Kraak de Code

Koen Stulens

T³ EUROPE

Teachers Teaching with Technology™

KRAAK DE CODE

Koen Stulens
k-stulens@ti.com

CRYPTOGRAGIE STAMT VAN HET GRIEKS: 'CRYPTOS = GEHEIM' , 'GRAFEIN = SCHRIJVEN'.

Sinds mensen met elkaar communiceren is er steeds nood geweest om dit ook op een geheime manier te doen. De eenvoudigste manier om in het geheim te communiceren is gebruik te maken van een sleutel om te coderen en te decoderen. In dit geval moeten zowel de verzender als de ontvanger deze geheime sleutel kennen. Men spreekt dan van ***symmetrische*** cryptografie.

Sinds de ontwikkeling van computernetwerken en internet is er regelmatig communicatie die men wil beschermen t.o.v. de buitenwereld. Denk bijvoorbeeld aan het online shoppen of bankzaken doen via het web. Dit gebeurt zonder een vooraf afgesproken sleutel. In dit geval spreken we van een publieke sleutel en ***asymmetrische*** cryptografie. De ontvanger plaatst de publieke sleutel online of op een keyserver zodat die voor iedereen beschikbaar is. Hij codeert hiermee een boodschap en stuurt deze naar de ontvanger. De verzender kan de boodschap niet ontgrendelen. Alleen de ontvanger heeft de geheime sleutel om de boodschap te decoderen.

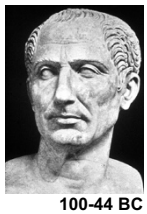
Grote priemgetallen spelen een zeer belangrijke rol in cryptografie.

Priemgetallen zijn de 'onbreekbare bouwstenen' van getallen. Euclides bewees dat elk natuurlijk getal op precies 1 manier kan geschreven worden als een product van priemgetallen alsook dat er geen grootste priemgetal bestaat.

Aantonen dat zeer grote getallen priemgetallen zijn, vraagt – ondanks snelle computers – heel wat rekenwerk. Het product van twee grote priemgetallen is zeer snel uitgevoerd maar omgekeerd is het ontbinden van een zeer groot getal een moeilijke opdracht.

We bespreken enkele systemen van coderen samen met de wiskundige principes erachter.

1. De Caesar-code


100-44 BC

Eén van de oudste coderingssystemen is genoemd naar Julius Caesar. Het systeem bestaat uit een verschuiving van het alfabet. Iedere letter wordt cyclisch verschoven over k plaatsen. Een voorbeeld, waarbij we iedere letter drie plaatsen naar rechts verschuiven om te coderen:

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

Indien we met deze code het woord “wiskunde” coderen bekomen we:

w	i	s	k	u	n	d	e	
↓	↓	↓	↓	↓	↓	↓	↓	+3
z	l	v	n	x	q	g	h	

Om in ons voorbeeld de code terug om te zetten in het origineel, verplaatsen we iedere letter 23 plaatsen naar rechts of drie plaatsen naar links.

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

z	l	v	n	x	q	g	h	
↓	↓	↓	↓	↓	↓	↓	↓	-3
w	i	s	k	u	n	d	e	

Dat de Caesar-code weinig veiligheid biedt, hoeft geen betoog, daar er maar 25 sleutels zijn. Men kan al deze sleutels één voor één uitproberen. Alle sleutels, codes, vind je in de onderstaande tabel.

0	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
2	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
3	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
4	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
5	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
6	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
7	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
8	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
9	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
10	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	l	j
11	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	l	j	k
12	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
13	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
14	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
15	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
16	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
17	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
18	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
19	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
20	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
21	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
22	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
23	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
24	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
25	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Om dit systeem te laten uitvoeren door een machine, associëren we met iedere letter als volgt een getal: a=0, b=1, c=2, x=23, y=24, z=25.

w	i	s	k	u	n	d	e	
↓	↓	↓	↓	↓	↓	↓	↓	
22	8	18	10	20	13	3	4	
↓	↓	↓	↓	↓	↓	↓	↓	+3
25	11	21	13	23	16	6	7	
↓	↓	↓	↓	↓	↓	↓	↓	
z	l	v	n	x	q	g	h	

Maar wat als we deze methode toepassen op:

x	y	z	
↓	↓	↓	
23	24	25	
↓	↓	↓	+3
26	27	28	

De getallen 26, 27 en 28 corresponderen niet met letters uit het alfabet. De getallen die we hier hebben zijn 0 (= a), 1 (= b) en 2 (= c).

2. Modulo-rekening

Het meest bekende voorbeeld van modulo-rekenen staat bekend als ‘klok-rekenen’.

Het is vrij normaal te denken dat bij een gehele deling het quotiënt belangrijker is dan de rest. Maar dit is niet altijd het geval.

Probeer je een situatie voor te stellen waar het zin heeft te zeggen dat 9 + 5 = 2.

Veronderstel dat het negen uur is en dat je gepland had om 5 uren te studeren.

Hoe laat zal de klok aangeven na deze 5 uren?

Nog enkele rekensommen op deze klok:

9 + 7 = 4 (= 16)	5 – 11 = 6 (= -6)
3 + 15 = 6 (= 6)	1 – 9 = 4 (= -8)
2 + 11 = 1 (= 1)	3 - 12 = 3 (= -9)

Merk op dat ieder resultaat gelijk is aan de rest bij deling van de som door 12.

We construeren de volgende rij: $3 \xrightarrow{+12} 15 \xrightarrow{+12} 27 \xrightarrow{+12} 39 \xrightarrow{+12} 51 \xrightarrow{+12} \ldots$

Voor al deze getallen geldt dat ze bij deling door 12 rest 3 hebben. Er geldt dat deze getallen aan mekaar gelijk zijn op een veelvoud van 12 na.

