Security: Cryptografie

D. Leeuw

10 februari 2022
v.0.1.0

Over dit Document

Dit boek behandelt Cryptografie.

Versienummering

Het versienummer van elk document bestaat uit drie nummers gescheiden door een punt. Het eerste nummer is het major-versie nummer, het tweede nummer het minor-versienummer en de laatste is de nummering voor bug-fixes.

Om met de laatste te beginnen als er in het document slechts verbeteringen zijn aangebracht die te maken hebben met type-fouten, websites die niet meer beschikbaar zijn, of kleine foutjes in de opdrachten dan zal dit nummer opgehoogd worden. Als docent of student hoef je je boek niet te vervangen. Het is wel handig om de wijzigingen bij te houden.

Als er flink is geschreven aan het document dan zal het minor-nummer opgehoogd worden, dit betekent dat er bijvoorbeeld plaatjes zijn vervangen, geplaatst of weggehaald, maar ook dat paragrafen zijn herschreven, verwijderd of toegevoegd, zonder dat de daadwerkelijk context is veranderd. Een nieuw cohort wordt aangeraden om met deze nieuwe versie te beginnen, bestaande cohorten kunnen doorwerken met het boek dat ze al hebben.

Als het major-nummer wijzigt dan betekent dat dat de inhoud van het boek substantieel is gewijzigd om bijvoorbeeld te voldoen aan een nieuw kwalificatiedossier voor het onderwijs. Een nieuw major-nummer betekent bijna altijd voor het onderwijs dat men in het nieuwe schooljaar met deze nieuwe versie aan de slag zou moeten gaan. Voorgaande versies van het document zullen nog tot het einde een schooljaar onderhouden worden, maar daarna niet meer.

Document ontwikkeling

Het doel is door middel van open documentatie een document aan te bieden aan zowel studenten als docenten, zonder dat hier hoge kosten aan verbonden zijn en met de gedachte dat we samen meer weten dan alleen. Door samen te werken kunnen we meer bereiken.

Bijdragen aan dit document worden dan ook met alle liefde ontvangen. Let u er wel op dat materiaal dat u bijdraagt onder de CC BY-NC-SA licentie vrijgegeven mag worden, dus alleen origineel materiaal of materiaal dat al vrijgegeven is onder deze licentie.

Versie	Auteurs	Wijzigingen
0.7.0	Dennis Leeuw	Uitwerking hoofdstuk Authenticiteit, plus wat updates en fixes
0.3.0	Dennis Leeuw	Toevoeging frequentie analyse, hoofdstuk over kraken is sectie geworden
0.2.0	Dennis Leeuw	Meer over Streaming en Block Ciphers, stukje over Enigma machine
0.1.0	Dennis Leeuw	Eerste release

Tabel 1: Document wijzigingen

Inhoudsopgave

Over dit Document i

1 Introductie 1
1.1 Alice en Bob . . . 2

2 Integriteit 3
2.1 Hashing . . . 3

3 Vertrouwelijkheid 5
3.1 Substitutie . . . 5
3.1.1 Caesarcijfer . . . 5
3.1.2 codebook . . . 6
3.1.3 Rozenkruizers . . . 7
3.2 Kolomtranspositie . . . 7
3.2.1 Matrix . . . 7
3.2.2 Matrix met passphrase . . . 8
3.3 Het kraken van de cijfertekst . . . 9
3.3.1 Frequentie-analyse . . . 9
3.3.2 Enigma machine . . . 9
3.4 Onkraakbare cijfertekst . . . 10
3.4.1 One-time pad . . . 10

4 Digitale cryptografie 13
4.1 Symmetrisch . . . 13
4.1.1 Streaming cipher . . . 13
4.1.2 Block cipher . . . 14
4.2 Asymmetrisch . . . 14
4.2.1 Public en Private Key . . . 15
4.3 Key Exchange . . . 16
4.3.1 Diffie-Hellman . . . 16
4.4 Hoe veilig is een algoritme? . . . 18

5 Authenticiteit **19**
5.1 PKI 19
5.1.1 Certificate Authority 20
5.1.2 Web of Trust 22
5.2 Signing 22

Appendices **25**

A Modulo rekenen **27**

B De XOR **29**

Index **31**

Hoofdstuk 1

Introductie

Cryptografies betekent het schrijven (grafie) in geheim (crypto) schrift. De bedoeling is dat een tekst omgezet wordt in iets dat niet meer leesbaar is. Van klare tekst (leesbaar) naar een cijfertekst (onleesbaar). De omzetting van klare tekst naar cijfertekst heet encryptie. Je hebt natuurlijk niets aan een cijfertekst als je die niet weer leesbaar kan maken. Het omzetten van cijfertekst naar klare tekst heet decryptie.

Er zijn verschillende methodes om klare tekst om te zetten naar cijfertekst. Vroeger werden de omzettingen met pen en papier gedaan de methodes die we daarvoor kennen noemen we daarom handcijfers, de moderne technieken gebruiken computers en wiskundige methodes en heten algoritmes.

Binnen de wetenschap is de cryptografie een onderdeel van de wiskunde. In de cryptografie houdt men zich bezich met het beveiligen van informatie door het om te zetten in een vorm die alleen door de beoogde ontvanger(s) weer verwerkt kan worden tot bruikbare informatie. Het het kraken van bestaande encryptie-methoden behoort ook tot de cryptografie.

Om data veilig met elkaar te delen zijn er een aantal zaken belangrijk:

Integer Een derde persoon kan de berichten niet veranderen

Vertrouwelijk Een derde persoon kan de berichten niet lezen (encryptie)

Authentiek Een derde persoon kan zich niet voordoen als de zender of ontvanger

1.1 Alice en Bob

In de security kom je vaak de karakters Alice, Bob, Eve en Malory tegen. Deze personen verbeelden de abstracte gedachte van data die loopt van A naar B met iemand in het Midden. Malory of Eve is de slechterick. Eve is generiek de Evil (slechte) persoon en Malory is The (Wo)Man in the Middle. Malory is afkomstig van het woord 'malice' dat kwaadaardigheid betekent. Alice en Bob betekenen niets meer dan van A naar B en hadden even goed andere namen kunnen hebben.

De karakters Alice en Bob zijn bedacht door Ron Rivest, Adi Shamir Leonard Adleman in hun artikel uit 1978 genaamd "A Method for Obtaining Digital Signatures and Public-key Cryptosystems"(https://dl.acm.org/doi/10.1145/359340.359342. Sinds het verschijnen van dit artikel zijn er vele karakters bijgekomen. Een overzicht van de verschillende karakters kan je vinden op https://en.wikipedia.org/wiki/Alice_and_Bob.

