Geheimschrift op de TI-83+

Gerard Tel

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2006-017

www.cs.uu.nl

ISSN: 0924-3275

Geheimschrift op de TI-83+

Gerard Tel
Universiteit Utrecht, Departement Informatica
Email: gerard@cs.uu.nl

17 maart 2006

Wat kun je verwachten?

Cryptografie is: het verzinnen en gebruiken van geheimschriften, oftewel "codes" waarmee je informatie onleesbaar kunt maken voor buitenstaanders. Handig als je iets geheims moet versturen over het Internet! Want je zorgt er natuurlijk voor dat jouw vriendjes de berichten wel weer leesbaar kunnen maken.

In deze workshop wordt een geheimschrift uitgelegd; het is bedacht door Taher Elgamal in 1985. Het leuke ervan is, dat het helemaal met getallen werkt. En als je over een TI-83+ (of compatibele) rekenmachine beschikt, kun je een programmaatje downloaden en alles zelf narekenen. Ga hiervoor naar de website www.cs.uu.nl/~gerard/Cryptografie/Elgamal

In de tekst staan opdrachten: dingen die je kunt uitzoeken of narekenen, experimenten die je kunt doen met een programma, of problemen die je zelf kunt uitprogrammeren op een computer of rekenmachine. De workshop is geschikt voor leerlingen die zelf met cryptografie en/of met hun programmeerbare rekenmachine aan de slag willen. Maar ook voor leraren, voor groepen, of andere geïnteresseerden in het Elgamal geheimschrift. Wil je precies weten hoe het programma werkt, lees dan de appendix (vanaf pagina 17).

1 Wat moet je weten?

Dit eerste hoofdstukje heeft het over een paar dingen die je van wiskunde, getallen en computers moet weten om het geheimschrift te kunnen begrijpen.

1.1 Computers en programmeren

De berekeningen van geheimschriften zijn al gauw te groot om ze uit je hoofd of met pen en papier na te doen. Cryptografie is werk voor rekenmachines! Bij deze workshop is een programma ELGAMAL.8xp dat je TI-83+ (of compatibele) rekenmachine omtovert in een echte Elgamal-calculator!

Opdracht 1.1 *Ga naar* `www.cs.uu.nl/~gerard/Cryptografie/Elgamal/`*, download het programma, en laad het in je* `TI-83+`*. Start het op en kijk welke versie je hebt (*Hoofdmenu, Parameters, Over programma*). Sluit het programma af (*Hoofdmenu, Stop, Ja*).*

Opdracht 1.2 *Heb je een tweede* `TI-83+`*, kopieer het programma dan van de ene naar de andere.*

Heb je een heel ander soort rekenmachine, of een PC, dan kun je proberen zelf een programma te maken.

1.2 Exponentiëren

Exponentiëren of machtsverheffen betekent: herhaald vermenigvuldigen. Met de notatie x^a bedoelen we het getal dat krijgt als je a factoren x met elkaar vermenigvuldigt, bijvoorbeeld $24^5 = 24 \cdot 24 \cdot 24 \cdot 24 \cdot 24$.

Een heel belangrijke rekenregel, waar het geheimschrift van Elgamal op gebaseerd is, zegt dat als je tweemaal machtsverheft, de volgorde van de exponenten niet uitmaakt. In formule: $(x^a)^b = (x^b)^a$. Dat die formule geldt zul je wel snappen als je $(x^3)^5$ en $(x^5)^3$ uitschrijft:

$$(x \cdot x \cdot x) \cdot (x \cdot x \cdot x) \cdot (x \cdot x \cdot x) \cdot (x \cdot x \cdot x) \cdot (x \cdot x \cdot x)$$

en

$$(x \cdot x \cdot x \cdot x \cdot x) \cdot (x \cdot x \cdot x \cdot x \cdot x) \cdot (x \cdot x \cdot x \cdot x \cdot x)$$

Bij vermenigvuldigen doen de haakjes er niet toe, en in beide gevallen staat er: x^{15}. Je ziet dat $(x^a)^b$ en $(x^b)^a$ allebei gewoon gelijk zijn aan $x^{(a \cdot b)}$.

Opdracht 1.3 *Als je x weet en x^{15} echt wilt uitrekenen, hoe vaak moet je dan vermenigvuldigen? En x^{16}?*

1.3 Rekenen met resten

Geheimschriften rekenen niet met reële of complexe getallen, en zelfs niet met alle gehele getallen. Men neemt een vast getal, dat de *modulus* heet, en rekent dan alleen met getallen die als rest kunnen overblijven bij een deling door die modulus. Bij modulus 7 (als in Kader 1) zijn dat de getallen 0, 1, 2, 3, 4, 5 en 6 (7 kan zelf natuurlijk nooit een rest zijn): precies 7 getallen dus.

We gaan de vermenigvuldigtabel iets beter bekijken. De rij achter de 0 bevat alleen maar nullen; logisch, want daar staan getallen $0 \cdot x$ (x één van de getallen bovenaan) en $0 \cdot x$ is altijd 0. In de rij achter de 3 zie je elk van de zeven getallen staan; uiteraard de 0 voorop want op die plaats staat $3 \cdot 0$, en verder staan de getallen door elkaar maar ze staan er wel allemaal. En dat geldt ook voor de andere rijen: elke rij (behalve die van 0) *bevat elk getal precies één keer.*

Kader 1: Optellen en vermenigvuldigen met modulus 7

Bij het optellen van twee resten moet je, als het resultaat groter dan 6 is, er weer 7 van aftrekken. Als je vermenigvuldigt, bijvoorbeeld 4 maal 5, kom je vaak boven de 6 uit, maar dan neem je weer de rest bij deling door 7, in dit geval 6.

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

·	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Hoe dit handig is voor een geheimschrift bespreken we verder in sectie 2.1, maar hier vast dit. Als ik een getal x heb vermenigvuldigd met 3 en ik geef je de uitkomst (bijvoorbeeld 5), dan kun je weten wat x was, omdat 5 maar éénmaal voorkomt in de rij van 3: x moet 4 zijn geweest.

Maar als ik een getal x heb vermenigvuldigd met getal z en je wel de uitkomst vertel (bijvoorbeeld 5), maar niet wat z was, dan weet je x niet. Dat komt omdat 5 in elke rij van de vermenigvuldigtabel voorkomt, en steeds op een verschillende plaats. Zoek zelf op waar alle vijven staan!

Opdracht 1.4 *Schrijf de optel- en vermenigvuldigtabellen eens uit voor modulus 6 en voor modulus 11. Welke tabellen hebben ook de eigenschap dat elk getal op elke rij vookomt?*

Opdracht 1.5 *Bedenk een* `TI-83+`*-opdracht (reductie modulo P) die van variabele V de rest bij deling door P berekent.*
Schrijf een `TI-83+`*-programma (tabel-analyse) dat, bij een gegeven modulus P, uitrekent of de vermenigvuldig-tabel de eigenschap heeft dat elk getal op elke rij voorkomt.*
Schrijf een programma (modulaire deling) dat, bij gegeven Z en Y, een getal X uitrekent dat voldoet aan ZX=Y (in het restrekenen!).

1.4 Wat kunnen computers aan?

Het belangrijkste wat je moet weten om een geheimschrift te kunnen bedenken is: welke dingen een computer *wel* en *niet* kan uitrekenen. We schrijven getallen op als een rijtje cijfers en dat rijtje is heel kort in verhouding tot het getal dat het voorstelt: getallen tot een miljoen kunnen we schrijven met maar zes cijfers, en getallen tot een biljoen met maar twaalf cijfers! Je snapt vast wel, dat je voor het opschrijven van een grote hoeveelheid n maar ongeveer $\log(n)$ cijfers gebruikt.

Omdat we kunnen optellen en vermenigvuldigen door die cijfers te manipuleren, kunnen we snel rekenen, ook met (heel) grote getallen. Wij vinden het heel vanzelfsprekend dat we

Kader 2: Rekenen met de holemens

Een (denkbeeldige) holemens noteert een getal, bijvoorbeeld 3, met drie steentjes; we noemen dat een *unaire* notatie. Voor grotere getallen heeft de holemens meer ruimte nodig, en wel: evenredig met het getal zelf.
Een holemens die 3 met 5 wilde vermenigvuldigen, maakte drie rijen van vijf steentjes, waarna hij de steentjes ging tellen. Dat is bij grote getallen veel werk, en wel: evenredig met de grootte van de uitkomst. De holemens zou met deze methode *in principe* 1.000.000 maal 1.000.000 kunnen uitrekenen, maar zal niet lang genoeg leven om de uitkomst *daadwerkelijk* te vinden.

grote getallen kunnen opschrijven, optellen en vermenigvuldigen, maar het systeem met cijfers (en de nul) is een vrij late uitvinding. De Holemensen uit Kader 2, Egyptenaren en Romeinen konden niet reken met hoeveelheden groter dan enige tienduizenden! Computers rekenen niet met een tientallig stelsel zoals wij, maar met een tweetallig stelsel; ook daarmee kun je heel snel optellen, aftrekken, vermenigvuldigen en delen met heel grote getallen.

