Toegepaste Wiskunde II
Cryptografie

Thomas Raes
75972

18 maart 2005

Inhoudsopgave

1 Inleiding

Cryptografie is de wetenschap van het beveiligen van communicatie. Oorspronkelijk werd enkel onderzocht hoe men boodschappen kon versleutelen. Vandaag houdt cryptografie zich ook bezig met het controleren van de identiteit van de afzender en de integriteit van de boodschap.

Informatie wordt gecodeerd en gedecodeerd met behulp van een algoritme en een sleutel. Wanneer men met dezelfde sleutel en bericht kan coderen en decoderen is het algoritme symmetrisch, anders is het asymmetrisch.

Cryptanalyse is de wetenschap die onderzoekt hoe men gecodeerde berichten kan kraken.

2 Symmetrische algoritmen

2.1 Algemeen

Wanneer een persoon A een bericht K wil versturen naar persoon B, spreken A en B eerst een sleutel S af. Persoon A berekent het gecodeerde bericht G met de coderingsfunctie c en de sleutel S ($G = c(K, S)$). Vervolgens verstuurt persoon A het gecodeerde bericht G naar persoon B. B decodeert G met de decoderingsfunctie d en de afgesproken sleutel S ($K = d(G, S)$).

Als een derde persoon de communicatie tussen de personen A en B afluistert, komt hij enkel het gecodeerde bericht G te weten. Zonder de sleutel S kan men het oorspronkelijke bericht K niet afleiden uit G.

Het voordeel van deze methode is dat ze vrij eenvoudig is, het nadeel is dat er eerst een sleutel moet worden afgesproken worden.

2.2 Substitutie

Bij een substitutie algoritme wordt elke letter vervangen door een andere letter. Als men over de sleutel beschikt weet men door welke letter elke letter vervangen is.

2.2.1 Mono-alfabetische substitutie

Als een letter altijd door dezelfde letter vervangen wordt spreekt men van een mono-alfabetisch substitutiecijfer. Men kan bijvoorbeeld elke letter vervangen door de letter die n plaatsen verder in het alfabet staat (Julius Caesar gebruikte $n = 3$ [7], ROT13 gebruikt $n = 13$ [6]). Op deze manier zijn er slechts 26 mogelijk sleutels. Het is ook mogelijk om elke letter te vervangen door een willekeurige andere letter, hierdoor zijn er 26! sleutels mogelijk.

2.2.2 Poly-alfabetische substitutie

Als een letter door verschillende letters vervangen kan worden spreekt men van een poly-alfabetisch substitutiecijfer. Men kan bijvoorbeeld het gebruikte alfabet laten afhangen van de positie van de te coderen letter in de boodschap.

Doordat er maar een beperkt aantal alfabetten zijn wordt elk alfabet verschillende keren gebruikt. Het aantal beschikbare alfabetten noemt men de periode.

Wanneer elke letter met behulp van een andere alfabet versleutelt is de periode even lang als het bericht, in dit geval spreekt men van een *one-time pad*. Wanneer men met binaire data werkt kan men de exclusieve disjunctie ($\oplus$) gebruiken om het bericht en de sleutel om te zetten in een gecodeerd bericht.

2.2.3 Cryptanalyse

Mono-alfabetische sustitutiecijfers kunnen gemakkelijk gebroken worden met behulp van frequentie-analyse. Men telt hoeveel keer elke letter voorkomt in het gecodeerde bericht, daarna kan men deze frequentie vergelijken met frequentietabellen om te achterhalen met welke letter de gecodeerde letter overeenkomt.

Poly-alfabetische substitutiecijfers waarbij de periode kleiner is dan de lengte van de boodschap kunnen op analoge wijze gekraakt worden vermits elk alfabet meerdere keren gebruikt wordt.

Een *one-time pad* is in theorie onkraakbaar, wanneer men alle mogelijke sleutels probeert bekomt men alle mogelijke boodschappen. Het nadeel van dit algoritme is dat de sleutel zeer willekeurige en even lang moet zijn als het te coderen bericht en dat elke sleutel hoogstens 1 keer gebruikt mag worden.

