4 Cryptografische beveiliging op het Internet
Benne de Weger

4.1 Hybride cryptografie

In de hoofdstukken 1 en 2 hebben we alle basisingrediënten gezien die de moderne cryptografie te bieden heeft, bijvoorbeeld voor het beveiligen van het World Wide Web en toepassingen als e-mail en WhatsApp[1]. Het gaat daarbij niet alleen om versleutelen, maar ook om digitale handtekeningen. En versleutelen kwam in twee smaken: symmetrisch (zoals AES) en asymmetrisch (zoals RSA).

Het grote voordeel van symmetrische versleuteling is snelheid. Het grote nadeel ervan is de onhandigheid van sleuteluitwisseling. Bij asymmetrische methoden zoals RSA is dat precies andersom: daar is de sleuteluitwisseling juist heel handig, maar het is veel te langzaam.

RSA kan overigens best, net als Diffie-Hellman, voor sleuteluitwisseling worden gebruikt, in plaats van het direct versleutelen van de gegevens zelf. Als Alice en Bob een symmetrische sleutel **k** willen uitwisselen voor gebruik bij AES, dan kan dat bijvoorbeeld als volgt: Alice maakt een random sleutel **k** (voor ieder te versturen bericht een andere), Bob maakt een RSA-sleutelpaar, Bob stuurt zijn publieke sleutel naar Alice, Alice versleutelt de symmetrische sleutel **k** met Bob's publieke sleutel, en stuurt die naar Bob toe over het onveilige Internet. Bob kan dat geheimschrift ontsleutelen en krijgt zo een perfecte kopie van **k**, terwijl de afluisterende Eva het nakijken heeft.

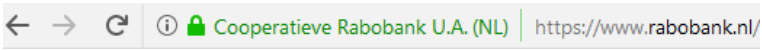
Alice kan natuurlijk met de sleutel **k** een echt bericht **m** versleutelen (met AES, lekker snel), en dat meteen aan Bob meesturen. Deze methode combineert de voordelen van symmetrische en asymmetrische cryptografie. Dat de asymmetrische methode langzaam is is niet zo heel erg, omdat ze alleen gebruikt wordt voor het versleutelen van een heel kleine symmetrische sleutel: dat is nog wel acceptabel.

[1] Herinnering: het Web is niet hetzelfde als het Internet; het Web is slechts één van de toepassingen van het Internet, naast bijvoorbeeld e-mail.

Deze gemengde toepassing van symmetrische en asymmetrische cryptografie heet *hybride cryptografie*. Bekende software als PGP – Pretty Good Privacy – werkt op deze manier.

4.2 Het TLS-protocol voor website-beveiliging

TLS betekent Transport Layer Security. Het is een protocol (vroeger SSL geheten) dat overal op het web (en verder op het Internet) gebruikt wordt voor cryptografische beveiliging. Denk aan internetbankieren: je wilt een bedrag overmaken aan iemand, dan moet je inloggen op de website van je bank en vervolgens, als je de betalingsopdracht hebt klaargezet, één of andere code intypen om de betaling te autoriseren. Je hebt dan twee belangrijke beveiligingseisen: natuurlijk wil je dat de gegevens die je over het onveilige Internet stuurt vertrouwelijk behandeld worden; niemand anders dan de bank heeft er iets mee te maken aan wie jij hoeveel geld geeft. Maar, veel belangrijker nog: je wilt zeker zijn dat je met de echte bank verbonden bent, en niet met een door criminelen nagemaakte website die bedrieglijk echt lijkt maar dat niet is, en daar vrolijk je wachtwoorden en autorisatiecodes afgeeft (in het verleden zijn dit soort dingen gebeurd!). Met andere woorden: je wilt dat de bank zichzelf aan jou authenticeert, bewijst dat het de echte bank is. Dat is wat het TLS-protocol verzorgt. Het is de techniek achter het slotje in de adresbalk van je browser, achter de web-adressen die niet met `http://` beginnen maar met `https://` (de `s` van `s`ecure) en de groene adresbalk (groen is veilig).



Figuur 4.1: TLS aan het werk.