Opdracht 1.6 *Bedenk hoe je getallen in het Romeinse systeem kunt optellen en aftrekken. Bedenk hoe je Romeinse getallen kunt vermenigvuldigen.*

Opdracht 1.7 *Zoek op wanneer de decimale notatie is uitgevonden en door wie.*

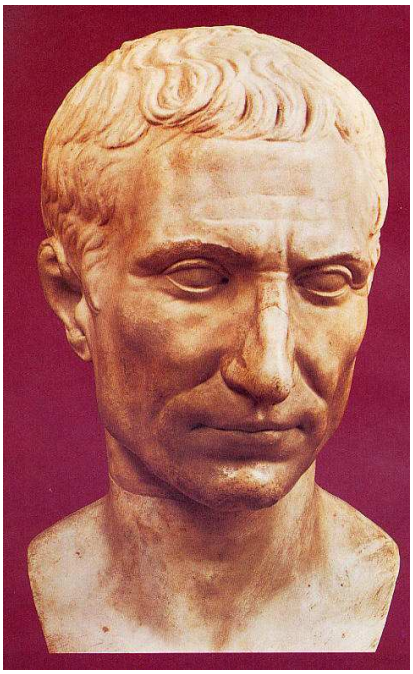
Maar hoe zit het nu met machtsverheffen? Gelukkig hoef je, om x^{21} uit te rekenen, niet 21 maal te vermenigvuldigen, en zelfs niet 20 maal. Kijk maar: door vier keer te kwadrateren kun je de machten x^2, x^4, x^8 en x^{16} uitrekenen. En omdat $21 = 16 + 4 + 1$, is $x^{21} = x^{16} \cdot x^4 \cdot x$, en met nog twee vermenigvuldigingen ben je klaar. Totaal 6 vermenigvuldigingen voor x^{21}.

Als de exponent een groot getal is van k cijfers, heb je op deze manier ongeveer $5k$ vermenigvuldigingen nodig. Een miljoenste macht kost 25 vermenigvuldigingen.

Opdracht 1.8 *Schrijf 21 eens als binair getal en kijk, hoe de zes vermenigvuldigingen bij het getal passen. Probeer dan te begrijpen waarom je de miljoenste macht met 25 vermenigvuldigingen kunt vinden.*

Opdracht 1.9 *Schrijf een programma (machtsverheffen) dat machten uitrekent met zo weinig mogelijk vermenigvuldigingen.*

Kader 3: De Caesar code op de TI-83+



Met de Caesar code kun je het TI-83+-programmaatje ELGAMAL leren kennen. Laad het programma in je rekenmachine en start het op; in het Elgamal hoofdmenu kun je kiezen voor Caesar.
Voordat je kunt versleutelen of ontsleutelen moet het programma de sleutel weten (het getal z). De optie Sleutel kiezen zal een random getal kiezen voor z (en die waarde laten zien), terwijl je met Sleutel instel zelf een waarde kunt ingeven. Als je met een andere TI-83+ berichten wilt uitwisselen, moet in beide rekenmachines dezelfde sleutel ingesteld zijn! Je kunt samen een getal bedenken en het op beide machines instellen met Sleutel instel, of op de ene machine Sleutel kiezen gebruiken en het getoonde getal op de andere machine invoeren met Sleutel instel. Met Sleutel tonen kun je de ingestelde sleutel en modulus bekijken.
Met Versleutelen kun je een getal x invoeren, waarna je de versleutelde waarde $z \cdot x$ te zien krijgt. Met Ontsleutelen kun je dat getal op een andere rekenmachine weer invoeren, en als dezelfde sleutel is ingesteld krijg je het oorspronkelijke getal weer terug.

2 Geheimschrift: Symmetrisch en Public-Key

Nu gaan we echt naar geheimschriften (soms codes genaamd) kijken. We doen daarbij net, alsof de boodschap die verstuurd wordt altijd een getal is. Je kunt vast zelf wel bedenken, hoe je tekst in getallen kunt omzetten en omgekeerd.

2.1 De Caesar code

De Caesar code is een geheimschrift dat gebaseerd is op vermenigvuldigen van resten. Twee partijen spreken een getal z af als sleutel. In plaats van de geheime boodschap, het getal x, wordt het getal $y = z \cdot x$ verstuurd. De ontvanger kent z en kan x vinden door y te delen door z. Als je een TI-83+ hebt, kun je hiermee experimenteren met het programma ELGAMAL (waarschijnlijk ben je hier vrij snel op uitgekeken).

De veiligheid van de Caesar code is erg goed zolang je maar 1 boodschap verstuurt (en: je de sleutel goed geheim houdt!). Maar als je twee getallen, x_1 en x_2, verstuurt als $y_1 = z \cdot x_1$ en $y_2 = z \cdot x_2$, dan blijven je getallen niet helemaal geheim. Iemand die de codeboodschappen afluistert, kan het quotient x_1/x_2 berekenen, want dat is gelijk aan y_1/y_2! Na een paar getallen is dat informatielekje al genoeg om de hele boodschap te

kunnen ontrafelen. Je zou dus eigenlijk voor elke boodschap een *nieuwe* sleutel moeten afspreken!

2.2 Public-Key codes

Als de Caesar code wilt gebruiken, moet je eerst samen een sleutel afspreken, want bij versleutelen en ontsleutelen moet je hetzelfde getal z gebruiken. En dat afspreken van een sleutel is meteen de grootste moeilijkheid van de Caesar code: want voordat je met iemand in het geheim berichten kunt uitwisselen... moet je eerst in het geheim een sleutel uitwisselen! Alle codes waarbij je voor het versleutelen en het ontsleutelen dezelfde sleutel gebruikt, heten *symmetrische codes*.

Deze foto uit 1977 toont drie wiskundigen die door het bedenken van geheimschriften beroemd zijn geworden: het zijn Ralph Merkle, Martin Hellman en Whitfield Diffie. In 1976 bedachten Diffie en Hellman dat je een geheimschrift kunt maken dat met *twee verschillende* sleutels werkt: een getal b voor het versleutelen, en een getal a voor het ontsleutelen.

Het handige hiervan is, dat je sleutel b helemaal niet geheim hoeft te houden. Je kunt iemand de sleutel gewoon vertellen, zonder je zorgen te maken dat iemand je afluistert. Want als jouw vriend een bericht versleutelt met b, dan kan dat bericht toch niet met sleutel b weer ontsleuteld worden; daarvoor is de andere sleutel a nodig. En die houd je natuurlijk *wel* goed geheim!

Sleutel b die iedereen mag weten heet de *publieke sleutel* (public key) en sleutel a heet de *geheime sleutel* (secret key). Geheimschriften die met twee sleutels werken heten *public key* codes. Nu moeten die twee sleutels wel goed bij elkaar passen, want met sleutel a wil een bericht ontsleutelen dat met sleutel b versleuteld is. Maar als je die a kunt berekenen uit b.... dan moet je b alsnog geheim houden en werkt het hele idee natuurlijk niet. Er zijn verschillende geheimschriften uitgevonden die publieke sleutels gebruiken; het bekendste is het systeem RSA, uitgevonden door Rivest, Shamir en Adleman. Over dat systeem gaat een ander boekje: [Stu05].

2.3 De Discrete Logaritme

Als je met reële getallen rekent, kun je een logaritme bijna net zo snel uitrekenen als een macht. Wanneer a en b moeten voldoen aan de relatie $b = e^a$, dan kun je als je a weet, b uitrekenen (met de e^x-knop), en als je b weet, a uitrekenen (met de ln-knop).

Als je rekent met resten is dat *niet* zo. In paragraaf 1.4 zagen we, dat je $b = g^a$ kunt uitrekenen door ongeveer $5 \log a$ vermenigvuldigingen te doen. Maar als je b weet en wilt uitrekenen welke macht van g dat is, dan kun je de vermenigvuldigingen niet zo handig combineren. Het aantal vermenigvuldigingen dat je nodig hebt om de logaritme van b uit te rekenen is ongeveer $3\sqrt{a}$.

Kader 4: Uitrekenen van macht en logaritme

Als je met grotere getallen gaat rekenen, neemt de tijd voor machtsverheffen en logaritme niet in dezelfde verhouding toe. Deze tabel geeft, bij het aantal cijfers van a, (1) het aantal vermenigvuldigingen om een a-de macht uit te rekenen; (2) het aantal vermenigvuldigingen om de logaritme uit te rekenen; (3) de tijd die dit kost bij een miljoen vermenigvuldigingen per seconde.

cijfers in a	vermenigv. voor macht	vermenigv. voor log	Tijd voor logaritme
6	30	3000	3 ms
12	60	3.000.000	3 sec
18	90	3 miljard	50 min
24	120	$3 \cdot 10^{12}$	5 weken
50	250	$3 \cdot 10^{25}$	10 miljard eeuw
60	300	$3 \cdot 10^{30}$	10^6 miljard eeuw
100	500	$3 \cdot 10^{50}$	
300	1500	$3 \cdot 10^{150}$	

Je ziet, dat met de zes-cijferige getallen waarmee de `TI-83+` rekent, het uitrekenen van de logaritme nog niet duur genoeg is om a echt geheim te houden. Daarom wordt bij echt gebruik van de Elgamal code met veel grotere getallen gerekend: 60 cijfers is heel gebruikelijk. De 300 vermenigvuldigingen voor machtsverheffen kunnen dan toch op een simpele rekenmachine in enkele seconden berekend worden, maar het berekenen van een logaritme is niet mogelijk!

Opdracht 2.1 *Onderzoek, waar de rekentijd voor een logaritme van afhangt.*

q	p	g
199	797	96
199	17911	1276
199	199399	27715
199	1995971	1514387
1999	19991	15572
1999	187907	3374
1999	1963019	1459733
19997	159977	29494
19997	1959707	1642561
199967	1199767	1102244

Via Hoofdmenu, Log en macht *krijg je toegang tot een envoudige logaritme-berekening (*Holemens log*) en een snel algoritme (*Pollard rho*). Met* Machtsverhef *kun je een macht berekenen, waarna je uit dat getal de logaritme weer kunt berekenen à la Holemens, of met Pollard's methode. Omdat Pollard's methode met random getallen begint, moet je per invoer de berekening enige malen herhalen en een gemiddelde rekentijd nemen (per iteratie doet Pollard drie vermenigvuldigingen).*

Kijk eerst eens, hoe de rekentijd afhangt van de invoerwaarde. Is, bij eenzelfde invoer, de variatie erg groot? Hoe is de verdeling van het aantal iteraties?