2.3 Transpositie

Transpositie coderingen worden soms ook permutatie coderingen genoemd. Bij een transpositie algoritme veranderen de letters van plaats. De sleutel bevat de nodige informatie om de letters terug op hun oorspronkelijke plaats te zetten.

In de praktijk gebruiken de meeste algoritmen een combinatie van transpositie en substitutie.

2.4 Enigma

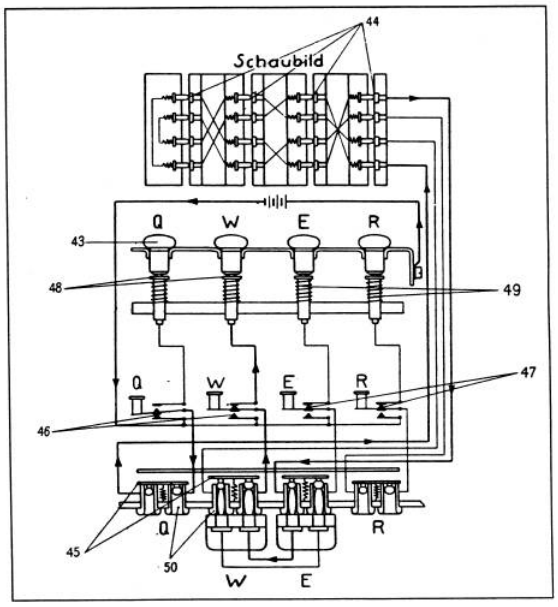
De Enigma werd tijdens de Tweede Wereldoorlog gebruikt door het Duitse leger om communicatie te versleutelen. De invoer werd ingetypt op een toetsenbord en de uitvoer werd weergegeven met behulp van lampjes.

Het versleutelen gebeurde met behulp van draaibare rotors en een stekkerpaneel. De sleutel bestond uit de beginstanden van de rotors en de plaatsingen van de stekkers op het paneel.

2.4.1 Werking

Rotors Elke rotor is een cilinder met aan elke kant metalen contactpunten (1 voor elke letter). Wanneer een elektrische stroom langs een contactpunt aan een zijde van de rotor binnenstroomt, stroomt ze buiten langs een contactpunt dat overeenstemt met een andere letter aan de andere zijder van de rotor. De interne bedrading van de rotor bepaalt in welke letter een bepaalde letter kon vervangen worden.

De oorspronkelijke versie (M3) bevatte 3 rotors, een geavanceerder type (M4) beschikte over 4 rotors. Voordat er een elektrische stroom door de rotors gestuurd werd, werd eerst de meest rechtse rotor 1 positie verdraaid. Wanneer



Figuur 1: Vereenvoudigde bedrading van Enigma M3 [11]

een rotor een bepaalde positie bereikte werd ook de volgende rotor tegelijk verdraaid. De positie waarbij een rotor zijn rotatie overdraagt op de volgende rotor is instelbaar en maakt deel uit van de sleutel.

Nadat de elektrisch stroom de 3 of 4 rotors doorlopen had werd ze door een ander type rotor teruggestuurd zodat de letter nogmaals versleuteld werd. Deze laatste rotor (*Umkehrwalze*) kan niet roteren en heeft slechts contactpunten aan 1 zijde.

Zowel de keuze van de te gebruiken rotors, de volgorde van hun plaatsing in het toestel, de beginstand en de positie waarop ze hun rotatie doorgeven aan de volgende rotor maken deel uit van de sleutel.

Stekkerpaneel Het stekkerpaneel is een bord waarop men met behulp van stekkers verschillende letters met elkaar kan verbinden. Wanneer een letter versleuteld is met behulp van de rotors wordt ze nogemaals versleuteld wanneer ze door de electrische stroom door de stekkers loopt. Vermits men ook het aantal gebruikte stekkers kan kiezen, zijn er veel mogelijke instellingen. De plaatsingen van de stekkers maken deel uit van de sleutel.

2.4.2 Coderen

Wanneer iemand een bericht wil coderen bepaalt hij eerst de instellingen van de rotors en het stekkerpaneel. Tijdens WO II verschilden deze per dag en stonden ze beschreven in een code-boek dat aan alle gebruikers werd uitgedeeld.