Op de achtergrond gebeurt ongeveer het volgende, het zogenaamde handenschudprotocol.

- De webbrowser (jij dus) stuurt een verzoek naar de webserver (de bank) om een beveiligde webpagina. In dat verzoek geeft de webbrowser een random getal r_1 dat hij zojuist vers aangemaakt heeft, en een lijstje van cryptografische algoritmen die hij kent.
- De webserver antwoordt met een ander random getal r_2 dat hij zojuist vers aangemaakt heeft, en zijn keuze uit de cryptografische algoritmen: de sterkste die zowel webbrowser als webserver kennen.
- De webserver stuurt zijn publieke sleutel (eigenlijk: zijn certificaat, zie hieronder) naar de webbrowser.

- De webbrowser maakt nu een nieuw random getal r_3, dat versleutelt hij met de publieke sleutel van de bank, en stuurt het op.
- De webserver en de webbrowser hebben nu beide de drie random getallen r_1, r_2, r_3, ze hebben beide bijgedragen aan de gegenereerde randomness. Uit deze drie getallen berekenen ze beide op dezelfde manier een aantal symmetrische sleutels, die in het vervolg gebruikt worden, zowel voor versleuteling van gegevens als voor een vorm van symmetrische authenticatie.
- De webserver en de webbrowser sturen elkaar een eindboodschap, met daarin een symmetrische handtekening die berekend wordt over *alle* voorgaande berichten. Deze eindboodschappen zijn al symmetrisch versleuteld. Hiermee is het handenschudden klaar.
- Vanaf dit moment kunnen de berichten die webbrowser en webserver aan elkaar willen sturen, volledig symmetrisch versleuteld en symmetrisch geauthenticeerd worden.

Wat opvalt in dit protocol is dat nergens een asymmetrische digitale handtekening wordt gezet. De authenticatie gebeurt in zekere zin door de asymmetrische versleuteling: allen de webserver met de juiste privé-sleutel kan het correcte getal r_3 berekenen, waarop alle verdere sleutels gebaseerd zijn. De toepassing van asymmetrische cryptografie wordt tot het uiterste beperkt, omdat dat nu eenmaal erg duur is.

Symmetrische authenticatie werkt bijvoorbeeld als volgt: als Alice en Bob al een symmetrische sleutel hebben uitgewisseld, kan Alice een hash van een bericht daarmee versleutelen, en dat geeft dan een handtekening; Bob kan die controleren door de hash van het bericht opnieuw te berekenen en te vergelijken met de ontsleutelde handtekening.

Zodra er ergens in het protocol iets foutgaat, zal de webbrowser of de webserver de verbinding verbreken en een foutmelding geven.

4.3 De mens in het midden

Tot nu toe lijkt het allemaal prachtig, die asymmetrische cryptografie: mooie veilige sleuteluitwisseling over een onveilig kanaal. Er is echter een enorme roze olifant in de kamer. Die gaan we nu aanwijzen, en vervolgens proberen lek te prikken.

Nog even terughalen hoe dat ging met standaard RSA-versleuteling: Alice

wil een geheim bericht $\mathbf{m}$ versleuteld naar Bob sturen. Bob stuurt zijn publieke sleutel $\mathbf{n}, \mathbf{e}$ over dat onveilige kanaal naar Alice op, zodat Alice het bericht $\mathbf{m}$ kan omzetten in een geheimschrift $\mathbf{c}$. Het is niet erg dat Eva, de afluisteraar, zowel de publieke sleutel $\mathbf{n}, \mathbf{e}$ als het geheimschrift $\mathbf{c}$ te zien krijgt: ze kan er niets leuks mee.

Maar Eva kan wel iets anders doen, als ze *actief* wordt in plaats van passief blijft afluisteren. Eva ziet eerst de publieke sleutel van Bob langskomen, waarschijnlijk in een berichtje waarin Bob vertelt wat hij aan Alice opstuurt, iets als:

> Lieve Alice, hierbij mijn publieke sleutel: $\mathbf{n}, \mathbf{e}$. Groetjes, Bob.