Hoofdstuk 2

Integriteit

Integriteit of in het Engels integrity gaat erover dat we kunnen nagaan of iets origineel of oorspronkelijk is en er niet mee gerommeld is. In een paspoort zitten bijvoorbeeld heel veel grafische elementen verwerkt wat het namaken moeilijk maakt. Het zelfde geldt voor papiergeld. Er wordt bij geld bijvoorbeeld gebruik gemaakt van speciaal papier, inkt en watermerken. In je paspoort zit zelfs een chip verwerkt.

Als we niet zeker weten of een document, kunstwerk of een stuk software origineel is dan kunnen we ook niets over de waarde zeggen. Het is dus heel belangrijk dat we integriteit controleren. In de digitale techniek gebruiken we hashes om te controleren of onze digitale informatie oorsponkelijk is.

2.1 Hashing

Een van de eerste dingen die we willen weten als we een bericht naar iemand sturen is of dit correct is aangekomen en met correct bedoelen we dat er onderweg geen wijzigingen hebben plaats gevonden. Het kan natuurlijk gebeuren dat door een slechte verbinding een 1 in een 0 veranderd, maar het kan ook zo zijn dat ergens onderweg iemand stiekem onze dat heeft gewijzigd. Het kan bijvoorbeeld zo zijn dat als er software van Internet gedownload wordt dat aan deze software een virus hangt die tegelijk met de software geïnstalleerd wordt. Om zeker te zijn dat we de software downloaden zoals deze op de website is gezet kan een leverancier ervoor kiezen om een hash over de software te berekenen en deze uitkomst openbaar op de website te zetten. Berekenen je over de gedownloade software zelf ook de hash dan moeten deze twee gelijk zijn. Is dat het geval dan weet je zeker dat er niet met de software gerommeld is. De

kunst van goede hashing software is dat er bij verschillende input nooit twee dezelfde antwoorden uit mogen komen.

Een simpele vorm van hashing zou kunnen zijn dat de hashing-software alle 1'en en alle 0'en telt en deze twee getallen achter elkaar zet. Het resultaat maakt de leverancier dan bekent. Als jij hetzelfde doet dan kan je de twee uitkomsten vergelijken. Dit is natuurlijk wel heel simpel en er is makkelijk mee te sjoemelen, toch is het principe van hashing-software hetzelfde.

MD5, ofwel Message-Digest 5, is bedacht door Ronald Rivest in 1991 en kent een 128 bits hash-waarde. Dit is lange tijd een veel gebruikt hashing-algoritme geweest tot ontdekt werd dat het helemaal niet zo moeilijk was om bij verschillende input een gelijke hash-waarde te creëren.

SHA, ofwel Secure Hashing Algorithems, is een collectie van algoritmes. Het oorspronkelijke algoritme (nu bekend on de naam SHA-0 of SHA) heeft maar heel kort bestaan omdat er een fout in zat. De eerste versie die wel veel gebruikt is (tot 2010) heeft een lengte van 160-bits en staat bekend als SHA-1 (of ook hier gewoon SHA). SHA-2 bestaat uit twee verschillende functies, één met 256-bits en één met 512-bits hash-waardes. Daarnaast is er ook nog SHA-3. Voor meer inhoudelijke informatie over SHA verwijzen we graag naar de Wikipedia-pagina https://en.wikipedia.org/wiki/Secure_Hash_Algorithms

Op Windows 10 is er een tool genaamd certutil.exe die gebruikt kan worden om hashing-waarden te berekenen. Deze tool kan alleen vanaf de commandline gebruikt worden door de administrator.

```
C:\> certutil -hashfile <PATH_TO_FILE> <HASH_ALGORITHM>
```

Het HASH_ALGORITHM kan MD5 zijn, maar bijvoorbeeld ook SHA256.

Linux en Mac OS X hebben afzonderlijk tools voor MD5 en SHA.

```
$ md5sum <PATH_TO_FILE>
$ sha256sum <PATH_TO_FILE>
$ sha512sum <PATH_TO_FILE>
```

Binnen de cryptografie is het kunnen berekenen van een unique hash-waarde over een bestand of bericht essentieel.

Hoofdstuk 3

Vertrouwelijkheid

Vertrouwelijkheid of in het Engels Confidentiality is ervoor zorgen dat bepaalde informatie alleen gelezen wordt door bepaalde mensen. Dit kan bijvoorbeeld door op documenten een stempel te zetten met confidential zodat we weten dat niet iedereen die mag lezen. Je kan natuurlijk ook de documenten opbergen in een kast waarvan maar enkele personen een sleutel hebben of je bergt ze op in een kluis. In de digitale techniek gebruiken we meestal een stuk software dat informatie versleuteld zodat het onleesbaar wordt en noemen we het cryptografie.

3.1 Substitutie

Substitutie is een ander woord (synoniem) voor vervanging. In de cyptografie betekent dat dat je een letter of cijfer vervangt door iets anders. Dat anders kan een andere letter zijn of een cijfer inplaats van een letter, maar ook bijvoorbeeld een symbool. Hoewel nooit bedoeld als geheimschrift is de Morse code ook een vorm van substitutie. Elke letter is vervangen door een korter en/of langer signaal om een boodschap over te brengen.

Meer informatie over de Morse code: https://en.wikipedia.org/wiki/Morse_code

3.1.1 Caesarcijfer

Eén van de oudste vorm van substitutie cryptografie die we nog kennen is de Caesarcijfer (Caesarrotatie), gebruikt door Julius Caesar (100 v. Chr - 44 v. Chr.) om te voorkomen dat zijn bevelen in handen vielen van vijanden als zijn boodschappers onderweg overvallen en vermoord werden. De substitutie cryptografie die Caesar gebruikte bestond uit het verschuiven

van de letters in het alfabet. Een verschuiving van 1 betekende dat alle letters A een B worden, alle letters B een C, etc. Een verschuiving van 4 kan er dan zo uit zien:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o

t	u	v	w	x	y	z
p	q	r	s	t	u	v

Als we in ons bericht in klaretekst de letter opzoeken in de bovenste regel en dan de letter uit de regel eronder noteren op een nieuw papiertje dan kunnen we de geencrypte tekst versturen zonder dat iemand meteen kan lezen wat er staat. Een voorbeeld:

```
Beste Bob, we gebruiken vandaag de Caesarrotatie, Groet Alice.
xaopa xkx, sa caxnqegaj rwjzwwc za ywaownnkfwpea, cnkap wheya.
```

Een andere naam voor de vorm van encryptie is ROT van rotatie. De bovenstaande encryptie heet dan ROT4 omdat de letters 4 posities zijn opgeschoven. Een bijzonder vorm is de ROT13:

a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z

Het nadeel van de Caesarrotatie is dat er maar 25 verschillende mogelijkheden zijn die simpel uit te proberen zijn, maar in Caesar zijn tijd waarin veel mensen niet lezen konden wat dit minder een probleem. Het wat de elite die kon lezen en dan moesten ze ook nog instaat zijn om te begrijpen dat het om cryptografie ging om het bericht te ontsleutelen.