Dan ga je experimenteren met andere waarden van q, p, en g. Die kun je instellen met Logaritmen, Instellen*, bijvoorbeeld op combinaties uit deze tabel. Met* Hoofdmenu, Parameters, Maak p,q,g *kun je andere geldige combinaties uitrekenen waarbij je de grootte van q en p ongeveer kunt kiezen.*

Uit deze experimenten kun je concluderen, waar de rekentijd voor de logaritmefunctie door wordt beïnvloed.

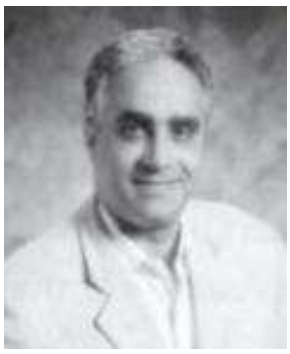
De grootste waarde die a kan hebben in het programma `ELGAMAL` is ongeveer 5 miljoen

(zie Kader 6). Het uitrekenen van de logaritme vraagt dan zo'n 7000 vermenigvuldigingen, wat enkele tientallen seconden kost. Maar door met grotere getallen te rekenen, kun je er voor zorgen dat een machtsverheffing *wel*, maar een logaritme *niet* uitgerekend kan worden. Hoeveel cijfers je daarvoor in je exponent moet stoppen kun je zien in Kader 4.

En dan zijn we klaar voor het geheimschrift van Elgamal....

3 Elgamal cryptografie

Je kunt nu lezen hoe het geheimschrift van Elgamal werkt en hoe je de berekeningen kunt doen met het programma `ELGAMAL` op de `TI-83+`.



Net zoals bij ons De Hond of De Leeuw een heel normale naam is, zijn er in Egypte mensen die De Kameel heten: Elgamal. Taher Elgamal bedacht zijn geheimschrift rond 1984 [ElG85]. Er wordt met resten gerekend, daarom moet er eerst een modulus p worden afgesproken; in het `TI-83+`-programma `ELGAMAL` is $p = 95917$ ingesteld als modulus. Ook moet er een getal g worden afgesproken als basis voor het machtsverheffen; in het programma is dat $g = 29609$. (Let op: bij serieus gebruik moet je een p van 300 cijfers kiezen!!) Met de opties HOOFDMENU - PARAMETERS - MAAK P,Q,G kun je andere waarden van p en g berekenen.

3.1 Een sleutelpaar

Had je bij de Caesar code één sleutel (getal z) die je voor berichten in beide richtingen kon gebruiken, bij Elgamal heb je altijd een *sleutelpaar* nodig: een publieke sleutel b om berichten af te sluiten en een geheime sleutel a om ze weer te openen. Tussen a en b bestaat deze heel simpele relatie:

$$b = g^a$$

Als je paragraaf 2.3 goed hebt begrepen dan weet je, dat het *wel* mogelijk is om uit a, b te berekenen, maar niet om a weer terug te vinden als je alleen b weet.

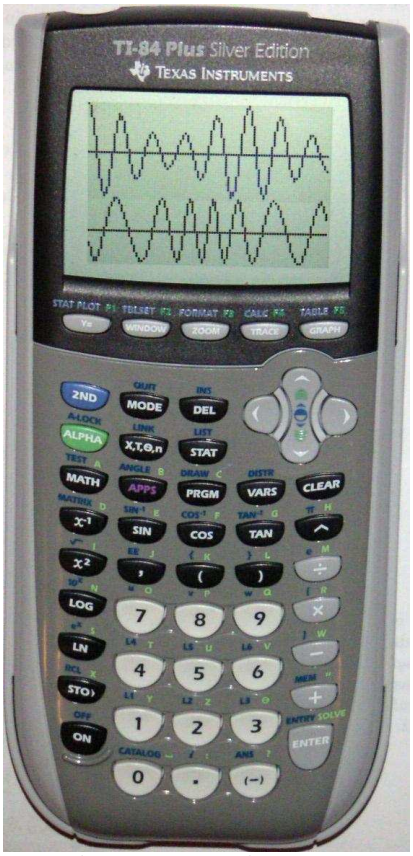
Stel nu dat Bob gecodeerde berichten wil sturen aan Alice; er zijn dan twee sleutels nodig, namelijk een geheime en een publieke sleutel van Alice. Het programma maakt altijd eerst de geheime sleutel a en berekent daarbij de publieke sleutel b door machtsverheffen.

Opdracht 3.1 *Dit is een opdracht voor twee* `TI-83+` *rekenmachines; noem ze Alice en Bob.*

1. *Alice kiest* ELGAMAL CODE *en dan* SLEUTEL MAKEN.
2. *Alice geeft voor a een getal op; zij moet dit getal onthouden en geheim houden!*
3. *Het sleutelpaar* (a, b) *verschijnt in het scherm; Alice geeft de publieke b aan Bob.*

Kader 5: De TI-83+-familie

De Texas Instruments TI-83+ is een rekenmachine die veel op middelbare scholen wordt gebruikt omdat hij onderwijs in verschillende exacte vakken ondersteunt.



Er zijn verschillende, onderling compatibele uitvoeringen: TI-83 Plus, TI-84 Plus, TI-84 Plus Silver. De belangrijkste verschillen zijn de hoeveelheid geheugen, rekensnelheid, en aansluitmogelijkheden; in het gebruik verschillen ze nauwelijks, en in het programmeren helemaal niet. Prijzen in Nederlandse winkels (eind 2005) varieren van 100 tot 150 euro.
De belangrijkste mogelijkheden, naast het gebruik als "gewone" wetenschappelijke rekenmachine, zijn de grafische functies, het installeren van applicatiemodules en het programmeren. Met de grafische functies kun je grafieken of geparametriseerde processen tekenen (foto: vergelijking van een in amplitude en een in frequentie gemoduleerde draaggolf). Applicaties zijn te verkrijgen over uiteenlopende onderwerpen: er is bijvoorbeeld een spreadsheet, een overzicht van het periodiek systeem van elementen, een spelletjespakket en ondersteuning van experimenten en metingen. Voor laboratorium-experimenten is het ook mogelijk, meetapparaten rechtstreeks op de rekenmachine aan te sluiten.
Berekeningen die niet zijn ingebouwd en niet als applicatie kunnen worden toegevoegd, kun je zelf programmeren in het ingebouwde Basicachtige programmeertaaltje (zie Kader 6).
Voor het maken van deze workshop werd door Texas Instruments Nederland een TI-84 Plus Silver Edition ter beschikking gesteld, waarvoor hartelijk dank!

4. *Bob kiest in het menu* Elgamal codering *voor* Invoer public *en voert de publieke sleutel van Alice in.*

Bob laat zijn geheime sleutel door de rekenmachine kiezen.

Opdracht 3.2 *Verwissel de rollen van Alice en Bob. Bij* Sleutel maken *geeft Bob aan a de waarde 0. Er verschijnt een willekeurig getal met de bijbehorende publieke sleutel. Bob vertelt de publieke sleutel aan Alice, die hem invoert met* Invoer public.

Nu zijn de sleutels ingesteld waarmee Bob en Alice berichten kunnen uitwisselen. Alice en Bob kennen nu elk hun eigen Public *en* Secret key, en van de ander alleen de Public key (zie Sleutels zien).

3.2 Versleutelen en ontsleutelen

Als Bob x wil sturen aan Alice heeft hij de publieke sleutel b nodig die hoort bij de geheime sleutel a van Alice. Voor ieder nieuw bericht van Bob wordt een nieuw getal z genomen en vermenigvuldigd met x. Samen met het product v, geeft Bob informatie u over z, maar zo, dat alleen Alice daaruit kan opmaken wat z is.

Opdracht 3.3 *Je gaat de Elgamal code nu echt gebruiken.*

1. *Bob verzint een getal x en kiest voor* VERSLEUTELEN.
2. *Voer x in; de versleuteling bestaat uit u en v.*
3. *Alice kiest voor* ONTSLEUTELEN *en voert u en v in.*

Als jullie alles goed hebben gedaan verschijnt bij Alice de x waar Bob mee begon. Is dat niet zo, kies dan SLEUTELS ZIEN *om te controleren of je elkaars publieke sleutel goed hebt ingesteld.*

In het programma gebeurt dit: Bob neemt een willekeurig getal k en kiest voor z het getal $z = b^k$. Net als in de Caesar code is v het product $z \cdot x$. Het getal u bevat informatie over k (en dus over z): $u = g^k$. Het lijkt niet erg geheim om samen met een gecodeerd bericht mee te sturen hoe je de sleutel hebt gemaakt. Maar toch kan dat in dit geval geen kwaad, want om k uit u te berekenen moet je een logaritme berekenen en dat is onmogelijk.

Opdracht 3.4 *Bob kiest weer* VERSLEUTELEN *en voert hetzelfde getal x in. Je krijgt een ander paar (u, v)! Geven al die paren bij Alice wel het goede antwoord?*

Hoe komt Alice nu aan z? Ook zij kan k niet berekenen. Maar z is gelijk aan b^k, ofwel $(g^a)^k$. En, zoals we in paragraaf 1.2 zagen, is dat gelijk aan $(g^k)^a$, oftewel u^a. Alice vindt dus de waarde van x door v te delen door u^a.

Samengevat komt het versleutelen en ontsleutelen hierop neer. Versleutelen van x door Bob: (1) kiest willekeurig k; (2) bereken $v = g^k$ en $u = b^k \cdot x$; (3) stuur op: u en v. Ontsleutelen van (u, v) door Alice: bereken x als $v \cdot u^{-a}$.

