Is RSA-cryptografie nu veilig genoeg en wat betekent dit voor de toekomst van digitale beveiliging?

Profielwerkstuk

Examenkandidaten:

Nahom Tsehaie ***(N&T en N&G)***

Jun Feng ***(N&T)***

Informatica en Wiskunde

Begeleiders:

David Lans

Albert van Westereenen

Samenvatting

In dit profielwerkstuk wordt de volgende hoofdvraag beantwoord:

Is RSA-cryptografie nu veilig genoeg en wat betekent het voor de toekomst van digitale beveiliging?

Daarnaast worden de volgende 6 deelvragen beantwoord:

a. Hoe heeft cryptografie zich ontwikkeld?
b. Welke soorten cryptografie bestaan er?
c. Wat is RSA en hoe werkt RSA?
d. Op welke gebieden is RSA inzetbaar?
e. Op welke manieren is het RSA te kraken?
f. Wat is de toekomst van cryptografie?

Deze vragen hebben wij kunnen beantwoorden door het maken van het profielwerkstuk en het maken van een RSA versleutelingsprogramma in Visual Basic.

Door het maken van dit profielwerkstuk hebben we bevonden dat in de loop van de tijd het versleutelen van berichten steeds geavanceerder is geworden. Deze geschiedenis begint bij de Romeinen en hun Caesar methode. Dit was een klassieke vorm van cryptografie. Tegenwoordig wordt veelal RSA als versleutelingsmethode gebruikt. De veiligheid van RSA berust op het probleem van de ontbinding in factoren bij heel grote getallen.
Ook is het duidelijk geworden welke soorten cryptografieën er bestaan. Er bestaan namelijk twee soorten cryptografieën: Klassieke en Moderne cryptografie. De moderne cryptografie kan onderverdeeld worden in asymmetrische en symmetrische cryptografie. De klassieke cryptografie kan onderverdeeld worden in substitutieversleuteling en transpositieversleuteling.
Daarna is bevonden dat RSA een cryptografiesysteem is, baanbrekend door het gebruik van asymmetrische cryptografie en eenrichtingsfuncties.
Tevens is nu bekend dat RSA op vele gebieden inzetbaar is, zelfs buiten het versleutelen van informatie.
Er zijn toch verschillende manieren gevonden om RSA te kraken, hoewel geen daarvan dat in een kort tijdsbestek kunnen.
De toekomst van cryptografie is erg veel belovend, omdat de druk op de cryptologische vooruitgangen erg groot is.

Door deze informatie en de opgedane ervaringen van het maken van het RSA-versleutel programma hebben wij onze hoofdvraag goed kunnen beantwoorden en een conclusie kunnen trekken:

RSA is op dit moment nog net veilig genoeg, zeker vergeleken met de oude cryptografie systemen die er waren voordat RSA in het leven was geroepen. Asymmetrische cryptografie is in het heden de beste techniek om mee te versleutelen, een techniek waar het veelgebruikte RSA onder valt. Doordat priemgetallen oneindig in aantal zijn, kan RSA altijd een stapje voor supercomputers zijn. RSA is echter niet heel efficiënt en er zijn al manieren gevonden om RSA te kraken, al zijn die manieren erg langzaam. Wel kan het zo zijn dat het versleutelen met RSA dusdanig lang en complex wordt bevonden, dat men kan overstappen naar een ander versleutelingsmethode.
De digitale beveiliging zal onder de huidige technologische snelheidsdruk steeds sneller ontwikkelen en wij verwachten dat deze steeds efficiënter zal worden, omdat RSA voor huidige standaarden nog redelijk zwaar is voor computers om informatie mee te beveiligen.

Inhoudsopgave

Inleiding

Waarom (RSA-)cryptografie?

Het schooljaar was pas begonnen en we moesten al aan de slag voor onze profielwerkstuk. Vorige schooljaar hadden wij, Jun Feng en Nahom Tsehaie, al besloten om samen het profielwerkstuk te maken. Toen we eenmaal achter de computer zaten, gingen we zoveel mogelijk onderwerpen zoeken, die onze wel leuk leken. Nadat wij een stuk of tien onderwerpen hadden, kozen we voor cryptografie, omdat dit een interessant onderwerp leek met veel wiskundige en ICT diepgang.
Cryptografie heeft betrekking op alles wat met beveiliging of versleuteling van informatie te maken heeft. Cryptografie kwam op onze top 10- lijst te staan door het recente nieuws omtrent de NSA en andere privacy schandalen . Er was sprake van beveiligingsproblemen en wellicht was er geheime informatie ontcijferd door de NSA. Wij vroegen ons af of dat echt zou kunnen en of je informatie wel beter kan beveiligen met de huidige technologische middelen. Daarom kozen we specifiek voor de RSA-cryptografie, die op dit moment veel in gebruik is bij het beveiligen van informatie en wilden we onerzoek doen naar de werking en toekomst van RSA. Tevens was het leuk om dit tegelijkertijd met het NSA-schandaal te doen, zodat men beter snapt wat er in de wereld gebeurt.
Daarnaast waren we erg blij met de vakken waarmee wij dit profielwerkstukonderwerp konden koppelen: informatica en wiskunde. In de lessen van bijvoorbeeld informatica hadden we al het een en ander wat gehad over beveiligingsprocessen en algoritmes die daarbij gebruikt worden. De wiskunde hierachter leek ons uitdagend, omdat de wiskundige kant van RSA-cryptografie een stapje hoger is dan het wiskunde wat je op school krijgt. Ook hadden wij bij het vak informatica in de 5e klas een programma gebouwd met behulp van *Visual Basic*. Dit vonden wij allebei zeer interessant. Ook hier wilden wij een stapje hogerop en wilden daarom een programma maken dat gebruikt maakt van het RSA-algoritme, om zo het praktische aspect van het profielwerkstuk in te vullen.
Tenslotte zal het onderzoek naar (RSA-)cryptografie goed aansluiten op onze toekomstige studies, die allebei aan de technische universiteit zal zijn. We zien dit dus als een soort *Minor* van de middelbare school, een verdiepende opdracht aan het eind van je derde studiejaar, waar je meerdere vakken combineert om een tijdje een onderzoek te doen en om jezelf te verrijken en beter te weten wat je in de toekomst staat te wachten. Zo heb je als het ware een kleine voorsprong op de rest en weet je ook wat je te wachten staat.

Hoofdvraag en deelvragen

Onze hoofdvraag met de daarbij behorende deelvragen luiden als volgt:

1. *Is RSA-cryptografie nu veilig genoeg en wat betekent het voor de toekomst van digitale beveiliging?*
    a. Hoe heeft cryptografie zich ontwikkeld?
    b. Welke soorten cryptografieën bestaan er?
    c. Wat is RSA en hoe werkt RSA?
    d. Op welke gebieden is RSA inzetbaar?
    e. Op welke manieren is het RSA te kraken?
    f. Wat is de toekomst van cryptografie?

Aanpak onderzoek

Allereerst hebben wij geprobeerd zoveel mogelijk nuttige informatie te verzamelen omtrent cryptografie, zodat we een goed beeld hadden van wat cryptografie was. Daarna hebben we specifiek gezocht naar RSA gerelateerde informatie. Op deze manier konden we een deel van de deelvragen beantwoorden.

Vervolgens hebben wij de wiskundige kant van de RSA-cryptografie onderzocht om daarna alle overige deelvragen te kunnen beantwoorden. Ook konden we met de opgedane kennis een programma maken in Visual Basic dat kon versleutelen en ontsleutelen middels het RSA-algoritme. Door het maken van het programma en het maken van het werkstuk zelf, kunnen wij een goede conclusie trekken.

Hoe heeft cryptografie zich ontwikkeld?

Cryptografie heeft een vroege oorsprong. Het onleesbaar maken van geschreven berichten is al eeuwen oud. Het doel van cryptografie was en is nog steeds het voorkomen dat derden kennis kunnen nemen van berichten die niet voor hen bedoeld zijn. Al duizenden jaren lang heeft cryptografie zich door ontwikkeld. Er zijn veel verschillende soorten methodes van cryptografie ontstaan. Wij behandelen de bekendste 7 methodes, namelijk: Methode van de Spartanen, De Caesarmethode; de techniek van Vigenère; de Vernam-codering; Enigma-code; DES; RSA. RSA-cryptografie zullen wij later ook nog uitgebreider behandelen. [3] [5]

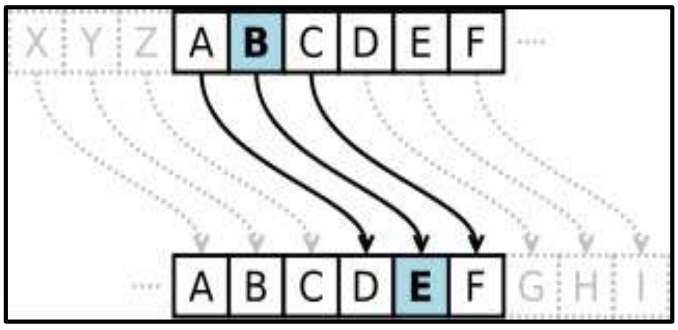
Methode van de Spartanen

Het woord cryptografie komt uit het Grieks. De Grieken zijn dan ook begonnen met het versleutelen van boodschappen. Rond 450 voor Christus gebruikten ze een stok (= scytale) waar ze stroken perkament of leer omwikkelden en schreven daar een boodschap op. De ontvanger moet ook een stok hebben met dezelfde diameter, zodat hij de boodschap kan lezen. Deze methode wordt ook wel de stafversleuteling genoemd, omdat de stok de sleutel is. [2] [5]

Figuur 1 Methode van de Spartanen

De Caesarmethode

De Romeinen waren actief op het gebied van cryptografie. De Caesarmethode heeft zijn naam te danken, aan het feit dat Julius Caesar dit cryptosysteem gebruikte. Hij vertrouwde zijn boodschappers niet en vercijferde eerst zijn bericht, voordat hij het verstuurde. De Caesarmethode houdt in dat elk letter in het alfabet wordt verschoven met een bepaald getal. Julius Caesar gebruikt de cijfer 3. Dit houdt dus in dat de letter 'A' de letter 'D' wordt. De oorspronkelijke tekst heet de ´klare tekst´ en de gecodeerde boodschap heet de ´cijfertekst´. De klare tekst 'VENI VIDI VICI' wordt in cijfertekst ´YHQL YLGL YLFL´. Zodra de ontvanger dus de cijfer weet, waarmee verschoven wordt, is de boodschap te ontcijferen. Julius Caesar gebruikte meestal het cijfer 3 voor de verschuiving, zijn opvolger Keizer Augustus gebruikte de cijfer 4. Desondanks wordt deze algemene theorie de Caesarmethode genoemd. Deze methode is echter niet betrouwbaar. Met een brute aanval (= het proberen van alle mogelijke oplossingen) is het mogelijk om de klare tekst terug te vinden. Er zijn immers maar 25 mogelijkheden. Deze Caesar methode is dus niet veilig. [1] [2] [5] [6]



Figuur 2: De Caesarmethode met verschuiving 3. 1

De techniek van Vigenère

In 1585 publiceerde de Fransman Blaise de Vigenère in zijn boek "*Traicté des chiffres ou secrètes manières d'escrire"* een techniek, waarbij meerdere keren alfabetisch versleuteld kan worden. Dit is een methode van *poly-alfabetische versleuteling.* In dit systeem vinden er meerdere verschuivingen

Figuur 3: Het Vigenère-tableau 1

plaats. Elk letter wordt met een andere verschuiving gecodeerd. Zo wordt de eerste letter met cijfer 10 verschoven en de tweede letter met cijfer 4. Op deze manier ontstaat er naast de klare tekst en de sleutel, ook cijfertekst. De sleutel kan dus zelf verzonnen worden. Hij kan van letters zijn, maar ook van cijfers die aangeven hoeveel er verschoven moet worde t.o.v. de klare tekst. De cijfertekst wordt gevonden door het snijpunt te vinden tussen de klare tekst en de sleutel in het Vigenère-tableau (zie figuur 2). Bij elk woord dat gecodeerd moet worden, begint de sleutel opnieuw. Wij geven een voorbeeld. De klare tekst is ´VENI VIDI VICI´. De sleutel is CRYPTO. De Cijfertekst is dan ´XVLX OWFZ TXVW´.

