

**Profielwerkstuk Wiskunde
2005**

Cryptografie

Het RSA Cryptosysteem



**Sander Wildeman
6VWO profiel NT**

Begeleider: Cor Steffens

Inhoudsopgave

Voorwoord	2
Introductie	3
1. Geschiedenis.....	4
1.1 De Caesar code.....	4
1.2 De Vigenère code	4
1.3 Het Vernam systeem	5
1.4 Enigma, Sigaba en Red and Purple	5
1.5 (Advanced) Data Encryption Standard	6
1.6 Het RSA Cryptosysteem	6
2. Asymmetrische of publieke-sleutel cryptografie	7
2.1 Symmetrische cryptografie	7
2.2 Asymmetrische cryptografie	7
3. Wiskundig principes.....	8
3.1 Modulo-rekenen	8
3.2 Euclides	12
3.3 Fermat.....	15
3.4 Euler	17
4. Het RSA-systeem	19
4.1 Coderen (versleutelen)	19
4.2 Decoderen (ontcijferen).....	20
4.3 Versleutelen in de praktijk	21
4.4 Toepassingen.....	22
5. Number Field Sieve.....	23
5.1 Waarom is RSA veilig?.....	23
5.2 Kraken, het principe van de Number Field Sieve (NFS)	24
5.3 Uitdaging.....	25
Conclusies	26
Appendix	27
Computerprogramma RSAcrypt	27
Literatuurlijst.....	28

Voorwoord

Maart 2004 heb ik een Masterclass cryptografie, oftewel geheimschrift, gevolgd aan de Vrije Universiteit in Amsterdam (VU). Door deze cursus ben ik geïnteresseerd geraakt in het onderwerp.

Cryptografie komt, door de steeds groter wordende informatienetwerken, meer en meer in de belangstelling. Want hoe meer informatie er wordt verstuurd, hoe groter de kans dat buitenstaanders die kunnen onderscheppen. Het is dus van groot belang dat deze informatie alleen te lezen is voor de bedoelde ontvanger. Voor banken en militaire instellingen is het vrij voor de hand liggend dat zij hun gegevens niet onbeschermd, dus versleuteld, versturen. Maar ook in de ‘huis tuin en keuken communicatie’, vooral bij e-mail en internet, wordt er tegenwoordig steeds meer waarde gehecht aan privacy.

Daarom heb ik besloten om in dit profielwerkstuk te gaan onderzoeken hoe één van de meest gebruikte versleutelingmethoden, RSA genaamd, precies in zijn werk gaat en hoe veilig deze methode dan wel niet is.

Hierbij heb ik de volgende hoofd- en deelvragen opgesteld:

Hoofdvraag:

Is het RSA cryptosysteem veilig?

Deelvragen:

- *Hoe heeft de cryptografie zich door de jaren heen ontwikkeld?*
- *Wat houdt openbare-sleutel of asymmetrische cryptografie precies in?*
- *Wat is het RSA cryptosysteem?*
- *Hoe is RSA te kraken met de Number Field Sieve methode?*

In het eerste hoofdstuk zal ik een beschrijving geven van hoe de cryptografie zich door de jaren heen heeft ontwikkeld. In hoofdstuk 2 zal ik uitleggen wat het verschil is tussen de twee hoofdstromen in de cryptografie: asymmetrische en symmetrische. In het derde hoofdstuk ga ik verder in op de wiskundige principes die ten grondslag liggen aan de moderne cryptografie en in hoofdstuk 4 zal ik het RSA cryptosysteem zelf bespreken. Tot slot ga ik in het laatste hoofdstuk een methode bespreken om RSA te kraken die, op het moment van schrijven, de beste resultaten heeft behaald, de zogenaamde Number Field Sieve methode.

Introductie

Het internet is een grote bron van informatie, maar niet alle informatie is voor iedereen bestemd. Mensen die bijvoorbeeld bankzaken doen via dit wereldwijde communicatie netwerk of vertrouwelijke post versturen, willen niet dat jan en alleman deze gegevens kunnen lezen.

Dit is waar de Cryptografie om de hoek komt kijken. Het woord cryptografie stamt af van de Griekse woorden *kryptos* (= verborgen) en *grafein* (= schrijven), en is dus te vertalen als 'verborgen schrijven', of geheimschrift. De tegenhanger van cryptografie is de cryptoanalyse: het analyseren van geheimschrift, oftewel het kraken van gecodeerde berichten.

Behalve voor coderen van boodschappen ter *geheimhouding*, wordt de cryptografie ook gebruikt om de *integriteit* en de *authenticiteit* van boodschappen te waarborgen. Ik zal deze begrippen nader toelichten:

- **Geheimhouding:** Dit is vrij duidelijk. Een onbevoegde mag niet in staat zijn een boodschap te lezen. Bijvoorbeeld bij uitbesteding van openbare werken gaat de opdracht naar de firma die de meest gunstige prijsofferte maakt. Het zou erg winstgevend zijn voor een bedrijf om de offerte van een concurrent te kunnen lezen, en vervolgens een paar euro lager te bieden.
- **Integriteit:** Het gaat hierbij om de vraag of er tussen verzending en ontvangst met een boodschap geknoeid is. Bijvoorbeeld, na ondertekening van een contract wijzigt de tegenpartij enkele details in haar voordeel.
- **Authenticiteit:** Hierbij gaat het om de vraag of de boodschap afkomstig is van wie ze afkomstig lijkt te zijn. Bijvoorbeeld, de tegenpartij in een contractuele verbintenis ontkent het contract te zijn aangegaan. Daarom worden verbintenissen ook vaak aangegaan in het bijzijn van een notaris.

In een tijd waarin contracten en transacties meestal op papier werden afgehandeld, werd gebruik gemaakt van praktische maatregelen, zoals een persoonlijke handtekening, verzegeling of de tussenkomst van een notaris, om bovenstaande punten te ondervangen. Maar nu steeds meer zaken via de elektronische snelweg gaan, moet men zoeken naar nieuwe beveiligingstechnieken. Per jaar vinden er immers alleen al in Nederland ongeveer 1,5 miljard girotransacties en 500 miljoen geldopnames uit de muur plaats, om maar niet te spreken van alle elektronische post die verstuurd wordt. Een elektronisch equivalent voor de notaris, de verzegelde envelop en de handtekening is dus een must!

Iedereen met wat fantasie kan wel manieren bedenken om een tekst zo te versleutelen zodat alleen de geadresseerde deze weer kan ontcijferen. Maar, bijvoorbeeld, de overheid of een bedrijf gaat niet in zee met amateuristische methoden. In deze instanties worden versleutelingmethoden vaak intensief en op grote schaal gebruikt. Waarbij er meestal veel op het spel staat. De gebruiker van cryptografie verlangt harde, liefst wetenschappelijke garanties op geheimhouding, integriteit en authenticiteit. De moderne cryptografie voldoet hieraan. De theorie berust op wiskundige technieken uit veel verschillende disciplines: onder andere kansrekening en statistiek, algebra en getallentheorie.

1. Geschiedenis

Cryptografie gaat heel ver terug in de tijd, al sinds mensen kunnen schrijven, is er behoefte aan manieren om geschreven berichten geheim te kunnen houden voor de nieuwsgierige ogen van derden.

In dit hoofdstuk zal ik een zestal beroemde geheimschriftsystemen van vroeger tot nu beschrijven. Veel principes van oude systemen zijn nu nog terug te vinden in de moderne cryptografie.

1.1 De Caesar code

De oudste bekende cryptografische methode stamt uit Romeinse tijd. Julius Caesar verstuurde, rond 50 voor Christus, zijn berichten aan Marcus Tullius Cicero in gecodeerde vorm volgens het volgende schema:

A	B	C	D	E	F	...	U	V	W	X	Y	Z
↓	↓	↓	↓	↓	↓		↓	↓	↓	↓	↓	↓
D	E	F	G	H	I		X	Y	Z	A	B	C

Elke letter in het alfabet wordt drie eenheden naar rechts opgeschoven (waarbij het einde in het begin overloopt). De tekst *dit is een groot geheim* wordt dus gecodeerd naar *glw lv hhq jurrw jhkhlp*.