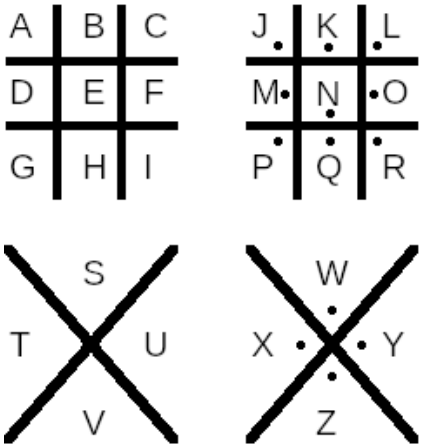
Meer informatie: https://nl.wikipedia.org/wiki/Caesarcijfer

3.1.2 codebook

In de codeboek versleuteling gaat het om de omzetting van hele stukken tekst door een andere. Er worden dan zinsdelen omgezet door gebruik te maken van dezelfde omzettingstabel. Stel dat een generaal een bericht over wil brengen over hoe troepen zich moeten verplaatsen. Bijvoorbeeld: "Infanterie devisie 304 naar het noorden."Als we afspreken de ïnfanterie divisie 304äangeduid wordt als "de olifantën het noorden als "de oever", dan zou de tekst kunnen worden "De olifant naar de oever". Voor de vijand zegt dit niets als ze niet over dezelfde codebook beschikken. De hoeveelheid combinaties is theoretisch eindeloos, maar het is nog steeds zo dat de substitutie vast is, dus altijd zal ïnfanterie divisie 304"vertaald worden met "de olifant", dus als je maar genoeg berichten onderschept en voldoende troepen bewegingen waarneemt dan zal je uiteindelijk ontdekken welk zinsdeel wat betekent.

3.1.3 Rozenkruizers

Een andere vorm van substitutie is het vervangen van letters en cijfers door symbolen. Eén van de bekende vormen is die van de Rozenkruisers. de oudste variant is afkomstig uit 1533 en is beschreven in de öcculta philosophia"van Agrippa von Nettesheim. Agrippa was een geleerd man uit Duitsland, geboren in Keulen in 1486 en gestorven in Grenoble in 1535. Zijn code kan beschreven worden zoals weergegeven in figuur 3.1.



Figuur 3.1: Rosenkruiserscode

Het deel waar de letter in staat is het symbool waardoor de letter vervangen wordt. Een a wordt dus een ⌟ en een W wordt dan ⩛. Zo kunnen we ook zinnen maken, probeer maar eens uit te vinden wat hier staat:
⌟⌜⟩ ⌜⩛ □□⊡ ⩓⌜⊡.
Hoewel minder makkelijk leesbaar is de decodering ook hier weer simpel omdat voor elke letter het symbool uniek is, dus er zijn maar 26 verschillende symbolen. Met voldoende tekst is deze versleuteling eenvoudig te kraken.

3.2 Kolomtranspositie

3.2.1 Matrix

De meest eenvoudige manier om een bericht te versleutelen is het gebruik van een matrix. In de tabel 3.1 hebben we de zin 'Deze zin wordt versleuteld' uitgeschreven in een matrix van 5x5. Door de tekst op te schrijven van boven naar beneden krijgen we de cijfertekst: didse entll zwved eoeu0

D	e	z	e	z
i	n	w	o	r
d	t	v	e	r
s	l	e	u	t
e	l	d	0	0

Tabel 3.1: Bericht in een matrix

zrrt0. Degene die deze geheime boodschap onderschept moet het formaat van de matrix weten om uit te vinden wat het bericht is. In het voorbeeld hebben we het geencrypte bericht uitgeschreven in blokken van 5, dat is natuurlijk niet zo handig en ook de twee 0'en verraden al veel, maar als we de 0'en vervangen door willekeurige letters en er 1 lange reeks letters van maken dan wordt het al weer een stuk moeilijker om de matrix te raden.

3.2.2 Matrix met passphrase

Als we dezelfde zin versleutelen met een passphrase wordt de encodering nog een stukje complexer. Een passphrase is hetzelfde als een wachtwoord, maar hoeft deze niet perse een woord te zijn, maar kan het ook een hele zin zijn. In de matrix 3.2 gebruiken we 'bosje' als passphrase. We schrijven eerst de passphrase in de tabel, deze bepaalt daardoor hoe breed onze tabel wordt. Onder de passphrase zetten we getallen en die verwijzen welke letter het eerst in het alfabet voorkomt. In ons voorbeeld is de 'b' letter die het meest vooraan in het alfabet staat, dus die krijgt waarde **1** de 'e' is de erop volgende letter en die krijgt dus waarde **2** en zo voort. De dikgedrukte nummering bepaalt nu de volgorde waarin we de kolom uitlezen. De cijfertekst wordt dan: didse zrrtg eoeuf entll zwved. De pas-

b	o	s	j	e
1	**4**	**5**	**3**	**2**
d	e	z	e	z
i	n	w	o	r
d	t	v	e	r
s	l	e	u	t
e	l	d	f	g

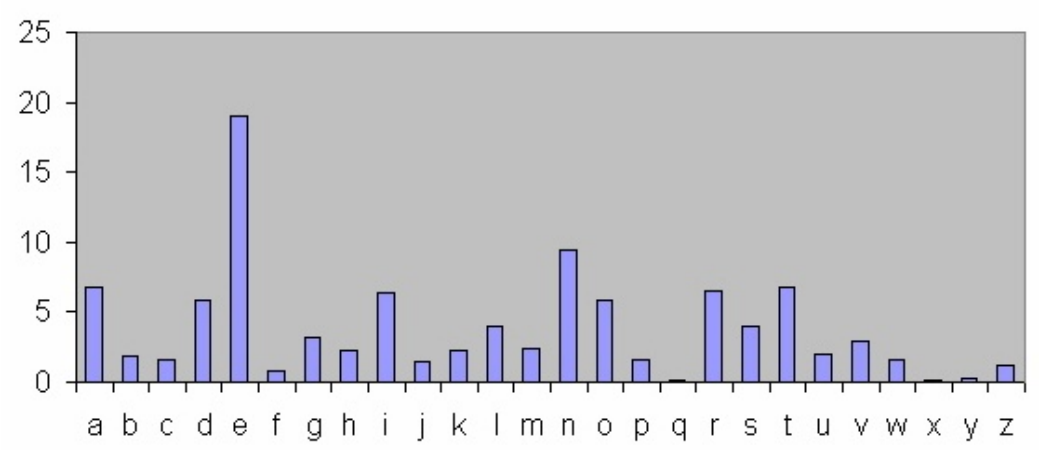
Tabel 3.2: Bericht in een matrix

sphrase bepaalt op deze manier hoe de kolommen uitgelezen worden.