Wat Eva dan kan doen is erg gemeen: ze kan dit berichtje even tegenhouden, en veranderen: Eva zorgt dat ze zelf een sleutelpaar $\mathbf{n}', \mathbf{e}', \mathbf{d}'$ heeft, en ze vervangt in Bob's bericht de publieke sleutel van Bob door haar eigen publieke sleutel. Dan stuurt Eva het volgende bericht door aan Alice:

> Lieve Alice, hierbij mijn publieke sleutel: $\mathbf{n}', \mathbf{e}'$. Groetjes, Bob.

Alice ontvangt dit bericht, en ziet nergens aan dat het gemanipuleerd is, ze denkt in al haar argeloosheid dat $\mathbf{n}', \mathbf{e}'$ de publieke sleutel van Bob is. Ze schrijft een brief aan Bob:

> Lieve Bob, ik hou van je. X X X, Alice.

versleutelt die met $\mathbf{n}', \mathbf{e}'$ tot een geheimschrift, en stuurt het geheimschrift het Internet op, naar Bob. Tenminste, dat denkt ze, want in feite onderschept Eva het bericht, en Eva kan het ontsleutelen (met $\mathbf{d}$;), en krijgt de klare tekst te lezen. Eva kan het bericht zelfs veranderen: ze maakt er bijvoorbeeld het volgende van:

> Bob, ik vind je niet leuk meer. Ga maar met die Eva. De groeten, Alice.

en Eva versleutelt dit bericht met de échte publieke sleutel van Bob, $\mathbf{n}, \mathbf{e}$. Eva stuurt het door naar Bob, die ontsleutelt het met succes (met $\mathbf{d}$), en denkt dat dit bericht echt van Alice komt. Bob heeft geen enkele reden om anders aan te nemen[2].

Dit probleem is levensgroot. Het staat bekend als het *mens-in-het-midden*-probleem. Het is bijvoorbeeld heel makkelijk om via openbare, niet beveiligde wifi-verbindingen de mens in het midden te gaan spelen. Denk hier eens aan, de eerstvolgende keer als je in een hotel zonder verder na te denken de daar gratis aangeboden onbeveiligde wifi gebruikt. Dit geldt net zo

[2] Behalve als Bob deze cursus volgt, natuurlijk.

in het TLS-protocol, dat je gebruikt om te internetbankieren: daar stuurt de bank je een publieke sleutel toe, maar wie garandeert je dat daar geen criminele organisatie tussen zit, die mens-in-het-midden aan het spelen is?

Wat is in cryptografische zin nu precies het probleem? Dat de publieke sleutel van Bob vervangen kon worden. Die sleutel is niet geheim, vertrouwelijkheid is hier niet aan de orde. Maar wat wel van groot belang is, is dat Alice zekerheid krijgt dat de sleutel die ze ontvangt, echt die van Bob is en niet van die stomme Eva. Maar dat is een authenticatie- en integriteitsprobleem. En daar heeft de cryptografie een oplossing voor: digitale handtekeningen.

Ziedaar de roze olifant. En de speld om hem lek te prikken. Hoe werkt dat dan? Er moet een handtekening komen onder een bericht van de vorm

> Ik verklaar hierbij dat de publieke sleutel $\mathbf{n}, \mathbf{e}$ van Bob is. Getekend, X.

Maar wie is die X? En hebben we nu niet opnieuw een Baron von Münchhausen-probleem als we zo'n digitale handtekening gaan zetten met een asymmetrisch cryptografisch systeem waar weer een andere publieke sleutel achter zit: kan díe niet ook ge-mens-in-het-midden-d worden?

4.4 Certificaten

De oplossing die hiervoor in het leven geroepen is, en die in feite veilig Internet pas écht mogelijk maakt, heet PKI – Public Key Infrastructure. Het belangrijkste concept daarin is het *certificaat*. Een certificaat is een digitaal document dat in ieder geval de volgende informatie bevat:

serial number serienummer,

signature algorithm combinatie van hashfunctie en asymmetrische handtekening-methode, zoals sha256RSA: hashfunctie SHA256, handtekening gezet met RSA