In dit geval zegt men dat ‘drie’ de sleutel is van de code. Als de ontvanger op de hoogte is van deze sleutel, kan hij het bericht eenvoudig weer leesbaar maken, door alle letters drie eenheden naar links op te schuiven in het alfabet. Keizer Augustus, de opvolger van Caesar, koos ‘vier’ als sleutel. De algemene methode staat nu bekend als de Caesar code. Deze code is niet echt veilig, want iemand die een tekst, gecodeerd met de Caesar code, ontvangt, hoeft hoogstens 25 mogelijkheden na te gaan.

1.2 De Vigenère code

Pas in de 16^e eeuw werd er door Blaise de Vigenère een beter alternatief bedacht voor de Caesar code. Het systeem maakt wel gebruik van het Caesar systeem, maar in plaats van één getal, wordt er in het Vigenère een willekeurige reeks van getallen als vercijfersleutel gebruikt. Bijvoorbeeld de reeks 10, 2, 4, 13 wordt gebruikt om de tekst *niemand kan dit lezen* te vercijferen:

N	I	E	M	A	N	D		K	A	N		D	I	T		L	E	Z	E	N
10	2	4	13	10	2	4		13	10	2		4	13	10		2	4	13	10	2
X	K	I	Z	K	P	H		X	K	P		H	V	D		N	I	M	O	P

De eerste letter wordt volgens de Caesar methode 10 plaatsen in het alfabet opgeschoven, de tweede letter 2 plaatsen, de derde letter 4 plaatsen, de vierde letter 13 plaatsen, de vijfde letter weer 10 plaatsen, ...enzovoort. Het kraken (decoderen of ontcijferen) van de Vigenère code is een stuk lastiger, want er zijn legio mogelijkheden voor de te gebruiken sleutel.

1.5 (Advanced) Data Encryption Standard

In 1974 werd de Data Encryption Standard (DES) ontwikkeld door IBM voor gebruik door de overheid en privé-sector. Het systeem werkt volgens een zogenaamde blokcodering, waarbij 'blokken' van 64 bits tegelijk worden gecodeerd door een computerchip. In feite is het een permutatiecode, toegepast op een 'alfabet' met 2^{64} symbolen. In plaats van letters worden er bits verwisseld en vervangen. Hierbij wordt gebruik gemaakt van een sleutel van 56 bits.

DES is in gebruik sinds 1977. De meningen over dit systeem zijn nogal verdeeld. De experts vinden de sleutel van 56 bits te klein. Volkomen terecht! In juni 1997 is het een groep in de USA gelukt een boodschap te kraken die met DES versleuteld was. Daarvoor hadden ze wel tienduizenden computers nodig die via Internet samenwerkten. Men probeerde eenvoudigweg alle sleutels; dat zijn er dus 2^{56} , ongeveer 72 biljard! Omdat computers tegenwoordig zo snel zijn deed men er 'slechts' 96 dagen over voor men de juiste sleutel gevonden had. Inmiddels (sinds 2001) is Advanced Encryption Standard (AES) de nieuwe standaard van de Amerikaanse overheid. Deze standaard werkt met langere sleutels, van 128 of zelfs 256 bits, en is dus een stuk moeilijker te kraken.

DES wordt bijvoorbeeld gebruikt in pinpasjes. DES mag dan 'minder' veilig zijn, het is wel razendsnel, vooral in hardware toepassingen (chips).

In 1977 werd een revolutionaire nieuwigheid geïntroduceerd door Diffie en Hellman: een *public distribution system*. Het bood de eerste oplossing voor het probleem hoe twee partijen een gemeenschappelijke sleutel kunnen afspreken zonder elkaar te ontmoeten en zonder de sleutel over het (onveilige) publieke elektronische kanaal te verzenden.

1.6 Het RSA Cryptosysteem

Kort na het public distribution system van Diffie en Hellman ontstond in 1978 het RSA Cryptosysteem. Het systeem is bedacht door en vernoemd naar de wiskundigen Rivest, Shamir en Adleman. RSA is tegenwoordig één van de meest gebruikte coderingssystemen. Naast beveiliging biedt RSA ook uitstekende mogelijkheden om de authenticiteit van berichten te waarborgen.

Ik zal hier niet verder op de werking van RSA ingaan. In hoofdstuk 4 zal ik uitleggen hoe het coderen en decoderen met RSA precies in zijn werk gaat. Verder zal ik in hoofdstuk 5 uitleggen hoe RSA te kraken is, of beter gezegd, waarom het zo moeilijk is om RSA te kraken.

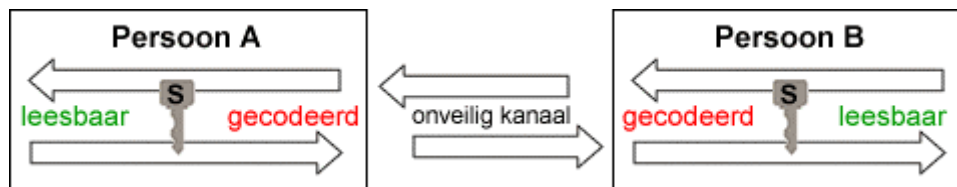
2. Asymmetrische of publieke-sleutel cryptografie

In de cryptografie zijn twee hoofdstromen te onderscheiden. De in het verleden veelvuldig toegepaste symmetrische, en de relatief nieuwe asymmetrische cryptografie.

In dit hoofdstuk zal ik kort uitleggen wat beide methodes precies inhouden en wat de verschillen zijn.

2.1 Symmetrische cryptografie

Het eerder genoemde Caesar systeem is een goed voorbeeld van symmetrische cryptografie. Het kenmerk van symmetrische cryptografie is dat zowel de zender als de ontvanger over *dezelfde* geheime sleutel moet bezitten, willen zij succesvol berichten kunnen uitwisselen.

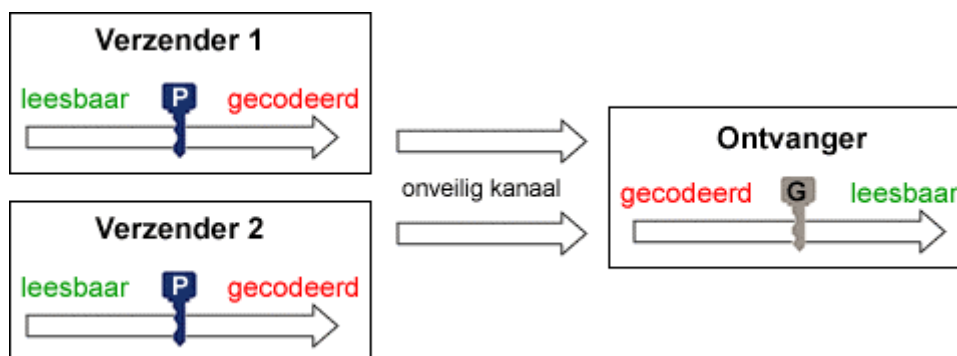


Het grote nadeel van deze methode is, dat de sleutel van te voren overlegd moet worden. Waarbij er dus een kans bestaat dat derden deze onderscheppen. Ook is het mogelijk dat één van de partijen de sleutel niet goed geheim houdt.

2.2 Asymmetrische cryptografie

Publieke-sleutel, of asymmetrische cryptografie is de tegenhanger van de hierboven beschreven symmetrische cryptografie. In 1978 werd een systeem bedacht, RSA, waarbij de sleutel waarmee het bericht kan worden ontcijferd *verschilt* van de sleutel waarmee het bericht is versleuteld. De verzender krijgt een zogenaamde *publieke* sleutel van de ontvanger waarmee hij zijn bericht kan coderen. Met dezelfde sleutel kan hij het bericht echter niet meer decoderen. Dat kan alleen de ontvanger, die over een *geheime* sleutel beschikt.

Voordelen van deze methode zijn: er hoeven geen geheime sleutels meer uitgewisseld te worden en meerdere personen kunnen met dezelfde publieke sleutel een gecodeerd bericht versturen aan degene met de geheime sleutel, zonder dat de verschillende zenders elkaars berichten kunnen lezen.



Hoe dit concreet in zijn werk gaat, zal ik in de volgende hoofdstukken uit de doeken doen.