3.3 Het kraken van de cijfertekst

3.3.1 Frequentie-analyse

Veel van de handmatige cijfers zijn te kraken door gebruik te maken frequentie analyse. Frequentie analyse is een techniek waarbij er gekeken wordt welke letters er in een taal in welke aantallen voorkomen. Door ervan uit te gaan dat elke encrypte tekst een gemiddelde verdeling heeft kunnen op basis van het voorkomen van bepaalde letters met een bepaalde frequentie de oorspronkelijke letters ingevuld worden. In figuur 3.2 staat de frequentie analyse van de Nederlandse taal. Hierbij is duidelijk dat de e de meest voorkomende letter is. Vinden we nu een encrypte tekst waarin p de meest voorkomende letter is dan kunnen we de p vervangen door de e. Als we zo alle letters af gaan dan kunnen we vele letters plaatsen en het bericht weer lezen.



Figuur 3.2: Frequentie Analyse van de Nederlandse taal (CC SA: Natastia at Dutch Wikipedia)

Met computers en digitale woordenboeken kunnen dit soort analyses heel snel gedaan worden. Het is natuurlijk wel belangrijk om te weten in wat voor taal het oorspronkelijke bericht was opgesteld.

3.3.2 Enigma machine

De Engima machine is populair geworden door zijn gebruikt door de Nazi's en het feit dat de geallieerden de code konden breken en de berichten mee konden lezen. Toch is de techniek al ouder dan de tweede wereldoorlog.

De Enigma gebruikt een rotor mechanisme om de 26 letter van het alfabet door elkaar te gooien. Bij het intoetsten van een letter op het toetsenbord lichte een letter op op het scherm boven het toetsenbord. De oplichtende letter was de encrypte variant. Dus voor elke toets aanslag moest de oplichtende letter opgeschreven worden, wat vaak door een tweede persoon gedaan werd. Als de encrypte tekst werd ingetoetst dan verscheen op het scherm de bijbehorende gedecodeerde letter, zo kon het oorspronkelijke bericht weer zichtbaar gemaakt worden. Het rotor mechanisme veranderde de verbindingen tussen de toetsten en de lampjes na elke toetsaanslag, dus elke letter werd unique gecodeerd.

Om te zorgen dat de machine niet elke keer alles op dezelfde manier codeert moet hij (dagelijks) worden voorzien van een nieuwe key, die beide partijen moeten kennen, dus een vorm van de one-time-pad. De lijsten met codes moeten dus van te voren bij de verzendende en ontvangende partij bekend zijn en beide partijen moeten op hetzelde moment van key wijzigen.

Het oorspronkelijke ontwerp werd al in 1932 gekraakt door een Poolse wiskundige: Marian Rejewski. Ook de verbeteringen die de Duitsers aanbrachten gedurende de oorlog werden steeds opnieuw gebroken, zodat de Polen en later de geallieerden met Poolse nagemaakte Enigma's de meeste berichten van de Duisters konden blijven volgen.

3.4 Onkraakbare cijfertekst

3.4.1 One-time pad

De onbreekbare code bestaat en die heet: One-time pad (OTP). De one-time pad maakt gebruik van modulo rekenen (zie Appendix A).

Er zijn een aantal eisen waaraan de versleuteling met een one-time pad moet voldoen:

1. De sleutel moet volledig geheim gehouden worden en mag alleen bekend zijn aan de versleutelaar en aan degene die het bericht moet ontcijferen

2. De sleutel mag niet in zijn geheel of gedeeltelijk herbruikt worden (vandaar one-time)

3. De sleutel moet net zo lang of langer zijn dan het bericht dat versleuteld wordt

4. De sleutel moet bestaan uit volledig willekeurige (random) waarden. Er mag dus geen algoritme worden gebruikt, maar er moet zoiets gebruikt worden als een hardware random number generator

In tabel 3.3 staan 2 rijen volledig willekeurige cijfers (eis 4). Ze zijn opgedeeld in blokken van 10 cijfers en er zijn 5 blokken per rij. Dat betekent dat we 100 willekeurige cijfers hebben. Als we een bericht schrijven in alleen hoofdletters en de letters omzetten via de ASCII tabel `https://en.wikipedia.org/wiki/ASCII` dan hebben we voor de letters A tot Z de decimale getallen 65 tot 90 nodig. We gebruiken dus 2 cijfers per letter. Met een maximale sleutellengte van 100 cijfers kunnen we dus maximaal een bericht van 50 characters verstuuren om te blijven voldoen aan de eis 3.

Serial: 5837424
7867562615 8641646198 7616531254 5327516761 2563621687
5674816856 9887216521 9563201567 3618568716 5768716568

Tabel 3.3: One-time pad

Als een spion een bericht wil versturen dan stelt hij eerst zijn bericht op van maximaal 50 tekens (zie regel 1 uit tabel 3.4), per letter zet hij de waarde van de ASCII tabel er onder (zie regel 2 uit tabel 3.4) en dan net zoveel cijfers van de one-time pad als nodig is om even veel cijfers te hebben als er voor het bericht nodig zijn (zie regel 3 van tabel 3.4). Tot slot komt er nog wat reken werk. We tellen het eerste cijfer van het eerste karacter op bij het eerste cijfer van de sleutel en daarna het tweede cijfer van het eerste karakter op bij het tweede cijfer van de sleutel, daar het eerste cijfer van het tweede karakter bij het derde cijfer van de sleutel, etc. (zie regel 4 van tabel 3.4) en tot slot bereken we van de twee verkregen cijfers bij elk karakter de modulo 10 (mod10) en zetten we de uitkomst weer achter elkaar zodat we per karakter weer een getal hebben van 2 cijfers (zie regel 5 van tabel 3.4).

1	O	P	H	A	L	E	N	O	P	J	F	K
2	81	80	72	65	76	69	78	81	80	74	70	75
3	78	67	56	26	15	86	41	64	61	98	76	16
4	15 9	14 7	12 8	8 11	8 11	14 15	11 9	14 5	14 1	16 12	14 6	8 11
5	59	47	28	81	81	45	19	45	41	62	46	81

Tabel 3.4: OTP: Encrypt

De spion moet hierna het serienummer van het bericht opschrijven plus de gemaakt te cijferreeks: 583742459472881814519454162468 1 en deze code is volledig onbreekbaar. Alleen degene met een papiertje met hetzelfde serienummer en dus dezelfde sleutel kan dit bericht decoderen.

Het decoderen verloopt door het omgekeerde proces te doorlopen. We verwijderen eerst het serienummer van de reeks en zetten de overgebleven losse cijfers in een rij (regel 1 van tabel 3.5. Plaats daar onder de cijfers van de sleutel (regel 2 van tabel 3.5. Trek het sleutelgetal van het cijfergetal af, is de uitkomst negatief tel dan 10 op bij het cijfergetal en trek nogmaals het sleutel getal ervan af (regel 3 van tabel 3.5. En tot slot zoek je de gevonden waarden op in de ASCII tabel zodat op regel 4 het oorspronkelijke bericht weer verschijnt.