3.3 Hoe geheim is het?

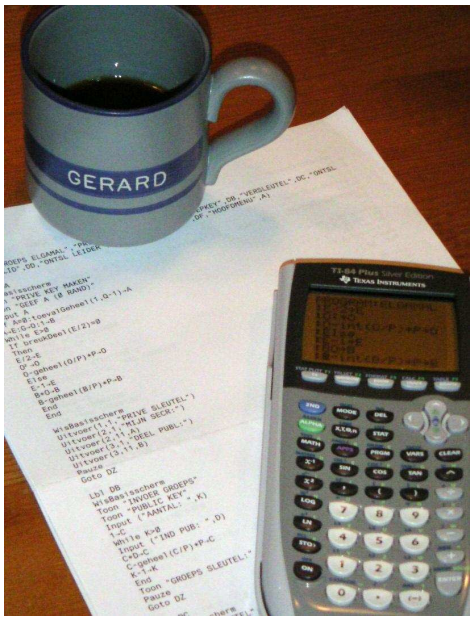
De Elgamal code is heel veilig. Wel is natuurlijk nodig, dat de getallen zo groot zijn dat je logaritmes echt niet kunt uitrekenen. In het `TI-83+`-programma is dat niet zo!

Denk hier eens over na. Carla weet dat Bob aan Alice gaat vertellen of hij haar in de eerste pauze ($x = 1$) of in de tweede pauze ($x = 2$) wil ontmoeten. Carla kan de gecodeerde berichten van Bob niet zomaar lezen, maar komt op het idee om zelf $x = 1$ en $x = 2$ te versleutelen met de publieke sleutel van Alice. Ze wil dan haar berichten vergelijken met die van Bob.

Opdracht 3.5 *Laat één persoon kiezen uit 1 of 2 en dat getal versleutelen. Een tweede persoon bekijkt de (u, v) en versleutelt zelf ook 1 en 2. Kan die erachter komen of het paar (u, v) een 1 of een 2 voorstelt?*

Kader 6: Programmeren op de `TI-83+`

De programmeertaal van de `TI-83+` lijkt erg op Basic, wat bv. ook voor de macro's in Microsoft Word en Excel wordt gebruikt. Je kunt het `ELGAMAL` programma bekijken op de `TI-83+` of met *TI Connect* op je computer.



Een programma bestaat uit een lijst van rekeninstructies, waarvan de uitkomst wordt opgeslagen in een geheugenplaats van de rekenmachine. Je kunt groepen instructies herhalen met de lusconditie `While`, je kunt tussen groepen instructies kiezen met de conditionele `If`, en je kunt de rekenmachine naar een bepaalde plaats in het programma sturen met een sprong `Goto/Lbl`. Net als elke rekenmachine heeft de `TI-83+` ook instructies om getallen in te lezen of te laten zien. Een heel handige instructie van de `TI-83+` is `Menu`, waarmee je de gebruiker snel kunt laten kiezen uit delen van het programma.
Het verwerken van de instructies gaat, in vergelijking met echte computers, niet zo snel, maar dat is voor een demo-programma als `ELGAMAL` geen probleem.
Als je een wat groter programma schrijft, kom je erachter dat het invoeren niet heel prettig is, omdat je de instructies niet rechtstreeks kunt intypen op een PC, maar allemaal moet ingeven via menu's. Wat ik zelf de lastigste beperking vond, was het ontbreken van een subroutine-mechanisme. Een `GoSub` instructie laat de rekenmachine naar een plaats in het programma springen, maar onthoudt ook, *van waar* werd gesprongen. Een bijbehorende instructie `End` of `Return` laat de rekenmachine dan terugkeren naar die plaats. Omdat dit mechanisme in de `TI-83+` ontbreekt, word je gedwongen om berekeningen die op meerdere plaatsen in je programma voorkomen (zoals machtsverheffen in `ELGAMAL`), ook meerdere keren compleet uit te schrijven.

4 Ontsleutelen met een team

Als je zomaar een willekeurig getal b als publieke sleutel instelt en een bericht versleutelt... kun je het wel vergeten dat je dat bericht ooit weer kunt ontsleutelen. Niemand kent de bijbehorende a of kan die uitrekenen.

Je kunt de sleutel ook zo kiezen, dat het bericht wel kan worden ontsleuteld als twee of meer mensen samenwerken.

4.1 Een gedwongen samenwerking

Wat gebeurt er als we van twee personen, Alice en Anton, de public key nemen en het *product* instellen als public key?

Opdracht 4.1 *Laat twee personen een Elgamal sleutelpaar maken (*Elgamal code, Sleutel maken*) en de publieke sleutels b_1 en b_2 vertellen.*
Ga nu naar Invoer public *en geef $b_1 \times b_2$ als "haar B" op. Versleutel een bericht (*Versleutel*).*

Om het bericht weer te ontsleutelen moeten we de logaritme van $b = b_1 \cdot b_2$ kennen. Maar het is niet mogelijk, die uit b te berekenen, en niemand kent dat getal!

Wel kent iedereen de geheime sleutel die bij zijn publieke hoort: Alice kent a_1 waarvoor geldt $g^{a_1} = b_1$, en Anton kent a_2 waarvoor geldt $g^{a_2} = b_2$. Uit deze twee gegevens volgt: $g^{a_1+a_2} = b$, kortom, de sleutel die we voor het ontsleutelen nodig hebben is $a = a_1 + a_2$. Deze sleutel is niet te berekenen uit de publieke sleutel b, maar wel uit de geheime sleutels van Alice en Anton.

Opdracht 4.2 *Laat Alice en Anton hun geheime sleutel vertellen, en voer (met* Sleutel maken*) de som in als geheime sleutel. Controleer dat je nu het bericht, versleuteld met het product van de publieke sleutels, weer kunt ontsleutelen.*

Je ziet dat *de som van geheime sleutels* past als geheime sleutel op *het product van publieke sleutels*. Ondertussen is het niet zo leuk voor Alice en Anton dat ze hun geheime sleutel hebben moeten vertellen. Want iedereen kan nu niet alleen dit speciale bericht, maar alle berichten lezen die ooit naar hun verstuurd zijn. Je kunt informatie niet "uitlenen" en "terugkrijgen": eens verteld, blijft verteld!

Gelukkig is er een truc, waardoor Alice en Anton met hun geheime sleutels kunnen samenwerken en het bericht ontsleutelen, zonder dat ze hun geheim aan iemand moeten vertellen. Om te begrijpen hoe dat kan, kijken we nog eens naar de formule die voor het ontsleutelen moet worden uitgerekend. Als a de geheime sleutel is, wordt het bericht (u, v) ontsleuteld door v te delen door u^a: $x = v \cdot u^{-a}$. Maar, met $a = a_1 + a_2$ kunnen we deze berekening herschrijven tot

$$x = (v \cdot u^{-a_2}) \cdot u^{-a_1} \ .$$

Alice en Anton kunnen na elkaar ontsleutelen met dezelfde waarde van u:

1. Anton ontsleutelt (u, v) en berekent $w = v \cdot u^{-a_2}$.
2. Alice krijgt w en ontsleutelt (u, w): $x = w \cdot u^{-a_1}$.

Probeer het maar eens! Misschien kun je nu wel bedenken hoe je zoiets kunt doen als er vijf of tien publieke sleutels met elkaar zijn vermenigvuldigd. Het kost wel veel tijd als alle personen *na elkaar* iets met het bericht moeten doen. In de volgende sectie laat ik zien dat je de berekening ook anders kunt organiseren, zodat de samenwerking sneller kan verlopen.

Kader 7: VOORBEELD VAN GROEPS-ELGAMAL

Als de opdrachten van paragraaf 4.2 niet lukken, kun je proberen de waarden hiernaast in te voeren. Het is een commissie van drie personen, Alice, Anton en Annet. In stap 1 maken ze ieder op hun rekenmachine een sleutelpaar (met PRIVESL MAKEN), en stap 2 combineert de deelsleutels tot één publieke sleutel voor het committee (met INV GROEPKEY).
In stap 3 wordt een bericht versleuteld (VERSLEUTEL). Bedenk dat er bij versleutelen een random getal wordt gekozen!! Je krijgt daarom (bijna zeker) een ander resultaat dan Bob, ook al versleutel je hetzelfde getal x met dezelfde sleutel b.
In stap 4 (met ONTSLEUTEL LID) gebruikt elk commissielid zijn eigen geheime sleutel om de bijdrage aan de gemeenschappelijke ontsleuteling te bepalen. Ben je niet zeker of je stap 4 goed doet, reken dan verder met de getallen uit dit voorbeeld. Bedenk wel, als je dit op minder dan drie rekenmachines doet, dat je voor elke handeling van Alice, Anton of Annet de betreffende geheime sleutel weer moet instellen met PRIVESL MAKEN! In deze stap worden geen random getallen gemaakt, dus als je de getallen uit het voorbeeld invoert, moet je ook dezelfde getallen eruit krijgen!
Stap 5 (ONTSL LEIDER) combineert de deelberekeningen van de leden. Als het jullie lukt om er 888 uit te krijgen, kunnen jullie stappen 4 en 5 herhalen met je eigen resultaat van stap 3.

Stap	Wie	Invoer	Resultaat
1	Alice	$a_1 = 2013$	$b_1 = 28848$
	Anton	$a_2 = 928$	$b_2 = 4170$
	Annet	$a_3 = 3718$	$b_3 = 80851$
2	samen	$k = 3$	
		$b_1 = 28848$	
	(of	$b_2 = 4170$	
	Bob)	$b_3 = 80851$	$b = 78212$
3	Bob	$x = 888$	$u = 89164$
			$v = 77469$
4	Alice	$u = 89164$	$z_1 = 54325$
	Anton	$u = 89164$	$z_2 = 93749$
	Annet	$u = 89164$	$z_3 = 54325$
5		$v = 77469$	
		aantal 3	
		$z_1 = 54325$	
		$z_2 = 93749$	
		$z_3 = 54325$	$x = 888$

4.2 Ontsleutelen zonder geheime sleutel?

Je kunt de Elgamal code zo gebruiken, dat een bericht alleen ontsleuteld kan worden door een aantal personen samen. Versleutelen kan wel door een persoon alleen worden gedaan. Het is niet zo moeilijk te begrijpen hoe dit werkt, en als jullie over meerdere rekenmachines beschikken, kun je het gemakkelijk naspelen met het `ELGAMAL` programma.