issuer de uitgever, die garant staat voor de handtekening,

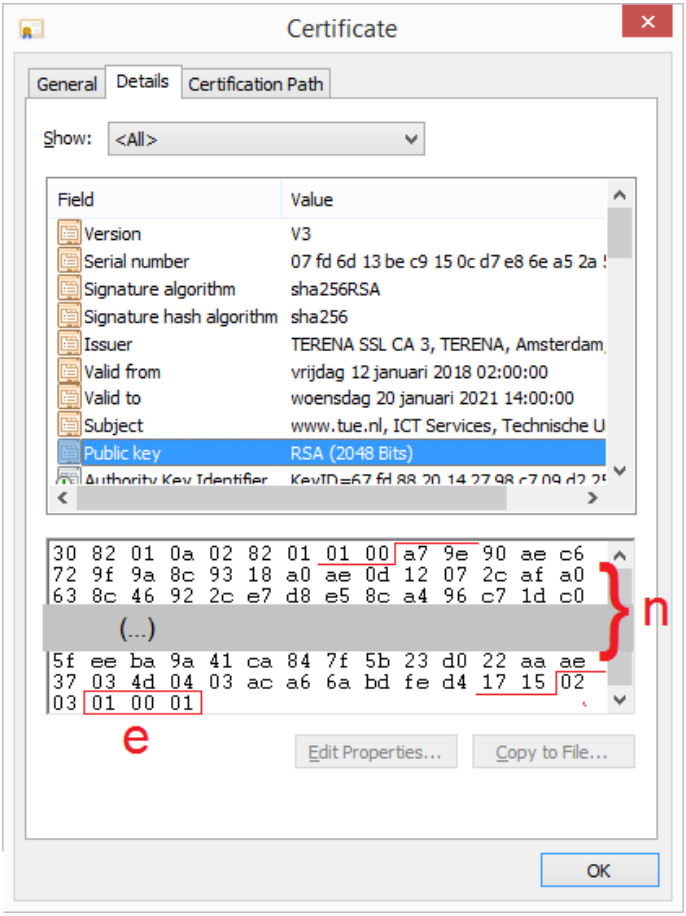
valid from/to geldigheidsduur,

subject naam van de eigenaar, zoals " www.tue.nl", of "Bob",

public key de publieke sleutel, in volle glorie, $\mathbf{n}$ en $\mathbf{e}$, in klare tekst

en natuurlijk de digitale handtekening, die browsers nooit laten zien[3]

[3] Ik heb geen idee waarom ze dat niet doen.



Figuur 4.2: Het nu geldende certificaat van de website van de TU/e.

Je ziet dat de TU/e zijn certificaat gekregen[4] heeft van een instantie die Terena SSL CA heet. De term “CA” staat hier voor “Certification Authority”. Terena is gewoon een bedrijf, dat op commerciële basis certificaten maakt en verkoopt. Zij staan garant voor de echtheid ervan. Zij hebben via procedurele weg gecontroleerd dat de publieke sleutel die in dit certificaat staat, echt van de TU/e is (wellicht is een TU/e-medewerker met een uittreksel van de Kamer van Koophandel en een door het College van Bestuur ondertekende machtiging en zijn eigen paspoort naar Amsterdam afgereisd om Terena ervan te overtuigen dat het goed zit).

[4]Nou ja, gekocht natuurlijk.

De handtekening onder dit certificaat is dus gemaakt met de privé-sleutel van Terena. Als het goed is heeft Terena netjes beschreven hoe zij de opslag van die privé-sleutel beveiligd hebben, want dat is natuurlijk een uiterst belangrijke sleutel. Er bestaat speciale hardware (zogenaamde HSMs: Hardware Security Module) voor opslag en gebruik van dit soort sleutels, met allerlei zware fysieke beveiliging ingebouwd. En als het goed is, laat Terena zijn procedures én de uitvoering daarvan regelmatig door een onafhankelijke partij auditen.

Figuur 4.3: Een soort HSM.

De publieke sleutel van Terena moet natuurlijk ook vertrouwd worden. Dat kan, door deze via een certificaat uitgegeven door een hogere instantie, een hogere Certification Authority, te laten ondertekenen. In het geval van Terena is dat DigiCert.