1	59	47	28	81	81	45	19	45	41	62	46	81
2	78	67	56	26	15	86	41	64	61	98	76	16
3	81	80	72	65	76	69	78	81	80	74	70	75
4	O	P	H	A	L	E	N	O	P	J	F	K

Tabel 3.5: OTP: Decrypt

Hoofdstuk 4

Digitale cryptografie

4.1 Symmetrisch

Als een bericht encrypt is zullen we deze ook weer willen decrypten. Als we voor het encrypten en voor het decrypten dezelfde sleutel of methode gebruiken dan noemen we dat symmetrische encryptie. Symmetrische encryptie en decryptie is snel, maar kent het nadeel dat de sleutel voor beide partijen bekent moet zijn. De vraag bij symmetrische encryptie is altijd hoe je ervoor zorgt dat alle partijen die een bericht moeten kunnen lezen de sleutel krijgen zonder dat deze zomaar over een onveilig medium (zoals het Internet) verstuurd wordt.

4.1.1 Streaming cipher

De digitale variant van de one-time pad is de streaming cipher. In een streaming cipher wordt de klare tekst gecombineerd met een keystream van volledig random data waarbij de klare tekst en de random tekst gecombineerd worden, bit voor bit. Een streaming cipher wordt gebruikt als we niet weten hoeveel data er komen gaat.

De keystream moet net zo lang of langer zijn als het te encrypten bericht en mag natuurlijk maar één keer gebruikt worden. Door deze eisen is een werkelijke digitale implementatie van de one-time pad zeer lastig en wordt deze slechts zelden gebruikt. Het probleem is vooral het veilig delen van de key.

De meeste streaming ciphers maken gebruik van een keystream die gegenereerd wordt op basis van een key (bijvoorbeel 128-bits), hierdoor is de keystream niet volledig random en spreken we van een pseudorandom keystream. Hierdoor is het bewijs dat de one-time pad onbreekbaar is niet

geldig voor deze streaming ciphers, maar wordt deze wel vaak gebruikt in praktische toepassingen.

RC4 is een van de meest bekende streaming ciphers die gebruikt wordt. RC4 wordt tegenwoordig niet meer als veilig gezien. RC4 staat voor Rivest Cipher versie 4.

RC4 wordt gebruikt door SSL (Secure Socket Layer) en WEP (Wired Equivalent Privacy)

4.1.2 Block cipher

Een Block Cipher wordt gebruikt als we weten hoeveel data we moeten versturen. Een Block Cipher gebruikt een omzetting die vergelijkbaar is met een transpositie matrix. De omzetting geschiedt per block (bijvoorbeeld 128 bits) en gebruikt voor het encrypten een symmetrische sleutel. De key is in moderne algoritmes evenlang of langer dan het te encrypten datablock.

Block ciphers gebruiken pseudorandom keys.

De bekendste Block Ciphers zijn AES (Advanced Encryption Standard), DES (Data Encryption Standard), Triple DES en Blowfish.

DES is eigenlijk een aangepaste versie van Lucifer een van de eerste Block Ciphers ontwikkeld door IBM. De Amerikaanse overheid koos het als standaard (Data Enryption Standard) in 1976. DES heeft een block grootte van 64 bits en een sleutel lengte van 56 bits. De kleine sleutel lengte is de zwakte van het systeem en DES wordt dan ook niet meer als veilig gezien. Sinds 1998 is aangetoond dat DES gebroken kan worden.

De opvolger van DES is AES (Advanced Encryption Standard). De wedstrijd voor een nieuwe standaard werd gewonnen door Joan Daemen en Vincent Rijmen, twee Belgische cryptografie wetenschappers, die wat zij de Rijndael cipher noemden instuurden. AES heeft een block grootte van 128-bits en gebruikt sleutels van 128, 192 of 256-bits. AES gebruikt een 4x4 bytes matrix.

4.2 Asymmetrisch

Als we een bericht willen decrypten dat encrypt is en we gebruiken voor het encrypten een andere sleutel dan voor het decrypten dan noemen we de vorm asymmetrisch. Asymmetrische encryptie vraagt om bijzondere wiskundige algoritmes waardoor deze over het algemeen trager is dan symmetrische encryptie.

4.2.1 Public en Private Key

Asymmetrische encryptie is encryptie met public en private keys. De private key (sleutel) wordt zoals de naam al zegt geheim (privé) gehouden, de public key wordt openbaar gemaakt. Iemand die iets geheims te versturen heeft gebruikt de openbare private key om het bericht te versleutelen. Alleen degene met de private key kan het bericht nu ontsleutelen (decrypten). Voor twee wegverkeer is het dus noodzakelijk dat beide partijen een private key hebben en dat de andere partij de publieke key kent.

Encrypt je iets met de private key, dan kan iedereen die de publieke sleutel heeft het bericht decrypten. Dat heeft dus niets met beveiliging te maken. Het is wel een handige feature waar we later op terug komen.

Met een public/private key combinatie kan op een veilige manier een symmetrische key worden uitgewisseld.

Er zijn verschillende technieken op Internet die gebruik maken van public/private cryptografie. HTTPs (TLS), S/MIME, PGP (GPG) zijn wel de voornaamste technieken die we bijna dagelijks gebruiken om onze data veilig over het Internet te sturen. Er zijn verschillende algritmes die gebruikt kunnen worden, de veiligheid van de verschillende algoritmes hangt erg van van de toepassing waarvoor ze gebruikt worden. Het is ook van belang welke andere systemen ermee moeten kunnen werken. Voor HTTPs is het natuurlijk van belang dat zoveel mogelijk systemen het gebruikte algoritme ondersteunen, dus hier zal een andere keuze gemaakt worden dan bijvoorbeeld het encrypten van documenten van het Pentagon waar security vele malen belangrijker is en de uitwisseling juist geen rol speelt.

RSA Rivest-Shamir-Adleman. De veiligheid van dit algoritme hangt van de key size af. 3072 of 4096 bits zijn nog veilig. Kleiner dan 2048 is onveilig. RSA bestaat al sinds 1977, dus er zijn veel implementaties voor veel systemen, dus dit algortime wordt door heel veel systemen ondersteund.

DSA Digital Signature Algorithm. Is onveilig en zou niet meer gebruikt moeten worden, vanaf OpenSSH 7.0 per default gedisabled.

ECDSA Elliptic Curves DSA. Redelijk veilig, een beetje afhankelijk van de random number generator die gebruikt wordt. Minimaal 256 bits key size.

ed25519 Op dit moment (2021) de meest veilige optie, maar ook de nieuwste dus niet overal al aanwezig.