Waarom mag niemand de sleutel weten? Je weet vast wel dat een bankkluis nooit door één persoon geopend kan worden. Er zijn bijvoorbeeld drie verschillende sleutels die alledrie nodig zijn, zodat alleen drie medewerkers samen de kluis kunnen openen. Zoiets kun je met berichten ook doen. Stel je eens voor, dat een paar personen samen in een

geheime commissie moeten zitten. Ze kunnen dan afspreken, dat ze berichten voor de commissie alleen lezen als ze er allemaal bij zijn. Met Elgamal kunnen ze ervoor zorgen, dat een bericht alleen dan gelezen *kan* worden. We gaan nu kijken hoe dat precies werkt.

Sleutelkeuze en versleutelen Elk commissielid maakt een eigen sleutelpaar, en dat gaat net zoals bij de gewone Elgamal code.

Opdracht 4.3 *Laat twee tot vier deelnemers de rol van commissielid spelen. Kies in het* HOOFDMENU *voor* GROEPS ELGAMAL. *Elk lid kiest* PRIVESL MAKEN *om een eigen sleutelpaar te maken. Net als bij de "gewone" Elgamal kun je zelf een geheime sleutel kiezen, of een 0 opgeven om een random getal als sleutel te kiezen. Het programma berekent de bijpassende publieke sleutel en toont de waarden a_i en b_i als* MIJN SECR *en* DEEL PUBL. *Elk lid mag zijn publieke sleutel aan iedereen bekend maken.*

De publieke sleutel van een lid kan worden gebruikt om op de gewone manier berichten naar dit lid te sturen. Die kent de bijpassende geheime sleutel en kan zo'n bericht alleen ontsleutelen. Het volgende wat we gaan doen is het berekenen en instellen van de publieke sleutel van de commissie als geheel.

Opdracht 4.4 *Kies* INV GROEPKEY *voor het invoeren van de groepssleutel. Je moet het aantal commissieleden opgeven, waarna je wordt gevraagd de individuele publieke sleutels op te geven (*IND PUB*). Dan verschijnt de publieke sleutel van de commissie (*GROEPS SLEUTEL*). De berekende sleutel wordt meteen als publieke sleutel onthouden.*

Als de publieke sleutel eenmaal is ingesteld, is er voor het versleutelen geen verschil meer met de gewone Elgamal.

Opdracht 4.5 *Versleutel een getal met de keuze* VERSLEUTEL *in het menu* GROEPS ELGAMAL.
Versleutel hetzelfde getal vanuit het menu ELGAMAL CODE *door eerst de commissiesleutel in te stellen als public key (met de keuze* INVOER PUBLIC*) en dan voor* VERSLEUTEL *te kiezen.*
Hoe kun je erachter komen of de twee codeberichten wel echt het opgegeven getal coderen?

Ontsleutelen. De geheime sleutel die hoort bij b is een getal a waarvoor geldt $g^a = b$. Omdat $b = b_1 \cdot \ldots \cdot b_k$, en $b_i = g^{a_i}$, geldt

$$b = g^{a_1 + \ldots + a_k} \ ,$$

dus het getal $a = a_1 + \ldots + a_k$ is de geheime sleutel die we nodig hebben. Maar dat getal kent niemand!

Opdracht 4.6 *Als de hele commissie bijeen is, kunnen ze a uitrekenen. Maar als er één lid ziek is, kunnen de andere leden dan a "ongeveer" uitrekenen?*

We zullen nu gaan zien, hoe het bericht ontsleuteld gaat worden, waarbij het *wel nodig is* dat alle leden meedoen, maar *niet* dat ze a echt uitrekenen of iemand zijn geheime sleutel hoeft te laten zien. Wat we met het getal a willen uitrekenen is de waarde $v \cdot u^{-a}$; dit is de gebruikelijke Elgamal ontsleutelformule. Gebruikend dat $a = a_1 + \ldots + a_k$, volgt dat

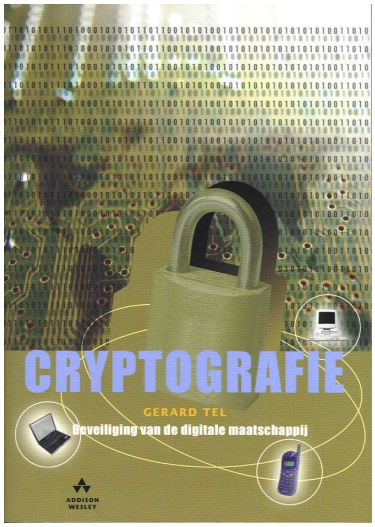
$$x = v \cdot u^{-a_1} \cdot \ldots \cdot u^{-a_k} \ .$$

Deze formule suggereert een mooie verdeling van de berekening over de commissieleden. Elk lid berekent alleen met u en de eigen geheime sleutel a_i de waarde $z_i = u^{-a_i}$ uit. De uiteindelijke vermenigvuldigingen worden door een willekeurig persoon uitgevoerd.

Opdracht 4.7 *Laat elk commissielid (in het menu* Groeps Elgamal*) kiezen voor* Ontsleutel lid*. Na het invoeren van u verschijnt als* Deel z *de waarde u^{-a_i}.*
Een persoon (dit mag een commissielid zijn) kiest Ontsl leider*, voert v, de commissiegrootte, en de resultaten van de vorige stap in.*

Tenslotte...

Je bent nu aan het eind van deze workshop. Misschien heb je nog wel ideeën die je zelf op een rekenmachine wilt uitproberen. Als je nog meer wilt lezen over cryptografie, kun je het boek hiernaast opzoeken [Tel02]. Het heet *Cryptografie: Beveiliging van de digitale maatschappij* en het is geschreven door Gerard Tel in 2002. De uitgever is Pearson Education.



Referenties

[ElG85] ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. In proc. *Advances in Cryptology: Proceedings of Crypto '84* (Berlin, 1985), G. R. Blakley and D. Chaum (eds.), Springer-Verlag, pp. 10–18. Lecture Notes in Computer Science Volume 196.

[Stu05] Stulens, K. Kraak de code. Tech. rep., Universiteit Hasselt, 2005. `www.scholennetwerk.be`.

[Tel02] Tel, G. *Cryptografie: Bescherming van de Digitale Maatschappij.* Addison-Wesley, 2002 (363 pp.).

Inhoudsopgave

A Programmabeschrijving

In deze appendix volgt een technische beschrijving van het programma `ELGAMAL` dat door Gerard Tel is geschreven voor de `TI-83+` rekenmachine.

Als je `Elgamal` opstart, kom je binnen in het hoofdmenu ELGAMAL HOOFD. Met CAESAR CODE kun je een heel eenvoudige code (uit paragraaf 2.1) uitproberen. De keuze ELGAMAL CODE laat je rekenen met het maken en gebruiken van sleutels. De keuze GROEPS ELGAMAL is voor ontsleutelen met een gedeelde sleutel. Met LOG EN MACHT kun je experimenteren met algoritmen om machten en logaritmen te berekenen. Met PARAMETERS kun je verschillende instellingen bekijken en experimenteren met waarden van p en g. Met STOP sluit je het programma af.

Je hebt misschien al gemerkt dat de hoogste waarde die voor de modulus is toegestaan, 10.000.000 is. Bij een modulaire vermenigvuldiging $a \cdot b$ wordt het product eerst "gewoon" uitgerekend, en daarna wordt de rest bij deling door p bepaald. Dit product moet zonder afronding worden gerepresenteerd en mag dus hoogstens 14 cijfers hebben omdat de `TI-83+` rekent met getallen tot 14 cijfers.

A.1 Het menu CAESAR CODE

De Caesar code gebruikt voor versleutelen en ontsleutelen dezelfde sleutel, een getal z. Versleutelen is vermenigvuldigen met z, en ontsleutelen is delen door z, allebei berekend met resten. De modulus die hierbij wordt gebruikt is ingesteld als 95917.

De keuze SLEUTEL KIEZEN neemt een willekeurig getal voor z. De keuzes SLEUTEL INSTEL en SLEUTEL TONEN laten je een zelfgekozen sleutel invoeren, of de sleutel en de modulus zien. De vermenigvuldiging en deling worden gedaan als je VERSLEUTEL of ONTSLEUTEL kiest.

Omdat een deling niet zo gemakkelijk te programmeren is, wordt het ontsleutelen berekend door eenmaal te machtsverheffen. Dat kan omdat p een priemgetal is, en volgens de Kleine Stelling van Fermat geldt dan: $z^{p-1} = 1 \pmod{p}$. Dat betekent dat de deling x/z, oftewel $x \cdot z^{-1}$, uit te drukken is als $x \cdot z^{p-2}$.

Hoe werkt machtsverheffen in `ELGAMAL`? Als voorbeeld bespreken we hier het berekenen van $x \cdot z^{p-2}$, maar machtsverheffen komt op diverse plaatsen in het programma voor, al is het steeds met andere variabelen. Het machtsverheffen begint met de waarden van het grondtal z en de exponent $p-2$ op te slaan in de variabelen o, resp. e. De gewenste uitkomst $x \cdot z^{p-2}$ is nu gelijk aan $x \cdot o^e$, en omdat e een variabele is mogen we de waarde ervan wijzigen. Het zou heel prettig voor ons zijn als e gelijk was aan 0, want dan was $x \cdot o^e$ namelijk gewoon x.

