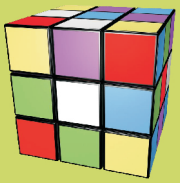
KATHOLIEKE UNIVERSITEIT
LEUVEN
CAMPUS KORTRIJK

Junior College 2010 – 2011

van priemgetal tot digitale handtekening



K.U.Leuven Campus Kortrijk
Wetenschap & Technologie

Fabien De Cruyenaere
Paul Igodt
Stijn Rebry

Proof by Poem

The RSA Encryption Algorithm

Take two large prime numbers, q and p.
Find the product n, and the totient φ.
If e and φ have GCD one
and d is e's inverse, then you're done!
For sending m raised to the e
reduced mod n gives secre-c.

Daniel G. Treat, NSA

Inhoudsopgave

Voorwoord

Wiskunde heeft veel gezichten en het Junior College wil enkele kanten belichten waarmee je misschien nog maar weinig kennisgemaakt hebt: wiskunde en wiskundige ideeën zitten namelijk verborgen in heel veel technologische toepassingen die het grote publiek als vanzelfsprekend vindt.

Het Junior College behandelt dit jaar de wiskunde achter Google en de wiskunde achter een modern cryptografiesysteem. Maar wiskunde speelt een even cruciale rol in de werking van je GSM, bij plaatsbepaling met GPS, in medische beeldverwerking, bij weersvoorspellingen, bij het al of niet succesvol voorspellen van beurskoersen, enzovoort. Telkens is er een achterliggend wiskundig probleem dat op het eerste zicht weinig met de concrete toepassing te maken heeft. Bij Google zijn dit vragen over matrices en eigenvectoren; bij cryptografie (RSA) zijn dit problemen over deelbaarheid, priemgetallen en modulorekenen.

De *unreasonable effectiveness of mathematics*, zoals Nobelprijswinnaar Eugene Wigner het noemde, is voor een groot deel te danken aan het universele en 'onbetwistbare' karakter van de wiskunde. Je zal daarom in het Junior College ook kennismaken met het concept bewijs: het is niet voldoende om met een aantal experimenten aan te tonen dat Google en RSA werken: we zullen aantonen waarom de methodes *altijd* tot de goede resultaten leiden. We zullen niet alleen leren hoe het RSA systeem informatie kan beschermen of de identiteit van de auteur kan verzekeren, maar ook begrijpen waarom dit steeds kan werken en welke uitdagingen er zijn.

In het kader van een groepswerk hebben eerste bachelor wiskunde studenten van K.U.Leuven Campus Kortrijk een verzorgde bijdrage geleverd tot de RSA-bundel. Dit deden ze door nagenoeg alle oefeningen zelf op te lossen, en door het leeuwendeel van het werk voor de kennismaking met het software-pakket `Maxima` en het gebruik ervan in het kader van deze lessenreeks, voor hun rekening nemen. Onze oprechte dank gaat hiervoor naar dit studententeam, met name Fien Aelter, Simon Decuypere, Thiebout Delabie, Heide Goethals, Tessa Seys en Sam Tertooy.

We willen jullie in dit Junior College tonen dat wiskunde een levende wetenschap is. Er zijn in de wiskunde veel open problemen, waarvan de oplossing een grote impact zou hebben op de wetenschap en de technologie en met enkele hiervan zal je tijdens de kick-off meeting van het Junior College kunnen kennismaken.

Van harte welkom, goede moed en veel succes, en laat van je horen door mee te werken en mee te denken.

Academiejaar 2010-2011.

Hoofdstuk 1

Basisprincipes van geheimcodering

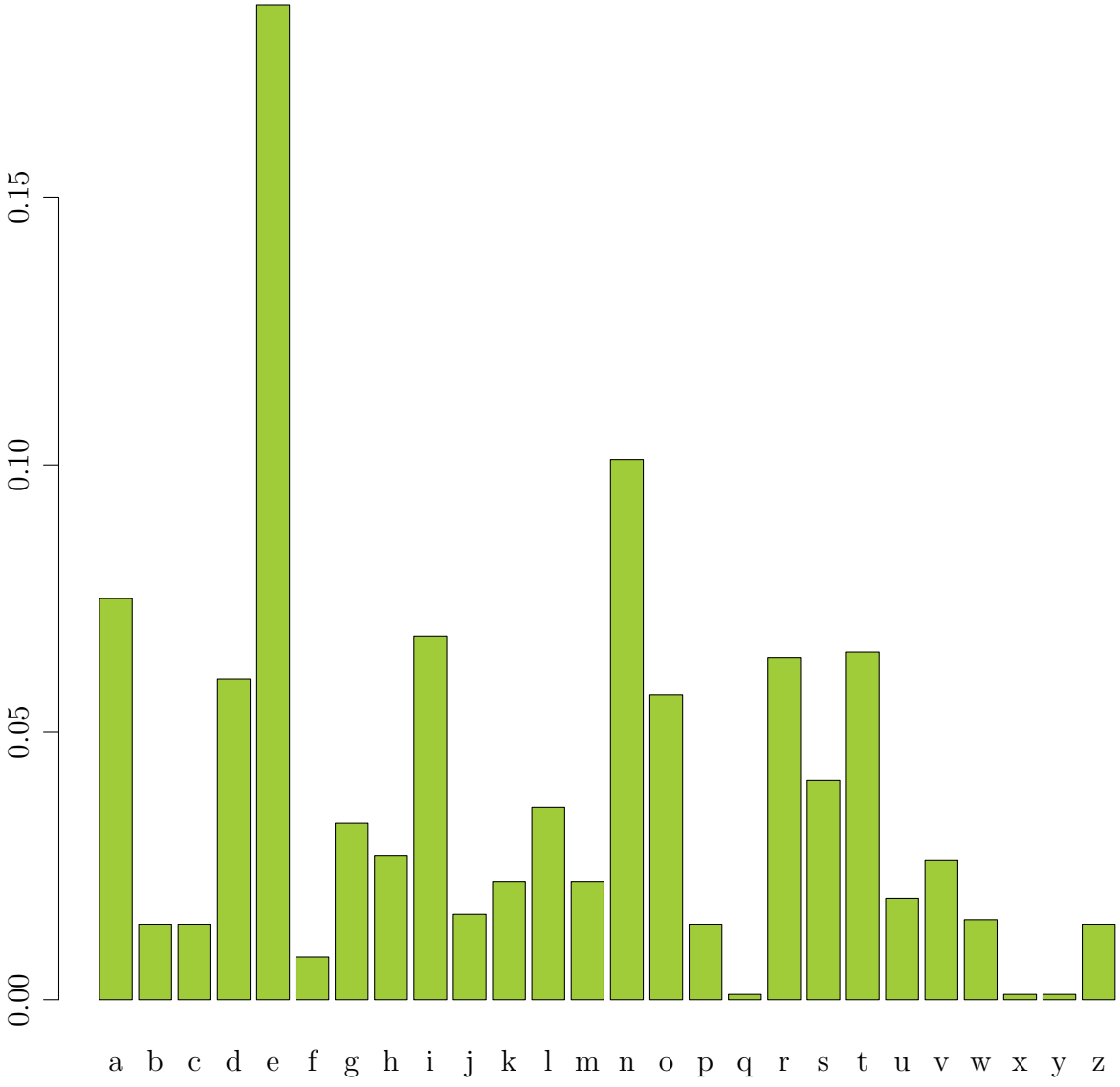
1. Klassieke geheimcodering

Doorheen de geschiedenis hebben mensen geheimen gehad, boodschappen die men wel met sommigen wenst uit te wisselen, maar absoluut niet algemeen mogen uitlekken. Hierbij kan men in eerste plaats aan militaire geheimen denken, strategieën, troepenlocaties, maar er zijn veel andere en meer vreedzame toepassingen te bedenken: bedrijfsgeheimen, bouwplannen, recepten en zo voort. Vandaag de dag komen daarbij nog alle persoonlijke gegevens die we tot onze privacy rekenen en die bescherming verdienen. Voeg hierbij nog de vele nieuwe communicatievormen, zoals elektronisch mail- en betaalverkeer, en het is duidelijk dat informatie beschermen een gerechtvaardigde wens is. E-mail en andere vormen van elektronische communicatie reizen typisch de halve wereld rond, langs tientallen computers, routers en knooppunten, waar ze onderschept en geobserveerd kunnen worden.

Een eerste manier om boodschappen ongezien bij de bestemmeling te brengen is **steganografie**, het fysiek verbergen van de boodschap. Het bekendste voorbeeld daarvan is onzichtbare inkt, maar in de geschiedenis werden een aantal markante methoden beproefd: boodschappen geschreven onder de laag was van een schrijftablet of op de kaalgeschoren hoofdhuid van de boodschapper. Steganografie is slechts veilig zolang de gebruikte methode geheim blijft en dus zeker niet geschikt voor toepassing op grote schaal.

Een tweede vorm van geheimschrift is het gebruik van een **monoalfabetisch substitutiecijfer**, waarbij elke letter uit het **klaarschrift**, het originele bericht, wordt vervangen door een vaste andere letter, het **geheimschrift**. De eenvoudigste vorm van substitutieversleuteling is een **rotatiecijfer**, waarbij elke letter vervangen wordt door de letter die een vast aantal posities verder in het alfabet staat. Zo verschoof Julius Caesar in zijn militaire communicatie elke letter drie posities in het alfabet. De **omzettingstabellen** van twee rotatiecijfers zijn te vinden in tabel 1.1.

Eens het gebruik van een rotatiecijfer algemeen bekend wordt, is het natuurlijk helemaal niet meer zo veilig, aangezien er maar een beperkt aantal mogelijke verschuivingen zijn. Worden echter willekeurige **permutaties** van het alfabet gebruikt als substitutiesleutel, dan zijn er zo veel mogelijkheden dat het onmogelijk lijkt deze code te kraken. Maar ook dat is niet helemaal waar. In een tekst komt elke afzonderlijke letter immers niet even vaak voor, waardoor een versleutelde tekst met wat zoek- en giswerk kan worden ontcijferd door de **frequenties** van de verschillende symbolen in de geheimtekst te vergelijken met de frequenties van de letters in een gemiddelde Nederlandstalige tekst (Figuur 1.1). Eerst kunnen de meest voorkomende letters als e en n worden geïdentificeerd, waarna alomtegenwoordige lidwoorden en werkwoordsvormen kunnen worden afgelezen en de overblijvende letters makkelijk kunnen worden ingevuld.



Figuur 1.1: Frequenties van letters in Nederlandstalige teksten

Tabel 1.1: Twee rotatiecijfers

Caesarcodering		`ROT13`	
klaar-schrift	geheim-schrift	klaar-schrift	geheim-schrift
a	d	a	n
b	e	b	o
c	f	c	p
⋮	⋮	⋮	⋮
x	a	x	k
y	b	y	l
z	c	z	m

Het grote probleem met deze methode is blijkbaar dat elke letter telkens op dezelfde manier wordt gecodeerd – vandaar de term monoalfabetisch. Om dit te omzeilen zijn tal van verbeteringen bij het substitutiesysteem bedacht, zoals het gebruik van verschillende versleutelingsalfabetten (**polyalfabetisch substitutiecijfer**), het vervangen van meerdere letters tegelijk (**polygrafisch substitutiecijfer**) of het vervangen van veelvoorkomende letters zoals e of n door verschillende symbolen (**homofoon substitutiecijfer**) zodat de frequentie van alle symbolen in de geheimtekst ongeveer gelijk is. Hoe ingewikkeld de bedachte varianten op het substitutiesysteem ook zijn, de geschiedenis bewijst dat ze toch te kraken zijn door een combinatie van inzicht, frequentie-analyse, rekenkracht, spionage of verraad.

1 Hoeveel monoalfabetische substitutiecijfers zijn er? Hoeveel substitutiecijfers laten geen enkele letter onveranderd? Hoeveel daarvan zijn rotatiecijfers?

2 Wat maakt `ROT13` zo bijzonder binnen de verzameling van de rotatiecijfers?

3 Ontcijfer volgende teksten. Wat voor soort encryptie is er gebruikt?

- ln nzdp, ln cdj, ln ryhuzrq
- rzo mpdelle, hlye op htdvfyop td nzydtdepye
- rtmt ztalz wtcqz ttf ugtrwtvqqkr utitod, dqqk dtz ttf wttzpt mgtavtka ol itz mtatk dgutsopa gd rt egrt zt gfzeopytktf

2. Publieke sleutelcodering

De hierboven beschreven klassieke encryptiesystemen zijn in wezen symmetrisch. Beide partijen spreken op voorhand een systeem af, een omzettingsmechanisme om klaartekst in geheimtekst om te zetten, en omgekeerd. Bijgevolg kan al wie het codeersysteem kent, teksten zowel coderen als decoderen want de omzettingstabel werkt in twee richtingen, coderen van links naar rechts en decoderen van rechts naar links.

Een dergelijk encryptiesysteem kan dus niet publiek zijn en moet zelf geheim gehouden worden, waardoor het niet geschikt is voor communicatie met onbekenden. Nochtans is het vandaag de

dag nodig om op een veilige manier gegevens uit te wisselen met partijen die men nog nooit heeft ontmoet: een rekening openen bij een internetbank, een nieuw mailadres aanmaken, een nieuw wachtwoord instellen, of een account openen in een webwinkel. Bij dergelijke transacties verloopt de communicatie bovendien via het internet, langs potentieel onveilige computers, waardoor men er van uit moet gaan dat al wat wordt verstuurd ook kan worden meegelezen door derden. Het is met andere woorden erg onverstandig om bijvoorbeeld een omzettingstabel voor een substitutiecijfer uit te wisselen.

Om al deze redenen moet een volledig nieuw systeem worden ontwikkeld, dat liefst ook voldoet aan de volgende voorwaarden:

1. het systeem moet frequentie-analyse bemoeilijken, in het bijzonder mag het dus *geen* monoalfabetisch substitutiesysteem zijn;
2. de veiligheid moet gewaarborgd zijn, zelfs als alle uitgewisselde gegevens door derden kunnen worden meegelezen;
3. het systeem moet eenvoudig uitbreidbaar zijn, zodat de codes veilig blijven naarmate computers over meer rekenkracht beschikken;
4. de werking van het codeersysteem moet publiek bekend zijn, zodat willekeurige correspondenten het samen kunnen gebruiken, ook al hebben ze elkaar nooit eerder ontmoet.

Deze vereisten lijken op het eerste zicht misschien onverenigbaar, maar worden allemaal voldaan in het algoritme van Rivest, Shamir en Adleman (RSA) dat in de volgende hoofdstukken zal worden bestudeerd. Het grote verschil tussen dit soort encryptiesystemen en de klassieke substitutiemethodes zal **asymmetrie** zijn: het is niet meer mogelijk om uit het codeermechanisme eenvoudig het decodeermechanisme af te leiden.

In asymmetrische cryptografie beschikt elke correspondent over twee sleutels, een publieke en een private. Wie geheime boodschappen wil ontvangen, maakt daarvoor een **publieke sleutel** bekend, die iedereen kan terugvinden en gebruiken om een tekst te encrypteren. Deze publieke sleutel laat echter *niet* toe om de geheimtekst daarna opnieuw te decoderen, waardoor de zender zijn eigen boodschap eigenlijk niet meer kan lezen. De ontvanger gebruikt voor het decrypteren de tweede sleutel, de **private sleutel**, die hij uiteraard strikt geheim houdt.

In RSA zal de publieke sleutel een enorm groot getal n zijn, product van twee verschillende priemgetallen p en q, die zelf de private sleutel vormen. Het coderen van een boodschap zal essentieel neer komen op machtsverheffingen a^r met a de boodschap en $r < n$ een goed gekozen exponent. Om de geheimtekst te decoderen zal een tweede exponent $s < n$ vereist zijn, zó dat $(a^r)^s = a$. Zonder overmatig veel rekenwerk uit te voeren, is dit getal s enkel te vinden indien p en q gekend zijn.

3. Rekenen met binaire getallen

Om berekeningen als a^r te kunnen uitrekenen, moeten de letters en leestekens uit de klaartekst eerst worden omgezet naar getallen. Hiervoor bestaan internationaal afgesproken standaarden, zoals de uitgebreide ASCII-omzettingstabel waarvan een fragment te zien is in tabel 1.2. Elke

letter wordt hierin voorgesteld door een getal bestaande uit 8 bits. Er wordt expliciet gekozen voor de binaire schrijfwijze van getallen in plaats van de decimale. Enerzijds omdat berekeningen in een computer steeds binair gebeuren, maar anderzijds vooral omdat de machtsverheffing a^r voor grote getallen in binaire notatie veel makkelijker uit te rekenen is. Daarom wordt eerst nagegaan hoe getallen in binaire schrijfwijze kunnen worden omgezet.

Vooraleer de binaire schrijfwijze van een getal te bestuderen, is het nodig goed te begrijpen hoe decimale getallen precies gevormd worden. In het **tiendelig talstelsel** wordt een natuurlijk getal x eigenlijk voorgesteld als een reeks van machten van het grondtal 10. Met de notatie $(x_k x_{k-1} \dots x_2 x_1 x_0)_{10}$ wordt het getal

$$x = x_0 \cdot 10^0 + x_1 \cdot 10^1 + \cdots + x_k \cdot 10^k = \sum_{i=0}^{k} x_i \cdot 10^i \tag{1.1}$$

bedoeld. De notatie $(\ )_{10}$ benadrukt hier dat het om een getal in decimale schrijfwijze gaat, en dat alle cijfers x_i dus element zijn van de verzameling $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Voorbeeld 1.1. Concreet wordt het getal $x =$ vijfduizendvierhonderdeenendertig genoteerd als $(5431)_{10}$. De rest bij Euclidische deling van x door 10 is gelijk aan 1, wat wil zeggen dat

$$5431 = 543 \cdot 10 + 1.$$

Dit bepaalt dat 1 het eindcijfer in de decimale notatie is. Als vervolgens het quotiënt 543 opnieuw Euclidisch wordt gedeeld door 10 is de rest 3,

$$543 = 54 \cdot 10 + 3.$$

De voorlaatste decimaal is dus 3. Op zijn beurt is 54 gelijk aan 5 keer 10 plus 4,

$$54 = 5 \cdot 10 + 4.$$

Dit alles wordt samengevat in volgende tabel:

Euclidische deling			decimale notatie
5431	$=$	$543 \cdot 10 + 1$	$(\dots 1)_{10}$
543	$=$	$54 \cdot 10 + 3$	$(\dots 31)_{10}$
54	$=$	$5 \cdot 10 + 4$	$(\dots 431)_{10}$
5	$=$	$0 \cdot 10 + 5$	$(5431)_{10}$

Allemaal samen geeft dit de ontbinding conform vergelijking (1.1)

$$\begin{aligned}(5431)_{10} &= \Big(\big(5 \cdot 10 + 4\big) \cdot 10 + 3\Big) \cdot 10 + 1 \\ &= 5 \cdot 10^3 + 4 \cdot 10^2 + 3 \cdot 10^2 + 1 \cdot 10^0.\end{aligned}$$

Om over te schakelen op het **binaire talstelsel**, dit wil zeggen met grondtal 2, worden de coëfficiënten $b_i \in \{0, 1\}$ gezocht waarvoor

$$x = b_0 \cdot 2^0 + b_1 \cdot 2^1 + b_2 \cdot 2^2 + \cdots + b_\ell \cdot 2^\ell = \sum_{i=0}^{\ell} b_i \cdot 2^i. \tag{1.2}$$

Tabel 1.2: Fragment uit de uitgebreide ASCII-tekenset

decimaal	binair	tekst	decimaal	binair	tekst
97	01100001	a	110	01101110	n
98	01100010	b	111	01101111	o
99	01100011	c	112	01110000	p
100	01100100	d	113	01110001	q
101	01100101	e	114	01110010	r
102	01100110	f	115	01110011	s
103	01100111	g	116	01110100	t
104	01101000	h	117	01110101	u
105	01101001	i	118	01110110	v
106	01101010	j	119	01110111	w
107	01101011	k	120	01111000	x
108	01101100	l	121	01111001	y
109	01101101	m	122	01111010	z

De binaire schrijfwijze van het getal x is dan $(b_l b_{l-1} \dots b_2 b_1 b_0)_2$, waarbij de aanduiding $(\ )_2$ ditmaal benadrukt dat het om een binaire notatie gaat. Het moet duidelijk zijn het hier om twee verschillende voorstellingen voor hetzelfde getal gaat,

$$x = (x_k x_{k-1} \dots x_2 x_1 x_0)_{10} = (b_l b_{l-1} \dots b_2 b_1 b_0)_2.$$

Voorbeeld 1.2. Een procédé analoog aan dat in voorbeeld 1.1 geeft de coëfficiënten voor de binaire schrijfwijze van het getal $x =$ vijfduizendvierhonderdeenendertig,

Euclidische deling			binaire notatie
5431	$=$	$2715 \cdot 2 + 1$	$(\dots 1)_2$
2715	$=$	$1357 \cdot 2 + 1$	$(\dots 11)_2$
1357	$=$	$678 \cdot 2 + 1$	$(\dots 111)_2$
678	$=$	$339 \cdot 2 + 0$	$(\dots 0111)_2$
339	$=$	$169 \cdot 2 + 1$	$(\dots 10111)_2$
169	$=$	$84 \cdot 2 + 1$	$(\dots 110111)_2$
84	$=$	$42 \cdot 2 + 0$	$(\dots 0110111)_2$
42	$=$	$21 \cdot 2 + 0$	$(\dots 00110111)_2$
21	$=$	$10 \cdot 2 + 1$	$(\dots 100110111)_2$
10	$=$	$5 \cdot 2 + 0$	$(\dots 0100110111)_2$
5	$=$	$2 \cdot 2 + 1$	$(\dots 10100110111)_2$
2	$=$	$1 \cdot 2 + 0$	$(\dots 010100110111)_2$
1	$=$	$0 \cdot 2 + 1$	$(1010100110111)_2$

Of alles samen,

$$\begin{aligned}(5431)_{10} &= 1 \cdot 2^{12} + 0 \cdot 2^{11} + 1 \cdot 2^{10} + 0 \cdot 2^9 + 1 \cdot 2^8 + 0 \cdot 2^7 \\ &\quad + 0 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= (1010100110111)_2\end{aligned}$$

Om getallen van decimale naar binaire notatie om te zetten is er het programma `binexp` (zie tabel 1.3). Dit programma heeft een natuurlijk getal `K` als invoer en geeft de binaire expansie van `K` als uitvoer `BIN` terug.

Tabel 1.3: `binexp`: decimaal getal omzetten in binaire notatie

```
PROGRAM:BINEXP
  Prompt K
  1 → dim(⌊BINR)
  1 → dim(⌊BIN)
  If K = 0
  Then
    Disp "DE BINEXP.IS{0}"
  Else
    1 → I
    While iPart(K/2) ≠ 0
      K-2*iPart(K/2)→ ⌊BINR(I)
      iPart(K/2)→K
      I+1→I
    End
    1 → ⌊BINR(I)
    dim(⌊BINR) → D
    For(J, 0, D − 1)
    ⌊BINR(D − J) → ⌊BIN(J + 1)
  End
  Disp"DE BINEXP.IS",⌊BIN
```

4. Veilige encryptie met RSA

De volgende hoofdstukken bestuderen grondig de werking van RSA en tonen in detail onder welke voorwaarden dit cryptosysteem feilloos werkt. Om de gedachten te vestigen wordt hieronder nog conceptueel uiteengezet hoe een tekst door het RSA-algoritme geëncrypteerd wordt en waarom deze methode zo veilig zal zijn. Deze tekst maakt vanzelfsprekend een vereenvoudiging van de werkelijkheid en licht enkel het RSA-aspect uit een veel ruimer ICT-verhaal toe. Zo zal bijvoorbeeld RSA nooit rechtstreeks gebruikt worden om boodschappen te coderen, maar zal bijvoorbeeld de **Advanced Encryption Standard** als tussenstap gebruikt worden.