3. Wiskundig principes

Voordat ik uit ga leggen hoe het RSA Cryptosysteem in zijn werk gaat, zal ik eerst enkele stellingen en wiskundige principes uiteenzetten die ten grondslag liggen aan het RSA systeem en aan de cryptografie in het algemeen.

3.1 Modulo-rekenen

Het meest bekende voorbeeld van modulorekenen, staat ook wel bekend als ‘klokrekenen’. Het is bijvoorbeeld 10 uur en iemand vraagt hoe laat het over 31 uur is. Omdat het 24 later precies dezelfde tijd op de dag is, kan vrijwel iedereen bedenken dat het dan 17 uur zal zijn. Immers $31 - 24 = 7$ en $10 + 7 = 17$. In dit geval zegt men dat men modulo 24 werkt, of beter gezegd *41 is congruent met 17 modulo 24* (te noteren als $41 \equiv 17 \pmod{24}$). Ook geldt dat $65 \equiv 17 \pmod{24}$). Verder is het zo dat $24 \equiv 0 \pmod{24}$.

Om een algemene definitie van modulorekenen te geven moet eerst worden vastgesteld wat onder het begrip deelbaarheid verstaan wordt.

Definitie: Als a en b gehele getallen zijn, dan is b deelbaar door a als er een geheel getal c bestaat zodanig dat $b = c \times a$. Men spreekt dan ook wel van a deelt b . Dit noteert men met $a|b$.

Een belangrijke stelling bij de definitie van deelbaarheid:

Stelling 3.1.1: *Laat x , y en z gehele getallen zijn en $x = y + z$. Als er een geheel getal d bestaat zodat $d|y$ en $d|z$ dan volgt $d|x$.*

Deze stelling kan als volgt bewezen worden: Als $d|y$ en $d|z$ dan bestaan er, volgens de hierboven gegeven definitie van deelbaarheid, gehele getallen c en k zodat $y = c \times d$ en $z = k \times d$. Er is gegeven dat $x = y + z$, dus $x = c \times d + k \times d$, oftewel $x = d \times (c + k)$. $(c + k)$ levert een geheel getal op, dus hiermee is aangetoond dat $d|x$.

Nu de definitie van modulorekenen:

Definitie: Laat m een positief geheel getal zijn. Als a en b gehele getallen zijn, dan zegt men dat a congruent is met b modulo m als $a - b$ deelbaar is door m , oftewel $m|(a - b)$. Dit noteert men met $a \equiv b \pmod{m}$.

Een andere manier om tegen de definitie aan te kijken is als volgt:

Als a en b gehele getallen zijn, dan betekent de uitdrukking $a \equiv b \pmod{m}$, dat er een geheel getal k bestaat zodat $a = b + km$. Omgekeerd, als er een geheel getal k bestaat zodanig dat $a = b + km$, dan geldt dat $a \equiv b \pmod{m}$.

Immers, als $a \equiv b \pmod{m}$ betekent dit volgens de definitie dat $m|(a - b)$. Dus is er een getal k te vinden zodanig dat $km = a - b$, maar dat betekent dus dat $a = b + km$. Ditzelfde geldt omgekeerd.

Verder zijn de volgende stellingen van groot belang bij het modulorekenen:

Stelling 3.1.2: *Laat a, b, c, d en k gehele getallen zijn en laat m een positief geheel getal zijn. Als $a \equiv b \pmod{m}$ en $c \equiv d \pmod{m}$ dan geldt*

1. $a + c \equiv b + d \pmod{m}$
2. $a \times c \equiv b \times d \pmod{m}$

Bewijs:

1. Uit $a \equiv b \pmod{m}$ en $c \equiv d \pmod{m}$ volgt volgens de definitie van modulorekenen meteen $m \mid (a - b)$ en $m \mid (c - d)$. Dus volgt (zie stelling 3.1.1) $m \mid (a - b) + (c - d)$, ofwel $m \mid (a + c) - (b + d)$. Dus $a + c \equiv b + d \pmod{m}$.

2. Als men schrijft $a \times c - b \times d = (a - b)c + b(c - d)$, dan volgt uit het gegeven, $m \mid (a - b)$ en $m \mid (c - d)$, meteen $m \mid (a \times c - b \times d)$, dus $a \times c \equiv b \times d \pmod{m}$

Voorbeelden:

Optellen:

$$15 + 31 \equiv 6 + 4 \pmod{9} \equiv 10 \pmod{9} \equiv 1 \pmod{9},$$

want $15 = 6 + 1 \times 9$, dus $15 \equiv 6 \pmod{9}$ en $31 = 4 + 3 \times 9$, dus $31 \equiv 4 \pmod{9}$.

$$124 + 18 \equiv 16 + 0 \pmod{18} \equiv 16 \pmod{18},$$

want $124 = 16 + 6 \times 18$, dus $124 \equiv 16 \pmod{18}$ en $18 = 0 + 1 \times 18$, dus $18 \equiv 0 \pmod{18}$.

Vermenigvuldigen:

$$12 \times 19 \equiv 2 \times 4 \pmod{5} \equiv 8 \pmod{5} \equiv 3 \pmod{5},$$

want $12 \equiv 2 \pmod{5}$ en $19 \equiv 4 \pmod{5}$.

$$20 \times 27 \equiv -1 \times -1 \pmod{7} = 1 \pmod{7},$$

want $20 \equiv -1 \pmod{7}$ en ook $27 \equiv -1 \pmod{7}$.

Naast optellen en vermenigvuldigen, is bij modularekenen ook mogelijk om een vermenigvuldiging weer ongedaan te maken. Een soort van delen dus, maar dan anders. Bij modularekenen kan men namelijk niet met breuken werken, dus de vorm $q/p \pmod{m}$ is geen optie.

Het ongedaan maken van een vermenigvuldiging, oftewel het vinden van een *inverse* bewerking, gaat bij modularekenen als volgt:

Stel men heeft een vermenigvuldiging

$$a \times x \pmod{m}$$

Wil men nu x terugkrijgen, dan is het de bedoeling een getal b te vinden zodat

$$b \times a \equiv 1 \pmod{m}$$

Dat betekent dus, dat ‘vermenigvuldigen met b ’ precies de operatie ‘vermenigvuldigen met a ’ ongedaan maakt. Kortom, ‘delen door a ’ is hetzelfde als ‘vermenigvuldigen met b ’:

$$b \times (a \times x) = (b \times a) \times x \equiv 1 \times x \pmod{m} = x \pmod{m}$$

Het getal b , noemt men de *inverse van a* . Omgekeerd is a de inverse van b , kortom a en b zijn elkaars inverse (modulo m gerekend), want

$$a \times b = b \times a = 1 \pmod{m}$$

Voorbeeld: Modulo 11 zijn 5 en 9 elkaars inverse, want

$$5 \times 9 = 9 \times 5 = 45 = 1 + (4 \times 11) \equiv 1 \pmod{11}$$

Men kan een hele tabel geven van getallen en hun inverse modulo 11:

Getal a:	1	2	3	4	5	6	7	8	9	10
Inverse van a:	1	6	4	3	9	2	8	7	5	10

Tabel 3.1: Inversen-tabel modulo 11.

Het lijkt er nu op dat een inverse altijd bestaat en daarbij ook uniek is. Het eerste is zeker niet waar en het tweede eigenlijk ook niet, maar daar heeft men een afspraak over gemaakt.

Als men kijkt naar de volgende vermenigvuldiging

$$20 \times 5 = 100 \equiv 1 \pmod{11}$$

dan kan men zeggen dat ook 20 een inverse van 5 is. Sterker nog: elk getal dat men kan noteren als $(9 + k \times 11)$ is dan een inverse van 5 modulo 11. De afspraak die nu meestal gemaakt wordt is dat men het getal dat tussen de 0 en 11 ligt (algemeen tussen 0 en m als er modulo m gewerkt wordt) aanduidt met *de* inverse.

Dan blijft nog over de vraag of alle getallen modulo een willekeurige m een inverse hebben. Het is meteen duidelijk dat 0 geen inverse kan hebben (delen door 0 kan ook niet), immers er is geen getal b te vinden zodat $b \times 0 = 1 \pmod{m}$.