En het verschil van twee willekeurige termen uit de bovenstaande rij is deelbaar door 12.

We zeggen dat ze congruent zijn modulo 12. We noteren dit als volgt: $15 \equiv 27 \; mod \; 12$.

Terug naar het coderen. Om de Caesar-methode uit te voeren, gebruiken we 'klok-rekenen' a.d.h.v. een de klok met de getallen 0, 1, 2, 3, ..., 25. Hiermee is ons probleem van x, y, z opgelost.

x	y	z	
↓	↓	↓	
23	24	25	
↓	↓	↓	+3
26	27	28	
			modulo 26
0	1	2	
↓	↓	↓	
a	b	c	

Het begrip congruentie modulo n werd ingevoerd door de Duitse wiskundige Carl Friedrich Gauss. (1777-1855)


1777-1855

Voor $n \in \mathbb{N}$ met $n \geq 2$ definiëren we:

$$\forall\, a, b \in \mathbb{Z} : a \equiv b \bmod n$$
$$\Updownarrow$$
$$n \mid (a - b)$$

Indien $a \equiv b \bmod n$ geldt dat a en b dezelfde rest hebben bij deling door n.

Intermezzo: Toepassing bij bankrekeningnummers

De controle van IBAN-nummers, bijvoorbeeld bij een overschrijving, gebeurt door gebruik te maken van modulo-rekenen 97. Merk op dat 97 het grootste priemgetal is bestaande uit twee cijfers.

Voor het IBAN-nummer NL82 ABNA 0566 8410 88 is 82 het controle getal. De controle wordt als volgt uitgevoerd:

1. Verplaats de eerste vier karakters naar het einde:
   ABNA0566841088NL82
2. Vervang elke letter door 2 cijfers, waarbij A=10, B= 11, …, Z=35:
   101123100566841088232182
3. Bereken de rest bij deling door 97 (modulo 97):
   mod(101123100566841088232182,97) = 1
4. Als de restwaarde gelijk is aan 1, is het IBAN-nummer hoogst waarschijnlijk valide.

Wat is het controlegetal voor het IBAN-nummer NLxx INGB 0006 3390 95?

1. Plaats de landcode achter de rekeningindentificatie:
   INGB0006339095NL
2. Alle letters vervangen door 2 cijfers, waarbij A=10, B= 11, …, Z=35:
   182316110006339095232 1
3. Twee nullen toevoegen aan het einde:
   18231611000633909523210 0
4. Bereken de rest bij deling door 97 (modulo 97):
   mod(18231611000633909523210 0,97) = 78
5. Het controlegetal is 98 min deze rest:
   98 – 78 = 20

Het IBAN-nummer is NL20 INGB 0006 3390 95.

3. Het systeem van Vigenère

Deze coderingsmethode gebruikt meer dan één substitutiealfabet. Dit systeem werd in 1586 ontworpen door de Fransman Vigenère. Zijn idee was de originele tekst te coderen met verschillende Caesar-codes.

Deze methode maakt gebruik van een sleutelwoord bv. “wiskunde”. Als hulpmiddel noteren we onder het standaardalfabet alle verschuivingen die beginnen met een letter uit het sleutelwoord.

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

We coderen de tekst “kraakdecode” als volgt. We noteren het sleutelwoord zo vaak als nodig onder de boodschap. Voor k gebruiken we het substitutiealfabet beginnende met w, voor r met i, voor a met s, ...

k	r	a	a	k	d	e	c	o	d	e
w	i	s	k	u	n	d	e	w	i	s
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
g	z	s	k	e	q	h	g	k	l	w

Men heeft lang gedacht dat deze code niet te kraken was tot in 1863 de Pruisische legerofficier F.W. Kasiski een methode ontwikkelde om de lengte van het sleutelwoord te bepalen en zo de code te kraken.

Om het systeem van Vigenère uit te voeren met een machine maken we ook hier gebruik van modulo-rekenen modulo 26.

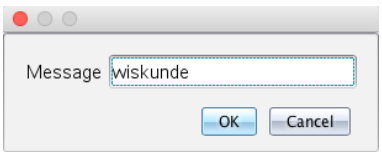
4. Coderen met TI-Nspire CX Technologie

4.1 Caesar-code

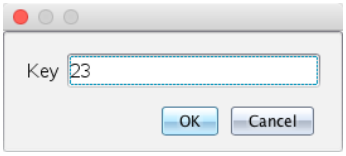
Ruwweg kunnen we het programma in drie onderverdelen: input, verwerking en output.

<u>Input</u>

- *Definitie van de String met de te gebruiken karakters*
  Een string, woord, is een rij van karakters ingesloten door aanhalingstekens:
  alfa:="abcdefghijklmnopqrstuvwxyz”
- *Input van de Boodschap*
  Het commando RequestStr laat toe een string in te geven en te bewaren in een variabele:
  RequestStr "Message",*message*



- *Input van de Sleutel*
  Het commando Request laat toe een getal in te geven en te bewaren in een variabele:
  Request "Key",*key*



Verwerking

- *We starten met een lege string voor de gecodeerde boodschap:*
  code:=" "
- *Coderen van de Boodschap*
  We coderen één voor één, ieder karakter van de boodschap - For *i*,1,dim(*message*)
  d.m.v. de volgende commando's:

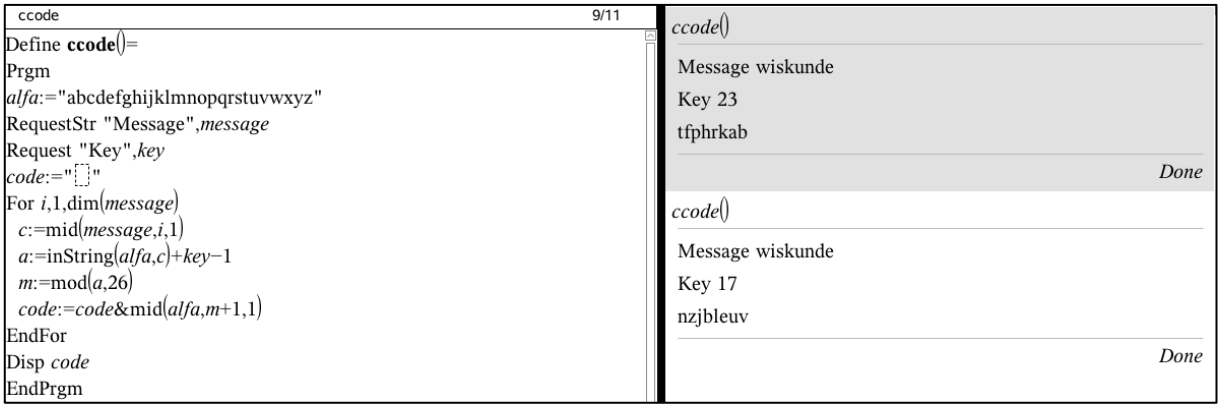
  - *c*:=**mid**(*message*,*i*,1) — i[e] karakter in de boodschap (*message*)
  - *a*:=**inString**(*alfa*,*c*) — positie van i[e] karakter in het alfabet (*alfa*)
  - **&** — samenvoegen van twee strings

  als volgt:

```
code:=" "
For i,1,dim(message)
  c:=mid(message,i,1)
  a:=inString(alfa,c)+key-1
  m:=mod(a,26)
  code:=code&mid(alfa,m+1,1)
EndFor
```

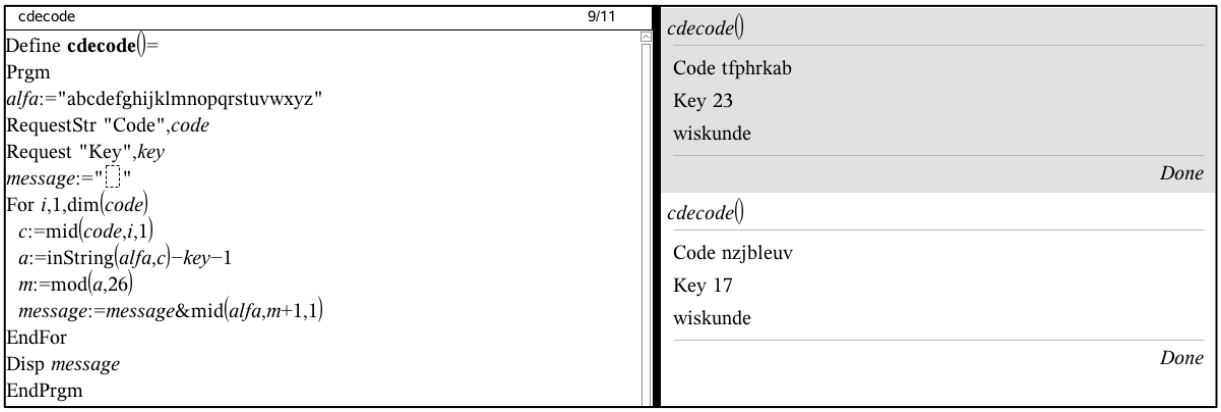
Output

- *Als laatste stap tonen we de gecodeerde boodschap met het commando:*
  Disp *code*



Noot
CTRL R intikken met het programmeervenster start de uitvoering van het programma.

Voor het decoderen kan een gelijkaardig programma gebruikt worden; met het verminderen van de key i.p.v. toevoegen.



Vanzelfsprekend kunnen er andere karakters toegevoegd aan de alfa-string. In dit geval dient vanzelfsprekend ook modulo 26 aangepast aan het aantal karakters van alfa:

b.v. alfa:="ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz@0123456789.-"

Na het invoeren van de boodschap kan ook vrij eenvoudig met een **If Then Else** statement een test ingevoegd worden om na te gaan of alle karakters van de boodschap (*message*) in het alfabet (*alfa*) voorkomen.

4.2 Het Vigenère-systeem

We beperken ons weer eerst tot letters en bekijken een voorbeeld, met als sleutel wis.

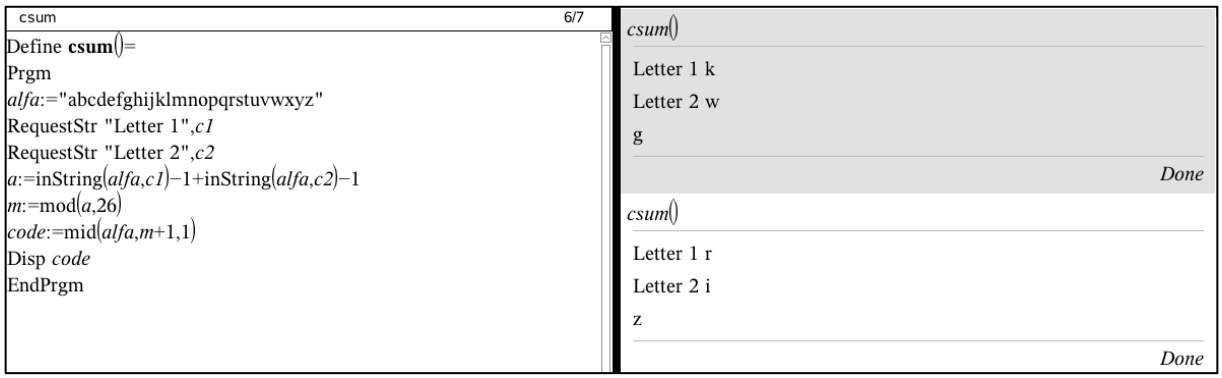
k	r	a	a	k	d	e	c	o	d	e
w	i	s	w	i	s	w	i	s	w	i
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
+22	+8	+18	+22	+8	+18	+22	+8	+18	+22	+8
g	z	s	w	s	v	e	k	g	z	m

Modulo 26 komt het uitvoeren van dit systeem in principe neer op het optellen van twee letters. Bijvoorbeeld k + w = g ($10 + 22 \equiv 6 \bmod 26$).

