In 1985 bedacht Taher Elgamal het naar hem genoemde algoritme om boodschappen te kunnen coderen. Ook rond die tijd werd de grafische rekenmachine ontwikkeld. Beide historische doorbraken werden door **Gerard Tel** verenigd in een workshop die hij onlangs op de NWD gaf.

De chocoladecode: Elgamal workshop

Inleiding

In dit artikel beschrijf ik materiaal voor een lespakket en interactieve workshop over het cryptografische algoritme van Elgamal, waarbij de deelnemers kunnen meerekenen en meedenken als ze een TI-83+ programmeerbare rekenmachine hebben. De workshop werd gehouden op de Nationale Wiskunde Dagen in 2006.
Veel leerlingen hebben belangstelling voor cryptografische algoritmen, zoals RSA of Caesars code, maar het is niet makkelijk om materiaal te vinden waarmee ze zelf aan de slag kunnen. Ik heb materiaal gemaakt dat is gebaseerd op het encryptie-algoritme van Elgamal. Het kan worden gebruikt als groepsactiviteit in een klas, maar leerlingen of groepjes kunnen ook zelf voor een werkstuk aan de slag met reken- of programmeerexperimenten.
Dit artikel beschrijft achtereenvolgens waaruit het pakket bestaat, de wiskundige achtergrond van Elgamals uitvinding, het Elgamal-algoritme zelf en de opzet van de workshop.

De Elgamal workshop

De Elgamal workshop legt het algoritme op een speelse en interactieve manier in ongeveer een uur uit. De deelnemers kunnen meerekenen als ze een TI-83+ rekenmachine hebben met het ELGAMAL-programma (van de website) erop. Als de workshopleider ruim van te voren aanwezig is met een TI-83+ met programma en snoertjes, kunnen in enkele minuten heel wat deelnemers van het programma worden voorzien. Het leukst is het als er per twee à drie deelnemers een rekenmachine is, maar een minimum is dat er drie machines zijn naast die van de workshopleider. Vooraf hang ik drie bordjes op met de letters C, G en B. Aan het begin vraag ik drie groepjes naar voren te komen en bij die bordjes te staan om de rollen van Julius Caesar (foto), generaal Gaius en de Barbaren te spelen. Na een korte blik op modulair rekenen en computers komen Caesar en Gaius erachter dat ze aan het Caesar-systeem (zie verderop) niet genoeg hebben, omdat de Barbaren meeluisteren als zij hun sleutel afspreken. Vervolgens voeren ze de Elgamal-sleutelgeneratie uit en versturen een bericht met Elgamal, waarbij de Barbaren alle uitgewisselde informatie zien, maar toch het geheim niet te weten komen. Alle handelingen voor de Caesar- en Elgamal-code worden direct op de TI-83+ uitgevoerd.

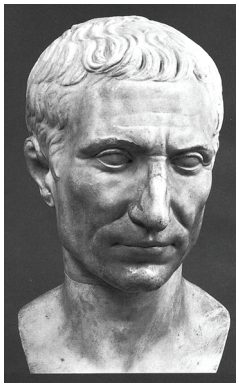


fig. 1 Julius Caesar

Een rekenvoorbeeld:

1. Caesar kiest als geheime sleutel het getal $a_1 = 299$ en vindt als publieke sleutel het getal $b_1 = 29090$. Vanaf nu is a_1 een soort pincode waarmee Caesar berichten kan lezen die zijn gecodeerd met het getal b_1, dat hij dan ook aan Gaius stuurt. Het getal a_1 vertelt hij niet!
2. Gaius kiest ook een geheime sleutel, a_2, en vindt een publieke sleutel $b_2 = 9814$, die hij aan Caesar vertelt. Omdat de Barbaren meeluisteren, kennen die nu de getallen b_1 en b_2, maar niet a_1 en a_2.
3. Om een bericht naar Caesar te sturen, moet Gaius de sleutel b_1 in de TI-83+ invoeren. Hierna geeft hij een getal x op en vindt het codebericht $(u, v) = (15563, 79792)$.
4. Omdat de bijbehorende geheime sleutel a_1 nog in Caesars TI-83+ zit, kan die het codebericht invoeren en het bericht $x = 888$ vinden.
5. De Barbaren zien alleen het codebericht (15563, 79792), maar kennen a_1 niet (alleen b_1) en kunnen de geheime boodschap niet lezen.