Zoals al gezegd wordt elk symbool uit de klaartekst omgezet in zijn (binaire) ASCII-code van 8 bits. Alle letters van de boodschap vormen zo een lange lijst van bits die worden beschouwd als binaire getallen a waarvan de grootte kleiner is dan de publieke sleutel n. Realistische waarden voor n liggen vandaag tussen de 1024 en 2048 bits, zodat één stuk binaire code a correspondeert met 1024/8=128 tot 2048/7=256 symbolen. Deze getallen a worden dan in hun geheel getransformeerd in a^r, waardoor een getal in de geheimtekst niet overeen komt met één letter maar met honderden symbolen. Hierdoor wordt frequentie-analyse vanzelfsprekend quasi onmogelijk.

De zwakheid van dit systeem is dat om het te kraken enkel de publieke sleutel n moet worden gefactoriseerd in $p \cdot q$. Er bestaan echter geen snelle methodes om de delers van een getal n te vinden. Essentieel moet daarvoor namelijk van alle getallen kleiner dan $\sqrt{n}$ worden nagegaan

of ze n delen. In de praktijk is dit nauwelijks haalbaar aangezien voor getallen met meer dan honderd decimalen, dit factoriseren met een gewone computer ettelijke jaren zou duren. De veiligheid is dus gewaarborgd zolang de priemgetallen p en q groot genoeg zijn.

Verder bestaan er oneindig veel priemgetallen, zodat de publieke sleutel n willekeurig groot kan worden. Als priemgetallen p en q zó worden gekozen dat de publieke sleutel n één decimaal meer telt, dan komen er meteen tien keer zoveel getallen in aanmerking als mogelijke delers van n. De rekentijd om n te factoriseren neemt dus blijkbaar exponentieel toe met het aantal decimalen van n. De complexiteit van het systeem is dus op een relatief eenvoudige manier op te drijven, zodat het kraken van de code buiten bereik van de rekenkracht van steeds snellere computers kan worden gehouden.

Omdat het ondoenbaar is uit de publieke sleutel n de private sleutel te berekenen, kan deze zonder gevaar publiek bekend worden gemaakt. Correspondenten die elkaar nooit hebben ontmoet, kunnen zo op een veilige manier boodschappen uitwisselen. Zelfs als derden de boodschap en de publieke sleutel onderscheppen, kunnen ze deze niet ontcijferen.

Hiermee blijkt dat het RSA-systeem aan alle vereisten voor een goed cryptosysteem zal voldoen: frequentie-analyse is onmogelijk, de complexiteit is onbeperkt uitbreidbaar, ook wie het systeem kent en een geheimtekst onderschept kan deze niet decoderen, zelfs niet als men over de publieke sleutel n beschikt. Er is echter nog een lange weg te gaan om van deze basisprincipes te komen tot een volledig werkend cryptosysteem.

5. Over bijecties en permutaties

Definitie 1.1. *Als A en B verzamelingen zijn, dan is*

$$A \times B = \{(a, b) \mid a \in A, b \in B\}$$

*de **productverzameling** van A met B. In het bijzondere geval dat $A = B$, wordt meestal de notatie A^2 gebruikt.*

*Elke deelverzameling $\mathcal{R}$ van $A \times B$ wordt een **relatie** van A naar B genoemd. Als $A = B$, spreken we kortweg van een relatie in A.*

In plaats van $(a, b) \in \mathcal{R}$, noteert men meestal $a\mathcal{R}b$. We lezen dan "a staat in relatie ($\mathcal{R}$) met b".

Bijvoorbeeld, als $A = \{a, b, c\}$ en $B = \{1, 2, 3, 4\}$, dan is $\mathcal{R} = \{(a, 2), (a, 3), (b, 4), (c, 3)\}$ een relatie van A naar B.

Voorbeeld 1.3. We geven enkele voorbeelden die allicht al bekend zijn, maar mogelijks niet als relatie werden opgemerkt.

1. $\{(x, y) \in \mathbb{Z}^2 \mid \ x < y\}$ is een relatie in $\mathbb{Z}$
2. $\{(x, y) \in \mathbb{Z}^2 \mid x \leq y\}$ is een relatie in $\mathbb{Z}$
3. $\{(x, y) \in \mathbb{N}^2 \mid x + y = \text{ even }\}$ is een relatie in $\mathbb{N}$

4. als $A = \{0, 1, 2, 3, 4, 5, 6\}$, dan is $\{(x, y) \in A^2 \mid x \text{ is deler van } y\}$ een relatie in A
5. $\mathcal{R} = \{(x, y) \in \mathbb{R}^2 \mid y = x^3\}$ is een relatie in $\mathbb{R}$
6. $\mathcal{R} = \{(x, y) \in \mathbb{R}^2 \mid x^2 + y^2 = 1\}$ is een relatie in $\mathbb{R}$
7. de relatie $\mathcal{R}$ in de verzameling van alle $m \times n$-matrices, gedefinieerd door "$A \ \mathcal{R} \ B$ als en slechts als er een eindig aantal elementaire rijbewerkingen bestaat die A omzetten in B"

We geven een kort overzicht van verschillende manieren om een relatie visueel voor te stellen.

Roosterdiagram (of **grafiek**): dit is de voorstellingswijze waar je het meest vertrouwd mee bent. Hierbij worden de elementen van A en B respectievelijk op de X-as en de Y-as voorgesteld: het roosterdiagram bestaat dan uit alle punten van het (X, Y)-vlak waarvan de coördinaat behoort tot de relatie. De grafiek van voorbeeld 1.3.3 hierboven is de verzameling van punten te zien in figuur 1.2 (a). De grafiek van voorbeeld 1.3.6 daarentegen is de cirkel in figuur 1.2 (b).

Gerichte graaf in het geval A en B eindige verzamelingen zijn: hierbij worden de elementen van A en B voorgesteld door stippen en telkens als $a \ \mathcal{R} \ b$, wordt een pijl getekend van a naar b.

Binaire matrix als A en B eindige, geordende verzamelingen zijn: als $A = \{a_1, a_2, \ldots, a_n\}$ en $B = \{b_1, b_2, \ldots, b_m\}$, dan is de binaire matrix $M = (m_{i,j})$ van een relatie $\mathcal{R}$ van A naar B een $n \times m$-matrix, waarbij

$$\begin{cases} m_{i,j} = 1, & \text{als } (a_i, b_j) \in \mathcal{R} \\ m_{i,j} = 0, & \text{als } (a_i, b_j) \notin \mathcal{R} \end{cases}$$

In voorbeeld 1.3.4 is

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

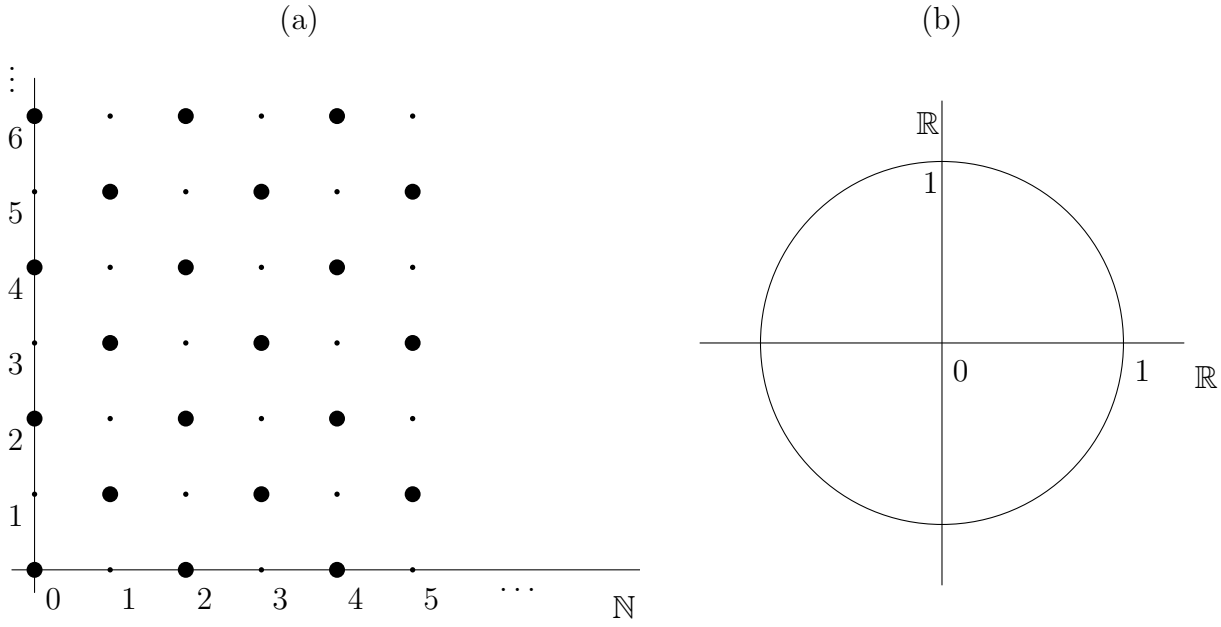
4 Maak een gerichte graaf voor de relatie in voorbeeld 1.3.4.

Definitie 1.2. *Als $\mathcal{R}$ een relatie is van A naar B, dan is de* ***omgekeerde relatie*** *$\mathcal{R}^{-1}$ een relatie van B naar A zodat $(a, b) \in \mathcal{R} \Leftrightarrow (b, a) \in \mathcal{R}^{-1}$.*

5 Hoe verkrijg je de graaf van $\mathcal{R}^{-1}$ uit de graaf van $\mathcal{R}$? Wat gebeurt er met de binaire matrix?

6 Hoe ontstaat het voorschrift van $\mathcal{R}^{-1}$ uit dat van $\mathcal{R}$?

Figuur 1.2: Grafieken van twee verschillende relaties



Definitie 1.3. *Een* ***functie*** f *van* A *naar* B *is een relatie van* A *naar* B *waarbij voor elk element* x *van* A *hoogstens één element* y *van* B *bestaat zodat* $(x, y) \in f$.

We noteren $y = f(x)$ *en noemen* y *het* ***beeld*** *van* x *onder* f. *We schrijven typisch* $f : A \to B : x \mapsto f(x) = y$.

Definitie 1.4. *De verzameling van alle elementen van* A *die een (één) beeld hebben onder* f *noemen we het* ***domein*** *van* f *en noteren we met* $\mathrm{dom}(f)$.

De verzameling van alle elementen van B *die beeld zijn van één of meerdere elementen van* A *onder* f *noemen we de* ***beeldenverzameling*** *van* f *of kortweg het beeld van* f *en noteren we met* $\mathrm{im}(f)$.

In de Analyse komen functies van $\mathbb{R}$ naar $\mathbb{R}$ (reële functies of functies in $\mathbb{R}$) overvloedig aan bod en worden verschillende klassen van functies zoals veeltermfuncties, rationale functies, irrationale functies, goniometrische en cyclometrische functies, exponentiële en logaritmische functies en dergelijke bestudeerd. We geven enkele voorbeelden

Voorbeeld 1.4.

1. $f_1 : \mathbb{R} \to \mathbb{R} : x \mapsto x^2 = y$
2. $f_2 : \mathbb{R} \to \mathbb{R} : x \mapsto x^3 = y$
3. $f_3 : \mathbb{R} \to \mathbb{R} : x \mapsto \sqrt{x} = y$
4. $f_4 : \mathbb{R} \to \mathbb{R} : x \mapsto \sqrt[3]{x} = y$
5. $f_5 : \mathbb{R} \to \mathbb{R} : x \mapsto \frac{1}{x} = y$

6. $f_6 : \mathbb{R} \to \mathbb{R} : x \mapsto \sin x = y$

7. $f_7 : \mathbb{R} \to \mathbb{R} : x \mapsto 2^x = y$

7 Zoek van elk van deze reële functies domein en beeldenverzameling.

Wij zullen in deze bundel bijna uitsluitend werken met functies in een eindige verzameling en zijn vooral geïnteresseerd in volgende kenmerken van functies.

Definitie 1.5.

- *Een functie f van A naar B is een **afbeelding** van A* in *B als en slechts als elk element van A een (één) beeld heeft onder f in B, of, equivalent, als* $\mathrm{dom}(f) = A$. *Indien $A = B$, dan spreken we van een **transformatie** van A.*
- *Een functie f van A naar B is **injectief** van A* in *B als en slechts als elke twee verschillende elementen uit het domein van A verschillende beelden hebben onder f in B, of equivalent als voor elk element b van B hoogstens één element a van A bestaat zodat het beeld van a onder f gelijk is aan b ;*
- *Een functie f van A naar B is **surjectief** van A* op *B als en slechts als voor elk element b van B minstens één element a van A bestaat zodat het beeld van a onder f gelijk is aan b, of, equivalent, als* $\mathrm{im}(f) = B$;
- *Een functie f van A naar B is **bijectief** van A* op *B als en slechts als f zowel surjectief als injectief van A op B is.*
- *Een functie f van A naar B is een **bijectie** als en slechts als f een injectieve en surjectieve afbeelding van A in B is. Indien $A = B$, dan spreken we van een **permutatie** van A. Analoog zijn een **injectie** en **surjectie** afbeeldingen die injectief respectievelijk surjectief zijn.*

8 Formuleer bovenstaande definities in symbolen en teken telkens een graaf die de eigenschap illustreert.

9 Ga na welke van deze kenmerken voldaan zijn voor de functies f_1 tot f_7 uit voorbeeld 1.4. Indien een functie niet injectief is, vind dan een maximale injectieve beperking ervan. Beperk met andere woorden het domein tot een zo groot mogelijke deelverzameling zodat de functie wel injectief wordt.

10 Teken de graaf van een bijectie van A op B waarbij A en B eindige verzamelingen zijn; wat kun je hieruit besluiten? Formuleer je besluit als een eigenschap.

11 Zoek een bijectie van $\mathbb{N}$ op $\mathbb{Z}$.

Het belang van bijecties, inzonder injectieve functies, blijkt uit het volgende. Aangezien een functie f van A naar B in de eerste plaats een relatie is van A naar B en voor elke relatie een omgekeerde relatie bestaat, is volgende vraag heel logisch.

Wanneer is de omgekeerde relatie f^{-1} opnieuw een functie (van B naar A)?

12 Teken de graaf van een functie waarvan de omgekeerde relatie *niet* opnieuw een functie is. Wat kun je hieruit besluiten?

We verkrijgen aldus

Stelling 1.6. *Zij f een functie van A naar B. Dan is f^{-1} een functie van B naar A als en slechts als f injectief is van A in B. In dit geval noemen we f^{-1} de* ***inverse functie*** *van f.*

In symbolen: $f^{-1} : B \to A : y \mapsto f^{-1}(y) = x \Leftrightarrow f(x) = y$.

13 Zoek voor alle injectieve functies of maximale injectieve beperkingen uit voorbeeld 1.4 het voorschrift van de inverse functie en teken van beide de grafiek ten opzichte van een orthonormaal assenstelsel in het vlak. Wat constateer je?

14 Bewijs dat f is een injectieve afbeelding is van A in B als en slechts als er een functie g van B naar A bestaat zodat $g \circ f$ gelijk is aan Id_A, de identieke afbeelding van A. Als bovendien $A = B$ een eindige verzameling is, dan is f een permutatie van A en dan is g de inverse functie van f.

Hoofdstuk 2

Delers en deelbaarheid

In het vorige hoofdstuk kwam al aan bod dat priemgetallen en deelbaarheid een belangrijke rol zullen spelen bij het versleutelen van gegevens en het maken van een digitale handtekening. Daarom worden in deze les een aantal essentiële zaken over deelbaarheid geïntroduceerd.

1. Deelbaarheid

Iedereen heeft wel een idee van wat deelbaarheid is. Nochtans kan er zonder strikte definitie bijvoorbeeld eindeloos gediscussieerd worden over het feit of nul zichzelf deelt, of niet. Het is dus absoluut niet overbodig om ondubbelzinnig in een definitie vast te leggen wat deelbaarheid nu precies is. Intuïtie is immers *niet* voldoende om een geldig argument te maken:

Definitie 2.1. *Gegeven twee gehele getallen, m en n, dan zeggen we dat m een **deler** is van n (we noteren dan $m \mid n$) als en slechts als er een geheel getal q bestaat zo dat $q \cdot m = n$. In symbolen:*

$$\forall m, n \in \mathbb{Z} : \quad m \mid n \Leftrightarrow (\exists q \in \mathbb{Z} : \quad m \cdot q = n).$$

De notatie $\text{del}(n)$ staat voor de **verzameling van de delers** van n. Het getal 0 is het enige geheel getal met een oneindige verzameling delers, $\text{del}(0) = \mathbb{Z}$. Voor elk geheel getal n, noemen we 1, -1, n en $-n$ de **onechte delers** van n. Alle eventuele andere delers noemen we **echte delers**. Een geheel getal n verschillend van 1 en -1 heeft dus minstens 4 delers.

Stelling 2.2. *Voor alle gehele getallen n gelden volgende uitspraken*

1. *Elk geheel getal n heeft zichzelf en $-n$ als delers,*

$$\forall n \in \mathbb{Z} : \quad (n \mid n \ \wedge \ (-n) \mid n).$$

2. *Als d een deler is van het geheel getal n, dan is ook $-d$ een deler van n. De getallen d en $-d$ worden **geassocieerde delers** genoemd. In symbolen,*

$$\forall n, d \in \mathbb{Z} : \quad (d \mid n) \Leftrightarrow ((-d) \mid n).$$

3. *De getallen 1 en -1 zijn deler van elk geheel getal n.*

4. *Elk geheel getal n is deler van 0.*

Bewijs. Hieronder het bewijs van 1. en 2. als voorbeeld.

1. Inderdaad bestaat er voor elk geheel getal n een getal $q \in \mathbb{Z}$ zodat $q \cdot n = n$, namelijk $q = 1$. Analoog geldt voor $q = -1$ dat $q \cdot (-n) = n$, dus is ook $-n$ een deler van n.

2. Als d een deler is van n bestaat er per definitie een getal q in $\mathbb{Z}$ waarvoor $d \cdot q = n$. Dan geldt voor het getal $-q$ dat $(-d) \cdot (-q) = d \cdot q = n$ en dus is ook $-d$ een deler van n. □

15 Schrijf uitspraken 3. en 4. uit stelling 2.2 zelf in symbolen en bewijs.

Stelling 2.3. *Als een geheel getal d deler is van de gehele getallen a en b, dan is d ook een deler van het* ***product*** *$a \cdot b$ en van elke* ***lineaire combinatie*** *$k \cdot a + l \cdot b$ met $k, l \in \mathbb{Z}$.*

Dit resultaat zal in het vervolg vaak gebruikt worden, vooral in het geval dat $k = l = 1$ of $l = 0$.

16 Schrijf stelling 2.3 zelf in symbolen en bewijs.

2. De Euclidische deling

Wat iedereen al uit het basisonderwijs kent, is de deling van een natuurlijk getal m, het **deeltal**, door een van nul verschillend natuurlijk getal n, de **deler**. Dit resulteert in een **quotiënt** $q \in \mathbb{N}$ en een positieve **rest** r die hoogstens gelijk is aan $n - 1$. Het berekenen van quotiënt en rest kan met behulp van een staartdeling.

Volgende stelling veralgemeent dit voor gehele getallen en toont dat er voor elke deler en deeltal slechts één quotiënt en positieve rest bestaan.

Stelling 2.4 (**Stelling van de Euclidische deling**).
Voor elke twee gehele getallen m en n met $n \neq 0$ bestaan unieke getallen q en r in $\mathbb{Z}$ zó dat $m = q \cdot n + r$ terwijl $0 \leq r < |n|$, of in symbolen,

$$\forall m \in \mathbb{Z} \ \forall n \in \mathbb{Z}_0 \ \exists! q \in \mathbb{Z} \ \exists! r \in \mathbb{N} \ : \quad m = q \cdot n + r \quad \wedge \quad r < |n|.$$

Bewijs. Beschouw de verzameling

$$A = \{m - \lambda \cdot n \ \mid \ \lambda \in \mathbb{Z}\}.$$

Het is duidelijk dat $A \cap \mathbb{N} \neq \emptyset$. Bijgevolg heeft $A \cap \mathbb{N}$ een (uniek) kleinste element. Noem dit element r en zij $r = m - q \cdot n$ voor een unieke $q \in \mathbb{Z}$.

Uit het **ongerijmde** is te bewijzen dat $r < |n|$. Stel dat $r \geq |n|$. Dan zou enerzijds $r - |n| \in \mathbb{N}$ en anderzijds

$$\begin{aligned} r - |n| &= r \pm n \\ &= (m - q \cdot n) \pm n \\ &= m - (q \pm 1) \cdot n \in A, \end{aligned}$$

dus ook $r - |n| \in A$. Maar dit is in tegenspraak met het feit dat r het kleinste element is van $A \cap \mathbb{N}$. □

17 Wat is de verzameling A uit het bewijs voor $m = -18$ en $n = 15$? Wat is r, het minimum van $A \cap \mathbb{N}$ in dat geval?

18 Bepaal het quotiënt q en de rest r bij Euclidische deling van

1. $m = 48$ door $n = 14$;
2. $m = -34$ door $n = 5$;
3. $m = 27$ door $n = -8$;
4. $m = -64$ door $n = -8$.

3. Grootste gemene deler en kleinste gemeen veelvoud

Definitie 2.5. *De* ***grootste gemene deler*** $d = \text{ggd}(m, n)$ *van een koppel gehele getallen* m *en* n *is een gemeenschappelijke deler* $d \in \mathbb{N}$ *van* m *en* n *zó dat elke gemeenschappelijke deler van* m *en* n *ook deler is van* d.

Definitie 2.6. *Het* ***kleinste gemeen veelvoud*** $k = \text{kgv}(m, n)$ *van een koppel gehele getallen* m *en* n *is een gemeenschappelijk veelvoud* $k \in \mathbb{N}$ *van* m *en* n *zó dat* k *een deler is van elk gemeenschappelijk veelvoud van* m *en* n.

19 Formuleer beide definities in symbolen.

Definitie 2.7. *Twee getallen* m *en* n *worden* ***relatief priem*** *of* ***onderling ondeelbaar*** *genoemd als en slechts als* $\text{ggd}(m, n) = 1$.