Maar als er modulo 11 gerekend wordt, heeft buiten het getal 0 ieder ander getal een inverse [zie tabel 3.1]. Probeert men echter een inverse-tabel te maken bij modulo 12, dan blijkt dit niet goed te gaan:

Getal a:	1	2	3	4	5	6	7	8	9	10	11
Inverse van a:	1	-	-	-	5	-	7	-	-	10	11

Tabel 3.2: Inversen-tabel modulo 12.

Slechts voor de getallen 1, 5, 7 en 11 is een inverse te vinden. Hoe komt dit? Men neemt bijvoorbeeld het getal 2. Stel dat er wel een inverse is modulo 12, en noem die b . Dan betekent dit dat $2 \times b \equiv 1 \pmod{12}$, oftewel: er is een geheel getal k zodat

$$2 \times b = 1 + 12 \times k$$

Dit is natuurlijk onmogelijk! Want $2 \times b$ is altijd *even* en $1 + 12 \times k$ is altijd *oneven*. Een dergelijke redenering geldt ook voor 4, 6, 8 en 10. Bij 3 en 9 gaat het net zo: stel bijvoorbeeld dat c modulo 12 gerekend de inverse is van 3. Dan betekent dit dat er een geheel getal k bestaat zodat

$$3 \times c = 1 + 12 \times k$$

Maar nu is de linkerkant altijd een drievoud en de rechterkant nooit een drievoud (namelijk altijd een drievoud-plus-één). Kortom modulo 12 heeft ook 3 geen inverse. Voor ik een algemene stelling voor deze redenering kan omschrijven moet ik eerst een definitie geven van het begrip grootste gemene deler.

Definitie: De grootste gemene deler (GGD) van twee gehele getallen a en b , die niet beide 0 zijn, is het grootste gehele getal dat zowel a als b deelt. Ik zal de GGD in het vervolg noteren met $\text{GGD}(a, b)$. Verder is afgesproken dat $\text{GGD}(0, 0) = 0$.

Als voor twee gehele getallen a en b geldt dat $\text{GGD}(a, b) = 1$, dan noemt men a en b *relatief priem*. Deze eigenschap is nodig voor het bestaan van een inverse. Zo zijn bijvoorbeeld de getallen 4 en 5 relatief priem, want het grootste gehele getal waar men beide getallen door kan delen is 1.

Stelling 3.1.3: Laat a en m gehele getallen zijn met $m > 0$ en $a \neq 0$. De vergelijking

$$a \times x = 1 \pmod{m}$$

heeft oplossingen x , de inversen van a modulo m , dan en slechts dan als $\text{GGD}(a, m) = 1$.

Als m_p een priemgetal is, geldt automatisch dat $\text{GGD}(a, m_p) = 1$ voor $a = 1, 2, 3, \dots, (m_p - 1)$. Dus op $0 \pmod{m_p}$ na, hebben modulo m_p gerekend alle getallen een inverse. Maar nemen we bijvoorbeeld het getal 8 uit de inversen-tabel van modulo 12, dan heeft de vergelijking $8 \times x = 1 \pmod{12}$ geen oplossing, want $\text{GGD}(8, 12) = 4 \neq 1$.

3.2 Euclides

De grootste gemene deler van twee gehele getallen speelt een belangrijke rol in de getallentheorie. Bij kleine getallen is het vaak vrij eenvoudig om de GGD te bepalen, bij grotere getallen wordt het moeilijker. De wiskundige Euclides heeft echter een methode gevonden om snel deze GGD te vinden.

Voor deze methode maakt men gebruik van het zogenaamde delingsalgoritme.

Stelling 3.2.1 (Delingsalgoritme): *Laat a en b twee gehele getallen zijn met $b > 0$. Dan bestaat er een uniek paar gehele getallen q en r zó dat $a = q \times b + r$, met $0 \leq r < b$. Men noemt q het quotiënt en r de rest.*

Voorbeeld:

$$a = 133, b = 21 \text{ dan zijn } q = 6, r = 7, \text{ want } 133 = 6 \times 21 + 7$$

Een methode om snel de GGD te bepalen staat bekend onder de naam “Het algoritme van Euclides”. Vooral met de computer gaat het razendsnel. De methode is gebaseerd op de volgende constatering:

Constatering: Laat c en d gehele getallen zijn waarvoor geldt $c = q \times d + r$, met q en r gehele getallen. Dan is $\text{GGD}(c, d) = \text{GGD}(d, r)$.

Voordat ik het algemene algoritme geef, eerst een voorbeeld:

Stel men wil de GGD van 252 en 198 bepalen.

$$252 = 1 \times 198 + 54$$

Volgens bovenstaande constatering geldt nu $\text{GGD}(252, 198) = \text{GGD}(198, 54)$, dus neemt men nu de getallen 198 en 54:

$$198 = 3 \times 54 + 36$$

Dus $\text{GGD}(198, 54) = \text{GGD}(54, 36)$. Tenslotte volgt

$$54 = 1 \times 36 + 18$$

$$36 = 2 \times 18 + 0$$

Dus $\text{GGD}(54, 36) = \text{GGD}(36, 18) = 18$. Hieruit volgt dat ook $\text{GGD}(252, 198) = 18$.

Algemeen: stel men wil de $\text{GGD}(a, b)$ bepalen waarbij $a \geq b > 0$. Verder neemt men $r_0 = a$ en $r_1 = b$.

Volgens het delingsalgoritme geldt nu $a = q \times b + r$, of uitgedrukt in r_j : $r_0 = q_1 \times r_1 + r_2$. Als men hierop nu het algoritme van Euclides los laat krijgt men achtereenvolgens:

$$\begin{aligned} r_0 &= q_1 \times r_1 + r_2 \\ r_1 &= q_2 \times r_2 + r_3 \\ &\vdots \\ r_{j-1} &= q_j \times r_j + r_{j+1} \\ r_j &= q_{j+1} \times r_{j+1} + 0 \end{aligned}$$

Dus als $r_{j+2} = 0$ dan is $\text{GGD}(a, b) = r_{j+1}$.

Stelling 3.2.2 (Het Euclidisch Algoritme): Laat $r_0 = a$ en $r_1 = b$ gehele getallen zijn met $a \geq b > 0$. Als men nu het delingsalgoritme

$$r_j = q_{j+1} \times r_{j+1} + r_{j+2}, \text{ met } 0 \leq r_{j+2} < r_{j+1}$$

herhaaldelijk toepast totdat $r_{j+2} = 0$ voor zekere j , dan is de GGD van a en b gelijk aan de rest r_{j+1} . Dus $\text{GGD}(a, b) = r_{j+1}$.

Nog een voorbeeld van het algoritme van Euclides:

Men bepaalt $\text{GGD}(222, 102)$.

$$\begin{aligned} 222 &= 2 \times 102 + 18, \\ 102 &= 5 \times 18 + 12, \\ 18 &= 1 \times 12 + 6, \\ 12 &= 2 \times 6 + 0. \end{aligned}$$

Dus $\text{GGD}(222, 102) = 6$.

Men kan dit voorbeeld ook de andere kant op lezen:

$$\begin{aligned} 6 &= 18 - 1 \times 12 \\ 6 &= 18 - (102 - 5 \times 18) = 6 \times 18 - 102 \\ 6 &= 6 \times (222 - 2 \times 102) - 102 \\ 6 &= 6 \times 222 - 13 \times 102 \end{aligned}$$

Nu is $6 = \text{GGD}(222, 102)$ dus uitgedrukt als een lineaire combinatie van 222 en 102. Dit kan men doen bij elke GGD .

Als men deze methode toepast op twee getallen a en b waarvan $\text{GGD}(a, b) = 1$, dan kan men op een makkelijke manier de inverse van $a(\text{mod } b)$ en $b(\text{mod } a)$ vinden.

Voorbeeld: Neem twee getallen a en m waarvoor geldt $\text{GGD}(a, m) = 1$ [zie stelling 3.1.3].

$$a = 27, m = 32$$

Pas nu het algoritme van Euclides toe:

$$32 = 1 \times 27 + 5$$

$$27 = 5 \times 5 + 2$$

$$5 = 2 \times 2 + 1$$

$$2 = 2 \times 1 + 0$$

Zoals verwacht is $\text{GGD}(27, 32) = 1$. Als men het algoritme in omgekeerde volgorde langsloopt krijgt men:

$$1 = 5 - 2 \times 2$$

$$1 = 5 - 2 \times (27 - 5 \times 5) = 11 \times 5 - 2 \times 27$$

$$1 = 11 \times (32 - 1 \times 27) - 2 \times 27 = 11 \times 32 - 13 \times 27$$

Als men dit modulo 32 bekijkt, krijgt men:

$$1 = 11 \times 32 - 13 \times 27 \equiv -13 \times 27 \pmod{32} \equiv 19 \times 27 \pmod{32}$$

Dus modulo 32 gerekend is 19 de inverse van 27.