Het enige dat we extra nodig hebben is een systeem om de sleutel onder de boodschap te plaatsen.

a. <u>De som van twee letters</u>

Het optellen van twee letters kan als volgt:



b. <u>Het plaatsen van de sleutel onder de boodschap</u>

Het plaatsen van de letters van de sleutel onder de boodschap moet zo vaak als de boodschap lang is.

1	2	3	4	5	6	7	8	9	10	11
k	r	a	a	k	d	e	c	o	d	e
w	i	s	w	i	s	w	i	s	w	i
1	2	3	1	2	3	1	2	3	1	2

Hetgeen we zeker al nodig hebben is bv. in dit concreet geval een lus van 1 to 11. Algemeen wordt dit een lus van 1 tot dim(*message*).

We zoeken de volgende functie van {1,2,3,4,5,6,7,8,9,10,11} naar {1,2,3}:

1	2	3	4	5	6	7	8	9	10	11
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	2	3	1	2	3	1	2	3	1	2

Een functie die dit proces weergeeft is: $x \mapsto (x-1) \bmod 3 + 1$ met 3 = dim(*key*).

c. <u>Het coderen van de boodschap</u>

Voor het coderen van de boodschap tellen we overeenkomstige letters van de boodschap en de sleutel met mekaar op.

Als alfabet kiezen we:
alfa:=”ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz@ 0123456789.-“

```
vcode                                                    0/13
Define vcode()=
Prgm
alfa:="ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz@0123456789.−"
RequestStr "Message",message
RequestStr "Key",key
code:="□"
For i,1,dim(message)
  m:=mod(i−1,dim(key))
  c1:=mid(message,i,1)
  c2:=mid(key,m+1,1)
  a:=inString(alfa,c1)−1+inString(alfa,c2)−1
  m:=mod(a,66)
  code:=code&mid(alfa,m+1,1)
EndFor
Disp code
EndPrgm
```

```
vcode()
Message Kraak de Code
Key WIS
fzswsrzmrYwv@
Done
vcode()
Message Kraak de Code
Key T3 Nederland
ci@nC27J.cP6x
Done
```

```
vdecode                                                  11/13
Define vdecode()=
Prgm
alfa:="ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz@0123456789.−"
RequestStr "Code",code
RequestStr "Key",key
message:="□"
For i,1,dim(code)
  m:=mod(i−1,dim(key))
  c1:=mid(code,i,1)
  c2:=mid(key,m+1,1)
  a:=inString(alfa,c1)−1−(inString(alfa,c2)−1)
  m:=mod(a,66)
  message:=message&mid(alfa,m+1,1)
EndFor
Disp message
EndPrgm
```

```
vdecode()
Code fzswsrzmrYwv@
Key WIS
Kraak de Code
Done
vdecode()
Code ci@nC27J.cP6x
Key T3 Nederland
Kraak de Code
Done
```

5. RSA-code

Voorgaande cryptosystemen hebben een geheime sleutel nodig, door beide partijen gekend. Deze sleutel wordt gebruikt voor het coderen en het decoderen; symmetrische systemen.

Bij het communiceren via computernetwerken wil men berichten versturen zonder vooraf gemeenschappelijke sleutels af te spreken. Ook wil de ontvanger vaak zekerheid over diegene die het bericht verstuurt, een zogenaamde digitale handtekening. Dergelijke systemen noemt men Public Key Systems.

5.1 Het principe

De communicatie met persoon X wordt gevoerd met twee functies die zorgen voor de codering en de decodering. Persoon X construeert:

- → een publieke functie PublicFuncX en
- → een geheime functie SecretFuncX.

PublicFuncX kan bv. bekendgemaakt worden via een website en SecretFuncX mag niet kunnen berekend worden uit PublicFuncX.

Veronderstel dat een persoon Y een geheim bericht M wil versturen naar X.

Y gebruikt de publieke functie PublicFuncX om M te coderen in C: C=PublicFuncX(M).

X gebruikt de geheime functie SecretFuncX om C te decoderen in M: M=SecretFuncX(C).

Onmiddellijk wordt duidelijk welk verband er moet zijn tussen beide functies nl.

SecretFuncX(PublicFuncX(M))=M (*).

Indien Y niet geïnteresseerd is in geheimhouding maar een handtekening wil meesturen, construeert Y zelf ook twee functies: PublicFuncY en SecretFuncY.

Y stuurt de volgende boodschap C=SecretFuncY(M).

X gebruikt de publieke functie van Y om te decoderen: M=PublicFuncY(C).

Nu moet gelden: **PublicFuncY(SecretFuncY(M))=M (**)**.

In geval van geheimhouding met digitale handtekening moeten (*) en (**) gelden. Persoon Y zal dan de volgende boodschap versturen:

C=PublicFuncX(SecretFuncY(M)).