20 Vul aan en verantwoord: voor elk geheel getal $n \in \mathbb{Z}_0$ geldt:

1. $\text{ggd}(0, n) = \ldots$
2. $\text{kgv}(0, n) = \ldots$
3. $\text{ggd}(1, n) = \ldots$
4. $\text{kgv}(1, n) = \ldots$
5. $\text{ggd}(n, n) = \ldots$
6. $\text{kgv}(n, n) = \ldots$
7. $\text{ggd}(0, 0) = \ldots$
8. $\text{kgv}(0, 0) = \ldots$

21 Zoek de grootste gemene deler van 357 en 231 als volgt. Bekijk de verzameling van de delers van beide getallen. Enerzijds

$$\text{del}(357) = \ldots$$

en anderzijds

$$\text{del}(231) = \ldots$$

De gemeenschappelijke delers zijn dan elementen van

$$\mathcal{D} = \text{del}(357) \cap \text{del}(231) = \ldots$$

Bijgevolg geldt $\text{ggd}(357, 231) = \ldots$.

4. Het algoritme van Euclides

In de vorige oefeningen werd de grootste gemene deler van twee getallen gezocht door expliciet de verzamelingen van delers op te stellen, wat eigenlijk een zeer rekenintensieve methode is. Bovendien zal het van belang blijken niet alleen de grootste gemene deler te vinden, maar ook de unieke getallen a en b te kennen waarvoor

$$\text{ggd}(m,n) = a \cdot m + b \cdot n.$$

In het geval dat m of n nul zijn of gelijk aan elkaar, werd de grootste gemene deler al gezocht in een voorgaande oefening. In het geval dat beide getallen verschillend zijn van elkaar en van nul kan met behulp van de stelling van de Euclidische deling een lijst instructies worden opgesteld die toelaat om de grootste gemene deler stap voor stap te berekenen, samen met de factoren a en b : het zogenaamde algoritme van Euclides.

Stelling 2.8 (**Algoritme van Euclides**).

Initialisatie. *Stel r_{-1} gelijk aan het getal uit $\{m,n\}$ met de grootste absolute waarde en r_0 aan het getal met de kleinste absolute waarde. Stel $i = 1$.*

Iteratiestap. *Bereken voor $i \geq 1$ het quotiënt q_i en de rest $0 \leq r_i < |r_{i-1}|$ volgens stelling 2.4 bij deling van r_{i-2} door r_{i-1},*

$$r_{i-2} = q_i \cdot r_{i-1} + r_i. \tag{2.1}$$

Stopcriterium. *Is r_i gelijk aan nul, dan is de grootste gemene deler gelijk aan r_{i-1} en stop. Is r_i niet gelijk aan nul, herhaal dan de iteratiestap voor $i+1$ en ga opnieuw het stopcriterium na.*

Hieronder wordt het concrete verloop van dit algoritme geschetst voor het geval dat $|m| > |n|$. In elke stap wordt de vergelijking (2.1) opgelost naar r_i in functie van m en n. Hieruit zullen later (Stelling 2.9) formules voor a en b kunnen worden afgeleid.

Stap 1. Deel $r_{-1} = m$ door $r_0 = n$. Je vindt zo een uniek quotiënt q_1 en een unieke positieve rest r_1 met $0 \leq r_1 < |n|$ zo dat

$$\begin{aligned} m = q_1 \cdot n + r_1 \quad \Rightarrow \quad r_1 &= 1 \cdot m - q_1 \cdot n \\ &= a_1 \cdot m + b_1 \cdot n. \end{aligned}$$

Stap 2. Deel vervolgens $r_0 = n$ door r_1. Je vindt een unieke q_2 en r_2, met $0 \leq r_2 < r_1$ waarvoor

$$\begin{aligned} n = q_2 \cdot r_1 + r_2 \quad \Rightarrow \quad r_2 &= 1 \cdot n - q_2 \cdot r_1 \\ &= 1 \cdot n - q_2 \cdot (m - q_1 \cdot n) \\ &= -q_2 \cdot m + (1 + q_2 \cdot q_1) \cdot n \\ &= a_2 \cdot m + b_2 \cdot n. \end{aligned}$$

Volgende stappen. Herhaal deze werkwijze door steeds opnieuw de deling uit te voeren met de twee laatst gevonden resten. Deel r_1 door r_2, r_2 door r_3 en zo voort. Merk op dat je telkens de nieuw verkregen rest kan schrijven als lineaire combinatie van m en n.

$$\begin{array}{lll} r_1 = q_3 \cdot r_2 + r_3 & \Rightarrow & r_3 = a_3 \cdot m + b_3 \cdot n, \\ r_2 = q_4 \cdot r_3 + r_4 & \Rightarrow & r_4 = a_4 \cdot m + b_4 \cdot n, \\ & \vdots & \\ r_{i-2} = q_i \cdot r_{i-1} + r_i & \Rightarrow & r_i = a_i \cdot m + b_i \cdot n. \end{array}$$

Laatste stap. Na een aantal stappen, zeg k, zou de deling dus voor het eerst opgaan en rest $r_{k+1} = 0$ opleveren,

$$r_{k-2} = q_k \cdot r_{k-1} + r_k, \tag{2.2}$$
$$r_{k-1} = q_{k+1} \cdot r_k. \tag{2.3}$$

Waarbij r_k nog steeds kan geschreven worden als een lineaire combinatie

$$r_k = a_k \cdot m + b_k \cdot n. \tag{2.4}$$

Het is nog altijd niet duidelijk dat dit algoritme inderdaad de grootste gemene deler als resultaat heeft. Dat vereist een **correctheidsbewijs**.

22 *Bewijs als oefening.* Het correctheidsbewijs is een oefening voor de lezer: toon eerst aan dat het algoritme wel degelijk stopt, vervolgens dat de laatste van nul verschillende rest een gemeenschappelijke deler is van m en n, tot slot dat dit de grootste gemene deler is. Formuleer een volledig en coherent bewijs op basis van volgende vragen.

1. Kijk naar de rest r_i om te tonen dat het algoritme stopt. Wat zijn de mogelijke waarden voor r_1? Tussen welke grenzen ligt r_i? Hoeveel stappen zal het algoritme dus maximaal zetten?
2. Toon aan dat r_k een deler is van r_{k-1} door vergelijking (2.3) in (2.2) te substitueren. Veralgemeen dit argument om te tonen dat r_k ook m en n deelt.
3. Welke eigenschap van deelbaarheid toont dat elke gemeenschappelijke deler d van m en n ook r_{k-1} deelt?

Om ten slotte ook de coëfficiënten a en b te vinden waarvoor

$$\text{ggd}(m, n) = a \cdot m + b \cdot n,$$

is er het uitgebreide algoritme van Euclides.

Stelling 2.9 (**Uitgebreide algoritme van Euclides**).

Initialisatie. *Neem aan dat* $|m| > |n|$. *Stel dan bijkomend bij het Algoritme van Euclides volgende waarden in*

$$\left\{ \begin{array}{lcl} a_{-1} & = & 1 \\ b_{-1} & = & 0 \end{array} \right. \quad \textit{en} \quad \left\{ \begin{array}{lcl} a_0 & = & 0 \\ b_0 & = & 1. \end{array} \right. \tag{2.5}$$

Iteratiestap. *Bereken voor $i \geq 1$ naast het quotiënt q_i en de rest $0 \leq r_i < |r_{i-1}|$ ook de coëfficiënten a_i en b_i als volgt*

$$\begin{cases} a_i &= a_{i-2} - q_i \cdot a_{i-1} \\ b_i &= b_{i-2} - q_i \cdot b_{i-1}. \end{cases} \tag{2.6}$$

Stopcriterium. *Is r_i gelijk aan nul, dan is de grootste gemene deler gelijk aan r_{i-1} en bovendien*

$$\text{ggd}(m,n) = r_{i-1} = a_{i-1} \cdot m + b_{i-1} \cdot n.$$

Indien niet, wordt de iteratiestap herhaald voor $i+1$.

Bewijs. Aangezien $|m| > |n|$ gelden de identiteiten $r_{-1} = m$ en $r_0 = n$ zodat de initialitaties $r_{-1} = m = 1 \cdot m + 0 \cdot n = 1 \cdot r_{-1} + 0 \cdot r_0$ en $r_0 = n = 0 \cdot m + 1 \cdot n = 0 \cdot r_{-1} + 1 \cdot r_0$ inderdaad aanleiding geven tot de vergelijkingen 2.5.

Stel als inductiehypothese dat de vergelijkingen $r_i = a_i \cdot m + b_i \cdot n$ geldig zijn voor $i \leq j$. Dan geldt voor $j+1$ dat

$$\begin{aligned} r_{j+1} &= r_{j-1} - q_{j-1} \cdot r_j \\ &= \big(a_{j-1} \cdot m + b_{j-1} \cdot n\big) - q_{j-1} \cdot \big(a_j \cdot m + b_j \cdot n\big) \\ &= \big(a_{j-1} - q_{j-1} \cdot a_j\big) \cdot m + \big(b_{j-1} - q_{j-1} \cdot b_j\big) \cdot n. \end{aligned}$$

Hieruit volgt inderdaad vergelijking 2.6 voor alle $i \geq 1$. □

Voorbeeld 2.1. Bepaal $\text{ggd}(98, 34)$. In de meest rechtse kolom staat telkens de laatst gevonden rest als lineaire combinatie van de oorspronkelijke deler en deeltal.

Stap 1.	$98 = 2 \cdot 34 + 30$	$30 = 98 - 2 \cdot 34$
Stap 2.	$34 = 1 \cdot 30 + 4$	$4 = 34 - 1 \cdot 30$ $= 34 - (98 - 2 \cdot 34)$ $= 3 \cdot 34 - 98$
Stap 3.	$30 = 7 \cdot 4 + 2$	$2 = 30 - 7 \cdot 4$ $= 98 - 2 \cdot 34 - 7 \cdot (3 \cdot 34 - 98)$ $= 8 \cdot 98 - 23 \cdot 34$
Stap 4.	$4 = 2 \cdot 2 + 0$	

$\Rightarrow \text{ggd}(98, 34) = 2$ en $2 = 8 \cdot 98 - 23 \cdot 34$.

23 Pas nu stap voor stap het algoritme toe om $\text{ggd}(4864, 3458)$ te vinden en deze als lineaire combinatie van 4864 en 3458 te schrijven.

5. Programmacode voor het algoritme van Euclides

In tabel 2.1 staat het programma `euclides`, een implementatie voor de grafische rekenmachine van het Algoritme van Euclides.

Het programma vraagt waarden voor A en B in te voeren, waarna het als uitvoer de grootste gemene deler $\text{ggd}(A, B) = d$ en de getallen X en Y geeft zodat $d = X \cdot A + Y \cdot B$.

Tabel 2.1: Programmacode voor het algoritme van Euclides

```
program:euclides
  Prompt A : A → F :  Prompt B : B → G
  If B = 0
  Then
    1 → X : 0 → Y
    Disp "DE GGD IS",F
  End
  1 → Q : 0 → P : 0 → U : 1 → T
  While B ≠ 0
    A-iPart(A/B)*B→R
    Q-iPart(A/B)*P→X
    U-iPart(A/B)*T→Y
    B→A : R→B : P→Q : X→P : T→ U : Y→T
  End
  Q→X : U→ Y
  Disp "DE GGD IS ",A
  Disp "X=",X
  Disp " Y=",Y
```

24 Test de werking van dit programma bij een input $A = 4864$ en $B = 3458$ door onderstaande tabel in te vullen.

iPart(A/B)	R	X	Y	A	B	Q	P	U	T
-	-	-	-	4864	3458	1	0	0	1
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...

$$\text{ggd}(4864, 3458) = \ldots$$

$$\ldots = 4864 \cdot \ldots + 3458 \cdot \ldots$$

Hoofdstuk 3

Priemgetallen en priemfactorisatie

Op basis van de deelbaarheidseigenschappen uit de vorige les kunnen we nu kennismaken met de uiterst belangrijke priemgetallen. Dit zijn natuurlijke getallen die men niet "echt" kan delen, een eigenschap die van priemgetallen als het ware de bouwstenen van de gehele getallen maakt.

1. Priemgetallen

Al in de oudheid werden priemgetallen bestudeerd. Zo zijn er belangrijke eigenschappen van priemgetallen te vinden in **Euclides' Elementen** en dateert het bekendste algoritme om priemgetallen te vinden, de **zeef van Eratosthenes** uit dezelfde periode. Ondanks het feit dat er geen toepassingen van priemgetallen bekend waren buiten de zuivere wiskunde, bleven wiskundigen doorheen de geschiedenis steeds grotere priemgetallen zoeken en algoritmes ontwikkelen om het priem zijn van getallen te bepalen.

Deze situatie veranderde drastisch in de jaren zeventig van de vorige eeuw, toen encryptiemethoden werden ontwikkeld die intensief gebruik maken van priemgetallen. Sindsdien kan het belang ervan nauwelijks worden overschat, elke dag worden ongemerkt priemgetallen gebruikt bij talrijke elektronische verrichtingen. Het is een cruciaal begrip in heel wat moderne coderingsmethodes.

Definitie 3.1. *Een* ***priemgetal*** *is een natuurlijk getal p dat* precies *twee delers in $\mathbb{N}$ heeft, namelijk* 1 *en p. Getallen die echte delers hebben, worden* ***samengestelde getallen*** *genoemd.*

25 Onderzoek of 0 en 1 priemgetallen zijn.

26 Kan een even getal priem zijn?

27 Bewijs dat er voor elke $n \in \mathbb{N} \setminus \{0, 1\}$ ook n opeenvolgende natuurlijke getallen bestaan die geen priemgetallen zijn.
Hint: volgende 999 natuurlijke getallen zijn deelbaar,

$$1000! + 2, 1000! + 3, \ldots, 1000! + 1000.$$

Stelling 3.2. *Zij p een priemgetal en $a, b \in \mathbb{N}$. Dan geldt*

$$p \mid a \cdot b \;\Rightarrow\; (p \mid a) \vee (p \mid b)$$

28 *Bewijs als oefening.* Voor deze stelling is een **direct bewijs** mogelijk. Neem aan dat de hypothese $p \mid a \cdot b$ geldig is, dan kan volgende equivalente vorm van de conclusie $(p \mid a) \ \vee \ (p \mid b)$ aangetoond worden,

$$\begin{cases} p \nmid a \Rightarrow p \mid b \\ p \nmid a \Rightarrow p \mid a. \end{cases}$$

Formuleer een volledig en coherent bewijs op basis van volgende instructies.

1. Neem aan dat $p \nmid a$. Wat is dan $\text{ggd}(p, a)$? Schrijf deze grootste gemene deler als lineaire combinatie van p en a volgens het algoritme van Euclides.
2. Substitueer deze lineaire combinatie in de identiteit $b = 1 \cdot b$. Argumenteer dat p deler is van beide termen. Welke stelling toont dan dat p ook deler is van b.
3. Het bewijs voor de tweede implicatie is volledig analoog.

29 Waarom gaat de vorige stelling niet op als p geen priemgetal is?

2. Hoofdstelling van de getaltheorie

Volgende stelling toont dat priemgetallen inderdaad de "bouwstenen" voor alle natuurlijke getallen vormen.

Stelling 3.3 (**Hoofdstelling van de Getaltheorie**).
Elk natuurlijk getal a verschillend van 0 en 1, kan geschreven worden als product van priemgetallen. Op de volgorde van de priemfactoren na, is deze ***priemfactorisatie*** *uniek.*

30 *Bewijs als oefening.* Eerst moet het bestaan van een priemfactorenontbinding worden bewezen, dat kan door **volledige inductie** op het getal a. Daarna dient nog de uniciteit (op de volgorde na) van deze priemfactorisatie te worden aangetoond. Dit kan door volledige inductie op het aantal priemfactoren van a. Formuleer een volledig en coherent bewijs op basis van volgende instructies.

Existentie: er bestaat een priemfactorenontbinding.

Basisstap. Bewijs dat er een priemfactorenontbinding bestaat voor het kleinste natuurlijk getal verschillend van 0 en van 1.

Inductiestap. Onderstel dat a een willekeurig natuurlijk getal is, strikt groter dan 1. Neem nu aan als inductiehypothese dat een priemfactorisatie bestaat voor alle natuurlijke getallen vanaf 2 tot en met $a - 1$. Maak een onderscheid voor priemgetallen en getallen met echte delers.

Uniciteit: deze priemfactorenontbinding is uniek.

Basisstap. Wat is de priemfactorisatie in het geval van een priemgetal?

Inductiestap. Stel dat a geen priemgetal is en stel als inductiehypothese dat alle factorisaties bestaande uit minder dan n priemfactoren op de volgorde na uniek zijn. Bewijs uit het ongerijmde dat er geen twee verschillende priemfactorisaties voor eenzelfde getal kunnen bestaan door te tonen dat elke factor uit de ene factorisatie ook in de andere moet voorkomen.

Deze priemfactorenontbinding kan gebruikt worden om de grootste gemene deler en het kleinste gemeen veelvoud te bepalen, voor het berekenen van het totale aantal delers van een getal en voor het bewijzen van volgend zeer belangrijk resultaat.

Stelling 3.4. *Er zijn oneindig veel priemgetallen.*

Bewijs. Onderstel dat er slechts een eindig aantal, namelijk n, priemgetallen zijn, en dat we die dus kunnen opsommen als $p_1, p_2, \ldots, p_n$. Vorm nu het getal

$$a = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_n + 1 \tag{3.1}$$

De vorige stelling leert dat a geschreven kan worden als product van priemfactoren q_i uit de verzameling $\{p_1, p_2, p_3, \ldots, p_k\}$,

$$a = q_1 \cdot q_2 \cdot q_3 \cdot \ldots \cdot q_k.$$

Elk van de priemfactoren q_i deelt enerzijds het getal a en anderzijds $p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_k$. Uit (3.1) en door stelling 2.3 volgt dan dat $p_i \mid 1$. Dit is een contradictie en derhalve is de onderstelling dat er slechts eindig veel priemgetallen zijn fout. □

Later zal blijken waarom dit resultaat zo belangrijk is. Het garandeert immers dat, ongeacht hoe groot een priemgetal is, er steeds nog grotere priemgetallen bestaan. In codetheorie wordt gewerkt met zéér grote priemgetallen en omdat computers steeds beter worden en sneller codes breken die gebaseerd zijn op kleinere priemgetallen, zullen in de toekomst steeds grotere priemgetallen gebruikt moeten worden om beveiliging te garanderen.

31 Ontbind in priemfactoren:

1. $42 = \ldots$
2. $72 = \ldots$
3. $180 = \ldots$
4. $224 = \ldots$
5. $37800 = \ldots$

32 Bepaal nu via de vorige oefening

1. $\text{ggd}(180, 72)$ en $\text{kgv}(180, 72)$
2. $\text{ggd}(224, 42)$ en $\text{kgv}(224, 42)$

33 Bepaal het aantal delers in $\mathbb{N}$ van 2^n $(n \in \mathbb{N})$.

34 Gebruik priemfactorenontbinding om te bepalen hoeveel delers 720 heeft in $\mathbb{N}$.

35 Als a en b onderling ondeelbaar zijn en a en b zijn beide delers van n, dan is $a \cdot b$ ook een deler van n.

3. Deelbaarheidscriteria

Om te bepalen of een getal priem is, of om codes gebaseerd op priemgetallen te breken, is het in essentie nodig om de delers, bij voorbeeld de priemfactoren, te zoeken van dat getal n. Dat lijkt eenvoudig, maar eigenlijk komt dat er op neer dat de rest bij deling van n moet worden berekend voor alle natuurlijke getallen d kleiner dan n. Voor grote getallen is dit een enorm tijdsintensief proces, dat gelukkig kan worden versneld door rekening te houden met een aantal deelbaarheidscriteria.

Eerst en vooral volstaat het om enkel voor priemgetallen te testen of ze deler zijn van n. Is een getal d deler van n, dan zijn immers ook alle priemfactoren van d dat. Dit lijkt een enorme vereenvoudiging, maar omdat er geen snelle manier is om alle priemgetallen op te sommen, is dat enkel zo voor relatief kleine getallen.

Verder heeft men alle echte delers van een getal gevonden zodra de deelbaarheid van de getallen $d \leq \sqrt{n}$ is getest. Voor elke gevonden deler $d \mid n$ is het quotiënt q waarvoor $d \cdot q = n$ immers ook een deler.

Tot slot zijn volgende deelbaarheidstesten welbekend.

- Een geheel getal is deelbaar door 2 (is even) als en slechts als het eindigt op 0, 2, 4, 6 of 8. Equivalent, als en slechts als het laatste cijfer van het getal deelbaar is door 2.
- Een geheel getal is deelbaar door 4 als en slechts als het getal gevormd door de laatste twee cijfers deelbaar is door 4.
- Een geheel getal is deelbaar door 3 als en slechts als de som van de cijfers van dat getal deelbaar is door 3.
- Een geheel getal is deelbaar door 5 als en slechts als het eindigt op 0 of 5.
- Een geheel getal is deelbaar door 9 als en slechts als de som van de cijfers van dat getal deelbaar is door 9.
- Een geheel getal is deelbaar door 6 als en slechts als het deelbaar is door 2 en deelbaar is door 3.

36 Bewijs de correctheid van deze testen.

37 Over het getal $N = 2^{10} \cdot 10^2$ worden volgende uitspraken gedaan:

1. N is deelbaar door 5.
2. N is niet deelbaar door 25.
3. N is deelbaar door 40.
4. N is niet deelbaar door 50.

Hoeveel van deze uitspraken zijn juist?[1]

[1] V.W.O. 02-03, 1e ronde, vraag 14

38 Door welk getal is het product $26 \cdot 38 \cdot 51 \cdot 55 \cdot 56 \cdot 107$ niet deelbaar?[2]
(A) 9 (B) 17 (C) 32 (D) 66 (E) 190

39 Is 432165 deelbaar door 15? En door 45?

40 Is 721512 deelbaar door 36?

[2] J.W.O. 03-04, 1e ronde, vraag 20

Hoofdstuk 4

Rekenen met restklassen

In de cryptografie wordt meestal modulair gerekend: niet met een onbeperkte verzameling getallen zoals $\mathbb{Z}$, maar enkel met de resten $0, 1, \ldots, n-1$ die overblijven na deling door een vast getal n. Daarom worden eerst de restklassen modulo n gedefinieerd, waarmee het mogelijk zal zijn om te rekenen.

1. Restklassen

Definitie 4.1. *Zij n een natuurlijk getal, verschillend van 0. Dan noemen we twee gehele getallen a en b* ***congruent modulo*** *n als en slechts als het verschil $a - b$ een n-voud is, of in symbolen*

$$a \equiv b \mod n \quad \Leftrightarrow \quad n \mid a - b.$$

Stelling 4.2. *Twee gehele getallen a en b zijn congruent modulo n als en slechts als a en b dezelfde rest hebben bij Euclidische deling door n.*

41 *Bewijs als oefening.* De equivalentie in stelling 4.2 kan bewezen worden door de twee implicaties afzonderlijk aan te tonen. Volg daartoe onderstaande instructies en formuleer op basis daarvan een volledig en coherent bewijs.

1. Pas de stelling van de Euclidische deling toe voor deling van a en b door n. Substitueer de gevonden uitdrukkingen in $a - b$.
2. Toon dat de congruentie tussen a en b inderdaad een **voldoende voorwaarde** is voor gelijke resten door de definitie van $a \equiv b \mod n$ op dit verschil $a - b$ toe te passen.
3. Herschrijf vervolgens het verschil $a - b$ in het geval dat de resten bij deling van a en b door n gelijk zijn, wat ook de **nodige voorwaarde** bewijst.