3.3 Fermat

Een van de belangrijkste stellingen in de moderne cryptografie (RSA) is de zogenaamde “kleine stelling van Fermat”, gedefinieerd door de wiskundige Pierre Fermat die leefde van 1601 tot 1665. Voordat ik de stelling geef, zal ik eerst een aantal wiskundige principes bespreken die eraan ten grondslag liggen.

Ter illustratie neem ik een vermenigvuldigingstabel bij het rekenen modulo 7:

×	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

Tabel 3.3: Vermenigvuldigingstabel modulo 7

In deze tabel heb ik niet met 0 vermenigvuldigd, want dat levert toch steeds 0 op. Opvallend is hier dat iedere regel dezelfde getallen bevat, alleen in een andere volgorde. Dit is niet altijd het geval:

×	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	0	2	4	6
3	3	6	1	4	7	2	5
4	4	0	4	0	4	0	4
5	5	2	7	4	1	6	3
6	6	4	2	0	6	4	2
7	7	6	5	4	3	2	1

Tabel 3.4: Vermenigvuldigingstabel modulo 8

Hierin valt op dat alleen in rijen waar het vermenigvuldigingsgetal en het modulo-getal geen gemeenschappelijke deler hebben, dus GGD is 1, de gehele reeks 1,2,3,...7 voorkomt. In rijen waar dit niet het geval is kan bijvoorbeeld nooit een 1 voorkomen, aangezien de vergelijking $a \times b \equiv 1 \pmod{m}$ alleen een oplossing heeft als $\text{GGD}(a, m) = 1$ [zie stelling 3.1.3].

Men kan dit als volgt in een stelling formuleren:

Stelling 3.3.1: *Laat a en m gehele getallen zijn met $m > 0$ en $a \in \{1, 2, 3, \dots, m-1\}$. Als de ggd van a en m gelijk is aan 1, dus als $\text{GGD}(a, m) = 1$, dan is in de vermenigvuldigingstabel de rij $a, 2a, 3a, \dots, (m-1)a \pmod{m}$ op volgorde na gelijk aan de rij $1, 2, 3, \dots, m-1$.*

Als men voor m een priemgetal p neemt, dan geldt natuurlijk dat $\text{GGD}(a, p) = 1$ voor $a = 1, 2, 3, \dots, p-1$. Daarom hadden in de vermenigvuldigingstabel van modulo 7 [tabel 3.3] alle rijen op de volgorde na dezelfde getallen.

Stel men neemt een vermenigvuldigingstabel waarbij modulo p gerekend wordt, waarbij p dus een priemgetal is. Zoals hiervoor besproken komen dan ik elke rij de getallen 1 t/m $p - 1$ voor. Men neemt nu het product P van deze getallen:

$$P \equiv 1 \times 2 \times 3 \times \dots \times (p - 1) \pmod{p}$$

De uitkomst is nu weer één van de getallen $1, 2, 3, \dots, p - 1$. De uitkomst is in dit geval niet zo belangrijk. Wel belangrijk is dat geldt:

$$P \times P \equiv 1 \pmod{p}$$

Dit kan men als volgt uitleggen: omdat $\text{GGD}(a, p) = 1$ voor $a \in \{1, 2, 3, \dots, p - 1\}$, hebben al die getallen a een inverse modulo p [zie stelling 3.1.3]. Als men dan in het product $P \times P$ steeds de a aan zijn inverse in de tweede P koppelt, dan krijgt men koppels die, modulo p , 1 als uitkomst hebben. Oftewel men krijgt totaal $p - 1$ koppels die allemaal als product 1 hebben. Het totale product is dus ook gelijk aan 1 .

Dan zal ik nu de “kleine stelling van Fermat” formuleren.

Stelling 3.3.2 (Kleine stelling van Fermat): *Als p een priemgetal is, dan geldt voor iedere a met $\text{GGD}(a, p) = 1$ dat $a^{p-1} \equiv 1 \pmod{p}$.*

Deze stelling is als volgt te bewijzen: men stelt het product P gelijk aan de getallen van de a -de rij van de vermenigvuldigingstabel. Modulo p gezien is dit ook gelijk aan $a, 1a, 2a, \dots, (p - 1)a$. Kortom,

$$\begin{aligned} 1a \times 2a \times 3a \times \dots \times (p - 1)a &\equiv P \pmod{p} \\ a^{p-1} (1 \times 2 \times 3 \times \dots \times (p - 1)) &\equiv P \pmod{p} \end{aligned}$$

Omdat $P = (1 \times 2 \times 3 \times \dots \times (p - 1))$ geldt dus:

$$a^{p-1} P \equiv P \pmod{p}$$

Als men nu links en rechts met P vermenigvuldigd en hierbij bedenkt dat $P \times P \equiv 1 \pmod{p}$, dan is de bovenstaande stelling als volgt bewezen:

$$\begin{aligned} a^{p-1} \times P \times P &\equiv P \times P \pmod{p} \\ a^{p-1} \times 1 &\equiv 1 \pmod{p} \\ a^{p-1} &\equiv 1 \pmod{p}. \end{aligned}$$

3.4 Euler

Met de stelling van Fermat kan men alleen modulo een priemgetal werken. De stelling van de Zwitserse wiskundige Leonhard Euler is een uitbreiding hierop, zodat men ook andere getallen kan gebruiken.

Ter illustratie neem ik weer de vermenigvuldigingstabel van modulo 8, maar nu met alleen de producten $a \times b$ waarvoor geldt $\text{GGD}(a, b) = 1$, oftewel rijen die op volgorde na gelijk zijn:

\times	1	3	5	7
1	1	3	5	7
3	3	1	7	5
5	5	7	1	3
7	7	5	3	1

Tabel 3.5: Gereduceerde vermenigvuldigingstabel modulo 8

In deze rijen komen precies de getallen voor waarvoor geldt $\text{GGD}(a, 8) = 1$. Het aantal getallen in zo'n rij drukt men uit met een functie die door Euler is geïntroduceerd: de Euler phi-functie (schrijf $\phi(x)$), in dit geval levert de functie $\phi(8)$ dus 4 op.

Men kan dit vatten in de volgende definitie:

Definitie: Laat m een positief geheel getal zijn. De Euler phi-functie $\phi(m)$ geeft het aantal getallen weer dat niet groter is dan m en dat relatief priem is met m .

Omdat ook bij de gereduceerde vermenigvuldigingstabel modulus een willekeurige m de getallen op volgorde na gelijk blijken te zijn, kan men voor deze tabel een zelfde soort stelling als de stelling van Fermat formuleren [zie stelling 3.3.2].

Stelling 3.4.1 (Stelling van Euler): Laat m een positief geheel getal zijn en laat a een geheel getal zijn waarvoor geldt $\text{GGD}(a, m) = 1$. Dan geldt

$$a^{\phi(m)} \equiv 1 \pmod{m}.$$

Ik zal nu op dezelfde wijze als bij de stelling van Fermat, een bewijs voor deze stelling geven: men neemt het product P van de getallen in een rij a van de gereduceerde vermenigvuldigingstabel. Deze getallen vermenigvuldigt men weer met a , bij modulo 8 levert dit het volgende op:

$$\begin{aligned} P &\equiv 1a \times 3a \times 5a \times 7a \pmod{8} \\ &\equiv a^{\phi(8)} (1 \times 3 \times 5 \times 7) \equiv a^{\phi(8)} P \pmod{8} \end{aligned}$$

Dus:

$$a^{\phi(8)} P \equiv P \pmod{8}$$

Links en rechts weer met P vermenigvuldigen geeft dan de stelling van Euler, immers net als bij de stelling van Fermat geldt $P^2 \equiv 1 \pmod{8}$.

$$a^{\phi(8)} \equiv 1 \pmod{8}.$$

Voor een priem getal p is $\phi(p)$ natuurlijk heel makkelijk te berekenen, immers alle getallen onder p zijn relatief priem met p , dus $\phi(p) = p - 1$. Verder geldt er nog de volgende belangrijke stelling voor het rekenen met de phi-functie van Euler:

Stelling 3.4.2: *Als twee positieve getallen m en n relatief priem zijn dan is*

$$\phi(m \times n) = \phi(m) \times \phi(n)$$

Deze stelling en de stellingen van Euler vormen de basis voor het moderne cryptografie systeem RSA, dat ik in het volgende hoofdstuk zal behandelen.