Vervolgens kiest hij willekeurige beginstanden voor de rotors, deze beginstanden kunnen worden voorgesteld door middel van letters.

De voorstelling van de gekozen beginstanden wordt versleuteld met de in het code-boek gevonden instellingen.

Daarna worden de rotors van het Enigma toestel terug ingesteld volgens de gekozen beginstanden en wordt het eigenlijke bericht gecodeerd en verzonden.

2.4.3 Decoderen

De ontvanger stelt eerst zijn toestel in volgens de in het code-boek vermelde instellingen. Dan decodeert hij met deze instellingen de ontvangen voorstelling van de door de zender gekozen beginstanden.

Vervolgens stelt hij de rotors van het Enigma toestel in volgens de gedecodeerde waarde van de ontvangen beginstanden en decodeert hij het eigenlijke bericht.

2.4.4 Cryptanalyse

Er zijn ongeveer $3 * 10^{23}$ mogelijke sleutels. Vermits er bij het uitbreken van WO II nog geen computers beschikbaar waren, was het geen eenvoudige opgave om de code te kraken.

Polen De eerste succesvolle pogingen om de Enigma code te kraken vonden plaats in Polen. Reeds vanaf 1928 luisterde de Poolse inlichtingendienst Duitse Enigma berichten af.

Poolse wiskundigen slaagden er in de interne werking van de Duitse Enigma te achterhalen op basis van door spionage verkregen informatie. Vervolgens ontwikkelde Marian Rejewski de *bomba kryptologicrna* (cryptologische bom) om Enigma berichten sneller te kunnen kraken.

In 1939 werd deze kennis overgedragen aan Frankrijk en Engeland.

Engeland In Engeland werd door de MI6 en Government Code & Cypher School (GC&CS) in Bletchley Park *Station X* opgericht. Hier werden op grote schaal Enigma berichten gekraakt. Er werkten 8 995 mensen, waaronder de wiskundige Alan Turing.

Turing verbeterde de Poolse bombe zodanig dat Enigma berichten sneller gekraakt konden worden. Na enige tijd werden er echter complexere coderingsalgoritmen in gebruik genomen (Enigma M4, Lorenz machine) zodat het kraken van berichten veel meer tijd in beslag nam.

Om deze complexere codes te kraken had men in Bletchley Park geavanceerdere middelen nodig. Daarom werd door Alan Turing de Colossus ontworpen, de eerste computer. De eerste Colussus werd in 1943 in gebruik genomen, in totaal werden er 10 exemplaren gebouwd.

Met behulp van deze Colussus computers en uit zinkende duikboten buitgemaakte Enigma toestellen was men in staat om gedurende de rest van het verloop van de oorlog Enigma berichten te kunnen kraken.

In 1954 pleegde Alan Turing zelfmoord, in 1960 werd de laatste Colussus vernietigd.

2.5 Data Encryption Standard (DES)

DES werd door IBM ontwikkeld en in 1977 als standaard aanvaard door de regering van de Verenigde Staten.

2.5.1 Werking

DES is een blokcijfer, dit betekent dat de te coderen data opgedeeld wordt in blokken van gelijke grote die onafhankelijk van elkaar kunnen worden versleuteld.

DES is een Feistel coderingsalgoritme. Een Feistel algoritme werkt in verschillende iteraties. Eerst wordt de data opgedeeld in een linkerdeel L^i en een rechterdeel R^i van gelijke grote. Bij elke iteratie i geldt $L^i = R^{i-1}$ en $R^i = L^{i-1} \oplus f(R^{i-1}, K^i)$ waarbij K^i de sleutel is. Het voordeel van Feistel algoritmen is dat de ze altijd inverteerbaar zijn, ongeacht de gekozen functie f. Bij het decoderen geldt immers: $L^{i-1} = R^i \oplus f(L^i, K^i)$ en $R^{i-1} = L^i$.

DES deelt de te coderen data op in blokken van 64 bit, de sleutel is 56 bit lang. Het algoritme gebruikt 16 iteraties (ook wel rondes genoemd). Tijdens elke ronde wordt het te verwerken blok in 2 delen van 32 bits opgesplitst.