42 Vul aan zodat je een ware uitspraak krijgt. Hoeveel mogelijkheden zijn er telkens?

$$\begin{aligned} 23 &\equiv \ldots \mod 7 \\ -14 &\equiv \ldots \mod 5 \\ 243 &\equiv \ldots \mod 16 \\ 19 &\equiv 5 \mod \ldots \end{aligned}$$

Definitie 4.3. *De* ***restklasse*** *$\overline{a}$ modulo n is de verzameling van alle getallen in $\mathbb{Z}$ die congruent zijn met a. Het getal a is een* ***representant*** *(of vertegenwoordiger) van deze restklasse modulo n. In symbolen*

$$\overline{a} = \{b \in \mathbb{Z} \mid a \equiv b \mod n\}$$

43 Geef vijf elementen van $\overline{3}$ modulo 5: $\ldots, \ldots, \ldots, \ldots$ en $\ldots$.
Merk op dat $\ldots = \ldots = \ldots = \ldots = \ldots$ modulo 5.

Stelling 4.4. *De restklassen modulo n vormen een* ***partitie*** *van $\mathbb{Z}$, dat wil zeggen dat ze voldoen aan volgende eigenschappen:*

- $\forall a \in \mathbb{Z}: \quad \emptyset \neq \overline{a}$
- $\bigcup_{a\in\mathbb{Z}} \overline{a} = \mathbb{Z}$
- $\forall a, b \in \mathbb{Z}: \quad (\overline{a} \cap \overline{b} \neq \emptyset \quad \Rightarrow \quad \overline{a} = \overline{b})$

44 *Bewijs als oefening.* Formuleer deze eigenschappen in eigen woorden en gebruik volgende instructies om een volledig en coherent bewijs te formuleren.

- Welk getal behoort zeker tot $\overline{a}$?
- Tot welke restklasse behoort een willekeurige element a uit $\mathbb{Z}$?
- Neem aan dat twee restklassen $\overline{a} \neq \overline{b}$ toch een element gemeenschappelijk hebben, bij voorbeeld $x \in \overline{a} \cap \overline{b}$. Druk uit dat x behoort tot beide klassen $\overline{a}$ en $\overline{b}$. Leid hier uit af dat a en b congruent zijn modulo n. Dit leidt tot een contradictie.

45 In figuur 4.1 worden de restklassen modulo n voor de waarden $n = 4$ en $n = 12$ gevisualiseerd. Vul nog wat bijkomende elementen aan in elke restklasse modulo 4. Van waar ken je de restklassen modulo 12?

De restklasse van 0 speelt een bijzondere rol. Voor elk natuurlijk getal n is de restklasse van 0 modulo n immers precies de deelverzameling van alle n-vouden in $\mathbb{Z}$, die we noteren met $n\mathbb{Z}$,

$$n\mathbb{Z} = \{a \cdot n \quad | \quad a \in \mathbb{Z}\}.$$

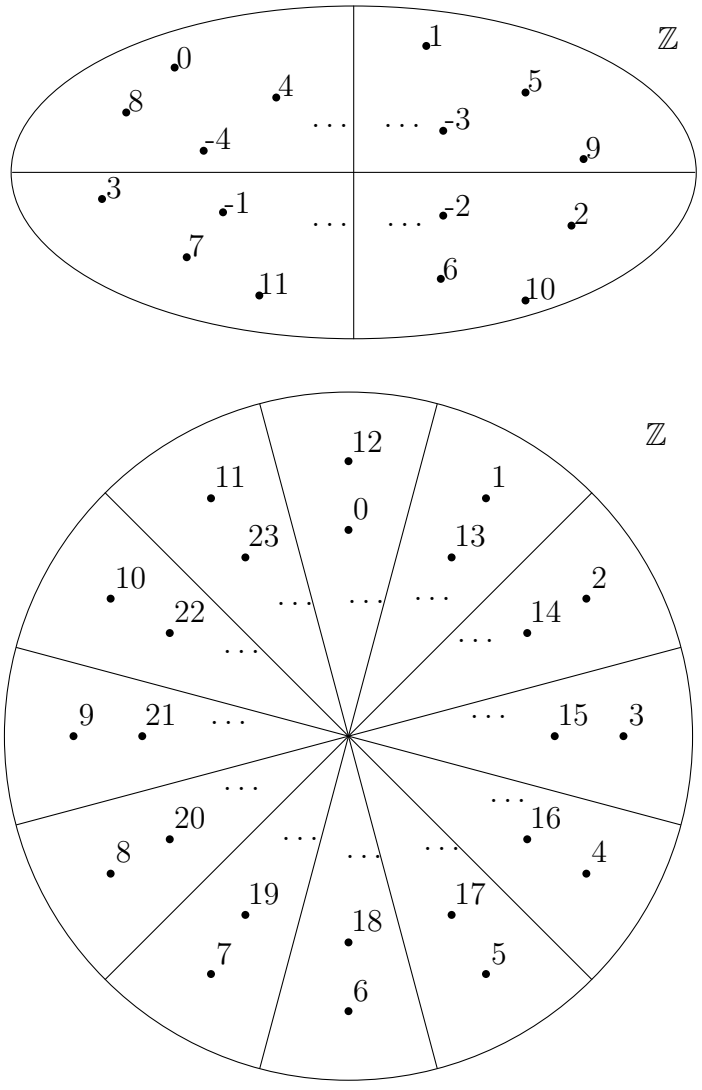
Een deelbaarheidstest uitvoeren, controleren of een gegeven geheel getal deelbaar is door n, komt dus neer op het nagaan of dat getal behoort tot de restklasse van 0 modulo n.

Definitie 4.5. *Voor elk natuurlijk getal $n \neq 0$, noteren we voortaan $\mathbb{Z}_n$ voor de verzameling van de restklassen modulo n.*

Merk op dat $\mathbb{Z}_n$ dus eigenlijk een *verzameling van verzamelingen* van gehele getallen is. De stap waarbij "restklassen modulo n" als elementen van een nieuwe verzameling $\mathbb{Z}_n$ genomen worden, vergt enige gewenning en is een niet-alledaagse abstrahering.

Voorbeeld 4.1.

- $\mathbb{Z}_3 = \{\overline{0}, \overline{1}, \overline{2}\}$. De elementen van $\mathbb{Z}_3$ zijn de verzamelingen $\overline{0}$, $\overline{1}$ en $\overline{2}$, dus $\mathbb{Z}_3 = \big\{\{\ldots, -3, 0, 3, 6, \ldots\}, \{\ldots, -2, 1, 4, 7, \ldots\}, \{\ldots, -1, 2, 5, 8, \ldots\}\big\}$.
- $\mathbb{Z}_{12} = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}, \overline{6}, \overline{7}, \overline{8}, \overline{9}, \overline{10}, \overline{11}\}$
- $\mathbb{Z}_2 = \{\overline{0}, \overline{1}\} = \{$"de even getallen", "de oneven getallen"$\}$



Figuur 4.1: Restklassen voor $n = 4$ en $n = 12$.

- $\mathbb{Z}_1 = \{\mathbb{Z}\}$

Voor elk strikt positief natuurlijk getal n geldt dus dat $\mathbb{Z}_n$ precies n elementen telt. Hoewel elk geheel getal een geldige representant is, zullen meestal de getallen $0, 1, \ldots, n-1$ gebruikt worden als "standaard"-representanten van de n verschillende restklassen modulo n.

De bedoeling in het vervolg van de tekst is om met de elementen van deze nieuwe verzameling $\mathbb{Z}_n$ te "rekenen", met als voor de hand liggende bewerkingen de optelling "+" en het product "$\cdot$" van restklassen.

Om verzamelingen van getallen te kunnen optellen en vermenigvuldigen is het natuurlijk wel nodig om precies te definiëren wat daarmee wordt bedoeld. Aangezien de elementen van de restklassen gehele getallen zijn, is het idee om de "gewone" bewerkingen als uitgangspunt te nemen voor de bewerkingen in $\mathbb{Z}_n$.

2. Optellen van restklassen

Om vast te leggen hoe de optelling van restklassen kan worden gedefinieerd kan de gewone optelling van getallen als inspiratie dienen.

46 Kies twee restklassen modulo 4, bijvoorbeeld $\overline{1}$ en $\overline{3}$. Kies binnen deze restklassen telkens een willekeurige (grote) representant. Tot welke restklasse behoort de som van deze representanten? Herhaal dit voor verschillende representanten en restklassen.

Er blijkt in de gekozen voorbeelden dat bij optelling van getallen uit twee restklassen, de som steeds tot dezelfde restklasse behoort, onafhankelijk van de gekozen representanten. Het controleren van enkele voorbeelden is natuurlijk geen bewijs dat dit steeds zal gebeuren, maar dit resultaat kan wel veralgemeend worden. Dankzij volgende stelling kan de som van twee restklassen $\overline{a}$ en $\overline{b}$ in $\mathbb{Z}_n$ gedefinieerd worden als de restklasse $\overline{a+b}$ met als representant de som van twee willekeurige representanten uit $\overline{a}$ en $\overline{b}$.

Stelling 4.6. *Als $\overline{a} = \overline{a'}$ en $\overline{b} = \overline{b'}$ dan geldt dat $\overline{a+b} = \overline{a'+b'}$ of, equivalent, als $a \equiv a' \mod n$ en $b \equiv b' \mod n$ dan geldt dat $(a+b) \equiv (a'+b') \mod n$.*

47 *Bewijs als oefening.* Vul de hiaten in volgende afleiding aan.

$$\left.\begin{array}{rcl} a \equiv a' \bmod n & \Rightarrow & n \mid \ldots \\ b \equiv b' \bmod n & \Rightarrow & n \mid \ldots \end{array}\right\} \begin{array}{l} \overset{\ldots}{\Rightarrow} \quad n \mid \ldots + \ldots = \ldots - \ldots \\ \Rightarrow \quad (a'+b') \equiv (a+b) \bmod n. \end{array}$$

Definitie 4.7. *In $\mathbb{Z}_n$ wordt een **optelling** $+$ tussen restklassen gedefinieerd als volgt,*

$$\forall \overline{a}, \overline{b} \in \mathbb{Z}_n : \quad \overline{a} + \overline{b} = \overline{a+b}.$$

Voorbeeld 4.2. Hieronder staan de **Cayley-tabellen** voor de optelling in $\mathbb{Z}_2$, $\mathbb{Z}_4$ en $\mathbb{Z}_5$. Merk op dat deze tabellen telkens een **Latijns vierkant** vormen – tabellen waarin elk symbool precies één keer voorkomt in elke rij en kolom.

$\mathbb{Z}_2, +$

$+$	$\overline{0}$	$\overline{1}$
$\overline{0}$	$\overline{0}$	$\overline{1}$
$\overline{1}$	$\overline{1}$	$\overline{0}$

$\mathbb{Z}_4, +$

$+$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{1}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{0}$
$\overline{2}$	$\overline{2}$	$\overline{3}$	$\overline{0}$	$\overline{1}$
$\overline{3}$	$\overline{3}$	$\overline{0}$	$\overline{1}$	$\overline{2}$

$\mathbb{Z}_5, +$

$+$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$
$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$
$\overline{1}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{0}$
$\overline{2}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{0}$	$\overline{1}$
$\overline{3}$	$\overline{3}$	$\overline{4}$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{4}$	$\overline{4}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$

48 Maak de Cayley-tabellen voor $+$ in $\mathbb{Z}_{12}$. Herken je in de resultaten de "uur"-bewerkingen?

3. Vermenigvuldigen van restklassen

In een volgende stap kan worden onderzocht of een analoge redenering mogelijk is om een "product" tussen elementen van $\mathbb{Z}_n$ in te voeren.

Stelling 4.8. *Als* $\overline{a} = \overline{a'}$ *en* $\overline{b} = \overline{b'}$, *dan geldt* $\overline{a \cdot b} = \overline{a' \cdot b'}$ *of, equivalent, als* $a \equiv a' \mod n$ *en* $b \equiv b' \mod n$ *dan geldt dat* $a \cdot b \equiv a' \cdot b' \mod n$.

49 *Bewijs als oefening.* Vul opnieuw volgende afleiding aan.

$$\left.\begin{array}{ll} a \equiv a' & \mod n \Rightarrow \quad n \mid \ldots \\ b \equiv b' & \mod n \Rightarrow \quad n \mid \ldots \end{array}\right\} \overset{\ldots}{\Rightarrow} n \mid \ldots \cdot \ldots$$

Nu geldt

$$(a' - a) \cdot (b' - b) = \ldots \cdot \ldots + \ldots \cdot \ldots - \ldots \cdot \ldots - \ldots \cdot \ldots,$$

en dus geldt, door gebruik te maken van eigenschap …, dat

$$n \mid (a' \cdot b' - a \cdot b),$$

of met andere woorden

$$a' \cdot b' \equiv a \cdot b \mod n.$$

Ook voor de vermenigvuldiging in $\mathbb{Z}_n$ is hiermee de onafhankelijkheid van de respresentanten aangetoond waardoor volgende vermenigvuldiging tussen restklassen goed gedefinieerd is.

Definitie 4.9. *In* $\mathbb{Z}_n$ *wordt een* ***product*** $\cdot$ *gedefinieerd als volgt,*

$$\forall \overline{a}, \overline{b} \in \mathbb{Z}_n : \quad \overline{a} \cdot \overline{b} = \overline{a \cdot b}.$$

50 Vul volgende Cayley-tabellen aan. Zijn dit ook Latijnse vierkanten? En als $\overline{0}$ buiten beschouwing wordt gelaten?

$\mathbb{Z}_2, \cdot$

.	$\overline{0}$	$\overline{1}$
$\overline{0}$		
$\overline{1}$		

$\mathbb{Z}_3, \cdot$

.	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{0}$			
$\overline{1}$			
$\overline{2}$			

$\mathbb{Z}_4, \cdot$

.	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$				
$\overline{1}$				
$\overline{2}$				
$\overline{3}$				

$\mathbb{Z}_5, \cdot$

.	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$
$\overline{0}$					
$\overline{1}$					
$\overline{2}$					
$\overline{3}$					
$\overline{4}$					

$\mathbb{Z}_6, \cdot$

.	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$
$\overline{0}$						
$\overline{1}$						
$\overline{2}$						
$\overline{3}$						
$\overline{4}$						
$\overline{5}$						

51 Dankzij het rekenen met restklassen is het nu zeer eenvoudig om het criterium voor deelbaarheid door negen te bewijzen. Ga na dat een getal tot dezelfde restklasse modulo 9 behoort als de som der cijfers van dat getal in tiendelige voorstelling.

4. Toepassingen van modulorekenen

De optelling en vermenigvuldiging van restklassen lijken misschien triviaal, maar ze zijn helemaal niet zo voor de hand liggend. Voor andere bewerkingen met getallen is het immers niet mogelijk een analogon voor restklassen te vinden, zo blijkt uit volgend voorbeeld.

Voorbeeld 4.3. Stel dat men de machtsverheffing $\overline{a}^{\overline{b}}$ van restklassen modulo n wil definiëren als de restklasse van de machtsverheffing van twee representanten, $\overline{a^b}$. In $\mathbb{Z}_3$ is $\overline{2^1} = \overline{2}$, terwijl voor andere representanten $5 \equiv 2 \mod 3$ en $4 \equiv 1 \mod 3$ geldt dat $\overline{5^4} = \overline{625} = \overline{1}$. Maar $\overline{2} \neq \overline{1}$ dus is de machtsverheffing niet onafhankelijk van de gekozen representanten en is het niet zinvol om $\overline{a}^{\overline{b}}$ te definiëren als $\overline{a^b}$.

De onafhankelijkheid van de representanten voor de optelling en de vermenigvuldiging modulo n laat zien dat het in $\mathbb{Z}$ niet nodig is met grote getallen te rekenen, maar dat het volstaat te werken met kleine representanten uit de respectievelijke restklassen. Een eerste, bekende toepassing is de negenproef.

De **negenproef** combineert de onafhankelijkheid van de representanten met het feit dat restklassen modulo 9 heel gemakkelijk kunnen bepaald worden om de correctheid van een berekende som of product extra te controleren:

$$n + m = s \text{ (optelling in } \mathbb{Z}) \Rightarrow n + m \equiv s \mod 9, \tag{4.1}$$

$$n \cdot m = p \text{ (product in } \mathbb{Z}) \Rightarrow n \cdot m \equiv p \mod 9. \tag{4.2}$$

Bij voorbeeld: ter controle van het product $2341 \cdot 5873 = 13748693$ berekent men deze vermenigvuldiging ook modulo 9. Dan vind je dat $2341 \equiv 2+3+4+1 \equiv 10 \equiv 1 \mod 9$ en dat $5873 \equiv 5+8+7+3 \equiv 23 \equiv 5 \mod 9$. Nu geldt $1 \cdot 5 = 5 \mod 9$ en we vinden inderdaad ook dat $13748693 \equiv 41 \equiv 5 \mod 9$. Dit is een bevestigende indicatie voor de correctheid van de oorspronkelijke bewerking.

De negenproef geeft inderdaad slechts een aanwijzing maar geen garantie dat de berekening klopt. De negenproef kan bijgevolg enkel gebruikt worden als ontdekker van fouten maar niet als bewijs voor de correctheid van de gemaakte berekening.

Ook in de context van deelbaarheidstesten komt modulorekenen van pas, zoals geïllustreerd in volgende voorbeelden. Dit lijken op het eerste zicht misschien nauwelijks te veralgemenen problemen met ad hoc oplossingen, maar het zijn precies dergelijke rekentrucs die zullen toelaten om met behulp van de juiste sleutel razendsnel geëncrypteerde gegevens te decoderen, terwijl dat zonder die sleutel ondoenbaar rekenwerk vergt.

Voorbeeld 4.4.

1. Bepaal de rest bij deling door 7 van $73 \cdot 52$. Anders gesteld, wat is de kleinste positieve representant van de restklasse waartoe $73 \cdot 52$ behoort?

   Gebruik makend van eigenschap 4.8 is het antwoord te vinden zonder het product $73 \cdot 52$ te moeten berekenen. Immers is $73 \equiv 3 \mod 7$ en $52 \equiv 3 \mod 7$, zodat $73 \cdot 52 \equiv 3 \cdot 3 \equiv 9 \equiv 2 \mod 7$. Het gevraagde antwoord is dus 2.

   Equivalent geldt in $\mathbb{Z}_7$ dat $\overline{73} \cdot \overline{52} = \overline{3} \cdot \overline{3} = \overline{9} = \overline{2}$.

2. Nagaan of $(2^{15}) \cdot (14^{40}) + 1$ deelbaar is door 11 komt er op neer om te bepalen of $(2^{15}) \cdot (14^{40}) + 1$ en 0 congruent zijn modulo 11.

   Opnieuw is het niet nodig om de volledige berekening uit te voeren – de uitkomst is een getal van vijftig cijfers! – maar kunnen alle termen gereduceerd worden modulo 11. Zoals aangetoond in voorbeeld 4.3 mogen de exponenten natuurlijk *niet* vervangen worden door een representant. Hier geldt dat $2^5 = 32$ en $32 \equiv -1 \mod 11$, zodat $2^{15} = (2^5)^3 \equiv (-1)^3 \equiv -1 \mod 11$. De factor 14^{40} kan achtereenvolgens gereduceerd worden tot $14^{40} \equiv 3^{40} \mod 11$; verder is $3^{40} = (3^2)^{20} \equiv (-2)^{20} \equiv 2^{20} \equiv (2^5)^4 \equiv (-1)^4 \equiv 1 \mod 11$. Dit resulteert in

   $$(2^{15}) \cdot (14^{40}) + 1 \equiv (-1) \cdot 1 + 1 \equiv 0 \mod 11.$$

   Equivalent geldt in $\mathbb{Z}_{11}$ dat $\overline{2^{15}} \cdot \overline{14^{40}} + \overline{1} = \overline{-1} \cdot \overline{1} + \overline{1} = \overline{0}$.

3. Bewijs dat $F_5 = 2^{2^5} + 1$ deelbaar is door 641 (Leonhard Euler, 1732).

   Enerzijds is $641 = 640 + 1 = 5 \cdot 2^7 + 1$ en bijgevolg $5 \cdot 2^7 \equiv -1 \mod 641$. Uit eigenschap 4.8 volgt dan dat

   $$5^4 \cdot 2^{28} \equiv 1 \mod 641. \tag{4.3}$$

   Anderzijds is $641 = 625 + 16 = 5^4 + 2^4$, waardoor

   $$5^4 \equiv -2^4 \mod 641. \tag{4.4}$$

   Uit vergelijkingen (4.3) en (4.4) volgt dan dat $-2^{32} \equiv 1 \mod 641$, of equivalent, $2^{32}+1 \equiv 0 \mod 641$.

Hoofdstuk 5

Modulorekenen en priemgetallen

In het vorige deel is gebleken dat de keuze van de representant bij optelling en vermenigvuldiging in $\mathbb{Z}_p$ geen rol speelt. Het heeft dus nog weinig zin om de notatie $\overline{a}$ voor restklassen aan te houden. Aangezien het geen verschil maakt of van twee getallen eerst de rest bij deling door n wordt genomen vooraleer ze worden opgeteld of vermenigvuldigd, dan wel of eerst de som of product wordt berekend en dan pas de rest bij deling door n wordt bepaald, "vergeet" men de restklasse-notaties en gaat men kortweg a noteren voor $\overline{a} \in \mathbb{Z}_n$. In plaats van te spreken van "rekenen met restklassen modulo n", spreekt men nu van "rekenen modulo n" of "rekenen met congruenties modulo n". Men zegt dat $3 + 2 = 1$ modulo 4 of dat $1 + 1 = 0$ modulo 2.

1. Invers element voor de vermenigvuldiging

Men kan zich nu afvragen of het in $\mathbb{Z}_n$ ook mogelijk zou zijn om te 'delen'? Met andere woorden, of er voor elk element $a \in \mathbb{Z}_n \setminus \{0\}$ een element $b \in \mathbb{Z}_n \setminus \{0\}$ bestaat zodat $a \cdot b = 1$ en $b \cdot a = 1$? Een dergelijk element b is een **invers element** van a voor de vermenigvuldiging. Ook al is het niet meteen duidelijk of zo een invers element steeds bestaat, het is wel eenvoudig om aan te tonen dat het invers element uniek is als het bestaat.

52 Toon aan dat een invers element, als het bestaat, uniek is. Neem daarvoor aan dat a twee inverse elementen heeft, zeg b en b' met $a \cdot b = b \cdot a = 1$ en $a \cdot b' = b' \cdot a = 1$. Schrijf $b \cdot a \cdot b'$ op twee manieren om te vinden dat $b = b'$.

Ook al heeft een element a hoogstens één invers element voor de vermenigvuldiging, is het nog niet duidelijk of zo een element steeds zal bestaan. Volgende oefening onderzoekt onder welke voorwaarden een invers element bestaat.

53 Beantwoord volgende vragen aan de hand van de Cayley-tabellen uit oefening 50.

1. In welk van deze Cayley-tabellen heeft elk niet-nul element een invers?
2. Voor welke klasse van getallen n lijkt dit het geval te zijn?
3. In welke Cayley-tabellen komen echte **nuldelers** voor? Dit zijn niet-nul elementen die toch als product nul hebben. Dit fenomeen doet zich niet voor in $\mathbb{R}$, maar bijvoorbeeld wel bij de vermenigvuldiging van matrices.