4. Het RSA-systeem

In dit hoofdstuk zal ik het tegenwoordig veel gebruikte RSA-cryptosysteem bespreken en uitleggen. De letters RSA zijn afgeleid van de beginletters van de namen van de mannen die het systeem in 1978 hebben bedacht: Rivest, Shamir en Adleman.



Bedenkers van RSA: Rivest, Shamir en Adleman (van links naar rechts)

RSA is een asymmetrisch systeem [zie hoofdstuk 2], waarbij dus iedereen met dezelfde openbare sleutel berichten kan coderen, zodat alleen de ontvanger met de geheime sleutel deze kan lezen. Het systeem is gebaseerd op het feit dat het heel moeilijk en tijdrovend is om grote getallen in factoren te ontbinden, terwijl het vermenigvuldigen van deze factoren in een oogwenk kan gebeuren (met een computer).

Eerst zal ik uitleggen hoe het coderen met RSA theoretisch in zijn werk gaat en daarna hoe men het bericht vervolgens weer moet decoderen. Ook zal ik enkele praktische problemen bij het versleutelen bespreken en de manier waarop men die oplost. Tot slot zal ik een aantal hedendaagse toepassingen noemen van het RSA systeem.

4.1 Coderen (versleutelen)

Stel er zijn twee personen Anna en Bob die elkaar een bericht willen sturen dat niemand anders mag lezen. Anna is degene die het bericht ontvangt en Bob degene die het bericht wil versturen.

Anna neemt nu twee grote gehele getallen, van minstens 110 cijfers, p en q , waarvan is vastgesteld dat het beide priemgetallen zijn. Met deze getallen berekend ze $n = p \times q$. Dit getal maakt ze openbaar. Daarnaast berekend ze ook $\phi(n) = \phi(p \times q) = \phi(p) \times \phi(q) = (p - 1)(q - 1)$, en houdt de uitkomst geheim. Verder kiest ze een willekeurig geheel getal e waarvoor geldt dat e relatief priem is met $\phi(n)$, oftewel $\text{GGD}(e, \phi(n)) = 1$. Ook e maakt ze openbaar. Ze weet dat e een inverse d heeft modulo $\phi(n)$, deze inverse houdt ze ook geheim. Dit is namelijk de geheime sleutel die bij het decoderen gebruikt wordt. Dus $e \times d \equiv 1 \pmod{\phi(n)}$, waarbij d de geheime en e de openbare sleutel is.

Een overzicht van openbare en geheime getallen:

Openbaar	Geheim
n ($n = p \times q$)	$\phi(n)$
e (relatief priem met $\phi(n)$)	d (inverse van $e \pmod{\phi(n)}$)
	p (priemgetal)
	q (priemgetal)

Anna en Bob gebruiken een ‘alfabet’ $1, 2, 3, \dots, M$ en zorgen dat $M < p$ en $M < q$. Omdat elk getal kleiner of gelijk aan M te ontbinden is in priemfactoren kleiner of gelijk aan M , is elk getal kleiner of gelijk aan M relatief priem met p en met q . Dus alle ‘letters’ van het alfabet $1, 2, 3, \dots, M$ zijn relatief priem met n . Dat dit heel belangrijk is zal ik later uitleggen.

Als Bob nu een letter L uit het alfabet $\{1, 2, 3, \dots, M\}$ wil versturen, versleutelt hij de letter als volgt:

$$V \equiv L^e \pmod{n}$$

V stuurt hij naar Anna.

4.2 Decoderen (ontcijferen)

Anna ontvangt nu de versleutelde letter V ($V \equiv L^e \pmod{n}$) van Bob.

Om de letter L terug te krijgen berekent zij

$$L \equiv V^d \pmod{n}.$$

Waarom gaat dit goed? Ten eerste zijn alle letters uit het gebruikte alfabet relatief priem met het product $n = pq$. Dus geldt

$$\text{GGD}(L, n) = 1.$$

En volgens de stelling van Euler [zie stelling 3.4.1] geldt dan

$$L^{\phi(n)} \equiv 1 \pmod{n}.$$

Omdat $de \equiv 1 \pmod{\phi(n)}$ is er een getal k zo dat $de = 1 + k\phi(n)$.

Dus Anna vindt

$$V^d \equiv (L^e)^d \equiv L^{de} \equiv L^{1+k\phi(n)} \equiv L \times L^{k\phi(n)} \equiv L \times (L^{\phi(n)})^k \equiv L \times 1 \pmod{n}.$$

Anna kan dus elke versleutelde letter die Bob stuurt, decoderen.

4.3 Versleutelen in de praktijk

Ik zal de problemen waar men in de praktijk rekening mee moet houden uitleggen aan de hand van een voorbeeld. Voor het gemak ga ik er weer vanuit dat Bob een versleuteld bericht aan Anna wil sturen. Verder werk ik in dit voorbeeld met kleinere RSA sleutels dan normaal gebruikt zouden worden.

Anna kiest twee priemgetallen: $p = 268435459$ en $q = 268435463$ en berekent het openbare product

$$n = pq = 72057596722282517$$

en het geheime product

$$\phi(n) = (p - 1)(q - 1) = 72057596185411596.$$

Nu kiest ze een willekeurige e zo dat $\text{GGD}(e, \phi(n)) = 1$. Ze neemt bijvoorbeeld $e = 87221$. De inverse van $e \pmod{\phi(n)}$ is $d = 5613686681875601$. Deze getallen berekent ze met de computer. De getallen e en n zijn openbaar en p , q , $\phi(n)$ en d niet.

Stel Bob wil het bericht ‘Kom!’ naar Anna versturen, hij zou nu met Anna kunnen afspreken om elke letter van het alfabet te nummeren van 1 t/m 26, maar met *frequentie analyse* zou het bericht dan vrij makkelijk te achterhalen zijn. Bij frequentie analyse, analyseert men het voorkomen van letters in teksten van een bepaalde taal. Door deze frequenties te vergelijken met de frequenties van de ‘letters’ in een gecodeerd bericht, kan men vaak het oorspronkelijke bericht achterhalen. Verder kan men met een alfabet van 26 letters niet speciale tekens zoals hoofdletters, interpunctie en accenten in een bericht opnemen.

Daarom werkt men met speciale tabellen, waarin elke letter, cijfer of symbool een bepaalde letterwaarde krijgt. Deze tabellen zijn gestandaardiseerd en worden dus over de hele wereld gebruikt.