Tijdens het coderen worden zowel substituties als permutaties uitgevoerd. De substituties gebeuren aan de hand van 8 tabellen (*S-boxes* genoemd in de specificatie), de permutaties gebeuren ook aan de hand van een tabel (*P-box* genoemd).

2.5.2 Operaties

De gebruikte operaties zijn:

Initiële permutatie De initiële permutatie heeft als doel om eenvoudig data in te laden als bytes. Deze stap heeft geen invloed op de veiligheid van de versleuteling maar had praktische voordelen bij het implementeren van het algoritme op oudere hardware.

Sleuteltransformatie Hoewel de door de gebruiker ingevoerde sleutel 64 bit lang is, worden slechts 56 bits door het algoritme gebruikt. De 8 overige bits kunnen gebruikt worden om fouten te detecteren. Uit deze 56 bit sleutel worden 16 verschillende 48 bit sleutels berekend, 1 voor elke ronde tijdens het versleutelen.

Expansie permutatie Bij het versleutelen worden de 64 bit datablokken opgedeeld in 2 blokken van 32 bit. De expansie permutatie zorgt er voor dat 1 van deze blokken uitgebreid wordt tot 48 bit. Dit gebeurt door bepaalde bits te herhalen. De volgorde van de bits wordt ook gewijzigd.

S-matrix substitutie De te versleutelen datablokken worden bewerkt met XOR operaties op de sleutelblokken. Nadien worden er substituties op uitgevoerd. Deze substituties gebeuren aan de hand van S-matrices (ook *S-boxes* genoemd). Er zijn 8 verschillende S-matrices.

P-matrix permutatie De bits van de uitvoer van elke S-matrix substitutie worden van volgorde verwisseld. De nieuwe volgorde van de bits staat beschreven in de P-matrix.

Laatste permutatie De laatste permutatie voegt geen extra veiligheid toe aan het algoritme. Deze operatie is louter uit praktische overwegingen toegevoegd, net als de initiële permutatie.

2.5.3 Cryptanalyse

Differentiële cryptanalyse Degene die de code wil kraken zoekt eerst naar verschillen tussen gecodeerde berichten (de uitkomst van de XOR operatie kan als verschil beschouwd worden). Met behulp van deze verschillen kan er aan elke sleutel een waarschijnlijkheid toegekend worden. Deze kansen kan men berekenen aan de hand van geobserveerde eigenschappen die informatie geven over de waarschijnlijkheid dat een verandering in de oorspronkelijke tekst een bepaalde wijziging in de gecodeerde tekst teweeg brengt. Hoe meer gecodeerde berichten men kan onderzoeken, hoe duidelijker de toegekende kans van een bepaalde sleutel zich zal onderscheiden van de kansen van alle andere sleutels.

De S-matrices van DES zijn ontworpen om differentiële cryptanalyse moeilijker te maken. Deze vorm van cryptanalyse heeft als nadeel dat men over zeer veel berichten moet beschikken.

Lineaire cryptanalyse Met deze methode probeert men met een lineaire benadering (XOR operaties) op de oorspronkelijke tekst en op de gecodeerde tekst delen van de sleutel te weten te komen. Deze methode geeft net zoals de differentiële cryptanalyse geen zekerheid.

Sleutels aflopen Wegens de beperkte sleutellengte is het mogelijk om alle sleutels uit te proberen binnen een redelijke tijd.

2.6 Advanced Encryption Standard (AES)

In 2001 werd het door V. Rijmen en J. Daemen ontwikkelde algoritme Rijndael als vervanger van DES gekozen. AES is een specifieke vorm van Rijndael waarbij de sleutels enkel 128, 192 of 256 bits lang moegen zijn en de data verwerkt wordt in blokken van 128 bits.

2.6.1 Werking

Net als DES is AES een blokcijfer, het deelt de invoer op in blokken van gelijke grootte en codeert deze blokken onafhankelijk van elkaar. Het aantal iteraties (rondes) hangt af van de lengte van de sleutel. Bij een 128 bit sleutel zijn er 10 rondes, bij een 192 bit sleutel zijn er 12 rondes en bij een 256 bit sleutel zijn er 14 rondes.