Volgende stelling bevestigt het vermoeden dat het optreden van nuldelers en het bestaan van een invers voor alle niet-nul elementen elkaar uitsluiten.

Stelling 5.1. *Als n geen priemgetal is, dan bevat $\mathbb{Z}_n$ echte nuldelers. Is n wel een priemgetal, dan heeft elk element van $\mathbb{Z}_n \setminus \{0\}$ een (uniek) invers element voor de vermenigvuldiging.*

54 *Bewijs als oefening.* Maak een gevalsonderscheid naargelang n priem is of niet. Formuleer een volledig en coherent bewijs op basis van de denkstappen hieronder.

Als n geen priemgetal is, dan heeft $n = a \cdot b$ echte delers a en b, die per definitie nuldelers zijn. Stel dat a een invers element c heeft, dan geeft de berekening van $(b \cdot a) \cdot c$ en $b \cdot (a \cdot c)$ een contradictie.

Als n wel een priemgetal is, construeer dan voor een willekeurig geheel getal a dat niet congruent is met nul modulo n de volgende deelverzameling van $\mathbb{Z}_n$

$$\{1 \cdot a, 2 \cdot a, 3 \cdot a, 4 \cdot a, \ldots, (n-1) \cdot a\} \subset \mathbb{Z}_n$$

Toon enerzijds aan dat geen enkel element $i \cdot a$ uit deze deelverzameling nul kan zijn. Anderzijds dat alle elementen $i \cdot a$ en $j \cdot a$ met $i \neq j$ uit deze deelverzameling verschillend zijn. Beide argumenten leiden tot

$$\{1 \cdot a, 2 \cdot a, 3 \cdot a, 4 \cdot a, \ldots, (p-1) \cdot a\} = \mathbb{Z}_n \setminus \{0\}, \tag{5.1}$$

waarmee het bewijs gemakkelijk te vervolledigen is.

In het geval van een priemgetal p bestaat dus voor elk getal a dat niet congruent is met nul modulo p een uniek invers element. We noteren dat element als a^{-1}.

Het bewijs van stelling 5.1 is een **niet-constructief bewijs** . Er wordt sluitend aangetoond dat er met zekerheid een invers element gevonden kan worden voor elk niet-nul element van $\mathbb{Z}_n$, maar er wordt niet beschreven hoe dat kan worden gevonden – geconstrueerd – zonder alle mogelijkheden te overlopen.

Voor het bepalen van de inverse van 53 in $\mathbb{Z}_{101}$ kan natuurlijk de volledige Cayley-tabel worden opgesteld, maar dat is een wel zeer rekenintensieve methode. Volgende redenering leidt veel sneller tot resultaat.

Zij p een priemgetal en a niet congruent met 0 modulo p, dan zijn a en p relatief priem. Wegens het algoritme van Euclides (stelling 2.8) bestaan gehele getallen x en y zodat

$$1 = x \cdot a + y \cdot p,$$

waaruit volgt dat

$$x \cdot a \equiv 1 \mod p.$$

Bijgevolg levert het algoritme van Euclides met x het invers element van a in $\mathbb{Z}_p$.

Voorbeeld 5.1. Voor het berekenen van 53^{-1} in $\mathbb{Z}_{101}$ levert het programma `euclides` in tabel 2.1 dat

$$1 = -40 \cdot 53 + 21 \cdot 101.$$

Vandaar geldt in $\mathbb{Z}_{101}$ dat $53^{-1} = -40 = 61$.

55 Bereken volgende inverse elementen.

- 321^{-1} in $\mathbb{Z}_{977}$
- 23^{-1} in $\mathbb{Z}_{65537}$
- 457861^{-1} in $\mathbb{Z}_{6700417}$

2. De kleine stelling van Fermat

Bij het bestuderen van de "deling" in $\mathbb{Z}_p$, vergelijking (5.1), is gebleken dat voor een priemgetal p de veelvouden van een getal $1 \leq a < p$ een **permutatie** van $\mathbb{Z}_p$ vormen. Nu kan de vraag rijzen hoe de oplopende machten van een element uit $\mathbb{Z}_p$ zich gedragen.

56 Bepaal voor elke $a \in \mathbb{Z}_p$ (p gelijk aan 5 of 7) de oplopende machten a^r met $r = 1, 2, \ldots$.

$\mathbb{Z}_5$

a \ r	1	2	3	4
0				
1				
2				
3				
4				

$\mathbb{Z}_7$

a \ r	1	2	3	4	5	6
0						
1						
2						
3						
4						
5						
6						

Uiteindelijk komt voor elk element verschillend van 0 de waarde 1 voor als macht van dat getal. Alhoewel dit niet voor alle elementen van $\mathbb{Z}_p$ even vaak of bij dezelfde exponenten gebeurt, is sowieso wel steeds a^{p-1} congruent met 1 voor elk niet-nul element a.

Dit fenomeen blijkt niet toevallig enkel in $\mathbb{Z}_5$ en $\mathbb{Z}_7$ voor te komen maar is te veralgemenen voor alle $\mathbb{Z}_p$ met p priem. Dit wordt bewezen in volgende stelling, een resultaat waarvan het belang voor de getaltheorie moeilijk te overschatten is.

Stelling 5.2 (**Kleine stelling van Fermat**). *Als p een priemgetal is, dan geldt voor alle elementen van $\mathbb{Z}_p \setminus \{0\}$ dat*

$$a^{p-1} = 1 \mod p \tag{5.2}$$

Bewijs. In vergelijking 5.1 werd reeds vastgesteld dat als a van nul verschilt in $\mathbb{Z}_p$, dan

$$\{1 \cdot a, 2 \cdot a, 3 \cdot a, 4 \cdot a, \ldots, (p-1) \cdot a\} = \{1, 2, 3, 4, \ldots, p-1\}.$$

Als beide verzamelingen gelijk zijn modulo p, moet het product van alle elementen in elke verzameling gelijk zijn modulo p,

$$\prod_{i=1}^{p-1} (i \cdot a) = \prod_{i=1}^{p-1} i$$

of, met $n! = 1 \cdot 2 \cdot \ldots \cdot n$,

$$(p-1)! \cdot a^{p-1} = (p-1)!.$$

Hieruit volgt dat

$$(p-1)! \cdot (a^{p-1} - 1) = 0$$

en omdat $(p-1)!$ niet congruent is met nul modulo p en er geen nuldelers zijn in $\mathbb{Z}_p$, geldt

$$a^{p-1} = 1.$$

□

Aangezien nu voor een priemgetal p elk element $a \in \mathbb{Z}_p \setminus \{0\}$ voldoet aan $a^{p-1} = 1$, is $a^{p-2} \cdot a = 1$ en is het invers element van a dus steeds

$$a^{-1} = a^{p-2}.$$

Door vermenigvuldiging van beide leden in vergelijking (5.2) met a verkrijgen we de volgende uitdrukking

$$a^p \equiv a \mod p.$$

Dit is een resultaat dat bovendien ook geldig is als $a \equiv 0 \mod p$; juist daarom is dit een populaire vorm waarin we de kleine stelling van Fermat vaak aantreffen.

57 Verifieer in de tabellen van oefening 50 dat het invers element van een element a telkens inderdaad gelijk is aan a^{p-2}.

3. Fermats priemgetaltest

De kleine stelling van Fermat kan gebruikt worden als elementaire **priemgetaltest**: door een getal a te vinden waarvoor a^{n-1} niet congruent is met 1 modulo n, is met zekerheid vastgesteld dat dit getal n *geen* priemgetal is. Omgekeerd is deze test niet sluitend, want voor een samengesteld getal n kan a^{n-1} voor bepaalde getallen a wel degelijk congruent zijn met 1 modulo n, al zullen de nuldelers nooit aan vergelijking 5.2 voldoen.

58 Waarom kunnen nuldelers onmogelijk aan vergelijking 5.2 voldoen?

In principe kan Fermats priemgetaltest wel herhaald worden, desnoods tot een nuldeler wordt gevonden, maar dit zou een nog meer rekenwerk vragen dan de deelbaarheid van n nagaan voor alle $1 < a < n$, omdat telkens eerst a^{n-1} moet worden berekend. De kracht van deze test zit in het feit dat een samengesteld getal met vrij grote waarschijnlijkheid zal worden ontmaskerd, terwijl een getal dat de test doorstaat *waarschijnlijk* priem is.

Voor een bijzondere, oneindig grote verzameling van getallen blijkt de test minder doeltreffend. De zogenaamde **Carmichael-getallen** die in 1912 werden ontdekt hebben de eigenschap dat enkel de nuldelers niet aan de vergelijking 5.2 voldoen, zodat de kans zeer klein is om van zo een getal te ontdekken dat het niet priem is. Het kleinste Carmichael-getal is $561 = 3 \cdot 11 \cdot 17$ en de enige andere kleiner dan 10000 zijn 1105, 1729, 2465, 2821, 6601 en 8911.

4. Het berekenen van machten modulo

RSA-coderingen en -decoderingen bestaan essentieel uit het berekenen van machten modulo n. Wil deze techniek in de praktijk bruikbaar zijn, dan is het nodig om deze heel snel te kunnen berekenen.

De meest voor de hand liggende manier om machten a^k modulo n te berekenen is eerst $a \cdot a$ berekenen, dan $a \cdot a \cdot a$ en $a \cdot a \cdot a \cdot a$, tot en met

$$a^k = \underbrace{a \cdot a \cdot \ldots \cdot a}_{k}$$

en vervolgens dit resultaat te delen door n om de gewenste rest te verkrijgen. Dit is om twee redenen een slecht idee. Eerst en vooral moeten k vermenigvuldigingen worden uitgevoerd, wat veel computertijd en werkgeheugen vergt. Anderzijds leiden berekeningen als bijvoorbeeld 2^{1110} mod n tot **overflow**.

Een efficiëntere manier is gebaseerd op de binaire schrijfwijze van de exponent k,

$$k = \sum_{i=0}^{t} k_i \cdot 2^i \text{ met } k_i \in \{0, 1\},$$

waarmee de berekening reduceert tot

$$a^k \equiv a^{\sum_{i=0}^{t} k_i \cdot 2^i} = \prod_{i=0}^{t} a^{k_i \cdot 2^i} = (a^{2^0})^{k_0} \cdot (a^{2^1})^{k_1} \cdot \ldots \cdot (a^{2^t})^{k_t} \mod n$$

waarbij

$$a^{2^{i+1}} \equiv (a^{2^i})^2 \mod n.$$

Elke factor is bijgevolg het kwadraat van de voorgaande factor modulo n en de vermenigvuldiging wordt slechts uitgevoerd als de binaire exponent 1 is. Een implementatie `machtmod` van dit algoritme is te vinden in tabel 5.1.

Voorbeeld 5.2. Voor het berekenen van 3^{55} modulo 7 dient 55 eerst op binaire wijze te worden geschreven,

$$(55)_{10} = (110111)_2.$$

Vervolgens moet niet de vijfenvijftigste macht van 3 worden berekend, maar slechts achtereenvolgende kwadraten modulo 7:

$$3^2 \equiv 2 \mod 7, \quad 3^4 \equiv 2^2 \equiv 4 \mod 7, \quad 3^8 \equiv 4^2 \equiv 2 \mod 7. \tag{5.3}$$

Tot slot worden de factoren met exponent 1 vermenigvuldigd,

$$\begin{aligned} 3^{55} &\equiv (3^{2^0})^1 \cdot (3^{2^1})^1 \cdot (3^{2^2})^1 \cdot (3^{2^3})^0 \cdot (3^{2^4})^1 \cdot (3^{2^5})^1 \\ &\equiv 3^1 \cdot 2^1 \cdot 4^1 \cdot 2^0 \cdot 4^1 \cdot 2^1 \\ &\equiv 3 \mod 7. \end{aligned}$$

59 Doorloop alle opeenvolgende stappen van het programma en vul volgende tabel aan voor de berekening van $5^{596} \mod 1234$.

I	1	2	3	4	5	6	7	8	9	10
⌊BINR(I)										
P										
B										

Tabel 5.1: `machtmod`: berekenen van de macht A^K modulo N

```
PROGRAM:MACHTMOD
  Prompt A
  prgmBINEXP
  Prompt N
  1 → B
  If K = 0
  Then
    Disp B
  Else
    A − N ∗ iPart(A/N) → P
    If ⌊BINR(1) = 1
    Then
      A − N ∗ iPart(A/N) → B
    End
    For(I, 2, dim(⌊BIN))
      P^2 − N ∗ iPart(P^2/N) → P
      If ⌊BINR(I) = 1
      Then
        P ∗ B − N ∗ iPart(P ∗ B/N) → B
      End
    End
  Disp B
```

60 Een bankrekeningnummer bestaat uit 12 cijfers,

$$a_1a_2a_3 - a_4a_5a_6a_7a_8a_9a_{10} - a_{11}a_{12}.$$

Neem het getal gevormd door de eerste 10 cijfers van een bestaand bankrekeningnummer en bereken dit getal modulo 97. De conclusie zal telkens dezelfde zijn.

61 Bereken $2^{11111110} \mod 11111111$.

62 Ga met de kleine stelling van Fermat en het programma `machtmod` na of volgende getallen priemgetallen zijn.

- 220987
- 1473407
- 65537
- 6601
- 8911

Hoofdstuk 6

Het RSA-codeersysteem

Eindelijk zijn alle elementen verzameld om een **publieke sleutel codeersysteem** te bouwen. Dit systeem zal alle eigenschappen van het rekenen met congruenties volledig uitbuiten. Het is natuurlijk de bedoeling dat bij elke codeer/decodeer operatie het resultaat van het decoderen identiek is aan de oorspronkelijk gecodeerde boodschap. Het coderen hoort dus een bijectie f te zijn van de verzameling van de klare boodschappen naar de verzameling van de gecodeerde boodschappen.

Om in de praktijk bruikbaar te zijn, moet het cryptosysteem bovendien eenvoudig en snel uit te voeren zijn, maar de code zelf moeilijk te kraken. De functie f moet dus makkelijk berekenbaar en tegelijk moeilijk inverteerbaar zijn: de kracht van een codeersysteem hangt af van hoe moeilijk het is om de inverse afbeelding van f te vinden.

In de volgende sectie wordt gezocht naar een geschikte functie f die aan al deze voorwaarden voldoet.

1. Op zoek naar de sleutelstelling voor RSA

Zoals in sectie 4 van hoofdstuk 1 geschetst wordt de klaartekst omgezet in bits, één lange rij van nullen en enen. Deze rij wordt vervolgens opgesplitst in kleinere stukken die worden geïnterpreteerd als getallen in binaire expansie die elk afzonderlijk zullen worden geëncrypteerd. Deze encryptie zal in de verzameling $\mathbb{Z}_n$ gebeuren. De gezochte functie f moet dus makkelijk modulo n te berekenen zijn.

In sectie 4 van hoofdstuk 5 is in detail besproken dat afbeeldingen f_r van de vorm

$$f_r : \mathbb{Z}_n \to \mathbb{Z}_n : a \mapsto f_r(a) = a^r$$

met exponent $r \in \mathbb{N}$ in binaire expansie, relatief eenvoudig en snel uit te rekenen zijn. Op de rekenmachine is deze berekening met het programma `machtmod` voor relatief kleine waarden van n mogelijk, maar binnen de hedendaagse technologie stellen de berekeningen geen bijzondere moeilijkheden, zelfs als r en n zeer groot zijn.

Het komt er nu enkel nog op aan te bepalen

1. voor welke waarden van n en r de afbeelding f_r een bijectie is en
2. wat dan de inverse afbeelding f_r^{-1} is.

Tabel 6.1: De functies f_r in $\mathbb{Z}_{10}$ en Z_{14}

		r			
		3	5	7	9
	0	0	0	0	0
	1	1	1	1	1
	2	8	2	8	2
	3	7	3	7	3
	4	4	4	4	4
a	5	5	5	5	5
	6	6	6	6	6
	7	3	7	3	7
	8	2	8	2	8
	9	9	9	9	9

		r					
		3	5	7	9	11	13
	0	0	0	0	0	0	0
	1	1	1	1	1	1	1
	2	8	4	2	8	4	2
	3	13	5	3	13	5	3
	4	8	2	4	8	2	4
	5	13	3	5	13	3	5
	6	6	6	6	6	6	6
a	7	7	7	7	7	7	7
	8	8	8	8	8	8	8
	9	1	11	9	1	11	9
	10	6	12	10	6	12	10
	11	1	9	11	1	9	11
	12	6	10	12	6	10	12
	13	13	13	13	13	13	13

63 In volgende tabellen kunnen de functies $f_r : \mathbb{Z}_n \to \mathbb{Z}_n$ worden geëxploreerd voor volgende waarden van n,

$n = p \cdot q$	tabel
$10 = 2 \cdot 5$	6.1
$14 = 2 \cdot 7$	6.1
$15 = 3 \cdot 5$	6.2
$21 = 3 \cdot 7$	6.2
$35 = 5 \cdot 7$	6.3

Ga na welke waarden van r bijecties f_r in Z_n opleveren op basis van de hiervolgende vragen.

1. Gebruik een bekende eigenschap van even machten om te motiveren waarom even exponenten geen bijecties zullen opleveren.
2. Bepaal voor elke n de verzameling A_n.

$$A_n = \{r \in \{3, \ldots, n-1\} \mid f_r : \mathbb{Z}_n \to \mathbb{Z}_n : a \mapsto a^r \text{ is een bijectie } \}.$$

$A_{10} = \{ \ldots \}$
$A_{14} = \{ \ldots \}$
$A_{15} = \{ \ldots \}$
$A_{21} = \{ \ldots \}$
$A_{35} = \{ \ldots \}$

3. Bepaal voor elke n de verzameling B_n.

$$B_n = \{x \in \{3, \ldots, n-1\} \mid \forall r \in \{3, \ldots, n-1\} : \text{ggd}(r, x) = 1 \Leftrightarrow r \in A_n\}.$$

Tabel 6.2: De functies f_r in $\mathbb{Z}_{15}$ en $\mathbb{Z}_{21}$

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

		r								
		3	5	7	9	11	13	15	17	19
	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1
	2	8	11	2	8	11	2	8	11	2
	3	6	12	3	6	12	3	6	12	3
	4	1	16	4	1	16	4	1	16	4
	5	20	17	5	20	17	5	20	17	5
	6	6	6	6	6	6	6	6	6	6
	7	7	7	7	7	7	7	7	7	7
	8	8	8	8	8	8	8	8	8	8
	9	15	18	9	15	18	9	15	18	9
a	10	13	19	10	13	19	10	13	19	10
	11	8	2	11	8	2	11	8	2	11
	12	6	3	12	6	3	12	6	3	12
	13	13	13	13	13	13	13	13	13	13
	14	14	14	14	14	14	14	14	14	14
	15	15	15	15	15	15	15	15	15	15
	16	1	4	16	1	4	16	1	4	16
	17	20	5	17	20	5	17	20	5	17
	18	15	9	18	15	9	18	15	9	18
	19	13	10	19	13	10	19	13	10	19
	20	20	20	20	20	20	20	20	20	20

Tracht eerst in woorden te omschrijven welke elementen B_n precies bevat.
$B_{10} = \{$ .... $\}$
$B_{14} = \{$ .... $\}$
$B_{15} = \{$ .... $\}$
$B_{21} = \{$ .... $\}$
$B_{35} = \{$ .... $\}$

4. Formuleer een hypothese waarbij je één van de elementen van B_n kan schrijven als een symmetrische functie van p en q (Herinner: $n = p \cdot q$.).

Tabel 6.3: De functies f_r in $\mathbb{Z}_{35}$

		r															
		3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	8	32	23	22	18	2	8	32	23	22	18	2	8	32	23	22
	3	27	33	17	13	12	3	27	33	17	13	12	3	27	33	17	13
	4	29	9	4	29	9	4	29	9	4	29	9	4	29	9	4	29
	5	20	10	5	20	10	5	20	10	5	20	10	5	20	10	5	20
	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	7	28	7	28	7	28	7	28	7	28	7	28	7	28	7	28	7
	8	22	8	22	8	22	8	22	8	22	8	22	8	22	8	22	8
	9	29	4	9	29	4	9	29	4	9	29	4	9	29	4	9	29
	10	20	5	10	20	5	10	20	5	10	20	5	10	20	5	10	20
	11	1	16	11	1	16	11	1	16	11	1	16	11	1	16	11	1
	12	13	17	33	27	3	12	13	17	33	27	3	12	13	17	33	27
	13	27	13	27	13	27	13	27	13	27	13	27	13	27	13	27	13
	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
	16	1	11	16	1	11	16	1	11	16	1	11	16	1	11	16	1
a	17	13	12	3	27	33	17	13	12	3	27	33	17	13	12	3	27
	18	22	23	32	8	2	18	22	23	32	8	2	18	22	23	32	8
	19	34	24	19	34	24	19	34	24	19	34	24	19	34	24	19	34
	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
	22	8	22	8	22	8	22	8	22	8	22	8	22	8	22	8	22
	23	22	18	2	8	32	23	22	18	2	8	32	23	22	18	2	8
	24	34	19	24	34	19	24	34	19	24	34	19	24	34	19	24	34
	25	15	30	25	15	30	25	15	30	25	15	30	25	15	30	25	15
	26	6	31	26	6	31	26	6	31	26	6	31	26	6	31	26	6
	27	13	27	13	27	13	27	13	27	13	27	13	27	13	27	13	27
	28	7	28	7	28	7	28	7	28	7	28	7	28	7	28	7	28
	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
	30	15	25	30	15	25	30	15	25	30	15	25	30	15	25	30	15
	31	6	26	31	6	26	31	6	26	31	6	26	31	6	26	31	6
	32	8	2	18	22	23	32	8	2	18	22	23	32	8	2	18	22
	33	27	3	12	13	17	33	27	3	12	13	17	33	27	3	12	13
	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34

2. De sleutelstelling voor RSA

Geduldig speurwerk, aan de hand van alle vorige voorbeelden, leidt uiteindelijk tot het volgende resultaat, dat als een sleutelstelling voor RSA kan beschouwd worden.

Stelling 6.1 (Sleutelstelling voor RSA). *Als p en q verschillende priemgetallen zijn met $n = p \cdot q$ en als*

$$\text{ggd}(r, (p-1)\cdot(q-1)) = 1,$$

dan is $f_r : \mathbb{Z}_n \to \mathbb{Z}_n : a \mapsto f(a) = a^r$ een permutatie van $\mathbb{Z}_n$.

64 *Bewijs als oefening.* Als $\text{ggd}(r, (p-1)\cdot(q-1)) = 1$, dan leert het algoritme van Euclides 2.1 dat deze grootste gemene deler te schrijven is als lineaire combinatie van r en $(p-1)\cdot(q-1)$,

$$1 = r \cdot s + (p-1)\cdot(q-1)\cdot t.$$

1. Gebruik deze identiteit en de kleine stelling van Fermat om te tonen dat $a^{r\cdot s}$ congruent is met a modulo p zowel als met a modulo q.
2. Hiermee is het mogelijk om aan te tonen dat $a^{r\cdot s} - a$ deelbaar is door $p \cdot q$ en dus dat
$$(a^r)^s \equiv a \mod p \cdot q.$$
3. Argumenteer met behulp van oefening 14 dat f_r inderdaad een permutatie is van $\mathbb{Z}_n$ en dat f_s de inverse afbeelding van f_r.