Voor het maken van een public/private key pair zijn dus een aantal zaken van belang:

- Key algoritm
- Key size
- Passphrase - Er kan op de private key aan passphrase gezet worden, maar dit hoeft niet.

Het maken van een public/private key paar is simpel. Op Linux en Mac OS X systemen kan je ssh-keygen gebruiken, op Windows PuTTY-gen, of ook met ssh-keygen (command line) als je OpenSSH voor Windows hebt geïnstalleerd.

4.3 Key Exchange

Encryptie en decryptie met symmetrische algoritmes is vele malen sneller dan met asymmetrische methodes. We willen in de digitale cryptografie dan ook ze snel mogelijk gebruik maken van een symmetrische sleutel, de vraag daarbij is hoe we zorgen dat een sleutel veilig wordt uitgewisseld tussen twee systemen. Deze sleutel uitwisseling noemen we in het Engels de key exchange.

Er zijn een aantal methodes om symmetrische sleutels met elkaar uit te wisselen:

Key Encapsulation Mechanism (KEM) Door gebruik te maken van asymmetrische encryptie om een veilige verbinding op te zetten en dan daarna een symmetrische sleutel uit te wisselen.

Key Exchange (KEX) En algoritme om keys te berekenen zonder dat de te gebruiken symmetrische sleutel over de lijn gaat, zoals Diffie-Hellman (DH).

Out-of-Band De symmetrische key delen via bijvoorbeeld een USB-stick, papier, of een andere methode.

4.3.1 Diffie-Hellman

In 1975 waren er twee wiskundigen, Whitfield Diffie en Martin Hellman, die een oplossing bedachten voor het probleem van het uitwisselen van

symmetrische sleutels (symmetric keys). Ze kwam met een elegante wiskundige oplossing die het versturen van de sleutel over het Internet overbodig maakt. De sleutel wordt berekend door gebruik te maken van modulo rekenen (zie appendix A) en priemgetallen (getallen die alleen door 1 en zichzelf deelbaar zijn). De techniek die gebruikt wordt is naar hun vernoemd: Diffie-Hellman, afgekort DH.

Tabel 4.1 beschrijft welke acties Alice en Bob moeten nemen om tot een gezamenlijke sleutel te komen. Terwijl Alice en Bob werken om tot een gezamenlijke sleutel te komen is Eve aanwezig die al het verkeerd tussen Alice en Bob afluisterd, want ook zij wil graag die geheime sleutel hebben.

Actie	Alice		Eve		Bob
	Alice en Bob spreken twee getallen af: p en q. De getallen zijn niet geheim en mogen over het Internet verstuurd worden. De eisen zijn dat beide getallen priemgetallen moeten zijn en dat q kleiner moet zijn dan p. Eve heeft nu p en q				
1	$p = 11$	->	p, q	<-	$q = 7$
	Alice en Bob verzinnen ieder een eigen getal dat ze geheim houden en niet met elkaar delen.				
2	$a = 3$				$b = 6$
	Alice en Bob gebruiken modulo rekenen om A en B te berekenen.				
3	$A = q^a \% p$ $A = 7^3 \% 11$ $A = 2$				$B = q^b \% p$ $B = 7^6 \% 11$ $B = 4$
	Alice en Bob sturen hun berekende getallen, A en B, naar elkaar. Eve heeft nu ook A en B				
4	$A = 2$	->	A, B	<-	$B = 4$
	Alice en Bob gebruiken modulo rekenen om S te berekenen				
5	$S = B^a \% p$ $S = 4^3 \% 11$ $S = 9$				$S = A^b \% p$ $S = 2^6 \% 11$ $S = 9$
	Eve heeft tijdens de afspraken van Alice en Bob meegeluisterd op de lijn en heeft nu de getallen p, q en A, B verzameld. Ze mist echter a of b om ook S te kunnen berekenen. Op deze manier hebben Alice en Bob via een rekenkundige weg een gezamenlijke sleutel S afgesproken zonder dat deze over het Internet gegaan is.				

Tabel 4.1: DH uitleg

Omdat de sleutel berekend wordt via een algortime (modulo rekenen) voldoet deze niet aan de eisen van een onbreekbare code, namelijk dat de sleutel volledig random moet zijn. In de praktijk is de techniek echter zo goed, als we maar voldoende grote priemgetallen nemen, dat we van een veilige verbinding kunnen spreken.

Diffie-Hellman houdt geen rekening met de identiteit van de zender en/of ontvanger, dus een man-in-the-middle attack is nog steeds mogelijk. Het gebruik van public en private keys lost dit probleem op.

4.4 Hoe veilig is een algoritme?

De veiligheid van een algoritme wordt beschreven in het aantal pogingen dat er nodig is om de sleutel te breken. Met het steeds sneller worden van compters kunnen er steeds meer pogingen gedaan worden in dezelfde tijd. De eis van het NIST is op dit moment dat een algoritme beter moet zijn dan 128-bits, 2^128 pogingen voordat het gebroken wordt. De veronderstelling is dat we daarmee tot 2030 vooruit kunnen. Voor RSA betekent dat een minimum van 3072 bits key size.

Om te bepalen welke cipher suites je wel en niet wil toestaan is er een site die voor je bijhoudt welke suites geschikt zijn en welke je absoluut niet meer moet gebruiken omdat ze te makkelijk te kraken zijn. Deze site is te vinden op https://ciphersuite.info/cs/?singlepage=true&security=secure. Door de optie security=secure mee te geven zie je alleen de suites die een kwalificatie secure of hoger hebben.

Hoofdstuk 5

Authenticiteit

Als we communiceren over het Internet hoe weten we dan dat bijvoorbeeld de website van de bank daadwerkelijk de website van de bank is en niet een website opgezet door een boef? Of hoe weten we zeker dat de e-mail die we gekregen hebben daadwerkelijk verzonden is door degene die in het afzenderveld staat? Al deze vragen hebben te maken met authenticiteit of Authenticity in het Engels. In het niet digitale leven zou je iemand om zijn paspoort of rijbewijs kunnen vragen zodat de persoon kan aantonen wie die is. In de digitale wereld gebruiken servers certificaten om aan te tonen dat zij echt de server van de bank zijn. Een certificaat is dus te vergelijken met een digitaal paspoort.

5.1 PKI

PKI staat voor Public Key Infrastructure en is het totaal van maatregelen en instanties die nodig zijn om publieke sleutels te beheren. PKI is bedoelt om gebruikers en apparaten te authenticeren in een digitale wereld, vergelijkbaar met het paspoort in de gewone wereld. Het gaat daarbij om vertrouwen, vertrouwen in de echtheid van bijvoorbeeld een website die je bezoekt, de echtheid van een document, of het vertrouwen dat een gebruiker die gebruik wil maken van een systeem ook werkelijk die gebruiker is.