De deelnemers in de zaal kunnen met elkaar berichten uitwisselen met deze codes, of meerekenen met de groepjes voorin.

Achtergrond: logaritmeprobleem

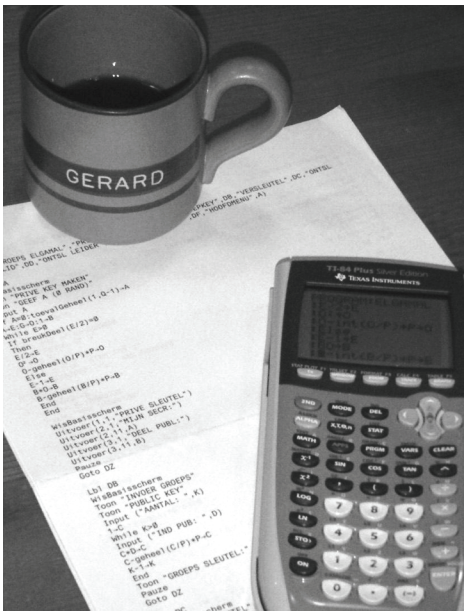


fig. 2 Gerard Tel

Vaak beperkt de uitleg van crypto-algoritmen zich ertoe te laten zien (meestal door een voorbeeld) dat versleuteling en ontsleuteling elkaars inverse zijn. Naar mijn mening is het voor een goed begrip van *public-key* crypto-algoritmen essentieel ook over de rekenkundige moeilijkheid van functies te spreken. Je moet laten zien dat de versleuteling en ontsleuteling (met kennis van de geheime sleutel) efficiënt uitvoerbaar zijn, maar dat het terugrekenen van de versleuteling zonder kennis van de geheime sleutel een te langdurige rekenkundige operatie is. Het Elgamal-algoritme biedt, omdat het op het logaritmeprobleem is gebaseerd, een iets gemakkelijker ingang in deze materie dan het bekendere algoritme RSA, dat op het factoriseren van grote getallen is gebaseerd.

Modulair rekenen

Het modulaire rekenen ('klokrekenen') is de belangrijkste voorkennis die deelnemers moeten hebben. Er wordt gerekend in de eindige groep van resten modulo een (groot) priemgetal, de verzameling $\mathbf{Z}_p$. In het programma heeft p de waarde 95917, maar hier rekenen we met $p = 37$ als klein voorbeeld. Er zijn dan maar 37 getallen (0 tot en met 36), en na elke rekenstap moet je de rest na deling door 37 nemen. Zo krijg je voor $20 \cdot 15$ eerst 300, maar $300 = 8 \cdot 37 + 4$, daarom schrijven we $20 \cdot 15 = 4$.
Door na elke tussenstap de rest te nemen, voorkom je dat je ooit met getallen groter dan 36^2 moet rekenen. Bijvoorbeeld, $11 \cdot 16 \cdot 21 \cdot 26$ is 96096, geeft bij deling door 37 rest 7. Maar ook: eerst $11 \cdot 16$ is 176, geeft rest 28, dan $28 \cdot 21$ is 588, geeft rest 33, tenslotte $33 \cdot 26$ is 858, geeft rest 7. Om het Elgamal-algoritme te begrijpen, moet je weten hoeveel werk het is om machten en logaritmen te berekenen.

Machtsverheffen

Machten worden uitgerekend door herhaald vermenigvuldigen. Als je aan leerlingen vraagt hoe vaak je moet vermenigvuldigen om 17^{21} te berekenen, krijg je als eerste antwoorden misschien 21 en 20. Maar als je uitlegt dat je rekenwerk kunt uitsparen door 17^{10} te kwadrateren, komen leerlingen al snel met een oplossing waarin maar zes keer wordt vermenigvuldigd:

(1) 17^2 is $17 \cdot 17 = 30$
(2) 17^4 is $17^2 \cdot 17^2 = 30 \cdot 30 = 12$
(3) 17^5 is $17^4 \cdot 17 = 12 \cdot 17 = 19$
(4) 17^{10} is $17^5 \cdot 17^5 = 19 \cdot 19 = 28$
(5) 17^{20} is $17^{10} \cdot 17^{10} = 28 \cdot 28 = 7$
(6) 17^{21} is $17^{20} \cdot 17 = 7 \cdot 17 = 8$

Omdat je de exponent in elke 'kwadrateerstap' (zoals stap 1, 2, 4 en 5) verdubbelt, is het aantal van die stappen hoogstens lg a om g^a uit te rekenen (de lg is de logaritme met grondtal 2). Omdat er tussen twee kwadrateerstappen geen of één extra stap zit, is het totale aantal vermenigvuldigingen tussen lg a en 2 lg a, met een gemiddelde van 1,5 lg a.