Dus:

VENI	VIDI	VICI	Klare tekst
CRYP	TOCR	YPTO	Sleutel
XVLX	OWFZ	TXVW	Cijfer tekst

Doordat de sleutel en de cijfer tekst niet tegelijk worden verstuurd, is dit veel veiliger dan de Caesarmethode. Zonder een van deze twee is het niet mogelijk om de klare tekst te achterhalen. Het kraken van de Vigenère-code (met evt. een brute aanval) is stuk een moeilijker omdat er voor de sleutel legio aan mogelijkheden zijn. Als de Sleutel én de cijfertekst door derden worden onderschept, is het mogelijk om de klare tekst te achterhalen en zal deze methode onveilig worden. Deze techniek is dus beter dan de Caesarmethode maar nog niet helemaal waterdicht. [9] [6] [5] [2]

De Vernam-codering

De Vernam-codering stond bekend als onbreekbare versleuteling in de cryptografie. In de loop van de eerste wereldoorlog werd deze bedacht. Het te versturen bericht mocht niet door de Duitsers achterhaald worden. Zo bedacht de Amerikaan Gilbert Vernam een coderingsmethode, genaamd *de Vernam- codering*. Dit was in die tijd niet te kraken. Het vernam systeem was een zeldzaam voorbeeld van een onbreekbare code. De tekst die gecodeerd moet worden, moet eerst versleuteld worden met een sleutel. Vervolgens stuurt persoon A de versleutelde boodschap aan persoon B met de sleutel. Zowel de boodschap als de sleutel moet volgens de ASCII-tabel omgezet worden. Het principe is hetzelfde als *De Techniek van Vigenère*. Hierbij hoeft echter de sleutel niet even groot te zijn als de versleutelde boodschap; bij Vernam-codering is dit wel het geval. Een nadeel van deze coderingsmethode is dat zodra de sleutel onderschept wordt, dit helemaal niet meer veilig is.

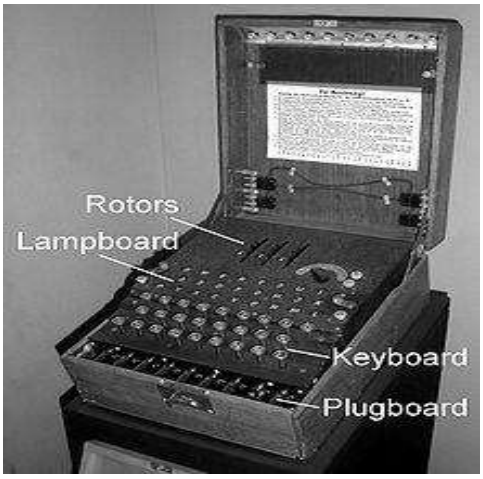
Dit probleem kan als volgt opgelost worden:

- Persoon A verstuurt de versleutelde boodschap met een willekeurige sleutel A.
- Persoon B versleutelt de versleutelde boodschap met sleutel A, nogmaals met een ander willekeurige sleutel B en stuurt deze weer naar persoon A. de boodschap is dus nu versleuteld met sleutel A en sleutel B.
- Persoon A haalt zijn eigen sleutel eraf, zodat er een versleutelde boodschap met sleutel B overblijft, en stuurt dit naar persoon B.
- Persoon B haalt zijn eigen sleutel eraf en kan de boodschap lezen.

Zo is het probleem opgelost. Er hoeft namelijk geen sleutel uitgewisseld te worden. Echter is deze methode niet waterdicht. Indien de drie berichten worden onderschept, is dit niet meer veilig, want dan kan de boodschap makkelijk vertaald worden. [7] [6] [1] [2]

Enigma-code

De Enigma-code was 'hip' tijdens de tweede wereldoorlog. Het werd gebruikt door de Duitsers en de techniek was al ontworpen door Hebern in 1920. Amerika gebruikte tijdens de tweede wereldoorlog de coderingsmethode *Sigaba.* Japan kon bovendien niet achterblijven en ontwikkelde hun eigen coderingsmethode *Red and Purple*. Allen zijn zogenaamde rotorsystemen en hebben ongeveer dezelfde techniek. De Duitsers typten een bericht op de Enigma en schreven op welke lampjes ging branden. Daarna werd de versleutelde boodschap via een telegraaf doorgeseind. De rotor is een schijf in isolerend materiaal, waar aan beide kanten elektronische contactpunten zitten, één voor elk symbool van het gebruikte materiaal. De machines hadden drie of vier van zulke rotoren, waarvan de posities bij elke gecodeerde letter wijzigden. Eerst werd via een geheim codeboek de begininstelling van de Enigma gekozen. Dat was elke dag een andere. De begininstelling zorgde ervoor dat een ingetypte C door de eerste rotor veranderde in een K. De tweede verandert de K in een R en derde verandert de R weer in een U. Uiteindelijk is dus de letter C verandert in een U. Na elke omwenteling draait dus de tweede en derde rotor een plaats door. De drie bewegende rotors kunnen dus 26^3 = 17576 mogelijke posities innemen. De Enigma-code kon in theorie dus niet gekraakt worden. Toch lukte England en Polen dit. De Engelse wiskundige Alan Turing vervulde hierbij een belangrijke rol. Hierbij gebruikte hij voorlopers van de computer. Het breken van de code leidde dus tot de uitvinding van de computer. Ook het ontcijferen van de Enigma-code heeft de nederlaag van de Duitsers versneld. [8] [1] [2] [5]

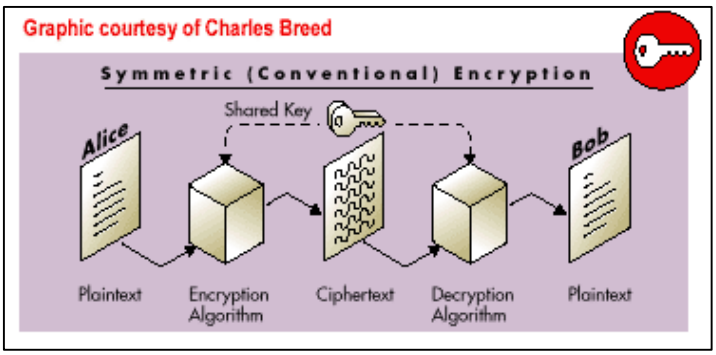


Figuur 4 Enigma machine in het Imperial War Museum, London.


Figuur 5 Rotorbladen van de Enigma

Data Encryption Standard (DES)

De Data Encryption Standard wordt ook wel DES genoemd. DES is in de jaren '70 ontwikkeld door *IBM* voor gebruik door de overheid en privésector. Het systeem werkt volgens een block cipher (blok codering). Er worden namelijk 'blokken' van 64 bits tegelijk gecodeerd door een computerchip. In plaats van letters worden er bits verwisseld en vervangen. Een blok bevat een sleutel van 56 bits en een controle van 8 controle bits. DES wordt sinds 1977 gebruikt. Bij de 56-bit sleutel die bij DES wordt gebruikt zijn er 2^{56} = 7, 2 x 10^{16} mogelijkheden. Hierdoor was het toen erg betrouwbaar, maar door de opkomst van snellere computers werd DES minder betrouwbaar. In 1997 de 56-bit sleutel gekraakt. Dit had men in 96 dagen gedaan door alle mogelijke oplossingen met tienduizenden computers langs te gaan, allen aangesloten via het internet. In 2001 is AES (Advanced Encryption Standard) de nieuwe standaard voor de Amerikaanse overheid geworden. Dit werkt met langere sleutels (128 of 256 bits) en is dus veel moeilijker te kraken. Ook zijn er andere verbeteringen van DES ontstaan zoals IDEA en 3DES. Triple DES of 3DES is een verbetering van DES die het algoritme van DES drie keer



Figuur 6 werking DES.

uitvoert en daarbij twee verschillende sleutels gebruikt. Ook is DES een vorm van een symmetrische cryptografie. Hierbij moet zowel de ontvanger als zender de sleutel bezitten. Derden kunnen de sleutel onderscheppen, waardoor deze methode minder betrouwbaar is. [5] [2] [6] [4]

RSA-Systeem

Doordat de hierboven besproken systemen symmetrische cryptografieën zijn, is het mogelijk voor derden om de boodschap op een of andere manier te achterhalen. Hierdoor was er een nieuw systeem ontwikkelt: het RSA-systeem. Dit is in 1978 ontworpen en wordt in het heden nog steeds gebruikt. Naast het dienen als beveiligingssysteem , biedt RSA ook mogelijkheden om de authenticiteit van berichten te waarborgen.

Wij zullen op dit moment niet verder gaan met het RSA-systeem. Later zullen wij de werking en toepassingen van het RSA-systeem veel uitgebreider behandelden.

[5]

Welke soorten cryptografieën bestaan er?