Bij het coderen worden er zowel substituties als permutaties uitgevoerd. De substituties gebeuren aan de hand van een tabel, *S-box* genaamd.

Elk blok (ook wel staat genoemd) bevat 16 bytes. Een byte

$$b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7$$

kan men interpreteren als de polynoom

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

.

De bytes kunnen beschouwd worden als elementen van een eindig veld, ook wel Galois veld genoemd. Vermits een byte 8 bits heeft en elke bit 2 mogelijke waarden kan hebben zijn er in totaal 2^8 mogelijke bytes. Rijndael maakt dus gebruik van een Galois veld met 2^8 elementen (notatie $GF(2^8)$).

Bewerkingen in $GF(2^8)$ kunnen efficiënt uitgevoerd worden op de binaire voorstellingen van de elementen.

De optelling van elementen in $GF(2^8)$ is equivalent aan de exclusieve disjunctie (XOR) van de overeenkomstige bits. De vermenigvuldiging van elementen in $GF(2^8)$ kan men implementeren met behulp van bitverschuivingen naar links. Wanneer men alle bits 1 plaats naar links opschuift, komt dit overeen met een vermenigvuldigen van de door de byte voorgestelde polynoom met x.

Wanneer men de elementen m en n vermenigvuldigt moet men voor elke bit met de waarde 1 in n een linker bitshift op m uitvoeren en het resultaat optellen bij de voorlopige uitkomst. Als de eerste bit van m 1 is krijgt men een overflow error, dit kan men oplossen door het resultaat van de linker bitshift op te tellen met de modulaire polynoom $x^8 + x^4 + x^3 + x^1 + x^0$ (binair voorgesteld als (1)00011011) vermits de berekening wordt uitgevoerd op $\mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x^1 + x^0)$.

2.6.2 Operaties

Op een staat zijn de volgende operaties gedefinieerd:

SubBytes Deze operatie voert een substitutie uit op de staat. Elke byte in de staat wordt vervangen door een andere. De vervangende byte kan worden opgezocht in een tabel (bekend als de *S-box*).

ShiftRows Deze operatie voert een transpositie uit op de staat. De bytes in de 2de rij van de staat worden 1 kolom naar links opgeschoven. De bytes in de 3de rij worden 2 kolommen naar links opgeschoven, en de bytes in de 4de rij worden 3 kolommen naar links opgeschoven.

MixColumns Deze operatie werkt onafhankelijk op elke kolom van de staat. In deze stap kan men elke kolom als een polynamiaal van de orde 4 beschouwen. Elke kolom wordt vervangen door zijn product met een matrix van eindige elementen in het eindige veld GF(256). Deze vermenigvuldiging is efficiënt te implementeren met behulp van bitshifts.

AddRoundKey Deze operatie is de enige die afhankelijk is van de gekozen sleutel. De staat wordt vervangen door het resultaat van de exclusieve disjunctie op de bits van de staat en een andere matrix van 128 bits. De gebruikte sleutel kan 128, 192 of 256 bits groot zijn. Hij wordt echter intern verlengd zodat elke keer dat AddRoundKey gebruikt wordt er een andere matrix wordt gebruikt om de staat te bewerken.

2.6.3 Coderen

De boodschap wordt opgedeeld in blokken van 16 bytes. Indien de boodschap niet exact in blokken van 16 bytes kan worden verdeeld, worden er tijdelijk extra bytes aan toegevoegd. De gegeven sleutel wordt verlengd en vervolgens wordt voor elke staat de volgende operaties uitgevoerd: Eerst wordt de operatie AddRoundKey uitgevoerd. Vervolgens worden tijdens verschillende rondes de operaties SubBytes, ShiftRows, MixColumns en AddRoundKey uitgevoerd. Tenslotte worden de operaties SubBytes, ShiftRows en AddRoundKey uitgevoerd. Elke staat wordt vervangen door de door bovenstaande operaties bekomen staat.