Logaritmen

Het logaritmeprobleem is het terugvinden van de exponent a bij gegeven grondtal g en resultaat y van de berekening $y = g^a$. Computers hebben weinig moeite met dit probleem in de reële getallen, omdat er van de ordening gebruik gemaakt kan worden. Is (bij $g > 1$) de waarde g^α groter dan y, dan zoeken we een $a < \alpha$ en als $g^\alpha < y$ dan is $a > \alpha$. Omdat in $\mathbf{Z}_p$ een (bruikbare) ordening op getallen ontbreekt, is er geen 'slimme' strategie om de logaritme snel te vinden.
Om logaritmen te berekenen in $\mathbf{Z}_p$, moet je een flink aantal machten en producten van g en y uitrekenen, totdat je twee resultaten ziet met dezelfde uitkomst. De methode van Shanks bijvoorbeeld, berekent log 8 door eerst de machten van g te vinden waarvan de exponent een zesvoud is. Dus, $17^6 = 27$, $17^{12} = 26$, etcetera, als in:

j	6	12	18	24	30	36
g^j	27	26	36	10	11	1

Vervolgens nemen we de invoer y en berekenen $y \cdot g$, $y \cdot g^2$ tot en met $y \cdot g^5$, als in:

i	0	1	2	3	4	5
$y \cdot g^i$	8	25	18	10	22	4

Je ziet in de tweede tabel één getal verschijnen dat in de eerste ook voorkomt, namelijk 10; dit noemen we een botsing. Blijkbaar is g^{24} gelijk aan $y \cdot g^3$, waaruit volgt dat $y = g^{21}$.

Hoe lang moet je rekenen om een botsing te laten voorkomen? Wanneer je willekeurig trekt (met teruglegging) uit een vaas met N ballen, krijg je een botsing (herhaalde bal) naar verwachting na ongeveer $\sqrt{2N}$ trekkingen. Hiermee hangt samen dat het aantal uit te rekenen getallen evenredig is met $\sqrt{a}$.

Omdat de functie lg a veel langzamer groeit dan $\sqrt{a}$, is het altijd mogelijk de gebruikte getallen zo groot te kiezen, dat machtsverheffen g^a wel, maar logaritmen log y niet te berekenen zijn. Bij de huidige stand van de techniek moeten hiervoor getallen van enkele honderden cijfers lang worden gebruikt! Het programma ELGAMAL.8xp kan getallen van zeven cijfers aan, en kan zowel machten als logaritmen uitrekenen.

Elgamal-encryptie

Een simpele manier van encryptie en decryptie (een rekenkundige variant van Caesars letterverschuiving) gebruikt een gemeenschappelijk getal z bij zender en ontvanger. De zender vermenigvuldigt met z, en de ontvanger deelt door z (deze berekeningen worden uitgevoerd met resten, dus in $\mathbf{Z}_p$):

$$\mathbf{Enc}_z(x) = x \cdot z$$
$$\mathbf{Dec}_z(y) = y \cdot z^{-1}$$

Zonder kennis van de sleutel z is het bericht y waardeloos, want v kan in $\mathbf{Z}_p$ door elk getal (ongelijk 0) worden gedeeld, en als resultaat is ook elk getal (ongelijk 0) mogelijk. Toch heeft dit systeem twee problemen, die beide door Elgamal werden opgelost.