Er bestaan vele soorten cryptografiesystemen. Alle cryptografie-methodes worden onderverdeeld in twee grote hoofdgroepen: moderne cryptografie en klassieke cryptografie. De moderne cryptografie kan vervolgens onderverdeeld worden in asymmetrische en symmetrische cryptografie. De klassieke cryptografie kan weer onderverdeeld worden in substitutieversleuteling en transpositieversleuteling. Wat dit allemaal precies inhoudt, wordt in dit hoofdstuk verteld. [2]

Substitutieversleuteling

Bij veel klassieke versleutelingsmethoden wordt gebruik gemaakt van substitutieversleuteling. Zoals de naam al verraadt, wordt bij substitutieversleuteling de oorspronkelijke tekst vervangen door andere symbolen. Dit hoeft niet per se het alfabet te zijn, vandaar noemen we ook symbolen. Deze methode van versleuteling is echter niet goed omdat door middel van frequentieanalyse de boodschappen makkelijk te kraken zijn. Frequentieanalyse houdt in dat men onderzoek doet naar de frequentie van symbolen of groepen symbolen van een versleutelde tekst. De letter *e* komt bijvoorbeeld het meest voor in een normale tekst, zodat men meestal de meest voorkomende versleutelde symbool kan koppelen aan de letter *e*. Er bestaan twee typen substitutieversleuteling: mono-alfabetische versleuteling en poly-alfabetische versleuteling. [5] [9] [2]

Mono-alfabetische versleuteling

Wanneer elk letter vervangen is door een ander letter of symbool spreken we van mono-alfabetische versleuteling. De letter wordt voor de hele boodschap met hetzelfde symbool gecodeerd. Voorbeeld van een mono-alfabetische versleuteling is de *Caesarmethode*, waarbij elk letter steeds een *x* aantal plaatsen geschoven wordt. Deze versleuteling werd tot in de 9^{e} eeuw als onbreekbaar beschouwd. [16] [9] [1]

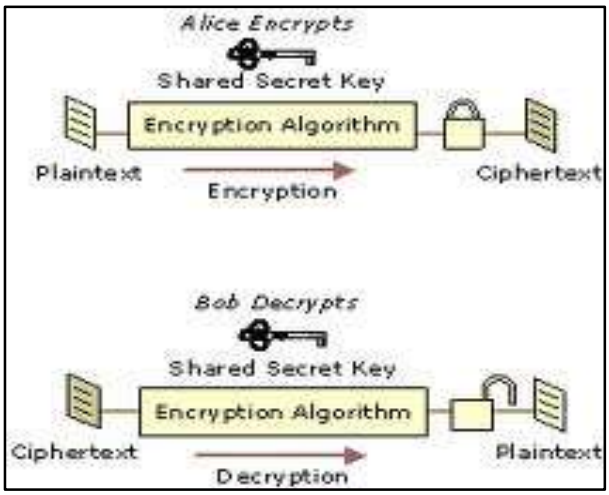
Poly-alfabetische versleuteling

Terwijl bij mono-alfabetische versleuteling elke letter steeds met dezelfde symbool wordt gecodeerd, is dit juist niet het geval bij poly-alfabetische versleuteling. Hierbij wordt gebruikt gemaakt van versleutelingsalfabetten, vaak in de vorm van tabellen. Zo wordt een boodschap meerder keren versleuteld. Elk letter van een boodschap is dus niet steeds met dezelfde symbool gecodeerd. Hierdoor is het dus ook veel moeilijker om frequentieanalyse methodes toe te passen, maar is dit nog wel mogelijk. Een voorbeeld van een poly-alfabetische versleuteling is het Vigenère-systeem. Bij deze versleutelingsmethode wordt elke letter met een andere rij uit het Vigenère tabel versleuteld. [16] [9] [14]

Transpositieversleuteling

Transpositieversleuteling is weer net iets anders dan substitutieversleuteling. Bij substitutieversleuteling draait het om substitueren van letters. Bij transpositieversleuteling draait het om veranderen van de positie van de letters. De letters kunnen dus op verschillende manieren verwisseld worden. In de klassieke cryptografie gebruikt men vooral *kolomtranspositie.* Enkele voorbeelden van kolomtranspositie zijn *transpositie cijfer* en de *dubbele transpositie. [21] [14]*

Symmetrische cryptografie

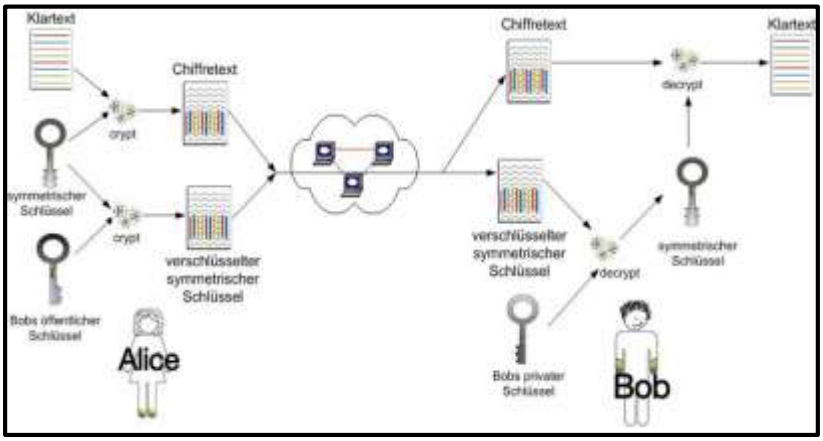


Symmetrische cryptografie wordt niet alleen in de moderne cryptografie gebruikt, maar ook al in de klassieke cryptografie. Bij symmetrische cryptografie wordt dezelfde sleutel gebruikt voor zowel het coderen door de zender als het decoderen door de ontvanger. Zulke sleutels wordt *symmetrische* sleutels genoemd. Hiernaast wordt de werking van symmetrische cryptografie afgebeeld. Het voordeel hierbij is dat het rekenwerk sneller is dan bij asymmetrische cryptografie. Het nadeel is natuurlijk dat zowel de ontvanger als de zender dezelfde sleutel nodig hebben. Dit moet dus ook van tevoren overlegd worden. De kans is groot dat derden deze sleutel onderscheppen. Daarnaast kan symmetrische cryptografie niet plaatsvinden, zodra berichten over minimaal drie zenders/ontvangers gaat. Het is namelijk niet duidelijk van wie het oorspronkelijke bericht is. Voor dergelijke gevallen gebruikt men asymmetrische cryptografie. Symmetrische cryptografie kan dus alleen gebruikt worden tussen één persoon/groep ontvanger en één persoon/groep zender. Voorbeeld van symmetrische cryptografie is Caesar-methode. Hierbij hebben zowel de ontvanger als zender het getal nodig, waarmee ze weten hoeveel plaatsen ze moeten verschuiven. DES (Data Encryption Standard) is een voorbeeld van symmetrische cryptografie in de moderne tijd.

[3] [18] [19] [1] [6] [17]

Asymmetrische cryptografie

Asymmetrische cryptografie is de tegenhanger van symmetrische cryptografie. Bij Asymmetrische cryptografie verschilt de sleutel waarmee gecodeerd wordt en de sleutel waarmee gedecodeerd wordt. De werking is als volgt (zie ook het plaatje hieronder):



1. De ontvanger *O* genereert 2 sleutels, een public en een private key. De private key weet alleen de ontvanger.
2. De public key verstuurt *O* naar de zender *Z*.
3. De zender *Z* codeert zijn boodschap met de public key en stuurt de boodschap naar de ontvanger *O*.
4. De ontvanger *O* decodeert de boodschap met de private key., waarnaar hij alleen het bericht overhoudt.

Hoe dit precies werkt volgt later in het volgende hoofdstuk. Een voorbeeld van asymmetrische cryptografie is RSA. Het voordeel van RSA is dat dit tot nu toe nauwelijks breekbaar is. Nadeel is dat het wel meer rekenwerk vereist en tijdrovender is ten opzichte van symmetrische cryptografie. [20] [2] [6] [18] [10]

Wat is RSA-cryptografie en hoe werkt het? [2][11][18][22]

Door het gevaar van onderschepping van de communicatie begon het idee te broeien van asymmetrische cryptografie, ook wel bekend als public-key cryptografie. Dit probleem is besproken aan het eind van het vorige hoofdstuk. Dit probleem is opgelost door het in het heden gebruikte RSA-cryptografie. Bij deze vorm van cryptografie is de sleutel bij het decoderen van informatie niet gelijk aan de sleutel die gebruikt wordt bij het coderen. Hierdoor is de gecodeerde informatie in principe niet te kraken.

In dit hoofdstuk zal uitgelegd worden hoe RSA-cryptografie uiteindelijk in zijn huidige staat is gekomen en hoe dit in de praktijk werkt, zowel het coderen als het decoderen van informatie.

Voorloper van RSA-cryptografie: Diffie-Hellman

In de jaren '60 begon de geschiedenis van de public-key cryptografie. In dit decennia werken de twee heren Whitfield Diffie en Martin Hellman aan een versleuteling die bedoeld was voor die iedereen, dus niet alleen overheidsinstanties. Ook wilden zij dat er niet meer moeilijk gedaan hoefde te worden om de decodeersleutel op een veilige manier bij iemand anders te brengen.
Zij bedachten de Diffie-Hellman cryptografie, die berust op een eenrichtingsfunctie. Zo'n functie is makkelijk toe te passen, maar uitgaand dat alleen de uitkomst en gebruikte functie bekend is, is het haast onmogelijk om de gebruikte beginwaardes te achterhalen. Zo kunnen twee personen veilig informatie versturen door gebruik te maken van een eenrichtingsfunctie, terwijl een afluisteraar nooit kan achterhalen wat de oorspronkelijke informatie was, omdat de eenrichtingsfunctie niet om te keren is.
Diffie-Hellman was baanbrekend, maar nog niet perfect: de twee personen die informatie uit wilden wisselen moesten in direct contact zijn op hetzelfde moment. Soms is dit niet praktisch. Daarom begonnen andere cryptografen met het perfectioneren van de Diffie-Hellman procedure. Dit leidde tot de ontwikkeling van RSA-cryptografie.

Het ontwikkelen van RSA-cryptografie

Uiteindelijk waren het de drie mannen Ron Rivest, Adi Shamir en Len Adleman die in 1977 kwamen met een public-key cryptografie waarmee direct mee kon worden versleuteld en ontsleuteld. Deze vorm van cryptografie kreeg de naam aan de hand van de achternamen van de bedenkers: RSA. Het RSA encryptie- en decryptiealgoritme berustte op eenrichtingsfuncties en de moeilijkheid van priemfactoriseren van grote getallen.
Het grote verschil met de Diffie-Hellman cryptografie is dat de twee informatie-uitwisselaars niet tegelijkertijd in contact hoeven te zijn. Persoon A maakt de encryptiesleutel openbaar, zodat persoon B zijn informatie kan versleutelen middels een eenrichtingsfunctie. Daarna kan persoon A met zijn eigen, geheime decryptiesleutel , die gekoppeld is aan de openbare encryptiesleutel, de informatie van B ontsleutelen. Omdat de encryptiesleutel openbaar is, hoeft A niet in contact te zijn met B als die wil versleutelen.
In de volgende paragrafen zal het RSA algoritme duidelijk worden uitgelegd.