De publieke sleutel van DigiCert moet natuurlijk ook vertrouwd worden. Dat kan, door deze via een certificaat uitgegeven door een hogere instantie, een hogere Certification Authority, te laten ondertekenen. In het geval van DigiCert is dat ... DigiCert. Het is logisch dat dit een keer moest ophouden. DigiCert ondertekent zijn eigen certificaten. DigiCert is een zogeheten "Trusted Root CA".

Hoezo is DigiCert vertrouwd? Had jij er ooit van gehoord, van dat bedrijf? Toch vertrouwt de hele wereld het, voor allerlei certificaten die onder het zelf-ondertekende root-certificaat van DigiCert hangen. En DigiCert is niet de enige: in je operating system (Windows, Linux, MacOS) zit een hele database met vertrouwde zelf-ondertekende root-certificaten, een grote waslijst. In mijn browser zie ik er al meer dan 50 (waaronder

die van de “Staat der Nederlanden”). Kennelijk krijg je die meegeleverd met je computer. Dan vertrouwen we er dus in feite op dat Microsoft (of Apple, of de Linux-verantwoordelijken, wie dat ook mogen zijn) in die database alleen root-certificaten opnemen van CA’s die voldoen aan allerlei procedurele eisen.

Overigens zit er in een certificaat ook een veld dat aangeeft waar het voor gebruikt mag worden, bijvoorbeeld alleen voor website-beveiliging, of alleen voor e-mail-beveiliging. Er is een speciale code voor autorisatie als CA-certificaat, en voor root-certificaat, zodat je die als gewone sterveling niet zomaar kunt krijgen.

4.5 En toch kan het misgaan

Hee, wat dacht je dan. Natuurlijk kan het misgaan. Meestal ligt dat niet aan de cryptografie, maar zelfs dat is niet uitgesloten. Twee voorbeelden.

Een bekende Nederlandse CA was het bedrijf DigiNotar, opgericht vanuit de notarissenwereld. Notarissen leven van de handel in vertrouwen, dus dat lijkt een natuurlijke gedachte: om als notaris ook digitale certificaten te gaan uitgeven. Van 1997 tot 2011 ging dat ook goed: het was een door iedereen gerespecteerde CA, die zelfs certificaten mocht verzorgen voor de Nederlandse Overheid. Maar toen kwam er een hacker langs, die wist in te breken in de DigiNotar-systemen. Natuurlijk had DigiNotar ook HSMs om hun supergevoelige privé-sleutels te beschermen. Die sleutels zijn dan ook niet gelekt. Maar de toegang tot de ondertekeningssoftware (die de ondertekeningsopdrachten doorgeeft aan de HSM) lag wel open voor de hacker. Die kon vervolgens een officieel door DigiNotar ondertekend, en dus via Microsoft enz. door de hele wereld vertrouwd, certificaat aanmaken op naam van `*.google.com`, en meer nep-certificaten. Daarmee zou je een mens-in-het-midden-aanval kunnen uitvoeren op Google, en mensen op allerlei andere manieren misleiden. Dat is toen ook gebeurd, met enkele honderden nep-certificaten die vooral in Iran verspreid zijn. Er gaan geruchten dat de Amerikaanse geheime diensten erachter zouden hebben gezeten. Hoe dan ook, door de nogal lakse reactie van DigiNotar zegde de Nederlandse overheid al gauw het vertrouwen in DigiNotar op. Enkele weken na de hack was het, eerst florerende, bedrijf failliet.

Deze DigiNotar-aanval heeft op zich niets met falende crypto te maken: de crypto was in orde. Ons tweede voorbeeld is er een van falende crypto.

Een bekende hashfunctie is MD5. Ontworpen in 1991 door Ron Rivest (de

R van RSA), werd het al gauw het werkpaard van de internetbeveiliging. Begin deze eeuw was een groot gedeelte van de op het Internet gebruikte certificaten gebaseerd op handtekeningen gemaakt met een MD5-hash.