De ontvanger, persoon X, decodeert op de volgende manier:

PublicFuncY(SecretFuncX(C))
= PublicFuncY(SecretFuncX(PublicFuncX(SecretFuncY(M))))
= PublicFuncY(SecretFuncY(M))
= M

Hoe de bovenstaande principes in de realiteit toegepast worden, tonen we met de RSA code. Dit systeem is genoemd naar de ontdekkers ervan (in 1978), R.L. Rivest, A. Shamir en L. Adleman, van het MIT (Massachusetts Institute of Technologie).

Maar vooraleer de RSA-code te behandelen, bekijken we enkele wiskundige eigenschappen die de fundamenten ervan vormen.

5.2 RSA Math

De RSA-code is gebaseerd op eigenschappen van priemgetallen.

Het vermenigvuldigen van priemgetallen is eenvoudig maar het ontbinden van zeer grote getallen in priemgetallen is een zeer moeilijk probleem, zelfs voor een product van twee grote priemgetallen.

En voor het machtsverheffen modulo n van een groot getal bestaan er efficiënte algoritmen.

Bovenstaande opmerkingen liggen aan de basis van de RSA-code.

5.2.1 Fermat


1601-1665

De Franse wiskundige Fermat ontdekte de volgende eigenschap.

De 'Kleine stelling van Fermat'
Als p een priemgetal is en a een getal met a en p onderling ondeelbaar, geldt $a^{p-1} \equiv 1 \bmod p$.

a	p	a^{p-1}	$a^{p-1} \equiv \cdots \bmod p$
2	5	16	1
3	7	729	1
2	11	1024	1
6	13	2176782336	1
4	10	262144	4
5	10	1953125	5

5.2.2 Euler


1707-1783

Ook Euler vond een gelijkaardig, maar algemener resultaat. De kleine stelling van Fermat kan hieruit afgeleid worden. Eerst voeren we een nieuw begrip in, de Eulerfunctie, ook totiëntfunctie genoemd.

Definitie
Voor ieder natuurlijk getal $n \neq 0$ definiëren we
$\varphi(n)$ = aantal getallen i zodat $1 \leq i \leq n$ en de $\gcd(i,n) = 1$.

Voorbeelden

a. $\varphi(2) = 1$ b. $\varphi(10) = 4$ c. $\varphi(3) = 2$

d. $\varphi(15) = 8$ e. $\varphi(5) = 4$ f. $\varphi(35) = 24$

g. $\varphi(p) = p - 1$ als p een priemgetal is.

Eigenschap
Voor iedere twee priemgetallen p en q geldt $\varphi(p \cdot q) = (p-1)(q-1)$.

Stelling van Euler
$\forall a, n \in \mathbb{N}$ met $n > 0$ en $\text{ggd}(a,n) = 1$ geldt $a^{\varphi(n)} \equiv 1 \bmod n$.

Voor $a = 5$ en $n = 8$ ($\text{ggd}(5,8) = 1$) geldt: $5^{\varphi(8)} = 5^4 \equiv 1 mod\ 8$.

Opmerking
Indien n een priemgetal is voor stelling van Euler, geeft dit de kleine stelling van Fermat.

5.3 Coderen en decoderen met RSA

a. Voorbereiding
We starten met twee priemgetallen.

In realiteit zijn dit zeer grote priemgetallen maar om het principe uit te leggen kiezen we $p = 5$ en $q = 11$ met als product $n = p \cdot q = 55$ en $\varphi(n) = 4 \cdot 10 = 40$.

b. Constructie van de geheime sleutel
We bepalen een getal e zodat $1 < e < \varphi(n)$ en $\text{ggd}\big(e, \varphi(n)\big) = 1$.
Hiervoor kiezen we bijvoorbeeld een priemgetal tussen 1 en $\varphi(n)$, b.v. $e = 23$.

De geheime sleutel is: 23 (= e) en 55 (= n).

c. Constructie van de publieke sleutel
We bepalen een getal d met $1 < d < \varphi(n)$ en $d \cdot e = 1\ mod\ \varphi(n)$. Bestaat zo'n d ?

Vermits $\text{ggd}\big(e, \varphi(n)\big) = 1$ geldt volgens de stelling van Euler dat $e^{\varphi(\varphi(n))} \equiv 1 \bmod \varphi(n)$ zodat $e \cdot e^{\varphi(\varphi(n))-1} \equiv 1 \bmod \varphi(n)$. $e^{\varphi(\varphi(n))-1}$ is een goede kandidaat voor d.

In ons geval geeft dit: $e^{\varphi(40)} \equiv 1 \bmod 40 \Leftrightarrow e^{16} \equiv 1 \bmod 40 \Leftrightarrow e^{15} \cdot e \equiv 1 \bmod 40$.

$e^{15} = 23^{15} \equiv 7 \bmod 40$ voldoet aan het gestelde.

De publieke sleutel is: 7 (= d) en 55 (= n).

d. Werkt RSA?
Veronderstel even dat we de letter w d.m.v. het getal 22 willen decoderen.

We coderen 22 met de publiek sleutel als volgt: $22^7 \equiv 33 \bmod 55$.

Voor het decoderen bepalen we: $33^{23} \equiv 22 \bmod 55$.

Stel algemeen dat we een numerieke boodschap M coderen als $C \equiv M^e \bmod n$.

De vraag die we ons dan stellen is of $C^d \equiv M\ mod\ n$?

Er geldt: $C^d \equiv (M^e)^d \equiv M^{e \cdot d} \equiv M^{1+k\varphi(n)}$ $\quad (e \cdot d \equiv 1 \bmod \varphi(n))$
$\equiv M \cdot M^{k\varphi(n)} \equiv M \cdot M^{\varphi(n)} \cdot M^{\varphi(n)} \cdot \ldots \cdot M^{\varphi(n)}$
$\equiv M \bmod n$ (*)

(*) We moeten nog tonen dat $M \cdot M^{\varphi(n)} = M^{\varphi(n)+1} \equiv M \bmod n$,
m.a.w. dat $n = p \cdot q$ deelt $M^{\varphi(n)+1} - M = M\big(M^{\varphi(n)} - 1\big)$.