Wiskundige theorie achter RSA-cryptografie

Achter RSA-cryptografie schuilt veel wiskundige theorie, wat eerst begrepen moet worden voordat uiteindelijk gecodeerd en gedecodeerd kan worden met het RSA-algoritme. Vandaar ook dat we deze theorie in deze paragraaf zullen uiteenzetten.

Gehele getallen en deelbaarheid

Bij RSA- cryptografie wordt alleen gebruikt gemaakt van gehele getallen. Dit houdt in dat er geen getallen zoals breuken, kommagetallen , wortels worden gebruikt bij het coderen of decoderen met
de RSA-methode. Deze verzameling van gehele getallen wordt genoteerd als $\mathbb{Z}$. De verzameling
gehele getallen $\mathbb{Z}$ is dus:

…, -100, -99, …, -3, -2, -1, 0, 1, 2, 3, …, 99, 100, …

Deelbaarheid is een eigenschap die gehele getallen kunnen hebben. Als getallen a en b twee gehele getallen zijn, waarbij a níet nul is en b = a*c, met c als een geheel getal, dan is b deelbaar door a. Dit kun je noteren als a|b. Een regel die hieruit voortvloeit is:

als a|b en a|c, dan geldt dat a|(bx+cy), waarbij alle getallen gehele getallen zijn

Een andere toepassing van deelbaarheid die nodig is bij RSA-cryptografie is de grootste gemeenschappelijke deler van twee getallen. Bij getallen a en b korten we dit af als ggd(a,b)=d.
Een voorbeeld hierbij is de ggd(12,30). De delers van 12 zijn 1, 2 ,3, 4, 6, 12. De delers van 30 zijn 1, 2, 3, 5, 6, 10, 15 en 30.
De gemeenschappelijke delers van 12 en 30 zijn dus 1, 2, 3 en 6. De grootste gemeenschappelijke deler van 12 en 30 is dus ggd(12,30)=6.

Priemgetallen

Een priemgetal is een geheel getal a dat als enige positieve delers de getallen 1 en a zelf heeft, waarbij a nooit 1 is. Een geheel getal dat groter dan 1 is en geen priemgetal is, wordt een samengesteld getal genoemd.
De reden dat deze getallen samengestelde getallen worden genoemd is de volgende regel:

Elk positief getal n groter dan 1 kan geschreven worden als het product van unieke priemgetallen.

Het opbreken van getallen in priemgetallen heet priemfactorisatie, wat verderop wordt besproken.

Het aantal priemgetallen is erg groot, oneindig zelfs. Dit is bewezen door Euclides. Daarnaast is door hem bewezen dat het aantal priemgetallen afneemt naarmate de grootte van de getallen toeneemt. Priemgetallen kunnen een rol spelen bij het berekenen van de grootste gemeenschappelijk deler van twee gehele getallen. Bij de berekening van ggd(a,b) kan namelijk gebruikt gemaakt worden van de priemfactoren van beide getallen, omdat priemgetallen zelf niet meer deelbaar zijn door een ander getal behalve 1.

Soms is het echter moeilijk om achter alle priemdelers van een getal te komen. Ook hiervoor heeft Euclides een oplossing bedacht, welke is vernoemd naar hemzelf: het algoritme van Euclides. Dit algoritme maakt gebruik van delen van twee getallen (a,b) en daarna het kleinste getal b van (a,b) te delen door de kleinst mogelijke gehele restgetal. Dit wordt herhaald totdat de rest 0 is. a en b zijn dan geheel deelbaar door elkaar. Het kleinste getal in de laatste deling is dan de ggd(a,b)

Het algoritme van Euclides was ook uitgebreid om tot de volgende stelling te komen:

"ax +by =ggd(a,b)"

Met x en y die deel uitmaken van de verzameling gehele getallen

Priemfactorisatie

Zoals in de vorige paragraaf is besproken, is het erg moeilijk om van grote getallen 'op te breken' in priemfactoren, ook wel bekend als priemfactorisatie. Dit is dan ook één van de dingen waar RSA-encryptie op berust.
De Fransman Pierre de Fermat had in de 17^e eeuw echt een methode gevonden om ook grote getallen te factoriseren in zijn priemgetallen.

Je kunt er dus ook achter komen of een getal een priemgetal is of niet! Als een getal namelijk niet te factoriseren is, dan is het getal zelf een priemgetal.
De methode van Fermat is echter erg langdradig, omdat het aantal priemgetallen steeds schaarser worden als de getallen ook groter worden. Het duurt daarom ook erg lang bij de grote getallen.
Er ook nog een andere manier om er achter te komen of een getal een priemgetal is. Deze manier is óók bedacht door Fermat en wordt 'de kleine stelling van Fermat' genoemd. Deze methode kun je alleen begrijpen als je modulo rekenen onder de knie hebt, wat behandeld zal worden in de volgende paragraaf.

Modulo rekenen

Modulo rekenen is een deel van de wiskunde die eigenlijk dagelijks wordt gebruikt door alle mensen, zonder dat zij dat weten. Een voorbeeld hiervan is het klokkijken: als een digitale klok aangeeft dat het 23.00 uur is, dan weten we gelijk dat het 11.00 uur is. Dit weten we omdat we de som 23 -12 =11 uitvoeren. In de wiskunde zeggen we dan dat drieëntwintig en elf modulo twaalf congruent zijn. Of nog korter: $23 \equiv 11 \pmod{12}$. Er gaat namelijk altijd $a = b + kn$, waar nu a =23, b =12 en k is een geheel getal.
De regels bij het modulo rekenen die we nodig hebben bij de RSA-encryptie zijn:

"Als $a \equiv b \pmod{n}$ en $b \equiv c \pmod{n}$, dan geldt $a \equiv c \pmod{n}$."

"Als $a \equiv b \pmod{n}$ en $b \equiv c \pmod{n}$, dan geldt $ac \equiv bd \pmod{n}$."

"Als $a \equiv b \pmod{n}$, dan geldt $a^m \equiv b^m \pmod{n}$."

"a mod b = a – (b* Int(a / b))"

Int(x) betekent dat van het decimale getal x alle cijfers achter de komma's worden weggehaald (omlaag afronden)"

Nu modulo rekenen is uitgelegd kunnen de kleine stelling van Fermat en de functie die de Zwitser Leonhard Euler daarbij bedacht heeft begrepen worden.

De kleine stelling van Fermat en de Euler φ-functie

De kleine stelling van Fermat is erg belangrijk in de RSA-cryptografie. Deze stelling is door Fermat bedacht, maar het bewijs ervoor had hij niet vrijgegeven. Deze werd uiteindelijk door Euler gevonden in 1736 en Euler heeft bij de kleine stelling van Fermat een nieuw begrip bedacht, dat óók bijdrage levert aan de RSA-cryptografie.
De kleine stelling van Fermat luidt als volgt:

"Als p een priemgetal is en a een positief geheel getal, zó dat a niet deelbaar is door p, dan geldt dat $a^{p-1} \equiv 1 \pmod{p}$"

Met deze stelling kun je er dus achter komen of een bepaald getal een priemgetal is, door deze in te vullen voor p.

Ook Euler heeft een toevoeging aan de wiskunde gedaan die leidde tot het kunnen maken van het RSA-algoritme. Allereerst moet de φ-functie worden gedefinieerd:
$\varphi(n)$ is het aantal getallen kleiner dan n, groter dan 0, die relatief priem zijn met n. Als getallen a en n relatief priem zijn, dan geldt dus dat ggd(a,n) = 1.

Euler breidde zijn eigen φ-functie (ook wel Euler totiënt-functie) uit en bewees het volgende:

$\varphi(n*m) = \varphi(n) * \varphi(m)$

Aangezien een priemgetal niet deelbaar is door andere getallen dan 1 of zichzelf, geldt voor priemgetal p: $\varphi(p) = (p -1)$

De kleine stelling van Fermat en de Euler totiënt-functie zijn van groot belang in het RSA-algoritme.

Nu is alle wiskunde achter RSA-cryptologie behandeld en kan worden uitgelegd hoe RSA-encryptie precies werkt en wat het RSA-algoritme is.

De RSA-encryptiemethode

Als je een bericht wilt sturen middels de RSA-encryptiemethode, dan moeten we eerst het bericht omzetten in getallen. Voor de eenvoud nemen we nu als voorbeeld het getal M. In de realiteit heeft elk teken een eigen code, de ASCII-code. Dan wordt elk teken door het RSA-algoritme versleuteld.
Stel dat persoon A getal M wil sturen naar persoon B, zónder dat een buitenstaander, persoon C, het bericht kan ontrafelen en zonder dat persoon A en B contact met elkaar moeten hebben over de codesleutel:

1. A neemt priemgetallen P en Q en berekent N = P*Q.

2. Daarna wordt $\varphi(N)$ berekent. $\varphi(N) = (P-1)*(Q-1)$ *(Euler totiënt-functie)*

3. A berekent een getal E dat kleiner is dan N en relatief priem is aan $\varphi(N)$. De ggd(E, $\varphi(N)$) = 1. *(algoritme van Euclides).* De openbare encryptiesleutel is (N,E).

4. A berekent zijn decryptiesleutel (N,D). Voor D geldt dat (E*D -1) deelbaar is door $\varphi(N)$. D is een geheel getal.*(kleine stelling van Fermat en ax +by =ggd(a,b))*

5. P, Q en $\varphi(N)$ worden weggegooid, zodat het algoritme nooit nagebootst kan worden als informatie wordt onderschept. Het getal D moet veilig bewaard worden door de ontvanger.

6. Informatie M wordt versleuteld door B met de openbare encryptiesleutel (N,E) tot versleutelde informatie C: $C= (M^E) \text{ Mod } N$

7. A ontvangt C en ontsleutelt C tot M met decryptiesleutel (N,D): $M = (C^D) \text{ Mod } N$ *(Gevolg klein stelling Fermat)*

Als C wordt onderschept kan M niet worden berekend, omdat de onderschepper niet beschikt over D of P en Q. P en Q zijn namelijk noodzakelijk als $\varphi(N)$ berekend moet worden.
En omdat grote getallen moeilijk zijn te factoriseren is het niet te doen om de juiste P en Q te berekenen aan de hand van N.
Ook geldt nog eens dat de versleuteling met de openbare encryptiesleutel (N,E) een eenrichtingsfunctie is.

Met het zelfgemaakte RSA programma is na te bootsen hoe het RSA-algoritme werkt.

Op welke gebieden is RSA inzetbaar?

In het vorige hoofdstuk is duidelijk gemaakt wat voor een soort cryptografie RSA eigenlijk is en welke stappen er moeten gezet worden om ermee te kunnen versleutelen en ontcijferen. Daarnaast is de wiskundige theorie uitgelegd en is bewezen dat het erg lastig is om RSA te kraken, als je een versleuteld bericht weet te onderscheppen.