1. Als meerdere berichten worden verzonden, kun je z niet hergebruiken, omdat dit informatie over de ratio tussen boodschappen laat lekken. Stel dat Gaius boodschappen x_1 en x_2 verzendt; de boodschappen zijn geheim en de Barbaren kennen eerst de verhouding x_1/x_2 niet. Maar Gaius verzendt boodschappen $y_1 = x_1 \cdot z$ en $y_2 = x_2 \cdot z$. Als de Barbaren die codeberichten zien, kunnen ze y_1/y_2 berekenen en vinden daarmee tegelijk x_1/x_2. Dat de Barbaren uit een reeks codeberichten gemakkelijk informatie over de geheime berichten kunnen uitrekenen, is een zwak punt van het systeem.
2. Zender en ontvanger moeten z vantevoren met elkaar afspreken, en je wilt in het internettijdperk ook veilig communiceren met nieuwe partners, bijvoorbeeld online winkels.

Elgamal lost het eerste probleem op door voor elk bericht (x) een nieuwe waarde voor z te laten kiezen door de verzender, dus bij versleuteling. Samen met het product $x \cdot z$ stuurt de verzender een hint u over de waarde van z, maar zo dat die hint alleen voor de beoogde ontvanger te begrijpen is. Dit is mogelijk omdat de ontvanger van te voren twee getallen heeft gekozen die samen een sleutelpaar vormen.

fig. 3 Taher Elgamal

Allereerst is er een groot priemgetal p bepaald dat als modulus wordt gebruikt, en een getal $g \in \mathbf{Z}_p$ dat het grondtal voor machtsverheffen is. De waarden p = 95917 en g = 29609 worden in het TI-83+-programma gebruikt; de getallen worden zo groot gekozen dat de logaritmen niet uitgerekend kunnen worden. Voor het maken van een sleutelpaar kiest de ontvanger een random getal a en berekent door machtsverheffen het getal $b = g^a$; het resultaat b is de public key van deze persoon. Voor iemand die de geheime sleutel a niet kent, is het onmogelijk die uit b te berekenen, omdat daarvoor een logaritme moet worden uitgerekend. Je kunt daarom je public key veilig aan iedereen vertellen die jou een bericht moet sturen.
Bij het versleutelen wordt z voor elk bericht nieuw gekozen als random macht van b: de zender kiest een random getal k en neemt $z = b^k$. De extra hint die wordt meegestuurd, geeft op een cryptische manier informatie over k en z; het is de waarde $u = g^k$.

$$\mathbf{Enc}_b(x) = \begin{cases} k = \text{random} \\ z = b^k \\ u = g^k;\ v = x \cdot z \\ \text{stuur } (u, v) \end{cases}$$

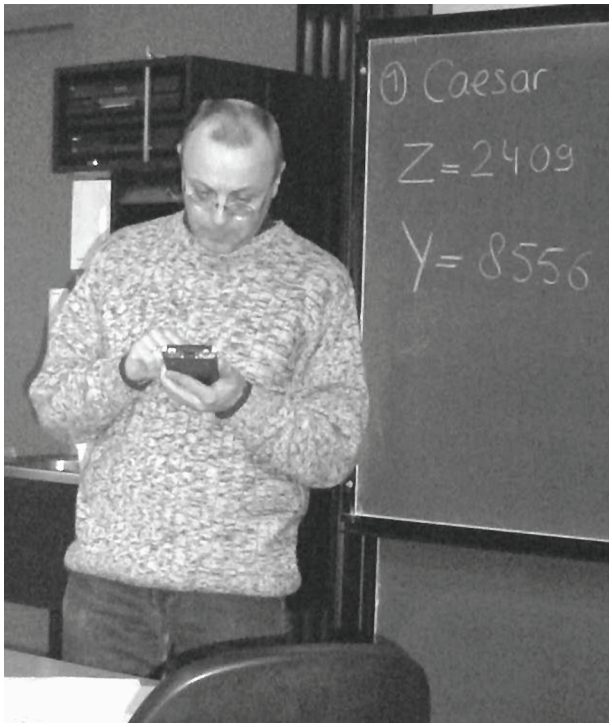
Net als in het eenvoudige systeem onthult v, het product $x \cdot z$, niets over x zolang z onbekend is. Maar wat zegt de hint u over z? Voor een onderschepper van het codebericht (u, v) is het jammer dat u de waarde van k wel uniek bepaalt, maar niet op berekenbare manier. Immers, k is de logaritme van u, en die is niet efficiënt berekenbaar.
Ook de houder van de geheime sleutel a kan k niet berekenen, maar hij kan z wel achterhalen via u, namelijk door u^a te berekenen. Immers, de zender berekende z als b^k en dat is gelijk aan $(g^a)^k$, en u is g^k dus u^a is $(g^k)^a$ en dat is ook gelijk aan $(g^a)^k$.

$$\mathbf{Dec}_a(u, v) = v \cdot u^{-a}$$

Wanneer je enkele malen hetzelfde getal x versleutelt (met dezelfde b) krijg je er toch steeds andere codegetallen uit. Dat komt omdat er een random getal k wordt gekozen, dat van keer tot keer verschilt. Die verschillende codegetallen geven wel allemaal hetzelfde resultaat bij ontsleutelen. Probeer maar!