Formuleer op basis van deze denkstappen een volledig en coherent bewijs.

Voorbeeld 6.1. Zij $n = 407 = 37 \cdot 11$ en $r = 77$, dan is $(p-1)\cdot(q-1) = 360$ en geldt inderdaad dat $\text{ggd}(77, 360) = 1$. Het algoritme van Euclides (bijvoorbeeld via het programma `euclid`) toont dat

$$1 = 77 \cdot 173 - 360 \cdot 37.$$

Zodoende is $s = 173$.

Ook het omgekeerde resultaat van de stelling is geldig. Als f_r een permutatie is van $\mathbb{Z}_n$, met p en q verschillende priemgetallen en $n = p \cdot q$, dan zijn r en $(p-1)\cdot(q-1)$ relatief priem. Een bewijs op basis van enkel deelbaarheidseigenschappen is echter niet zo eenvoudig te geven.

Het is best mogelijk dat het getal s, dat geconstrueerd wordt in het algoritme van Euclides, negatief is, maar machtsverheffingen met negatieve exponenten zijn niet gedefinieerd in $\mathbb{Z}_n$. De uitdrukking $a^{r\cdot s} = a \cdot (a^{(p-1)})^{-t\cdot(q-1)}$ in het bewijs verandert echter niet als bij s een geheel veelvoud van $(p-1)\cdot(q-1)$ wordt opgeteld:

$$a^{r\cdot(s+k\cdot(p-1)\cdot(q-1))} = a^{r\cdot s} \cdot (a^{p-1})^{k\cdot(q-1)\cdot r} = a^{r\cdot s}$$

In s negatief, dan kan deze dus vervangen worden door een getal tussen 0 en $(p-1)\cdot(q-1)$, de standaardrepresentant modulo $(p-1)\cdot(q-1)$.

3. Concrete werking van het publieke sleutel systeem

Alle elementen zijn nu voorhanden om tot de constructie van het ingenieus publieke sleutel codeersysteem over te gaan dat in de literatuur bekend staat als het RSA-systeem, genoemd naar Ronald Linn **Rivest**, Adi **Shamir** en Leonard Max **Adleman** die het omstreeks 1975 voorstelden. Het was het eerste en is nog altijd het belangrijkste publieke sleutel codeersysteem. Het algoritme wordt hier stap voor stap beschreven en schematisch voorgesteld in tabel 6.4.

De eerste stap om met dit systeem aan de slag te gaan, is het bepalen van een geschikt getal n, en dus van twee priemgetallen p en q door de ontvanger. Gelukkig bestaan er vrij goede algoritmes om na te gaan of een gegeven *groot* getal priem of niet priem is, zonder het effectief te moeten factoriseren, zoals **Fermats priemgetaltest** in sectie 3 van hoofdstuk 4.

Om vervolgens een boodschap te encrypteren, wordt de klaartekst omgezet in een rij van bits (zie sectie 4 in hoofdstuk 1). Deze bits worden dan samengevoegd tot binaire getallen die beschouwd zullen worden als standaardrepresentanten binnen $\mathbb{Z}_n$, en dus kleiner moeten zijn dan n. Typisch werkt een computer met getallen die een zuivere tweemacht 2^k aan bits tellen, zodat n méér dan 2^k bits moet tellen. Ter versleuteling wordt voor elk van deze 2^k-bitsgetallen f_r uitgerekend. Hiervoor is het enkel nodig om r en n te kennen, de getallen waarover alle zenders moeten kunnen beschikken en welke bijgevolg de **publieke sleutel** vormen.

Dankzij het algoritme van Euclides (sectie 4 in hoofdstuk 2) is het mogelijk om s te construeren en daarmee de functie f_s, nodig voor het decrypteren. Hiervoor moeten wel de priemgetallen p en q gekend zijn, die dus de **private sleutel** vormen en de ontvanger angstvallig geheim dient te houden.

Boodschappen encrypteren en decrypteren met RSA gebeurt essentieel door machten r en s te berekenen van (grote) getallen modulo een getal n. Dit is opnieuw relatief werkbaar omdat deze berekeningen zeer efficiënt kunnen uitgevoerd worden, zoals omstandig uitgelegd in sectie 4 van hoofdstuk 5.

Het is nodig om s te vinden en dus om n te factoriseren (hoofdstuk 3) om RSA te kraken. In de praktijk is dit enorm tijdsintensief. Het is immers nog steeds niet mogelijk om een willekeurig gegeven zeer groot getal, waarvan men weet dat het geen priemgetal is, snel te ontbinden in priemfactoren. Hierop is precies de kracht van RSA gebouwd. Vandaag wordt 512-bit-encryptie

Tabel 6.4: Schematische voorstelling van het RSA-systeem

Ontvanger	Publiek netwerk	Zender
		Schrijf boodschap
		Converteer naar ASCII
Zoek p en q		Breek in stukken $a_i, i = 1, \ldots, m$
Bereken n en kies r $\rightarrow$	Publiceer n en r $\rightarrow$	Download n en r
Ontvang $f_r(a_i), i = 1, \ldots, m \leftarrow$	Verzend $f_r(a_i), i = 1, \ldots, m \leftarrow$	Bereken $f_r(a_i), i = 1, \ldots, m$
Bereken s uit algoritme 4		
Decrypteer de code $f_s(f_r(a_i)), i = 1, \ldots, m$		
Converteer naar klaartekst		

al niet meer als helemaal veilig beschouwd, terwijl een getal n van méér dan $2^9 = 512$ bits meer dan 150 decimalen telt. Zelfs een krachtige computer doet er ettelijke jaren over om zo een getal in priemfactoren te ontbinden. Voor 1024-bits getallen wordt deze ontbinding vandaag nog als technologisch onmogelijk beschouwd. Het RSA-systeem is dus gebaseerd op de afwezigheid van een voldoende snel algoritme voor priemfactorisatie. Hierin schuilt een potentiële bedreiging voor het systeem: indien de wiskunde hier een onverwachte stap voorwaarts zou maken, is het mogelijk dat de meeste gebruikte beveiliging op slag waardeloos wordt.

Uit dit overzicht blijkt overduidelijk dat alle hoofdstukken uit deze lessenreeks van pas komen en dat RSA handig gebruik maakt van alle geziene eigenschappen in verband met deelbaarheid.

4. Rekenvoorbeelden van RSA

Dit rekenvoorbeeld illustreert eenvoudige 16-bits-encryptie die natuurlijk zeer eenvoudig te kraken is. De private sleutel voor zo een systeem moet uit twee priemgetallen bestaan waarvan het product meer dan 16 bits telt en bijgevolg groter moet zijn dan $(1111111111111111)_2 = (65536)_{10}$, maar tegelijk kleiner moet zijn dan 2^{17} zoals verderop zal blijken.

Ontvanger. Met de kleine priemgetallen $p = 271$ en $q = 257$ berekent de ontvanger het getal $n = 271 \cdot 257 = 69647$ dat groot genoeg is voor 8-bits-encryptie. Hij kiest verder een getal r dat relatief priem is met $(p-1)\cdot(q-1) = 270 \cdot 256 = 69120 = 2^9 \cdot 3^3 \cdot 5$, bijvoorbeeld $r = 53$. Als publieke sleutel maakt de ontvanger dus de getallen $n = 69647$ en $r = 53$ bekend: wie hem een geëncrypteerde boodschap wil sturen, gebruikt de permutatie

$$f_{53} : \mathbb{Z}_{69647} \to \mathbb{Z}_{69647} : a \mapsto f_{53}(a) = a^{53}.$$

Zender. Wil de zender een boodschap

"Bewijs gevonden"

doorsturen naar de ontvanger, dan zet hij deze eerst om in een rij van getallen, bijvoorbeeld volgens de 8-bit ASCII-codering in tabel 1.2,

66 101 119 105 106 115 32 103 101 118 111 110 100 101 110.

Voor een computer is deze getallenrij één lange opeenvolging van $15 \cdot 8 = 120$ bits,

0 1 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0 1 0 0 1 1 1 0 0
1 1 0 0 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 0
1 1 1 0 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 1 0.

Deze rij wordt vooraan aangevuld met 8 nullen zodat een zestienvoud van $128 = 8 \cdot 16$ bits ontstaat, te beschouwen als 8 opeenvolgende 16-bitsgetallen a_i, $i = 1, \ldots, 8$. De functie f_{53} beeldt elk van deze getallen dan af op de geëncrypteerde boodschap (zie tabel 6.5) die opnieuw uit acht opeenvolgende binaire getallen $f_{53}(a_i)$, $i = 1, \ldots, 8$ bestaat. Deze getallen tellen echter elk één bit meer: het is weliswaar zo dat $a_i < 2^{16} < n$ maar a_i^r kan elk element van $\mathbb{Z}_n$ zijn,

dus ook een getal tussen 2^{16} en n. Door n tussen 2^{16} en 2^{17} te kiezen wordt elk getal a_i^r met leidende nullen inbegrepen dus één bit langer dan a_i, $i = 1, \ldots, 8$, zodat de geëncrypteerde code uit $8 \cdot 17 = 136$ bits bestaat.

$$\begin{array}{l} 0\,0\,1\,1\,0\,0\,0\,1\,0\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0 \\ 1\,1\,1\,1\,1\,0\,1\,0\,1\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0\,1\,0\,0\,1\,1\,1\,1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,1\,1\,0\,1\,1\,1\,1\,1\,0\,1 \\ 1\,1\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,0\,1. \end{array} \tag{6.1}$$

De ontvanger. De ontvanger splitst de boodschap (6.1) bestaande uit 136 bits opnieuw in 8 getallen van 17 bits. Een functie f_s kan deze getallen decrypteren. Hiervoor berekent de ontvanger de exponent s uit het algoritme van Euclides,

$$1 = (-26083) \cdot r + 20 \cdot (p-1) \cdot (q-1).$$

Het gevonden getal is negatief, maar kan worden verhoogd met $(p-1) \cdot (q-1)$ tot als geldige exponent s een getal tussen 0 en $(p-1) \cdot (q-1)$ wordt bekomen.

$$s = -26083 + (p-1) \cdot (q-1) = 43037.$$

De decrypteerfunctie is bijgevolg

$$f_{43037} : \mathbb{Z}_{69647} \to \mathbb{Z}_{69647} : a \mapsto f_{43037}(a) = a^{43037}.$$

De lezer verifieert dat voor $i = 1, \ldots, 8$ het getal $f_s(f_r(a_i))$ inderdaad gelijk is aan a_i.

Merk op dat de publieke sleutel in dit rekenvoorbeeld zeer makkelijk te kraken is. Het vereist immers niet exuberant veel rekenkracht om $n = 69647$ te factoriseren. In zekere zin is dit codeersysteem een polygafisch substitutiecijfer, aangezien symbolen twee aan twee gecodeerd worden als getallen a_i. In het algemeen, met realistische waarden voor p en q (en waarbij het getal k met $2^k < n < 2^{k+1}$ geen veelvoud is van 8), corresponderen deze getallen a_i met meer dan honderd symbolen (en zelfs stukken van symbolen) zodat klassieke frequentie-analyse helemaal onmogelijk wordt.

Tabel 6.5: Toepassing van de encryptie-functie in een concreet geval

$$\begin{array}{lll}
a_1 = (0000000001000010)_2 = (66)_{10} & \Rightarrow & f_{53}(a_1) = (00110001000110101)_2 = (25141)_{10} \\
a_2 = (0110010101110111)_2 = (25975)_{10} & \Rightarrow & f_{53}(a_2) = (01010101001001000)_2 = (43592)_{10} \\
a_3 = (0110100101101010)_2 = (26986)_{10} & \Rightarrow & f_{53}(a_3) = (01101111011011111)_2 = (57055)_{10} \\
a_4 = (0111001100100000)_2 = (29472)_{10} & \Rightarrow & f_{53}(a_4) = (01010001111110001)_2 = (41969)_{10} \\
a_5 = (0110011101100101)_2 = (26469)_{10} & \Rightarrow & f_{53}(a_5) = (00111101101011011)_2 = (31579)_{10} \\
a_6 = (0111011001101111)_2 = (30319)_{10} & \Rightarrow & f_{53}(a_6) = (01111011101000011)_2 = (63299)_{10} \\
a_7 = (0110111001100100)_2 = (28260)_{10} & \Rightarrow & f_{53}(a_7) = (01010101100111000)_2 = (43832)_{10} \\
a_8 = (0110010101101110)_2 = (25966)_{10} & \Rightarrow & f_{53}(a_8) = (00111000001111101)_2 = (28797)_{10}
\end{array}$$

65 Ontcijfer volgende boodschap bestaande uit 9-bitsgetallen die is bestemd voor een ontvanger die als publieke sleutel $n = 323$ en $r = 5$ gebruikt:

0 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 1 1 1 1 0
1 1 0 1 1 1 1 1 0 1 0 1 0 0 1 0 1 1 0 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0
1 1 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 1 1 0.

66 Kies elk twee priemgetallen p en q zodat $n = p \cdot q$ minstens 16 bits telt. Zoek een geschikte r en wissel de publieke sleutel uit met elkaar. Encrypteer hiermee een boodschap van ongeveer dertig letters en laat deze ontcijferen door de eigenaar van de publieke sleutel.

5. De digitale handtekening

Hierboven werd het RSA-cryptosysteem gebruikt als encryptiesysteem met als bedoeling informatie onleesbaar te maken en geheim te houden voor derden. Een andere toepassing van dit cryptosysteem kan er in bestaan de authenticiteit van een document aan te tonen of de ondertekenaar ervan te identificeren. Dit is wat bekend staat als **digitale handtekening**.

In de praktijk zal een digitaal gehandtekende boodschap natuurlijk ook vaak geëncrypteerd worden, zodat het RSA-systeem twee keer toegepast wordt. Eén keer met de cryptofuncties van de zender en een tweede keer met de cryptofuncties van de ontvanger. De sleutelset p, q, n, r, s van de ontvanger zal zoals hierboven met kleine letters genoteerd worden, de sleutelset P, Q, N, R, S van de zender met hoofdletters

Veronderstel dat de zender een brief stuurt naar de ontvanger, en dat het hierbij erg belangrijk is dat de ontvanger de identiteit van de zender kan verifiëren. De zender moet dus een stuk informatie toevoegen op een manier waartoe enkel hijzelf in staat is én zo dat de ontvanger dat ook kan nagaan. Daarom zal de zender de brief ondertekenen met een clausule die hij encrypteert met de functie f_S, op basis van zijn eigen private sleutel uit het RSA systeem. Als de ontvanger deze informatie krijgt, is hij in staat deze met de functie f_R te ontcijferen. Hij gebruikt daarvoor de publiek toegankelijke sleutel van de zender. Blijkt de ontsleutelde clausule leesbaar te zijn, dan kan deze enkel door de eigenaar van de private sleutel zijn geëncrypteerd.

Typisch wil de zender er natuurlijk ook voor zorgen dat de inhoud van de brief zelf niet zomaar ter beschikking komt en beschermd blijft. Hij zal daarom het geheel van de brief samen met de ondertekeningsclausule encrypteren met de publieke sleutel f_r van de ontvanger. Zo kan enkel de ontvanger, door gebruik te maken van zijn private sleutel f_s, de briefinhoud ontcijferen. Pas dan zal hij de nog steeds geëncrypteerde ondertekening aantreffen, die hij daarna kan verifiëren zoals hierboven beschreven.

Het is echter niet de bedoeling dat een oude correspondent van de zender, die ooit een gehandtekend document heeft gekregen, de clausule uit dat document kan recycleren om in de naam van die zender boodschappen te sturen. Daarom wordt de clausule doorgaans berekend op basis van de volledige inhoud van de boodschap, zodat deze verschilt naargelang de boodschap. Hiervoor wordt een zogenaamde **hash function** gebruikt die, ongeacht de lengte van de boodschap, een controlecode, de **hash sum** of **checksum** van een vaste lengte berekent. Hash functies vormen een studie op zich, maar in het volgende voorbeeld wordt een naïeve methode geïllustreerd.

Voorbeeld 6.2. In voorbeeld dat in de vorige sectie werd uitgewerkt, zou bijkomend kunnen worden afgesproken om als controle de som van alle getallen $a_i, i = 1, \ldots, 8$ modulo n te berekenen, te encrypteren met f_S en toe te voegen aan de boodschap als digitale handtekening.

De som $\sum_{i=1}^{8} a_i$ geeft modulo $n = 69647$ een bijkomend getal

$$a_9 = (54219)_{10} = (1101001111001011)_2.$$

Gebruikt de zender als private sleutel $P = 457$, $Q = 157$ en $S = 30487$, als publieke sleutel $N = 71749$ en $R = 7$, dan handtekent hij de hash sum a_9 als $f_S(54219) = 54219^{30487} \mod 71749$ of

$$f_S(a_9) = (49761)_{10} = (1100001001100001)_2.$$

Wordt de boodschap dan voor verzending geëncrypteerd, dan wordt op deze controlesom nog eens de functie f_r toegepast, $f_r(49761) = 49761^{53} \mod 69647$ of

$$f_r\big(f_S(a_9)\big) = (14947)_{10} = (11101001100011)_2$$

Dit getal telt slechts veertien bits, zodat het met drie leidende nullen aan de boodschap in formule 6.1 wordt toegevoegd: 0 0 0 1 1 1 0 1 0 0 1 1 0 0 0 1 1.

De ontvanger past de decrypteerfunctie f_s toe op de boodschap, controlebits inclusief. Deze laatste worden daardoor $f_s(14947) = 14947^{43037} \mod 69647$ of

$$f_s\Big(f_r\big(f_S(a_9)\big)\Big) = (49761)_{10} = (1100001001100001)_2.$$

Om de oorspronkelijke checksum te vinden, zoekt de ontvanger de publieke sleutel van de zender, en past hij f_R toe, $f_R(49761) = 49761^{7} \mod 71749$ of

$$f_R\Big(f_s\Big(f_r\big(f_S(a_9)\big)\Big)\Big) = (54219)_{10} = (1101001111001011)_2.$$

Als finale proef op de som berekent de ontvanger zelf de hash functie van de gedecrypteerde boodschap. Komt deze overeen met $()_{10} = ()_2$, dan is de identiteit van de zender bevestigd.

67 Vervolledig het schema in tabel 6.4 zodat het ook de werking van de digitale handtekening bevat.

6. Enkele historische noten

Ronald L. Rivest werd geboren in 1947 in Schenectady, NY, USA. Hij werd Bachelor of Science in Wiskunde in 1969 aan Yale University en behaalde een doctoraat in computerwetenschappen in 1974 aan Stanford University. Hij is nu professor computerwetenschappen aan M.I.T. (Massachusetts Institute of Technology), waar hij voorzitter is van de "Cryptography and Information Security Group". Hij verricht onderzoek in cryptografie, netwerk- en computerbeveiliging en algoritmen. Hij ontwierp naast het RSA-algoritme, samen met Shamir en Adleman in 1977, ook de algoritmes voor symmetrische sleutel-encryptie RC2, RC4, RC5 en was co-uitvinder van RC6 (RC staat voor 'Rivest cypher' of 'Ron's code'). Hij is tevens lid van de National Academy of Engineering en van de National Academy of Sciences. Samen met Shamir en Adleman kreeg hij in 2000 de IEEE Koji Kobayashi Computers and Communications Award en in 2002 de ACM Turing Award (genaamd naar de beroemde Engelse Wiskundige Allan Turing, die er in de tweede wereldoorlog in slaagde de nazi-code Enigma te kraken).

Adi Shamir werd geboren in 1952 in Tel Aviv, Israel. Hij werd Bachelor of Science in Wiskunde aan de Tel-Aviv University in 1973. Hij werd Master of Science en doctor in de computerwetenschappen in resp. 1975 en 1977 aan het Weizmann Institute in Rehovor, Israel, waar hij sedert 1984 professor Wiskunde is en computerwetenschappen doceert. Tussendoor verrichtte hij van 1977 tot 1980 onderzoek aan M.I.T. Naast RSA is hij ook bekend voor het breken van het Merkle-Hellman systeem en verrichtte hij onderzoek in visuele cryptografie. Hij ontdekte, samen met Eli Biham, de zogenaamde differentiaal-crypto-analyse (later bleek dat dit al ontdekt was maar geheim gehouden werd door zowel I.B.M. als N.S.A.). Hij is verreweg de meest eminente specialist in crypto-analyse.

Leonard (Len) Adleman werd geboren in 1945 in Tel Aviv, Israel. Hij werd Bachelor of Science en doctor in de Wiskunde aan de University of California, Berkeley in respectievelijk 1968 en 1976. Naast RSA is hij ook gekend voor het berekenen van DNA en was hij actief in Grafentheorie. In zijn oplossing voor een onderdeel van het Hamiltoniaanse graaf probleem werd voor de eerste keer DNA gebruikt om een algoritme te berekenen. Het berekenen van DNA blijkt geschikt te zijn voor het oplossen van verschillende combinatorische problemen.

Hij zou ook de eerste geweest zijn om de term 'virus' te gebruiken in de context van computervirussen.

Als leuke bijkomstigheid willen we nog vermelden dat hij de wiskundige consulent was bij de film 'Sneakers' van Larry Lasker.

Hoofdstuk 7

RSA in Maxima

Ter illustratie van alle eigenschappen en algoritmes waarop RSA steunt, worden in dit hoofdstuk een aantal voorbeelden uitgewerkt aan de hand van het wiskundepakket `Maxima`. Dit vrij beschikbare computerprogramma is in eerste plaats ontwikkeld om aan symbolisch rekenen te doen: het is in staat om te rekenen met uitdrukkingen als $\sqrt{2}$ of π, waar een rekenmachine enkel kan rekenen met 1.42 en 3.14 – getallen met eindige precisie. Maar niet alleen dat, `Maxima` kan ook stelsels oplossen, functies afleiden en integreren, de mogelijkheden zijn schier eindeloos. In het eerste onderdeel wordt de algemene werking van `Maxima` uiteengezet, waarbij de commando's vooral de mogelijkheden van `Maxima` willen illustreren. Na dit inleidende deel worden bij elk eerder hoofdstuk een aantal voorbeelden uitgewerkt en oefeningen aangereikt.

0. Werking van Maxima

Installatie van `Maxima`. Het programma kan samen met de grafische interface `wxMaxima` vrij worden gedownload van `wxMaxima.sourceforge.net`, zowel voor Microsoft Windows, Mac OS X als voor Linux. De rest van deze tekst behandelt de Windowsversie, al werkt het programma identiek onder de andere besturingssytemen.

Na downloaden kan het bestand `Maxima-5.xx.x.exe` worden uitgevoerd en zal het programma automatisch worden geïnstalleerd door de instructies te volgen. Klik op `wxMaxima` in de programmalijst om het programma te starten.