Het is voldoende te tonen dat $M\left(M^{\varphi(n)}-1\right)$ deelbaar is door p en q.

Voor bijvoorbeeld p geldt:

(i) p deelt M of
(ii) p deelt M niet.

Uit geval (i) volgt dat $M\left(M^{\varphi(n)}-1\right)=M^{\varphi(n)+1}-M$ ook deelbaar is door p en voor (ii) geldt volgens Fermat dat $M^{p-1}\equiv 1 \bmod p$.

M.a.w. . $M^{\varphi(n)+1}=M^{\varphi(n)}\cdot M=M^{(p-1)(q-1)}\cdot M=(M^{p-1})^{q-1}\cdot M\equiv M \bmod n$

Een analoge redenering kan gevoerd worden voor q.

e. Veiligheid?

De veiligheid van RSA is relatief goed gewaarborgd als de geheime sleutel e niet te achterhalen is binnen een aanvaardbare tijd. Na n te factoriseren in priemfactoren, berekenen we $\varphi(n)$. De geheime sleutel volgt dan uit $d\cdot e\equiv 1 \bmod \varphi(n)$, daar d publiek is.

Het factoriseren van zeer grote getallen is zelfs voor hedendaagse computers een werk dat veel tijd in beslag neemt. Hierdoor is er heel wat onderzoek naar snelle algoritmen om getallen te factoriseren in priemfactoren enerzijds en anderzijds naar het vinden van steeds grotere priemgetallen. Nog steeds is geen formule gevonden voor het n^{e} priemgetal, maar wel kan men gemakkelijk bewijzen dat er oneindig veel priemgetallen zijn.

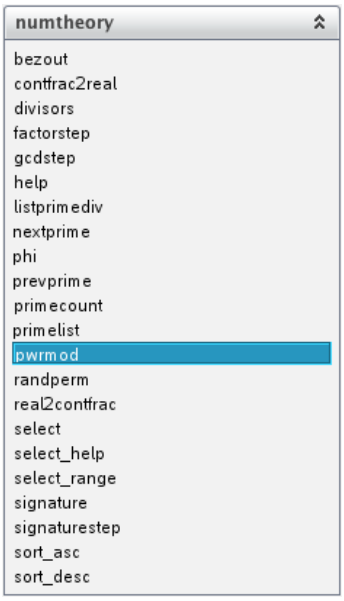
Bijvoorbeeld werd in 1996 in het Centrum voor Wiskunde en Informatica te Amsterdam een methode ontwikkeld waarmee een getal van 130 cijfers kan ontbonden worden.

De Duitse oogarts Martin Nowak heeft op 18 februari 2005 een nieuw grootste priemgetal ontdekt. Het bestaat uit maar liefst 7.816.230 cijfers en is het 42ste gekende Mersenne-priemgetal, een priemgetal van de vorm 2^p-1 met $p=25\,964\,951$. Dr. M. Nowak is een van de vele vrijwilligers die deel uit maken het *Great Internet Mersenne Prime Search* (GIMPS - www.mersenne.org) project. Duizenden vrijwilligers stellen rekentijd op hun computer beschikbaar om te zoeken naar nieuwe Mersenne-priemgetallen. Het rekenwerk nam meer dan 50 dagen in beslag op zijn Pentium 4, 2.4 GHz. Dr. Martin Nowak nam voor het eerst deel aan het project in 1999 en had zes jaren later 24 computers die rekenwerk verrichten voor GIMPS.

En op 26 december 2017 ontdekte Jonathan Pace, een 51 jarige Electrical Engineer, het tot nu toe grootste priemgetal, $2^{77232917}-1$, een getal van 23 249 425 cijfers en het 50ste Mersenne priemgetal. Het bewijs nam zes dagen non-stop rekentijd in beslag op een computer met een Intel i5-6600 CPU. Jonathan kreeg hiervoor een $ 3000 GIMPS reward.

5.4 RSA met TI-Nspire CX technologie

Voor het uitvoeren van RSA-codering met TI-Npsire CX technologie gebruiken we de volgende commando's uit de Library numtheory:



- nextprime(n) — kleinste priemgetal ≥ n
- prevprime(n) — grootste priemgetal ≤ n
- pwrmod(a,n,b) — a^b mod n
- phi(n) — Eulerfunctie

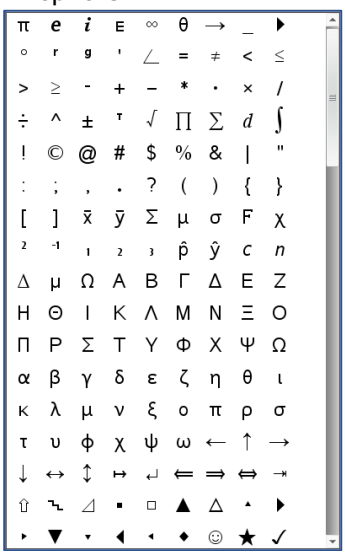
numtheory\nextprime(123)	127
numtheory\prevprime(123)	113
numtheory\pwrmod(123456,789,11)	4
numtheory\phi(55)	40

Voor het coderen en decoderen van de boodschap gebruiken we de ASCII codes van de volgende karakters, samen met de TI-Nspire codes voor speciale karakters:

ASCII

Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
32	20	040		Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	Del

TI-Nspire CX



Het omzetten van een karakter in de numerieke representatie en omgekeerd gebeurt via char("x") en ord(n).