Wat er nu steeds in de `While` lus gebeurt, is dat de waarde van e kleiner wordt gemaakt, *zonder dat* de waarde van $x \cdot o^e$ verandert! Als e *even* is, wordt e gehalveerd en o gekwadrateerd; dit is correct omdat $x \cdot (oo)^{e/2}$ weer gelijk is aan $x \cdot o^e$. Als e *oneven* is, wordt e verlaagd en x met o vermenigvuldigd; dit is correct omdat $(xo) \cdot o^{e-1}$ weer gelijk is aan $x \cdot o^e$. (Na elke vermenigvuldiging moet de rest bij deling door p worden bepaald, maar die bewerking is hier weggelaten.)

```
Z -> 0 ; P-2 -> E
While E>0
  If breukDeel(E/2)=0
  Then E/2 -> E
       0 0 -> 0
  Else E-1 -> E
       0 X -> X
  End
End
```

Als de `While` lus termineert is $x \cdot o^e$ dus *niet veranderd*, oftewel, nog steeds gelijk aan de gewenste waarde. Maar omdat nu $e = 0$, wordt x als uitvoer gegeven.

Hoe vaak wordt de lus doorlopen? Als we beginnen met e een getal van k bits lang (dus: kleiner dan 2^k), dan kan e hoogstens k maal worden gehalveerd voordat de waarde 0 wordt; het `Then` gedeelte wordt dus hoogstens k maal uitgevoerd. Als een oneven getal met 1 wordt verlaagd ontstaat een even getal; daarom wordt, tussen twee uitvoeringen van het `Then` gedeelte, het `Else` gedeelte hoogstens eenmaal uitgevoerd. Het aantal rondes van de lus is daarom hoogstens $2k$.

A.2 Het menu Elgamal code

Onder dit menu zijn de verschillende berekeningen van het Elgamal geheimschrift aan te roepen. Voor het wijzigen van de parameters (p, q, g) moet je via het menu Parameters.

Met Sleutel maken genereer je een paar bestaande uit een geheime en een publieke sleutel. Dit begint met het bepalen van de geheime sleutel, a; die kun je zelf kiezen en invoeren, maar als je een 0 invoert wordt door het programma een random getal tussen 1 en $q - 1$ gekozen. De publieke sleutel b wordt met machtsverheffen berekend: $b = g^a$. De sleutels worden opgeslagen in geheugenplaatsen A en B.

Met Invoer public stel je een getal in als publieke sleutel voor versleutelen. Je hoeft van de tegenpartij natuurlijk alleen de publieke sleutel op te geven. Omdat geheugenplaats B al bezet is met de eigen publieke sleutel, wordt de "vreemde" publieke sleutel in locatie C bewaard.

Met Versleutel moet je een getal opgeven, wat dan wordt versleuteld tot een getallenpaar. Uiteraard wordt hierbij de *vreemde* publieke sleutel, opgeslagen in geheugen C, gebruikt. De versleuteling kiest een random getal k, en berekent $u = g^k$ en $v = x \cdot c^k$ (zoals uitgelegd in hoofdstuk 3). Omdat bij de twee machten dezelfde exponent k wordt gebruikt, kunnen ze in één gecombineerde `While`-lus worden berekend.

Met Ontsleutel kun je een gecodeerd paar (u, v) weer terugrekenen tot een getal x. Zoals in hoofdstuk 3 is uitgelegd moet x worden berekend als $x = v/u^a$, maar om geen modulaire deling te hoeven programmeren wordt dit feitelijk gedaan als $v \cdot u^{-k}$. De waarde van k wordt modulo q genomen (omdat $u^q = 1$).

Met Sleutels zien kun je zien welke getallen als eigen en vreemde sleutel zijn ingesteld, en met Hoofdmenu ga je uiteraard weer naar het beginscherm van `ELGAMAL`.

Kader 8: Overzicht van menu-opties en bijbehorende labels

Dit overzicht toont de menustructuur van `ELGAMAL` en ook bij welke labels de diverse berekeningen in het programma te vinden zijn. Bijvoorbeeld, achter `Lbl FZ` in het programma vind je het Parameter menu, waar je als gebruiker kunt kiezen uit Maak p,q,g, Geef settings, etc, en dan het programma doorstuurt naar `Lbl FA`, respectievelijk `Lbl FE`, etc. Dat programmadeel eindigt steeds met een verwijzing terug naar het betreffende menu, bv. `Goto FZ`.

Behalve de genoemde labels zijn nog in gebruik:

`Lbl FY`: spring hierheen binnen parametergeneratie als q verlaagd moet worden.

`Lbl LX`: het trekken van de volgende randomgetallen binnen Pollard's logaritmen-algoritme.

Het programma begint (eerste instructie) met het instellen van de parameters en springt dan naar `Lbl A`.

Label	Menu of functie	Doorgaan naar	Label
A	Hoofdmenu	Caesar code	HZ
		Elgamal code	AZ
		Groeps Elgamal	DZ
		Log en macht	LZ
		Parameters	FZ
		Stop	GZ
AZ	Elgamal code	Sleutel maken	AA
		Invoer public	AB
		Versleutel	AC
		Ontsleutel	AD
		Sleutels zien	AE
DZ	Groeps Elgamal	Privesl maken	DA
		Inv groepkey	DB
		Versleutel	DC
		Ontsleutel lid	DD
		Ontsl leider	DE
		Sleutels zien	DF
FZ	Parameter	Maak P,Q,G	FA
		Geef settings	FE
		Over programma	FF
LZ	Log en macht	Holemens log	LA
		Pollard rho	LB
		Machtsverhef	LC
		Instellen	LD
GZ	Echt stoppen?	Ja	GY
		Nee	A
HZ	Caesar code	Sleutel kiezen	HA
		Sleutel instel	HB
		Sleutel tonen	HE
		Versleutel	HC
		Ontsleutel	HD

A.3 Het menu Groeps Elgamal

Hier vind je de algoritmen om te ontsleutelen met een verdeelde sleutel, zoals uitgelegd in paragraaf 4.

Met Privesl maken kan een groepslid een sleutelpaar maken; het geheime deel ervan wordt onthouden in geheugen A. Deze functie is gelijk aan Sleutel maken onder het gewone Elgamal code menu. Met Invoer groeps wordt de publieke sleutel van de

Kader 9: Gebruik van variabelen in ELGAMAL

Wil je ooit dingen toevoegen of wijzigen in het programma, dan moet je oppassen om geen variabelen te gebruiken waarvan de waarde later nog belangrijk is. Daarom is hier een overzicht van het gebruik van de variabelen.

Var	Omschrijving
A	De eigen geheime sleutel
B	De eigen publieke sleutel
C	De vreemde publieke sleutel
D	Te testen delers in priem-tests
E	In machtsverheffen: dalende exponent
F	In machtsverheffen: groeiend grondtal
G	Generator cq. grondtal
H	
I	Telt twee iteraties in Pollard
J	
K	Random exponent in encryptie
L	Telt iteraties bij logaritmen
M	
N	
O	In machtsverheffen: groeiend grondtal
P	Modulus
Q	Orde van generator
R	Parametergeneratie: grens op Q
S	Parametergeneratie: grens op R
T	
U	Deel 1 van codebericht
V	Deel 2 van codebericht
W	Parametergeneratie: keuzes voor G
X	Bericht en output logaritme
Y	Bericht (in Caesar) en input logaritme
Z	Sleutel in Caesar, tussenresultaat Groeps Elgamal

groep ingesteld; je moet het aantal groepsleden opgeven, daarna zoveel getallen, die worden met elkaar vermenigvuldigd in geheugen C. De functie versleutel is weer gelijk aan de functie bij Elgamal code.

Voor ontsleutelen zijn er twee functies: Ontsleutel lid waarin de groepsleden hun privesleutels gebruiken (door bij invoer u de waarde u^{-a} op te leveren), en Ontsl leider waarin de deelresultaten worden gecombineerd (vermenigvuldig v met de deelresultaten van de groepsleden). Met Sleutels zien kun je de ingestelde sleutels bekijken.

A.4 Machten en logaritmes: Log en Macht

Onder de keuze Log en macht zijn drie rekenkundige algoritmes beschikbaar, waarmee je kunt experimenteren met de berekening van machten en logaritmen. Ze werken weer

met een grondtal p, een generator g en een orde q, en die kun je (nu willekeurig!) instellen met de keuze INSTELLEN.

Met MACHTSVERHEF kun je bij een getal x de waarde van g^x opvragen, waarbij ook wordt verteld, hoeveel vermenigvuldigingen het programma hiervoor gebruikt. Je kunt hierbij zien dat bij een exponent van k cijfers nooit meer dan $6.64k$ vermenigvuldigingen nodig zijn. Welk getal van 6 cijfers heeft de meeste nodig en waarom?

Met HOLEMENS LOG kun je van het resulterende getal de logaritme weer berekenen, maar het algoritme van de Holemens is veel te duur. Achtereenvolgens worden de waarden g^0, g^1, g^2, g^3, ... vergeleken met y, totdat gelijkheid wordt geconstateerd. Het zal duidelijk zijn dat, hoe groter de exponent x was die bij MACHTSVERHEF werd gebruikt, des te meer tijd de Holemens kwijt is om het weer terug te vinden.