De opdrachtprompt. Bij opstarten van `Maxima` verschijnt een leeg werkblad waarin rechtstreeks commando's kunnen worden getypt. Na het intypen van "1+1" verschijnt automatisch

```
--> 1+1
```

Druk op `shift+enter` om dit commando uit te voeren. Er verschijnt nu

```
(%i1) 1+1;
(%o1) 2
```

Hierin betekent `(%i1)` dat deze lijn de eerste invoerlijn is (input), waarvan het antwoord (output) op de lijn `(%o1)` staat. In principe wordt een invoerlijn afgesloten met een puntkomma, die door `Maxima` automatisch wordt gezet. Een vorige in- of uitvoerlijn kan worden gerecycleerd door het lijnnummer over te typen. De vorige berekening kan steeds worden opgeroepen met `%`.

```
(%i2) %o1+1;
(%o2) 3

(%i3) %+1;
(%o3) 4
```

Basisbewerkingen. Som `+`, verschil `-`, product `*`, deling `/` en machtsverheffing `^` gebeuren met de gebruikelijke symbolen. Voor de meeste andere gangbare bewerkingen bestaan ingebouwde functies, zie tabel 7.1. Ingebouwde constanten worden steeds vooraf gegaan door een procent-teken. `Maxima` berekent standaard geen numerieke resultaten, maar geeft waar mogelijk de exacte symbolische vorm. De functie `float()` laat toe om toch een welbepaalde getalwaarde te berekenen waar nodig.

```
(%i4) 3/2;
(%o4) 3/2
(%i5) sin(%pi/4);
(%o5) 1/sqrt(2)
(%i6) %e;
(%o6) %e
(%i7) float(%e);
(%o7) 2.718281828459045
(%i8) %e^(%i*%pi)+1;
(%o8) 0
```

De snelste manier om informatie te krijgen over een ingebouwde `Maxima`-functie is via `?`, meer zoekmogelijkheden zijn er binnen het menu `Help`. Zo blijkt dat de functie `log()` (zoals in de meeste computerpakketten) het natuurlijk logaritme is.

```
(%i9) ?  log;
-- Function:  log (<x>)
Represents the natural (base e) logarithm of <x>.
Maxima does not have a built-in function for the base 10 logarithm
or other bases.  'log10(x) := log(x) / log(10)' is a useful
definition.
[...]
```

Namen en variabelen. `Maxima` kan algebraïsch rekenen met onbekenden. Volgende voorbeelden gebruiken de commando's `expand()` en `factor()` om uitdrukkingen uit te werken respectievelijk te ontbinden in factoren.

```
(%i10) expand((a+b)^2);
(%o10) b^2+2*a*b+a^2
(%i11) factor(x^3-1);
(%o11) (x-1)*(x^2+x+1)
```

Met een **toekenning** ":" kan het resultaat van een bewerking worden opgeslagen onder een welbepaalde naam, waarna `Maxima` deze naam overal interpreteert als de toegekende waarde of uitdrukking. Een toekenning wordt opgeheven met `kill()`.

```
(%i12) a:2;
(%o12) 2
(%i13) a:a+1;
(%o13) a : 3
```

Tabel 7.1: Ingebouwde functies, constanten en operatoren in `Maxima`

Absolute waarde	`abs()`
Vierkantswortel	`sqrt()`
Natuurlijk logaritme	`log()`
Exponentiële functie	`exp()`
Goniometrische functies	`cos()`, `sin()`, `cot()`, `sec()`, `csc()`, `tan()`
Cyclometrische functies	`acos()`, `asin()`, `acot()`, `asec()`, `acsc()`, `atan()`
Hyperbolische functies	`cosh()`, `sinh()`, `coth()`, `sech()`, `csch()`, `tanh()`
Inverse hyperbolische functies	`acosh()`, `asinh()`, `acoth()`, `asech()`, `acsch()`, `atanh()`
Getal van Euler	`%e`
Het getal π	`%pi`
Imaginaire eenheid	`%i`
Gulden snede	`%phi`
Oneindig	`%inf`
Is gelijk aan	`=`
Is niet gelijk aan	`#`
Kleiner dan	`<`
Groter dan	`>`
Kleiner dan of gelijk aan	`<=`
Groter dan of gelijk aan	`>=`

```
(%i14) trigexpand(sin(a*x));
```
(%o14) $\cos(x)^2\sin(x)-\sin(x)^3$

```
(%i15) kill(a);
```
(%o15) *done*

```
(%i16) solve(a*x^2+b*x+c,x);
```
(%o16) $[x=-\frac{\sqrt{b^2-4ac}+b}{2a},x=\frac{\sqrt{b^2-4ac}-b}{2a}]$

Lijsten. Een belangrijke soort objecten die in deze tekst nog vaak aan bod zullen komen, zijn **lijsten**. Lijsten bestaan uit verschillende elementen die worden opgesomd tussen vierkante haakjes, gescheiden door komma's. Het `i`-e element uit een lijst `L` halen, kan met `L[i]`.

```
(%i17) K:[1,4,9,16,25];
```
(%o17) $[1,4,9,16,25]$

```
(%i18) K[3];
```
(%o18) 9

Met `makelist()` kan een lijst eenvoudig op basis van een voorschrift worden gegenereerd. Het commando `makelist()` heeft vier argumenten: het voorschrift (bijvoorbeeld `i^2`), een veranderlijke (hier `i`) en een begin- en eindwaarde (van 1 tot 5). `Maxima` zal dan het voorschrift uitrekenen voor alle natuurlijke waarden van de veranderlijke van de beginwaarde tot de eindwaarde en de uitkomsten bundelen in een lijst.

```
(%i19) makelist(i^2,i,1,5);
```
(%o19) $[1,4,9,16,25]$

Lijsten samenvoegen kan met `append()`. Dit lukt *niet* met afzonderlijke getallen: er is immers een essentieel verschil tussen `100` en `[100]`.

```
(%i20) L:makelist(i^2,i,6,9);
```
(%o20) $[36, 49, 64, 81]$

```
(%i21) K:append(K,L);
```
(%o21) $[1, 4, 9, 16, 25, 36, 49, 64, 81]$

```
(%i22) append(K,100);
```
append: argument must be a non-atomic expression; found 100

```
(%i23) append(K,[100]);
```
(%o23) $[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]$

Andere nuttige bewerkingen op lijsten zijn `length()`, dat het aantal elementen in een lijst telt, en `sum()`, dat uitdrukkingen van de vorm $\sum_{i=0}^{n} a_n$ uitrekent en de zelfde vier argumenten heeft als `makelist()`.

```
(%i24) length(K);
```
(%o24) 9

```
(%i25) sum(K[i],i,1,9);
```
(%o25) 285

De elementen van een lijst zijn niet noodzakelijk getallen, maar kunnen ook bijvoorbeeld expressies, vergelijkingen of andere lijsten zijn.

```
(%i26) M:[1,sin(x),a*x^2+b*x+c=0,K];
```
(%o26) $[1, \sin(x), ax^2 + bx + c = 0, [1, 4, 9, 16, 25, 36, 49, 64, 81]]$

```
(%i27) M[4];
```
(%o27) $[1, 4, 9, 16, 25, 36, 49, 64, 81]$

```
(%i28) M[4][6];
```
(%o28) 36

Functies en functie-analyse. Een functie als $f : \mathbb{R} \to \mathbb{R} : x \mapsto x^2$ kan in `Maxima` worden gedefinieerd met `f(x):=x^2`. Functies kunnen parameters bevatten, waaraan met `assume()` voorwaarden kunnen worden opgelegd.

```
(%i29) f(x):=sin(x)/x;
```
(%o29) $f(x) := \frac{\sin(x)}{x}$

```
(%i30) f(%pi/2);
```
(%o30) $\frac{2}{\pi}$

```
(%i31) g(x):=abs(x)/x;
```
(%o31) $g(x) := \frac{|x|}{x}$

```
(%i32) g(b);
```
(%o32) $\frac{|b|}{b}$

```
(%i33) assume(b<0);
```
(%o33) $[b < 0]$

```
(%i34) g(b);
```
(%o34) -1

```
(%i35) h(x):=a/x;
```
(%o35) $h(x) := \frac{a}{x}$

```
(%i36) h(a^n);
```
(%o36) $h(x) := a^{1-n}$

De meest gangbare bewerkingen met functies zijn ingebouwd. Zo berekent `Maxima` probleemloos limieten, afgeleiden, integralen, Taylorontwikkelingen en nog veel meer. Merk op dat *und* staat voor "niet gedefinieerd", *infinity* voor "plus of min oneindig" en ∞ voor "plus oneindig".

```
(%i37) limit(f(x),x,0);
```
(%o37) 1

```
(%i38) limit(f(x),x,inf);
```
(%o38) 0

```
(%i39) limit(g(x),x,0);
```
(%o39) *und*

```
(%i40) limit(g(x),x,0,minus);
```
(%o40) -1

```
(%i41) limit(g(x),x,0,plus);
```
(%o41) 1

```
(%i42) limit(h(x),x,0,plus);
```
(%o42) $infinity$

```
(%i43) assume(a<0);
```
(%o43) $[a < 0]$

```
(%i44) limit(h(x),x,0,plus);
```
(%o44) $-\infty$

```
(%i45) diff(f(x),x);
```
(%o45) $\frac{\cos(x)}{x} - \frac{\sin(x)}{x^2}$

```
(%i46) diff(f(x),x,2);
```
(%o46) $-\frac{\sin(x)}{x} + \frac{2\sin(x)}{x^3} - \frac{2\cos(x)}{x^2}$

```
(%i47) diff(exp(y(x)),x);
```
(%o47) $e^{y(x)}\left(\frac{d}{dx}y\left(x\right)\right)$

```
(%i48) integrate(x^2,x);
```
(%o48) $\frac{x^3}{3}$

```
(%i49) integrate(x^2,x,0,1);
```
(%o49) $\frac{1}{3}$

```
(%i50) integrate(h(x),x);
```
(%o50) $a\log\left(x\right)$

```
(%i51) integrate(h(x),x,1,%e);
```
(%o51) a

```
(%i52) taylor(exp(x),x,0,5);
```
(%o52) $1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \ldots$

Grafieken. Uiteraard kunnen functies ook grafisch worden voorgesteld. `Maxima` tekent trouwens niet enkel functies maar ook parameterkrommen en puntenplots. De syntax voor het tekenen van een Cartesische vergelijking $y = f(x)$ voor x van a tot b is `plot2d(f(x),[x,a,b])`. Door deze op te sommen in een lijst `[f(x),g(x),h(x),...]` kunnen meerdere functies tegelijk worden getekend. In de eerste voorbeelden verschijnt een waarschuwing dat minstens één punt niet kan worden getekend (welk?) maar de output is desalniettemin correct.

```
(%i53) plot2d(f(x),[x,-2*%pi,2*%pi]);
plot2d: expression evaluates to non-numeric value somewhere in plotting range.
(%o53)
```

```
(%i54) plot2d([f(x),sin(x)],[x,-2*%pi,2*%pi]);
plot2d: expression evaluates to non-numeric value somewhere in plotting range.
(%o54)
```

```
(%i55) plot2d(1/x^2,[x,-2,2]);
plot2d: expression evaluates to non-numeric value somewhere in plotting range.
(%o55)
```

Worden de functiewaarden te groot, dan kan de optie `[y,c,d]` worden toegevoegd. In dat geval verschijnt de waarschuwing dat niet alle punten van de grafiek worden getoond. Om een functie te kunnen plotten, mogen er natuurlijk *geen* parameters meer in staan, zoniet volgt de foutmelding dat er niets te plotten is.

```
(%i56) plot2d(1/x^2,[x,-2,2],[y,0,4]);
plot2d: expression evaluates to non-numeric value somewhere in plotting range.
plot2d: some values were clipped.
(%o56)
```

```
(%i57) plot2d(h(x),[x,-2,2]);
plot2d: expression evaluates to non-numeric value everywhere in plotting range.
plot2d: nothing to plot.
(%o57)
```

Voor een parametrische of puntenplot is de syntax telkens een beetje verschillend.

```
(%i58) plot2d([parametric,cos(t),sin(t),[t,0,2*%pi]]);
(%o58)
```

```
(%i59) plot2d([discrete,[1,2,3,4],[4,2,3,1]],[style,points]);
(%o59)
```

Matrices en stelsels. Vergelijkingen of stelsels van vergelijkingen oplossen kan met `solve()`. Als er meerdere oplossingen zijn, worden deze in een lijst opgesomd. Ook voor niet-lineaire stelsels geeft `Maxima` in de mate van het mogelijke oplossingen, eventueel met de waarschuwing dat niet alle oplossingen zijn gevonden.

```
(%i60) stelsel1:[3*x+5*y=4,2*x-6*y=5];
```
(%o60) $[5y + 3x = 4, 2x - 6y = 5]$

```
(%i61) solve(stelsel1,[x,y]);
```
(%o61) $[[x = \frac{7}{4}, y = -\frac{1}{4}]]$

```
(%i62) stelsel2:[x^2-y^2=1,x*y=x];
```
(%o62) $[x^2 - y^2 = 1, xy = x]$

```
(%i63) solve(stelsel2,[x,y]);
```
(%o63) $[[x = 0, y = -i], [x = 0, y = i], [x = -\sqrt{2}, y = 1], [x = \sqrt{2}, y = 1]]$

```
(%i64) %[1];
```
(%o64) $[x = 0, y = -i]$

```
(%i65) solve(cos(x)=1,x);
```
solve: using arc-trig functions to get a solution.
Some solutions will be lost.
(%o65) $[x = 0]$

Matrices ingeven gebeurt rij per rij. De optelling gebeurt met de klassieke "`+`" plus, de matrix-vermenigvuldiging kan met een "`.`" punt. Het element op positie `(i,j)` uit een matrix `A` halen kan met `A[i,j]`. Het commando `echelon()` voert Gauss-eliminatie uit.

```
(%i66) A:matrix([17,3],[-8,11]);
```
(%o66) $\begin{pmatrix} 17 & 3 \\ -8 & 11 \end{pmatrix}$

```
(%i67) B:matrix([4,-7],[-3,12]);
```
(%o67) $\begin{pmatrix} 4 & -7 \\ -3 & 12 \end{pmatrix}$

```
(%i68) A+B;
```
(%o68) $\begin{pmatrix} 21 & -4 \\ -11 & 23 \end{pmatrix}$

```
(%i69) A.B;
```
(%o69) $\begin{pmatrix} 59 & -83 \\ -65 & 188 \end{pmatrix}$

```
(%i70) B[1,1]:6;
```
(%o70) 6

```
(%i71) B;
```
(%o71) $\begin{pmatrix} 6 & -7 \\ -3 & 12 \end{pmatrix}$

```
(%i72) echelon(A);
```
(%o72) $\begin{pmatrix} 1 & \frac{3}{17} \\ 0 & 1 \end{pmatrix}$

Determinant, inverse matrix en karakteristieke polynoom berekenen, kan eenvoudig, maar de output bij berekening van eigenwaarden en eigenvectoren kan in eerste instantie wel een beetje vreemd lijken. De procedure `eigenvalues()` geeft als resultaat twee lijsten: de eerste bevat de eigenwaarden, de tweede de corresponderende multipliciteiten. Analoog geeft `eigenvectors()` drie lijsten: eigenwaarden, multipliciteiten en eigenvectoren. De uitvoer is hier dus een lijst van lijsten van lijsten.

```
(%i73) determinant(A);
```
(%o73) 211

```
(%i74) invert(A);
```
(%o74) $\begin{pmatrix} \frac{11}{211} & -\frac{3}{211} \\ \frac{8}{211} & \frac{17}{211} \end{pmatrix}$

```
(%i75) charpoly(A,x);
```
(%o75) $(11-x)\,(17-x)+24$

```
(%i76) eigenvalues(A);
```
(%o76) $[[14-\sqrt{15}\,i,\sqrt{15}\,i+14],[1,1]]$

```
(%i77) eigenvectors(A);
```
(%o77) $[[[14-\sqrt{15}\,i,\sqrt{15}\,i+14],[1,1]],[[[1,-\frac{\sqrt{15}\,i+3}{3}]],[[1,\frac{\sqrt{15}\,i-3}{3}]]]]$

1. Basisprincipes van geheimcodering

Procedures voor RSA. Deze en volgende secties gebruiken speciaal voor Junior College geschreven functies, gebundeld in een pakket. Dit pakket is te vinden in de leeromgeving, dient te worden gedownload naar de lokale PC en kan als volgt worden geopend:

1. Klik in `Maxima` op `File`,
2. Daarna op `Load Package...`,
3. Zoek het bestand `proceduresRSA.mac`,
4. Klik op `open`.

Dit hoort te resulteren in een lijn als volgt.

```
(%i78) load(".../proceduresRSA.mac")$;
```

In tabel 7.2 staat een overzicht van alle procedures in dit pakket.

Rotatiecodering. Met de hand een tekst coderen of decoderen is zeer tijdsintensief, zelfs voor een eenvoudige rotatiecodering. Daarom is rotatiecodering geïmplementeerd in dit pakket.

`rotatiecodering(getal,tekst)`	
Input	`getal`: een geheel getal `tekst`: een stuk tekst, tussen aanhalingstekens
Omschrijving	Deze procedure verschuift alle lettertekens in `tekst` over een aantal `getal` letters in het alfabet.
Voorbeeld	`(%i79) rotatiecodering(3,"alea iacta est");` (%o79) *dohd ldfwd hvw*

68 Hoe kan de procedure `rotatiecodering()` worden gebruikt om een tekst opnieuw te decoderen?

Tabel 7.2: Procedures in het pakket `proceduresRSA.mac`

Frequentie-analyse. Om de frequenties van alle letters in een tekst te bepalen, is er de procedure `frequentiegrafiek(tekst)`. In volgende voorbeelden zal voor de duidelijkheid de klaartekst telkens in kleine letters en het geheimschrift in hoofdletters geschreven worden. Met behulp van de procedure `ontcijfer()` kan een stuk geheimtekst dan letter voor letter worden ontcijferd.

`frequentieDiagram(tekst)`	
Input	`tekst`: een stuk tekst, tussen aanhalingstekens
Omschrijving	Deze procedure geeft als uitvoer een staafdiagram dat de frequentie van elke letter van het alfabet in `tekst` toont.
Voorbeeld	`(%i80) frequentieDiagram(nederlands);` `(%o80)`

`ontcijfer(sleutel,tekst)`	
Input	`sleutel`: de zesentwintig letters van het alfabet, tussen aanhalingstekens `tekst`: een stuk tekst, tussen aanhalingstekens
Omschrijving	Deze procedure verandert hoofdletters uit de invoer `tekst` in kleine letters als volgt: de i-e letter uit het alfabet wordt vervangen door de i-e letter uit `sleutel`. Kleine letters worden ongemoeid gelaten.

Voorbeeld	Volgend voorbeeld ontcijfert `"UTITOD"`. Eerst wordt de letter T ontmaskerd als e, daarna de U als g en tot slot worden de andere letters gevonden.
	`(%i81) ontcijfer("ABCDEFGHIJKLMNOPQRSeUVWXYZ","UTITOD");`
	`(%o81)` *UeIeOD*
	`(%i82) ontcijfer("ABCDEFGHIJKLMNOPQRSegVWXYZ",%);`
	`(%o82)` *geIeOD*
	`(%i83) ontcijfer("ABCmEFGHhJKLMNiPQRSegVWXYZ",%);`
	`(%o83)` *geheim*

69 Het pakket `proceduresRSA.mac` bevat naast een resem procedures ook twee langere gecodeerde teksten. De eerste tekst (`nederlands`) bevat een Nederlandstalige tekst, de tweede (`engels`) een Engelstalige.

```
(%i84) nederlands;
```
(%o84) $SSG\ UJQQXWCQENQT\ IQG\ WS\ XNSHMSGJCS\ldots$

```
(%i85) engels;
```
(%o85) $M\ NMPIRUK\ UX\ KAC\ XPCOJCYBQCL\ UX\ldots$

Identificeer eerst de meest voorkomende letters in deze teksten met `frequentieDiagram()`. In een Engelstalige tekst komen na de e, de letters a en t het meeste voor.

Gebruik vervolgens de procedure `ontcijfer()` om op basis van deze frequentie-analyse de geheimtekst letter voor letter te ontrafelen. Voor de duidelijkheid is het nuttig in de sleutel hoofdletters te gebruiken voor letters die nog niet ontcijferd zijn en kleine letters te gebruiken waar de code al is ontcijferd, zoals in de voorbeelden hierboven.

Binaire getallen. In het pakket `proceduresRSA.mac` zijn twee procedures voorhanden om decimale getallen om te zetten in binaire en omgekeerd.

`decimaalNaarBinair(getal)`	
Input	`getal`: een natuurlijk getal in decimale notatie
Omschrijving	Deze procedure heeft als output een lijst van bits, die samen de binaire expansie van `getal` vormen.
Voorbeeld	`(%i86) decimaalNaarBinair(2011);` (%o86) $[0,1,1,1,1,1,0,1,1,0,1,1]$

`binairNaarDecimaal(lijst)`	
Input	`lijst`: een lijst van bits (0 of 1)
Omschrijving	Deze procedure interpreteert `lijst` als een getal in binaire schrijfwijze en geeft als uitvoer datzelfde getal in decimale notatie.
Voorbeeld	`(%i87) binairNaarDecimaal([0,1,1,1,1,1,0,1,1,0,1,1]);` (%o87) 2011

70 Ter illustratie is de code van de procedure `decimaalNaarBinair(getal)` opgenomen in tabel 7.3. Verifieer de werking van dit programma door onderstaande tabel in te vullen voor invoer `decimaalNaarBinair(2011)`. Gebruik de ingebouwde help-functie `?` om te achterhalen wat de gebruikte commando's precies doen.

Stap	1	2	3	4	5	6	7	8	9	10	11
`quotient`											
`rest`											

Zoekopdracht. Zoek informatie over volgende onderwerpen.

1. Het fysiek verbergen van boodschappen, **steganografie**, is met elektronische communicatie nog steeds mogelijk. Welke technieken kunnen daarvoor worden gebruikt?

Tabel 7.3: Programmacode voor `decimaalNaarBinair()`

```
decimaalNaarBinair(getal) := block(
   [deling:euclidischeDeling(getal,2),quotient,rest,binair],
   quotient:deling[1],
   rest:deling[2],
   binair:[rest],
   while quotient>0 do (
      deling:euclidischeDeling(quotient,2),
      quotient:deling[1],
      rest:deling[2],
      binair:append([rest],binair)),
   binair
)$
```

2. De frequentietabel van letters in anderstalige teksten ziet er anders uit dan die in het Nederlands. Hoe ziet de verdeling van de letters er uit in het Engels, Frans, Duits, ... en welke zijn de meest voorkomende letters in deze talen?

3. In de tekst worden polyalfabetische, polygrafische en homofone substitutiecijfers vermeld, deze technieken zijn niet in het pakket geïmplementeerd. Op het internet is hierover wel veel informatie te vinden en ook tools voor codering, decodering en kraken van deze codes. Het is in het bijzonder zeer interessant om na te gaan hoe volgende historische cijfers werken – en gekraakt werden...

   - De **Vigenère**-code,
   - het **Playfair**-cijfer,
   - le **Grand Chiffre**,
   - de **Enigma** codeermachine.

2. Delers en deelbaarheid

Euclidische deling. De procedure `euclidischeDeling(deeltal,deler)` geeft een lijst met daarin quotiënt en rest terug. De grootste gemene deler (*greatest common divisor*) is standaard ingebouwd in `Maxima` en kan berekend worden met `gcd()`, het kleinste gemeen veelvoud (*least common multiple*) met `lcm()`.

