09. Les 9 Het Computertijdperk

Auteur	Bètapartners
Team	Wikiwijs Maken Auteurs
Laatst gewijzigd	18 december 2014
Licentie	CC Naamsvermelding-GelijkDelen 3.0 Nederland licentie
Webadres	https://maken.wikiwijs.nl/45943/

Inhoudsopgave

Les 9 Het Computertijdperk

Inhoud van les 9: Het computertijdperk

9.1 De programmeerbare codebreker

Voor de geheime communicatie tussen de Führer en zijn generaals beschikten de Duitsers over een nog ingewikkelder mechanisch cijfer dan de Enigma. Dit staat bekend als het Lorenz cijfer. De strijd tegen het Lorenz cijfer werd op Bletchley Park gewonnen door Tommy Flowers. Max Newman, een wiskundige, bedacht de Colossus, een programmeerbare codebrekende machine, gebaseerd op het concept van Alan Turing. Het werd technisch onuitvoerbaar geacht deze machine te bouwen, maar Tommy Flowers bouwde daarmee een machine, bestaande uit 1500 electronenbuizen, die veel sneller waren dan de relais in de bombes.
Over het verhaal van de Lorenz machine en de Colossus is veel te vinden op het internet. Volg bijvoorbeeld de link naar Britain's Best Kept Secret of naar Colossus.
Na de oorlog werd alles op Bletchley Park vernietigd, zo ook de bouwtekeningen van de Colossus en de Colossus zelf. Zo is in de geschiedenis de ENIAC (Electronic Numerical Integrator And Calculator) bekend geworden als de eerste, in 1945 gerealiseerde computer van J. Preper Eckert en John W. Mauchley. Deze telde 18.000 electronenbuizen en kon 5000 berekeningen per seconde uitvoeren. Het computervermogen zou gebruikt gaan worden om steeds ingewikkelder cijfersystemen te bedenken en te breken.

Het voordeel van het gebruik van een computer is, dat je Enigma's kunt simuleren met honderd scramblers die alle kanten opdraaien en van tijd tot tijd een letter overslaan of met verschillende snelheid gaan roteren, terwijl deze in werkelijkheid niet te bouwen zijn. Bovendien stelt de computer je in staat het proces te automatiseren zodat je met grote snelheid kunt encrypten en decrypten. Het belangrijkste voordeel en verschil ten opzichte van de mechanische encryptie is, dat de computer werkt met getallen. We hebben daar een voorproefje van gezien in les 2: de affiene versleuteling.

9.2 Binaire getallen

De computer werkt uitsluitend met binaire getallen: 0 en 1. Het kleinste deel dat naar het geheugen van een computer geschreven kan worden of uit het geheugen kan worden uitgelezen noemen we een *byte.*
Een byte bestaat uit 8 nullen of enen en kan bestaan uit 2^8=256 verschillende binaire getallen oplopend van 00000000 tot 11111111.
De 256 binaire getallen corresponderen met onze decimale getallen van 0 tot en met 255 op de manier zoals hiernaast voor een deel is aangegeven:

Binair	Decimaal
00000000	0
00000001	1
00000010	2
00000011	3
00000100	4
00000101	5
00000110	6
00000111	7
00001000	8

Het telsysteem werkt hetzelfde als het decimale systeem. We tellen van 0 t/m 9 met 10 symbolen en we tellen van 0 t/m 1 binair met 2 symbolen.
Met 10_{dec} bedoelen we het decimale getal 10 en geven we aan 1 tiental en 0 eenheden.
Met 10_{bin} bedoelen we het binaire getal 10 en geven we aan 1 tweetal en 0 eenheden.
En zo zetten we het voort:
Met het decimale getal 111 bedoelen we 1 hondertal, 1 tiental en 1 eenheid, dus 1x100+1x10+1x1.
Met het binaire getal 111 bedoelen we 1 viertal, 1 tweetal en 1 eenheid, dus 1x4+1x2+1x1.
111_{bin} is dus gelijk aan 7_{dec}
Merk op dat decimaal elk cijfer een aantal maal een macht van 10 voorstelt en dat evenzo binair elk cijfer een aantal maal een macht van 2 voorstelt.
Het getal 10010101 is daarom gelijk aan
$1x2^0+0x2^1+1x2^2+0x2^3+1x2^4+0x2^5+0x2^6+1x2^7=$
1 + 0 + 4 + 0 + 16 + 0 + 0 +128 = 149

Andersom kunnen we bijvoorbeeld het decimale getal 221 schrijven als 1x128 + 1x64 + 0x32 + 1x16 + 1x8 +1x4 + 0x2 + 1x1.
Een handige manier om dit te vinden is door te kijken naar de hoogste macht van 2 die nog in het getal past, deze er af te trekken en zo verder te tellen. Hieronder wordt het voorgedaan. De machten van 2 zijn 1, 2, 4, 8, 16, 32, 64 en 128.

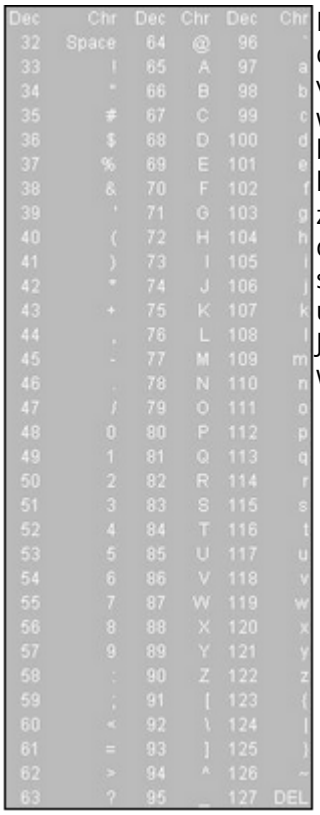
221-128 = 93 _____1
93 - 64 = 29 _____1
29 -32 kan niet _____0
29 - 16 = 13 _____1
13 - 8 = 5 _____1
5 - 4 = 1 _____1
1-2 kan niet _____0
1-1 = 0 _____1

De binaire representatie (voorstelling) van 221 is dus 11011101. We noteren 221_{dec} = 11011101_{bin}.