In 2004 was er, zoals ieder jaar, in augustus in Santa Barbara, Californië, de "Crypto"-conferentie. Op dit soort top-conferenties is er altijd een "rump session", waar iedereen onaangekondigd iets mag vertellen, als het maar niet langer duurt dan een minuut of 3. Op deze rump session kwam er een relatief onbekende Chinese dame, prof. Xiaoyun Wang, met een korte aankondiging: hier zijn twee verschillende (random uitziende) berichten $\mathbf{m}_1, \mathbf{m}_2$ (in slechts enkele bits verschillend, maar toch) met dezelfde MD5-hash.

Toen brak de pleuris[5] uit in cryptoland.

Driekwart jaar later publiceerde Wang een artikel met tot in detail haar methode uitgelegd. De vraag was wel wat je daar nou direct voor kwaad mee zou kunnen, met die botsingen die nu min of meer op bestelling gemaakt konden worden, want de methode levert geen "zinvolle" botsingen op, alleen random uitziende berichten $\mathbf{m}_1, \mathbf{m}_2$. Arjen Lenstra en ondergetekende bedachten toen het volgende: in een certificaat zit een publieke sleutel, die ziet er ook erg random uit. Zou je dit soort "botsende" berichten niet in zo'n publieke sleutel kunnen verstoppen, op zo'n manier dat enerzijds je er toch echt geldige publieke sleutels van krijgt, en anderzijds er twee verschillende certificaten mee kunt maken met precies dezelfde MD5-hash, en dus precies dezelfde digitale handtekening? Dat konden we inderdaad, hoewel het hier eenvoudiger klinkt dan het in werkelijkheid was. Maar daarmee hadden we aangetoond dat willekeurig uitziende berichten toch wel degelijk voor misbruik kunnen dienen. Onze student Marc Stevens heeft vervolgens de methode van Wang zoveel verscherpt dat het berekenen van de botsingen heel veel sneller werd (seconden op een PC in plaats van uren op een supercomputer), en hij kon een veel flexibeler type botsingen maken. In 2008 hebben we de wereldpers daarmee gehaald, door bij een Certification Authority die (4 jaar na dato!) nog steeds op MD5 gebaseerde certificaten verkocht, een speciaal certificaat aan te schaffen, dat alleen gebruikt mocht worden voor website-beveiliging, maar waar wij een speciaal soort botsing erin verstopt hadden, waardoor we de handtekening direct konden overzetten in een door ons gefabriekt certificaat dat geautoriseerd was als certificaat-uitgevend certificaat. En dat met een officiële handtekening van een echte CA, die alleen niet voor dat doel was uitgegeven.

[5] Excusez le mot.

Vijf dagen nadat we dit publiek hadden gemaakt, is de betreffende CA gestopt met het gebruik van MD5. Sommige bedrijven hebben kennelijk af en toe een zetje nodig om veilig te worden. En daar was het ons om te doen: je maakt de wereld niet veiliger door problemen in beveiligingssystemen te verzwijgen, maar juist door ze openbaar te maken.

Overigens heeft Marc Stevens later een proefschrift geschreven waarin hij het nog eens dunnetjes overdoet voor SHA1; in 2012 kwam hij met de eerste botsing voor SHA1. Dat was aanzienlijk veel lastiger dan voor MD5. Gelukkig heeft SHA256 een zodanig ander ontwerp dan MD5 en SHA1 dat de technieken van Wang en Stevens daar vooralsnog vruchteloos zijn gebleven.

Ook in 2012 werd het Flame-virus ontdekt, sterk gerelateerd aan het Stuxnet-virus. Uit analyses bleek dat Flame ook op MD5-botsingen gebaseerde nep-certificaten gebruikte, met als doel kwaadaardige software te kunnen installeren op bepaalde Microsoft-systemen (vaak worden certificaten ook gebruikt om de oorsprong van software te kunnen garanderen). Vermoedelijk is Flame / Stuxnet ontwikkeld en in omloop gebracht door geheime diensten van de Verenigde Staten en Israel, om nucleaire installaties in landen als Iran te kunnen besmetten. Je mag ervan vinden en geloven wat je wilt, maar de elektronische oorlogsvoering lijkt allang aan de gang te zijn, en het komt zo wel erg dichtbij, als je iets wat toch wel verdacht veel lijkt op het uitstekende werk van je student zo ineens in een bloedserieus virus terugziet. Ik sta nog steeds achter de volgende stelling bij mijn proefschrift (uit 1988):