Vertrouwen tussen digitale entiteiten wordt bepaald door het vertrouwen dat we hebben in de public key van een entiteit. De private key moet altijd geheim blijven, maar de publieke sleutel mogen we met iedereen delen. De vraag bij de verkregen publieke sleutel is altijd of dit de sleutel is van de machine of persoon waarmee ik contact wil hebben of dat er ergens een man-in-the-middle zit die mij een valse sleutel heeft gegeven en dus

stiekem met ons meeleest.

Er zijn verschillende oplossing om dit vertrouwen te bewerkstelligen:

CA Certificate Authority

WoT Web of Trust

5.1.1 Certificate Authority

Voordat jij vertrouwd wordt om op Schiphol in een vliegtuig te stappen moet je eerst je paspoort laten zien. In je paspoort zit een foto en een beschrijving van een aantal lichamelijke kenmerken waardoor de douane beambte kan controleren dat jij en je paspoort bij elkaar horen. Het feit dat de douane beambte het paspoort vertrouwd heeft te maken met een aantal zaken. Allereerst zal hij of zij het controleren op echtheidskenmerken, zoals watermerken etc. Als dat echt blijkt te zijn, dan is de onderliggende aanname dat het document is uitgeven door de overheid en dat de overheid voor uitgifte een aantal controles heeft gedaan zodat de overheid weet dat het paspoort ook echt bij jou hoort. Het is dus de overheid die ervoor zorgt dat jij alleen jouw paspoort hebt en ook alleen jij het paspoort hebt. Tevens is het paspoort maar beperkt geldig.

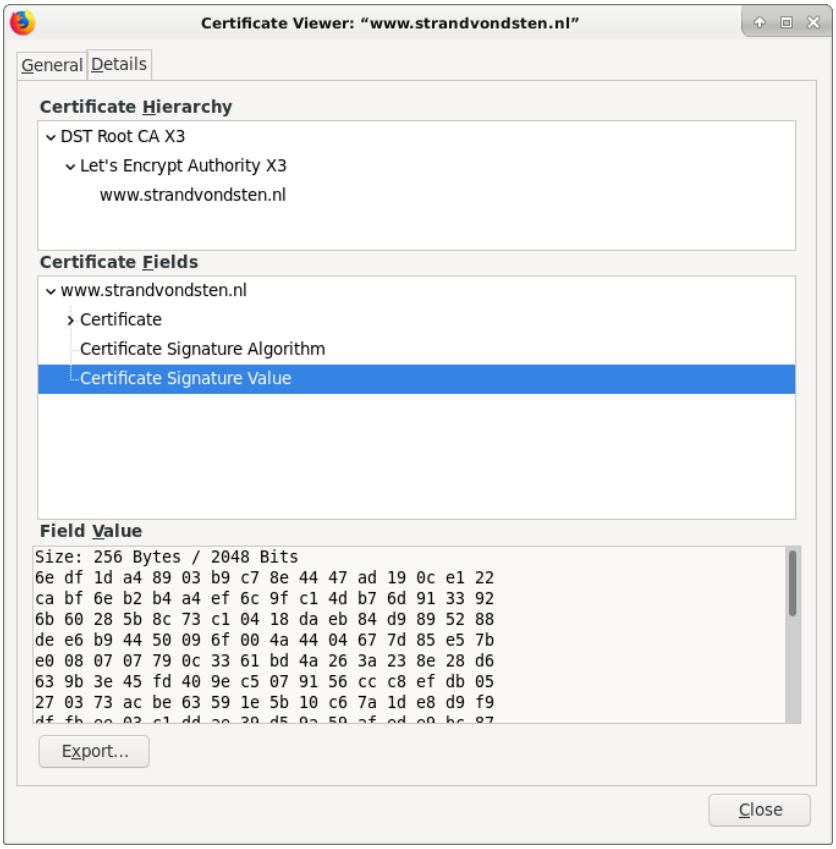
Al deze dingen koment we ook tegen bij het digitale paspoort, het certificaat, dat door de Certificate Authority (CA) wordt uitgegeven. Als je een certificaat nodig hebt, dan zul je dat moeten aanvragen. Om dit te kunnen doen moet je eerst een public en private key paar genereren. Op basis van de public key maak je een Certificate Signing Request CSR. In dit verzoek tot het tekenen van het certificaat neem je een aantal zaken op zoals bijvoorbeeld waar het certificaat voor bedoeld is. Als het voor een webserver is dan neem je de domeinnaam op in het CSR, de geldigheidsduur en de eigenaar van de website. Dit verzoek stuur je op naar een CA. De CA controleert de gegevens. Dus die kijkt na of jij echt de eigenaar bent van het domein en of jij echt bent wie je bent. Als alles klopt dan krijg je van de CA een certificaat met de gegevens uit de CSR, maar nu getekend met de handtekening van de CA. De CA ondertekend het document met zijn private key. Hiermee heb je het digitale paspoort waarmee je kan aantonen dat jouw server de echte server is die bij je domeinnaam hoort.

Een certificaat voor een webserver bevat:

- De public key van de server
- De domeinnaam waarvoor het certificaat geldig is
- De geldigheidsduur

- De handtekening (signature) van de CA

Een voorbeeld van een signature is te zien in figuur 5.1.



Figuur 5.1: Let’s Encrypt certificaat

Een webserver heeft zijn eigen private key en het certificaat geïnstalleerd staan, en eventueel nog alle tussen liggende certificaten tussen het eigen certificaat en dat van de Certificate Authority.

Zodra een webbrowser contact maakt met een server en om een beveiligde verbinding vraagt stuurt de webserver zijn certificaat op naar browser. De browser controleert het certificaat door controle van het domeinnaam waarvoor het cerificaat geldig is, de geldigheidsduur van het certificaat en hij controleert de handtekening van de CA met die welke hij kent uit zijn database. Elke browser moet dus een lijst met bekende CA’s hebben om deze controle te kunnen doen. Als het certificaat in orde wordt bevonden kan de public key uit het certificaat gebruikt worden om een encrypt bericht aan de server te sturen die dat bericht can decrypten met zijn private key.

5.1.2 Web of Trust

Het Web of Trust (WoT) is ooit bedacht door Phil Zimmermann in 1992 toen hij bezig was met PGP, Pretty Good Privacy. Dit is gebaseerd op onderling vertrouwen zoals in een vriendenkring. Als Bob Alice vertrouwt en Alice vertrouwt Carlos dan zou Bob Carlos ook kunnen vertrouwen. Er is dus geen centrale partij die bepaalt of een certificaat of een publieke sleutel te vertrouwen is en er is ook niet één web of trust, net zomin als er één vriendenkring is. Er kunnen verschillende vriendenkringen wereldwijd zijn met hun eigen collectie van publieke sleutels die ze vertrouwen.