De efficiëntie van RSA [23]

RSA is dus een versleutellogaritme dat ideaal is voor beveiligde communicatie. RSA zouden we dus in principe kunnen gebruiken om berichten veilig te versturen naar specifieke personen, al is het wel een voorwaarde dat diegene wel zijn publieke sleutels openbaar maakt, aangezien RSA berust op public-key cryptografie.

Het is echter wel zo, dat het versleutelen en ontcijferen van (lange) boodschappen erg veel rekenwerk vereist en dat het RSA-versleutelproces met de huidige technologie nog niet snel genoeg uitgevoerd kan worden om er constante, directe en tegelijkertijd beveiligde communicatie mee uit te voeren. Chatten met RSA-beveiligde berichten zonder vertraging is op dit moment bijvoorbeeld niet mogelijk, omdat de rekenkracht van de huidige standaard computers niet groot genoeg is om het RSA-algoritme snel uit te voeren met de grote getallen (bits) die nodig zijn om de RSA-versleuteling veilig te houden.

Combineren van RSA [23]

Een veel voorkomende soort versleuteling die door de langdradigheid van het RSA-algoritme wordt toegepast is een combinatie van RSA-versleuteling en symmetrische versleuteling. Symmetrische cryptografie maakt meestal gebruik van een simpeler algoritme dan de algoritmes van asymmetrische cryptografie, waardoor de huidige standaard computers deze makkelijker kunnen toepassen is combinatie met RSA-cryptografie, wat valt onder de asymmetrische cryptografieën. Berichten kunnen versleuteld worden middels een symmetrische versleuteling, maar om de veiligheid te waarborgen wordt de sleutel van de gebruikte symmetrische versleuteling versleuteld met het RSA-algoritme. Op deze manier hoeft een computer, terwijl de berichten veilig blijven, minder vaak het RSA-algoritme uit te voeren en juist vaker een simpeler algoritme dat bij een symmetrische versleuteling hoort. Zo kan RSA bijvoorbeeld efficiënt worden toegepast op het gebied van communicatie.

Cryptosystemen die gebruikt worden en berusten op het combineren van RSA met een symmetrisch cryptosysteem zijn onder andere S/MIME en PGP (Pretty Good Privacy). Het verschil tussen de twee cryptosystemen is dat S/MIME gebruikt maakt van (digitaal) gecertificeerde publieke sleutels, waar we in de vorige paragraaf op komen. De twee genoemde cryptosystemen worden onder andere toegepast bij mailservices van webbrowsers.

Digitale certificaten[24][25]

Hoewel cryptografie in het verleden vooral werd gebruikt bij communicatie is door de komst van het internet en RSA het aantal gebieden waarin in cryptografie zoals RSA kan worden ingezet, toegenomen. Met de komst van het internet is het erg belangrijk geworden om te weten of het veilig is informatie uit te wisselen met een site én of je daadwerkelijk op de site zit waar je denkt dat je op zit. Een goed voorbeeld hierbij is internetbankieren. Het zou erg gevaarlijk zijn als de informatie die je nodig hebt om te internetbankieren in andermans handen vallen.

Om die redenen zijn digitale certificaten en digitale handtekeningen in het leven geroepen, waarvan de werking berust op RSA-cryptografie. Deze digitale certificaten worden alleen

verstrekt door instanties die bepalen of een site geschikt is om hun certificaat te gebruiken. Deze sites worden dus geverifieerd door een objectieve instantie die daartoe is aangewezen. Er moet duidelijk worden gemaakt dat niet élke site een certificaat heeft. Vooral sites waarbij het belangrijk is dat informatie veilig blijft hebben een certificaat aangevraagd. In het digitale certificaat staat de naam van de site, de sitebeheerder, de soort beveiliging die wordt gebruikt op de site en de publieke sleutels die gebruikt worden bij de beveiliging. Op deze manier kan een computer bevestiging krijgen dat de site geverifieerd is en weet hij op welke manier de te versturen informatie moet worden versleuteld, zodat de informatie veilig blijft. Als de URL van een website met "*https://*" , dan weet je dat al je verstuurde informatie wordt versleuteld met de soort cryptografie die in het certificaat van de website staat. [26]

Figuur 2 Digitaal Certificaat

Digitale handtekeningen[25]

Met een digitaal certificaat kan je computer dus veilig informatie uitwisselen met een site door middel van de publieke sleutel van de website die in zijn certificaat staat. Er is echter nog steeds een kans op een lek in het certificaat. Het certificaat kan aangepast zijn door criminelen, zodat buitenstaanders de uitgewisselde informatie kunnen onderscheppen en ontcijferen. We moeten dus weten of het digitale certificaat echt door de vertrouwde instantie zelf is gemaakt en niet is aangepast. Dit is waar de digitale handtekening komt kijken, die ook op RSA- cryptografie berust. Aan de digitale handtekening is te zien of bepaalde informatie, in dit geval een digitaal certificaat, echt door diegene is gemaakt van wie je de informatie ontvangt. De digitale handtekening wordt namelijk als volgt gemaakt en gebruikt:

1. Bereken de hashwaarde van je te versturen informatie. De hashwaarde wordt berekend met het hashalgoritme.
2. Versleutel de hashwaarde met je persoonlijke RSA-sleutels.
3. Voeg je versleutelde hashwaarde bij je te versturen informatie.
4. De ontvanger berekent de hashwaarde van de ontvangen informatie (exclusief de versleutelde hashwaarde).
5. De ontvanger ontcijfert de bijgevoegde hashwaarde met de publieke sleutels van de verstuurder.
6. De ontvanger vergelijkt de twee hashwaardes.
7. Als de hashwaardes gelijk zijn weet de ontvanger dat het bericht niet is aangepast en dat de publieke sleutels van de verstuurder te vertrouwen zijn en de informatie authentiek is; als het bericht zou zijn aangepast, dan zou de hashwaarde van de informatie zijn aangepast, zodat bij vergelijking van de waardes er geen gelijkenis zou zijn.

Voordat de publieke sleutels in het certificaat van een website worden toegepast, wordt eerst gecheckt door je computer of het certificaat is aangepast middels de digitale handtekening die bij het certificaat hoort.

Digitale handtekeningen worden ook toegepast door uitgevers van software. Als je een programma downloadt van het internet is het belangrijk dat het programma veilig is en niet is aangepast tijdens de digitale 'transport' . Het zou zomaar kunnen dat een internetcrimineel een programma zo heeft aangepast dat er virus in zit. Met digitale

handtekeningen weet de computer bijvoorbeeld dat als je Microsoft Office downloadt, dat het daadwerkelijk van Microsoft komt en niet is aangepast.

De enige zwakte van de digitale handtekening is dat er maar een beperkte hoeveelheid hashwaardes zijn. Het gevaar is dan dat na het aanpassen van informatie de hashwaarde nog steeds hetzelfde is. Toch worden hashwaardes gebruikt bij het maken van digitale handtekeningen, omdat de hoeveelheid hashwaardes zo groot is dat de kans op dezelfde hashwaarde na aanpassing verwaarloosbaar klein is. Bij het algoritme van de soort MD5-hash zijn er bijvoorbeeld 2^{128} verschillende hashwaardes, wat ongeveer overeenkomt met $3{,}4 \times 10^{38}$.

Verscheidene mogelijkheden van RSA

In tegenstelling tot het verleden, waar cryptografie slechts werd gebruikt om veilig te communiceren in tijden van oorlog, wordt RSA-cryptografie in het heden toegepast in vele gebieden, waarvan de oorzaak gezocht kan worden in de komst van de computer. Op dit moment wordt RSA vooral toepast op het gebied van authenticatie en beveiliging van informatie. Soms wordt RSA zelfs gecombineerd met een andere soort cryptografie.

Aangezien computers steeds sneller worden, exponentieel snel zelfs, en cryptografiealgoritmes zoals RSA steeds ingewikkelder worden blijft het een verassing in welke nieuwe gebieden RSA kan worden in gezet en wat het toekomstbeeld voor cryptografie in het algemeen is.

Op welke manieren is het RSA-systeem te kraken?

Hoewel RSA erg moeilijk is te kraken door de grote getallen die gebruikt worden, zoals in het hoofdstuk over de werking van RSA te lezen was, zijn er in de loop der tijd toch cryptografen, hobbyisten of zelfs criminelen die de afgelopen jaren oplossingen hebben gezocht om het 'onverwoestbare' RSA toch te kunnen kraken. In dit hoofdstuk zullen we de gevonden 'oplossingen' uiteenzetten.

Trial division

Computers kunnen heel snel zeer grote getallen met elkaar vermenigvuldigen, maar een getal (in dit geval *n*) ontbinden gaat veel moeilijker. Het ontbinden van *n* wordt ook wel *trial division* genoemd. Met deze methode vindt men gegarandeerd een deler van *n*. Op supercomputers worden met grote processoren de sleutels geprobeerd te berekenen. Bovendien zijn er de afgelopen jaren nieuwe methodes toegevoegd aan de computers. Voor hele grote getallen, wat bij het RSA het geval is, is er echter zeer veel tijd voor nodig. Toch is het gelukt om de 512 bits RSA modulus te kraken. Vandaar dat men is overgestapt naar 1024 bits RSA modulus. Dit hebben zij gedaan, omdat het benodigde werk om zo *n* te ontbinden nu ook exponentieel toeneemt. Zo is het steeds mogelijk om de techniek van het ontbinden een stap voor te blijven. *Trial division* is dus praktisch onmogelijk en dan ook niet bruikbaar bij RSA. [15] [11] [10]

Brute force hacking

De naam zegt al genoeg. Bij *brute force hacking* worden geen lastige berekeningen uitgevoerd, maar probeert men simpel 'met brute kracht' alle mogelijkheden uit. Bijvoorbeeld: Er is een willekeurige klare tekst *m* . Men codeert *m* aan de hand van de publieke sleutels ($c = m^e \bmod n$). Vervolgens controleert men of de werkelijke tekst overeenkomt met het gevonden cijfer. Indien dit gelijk aan elkaar is, dan was het goed gegokt. Theoretische zal de code eens gevonden worden, maar net als *trial division* kan het zeer lang duren voordat dit gevonden zal worden. [11] [10] [14]

Number Field Sieve (NFS)

De *getallenlichamenzeeef* , in het Engels Number Field Sieve (NFS), is een manier om samengestelde getallen te ontbinden in priemfactoren. Dit is een andere techniek dan trial division en de techniek van NFS is gebaseerd op 'verschil van twee kwadraten'. Deze methode is rond 1988 ontwikkeld door John Pollard, die het zevende fermatgetal factoriseerde. Het zevende fermatgetal staat gelijk aan $2^{256} + 1 =$ $59649589127497217 \cdot 5704689200685129054721$.