0	NULL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BELL
8	BS	9	HT	10	LF	11	VT	12	FF	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	spatie	33	!	34	“	35	#	36	\$	37	%	38	&	39	‘
40	(41)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[92	\	93]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	r	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124		125	}	126	~	127	del

Tabel 4.1: De ISO-tabel van computertekens

Het bericht 'Kom!' kan men dus als volgt vertalen naar letterwaarden:

K	o	m	!
75	111	109	33

Om frequentie analyse onmogelijk te maken, codeert men in plaats van aparte letters, lettergroepen. Bijvoorbeeld groepen van 4 letters. Een manier om dit te doen is door groepjes van 4 letterwaardes in het 128-tallige stelsel om te zetten. Net als 408 in het decimale stelsel gelijk is aan $(4, 0, 8)_{10}$ of $4 \times 10^2 + 0 \times 10^1 + 8 \times 10^0$, is $(75, 111, 109, 33)_{128}$ in het 128-tallige stelsel gelijk aan

$$75 \times 128^3 + 111 \times 128^2 + 109 \times 128^1 + 33 \times 128^0 = 159119009 \text{ (decimaal).}$$

Dit is het getal dat Bob kan versleutelen met de publiek sleutel e van Anna. Dus Bob doet

$$159119009^e \equiv 63796541030234435 \pmod{n}. \quad (e = 87221)$$

Dit getal verstuurd Bob naar Anna. Als Anna het bericht wil lezen, moet ze het ontvangen getal eerst weer decoderen met haar geheime sleutel, oftewel

$$63796541030234435^d \equiv 159119009 \pmod{n}. \quad (d = 5613686681875601)$$

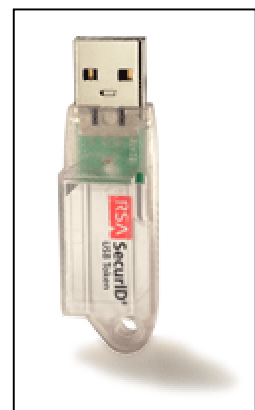
4.4 Toepassingen

RSA wordt vooral gebruikt bij het versturen van versleutelde berichten over de elektronische snelweg: het internet. Banken gebruiken het bijvoorbeeld bij het overboeken van geld via het internet. Dat dit soort gegevens niet in handen van derden mogen vallen is natuurlijk duidelijk. Vaak, omdat RSA nogal wat rekenkracht vergt, wordt RSA enkel gebruikt om geheime sleutels van symmetrische cryptografie systemen, zoals DES, veilig over onveilige kanalen (internet) te versturen. De rest van het bericht is dan bijvoorbeeld gecodeerd met het snellere DES.

Maar ook dichterbij huis wordt RSA tegenwoordig veel gebruikt. Bijvoorbeeld bij het zetten van een digitale handtekening in een e-mail. RSA bewaakt hierbij de authenticiteit [zie introductie] van het e-mail bericht. Dit gaat als het volgt in zijn werk:

Anna, die in dit geval een bericht wil verzenden, ondertekent haar bericht aan Bob met bijvoorbeeld haar naam. Deze handtekening codeert ze vervolgens met haar *geheime* sleutel. Het ondertekende bericht verstuurt ze naar Bob. Nu kan Bob de bijpassende *openbare* sleutel van Anna gebruiken om de gecodeerde handtekening weer leesbaar te maken. Lukt dit, dus komt de naam van Anna te voorschijn, dan weet Bob *zeker* dat het bericht afkomstig moet zijn geweest van Anna. Immers, zij is, als het goed is, de enige die de geheime sleutel kan hebben gebruikt!

Verder zijn er nog tal van andere handige toepassingen bedacht voor het publieke-sleutel systeem. Zo zijn er bijvoorbeeld speciale USB-sticks, beveiligd met RSA, waarop men allerlei gevoelige gegevens zoals wachtwoorden en digitale certificaten kan opslaan.



5. Number Field Sieve

Omdat RSA door veel grote instellingen, zoals banken en het leger, wordt gebruikt, is de vraag natuurlijk: is het wel veilig? Kan de code niet gekraakt worden?

Er zijn verschillende methodes ontwikkeld om RSA met een computer kraken. De snelste methode op dit moment is de Number Field Sieve. In dit hoofdstuk zal ik bespreken wat de achterliggende gedachtes zijn bij deze methode. Maar eerst zal ik uitleggen waarom RSA eigenlijk als veilig beschouwd wordt.

5.1 Waarom is RSA veilig?

Als men een bericht, gecodeerd met RSA, wil ontcijferen, heeft men de geheime sleutel d nodig. Zoals ik in het vorige hoofdstuk heb besproken is d de inverse van het publieke getal e modulo $\phi(n)$. Dus als men $\phi(n)$ weet, kan men vrij snel de geheime sleutel d vinden. Daarom wordt $\phi(n)$ geheim gehouden en alleen $n (= pq)$ publiek gemaakt.

Eerder heb ik laten zien dat $\phi(n) = (p - 1)(q - 1)$, dus het vinden van $\phi(n)$ kan men ook zien als het vinden van één van de priemfactoren p of q van n . Immers als men één van deze factoren weet kan men ook de andere berekenen en daarmee $\phi(n)$.

Computers kunnen razendsnel grote getallen vermenigvuldigen, maar delen gaat een stuk langzamer. Men zou voor het vinden van p en q een methode kunnen bedenken, waarbij de computer n deelt door elk priemgetal, lager dan \sqrt{n} . Dit principe wordt ook wel trial division genoemd. Met deze methode vindt men gegarandeerd een deler van n , maar voor hele grote getallen kan de benodigde tijd behoorlijk oplopen.

Een klein rekenvoorbeeld: stel men wil de factoren van een geheel getal n van 60 cijfers vinden. Men moet dan alle priemgetallen tot een grootte van ongeveer 10^{30} uitproberen. Als we van het gunstige geval uit gaan, dat slechts 0,1% van deze getallen priemgetallen zijn, betekent dat, dat de computer ongeveer 10^{27} delingen moet uitvoeren. Als we er weer optimistisch vanuit gaan dat er genoeg computercapaciteit beschikbaar is om 10^{15} van deze delingen per seconden te verrichten, dan duurt dat nog steeds grofweg 10^{12} seconden, oftewel 31.709 jaar. Natuurlijk kan het zijn dat men niet helemaal tot de wortel van n hoeft te gaan om de deler te vinden, dus misschien duurt het duizend jaar minder. Aan de andere kant, de gemaakte schatting ging uit van optimale omstandigheden, dus waarschijnlijk duurt het langer. Men praat hier in elk geval over een proces dat duizenden jaren kan duren, tegen die tijd is de mens misschien al weer uitgestorven!

Voor de meeste problemen zijn er echter meerde oplossingen. De methodes die tot nu toe het meeste succes hebben behaald, maken allemaal gebruik van hetzelfde basisprincipe. Deze methodes maken gebruik van een slimmere manier om factoren van n te vinden, desondanks kan het ook met deze methodes jaren duren. Eén van die succesvolle methodes is de Number Field Sieve. Het grootste product n dat tot nu toe is gefactoriseerd, is een getal van 174 decimale cijfers.

Zoals nu wel duidelijk zal zijn is de veiligheid van RSA evenredig met de grootte van het product n . Tegenwoordig gebruiken de meeste organisaties, RSA-getallen, bestaande uit minimaal 1024 bits, oftewel een decimaal getal van 309 cijfers!

5.2 Kraken, het principe van de Number Field Sieve (NFS)

Omdat trial division teveel tijd in beslag neemt, wordt bij de moderne factoriseringsmethoden gebruik gemaakt van een techniek die dateert uit de tijd van Fermat, de zogenaamde “verschil van twee kwadraten” techniek.

Als men de volgende uitdrukking heeft

$$x^2 - y^2$$

kan men die als volgt ontbinden in factoren

$$(x - y)(x + y).$$

Deze techniek past men ook toe bij het factoriseren van het RSA-getal n . Het probleem van factoren van n vinden wordt er dus één van getallen x 'en en y 'en vinden die voldoen aan de voorwaarde

$$x^2 \equiv y^2 \pmod{n},$$

want er bestaat dan een getal k zo dat $x^2 - y^2 = kn$ [zie paragraaf 3.1]. Hieruit volgt dat $\text{GGD}(x - y, n)$ en $\text{GGD}(x + y, n)$ niet-triviale factoren van n zijn. En aangezien n , het product is van twee priemgetallen p en q , blijkt de kans $2/3$ dat men één van die factoren vindt op deze manier. Dit kan men illustreren aan de hand van de volgende tabel ($p|(x + y)$ betekent, p deelt $(x + y)$ [zie paragraaf 3.1]):

$p (x + y) ?$	$p (x - y) ?$	$q (x + y) ?$	$q (x - y) ?$	$\text{GGD}(x + y, n)$	$\text{GGD}(x - y, n)$	Priemfactor?
Ja	Ja	Ja	Ja	n	n	Nee
Ja	Ja	Ja	Nee	n	p	Ja
Ja	Ja	Nee	Ja	p	n	Ja
Ja	Nee	Ja	Ja	n	q	Ja
Ja	Nee	Ja	Nee	n	1	Nee
Ja	Nee	Nee	Ja	p	q	Ja
Nee	Ja	Ja	Ja	q	n	Ja
Nee	Ja	Ja	Nee	q	p	Ja
Nee	Ja	Nee	Ja	1	n	Nee

Tabel 5.1: mogelijk situaties voor $x^2 \equiv y^2 \pmod{n}$

In de laatste kolom van de tabel hierboven is te zien dat in 6 van 9 mogelijke gevallen een factor p of q van n wordt gevonden.