12.
De eerste wereldoorlog wordt wel de oorlog van de scheikundigen genoemd, de tweede die van de natuurkundigen. Gezien de huidige ontwikkelingen in de wapentechnologie ziet het er naar uit dat een eventuele derde wereldoorlog de oorlog van de wiskundigen genoemd zal kunnen worden. De gangbare indeling van de exacte wetenschappen doet dan ook al vermoeden dat dat wel eens de laatste wereldoorlog zou kunnen zijn.

4.6 Gereedschap

Je leert iets het beste door het te doen. Er is genoeg cryptografische software beschikbaar om mee te spelen, en er zo meer vertrouwd mee te raken. Een kleine selectie:

Cryptool Cryptool (`https://www.cryptool.org`) is educatieve open source software waarmee je uitgebreid kunt spelen, zowel met cryptografische algoritmen als met methoden om te kraken.

PGP en GPG PGP is Pretty Good Privacy, één van de eerste softwarepakketten die probeerde om sterke cryptografie beschikbaar te maken voor een groot publiek. Zie `https://www.openpgp.org/`. Het is gecommercialiseerd, maar er zijn nog altijd gratis versies, waarvan GPG – Gnu Privacy Guard de belangrijkste. Zie `https://www.gnupg.org/`.

VeraCrypt `https://www.veracrypt.fr/`, een opvolger van TrueCrypt. Gratis open source software waarmee je (delen van) je harde schijf kunt versleutelen. Onmisbaar voor laptops. Heeft als aparte functie de mogelijkheid om te verbergen dat je versleuteling hebt gebruikt ("plausible deniability").

Hash-tools Kleine programmaatjes als md5sum, sha1sum, sha256sum helpen je om hashes van bestanden te berekenen. Onmisbaar voor de nerd.

MCR, MCRE MCR – Modulaire Cryptografische Rekenmachine, een educatief java-programma dat ik ontwikkeld heb om de wiskunde van asymmetrische cryptografie zelf te kunnen doen; eigenlijk niet meer dan een rekenmachine, maar dan wel een die kan modulorekenen, en met hele grote getallen. De variant MCRE kan optellen op bepaalde Elliptische Krommen. Niet heel erg gebruikersvriendelijk. Zie `https://www.win.tue.nl/~bdeweger/MCR/`.

4.7 De toekomst van de cryptografie

In de cryptografie staan er de komende, pakweg, 10 jaar twee belangrijke dingen te gebeuren.

4.7.1 Geavanceerde cryptografische technieken

Als eerste de komst van meer geavanceerde cryptografische technieken. In deze cursus, in Hoofdstuk 5, zal Greg Alpár daar een mooi inkijkje in geven: hoe kun je cryptografie inzetten om niet meteen al je privacy te hoeven inleveren als je een bepaalde dienst wilt afnemen. Andere nieuwigheden worden wellicht toepassingen als veilige meer-partijen-berekeningen. Denk

aan het probleem van stemmen over het Internet. Aan de ene kant wil je je stem geheim houden: die moet je dus versleuteld versturen. En je moet je authenticeren omdat niet iedereen mag stemmen. Aan de andere kant wil je aan de stembureau-kant niet dat de combinatie van identificerende digitale handtekening en ontsleutelde stem gemaakt kan worden. Met nieuwe technieken kun je stemmen optellen zonder ze te hoeven ontsleutelen: "rekenen onder encryptie". Alleen het totaalresultaat van de stemming hoeft dan maar ontsleuteld te worden. Dergelijke fantastische mogelijkheden bestaan allang in theorie; de uitdaging is om ze rijp te maken voor breed gebruik.

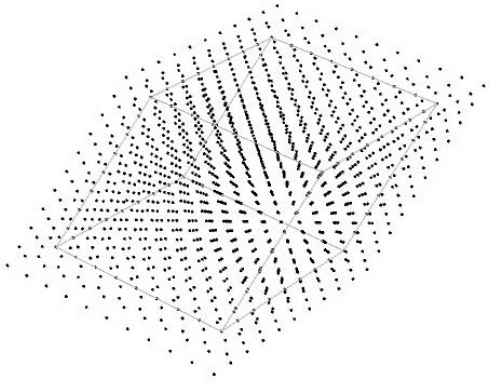
4.7.2 Quantum, en wat dan?