© ASCII	
ord("w")	119
char(119)	"w"
ord("@")	64
char(64)	"@"
© TI−Nspire CX	
ord("π")	960
char(960)	"π"
ord("∫")	8747
char(8747)	"∫"

Als voorbeeld van RSA-codering starten we met als voorbeeld het coderen van “Kraak de Code” met de sleutels waarmee deze coderingstechniek voor het eerst werd uitgevoerd door R.L. Rivest, A. Shamir en L. Adleman:

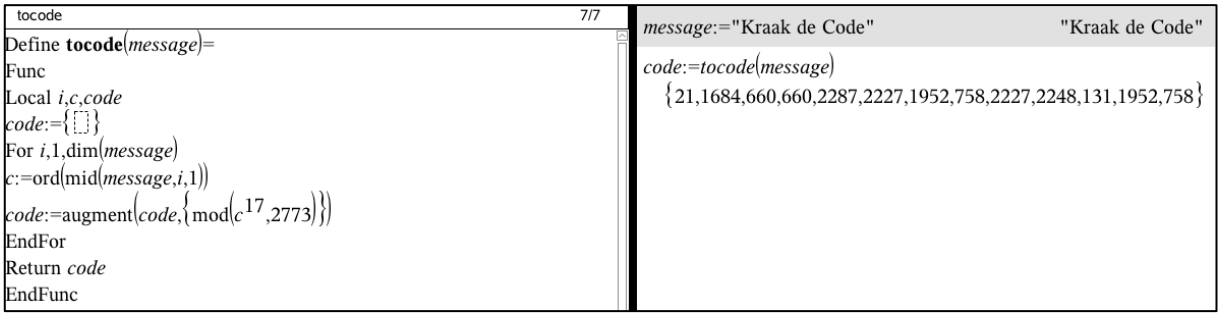
- De priemgetallen $p = 47$ en $q = 59$ ($n = 2773$ en $\varphi(n) = 2668$)
- De publieke sleutel 17 en 2773
- De geheime sleutel 157 en 2773

Voor het coderen, zullen we de boodschap(string) omzetten naar een lijst met de numerieke representaties en voor het ontcijferen terug naar de oorspronkelijke string (boodschap).

Hiervoor definiëren we de volgende twee functies:

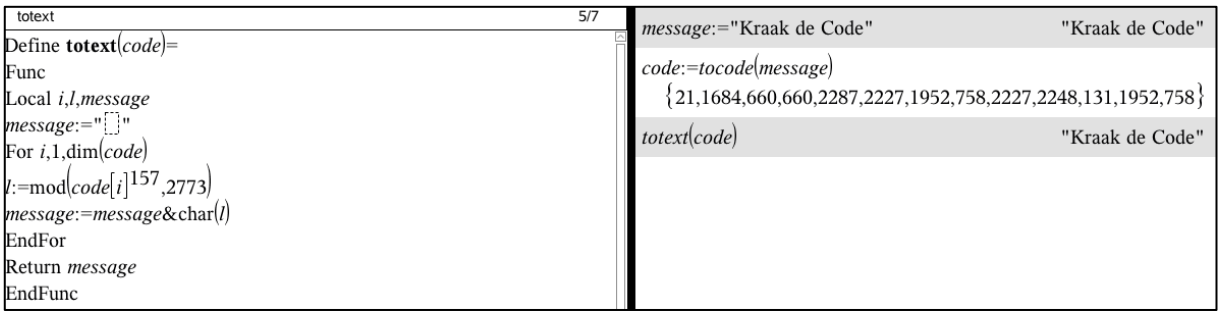
<u>Coderen</u>

```
Define tocode(message)=
Func
Local i,c,code
code:={ }
For i,1,dim(message)
  c:=ord(mid(message,i,1))
  code:=augment(code,{mod(c^17,2773)})
EndFor
Return code
EndFunc
```



<u>Ontcijferen</u>

```
Define totext(code)=
Func
Local i,l,message
message:=" "
For i,1,dim(code)
  l:=mod(code[i]^157,2773)
  message:=message&char(l)
EndFor
Return message
EndFunc
```



JE EIGEN RSA-CODE OPZETTEN

Stap 1 – Keuze priemgetallen

Kies twee willekeurige priemgetallen *p* en *q* voldoende groot.

$p:=numtheory\backslash nextprime(123456789)$	123456791
$q:=numtheory\backslash prevprime(987654321)$	987654319

Bereken $n = p \cdot q$ en $\varphi(n)$.

$n:=p\cdot q$	121932632841030329
$phin:=numtheory\backslash phi(n)$	121932631729919220

Stap 2 – Geheime sleutel

Kies als geheime sleutel, e, b.v. een priemgetal kleiner dan $\varphi(n)$.
Zo is altijd voldaan aan de voorwaarde $ggd(e, \varphi(n)) = 1$.

$e:=numtheory\backslash nextprime(123456)$	123457

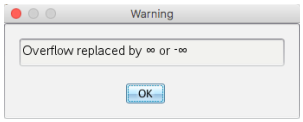
Stap 3 – Publieke sleutel

We zoeken als publieke sleutel een getal d kleiner dan $\varphi(n)$ waarvoor geldt dat $d \cdot e = 1 \, mod \, \varphi(n)$.

Een getal dat hieraan voldoet is: $e^{\varphi(\varphi(n))-1} \, mod \, \varphi(n)$.

Dergelijke machten kunnen erg groot worden en zo een overflow veroorzaken. Daarom gebruiken we het pwrmod commando i.p.v. mod.

