelde puntjes: zo staan er dus allemaal namen en combinaties van letters in. Ook berekent het programma mogelijke variaties.

Tijdens de 2^e wereldoorlog werd deze techniek gebruikt om de Enigma-berichten te kraken. 'Eins' werd voor meer als 90% in de Duitse plainteksten gezet en vormde daarom ook de basis om de code te kraken.

5.4 Brute-force-attack

Als de vorige 3 methodes niet werken dan is er nog altijd de hoop op brute-force-attacks die altijd wel het gecodeerde bericht kraken omdat ze elke mogelijke combinatie uitproberen. In theorie is het wel altijd een succes maar in de praktijk kan deze bezigheid zo lang duren dat het praktisch niet meer te doen is. Bij bijvoorbeeld een paswoord van max. 7 karakters zijn er per karakter 69 mogelijkheden dus komen we op $69^7 + 69^6 + 69^5 + 69^4 + 69^3 + 69^2 + 69^1 = 7555858447479$. Omdat niet duidelijk is hoeveel karakters er gebruikt zijn moeten we ze allemaal testen tot en met 7. Op een gewone desktop pc kunnen er ongeveer 500 000 paswoorden per seconde gegeneerd worden. Dit zou ongeveer 7 555 858 447 479 uur oftewel ongeveer 5 maand en 21 dagen duren eer de volledige cyclus doorlopen is. De berekeningstijd kan men naar beneden halen door 'supercomputers' te gebruiken zoals de NASA er een voor gebruikt.

Meestal wordt deze techniek gecombineerd met de dictionary-attack om sneller tot de plaintekst te komen.

5.5 Precomputation

In tegenstelling tot de brute-force-attack wat meestal een zeer tijdrovende klus is, werkt de precomputing manier net anders om. Bij precomputing worden op voorhand lijsten gecreëerd waarin random combinaties al gecodeerd of gehasht worden. Vervolgens wordt de cijfertekst of hash opgezocht in deze lijsten en de plaintekst die erbij staat, is dan de uiteindelijk plaintekst. Net als bij brute-force-attack is het genereren van deze lijsten een lange klus en neemt het zeer veel opslagcapaciteit in. Voor het vorige voorbeeld is er ongeveer 119 GB nodig om deze lijsten op te slaan. Het voordeel is wel dat in een paar seconden of minuten de plaintekst al kan worden teruggevonden.

Omdat via precomputing, hashes sneller en sneller worden gekraakt, is het beter om een extra stuk willekeurige tekst bij de plaintekst toe te voegen en het dan pas te hashen. Dit wordt *salting* genoemd en het willekeurig stuk tekst dat er bij wordt gevoegd, wordt *salt* genoemd. Omdat de mogelijke combinaties dan zo groot zijn, kunnen ze niet in de precomputing-lijsten staan omdat de lijsten dan astronomisch groot zouden zijn.

Lijst van figuren

Bibliografie

A. E. Brouwer (2007). Math: Algebra. URL http://www.win.tue.nl/~aeb/.

P. J. Denef (2003). Wiskundige logica en getallenleer.

G. D. Samblanx (2004). Geheime berichten. URL http://geheimeberichten.denayer.wenk.be.

S. Singh (2003). Black chamber. URL http://www.simonsingh.net/The_Black_Chamber/home.html.

M. van Gompel Marc van Hintum (2007). Cryptologie. URL http://anaproy.homeip.net/proycon/standalone/cryptologie/.

Wikipedia (2006). Wikipedia - euler. URL http://nl.wikipedia.org/wiki/Stelling_van_Euler.

Wish-E (2002). Cryptografie. URL http://proto.thinkquest.nl/~klb024/wiskunde01.htm.