Chocoladepuzzel

Na al die ingewikkelde uitleg is het tijd voor een puzzeltje voor alle deelnemers! Vantevoren heb ik tegen Caesar en Gaius gezegd, dat ze hun geheime sleutel nooit aan iemand mogen vertellen, op straffe van een chocoladereep als boete. Dan neem ik het product van hun twee publieke sleutels en gebruik dat als sleutel om de streepjescode van een reep chocola te versleutelen.



Eerder in dit artikel zagen we als voorbeeld van de publieke sleutels van Caesar en Gaius de getallen $b_1 = 29090$ en $b_2 = 9814$. De modulus is $p = 95917$, het product is dus $b = b_1 \cdot b_2 = 285489260 = 40268$.
Met 40268 als ingestelde publieke sleutel codeer ik de streepjescode en verkrijg $(u, v) = (3421, 74306)$. De puzzel is om de streepjescode (vier cijfers, beginnend met 2) hieruit te vinden en uiteraard is de reep de prijs. Tijdens het nadenken vertelde ik iets over verdere mogelijkheden van het algoritme, maar deelnemers lieten me achteraf weten dat ze liever wat tijd hebben om er echt rustig over na te denken.
Om de chocoladecode te ontrafelen moet je de logaritme van de gebruikte publieke sleutel 40268 weten, en meestal komt er wel iemand op het idee dat het hier gaat om de som van de geheime sleutels van Caesar en Gaius $(a_1 + a_2)$. Dus kan Caesar zijn geheim aan Gaius vertellen en die de reep laten incasseren, maar door het eerder ingestelde boetebeleid krijgt de workshopleider de reep dan van Caesar vergoed! Ik heb de workshop driemaal gehouden en iedere keer bedacht er wel iemand een juiste

manier om de code te kraken, maar slaagde er door fout intypen van getallen en berekeningen niet in het chocoladegetal echt te produceren.

Ten slotte...

Het Elgamal-algoritme zelf werd niet voor deze workshop uitgevonden, maar is al bekend sinds 1985 (ontwikkeld door de Egyptenaar Taher Elgamal). Het is rekenkundig: boodschappen en sleutels zijn allemaal getallen in de eindige groep $\mathbf{Z}_p$. Het modulair rekenen wordt als voorkennis verondersteld; mijn boek *Cryptografie: Bescherming van de Digitale Maatschappij* behandelt de nodige getaltheorie (bijvoorbeeld de stellingen van Fermat en Lagrange), maar dit is voor leerlingen misschien te hoog gegrepen. De uitleg van het RSA-cryptosysteem door Koen Stulens (*Kraak de code*) legt het klokrekenen uit, en ook hoe tekstboodschappen als getallen kunnen worden gecodeerd. Hoewel zijn workshop ook uitnodigt tot het werken met de TI-83+, heeft hij geen implementatie van het crypto-algoritme op de rekenmachine, maar wijkt hij daarvoor uit naar een wiskundepakket voor de PC.
Tot nu toe heb ik de workshop alleen gehouden met studenten van de Universiteit Utrecht en wiskundedocenten. Ervaringen met scholieren en suggesties voor verbetering of verduidelijking zijn daarom altijd welkom. Bij het schrijven van het pakket had ik de bovenbouw van het VWO als doelgroep voor ogen, maar het is mogelijk ook in andere schooltypen of leerjaren bruikbaar, als de docent meer uitlegt en begeleidt.

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

Literatuur

Stulens, K. (2005). *Kraak de code*. Tech. rep., Universiteit Hasselt. www.scholennetwerk.be.
Tel, G. (2002). *Cryptografie: Bescherming van de Digitale Maatschappij*. Amsterdam: Addison-Wesley