⚠ $d:=\mathrm{mod}(e^{numtheory\backslash phi(phin)-1}, phin)$	$\mathrm{mod}(\infty, 121932631729919220)$



$d:=numtheory\backslash pwrmod(e,numtheory\backslash phi(phin)-1,phin)$	48450287809868353

Stap 4 – Coderen

We coderen met de publieke sleutel de volgende tekst:

Het coderen van de opdracht kost $ 50000!

We vervangen in voorgaande RSA-programma's de mod-commando's ook hier door pwrmod en voegen variabelen voor de publieke en geheime sleutel toe.

```
tocode                                   5/7
Define tocode(message,d,n)=
Func
Local i,c,code
code:={}
For i,1,dim(message)
c:=ord(mid(message,i,1))
code:=augment(code,{numtheory\pwrmod(c,d,n)})
EndFor
Return code
EndFunc
```

$[d \quad e \quad n]$ $[48450287809868353 \quad 123457 \quad 121932632841030329]$

message:="Het uitvoeren van de opdracht kost $ 50000!"
"Het uitvoeren van de opdracht kost $ 50000!"

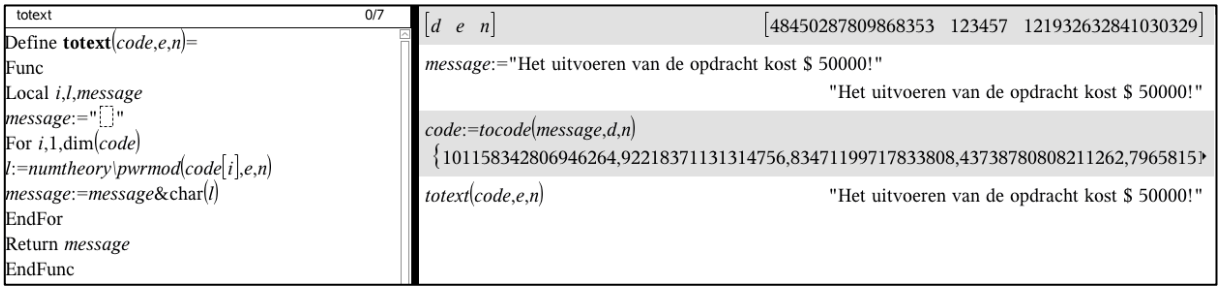
code:=*tocode*(*message*,*d*,*n*)
{101158342806946264,92218371131314756,83471199717833808,43738780808211262,7965815▸

list▸mat(*code*,4)

101158342806946264	92218371131314756	83471199717833808	43738780808211262
79658151354716976	115540912663318763	83471199717833808	97882236561230572
47359107480565251	92218371131314756	53701580463404063	92218371131314756
99470674787274437	43738780808211262	97882236561230572	45057900499379783
99470674787274437	43738780808211262	112649766179062639	92218371131314756
43738780808211262	47359107480565251	50218069477398789	112649766179062639
53701580463404063	45057900499379783	45479746214356753	69091787239233476
83471199717833808	43738780808211262	4575103614494105	47359107480565251
62661293777653808	83471199717833808	43738780808211262	59891434506838801
43738780808211262	106216462786536230	70999687221228391	70999687221228391
70999687221228391	70999687221228391	93008951080054655	0

***Stap 5* – Ontcijferen**

Met de geheime sleutel ontcijferen we de zojuist bekomen code.



RSA MET HANDTEKENING EN ZONDER GEHEIMHOUDING

In dit geval keren we de rollen van sleutels om. We coderen met de geheime sleutel en decoderen met de publieke sleutel. De verzender bepaalt de sleutel.

$p:=numtheory\backslash nextprime(123456789)$	123456791
$q:=numtheory\backslash prevprime(987654321)$	987654319
$n:=p\cdot q$	121932632841030329
$phin:=numtheory\backslash phi(n)$	121932631729919220
$e:=numtheory\backslash nextprime(123456)$	123457
$d:=numtheory\backslash pwrmod(e,numtheory\backslash phi(phin)-1,phin)$	48450287809868353
$message:="OK"$	"OK"
$code:=tocode(message,e,n)$	{61761590546380948,23331560135946813}
$totext(code,d,n)$	"OK"

Merk wel op dat in dit geval alleen de verzender kan coderen en iedereen de boodschap kan decoderen met de publieke sleutel.

Gebruikmakend van een interactieve Notes pagina kunnen we bovenstaande programma's in een template toepassen om te coderen en te decoderen.

De RSA-code

Keuze priemgetallen: p = 123457 en q = 654307 ⇒ n=80778779299 en φ(n) =80778001536 .

Geheime sleutel e met ggd(e,φ(n)) = 1: e = 456679 .

Publieke sleutel: d = $e^{\varphi(\varphi(n))-1} \bmod \varphi(n) = 15047120215$.

Boodschap: Kraak de Code !

De versleutelde boodschap, code = $\text{boodschap}^{d} \bmod n$:

{49655647965,34417631668,79765089453,79765089453,47044951707,6652653117,833883465,6806858

of in matrixvorm:

$$\begin{bmatrix} 49655647965 & 34417631668 & 79765089453 & 79765089453 & 47044951707 \\ 6652653117 & 833883465 & 68068589928 & 6652653117 & 72333782997 \\ 38770893582 & 833883465 & 68068589928 & 6652653117 & 31697257369 \end{bmatrix}$$

Gebruik de slider – – om te ontcijferen:

Niet ontcijferd
Please
Wait

Gebruik de slider – – om te ontcijferen:

Kraak de Code !

{75,114,97,97,107,32,100,101,32,67,111,100,101,32,33}

{"K","r","a","a","k"," ","d","e"," ","C","o","d","e"," ","!"}

www.t3europe.eu

T³ EUROPE

Teachers Teaching with Technology™

Texas Instruments