PGP is ooit ontworpen om e-mail veilig te maken. De gedachte voor de Web of Trust is dat als je public keys met je vrienden uitwisseld je een database kan opbouwen in je e-mail client en als jij aangeeft dat je deze public keys daadwerkelijk van je vrienden hebt gekregen dan ben jij degene die kan zeggen dat ze echt zijn. Als anderen dat ook doen, dan kan je daarna fysiek of electronisch de verzamelde sleutels met elkaar uitwisselen. Daarbij zou het handig zijn als je een gradatie van vertrouwen zou kunnen aangeven. Bob vertrouwt Alice, dus die sleutels zijn volledig te vertrouwen, maar Bob kent Carlos niet, dus het vertrouwen in de sleutels van Carlos zouden een iets lagere vertrouwens waarde kunnen krijgen dan die van Alice.

Het web of trust zoals dat geïmplementeerd is in PGP en GPG (GNU Privacy Guard, de open source implementatie van de OpenPGP standaard, heeft de mogelijkheid om ook gebruik te maken van een CA. Binnen een organistatie kan een eigen CA opgezet worden die de certificaten die de public keys bevatten kan certificeren. Op deze manier kan door het vertrouwen van één certificaat alle certificaten binnen een web of trust vertrouwd worden.

5.2 Signing

Zoals al eerder aangegeven kan je ook met een private key berichten encrypten en deze de wereld in sturen, iedereen met de public key kan deze berichten decrypten, maar omdat iedereen bij de publiek kan is dit geen vorm van beveiliging, het is wel een vorm van authenticatie, Het is een vorm die we signing noemen. Een ieder die de public key heeft kan het bericht decrypten en weet op dat moment 100% zeker dat het bericht encrypt is door de degene met de private key. Het is dus een bewijs dat alleen degene die de private key heeft het bericht heeft kunnen versturen. Hiermee kan dit gebruikt worden om bijvoorbeeld contracten te onderte-

kenen.

Ook bij signing is het natuurlijk weer van belang om de publieke sleutel te kunnen vertrouwen, dus ook hier hebben we weer een certificaat nodig van een CA of een web of trust.

Appendices

Bijlage A

Modulo rekenen

Voor het rekenwerk binnen de crypografie hebben we soms modulo rekenen nodig, ofwel klok rekenen. Modulo rekenen wordt wel klok rekenen genoemd omdat dat de meest bekende vorm is die iedereen bijna dagelijks uitvoert. Als iemand tegen je zegt we spreken om 13:00 uur af dan weet iedereen dat die persoon 1 uur 's middags bedoelt. We hebben in gedachte even snel 12 van 13 afgetrokken en hielden 1 over. Modulo rekenen is dus een rekenvorm waarbij we de rest waarde berekenen op basis van een grondgetal. Dat grondgetal bij het klokrekenen is 12. We zouden wiskundig kunnen schrijven 13 mod 12 = 1. Onze klok gaat niet verder dan 24 uur en dus ik klok rekenen een heel beperkte versie van modulo rekenen.

Als we een klok zouden nemen die iets verder doorloopt dan zouden we kunnen zeggen we spreken af om 25:00 uur. 25 mod 12 is weer 1, want we kunnen 2x12 aftrekken van 25 en dan blijven we met een rest 1 achter wat dus weer 1 uur is, alleen nu wel midden in de nacht. Ook de basis kunnen we natuurlijk veranderen: 25 mod 5 is 0, want 5x5 is 25 en 25-25 = 0. We hebben dan dus geen rest getal. Of 38 mod 3 is 2, want 12x3 is 36 en 38-36 = 2.

In programmeertalen en op rekenmachines wordt modulo rekenen vaak weergegeven met een %. We schrijven dan 25%12=1, of 25%5=0.

Bijlage B

De XOR

Een veel gebruikte functie uit de boolean algebra in de cryptografie is de XOR. Het principe van de XOR is gebaseerd op het optellen van bits waarbij alleen het laatste digit wordt gebruikt:

```
0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 0
```

Daarmee komt de waarheidstabel van de XOR op de tabel zoals weergegeven in B.1

Input Input A	Input Input B	Output A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Tabel B.1: XOR

Computers kunnen via een XOR heel snel een stroom met bits encrypten. De manier waarop dat gebeurt is weergegeven in tabel B.2.

Klare tekst stream	0	1	1	0	0	1	0	1
Sleutel	1	0	1	1	0	1	1	0
Cijfertekst	1	1	0	1	0	0	1	1

Tabel B.2: XOR Encrypt

Het leuke van de XOR functie is dat je ook weer een XOR gebruikt voor de decryptie en dat zie er dan uit zoals in tabel B.3

Cijfertekst stream	1	1	0	1	0	0	1	1
Sleutel	1	0	1	1	0	1	1	0
Klare tekst	0	1	1	0	0	1	0	1

Tabel B.3: XOR decrypt

Index

Advanced Encryption Standard, 14
AES, 14
Agrippa, 7
Algoritme, 1
Alice, 2
Authenticiteit, 19
Authenticity, 19

Block Cipher, 14
Bob, 2

CA, 20
Caesarcijfer, 5
Caesarrotatie, 5
Certificate Authority, 20
Certificate Signing request, 20
certutil, 4
Cijfertekst, 1
Codebook, 6
Confidentiality, 5
Cryptografie, 1
CSR, 20

Data Encryption Standard, 14
Decryptie, 1
DES, 14
DH, 17
Diffie, Whitfield, 16
Diffie-Hellman, 17

Encryptie, 1
Enigma, 9
Eve, 2

Frequentie analyse, 9

GNU Privacy Guard, 22
GPG, 22

Handcijfer, 1
Hashing, 3
    MD5, 4
    SHA, 4
Hellman, Martin, 16

Integriteit, 3
Integrity, 3

Key exchange, 16
Klare tekst, 1
Klok rekenen, 27

Lucifer, 14

Malory, 2
MD5, 4
md5sum, 4
Message-Digest 5, 4
Modulo rekenen, 27

Onbreekbare code, 10
One-time pad, 10, 13
Oorspronkelijk, 3
OpenPGP, 22
Origineel, 3
OTP, 10

Passphrase, 8
PGP, 22
PKI, 19
Pretty Good Privacy, 22
Private key, 15
Public key, 15
Public Key Infrastructure, 19

RC4, 14
Rivest Cipher 4, 14
ROT, 6
ROT13, 6
Rozenkruisers, 7

Secure Hashing Algorithms, 4
Secure Socket Layer, 14
Security Karakters, 2
SHA, 4
sha256sum, 4
sha512sum, 4
signing, 22
Sleutel uitwisseling, 16
SSL, 14
Streaming cipher, 13
Substitutie, 5
    Codeboek, 6
    Rotatie, 5
    Symbolen, 7

Vertrouwelijkheid, 5

Web of Trust, 22
WEP, 14
Wired Equivalent Privacy, 14
WoT, 22