`euclidischeDeling(n,m)`	
Input	`n`: een geheel getal, het deeltal `m`: een geheel getal, de deler
Omschrijving	Deze procedure heeft als uitvoer een lijst met twee getallen $[q, r]$ waarvoor $n = q \cdot m + r$ en $0 \leq r < \lvert d \rvert$.
Voorbeeld	`(%i88) euclidischeDeling(385,26);` `(%o88)` $[14, 21]$

```
(%i89) %[1]*26+%[2];
(%o89) 385

(%i90) gcd(12,15);
(%o90) 3

(%i91) lcm(12,15);
define: warning: redefining the built-in function lcm
(%o91) 60
```

Algoritme van Euclides. De implementatie van het uitgebreide algoritme van Euclides in tabel 7.4 geeft als uitvoer een lijst met drie getallen `[ggd(m,n),a,b]` waarvoor `ggd(m,n)=a*m+b*n`.

`uitgebreidAlgoritmeEuclides(m,n)`	
Input	`m`: een geheel getal `n`: een geheel getal
Omschrijving	Deze procedure heeft als uitvoer een lijst met drie getallen `[ggd(m,n),a,b]` waarvoor `ggd(m,n)=a*m+b*n`.
Voorbeeld	`(%i92) uitgebreidAlgoritmeEuclides(98,34);` `(%o92)` $[2, 8, -23]$

71 Test de werking van dit programma bij een input $A = 4864$ en $B = 3458$ door onderstaande tabel in te vullen. Gebruik de procedure `euclidischeDeling()` voor het berekenen van de tussenstappen.

stap	q	r	r0	r1	a	a0	a1	b	b0	b1
0			4864	3458		1	0		0	1
1										
2										
3										
4										
5										
6										

3. Priemgetallen en priemfactorisatie

Primaliteit. `Maxima` kent de ingebouwde functie `primep()` die van een gegeven getal nagaat of het priem is, `prev_prime()` en `next_prime()` die het vorige en volgende priemgetal opsporen en `ifactors` die de priemontbinding van een getal berekent als een lijst van koppels, waarbij elk koppel een priemfactor en corresponderende exponent bevat. Het programma `factor()` geeft de priemontbinding in een aanschouwelijkere vorm. Deze methoden blijken vrij snel voor kleinere getallen, maar kunnen `Maxima` doen blokkeren voor zeer grote getallen. In het geval een berekening te lang duurt, kan op de rode knop in de knoppenbalk worden geklikt om de berekening af te breken.

```
(%i93) primep(2011);
(%o93) true
```

Tabel 7.4: Programmacode voor `uitgebreidAlgoritmeEuclides()`

```
uitgebreidAlgoritmeEuclides(m,n) := block(
   [r0,r1,r,q,a0:1,a1:0,b0:0,b1:1,a,b],
   if abs(m)>abs(n)
      then (r0:m, r1:n)
      else (r0:n, r1:m),
   while r1#0 do (
      [q,r]:euclidischeDeling(r0,r1),
      r0:r1, r1:r,
      a:a0-q*a1,a0:a1,a1:a,
      b:b0-q*b1,b0:b1,b1:b),
   if abs(m)>abs(n)
      then ([r0,a0,b0])
      else ([r0,b0,a0])
)$
```

```
(%i94) next_prime(2011);
(%o94) 2017
(%i95) prev_prime(2011);
(%o95) 2003
(%i96) ifactors(10!);
(%o96) [[2, 8], [3, 4], [5, 2], [7, 1]]
(%i97) factor(10!);
(%o97) 2^8 3^4 5^2 7
```

Complexiteit. Om wat simulaties te kunnen uitvoeren, is ook een procedure voorzien die een willekeurig priemgetal genereert.

`randomPriem(n)`	
Input	`n`: een natuurlijk getal
Omschrijving	Deze procedure berekent een willekeurig priemgetal met n decimalen. De uitvoer is dus telkens anders.
Voorbeeld	`(%i98) randomPriem(10);` `(%o98)` 9547546447

72 Om een sleutel voor RSA samen te stellen, is het nodig grote priemgetallen te vinden.

- Zoek priemgetallen van verschillende grootte-ordes. Hoeveel decimalen tellen de priemgetallen die `Maxima` nog vlot weet te vinden? Voor hoeveel decimalen gaat dit niet meer?
- Ga na uit hoeveel tekens de binaire expansie van $n = p \cdot q$ bestaat. Hoe lang duurt het om n opnieuw te factoriseren? Doe dit opnieuw voor verschillende grootte-ordes van p en q. Vanaf welke grootte-ordes kan `Maxima` n niet meer factoriseren?

Onthou dat als een berekening te lang duurt, deze kan worden onderbroken met de rode stop-knop (*Interrupt current computation*).

Zoekopdracht. Het vinden van extreem grote priemgetallen is niet zomaar een aardigheidje, vandaar volgende zoekopdracht.

1. Wie was Pierre de **Fermat**? Hoe zijn Fermat-getallen gedefinieerd? Bereken de eerste vijf Fermat-getallen. Zijn dit priemgetallen? Welk is het kleinste Fermat-getal dat geen priemgetal is? Wanneer werd dit ontdekt? Wanneer werden de volgende Fermat-getallen die geen priemgetallen zijn, ontdekt? Wat is de stand van zaken op dit ogenblik? Wat is de *Fermat search*?

2. Wat zijn **tweelingspriemgetallen**? Geef enkele voorbeelden. Wat is de *twin prime conjecture*?

3. Wie was Marin **Mersenne**? Wat zijn Mersenne-getallen? Ga na of $M_2, M_3, M_5, M_7, M_{11}$ en M_{13} priemgetallen zijn. Waarvoor staat GIMPS en wat behelst dit? Welk is het grootste op dit ogenblik gekende priemgetal? Wanneer werd het ontdekt en hoeveel decimalen telt het?

4. Rekenen met restklassen

Modulorekenen. Het commando `euclidischeDeling()` laat toe de rest bij Euclidische deling te vinden, maar voor modulorekenen kan in het algemeen kortweg het ingebouwde `mod()` gebruikt worden.

```
(%i99) mod(243,16);
(%o99) 3
```

Cayleytabellen. Voor grotere ordes komt een procedure voor het berekenen van Cayleytabellen van pas.

`Cayleytabel(bewerking,n)`	
Input	`bewerking`: het woord `plus` of `maal`, `n`: een natuurlijk getal
Omschrijving	Deze procedure berekent de Cayleytabel van $\mathbb{Z}_n, +$ of $\mathbb{Z}_n, \cdot$.

Voorbeeld	`(%i100) Cayleytabel(plus,3);`
	`(%o100)` $\begin{pmatrix} plus & 0 & 1 & 2 \\ 0 & 0 & 1 & 2 \\ 1 & 1 & 2 & 0 \\ 2 & 2 & 0 & 1 \end{pmatrix}$
	`(%i101) Cayleytabel(plus,3);`
	`(%o101)` $\begin{pmatrix} maal & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 0 & 2 & 1 \end{pmatrix}$

Inverse elementen. Het vervolg van de tekst maakt onderscheid tussen nuldelers en inverteerbare elementen in $Z_n \setminus \{0\}$:

- Een element $a \in Z_n \setminus \{0\}$ is een echte nuldeler als en slechts als er een element $b \in Z_n \setminus \{0\}$ bestaat waarvoor $a \cdot b = 0$.
- Een element $a \in Z_n \setminus \{0\}$ is inverteerbaar als en slechts als er een element $b \in Z_n \setminus \{0\}$ bestaat waarvoor $a \cdot b = 1$.

Vandaar volgende verkennende oefeningen.

73 Genereer Cayleytabellen voor de vermenigvuldiging voor enkele verschillende ordes n. In welk van deze Cayleytabellen heeft elk element van $\mathbb{Z}_n \setminus \{0\}$ een invers? In welk van deze Cayleytabellen komen nuldelers voor?

5. Modulorekenen en priemgetallen

Machten modulo. Omdat verderop vaak machten modulo een getal zullen moeten worden berekend, is de methode via binaire expansie van de exponent geïmplementeerd. De code van deze procedure `machtenModulo()` is te zien in tabel 7.5.

`machtenModulo(a,r,n)`	
Input	`a`: een natuurlijk getal, het grondtal, `r`: een natuurlijk getal, de exponent, `n`: een natuurlijk getal, de orde.
Omschrijving	Deze procedure berekent `a` tot de macht `r` modulo `n` via de binaire expansie van `r`.
Voorbeeld	`(%i102) machtenModulo(3,55,7);` `(%o102) 3`

Tabel 7.5: Programmacode voor `machtenModulo()`

```
machtenModulo(a,r,n) := block(
   [b,c:a,d,k,m:1],
   b:decimaalNaarBinair(r),
   d:length(b),
   for k:1 thru d do (
      if b[d+1-k]=1
         then (m:mod(m*c,n)),
      c:mod(c*c,n)),
   m
)$
```

74 Bestudeer de werking van het programma `machtenModulo()` bij berekening van $3^{55} \mod 7$ door volgende tabel in te vullen.

k	b[d+1-k]	m	c
		1	3
1			
2			
3			
4			
5			
6			

Fermats priemgetaltest. Met de procedure `machtenModulo()` is het nu zeer eenvoudig om Fermats priemgetaltest uit te voeren.

75 Onderzoek of volgende getallen priem zijn met zelf gekozen grondtallen. Verifieer met `primep()`, dat meer geavanceerde priemgetaltests gebruikt.

- $2^{59} - 1$
- $2^{61} - 1$
- $2^{63} - 1$
- 975050851321

Aangezien de Carmichael-getallen het moeilijkst te ontmaskeren zijn als niet-priem, is het nuttig om te bekijken wat de kans is dat de Fermat-priemgetaltest voor zo een getal faalt. Daartoe is er de procedure `fermatTotaleTest()` die telt voor hoeveel getallen de test *niet* opgaat.

`fermatTotaleTest(n)`	
Input	`n`: een natuurlijk getal
Omschrijving	Deze procedure telt voor hoeveel getallen $1 < a < n$ de congruentie $a^{n-1} \equiv 1 \mod n$ *niet* geldt.
Voorbeeld	`(%i103) fermatTotaleTest(561);` `(%o103)` 240

76 Wat is de uitvoer van de procedure `fermatTotaleTest(n)` als `n` een priemgetal is?

77 Welke proportie getallen ontmaskert de Carmichael-getallen van maximaal vier decimalen: 561, 1105, 1729, 2465, 2821, 6601 en 8911? Ga met een aantal voorbeelden na wat deze proportie is voor getallen van maximaal vier cijfers die priem- noch Carmichael-getallen zijn.

Merk op dat deze procedure dus met zekerheid bepaalt of een getal priem is, maar gezien de benodigde rekentijd zeker geen geschikte test levert.

78 Voor hoeveel willekeurig gekozen getallen moet de Fermat-priemgetaltest uitgevoerd worden om toch 95% zekerheid te hebben ook Carmichael-getallen kleiner dan 10000 correct te identificeren als niet-priem?

6. Het RSA-codeersysteem

In het pakket `proceduresRSA.mac` zijn de procedures `RSAencrypteer()` en `RSAdecrypteer()` voorzien, die toelaten om een boodschap rechtstreeks te encrypteren en te decrypteren.

`RSAencrypteer(n,r,tekst)`	
Input	`n`: een natuurlijk getal, product van twee verschillende priemgetallen p en q, `r`: een natuurlijk getal, relatief priem met $(p-1)\cdot(q-1)$, `tekst`: de boodschap, tussen aanhalingstekens
Omschrijving	Deze procedure zet `tekst` om in ASCII en voert op de bekomen bitrij RSA-encryptie toe met de functie $f_r : \mathbb{Z}_n \to Z_n : a \mapsto a^r$. De output is een lijst van bits.
Voorbeeld	`(%i104) RSAencrypteer(69647,53,"Bewijs gevonden");` `(%o104)` $[0,0,1,1,0,0,0,1,\ldots,0,1,1,1,1,1,0,1]$

`RSAdecrypteer(p,q,r,lijst)`	
Input	`p`, `q`: twee verschillende priemgetallen, `r`: een natuurlijk getal, relatief priem met $(p-1)\cdot(q-1)$, `lijst`: een lijst van bits
Omschrijving	Deze procedure zet `lijst` om in ASCII na er eerst $f_s : \mathbb{Z}_n \to Z_n : a \mapsto a^s$ op toe te passen met $1 = r\cdot s + (p-1)\cdot(q-1)\cdot t$.
Voorbeeld	`(%i105) RSAdecrypteer(271,257,53,%);` `(%o105)` *Bewijs gevonden*

Deze procedures zijn zelf niets meer dan een opeenvolging van functies die gegeven een geschikte sleutelcombinatie de afzonderlijke stappen uit het RSA-systeem uitvoeren. De lezer kan als volgt zelf elke stap zetten.

Ontvanger zoekt geschikte sleutel. Kies twee verschillende priemgetallen en bereken het product.

```
(%i106) p:randomPriem(3);
(%o106) 271

(%i107) q:randomPriem(3);
(%o107) 257

(%i108) n:p*q;
(%o108) 69647
```

Deze priemgetallen laten 16 bits-encryptie toe:

```
(%i109) k:floor(float(log(n)/log(2)));
(%o109) 16

(%i110) factor((p-1)*(q-1));
(%o110) 2^9 · 3^3 · 5

(%i111) r:53;
(%o111) 53
```

Een geschikte exponent r moet relatief priem zijn met 2, 3 en 5, dus is $r = 53$ een goede keuze. De publieke sleutel bestaat uit de getallen

$$n = 69647 \quad \text{en} \quad r = 53.$$

Zender encrypteert de klaartekst. Eerst en vooral wordt de klaartekst omgezet naar ASCII,

```
(%i112) boodschap:tekstNaarDecimaal("Bewijs gevonden");
(%o112) [66, 101, 119, 105, 106, 115, 32, 103, 101, 118, 111, 110, 100, 101, 110]
```

Deze lijst van 15 getallen moet nu omgezet worden naar een rij van bits, zo dat elk symbool correspondeert met 8 bits.

`decimaalNaarBinairLijst(lijst,k)`	
Input	`lijst`: een lijst van natuurlijke getallen kleiner dan 2^8.
Omschrijving	Deze procedure zet een lijst getallen in decimale schrijfwijze om in een lijst van binaire getallen, zelf genoteerd als lijsten van `k` bits.
Voorbeeld	`(%i113) binair:decimaalNaarBinairLijst(boodschap);` `(%o113)` [[0, 1, 0, 0, 0, 0, 1, 0], [0, 1, 1, 0, 0, 1, 0, 1], . . . , [0, 1, 1, 0, 0, 1, 0, 1], [0, 1, 1, 0, 1, 1, 1, 0]]

Deze procedure past essentieel `decimaalNaarBinair()` toe op elk getal uit de lijst, maar voegt daar telkens leidende nullen aan toe zodat elke binaire expansie hier 8 bits telt. De boodschap is nu gecodeerd in $15 \cdot 8 = 120$ bits.

In de bekomen rij moeten de bits nu samengevoegd worden tot getallen $a \in \mathbb{Z}_n$, waarvoor 8 leidende nullen worden toegevoegd.

`rijLengte(lijst,k)`	
Input	`lijst`: een lijst van (lijsten van) bits, `k`: het natuurlijk getal, het aantal bits waarin de lijst moet worden verdeeld.
Omschrijving	Deze procedure verdeelt `lijst` in stukken van telkens `k` bits. Indien nodig worden vooraan in de lijst extra nullen toegevoegd. Indien `k` gelijk is aan 0 is het resultaat één lange lijst van bits.
Voorbeeld	`(%i114) kbits:rijLengte(binair,16);` `(%o114)` [[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0], . . . , [0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0]]

Op deze lijst van getallen in Z_n kan nu telkens de encryptie worden uitgevoerd met de procedure `machtenModuloLijst()`.

`machtenModuloLijst(lijst,r,n,bits)`	
Input	`lijst`: een lijst van binaire getallen, `r`: een natuurlijk getal, de exponent, `n`: een natuurlijk getal, de orde.
Omschrijving	Deze procedure berekent elk binair getal in `lijst` tot de macht `r` modulo `n`. Het argument `bits` bepaalt uit hoeveel bits de getallen in de resulterende lijst moeten bestaan.
Voorbeeld	`(%i115) machten:machtenModuloLijst(kbits,r,n,k+1);` `(%o115)` $[[0,0,1,1,0,0,0,1,0,0,0,1,1,0,1,0,1],$ $\ldots,$ $[0,0,1,1,1,0,0,0,0,0,1,1,1,1,1,0,1]]$

Deze binaire getallen worden als één lange rij van 136 bits doorgestuurd naar de ontvanger.

`(%i116) geheimtekst:rijLengte(machten,0);`
`(%o116)` $[0,0,1,1,0,0,0,1,0,0,0,1,1,0,1,0,\ldots,0,1,1,1,0,0,0,0,0,1,1,1,1,1,0,1]$

Ontvanger decrypteert de boodschap. De ontvanger gebruikt het uitgebreide algoritme van Euclides om uit de private sleutel de decryptiefunctie te bepalen:

`(%i117) uitgebreidAlgoritmeEuclides(r,(p-1)*(q-1));`
`(%o117)` $[1,-26083,20]$

Het bekomen getal is negatief, maar kan worden verhoogd met $(p-1)\cdot(q-1)$ tot een getal tussen 0 en $(p-1)\cdot(q-1)$ wordt bekomen.

`(%i118) s:%[2]+(p-1)*(q-1);`
`(%o118)` 43037

De geheimtekst dient eerst opnieuw te worden geïnterpreteerd als een lijst van 17-bits getallen uit $\mathbb{Z}_n$, waarna de decryptiefunctie f_{43037} kan worden toegepast.

`(%i119) rijLengte(geheimtekst,k+1);`
`(%o119)` $[[0,0,1,1,0,0,0,1,0,0,0,1,1,0,1,0,1],$
$\ldots,$
$[0,0,1,1,1,0,0,0,0,0,1,1,1,1,1,0,1]]$

`(%i120) ontcijferd:machtenModuloLijst(%,s,n,k);`
`(%o120)` $[[0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0],$
$\ldots,$
$[0,1,1,0,0,1,0,1,0,1,1,0,1,1,1,0]]$

Omzetting naar ASCII besluit de decodering.

`(%i121) rijLengte(ontcijferd,8);`
`(%o121)` $[[0,0,0,0,0,0,0,0],[0,1,0,0,0,0,1,0],$
$\ldots,$
$[0,1,1,0,0,1,0,1],[0,1,1,0,1,1,1,0]]$

`(%i122) binairNaarDecimaalLijst(%);`
`(%o122)` $[0,66,101,119,105,106,115,32,103,101,118,111,110,100,101,110]$

`(%i123) decimaalNaarTekst(%);`
`(%o123)` *Bewijs gevonden*

Tekst wordt onderschept. Als een derde de geheimtekst onderschept, deze interpreteert als ASCII en tracht te decoderen, dan bestaat het resultaat uit een willekeurige rij tekens, voornamelijk onleesbare ASCII-symbolen voorgesteld door "?".

```
(%i124) rijLengte(geheimtekst,8);
```
(%o124) $[[0,0,1,1,0,0,0,1],[0,0,0,1,1,0,1,0],$
$\ldots,$
$[0,1,1,1,0,0,0,0],[0,1,1,1,1,1,0,1]]$

```
(%i125) binairNaarDecimaalLijst(%);
```
(%o125) $[49,26,170,146,27,219,234,63,19,218,219,221,13,86,112,112,125]$

```
(%i126) decimaalNaarTekst(%);
```
(%o126) $1?????????????Vpp\}$

Als de onderschepper de geheimtekst eerst tracht de ontsleutelen met een willekeurig gekozen decryptiefunctie, zal het resultaat een analoge onleesbare tekst opleveren.

Sleutelset voor de digitale handtekening. Wil de zender een digitale handtekening toevoegen, dan kiest deze ook een sleutelset.

```
(%i127) P:randomPriem(3);
```
(%o127) 457

```
(%i128) Q:randomPriem(3);
```
(%o128) 157

```
(%i129) N:P*Q;
```
(%o129) 71749

Deze priemgetallen laten eveneens 16 bits-encryptie toe:

```
(%i130) K:floor(float(log(N)/log(2)));
```
(%o130) 16

```
(%i131) factor((P-1)*(Q-1));
```
(%o131) $2^5 \cdot 3^2 \cdot 13 \cdot 19$

```
(%i132) R:7;
```
(%o132) 7

Een geschikte exponent R moet relatief priem zijn met 2, 3, 13 en 19, dus is $R = 7$ een goede keuze. De publieke sleutel bestaat uit de getallen

$$N = 71749 \quad \text{en} \quad R = 7.$$

Bij de private sleutel hoort de functie $F_{30487} : \mathbb{Z}_N \to \mathbb{Z}_N : a \mapsto a^{30487}$:

```
(%i133) uitgebreidAlgoritmeEuclides(R,(P-1)*(Q-1));
```
(%o133) $[1, 30487, -3]$

Digitale handtekening toevoegen. De zender telt alle getallen uit de lijst `kbits` op en berekent de rest bij deling door n. Voor de aanschouwelijkheid worden alle berekeningen hier decimaal uitgevoerd.

```
(%i134) binairNaarDecimaalLijst(kbits);
(%o134) [66, 25975, 26986, 29472, 26469, 30319, 28260, 25966]

(%i135) sum(%[i],i,1,length(%));
(%o135) 193513

(%i136) mod(%,n);
(%o136) 54219
```

Dit getal is de controlesom die de zender bij wijze van handtekening encrypteert met zijn private sleutel.

```
(%i137) machtenModulo(%,S,N);
(%o137) 49761
```

Het resultaat wordt aan de boodschap toegevoegd en voor verzending versleuteld met de publieke sleutel van de ontvanger.

```
(%i138) handtekening:machtenModulo(%,r,n);
(%o138) 14947

(%i139) decimaalNaarBinair(%);
(%o139) [1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1]
```

Het is deze rij, met leidende nullen aangevuld tot 17 bits, die aan de geheimtekst wordt toegevoegd, zodat in totaal 153 bits worden verzonden.

Digitale handtekening controleren. De ontvanger decrypteert samen met de boodschap ook de digitale handtekening met de eigen private sleutel.

```
(%i140) machtenModulo(handtekening,s,n);
(%o140) 49761
```

Vervolgens decrypteert hij de controlesom met behulp van de publieke sleutel van de zender.

```
(%i141) controle1:machtenModulo(handtekening,R,N);
(%o141) 54219
```

Om tot slot de authenticiteit van de afzender te verifiëren, berekent de ontvanger zelf de controlesom op basis van de gedecrypteerde boodschap.

```
(%i142) binairNaarDecimaalLijst(ontcijferd);
(%o142) [66, 25975, 26986, 29472, 26469, 30319, 28260, 25966]

(%i143) sum(%[i],i,1,length(%));
(%o143) 193513

(%i144) controle2:mod(%,n);
(%o144) 54219

(%i145) is (controle1=controle2);
(%o145) true
```

79 Kies zelf twee priemgetallen p en q zodat $n = p \cdot q$ door `Maxima` niet meer gefactoriseerd kan worden (zie oefening 72). Zoek een geschikte r en ruil de bekomen publieke sleutel met iemand anders. Versleutel een zelf geschreven boodschap, voeg er een digitale handtekening aan toe en laat de ander de code ontcijferen en de authenticiteit van de boodschap nagaan.

Index