2.6.4 Decoderen

Het decoderen verloopt analoog aan het coderen. In plaats van SubBytes, ShiftRows en MixColumns worden hun respectievelijke inverse operaties InvSubBytes, InvShiftRows en InvMixColumns gebruikt. Om de inverse van deze operaties te bekomen volstaat het om enkel constanten te vervangen. De operatie AddRoundKey (eigenlijk een gewone *XOR* operatie) is zijn eigen inverse.

Het voornaamste verschil is de volgorde waarin de operaties worden uitgevoerd. Bij de rondes is dit: InvShiftRows, InvSubBytes, AddRoundKey en vervolgens InvMixColumns. Na het uitvoeren van de verschillende rondes worden nog InvShiftRows, InvSubBytes en AddRoundKey uitgevoerd.

2.6.5 Cryptanalyse

Tot nu toe zijn er geen algoritmen gekend die AES sneller kunnen kraken dan de tijd die nodig is om alle mogelijke sleutels te proberen.

2.7 Overige

DES en AES zijn de bekendste symmetrische algoritmen, er bestaan er echter nog andere soortgelijke algoritmen.

Er zijn verschillende varianten van DES zoals 3-DES, dat DES 3 keer toepast met verschillende sleutels. Bij een andere variant worden de *S-boxes* bepaald door de sleutel. Meer details en nog andere varianten op DES zijn te vinden in [6].

GOST (*Gosudarstvennyi Standard*) is een algoritme dat gebruikt werd in de voormalige Soviet Unie. Het gebruikt 64 bit blokken en is gebaseerd op DES. GOST heeft echter een 256 bit sleutel (DES heeft een 56 bit sleutel) en gebruikt 32 iteraties (DES gebruikt 16 rondes).

Andere algoritmen zijn CAST, Blowfish, Twofish en SkipJack.

3 Asymmetrisch algoritmen

3.1 Algemeen

Symmetrische algoritmen hebben als nadeel dat dezelfde sleutel zowel door de zender als de ontvanger gekend moet zijn. Asymmetrische algoritmen lossen dit probleem op door 2 sleutels te gebruiken, namelijk een sleutel S_e om een bericht

te coderen en een andere sleutel S_d om een versleuteld bericht te decoderen. Iedereen bezit een sleutelpaar (S_e, S_d) waarvan S_e door iedereen gekend mag zijn en S_d geheim gehouden moet worden. Daarom wordt S_e soms ook de publieke sleutel genoemd en S_d de private sleutel.

Wanneer persoon A een bericht K wil sturen naar persoon B, zoekt persoon A eerst de publieke sleutel S_{eB} van persoon B op. Vervolgens codeert hij K met het coderingsalgoritme c en de sleutel S_{eB} $(G = c(K, S_{eB}))$. Daarna verstuurt hij het gecodeerde bericht G naar persoon B. Persoon B kan het gecodeerde bericht G decoderen met behulp van zijn private sleutel S_{dB} $(K = d(G, S_{dB}))$.

Vermits de encryptiesleutel van de ontvanger publiek is kan iedereen berichten coderen, maar enkel degene die over de decryptiesleutel van het zelfde sleutelpaar beschikt kan de gecodeerde berichten decoderen. Op deze manier moet er dus nooit een geheime sleutel worden afgesproken.

Men kan ook de private sleutel gebruiken om een bericht te coderen. Op deze manier kan iedereen het bericht decoderen met behulp van de publieke sleutel van de verzender. Deze methode verzekert echter dat het bericht wel degelijk van de vermelde zender afkomstig is. Daarom wordt deze methode ook wil signeren van een bericht genoemd.

De moeilijkheid bij het ontwerpen van een asymmetrisch algoritme is het verband tussen S_e en S_d. De private sleutel moet de werking van de publieke sleutel opheffen $(K = d(c(K, S_e), S_d)$, maar het mag niet eenvoudig zijn om de private sleutel te berekenen uit de publieke sleutel. Met andere woorden, de functie f zodat $f(S_e) = S_d$ moet zo lang mogelijk duren om uit te rekenen. Daarom worden de relaties tussen S_e en S_b meestal gekozen op basis van problemen waarvan men verwacht dat ze enkel in niet-polynomiale tijd kunnen worden opgelost.