De tweede belangrijke ontwikkeling is de voorziene komst van quantumtechnologie. Aan de ene kant is dat quantum-computers. Als die apparaten eenmaal echt wat kunnen (de optimisten zeggen: over 10 jaar, de pessimisten zeggen: nooit) dan hebben we met z'n allen een gigantisch probleem, want het is nu al bekend dat je met de principes van quantum-computing alle nu veel gebruikte asymmetrische cryptografie de prullenbak in kunt gooien. Peter Shor heeft in 1994 al bedacht hoe een quantum-computer getallen in factoren kan ontbinden in polynomiale tijd, en net zo voor de diverse varianten van Discrete Logaritmen. Niet alleen moet je dan ogenblikkelijk stoppen met het gebruik van RSA, Diffie-Hellman, ElGamal, DSA, en de op Elliptische Krommen gebaseerde varianten, maar ook alle opgeslagen versleutelde gegevens zijn dan ogenblikkelijk kwetsbaar geworden. Voor symmetrische cryptografie is de situatie minder erg: er is een algoritme van Grover dat een k bits brute-kracht-aanval versnelt van 2^k naar $2^{k/2}$, maar daar hoef je alleen maar alle sleutels tweemaal zo lang te maken: 128 bit AES is niet meer quantum- veilig, maar 256 bit AES nog wel.

Wat moeten we dan? Als je deze vraag stelt aan de quantum-natuurkundigen dan roepen die allemaal in koor: geen nood, de quantumtechnologie biedt ook de oplossing, in de vorm van quantum-cryptografie. Je kunt namelijk ook quantumtechnologie gebruiken om over een onveilig kanaal symmetrische sleutels uit te wisselen, de zogenaamde Quantum Key Distribution (QKD). Vanwege de quantumprincipes kun je namelijk niet ongemerkt afluisteren: observatie verandert de data, en dat merken Alice en Bob dus vanzelf. Je leest regelmatig juichverhalen in de krant over de komst van onkraakbare cryptografie gebaseerd op quantumtechnologie. Wat deze quantumjongens er vervolgens nooit bijvertellen is dat deze

technologie alleen werkt als er een directe fysieke lijn is tussen te communicatiepartners (en dat is toch een beetje lastig op het Internet), en dat authenticatie van het kanaal ook nog nodig is, en daar heeft quantumtechnologie helaas geen oplossing voor. Mooi hoor, die Quantum Key Distribution. Ik juich het van harte toe[6]. Maar het lost je probleem alleen op als je er nog voldoende "klassieke" asymmetrische crypto aan toevoegt, die daarmee dus zeker niet overbodig is geworden.

Nogmaals dus: wat moeten we dan? We hebben asymmetrische cryptografie nodig die quantum-bestendig is: waar de algoritmen van Shor en Grover geen vat op hebben. Sommigen noemen dat "Post-Quantum-Cryptografie": cryptografie van een soort die de komende "cryptapocalyps" doorstaat. Daar wordt op dit moment hard aan gewerkt. Een van de meestbelovende ideeën is op roosters gebaseerde cryptografie, waar ook in Nederland op verschillende plekken aan gewerkt wordt (waaronder het CWI en de TU/e). Zie Figuur 4.4 voor een plaatje van zo'n rooster, maar denk dan aan dimensie 400 in plaats van dimensie 3. In die roosterwereld zijn nieuwe moeilijke problemen geformuleerd, waarop cryptosystemen bedacht kunnen worden voor versleuteling en digitale handtekeningen. De laatste twee jaar beginnen de eerste systemen min of meer praktisch te worden, en de industrie raakt er nu echt in geïnteresseerd. Je gaat daar de komende 10 jaar vast meer van horen.



Figuur 4.4: Een rooster – de toekomst van de cryptografie?

[6] Ik meen het!