Het algoritme van Pollard, dat je met POLLARD RHO kunt aanroepen, werkt veel slimmer. Er worden willekeurige getallen w getrokken, maar zo, dat er bij elk getal w ook getallen u en v bekend zijn waarvoor $w = y^u \cdot g^v$; die u en v heten een *representatie* van w. Hiermee gaat Pollard door, totdat er onder de trekkingen twee getallen "toevallig" aan elkaar gelijk zijn. Van dat getal zijn dan twee representaties bekend: $w = y^u \cdot g^v$ en $w = y^r \cdot g^s$. Uit die representaties is dan de logaritme te berekenen, want uit $y^u \cdot g^v = y^r \cdot g^s$ volgt $y = g^{\frac{s-v}{u-r}}$.

A.5 Het menu PARAMETERS

Hier kun je verschillende parameters bekijken die betrekking hebben op de Elgamal code. Met GEEF SETTINGS kun je de modulus p en het grondtal g zien, met een priemgetal q. Het drietal voldoet aan $g^q = 1 \pmod{p}$. Ook zie je de eventueel ingestelde eigen key (a en b, geheim en publiek deel) en een vreemde publieke sleutel xb.

Nieuwe parameters berekenen: MAAK P,Q,G Wil je uitgebreid gaan experimenteren met de Elgamal code, dan kun je met MAAK P,Q,G andere waarden voor de modulus en het grondtal instellen. Vanwege de relatie $g^q = 1$ die moet gelden, mogen dit geen willekeurige getallen zijn maar is hiervoor een berekening nodig.

Als je met een priem modulus rekent, heeft elk getal (behalve 0) een macht die gelijk is aan 1; kijk maar eens in Kader 1 en zie dat $3^6 = 1$, $6^2 = 1$, etc. Dat betekent, dat als p priem is, er bij elk getal g wel een q te vinden is waarvoor $g^q = 1$ geldt; die q heet de *orde* van g. Voor de Elgamal code is nodig, dat die q een priemgetal is. Als je geschikte p, q en g wilt uitrekenen, moet je eerst een grens op q en p geven, waarmee je aangeeft hoe groot je die getallen ongeveer wilt hebben.

- *Het vinden van de orde:* Vanaf de opgegeven grens worden oneven getallen getest of ze priem zijn. Dat gebeurt door ze te delen door alle oneven getallen kleiner dan hun wortel; als een deler wordt gevonden, probeert het programma een kleinere waarde van q.

- *Het vinden van de modulus:* De modulus moet een priemgetal zijn, en om straks

een orde te kunen vinden zorgen we ervoor dat q een deler is van $p-1$. Het programma probeert daarom oneven q-vouden plus 1, beginnend bij de opgegeven grens. Voorbeeld: als $q = 11$ en de grens op p is 170, dan probeert het programma achtereenvolgens 155, 133, 111, etc. Als geen priemgetal gevonden wordt (het programma komt dan uit bij 1), wordt een lagere waarde van q geprobeerd.

- *Het vinden van de generator:* We hebben nu priemgetallen p en q, en zoeken een getal g dat orde q heeft. Volgens de kleine stelling van Fermat (vraag hier je wiskundeleraar maar eens naar!) geldt, als je modulo p rekent, voor elk getal w, dat $w^{p-1} = 1$. Omdat q een deler is van $p-1$ kun je $p-1$ schrijven als $p-1 = r \cdot q$. Dan zie je dat voor elk getal w geldt: $w^{r \cdot q} = 1$, ofwel: $w^{rq} = 1$. Het programma neemt voor w getallen vanaf 100 en neemt daarvan de macht $(p-1)/q$. In het (weinig voorkomende) geval dat het resultaat 1 is, wordt de volgende waarde van w geprobeerd.

Met de maximale grenzen van 5.000.000 op q en 1.000.0000 op p vind je $q = 4.999.523$ en $p = 9.999.047$ (het duurt wel een paar minuten).

Andere opties onder PARAMETERS Met de keuze OVER PROGRAMMA kun je zien welke versie van het programma je hebt. Je kunt ook naar het HOOFDMENU of rechtstreeks naar ELGAMAL CODE.

A.6 Het menu STOP

Tja, wat moet ik daar over zeggen.... Als je deze optie hebt gekozen kun je daarna nog een keer kiezen of je echt wilt stoppen of terugkeren naar het hoofdmenu.