De theorie van asymmetrische algoritmen werd door Diffie en Hellman in 1976 bedacht. Het duurde nog een jaar voordat de eerste praktische implementatie ervan (RSA) uitgevonden werd. In 1997 beweerde de Britse regering dat ze reeds in 1970 over een dergelijk systeem beschikte, maar het altijd geheim gehouden heeft.

3.2 Rivest, Shamir, Adleman (RSA)

RSA werd in 1977 uitgevonden door R. Rivest, A. Shamir en L. Adleman. De sleutels zijn gebaseerd op priemgetallen. Het berekenen van S_d wanneer S_e gegeven is, is even moeilijk als het ontbinden van een getal in priemfactoren.

3.2.1 Genereren van een sleutelpaar

Men kiest eerst twee grote priemgetallen P en Q, het product van P en Q noemt men N. Vervolgens kiest men een getal L dat relatief priem is met $\Phi(N)$. Vermits P en Q priem zijn is $\Phi(N) = (P-1)(Q-1)$.

Men moet ook een waarde M kiezen zodat $LM \equiv 1(mod(\Phi(N)))$.

De publieke sleutel S_e is (L, N), de private sleutel S_d is M. Sleutel S_e mag publiek gemaakt worden, sleutel S_d moet geheim gehouden worden.

3.2.2 Coderen

Wanneer persoon A een bericht K wil versturen naar persoon B, zoekt hij eerst de publieke sleutel S_{eB} van persoon B op. Vervolgens codeert hij het bericht K tot het gecodeerde bericht G ($G = c(K, S_{eB})$) en verstuurt hij G naar persoon B.

3.2.3 Decoderen

Persoon B ontvangt het gecodeerde bericht G. Vermits persoon B over de private sleutel S_{dB} beschikt kan hij het bericht decoderen ($K = d(G, S_{dB})$).

3.2.4 Cryptanalyse

Men kan het bericht G enkel decoderen indien men beschikt over de private sleutel S_d. Indien men niet over deze sleutel beschikt moet men hem berekenen met behulp van de publieke sleutel $S_e(S_e = (L, N))$. Om S_d te bekomen moet men de waardes van P en Q achterhalen, dit kan men door N te factoriseren (want $N = PQ$). Er is voor het moment geen performant algoritme bekend om dit te doen.

3.3 Overige

RSA is het bekendste asymmetrische coderingsalgoritme. Er bestaan echter nog andere algoritmen. ElGamal is gebaseerd op het zoeken van discrete logaritmen in een eindig veld. Er zijn ook algoritmes die gebaseerd zijn op elliptische krommen.

4 Besluit

Er worden alsmaar betere methoden gevonden om data te beveiligen en te kraken. De veiligheid van elk coderingsalgoritme is gebaseerd op de moeilijkheid van een probleem.

Eerst waren dit mechanische problemen (bv. Enigma), vandaag worden vooral wiskundige problemen gebruikt. Momenteel wordt er geëxperimenteerd met het gebruik van problemen uit de fysica om communicatie te beveiligen [1].

Referenties

[1] Best-kept Secrets, G. Stix, Scientific American, January 2005

[2] A Specification for the AES Algorithm, Dr. B. Gladman, http://www.comms.scitech.susx.ac.uk/fft/crypto/aesspec.pdf

[3] AES JavaScript implementation, http://cs.eku.edu/faculty/styer/460/Encrypt/JS-AES.html

[4] The Code Book, S. Singh, 1999, Doubleday Random House Inc.

[5] Cryptography: Theory and Practice 2nd Edition, D. R. Stinson, 2002, CRC Press LLC

[6] Applied Cryptograpy, B. Schneier, 1996, John Wiley & Sons Inc.

[7] Codemaster, H. Nickels, 1990, Paladin Press Inc.

[8] Enigma: the Battle for the Code, H. Sebag-Montifiore, 2001, Orion Books Ltd.

[9] Discrete and Combinatorial Mathematics, R. P. Grimaldi, 2004, Pearson Education Inc.

[10] The New Turing Omnibus, A. K. Dewdney, 2001, Owl Books

[11] http://w1tp.com/enigma/mewirg.htm