Number Field Sieve werkt als volgt: Als men de uitdrukking $x^2 - y^2$ heeft, kan men die factoriseren in $(x - y)(x + y)$. Deze techniek past men ook toe bij het getal *n*. De factoren moeten dus voldoen aan de voorwaarde $x^2 \equiv y^2 \pmod n$, want er bestaat dan een getal *k* zo dat $x^2 - y^2 = kn$. Hieruit volgt dat $GGD(x - y, n)$ en $GGD(x + y, n)$ niet-triviale factoren van *n* zijn. En aangezien *n,* het product is van twee priemgetallen *p* en *q,* blijkt de kans 2/3 dat men één van die factoren vindt op deze manier.

Natuurlijk geldt hier: Hoe sneller de computers, hoe sneller de bewerkingen uitgevoerd kunnen worden. Er zijn echter oneindig veel priemgetallen, dus is het altijd mogelijk om snelle computers moeilijk te maken. Hoe men precies de x's en y's vindt met de NFS is met een ingewikkeld en lang algoritme. [1] [13] [1]

Gemeenschappelijke priemfactor

Vaak wordt een van de twee priemfactoren herhaaldelijk gebruikt en verandert men alleen de tweede priemfactor. Iedereen kan zien dat dit fout zal gaan. Met de algoritme van Euclides is van het product *n* gemakkelijk de overige priemgetallen met p als gemeenschappelijke priemfactor te vinden, omdat die wordt gevonden door de grootste gemeenschappelijke deler te vinden. Het is dus belangrijk om steeds twee verschillende priemgetallen te nemen.

Echter zullen er altijd wel op de wereld toevallig twee dezelfde priemgetallen gebruikt worden. Er kan zelfs hetzelfde product van priemfactoren gemaakt worden. Op allerlei plaatsen in de wereld worden immers priemfactoren voor RSA gebruikt. Onlangs is er onderzoek gedaan naar ruim zes miljoen RSA sleutels. Daaruit blijkt dat 12.000 sleutels een dezelfde gemeenschappelijke factor hadden en meer dan 70.000 dezelfde product n. Deze sleutels en producten waren allen onafhankelijk van elkaar gemaakt. Toch is kan voor de gebruikers een groot risico zijn. [10]

Overige mogelijkheden

Men is jarenlang bezig om een andere, makkelijkere methode voor ontcijfering te vinden. Niks wijst erop dat dit makkelijker kan. Er is zelfs theoretisch werk verricht om aan te tonen dat het vinden van de priemgetallen niet makkelijker kan dan met factoriseren. Ook is er een bewijs dat het ontcijferen van één bit van een RSA-bericht net zo moeilijk is als het compleet ontcijferen. Dit laat dus zien dat bij RSA-ontcijfering het óf alles óf niets is en niet beetje bij beetje ontcijferd kan worden. Ook vraagt men zich af of *d* de enige sleutel is, waarmee ontcijferd kan worden. Er zijn wel méér sleutels, maar uit onderzoek is gebleken dat ze allemaal op hetzelfde neerkomen. Dit maakt het dus niet makkelijker op. We kunnen dus zeggen dat er geen achterdeur is bij RSA-ontcijfering. [10] [11]

Uitdagingen van RSA-security

De bedenkers van RSA hebben een commerciële site en een site voor het laboratorium binnen RSA-security. Respectievelijk heten ze www.rsasecurity.com en www.rsasecurity.com/rsalabs. De commerciële website biedt producten en diensten aan die gebruik maken van versleuteling. De site voor het laboratorium onderzoekt naar technieken, die toegepast kunnen worden in hun nieuwe producten. Een onderdeel van de website www.rsasecurity.com zijn de uitdagingen. Hier daagt RSA-security mensen uit om een bepaald getal te factoriseren, oftewel te kraken. Wie deze als eerst voltooid, ontvangt een geldbedrag. Dit geldbedrag wordt hoger naarmate je een groter getal ontbind. Door deze *uitdagingen* heeft RSA security een soort controle op welke grootte van *n* nog als veilig beschouwd kan worden. In 2007 is RSA security gestopt met deze challenges, omdat ze bang waren veel geld te verliezen door de komst van supercomputers en geavanceerde technologieën. Tegenwoordig krijgt men dus geen geldprijzen, maar de ontdekkers worden wel opgenomen in de RSA Challenge tabel (zie volgende blz.). [12] [1]

Tabel 1 RSA Challenge numbers

RSA Number	***Decimal digits***	***Binary digits***	***Cash prize offered***	***Factored on***	***Factored by***
RSA-100	100	330	$1,000 USD	April 1, 1991	Arjen K. Lenstra
RSA-576	174	576	$10,000 USD	December 3, 2003	Jens Franke, University of Bonn
RSA-640	193	640	$20,000 USD	November 2, 2005	Jens Franke, University of Bonn
RSA-704	212	704	$30,000 USD	July 2, 2012	Shi Bai, Emmanuel Thomé and Paul Zimmermann
RSA-768	232	768	$50,000 USD	December 12, 2009	Thorsten Kleinjung
RSA-896	270	896	$75,000 USD	Not Factored	/
RSA-1024	309	1024	$100,000 USD	Not Factored	/
RSA-1536	463	1536	$150,000 USD	Not Factored	/
RSA-2048	617	2048	$200,000 USD	Not Factored	/

Wat is de toekomst van cryptografie?

De huidige efficiëntie van RSA in het achterhoofd houdend kunnen we stellen dat het niveau van cryptografie op dit moment net niet hoog genoeg is om de veiligheid van informatie te waarborgen.

Dat probleem twee oorzaken die met elkaar in verband zijn. Aan de ene kant kan met de huidige technologie de beste cryptoalgoritmes niet snel genoeg worden uitgevoerd. Overdracht van informatie kan niet snel en veilig tegelijkertijd zijn. Aan de andere kant is het niveau van de huidige cryptosystemen niet hoog genoeg. Van veel cryptosystemen is de zwakke plek al bekend, maar worden toch gebruikt, juíst omdat de snelheid van deze 'zwakke' cryptosystemen wel snel uitvoerbaar zijn.

Technologische vooruitgang

De afgelopen decennia is het aantal onderdelen dat op een computerchip past sterk gestegen, zoals voorspeld door Gordon Moore, één van de oprichters van chipfabrikant Intel. Als je kijkt in tijdvakken van 2 jaar, dan zie je dat het aantal onderdelen steeds verdubbeld. Hoewel het aantal onderdelen niet gelijk staat de snelheid, is wel te zeggen dat de technologie zich steeds sneller ontwikkeld en zo ook de snelheid en rekenkracht van computers. In toekomst kunnen langere en intensievere cryptografiealgoritmes worden toegepast die nu nog niet in gebruik zijn door de beperkte rekenkracht van de huidige computers. Het is echter niet zeker of de voorspelling van Moore tot in de eeuwigheid zal voortduren en de vooruitgang zal stagneren. Het komende decennia zal de vooruitgang in technologie zich vrijwel zeker voortzetten zoals die nu is, wat af te leiden is aan de toename van integratie van technologie in ons dagelijks leven. [23]

Verbetering van efficiëntie

Naast het kunnen gebruiken van ingewikkeldere algoritmes zullen er ook nieuwe, efficiëntere algoritmes worden bedacht. Deze algoritmes vloeien voort uit de groeiende wiskunde. In de wiskunde zijn er nog steeds onopgeloste problemen welke mensen dagelijks problemen op te lossen. Elk opgelost probleem kan leiden tot een nieuw cryptosysteem of algoritme. In plaats van een ingewikkeld algoritme kunnen er dus ook simpele algoritmes worden bedacht, die behalve sneller ook veiliger zijn.

De vraag naar dit soort algoritmes is groot en er zijn dan ook veel bedrijven die grof geld zouden betalen voor een patent op zo'n algoritme. Zodoende zijn er nu, en ongetwijfeld ook in de toekomst altijd mensen die door middel van toepassen van wiskunde steeds efficiëntere cryptosystemen bedenken. [23]

Privacy en beveiliging

De druk op ontwikkeling van betere cryptosystemen is sinds het begin van de 21ste eeuw met grote hoeveelheid toegenomen. Door de toename van onder andere social media zijn steeds meer gegevens van burgers te vinden op het internet. Zo is het aantal incidenten waarbij de privacy van de burger werd geschonden de afgelopen jaren sterk toegenomen. Voorbeelden zijn het Diginotar-incident, waarbij het DigiD niet meer veilig was, en de NSA, die gegevens van burgers over de hele wereld kon achterhalen. Het is dan ook onvermijdelijk dat overheden en bedrijven in de toekomst beter zullen samenwerken om gegevens van burgers beter te beschermen en dat studies omtrent beveiliging van informatie zullen worden gestimuleerd. Op deze manier zal de cryptografie zich sneller ontwikkelen onder druk van burgers wereldwijd. [23]

Conclusie

Met dit profielwerkstuk hebben wij de onderstaande hoofdvraag met bijbehorende deelvragen beantwoord:

"Is RSA-cryptografie nu veilig genoeg en wat betekent het voor de toekomst van digitale beveiliging?"

g. Hoe heeft cryptografie zich ontwikkeld?
h. Welke soorten cryptografieën bestaan er?
i. Wat is RSA en hoe werkt RSA?
j. Op welke gebieden is RSA inzetbaar?
k. Op welke manieren is het RSA te kraken?
l. Wat is de toekomst van cryptografie?

Elk hoofdstuk van dit profielwerkstuk beantwoordt respectievelijk één van de deelvragen a t/m f. Hier volgt dan ook een samenvatting van elk hoofdstuk met, onderstreept, de kernzin van elk hoofdstuk.

Door het maken van dit profielwerkstuk hebben we bevonden dat in de loop van de tijd het versleutelen van berichten steeds geavanceerder geworden. Deze geschiedenis begint bij de Romeinen en hun Caesar methode. Dit was een klassieke vorm van cryptografie. Vervolgens ging men vele eeuwen later over naar de Vigenère techniek. Hierbij gebruikt met de Vigenère tabel om de klare tekst twee maal te versleutelen. Dit is dan ook een vorm van poly alfabetische versleuteling. De daarna volgende vernam-codering is soortgelijk aan de techniek van Vigenère. Echter hoefde bij vernam-codering de lengte van sleutel niet even lang te zijn als de klare tekst. In de moderne cryptografie kwam Enigma-codering een rol spelen. Dit was een techniek met rotorsystemen en werd veelal toegepast in WOII. De rotorbladen konden samen $26^3 = 17576$ mogelijk posities innemen. DES is een vorm van symmetrische versleuteling in de moderne cryptografie. Het systeem werkt volgens een block cipher (blok codering). AES en 3DES werken op een soortgelijke manier. Tegenwoordig wordt veelal RSA als versleutelingsmethode gebruikt. De veiligheid van RSA berust op het probleem van de ontbinding in factoren bij heel grote getallen.