A.7 Programmalisting

Hier volgt de volledige lijst van instructies (programmaversie: 2 februari 2006):

```
95917->P
7993->Q
29609->G
0->A
0->B
0->C
3->Z
Goto A

Lbl A
Menu("HOOFDMENU",
"CAESAR CODE",HZ,
"ELGAMAL CODE",AZ,
"GROEPS ELGAMAL",DZ,
"LOG EN MACHT",LZ,
"PARAMETERS",FZ,
"STOP",GZ)

Lbl AZ
Menu("ELGAMAL CODE",
"SLEUTEL MAKEN",AA,
"INVOER PUBLIC",AB,
"VERSLEUTEL",AC,
"ONTSLEUTEL",AD,
"SLEUTELS ZIEN",AE,
"HOOFDMENU",A)

Lbl AA
WisBasisscherm
Toon "SLEUTEL MAKEN"
Toon "GEEF A (0 RAND)"
Input A
If A=0:toevalGeheel(1,Q-1)->A
A->E:G->O:1->B
While E>0
If breukDeel(E/2)=0
Then
E/2->E
O^2->O
O-geheel(O/P)*P->O
Else
E-1->E
BO->B
B-geheel(B/P)*P->B
End
End

WisBasisscherm
Uitvoer(1,1,"SLEUTEL GEMAAKT:")
Uitvoer(2,1,"SECRET: ")
Uitvoer(2,9,A)
Uitvoer(3,1,"PUBLIC: ")
Uitvoer(3,9,B)
Pauze
Goto AZ

Lbl AB
WisBasisscherm
Toon "INVOER VREEMDE"
Toon "PUBLIC KEY"
Toon "GEEF HAAR B:"
Input C
C-geheel(C/P)*P->C
Toon "INGESTELD:",C
Pauze
Goto AZ

Lbl AC
WisBasisscherm
Toon "VERSLEUTEL"
Toon "MET HAAR KEY"
Input ("GEEF X:",X)
X-geheel(X/P)*P->X
toevalGeheel(1,Q-1)->K
G->O:C->F
1->U:X->V
While K>0
If breukDeel(K/2)=0
Then
K/2->K
O^2->O
O-geheel(O/P)*P->O
F^2->F
F-geheel(F/P)*P->F
Else
K-1->K
```

```
U*O->U
U-geheel(U/P)*P->U
V*F->V
V-geheel(V/P)*P->V
End
End

Uitvoer(5,1,"U= ")
Uitvoer(5,4,U)
Uitvoer(6,1,"V=")
Uitvoer(6,4,V)
Pauze
Goto AZ

Lbl AD
WisBasisscherm
Toon "ONTSLEUTEL"
Input ("GEEF U:",U)
Input ("GEEF V:",V)
U-geheel(U/P)*P->U
V-geheel(V/P)*P->V
(-)A->K
K-geheel(K/Q)*Q->K
V->X
While K>0
If breukDeel(K/2)=0
Then
K/2->K
U^2->U
U-geheel(U/P)*P->U
Else
K-1->K
XU->X
X-geheel(X/P)*P->X
End
End

Uitvoer(6,1,"X IS ")
Uitvoer(6,6,X)
Pauze
Goto AZ

Lbl AE
WisBasisscherm
Uitvoer(1,1,"INGESTELDE KEYS")
Uitvoer(3,1,"MIJN PUB")
Uitvoer(3,10,B)
Uitvoer(4,1,"MIJN SEC")
Uitvoer(4,10,A)
Uitvoer(6,1,"HAAR PUB")
Uitvoer(6,10,C)
Pauze
Goto AZ

Lbl DZ
Menu("GROEPS ELGAMAL",
"PRIVESL MAKEN",DA,
"INV GROEPKEY",DB,
"VERSLEUTEL",DC,
"ONTSLEUTEL LID",DD,
"ONTSL LEIDER",DE,
"SLEUTELS ZIEN",DF,
"HOOFDMENU",A)

Lbl DA
WisBasisscherm
Toon "PRIVE KEY MAKEN"
Toon "GEEF A (0 RAND)"
Input A
If A=0:toevalGeheel(1,Q-1)->A
A->E:G->O:1->B
While E>0
If breukDeel(E/2)=0
Then
E/2->E
O^2->O
O-geheel(O/P)*P->O
Else
E-1->E
B*O->B
B-geheel(B/P)*P->B
End
End

WisBasisscherm
Uitvoer(1,1,"PRIVE SLEUTEL")
```

```
Uitvoer(2,1,"MIJN SECR:")
Uitvoer(2,11,A)
Uitvoer(3,1,"DEEL PUBL:")
Uitvoer(3,11,B)
Pauze
Goto DZ

Lbl DB
WisBasisscherm
Toon "INVOER GROEPS"
Toon "PUBLIC KEY"
Input ("AANTAL: ",K)
1->C
While K>0
Input ("IND PUB: ",D)
C*D->C
C-geheel(C/P)*P->C
K-1->K
End
Toon "GROEPS SLEUTEL:",C
Pauze
Goto DZ

Lbl DC
WisBasisscherm
Toon "VERSLEUTEL"
Toon "MET GROEP KEY"
Input ("GEEF X: ",X)
toevalGeheel(1,Q-1)->K
G->O:C->F
1->U:X->V
While K>0
If breukDeel(K/2)=0
Then
K/2->K
O^2->O
O-geheel(O/P)*P->O
F^2->F
F-geheel(F/P)*P->F
Else
K-1->K
U*O->U
U-geheel(U/P)*P->U
V*F->V
V-geheel(V/P)*P->V
End
End
Uitvoer(5,1,"U = ")
Uitvoer(5,5,U)
Uitvoer(6,1,"V = ")
Uitvoer(6,5,V)
Pauze
Goto DZ

Lbl DD
WisBasisscherm
Toon "ONTSLEUTEL LID"
Input ("GEEF U:",U)
(-)A->E
E-geheel(E/Q)*Q->E
1->Z
While E>0
If breukDeel(E/2)=0
Then
E/2->E
U^2->U
U-geheel(U/P)*P->U
Else
E-1->E
Z*U->Z
Z-geheel(Z/P)*P->Z
End
End
Uitvoer(6,1,"DEEL Z IS ")
Uitvoer(6,11,Z)
Pauze
Goto DZ

Lbl DE
WisBasisscherm
Toon "ONTSL LEIDER"
Input ("GEEF V: ",V)
Input ("AANTAL LEDEN: ",K)
V->X
While K>0
Input ("DEEL Z: ",Z)
```

```
X*Z->X
X-geheel(X/P)*P->X
K-1->K
End
Toon "X IS:"
Toon X
Pauze
Goto DZ

Lbl DF
WisBasisscherm
Uitvoer(1,1,"INGESTELDE KEYS")
Uitvoer(3,1,"DEEL PUB")
Uitvoer(3,11,B)
Uitvoer(4,1,"DEEL SEC")
Uitvoer(4,11,A)
Uitvoer(6,1,"GROEP PUB")
Uitvoer(6,11,C)
Pauze
Goto DZ

Lbl FZ
Menu("PARAMETER",
"MAAK P,Q,G",FA,
"GEEF SETTINGS",FE,
"OVER PROGRAMMA",FF,
"HOOFDMENU",A)


Lbl FA
WisBasisscherm
0->R
Toon "MAAK P,Q,G "
While R<3 of R>5000000
Toon "GRENS OP Q"
Input R
If R=0:Goto FZ
If R<3:Toon "MINSTENS 3"
If R>5000000:Toon "MAX 5000000"
End
Toon "GRENS OP P:"
10000001->S
While S>10000000
Toon "MAX 10000000"
Input S
End

Lbl FY
2*geheel((R-1)/2)+1->R
root(R)->W
Toon "TEST Q OP PRIEM: ",R
3->D
0->Q
While Q=0
If D>W
Then
R->Q
Else
If breukDeel(R/D)=0
Then
R-2->R
root(R)->W
Toon "TEST Q OP PRIEM: ",R
3->D
Else
D+2->D
End
End
End
Toon "PRIEM Q: ",Q
S->R
2Q*geheel((R-1)/(2Q))+1->R
root(R)->W
Toon "TEST P OP PRIEM: ",R
3->D
0->P
While P=0
If D>W
Then
R->P
Else
If breukDeel(R/D)=0
Then
R-2Q->R
root(R)->W
Toon "TEST P OP PRIEM: ",R
```

```
3->D
Else
D+2->D
End
End
End
If P=1
Then
Toon "NIET GESLAAGD"
Q-2->R
Goto FY
End
Toon "MODULUS P: ",P

1->G:100->W
While G=1
W+1->W
Toon "TEST",W
(P-1)/Q->E
W->O:1->G
While E>0
If breukDeel(E/2)=0
Then
E/2->E
O^2->O
O-geheel(O/P)*P->O
Else
E-1->E
G*O->G
G-geheel(G/P)*P->G
End
End
End
Toon "GENERATOR:",G
Goto FE


Lbl FE
WisBasisscherm
Toon "PARAMETERS:"
Uitvoer(2,1,"P: ")
Uitvoer(2,4,P)
Uitvoer(3,1,"Q: ")
Uitvoer(3,4,Q)
Uitvoer(4,1,"G: ")
Uitvoer(4,4,G)
Uitvoer(5,1,"A: ")
Uitvoer(5,4,A)
Uitvoer(6,1,"B: ")
Uitvoer(6,4,B)
Uitvoer(7,1,"XB:")
Uitvoer(7,4,C)

Pauze
Goto FZ

Lbl FF
WisBasisscherm
Toon "ELGAMAL"
Toon "DOOR GERARD TEL"
Toon "19 DEC 2005"
Pauze
Goto FZ

Lbl LZ
Menu("LOG EN MACHT",
"HOLEMENS LOG",LA,
"POLLARD RHO",LB,
"MACHTSVERHEF",LC,
"INSTELLEN",LD,
"HOOFDMENU",A)

Lbl LA
WisBasisscherm
Toon "HOLEMENS LOG"
Input "GEEF Y: ",Y
Y-geheel(Y/P)*P->P
Toon "BEREKEN LOG..."
0->X
1->O
While O!=Y
X+1->X
If X=Q
Then
Toon "LOG BESTAAT NIET"
Pauze
```

```
Goto LZ
End
O*G->O
O-geheel(O/P)*P->O
End
Toon "LOG Y IS ",X
Toon "VERMENIGV ",X
Pauze
Goto LZ

Lbl LB
WisBasisscherm
Toon "POLLARD RHO"
Input "GEEF Y: ",Y
Y-geheel(Y/P)*P->Y
Toon "BEREKEN LOG..."
0->L
0->U:0->R
While U=R
toevalGeheel(1,Q-1)->U
toevalGeheel(1,Q-1)->V

1->W
Y->F:U->E
While E>0
If breukDeel(E/2)=0
Then
E/2->E
FF->F
F-geheel(F/P)*P->F
Else
E-1->E
W*F->W
W-geheel(W/P)*P->W
End
End
V->E:G->F
While E>0
If breukDeel(E/2)=0
Then
E/2->E
F*F->F
F-geheel(F/P)*P->F
Else
E-1->E
W*F->W
W-geheel(W/P)*P->W
End
End
U->R:V->S:W->T
Lbl LX
L+1->L
W-geheel(W/3)*3->K
If K=0
Then
W*Y->W
W-geheel(W/P)*P->W
U+1->U
Else
If K=1
Then
W*G->W
W-geheel(W/P)*P->W
V+1->V
Else
W*W->W
W-geheel(W/P)*P->W
U+U->U
If U>=Q:U-Q->U
V+V->V
If V>=Q:V-Q->V
End
End

For(I,1,2)
T-geheel(T/3)*3->K
If K=0
Then
T*Y->T
T-geheel(T/P)*P->T
R+1->R
Else
If K=1
Then
T*G->T
T-geheel(T/P)*P->T
```

```
S+1->S
Else
T*T->T
T-geheel(T/P)*P->T
R+R->R
If R>=Q:R-Q->R
S+S->S
If S>=Q:S-Q->S
End
End

End

If W!=T:Goto LX
U-geheel(U/Q)*Q->U
R-geheel(R/Q)*Q->R
End
S-V->X
If X<0:X+Q->X
U-R->F
If F<0:F+Q->F
Q-2->E
While E>0
If breukDeel(E/2)=0
Then
E/2->E
F^2->F
F-geheel(F/Q)*Q->F
Else
E-1->E
X*F->X
X-geheel(X/Q)*Q->X
End
End
Toon "LOG Y IS ",X
Toon "ITERATIES ",L
Pauze
Goto LZ


Lbl LC
WisBasisscherm
Toon "MACHTSVERHEFFEN"
Input "GEEF X: ",X

0->L
1->Y:G->F:X->E
While E>0
L+1->L
If breukDeel(E/2)=0
Then
E/2->E
F*F->F
F-geheel(F/P)*P->F
Else
E-1->E
Y*F->Y
Y-geheel(Y/P)*P->Y
End
End
Toon
Toon "G^X = ",Y
Toon "VERMENIGV ",L
Pauze
Goto LZ

Lbl LD
WisBasisscherm
Toon "PARAMETERS"
Toon "INSTELLEN"
Input "MODULUS P: ",P
Input "GRONDTAL G: ",G
Input "ORDE Q: ",Q
Toon "INGESTELD"
Pauze
Goto LZ

Lbl GZ
Menu("ECHT STOPPEN?",
"JA",GY,"NEE",A)

Lbl GY
WisBasisscherm
Toon "TOT ZIENS"
Stop
```

```
Lbl HZ
Menu("CAESAR CODE",
"SLEUTEL KIEZEN",HA,
"SLEUTEL INSTEL",HB,
"SLEUTEL TONEN",HE,
"VERSLEUTEL",HC,
"ONTSLEUTEL",HD,
"HOOFDMENU",A)

Lbl HA
WisBasisscherm
Toon "SLEUTEL KIEZEN"
toevalGeheel(2,P-1)->Z
Uitvoer(3,1,"SLEUTEL:")
Uitvoer(3,10,Z)
Pauze
Goto HZ

Lbl HB
WisBasisscherm
Toon "SLEUTEL INSTEL"
Toon "GEEF SLEUTEL:"
Input Z
Z-geheel(Z/P)*P->Z
Toon "INGESTELD"
Pauze
Goto HZ

Lbl HE
WisBasisscherm
Uitvoer(1,1,"CAESAR CODE")
Uitvoer(2,1,"INSTELLINGEN")
Uitvoer(3,1,"SLEUTEL:")
Uitvoer(3,10,Z)
Uitvoer(4,1,"MODULUS: ")
Uitvoer(4,10,P)
Pauze
Goto HZ

Lbl HC
WisBasisscherm
Toon "VERSLEUTELEN"
Toon "GEEF BERICHT:"
Input X
X-geheel(X/P)*P->X
XZ->Y
Y-geheel(Y/P)*P->Y
Toon "CODEBERICHT:"
Toon Y
Pauze
Goto HZ

Lbl HD
WisBasisscherm
Toon "ONTSLEUTELEN"
Toon "CODEBERICHT:"
Input X
X-geheel(X/P)*P->X
Z->O:P-2->E
While E>0
If breukDeel(E/2)=0
Then
E/2->E
OO->O
O-geheel(O/P)*P->O
Else
E-1->E
OX->X
X-geheel(X/P)*P->X
End
End
Toon "BOODSCHAP:"
Toon X
Pauze
Goto HZ
```