Natuurlijk geldt hier, dat hoe sneller de computers, hoe sneller de bewerkingen uitgevoerd kunnen worden. Maar aangezien men veronderstelt dat het aantal priemgetallen oneindig is, is het dus altijd mogelijk om het ook snellere computers moeilijk te maken.

Met de Number Field Sieve probeert men dus op een slimmere en snellere manier factoren van n te vinden dan met trial division. Hoe men precies de x 's en y 's vindt met de NFS is in het kader van dit werkstuk te complex en bovendien is mijn huidige wiskundige kennis op dit gebied (nog) onvoldoende. Maar de hierboven beschreven methode ligt ten grondslag aan de NFS en andere moderne methoden.

5.3 Uitdaging

De bedenkers van RSA hebben een commerciële website, www.rsasecurity.com, in het leven geroepen om producten en diensten te verkopen die gebruik maken van versleuteling. Naast het commerciële gedeelte is er ook een site voor het laboratorium binnen RSA security, www.rsasecurity.com/rsalabs, hier doet men onderzoek naar technieken om toe te passen in nieuwe producten.

Een onderdeel van deze site zijn de zogenaamde ‘challenges’, of uitdagingen. Hier daagt RSA security mensen uit om RSA-getallen n van verschillende groottes, te ontbinden, oftewel te kraken. Er worden grote geldprijzen uitgereikt aan degene die het lukt, en hoe groter het getal, hoe groter de prijs. Door deze uitdagingen heeft RSA security een soort controle op welke grootte van n nog als volkomen veilig beschouwd kan worden.

Hieronder een overzicht van de uitdagingen van dit moment:

Challenge Number	Prize (\$US)	Status	Submission Date	Submitter(s)
RSA-576	\$10,000	Factored	December 3, 2003	J. Franke et al.
RSA-640	\$20,000	Not Factored		
RSA-704	\$30,000	Not Factored		
RSA-768	\$50,000	Not Factored		
RSA-896	\$75,000	Not Factored		
RSA-1024	\$100,000	Not Factored		
RSA-1536	\$150,000	Not Factored		
RSA-2048	\$200,000	Not Factored		

Tabel 5.2: RSA challenge numbers

In RSA-xxx staat xxx voor de grootte van het getal in bits. Zoals te zien in de bovenstaande tabel is van deze getallen alleen RSA-576 gefactoriseerd. Dit is gedaan met een groot netwerk van snelle computers die, verspreid over de hele wereld, parallel gegevens verwerken. Niet iets wat men ‘even’ thuis met een normale pc kan doen dus!

Conclusies

Op basis van voorgaande hoofdstukken is mijn conclusie: theoretisch gezien is het RSA cryptosysteem op dit moment volkomen veilig.

Er zijn echter wel enkele kanttekeningen bij deze bewering te plaatsen. Als er bijvoorbeeld een directe methode wordt gevonden om de factoren p en q of het getal $\phi(n)$ te berekenen uit de openbare sleutel n , dan is het RSA systeem meteen niet meer veilig!

Ook is de veiligheid van het RSA systeem voor een groot deel afhankelijk van de manier waarop er door de gebruikers met de geheime sleutels wordt omgesprongen. Hier zijn wel allerlei procedures voor te bedenken, maar deze moeten dan wel altijd nageleefd worden. En mensen zouden geen mensen zijn als ze hierin nooit fouten zouden maken.

Verder is het bij RSA ook belangrijk, dat de gebruikte getallen voor het maken van de sleutels, volkomen willekeurig gegenereerd zijn. Vaak wordt hiervoor computerprogrammatuur gebruikt. Als een dergelijk programma niet goed omgaat met het maken van willekeurige getallen, en dus getallen genereert die níet volkomen willekeurig zijn, dan kan dit tot onveilige sleutels leiden.

Tevens speelt de ontwikkeling op hardwaregebied een grote rol. Op dit moment doet men nog enkele jaren over het kraken van een sleutel van gemiddelde lengte, bijvoorbeeld 150 cijfers. Maar over 50 jaar, staat misschien de supersnelle kwantumcomputer al voor de deur, die deze getallen in enkele seconden kraakt.

Tot slot is, bij de vraag hoe veilig RSA is, ook de ‘houdbaarheid’ van versleutelde gegevens van groot belang. Voor het versturen van berichten of gegevens die maar korte tijd geheim hoeven te blijven, voldoet waarschijnlijk een relatief kleine sleutel. Tegen de tijd, bijvoorbeeld 10 maanden later, dat die gegevens ontcijferd zijn, is de veiligheid al niet meer van belang. Maar er zijn ook sleutels die langer stand moeten houden, bijvoorbeeld de sleutels die in een computermainframe de toegang tot andere geheime sleutels bewaken! Deze moeten minstens 20 jaar veiligheid garanderen. Deze sleutels zijn dan ook het meest onderhevig aan kraakpogingen.

Appendix

Computerprogramma RSAcrypt

Als praktisch handelingsdeel bij dit profielwerkstuk over RSA, heb ik zelf een computerprogramma geschreven waarmee men willekeurige RSA-sleutels kan generen en stukken tekst kan coderen of decoderen.

In dit programma, gedoopt tot RSAcrypt, heb ik gebruik gemaakt van de theoretische achtergronden in dit werkstuk [zie hoofdstuk 4] en enkele, in dit werkstuk niet besproken, technieken voor het genereren van priemgetallen en het vinden van inversen.

Ik zal nu een korte uitleg geven over de werking van het programma. Het programma is geschreven in Java. Java-programma's worden *geïnterpreteerd*. Dat wil zeggen dat de Java-code niet direct wordt omgezet naar machinetaal, maar dat hiervoor extra software nodig is, in dit geval de zogenaamde Java Virtual Machine (JVM). Deze moet dus eerst op de pc geïnstalleerd worden. Voor het gemak heb ik JVM al op de bijgevoegde cd-rom geïnstalleerd.

Om het programma te gebruiken opent men eerst een DOS venster: Klik op start → Uitvoeren → typ 'cmd' of 'command' (afhankelijk van het besturingssysteem) en druk Enter.

Doe vervolgens de cd in de cd-romspeler en type in het DOS venster:

E:

(waarbij E de stationsnaam van de cd-romspeler is)

Druk Enter en typ:

rsacrypt.bat -optie sleutelmap [tekstbestand]

(zie hieronder voor een voorbeeld)

Voorbeelden:

Ik ga er in dit voorbeeld vanuit dat er een tekstbestand *C:\geheim.txt* op de computer staat met een korte tekst erin. Verder is *C:\sleuteldir* de map met de RSA sleutels. Als deze map niet bestaat wordt hij automatisch aangemaakt bij het genereren van de sleutels.

Om sleutel bestanden genereren en in *C:\sleuteldir* te plaatsen, typ:

rsacrypt.bat -k C:\sleuteldir (en druk Enter)

Om het tekstbestand *C:\geheim.txt* te coderen met de publieke sleutel in *C:\sleuteldir*, typ:

rsacrypt.bat -e C:\sleuteldir C:\geheim.txt (en druk Enter)

Om het gecodeerde tekstbestand *C:\geheim.txt.enc* te decoderen met de sleutels in *C:\sleuteldir*, typ:

rsacrypt.bat -d C:\sleuteldir C:\geheim.txt.enc (en druk Enter)

Literatuurlijst

Boeken

- Swarttouw, R. en Pik, D., *dictaat: Cryptografie*, Vrije Universiteit Amsterdam, Amsterdam, 2004, 58 blz.
- Briggs, M.E., *dictaat: An Introduction to the General Number Field Sieve*, Faculty of the Virginia Polytechnic Institute and State University, 1998, 77 blz.

Internet

<http://www.rsasecurity.com/>

<http://www.rsasecurity.com/rsalabs>

<http://www.tammo80.nl/java/vigenere.php>

<http://www.cryptonet.tk/>

Andere

Masterclass cryptografie aan de Vrij Universiteit (VU) van Amsterdam