Ook is het duidelijk geworden welke soorten cryptografieën er bestaan. Er bestaan namelijk twee soorten cryptografieën: Klassieke en Moderne cryptografie. De moderne cryptografie kan onderverdeeld worden in asymmetrische en symmetrische cryptografie. De klassieke cryptografie kan onderverdeeld worden in substitutieversleuteling en transpositieversleuteling. Bij substitutieversleuteling wordt de oorspronkelijke tekst vervangen door andere symbolen. Er bestaan twee vormen van substitutieversleuteling: mono en poly alfabetische versleuteling. Bij transpositieversleuteling draait het veranderen van de positie van de letters. Bij symmetrische cryptografie wordt dezelfde sleutel gebruikt voor zowel het coderen door de zender als decoderen door de ontvanger. Bij asymmetrische cryptografie verschilt de sleutel waarmee gecodeerd wordt en de sleutel waarmee gedecodeerd wordt.

Daarna is bevonden dat het RSA cryptosysteem is voortgekomen is uit het idee van de Diffie-Hellman cryptografie. RSA is een cryptografie, baanbrekend door het gebruik van asymmetrische cryptografie en eenrichtingsfuncties. De wiskunde achter het RSA-algoritme berust grotendeels op priemgetallen, deelbaarheid en modulo rekenen. Ook spelen Euler, Euclides en Fermat een grote rol in de wiskunde achter RSA.

Tevens is bekend dat RSA op vele gebieden inzetbaar is, zelfs buiten het versleutelen van informatie. Door de redelijk langzame snelheid van RSA kan het worden gecombineerd met een symmetrische cryptografie om slechts de sleutel te encrypten. Ook kan RSA worden ingezet bij digitale handtekeningen en certificaten. Hier wordt RSA gebruikt om informatie veilig te stellen maar versleuteld op zich niet de te beveiligen informatie.

Er zijn toch verschillende manieren gevonden om RSA te kraken, hoewel geen daarvan dat in een kort tijdsbestek kunnen. Eén van die manieren is Trial division. Voor hele grote getallen, wat bij het RSA het geval is, is er echter zeer veel tijd voor nodig. Een andere manier is Brute force hacking. Bij brute force hacking worden geen lastige berekeningen uitgevoerd, maar probeert men simpel 'met brute kracht' alle mogelijkheden uit. Theoretische zal de code eens gevonden worden, maar net als trial division kan het zeer lang duren voordat deze gevonden zal worden. Ook is Number Field Sieve (NFS) een techniek voor het kraken van RSA. NFS is een manier om samengestelde getallen te ontbinden in priemfactoren. Dit is een andere techniek dan trial division en de techniek van NFS is gebaseerd op 'verschil van twee kwadraten'. Er zijn echter oneindig veel priemgetallen, dus is het altijd mogelijk het om snelle computers moeilijk te maken. Het zal dus heel moeilijk worden om RSA te kraken. De bedenkers hebben ook een zogenaamde RSA challenge gemaakt. Door deze uitdagingen heeft RSA security een soort controle op welke grootte van priemgetallen nog als veilig beschouwd kan worden.

De toekomst van cryptografie is erg veel belovend, omdat de druk op de cryptologische vooruitgangen erg groot is. Hoewel het niveau van de huidige cryptografiesystemen net niet hoog genoeg zijn gezien de snelheid van de komende generatie computers, zullen ontwikkelingen rond cryptografie steeds versnellen, mede door schandalen als NSA en Diginotar.

Door deze informatie en de opgedane ervaringen van het maken van het RSA-versleutel programma kunnen wij onze hoofdvraag goed beantwoorden en een conclusie trekken:

RSA is op dit moment nog net veilig genoeg, zeker vergeleken met de oude cryptografie systemen die er waren voordat RSA in het leven was geroepen. Asymmetrische cryptografie is in het heden de beste techniek om te versleutelen, een techniek waar het veelgebruikte RSA onder valt. Doordat priemgetallen oneindig in aantal zijn, kan RSA altijd een stapje voor supercomputers zijn. RSA is echter niet heel efficiënt en er zijn al manieren gevonden om RSA te kraken, al zijn die manieren erg langzaam. Wel kan het zo zijn dat het versleutelen met RSA dusdanig lang en complex wordt bevonden, dat men kan overstappen naar een ander versleutelingsmethode.
De digitale beveiliging zal onder de huidige technologische snelheidsdruk steeds sneller ontwikkelen en wij verwachten dat deze steeds efficiënter zal worden, omdat RSA voor huidige standaarden nog redelijk zwaar is voor computers om informatie mee te beveiligen.

Bronnenlijst

1. http://pws.phikwadraat.nl/pws/pws%20cryptografie.pdf
2. http://lcinformatica.amsterdams.info/klas6/h17/h17.htm
3. http://nl.wikipedia.org/wiki/Cryptografie
4. http://www.kennislink.nl/publicaties/cryptografie
5. http://www.stg-ece.nl/informatica/moduled/lesd5_cryptografie.pdf
6. Aoufi, el S.(2006), "Cryptografie en ICT", Den Haag: Bim Media
7. https://mice.cs.columbia.edu/getTechreport.php?techreportID=1460
8. http://nl.wikipedia.org/wiki/Enigma_(codeermachine)
9. http://www.recursive.nl/papers/telematica.html
10. http://staff.science.uva.nl/~hansm/pubs/syllabus-minor-s.pdf (vanaf blz. 283-292)
11. http://tweakers.net/reviews/189/cryptografie-uitleg-aan-de-hand-van-rsa.html
12. http://en.wikipedia.org/wiki/RSA_Factoring_Challenge
13. http://nl.wikipedia.org/wiki/Getallenlichamenzeef
14. http://www.cs.uu.nl/docs/vakken/cry/crypto.pdf
15. http://en.wikipedia.org/wiki/Trial_division
16. http://nl.wikipedia.org/wiki/Substitutieversleuteling
17. http://www.pinkelephant.nl/iso-27002-informatiebeveiliging/cryptografie/
18. Weger, de B.M.M.(2009), "Elementaire getaltheorie en asymmetrische cryptografie", ONBEKEND
19. http://nl.wikipedia.org/wiki/Symmetrische_cryptografie
20. http://nl.wikipedia.org/wiki/Asymmetrische_cryptografie
21. http://nl.wikipedia.org/wiki/Transpositieversleuteling
22. Aldine, A. & C. Kraaikamp (2013), "Verborgen boodschappen", Den Haag: Epsilon Uitgaven
23. http://www.rsaconference.com/blogs
24. http://answers.yahoo.com/question/index?qid=20111203161258AAz61tD
25. http://pinpointlabs.com/2010/12/what-is-a-hash-value/
26. http://www.nrc.nl/tech/2011/09/03/digitaal-certificaat-het-krakend-slot-op-de-browser/

Bijlage

Code RSA-programma (*1 functie, 5 private subs*)

```
'Globale Declaratie van variabelen
  Dim p1 As Long
  Dim p2 As Long
  Dim n As Long
  Dim phien As Long
  Dim e As Long
  Dim d As Long
```

FUNCTIE 1

```
'Testfunctie voor priemgetallen
Public Function Priem(ByVal Testpriem As Long) As Boolean
   'declaraties variabelen
   Dim deler As Long
   Dim grens As Long

   'Herkennen van het priemgetal 2
   If Testpriem = 2 Then
      Priem = True
      Exit Function
   'Herkennen van even getallen (die zijn nooit priem)
   ElseIf Testpriem Mod 2 = 0 Then
      Exit Function
   End If

   'Loop door alle even getallen, beginnend met 3
   deler = 3
   grens = Testpriem
   Do While grens > deler 'Grens opstellen van de Loop

      If Testpriem Mod deler = 0 Then
         Exit Function
      'De functie stopt als TestPriem deelbaar is
      End If

      'De grens wordt verkleind, door te delen door de huidige deler.
      'Dit omdat de huidige deler geen deler is van Testpriem
      grens = Testpriem \ deler

      'Deler kan met twee stappen vergroot worden, omdat al gecheckt is of Testpriem een even getal is.
      deler = deler + 2
   Loop

   'Als het gelukt is om door de loop heen te komen, dan is Testpriem een priemgetal.
   Priem = True
End Function
```

PRIVATE SUB 1

```
Private Sub cmdNphieN_Click()
  'Inlezen P en Q
  p1 = CInt(txtP.Text)
  p2 = CInt(txtQ.Text)

  'Controle en berekenen
  If p1 > 99 Then 'Check of P een getal is onder de 100 (dit is om snelheid en eenvoudigheid vh programma) te
bevorderen
     txtP.Text = ""
     MsgBox "P is géén priemgetal onder de 100. Voer een juist getal in. Check gelijk of Q ook klopt."
     Exit Sub

  ElseIf p2 > 99 Then 'Idem als boven, maar dan voor Q.
     txtQ.Text = ""
```

```
    MsgBox "Q is géén priemgetal onder de 100. Voer een juist getal in."
    Exit Sub
  ElseIf Priem(p1) = True And Priem(p2) = True Then 'p1 en p2 checken met functie Priem
    'Berekenen n en phie(n)
    n = p1 * p2
    phien = (p1 - 1) * (p2 - 1)
    txtN.Text = Str(n)
    txtphieN.Text = Str(phien)
    Exit Sub

  ElseIf Priem(p1) = False Then 'Alleen p1 is niet priem
    txtP.Text = ""
    MsgBox "Het eerste getal is geen priemgetal onder de 100. Voer een juist getal in."
    Exit Sub

  ElseIf Priem(p2) = False Then 'Alleen p2 is niet priem
    txtQ.Text = ""
    MsgBox "Het tweede getal is geen priemgetal onder de honderd. Voer een juist getal in."
    Exit Sub

  Else 'Enige mogelijkheid die over is, is dat p1 én p2 niet priem zijn
    txtP.Text = ""
    txtQ.Text = ""
    MsgBox "Beide getallen zijn geen priemgetallen onder de honderd. Voer juiste getallen in."
    Exit Sub
  End If

End Sub
```

PRIVATE SUB 2

```
Private Sub cmdE_Click()
  'Declaraties van lokale variabelen
  Dim gr As Long
  Dim kl As Long
  Dim rest As Long

  'Definiëren e en kl. e moet kleiner zijn dan n.
  e = n - 1
  kl = e

  'Algoritme van Euclides met Loop en While...Wend om e te berekenen. e is priem aan phie(n) en kleiner dan n.
  While gr <> 1
    gr = phien
      Do 'De Do..Loop is het algoritme van Euclides
        rest = gr Mod kl
        gr = kl
        kl = rest
      Loop While rest <> 0
    e = e - 1
    kl = e 'stappen van 1 bij het vinden naar het getal e, waarvoor geldt ggd(phie(n), e) = 1 (relatief priem)
  Wend

  'Uiteindelijke waarde van e is kl + 1 ,
  'omdat in de While...Wend aan het einde 1 wordt afgetrokken van e, ook als het antwoord is gevonden
  e = kl + 1

  'Uitvoer
  txtE.Text = Str(e)

End Sub
```

PRIVATE SUB 3

```
Private Sub cmdD_Click()
   'We checken de deelbaarheid van (e*d - 1) en phie(n).
   'Dit doen we door te kijken naar de gelijkwaardighid van het quotient en Int(quotient)
   'We beginnen met d = 0 en tellen daar steeds 1 bij op; er zijn meerdere waarden van d mogelijk. We gaan
voor de positieve waarde
   d = 0

   'Door middel van een Loop kunnen we vergelijken totdat we de juiste d hebben
   Do
      d = d + 1
   Loop While (((d * e) - 1) / phien) <> Int((((d * e) - 1) / phien))

   'Een message box verschijnt met N, E en D. Daarna resetten we de waarden van p1, p2 en phien (P, Q,
phie(n))
   MsgBox "De publieke encryptiesleutel (N,E) is (" & Str(n) & "," & Str(e) & "). Je eigen decrytpiesleutel (N,D) is
(" & Str(n) & "," & Str(d) & "). Goed onthouden!"

   txtP.Text = ""
   txtQ.Text = ""
   txtphieN = ""
   p1 = p2 = phien = d = 1

End Sub
```

PRIVATE SUB 4

```
'versleutelen met rsa algoritme
Private Sub cmdC_Click()

   'Declaraties lokale variabelen
   Dim n As Long
   Dim e As Long
   Dim m As Long
   Dim m1 As Long
   Dim m2 As Long
   Dim m3 As Long
   Dim c1 As Long
   Dim c2 As Long
   Dim c3 As Long
   Dim ms As String
   Dim cs As String

'inlezen M
   ms = txtME.Text

 'Controle lengte M, niet meer dan 3 om snelheid te bevorderen
If Len(ms) <> 3 Then
   txtME.Text = ""
   MsgBox "Voer een getal van 3 cijfers in om de snelheid van het programma te bevorderen!"

Else
      m1 = CInt(Mid(ms, 1, 1))
      m2 = CInt(Mid(ms, 2, 1))
      m3 = CInt(Mid(ms, 3, 1))

      n = CInt(txtNE.Text)
      e = CInt(txtEE.Text)

     'rsa algoritme
      c1 = ((m1) ^ (e)) Mod n
      c2 = ((m2) ^ (e)) Mod n
      c3 = ((m3) ^ (e)) Mod n
```

```
      'uitvoer
       cs = Str(c1) + Str(c2) + Str(c3)
       txtCE.Text = cs

      'legen tekstvak M
       txtME.Text = ""

   End If

End Sub
```

PRIVATE SUB 5

```
Private Sub cmdM_Click()
   'Declaraties lokale variabelen
   Dim n As Long
   Dim d As Long
   Dim m As Long
   Dim m1 As Long
   Dim m2 As Long
   Dim m3 As Long
   Dim c1 As Long
   Dim c2 As Long
   Dim c3 As Long
   Dim ms As String
   Dim cs As String

'inlezen C
   cs = txtCD.Text

 'Controle lengte C en ontcijferen met rsa
   If Len(cs) <> 3 Then
      txtCD.Text = ""
      MsgBox "Voer een getal van 3 cijfers in, net als het berekende C!"

   Else
       c1 = CInt(Mid(cs, 1, 1))
       c2 = CInt(Mid(cs, 2, 1))
       c3 = CInt(Mid(cs, 3, 1))

       n = CInt(txtND.Text)
       d = CInt(txtDD.Text)

     'rsa algoritme
       m1 = ((c1) ^ (d)) Mod n
       m2 = ((c2) ^ (d)) Mod n
       m3 = ((c3) ^ (d)) Mod n

     'uitvoer
       ms = Str(m1) + Str(m2) + Str(m3)
       txtMD.Text = ms

     'legen tekstvak C
       txtCD.Text = ""

   End If
End Sub
```

Terminologie

A

Advanced Encryption Standard (AES): Een methode om gegevens te versleutelen en de opvolger van de AES.

Algoritme: Een reeks instructies die moet worden uitgevoerd om een doel te bereiken. Een voorbeeld is het versleutelen van informatie middels het RSA-algoritme.

Algoritme van Euclides: Het algoritme van Euclides is een efficiënte methode voor het berekenen van de grootste gemene deler (GGD) van twee positieve gehele getallen. Het algoritme berust erop dat de GGD van twee gehele getallen ook de GGD is van zowel het kleinste- en het restgetal (bij deling van het grootste getal door het kleinste getal).

ASCII-(tabel): Een standaard om letters, cijfers, leestekens en enkele andere tekens aan ieder teken in die reeks een geheel getal te koppelen, waarmee dat teken kan worden aangeduid.

Asymmetrische cryptografie: Bij Asymmetrische cryptografie verschilt de sleutel waarmee gecodeerd wordt en de sleutel waarmee gedecodeerd wordt. Er worden dus twee sleutels gebruikt.

Authenticiteit: Het feit dat gebruikers zich moeten aanmelden (legitimeren) voor gebruik.

B

Block cipher: Codering in de vorm van 'blokken'. Er worden 'blokken' van 64 bits gecodeerd.

Brute force hacking: Bij brute force hacking worden geen lastige berekeningen uitgevoerd, maar probeert men simpel 'met brute kracht' alle mogelijkheden uit.

Bit: De bit is de kleinste eenheid van informatie, namelijk een symbool of signaal dat twee waarden kan aannemen.

C

Cijfertekst: Ook wel de vercijferde tekst of versleutelde tekst.

Cryptografie: Houdt zich bezig met technieken voor het verbergen of zodanig versleutelen van te verzenden informatie, dat het voor een persoon die toegang wilt hebben tot het kanaal tussen zender en ontvanger, onmogelijk is uit de getransporteerde data af te leiden welke informatie er door de zender was verzonden en welke partijen daarbij betrokken waren.

D

Data Encryption Standard (DES): Een methode om gegevens met symmetrische cryptografie te versleutelen, ontworpen door IBM.

Dubbele transpositie: Is een klassiek handcijfer. Het dubbele-transpositiecijfer bestaat uit twee verschillende kolom-transposities. Men kan hierbij hetzelfde sleutelwoord voor beide stappen gebruiken, of twee verschillende sleutelwoorden kiezen.

Decryptie: Het ontsleutelen of ontcijferen van informatie

E

Encryptie: Het versleutelen van informatie

F

Factoriseren: De factorisatie of het ontbinden in factoren van een product, is het herschrijven van dat product in kleinere delen, die met elkaar vermenigvuldigt weer het oorspronkelijke product opleveren.

F

Fermatgetal: Een fermatgetal is een natuurlijk getal van de vorm $F_n = 2^{(2n)}$. Fermat vermoedde dat elk fermatgetal een priemgetal is. Zijn vermoeden is onjuist gebleken, maar klopt wel voor de eerste vijf Fermat getallen.

Frequentieanalyse: Frequentieanalyse in houdt dat men onderzoek doet naar de frequentie van symbolen of groepen symbolen van een versleutelde tekst.

G

Getallenzeeflichaam: Zie *Number Field Slieve*.

GGD: De grootste gemene deler, afgekort tot GGD, is het grootste positieve gehele getal, waar al deze gehele getallen door gedeeld kunnen worden zonder dat er een rest overblijft. De grootste gemene deler van de getallen 8 en 12 is bijvoorbeeld 4.

H

Hashwaarde: Een waarde die kenmerkend is voor digitale informatie, zoals de vingerafdruk van een mens. Deze waarde wordt berekend middels het hashalgoritme.

I

IDEA: International Data Encryption Algorithm, is een encryptiemethode ontwikkeld in Zwitserland en is op een vergelijkbare manier als DES opgebouwd.

K

Klare tekst: Hiermee wordt de oorspronkelijke, nog niet vercijferde tekst bedoeld.

Kolomtranspositie: Deze techniek komt vaak voor in de klassieke cryptografie en bestaat uit transpositie cijfer en de dubbele transpositie.

M

Modulo-rekenen: Modulo-rekenen is een manier van rekenen waarbij een getal als bovengrens fungeert. Dit getal noemen we de modulus. Na dit getal begint men weer opnieuw vanaf 0 te tellen. Je zou kunnen zeggen dat onze uur telling een vorm van modulo-rekenen is: Na 24 uur beginnen we weer opnieuw te tellen.

Mono-alfabetische versleuteling: Elke letter is vervangen door een ander letter of symbool. Een voorbeeld hiervan is de Caesar-Methode.

N

Number Field Sieve (NFS): NFS is een manier om samengestelde getallen te ontbinden in priemfactoren. Techniek van NFS is gebaseerd op 'verschil van twee kwadraten'.

P

PGP: Een cryptosysteem dat gebruik maakt van RSA en een symmetrisch cryptosysteem.

Poly-alfabetische versleuteling: Hierbij wordt juist gebruikt gemaakt van versleutelingsalfabetten, vaak in de vorm van tabellen. Zo wordt een de boodschap meerder keren versleuteld. Een voorbeeld hiervan is Vigenère- code.

Priemgetallen: Een priemgetal is een natuurlijk getal groter dan 1 dat slechts deelbaar is door 1 en door zichzelf.

Public-key cryptografie: Een vorm van asymmetrische cryptografie waarbij de encryptiesleutel openbaar wordt gemaakt. RSA valt hieronder.

R

Red and Purple: Soortgelijke machine als de Enigma, alleen de Japanse Versie daarvan.

S

Sigaba: Soort gelijke machine als de Enigma, alleen de Amerikaanse versie daarvan

S/MIME: Een cryptosysteem dat berust op het gebruik van RSA en symmetrische cryptografie. Ook maakt S/MIME gebruik van gecertificeerde sleutels.

Substitutieversleuteling: Zoals de naam al verraad, wordt bij substitutieversleuteling de oorspronkelijke tekst vervangen door andere symbolen. Dit hoeft niet per se het alfabet te zijn, vandaar de symbolen.

Symmetrische cryptografie: Bij symmetrische cryptografie wordt dezelfde sleutel gebruikt voor zowel het coderen door de zender als decoderen door de ontvanger.

Symmetrische sleutels: De sleutel, die zowel voor het coderen als voor het decoderen wordt gebruikt.

T

Transpositieversleuteling: Bij transpositieversleuteling draait het om veranderen van de positie van de letters.

Transpositiecijfer: Bij transpositie worden letters of tekens van plaats verwisseld met andere letters of tekens.

Trial division: Ontcijferingstechniek, waarbij grote getallen worden ontbonden in de hoop dat de deler de klare tekst is.

V

Vigenère- tableau: Een tabel waarmee poly-alfabetisch versleuteld kan worden. Dit gebeurt door het snijpunt tussen de klare tekst en de sleutel.

Z

$\mathbb{Z}$: De verzameling van alle gehele getallen

Overig

3DES: Een encryptiealgoritme. Het maakt gebruik van Data Encryption Standard of kortweg DES-algoritme, maar op zodanige wijze dat het veel moeilijker te kraken is.