Opgave 1

Wat is de binaire representatie van de volgende decimale getallen?
a. 55
b. 152
c. 78

Wat is de decimale representatie van de volgende binaire getallen?
d. 01100110
e. 10101010
f. 01010101



Dec	Chr	Dec	Chr	Dec	Chr
32	Space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DEL

In de ASCII-tabel is te zien hoe elke letter correspondeert met een vast decimaal getal. Hieronder is een deel van de tabel afgebeeld. Zo wordt de*A* voorgesteld als 65 en de *a* als 97. De eerste 31 symbolen in de ASCII-tabel worden gebruikt voor machine-instructies zoals de *carriage return*, beter bekend als de enter-toets, onder nummer 15. Als je de ASCII-tabel aan een klein onderzoek onderwerpt zie je dat alle toetsen van ons toetsenbord keurig zijn ondergebracht onder de getallen 32 (de spatie) t/m 126 (de tilde ~). Als je de link volgt zul je zien dat de getallen 128 t/m 254 voor allerlei andere symbolen worden gebruikt volgens de uitgebreide ASCII-tabel. Omdat de tabel uiteindelijk te weinig ruimte biedt om bijvoorbeeld ook Chinese, Arabische en Japanse tekens onder te brengen zijn er meerdere tekensets bedacht waaronder de Unicode.

9.3 Coderen en decoderen volgens de ASCII-tabel

De encryptie met gebruik van getallen gaat op dezelfde manier als de encryptie met letters, namelijk met substitutie (vervanging) en transpositie (verplaatsing).

Bij diverse cryptosystemen is het nodig om de symbolen van de klare tekst eerst om te zetten naar getallen. De zender en ontvanger moeten op dezelfde manier tekens en getallen met elkaar in verband brengen, maar de manier waarop ze dat doen hoeft niet geheim te blijven. Sterker nog: je kunt veel beter gebruik maken van een (internationale) standaard. Dan hoef je ook niet met alle partijen waar je mee wilt communiceren afzonderlijk af te spreken welke koppeling tussen symbolen en getallen je gaat gebruiken. Eigenlijk valt dit deel van het uitwisselen van berichten dus niet onder cryptografie. We geven het daarom een andere naam: onder coderen verstaan we het omzetten van symbolen naar getallen; het terugvertalen van getallen naar symbolen noemen we decoderen. Vergelijk dit met de definitie in les 1.

De internationale standaard die we in de rest van deze lessenserie gaan gebruiken is de ASCII-codering.

Voorbeeld
De zin: "Ik wou dat ik 2 hondjes was".
wordt gecodeerd in de volgende rij gehele getallen

073 107 032 119 111 117 032 100 097 116 032 105 107 032
050 032 104 111 110 100 106 101 115 032 119 097 115 046

Opgave 2

Tel het aantal tekens in de zin uit het voorbeeld en het aantal getallen in de codering. Is dit gelijk? Zo niet, wat heb je dan bij het tellen over het hoofd gezien?

Opgave 3

Codeer de zin "Tussen Keulen en Parijs.".

Cryptosystemen die een boodschap letter voor letter versleutelen zijn kwetsbaar. We hebben al gezien dat een eenvoudige methode als frequentieanalyse zulke systemen vaak kan kraken. Dit wordt ondervangen door steeds een vast aantal getallen uit de codering aan elkaar te plakken en de versleuteling uit te voeren op deze (grote) getallen. In de praktijk zijn dat er meer, maar wij zullen steeds ons beperken tot het samennemen van twee getallen. Dat maakt aan de ene kant het kraken met behulp van frequentie-analyse al een stuk lastiger en aan de andere kant blijft het nog net hanteerbaar: een deel van de berekeningen is nog uit te voeren met behulp van de grafische rekenmachine. Als we verderop berekeningen moeten maken waar de grafische rekenmachine het niet meer aankan, dan stappen we terug naar losse letters of we zetten de computer in.

Opgave 4

Decodeer de volgende boodschap:
079112 032077 097114 115032 105115 032119 097116 101114 032103 101118 111110 100101 110033

Opgave 5

Leg uit waarom het bij deze manier van werken nodig is dat alle getallen door evenveel cijfers worden weergegeven.

9.4 Computercryptografie

Zoals hierboven al aangegeven werkt de computer alleen met nullen en enen. Om te laten zien dat het geen principieel verschil oplevert met bovenstaande werkwijze en dat alleen de getallen anders zijn, proberen we een voorbeeld te volgen in het binaire systeem.

Laten we de tekst "**morgen bij paal 22**" vertalen in binaire code.
De volgende tabel toont de binaire code per letter, spatie en cijfer:

Letter	ASCII-code	Binaire code	Letter	ASCII-code	Letter	Letter	ASCII-code	Tabel	Letter	ASCII-code	Tabel
a	97	01100001	h	104	01101000	o	111	01101111	v	118	01110110
b	98	01100010	i	105	01101001	p	112	01110000	w	119	01110111
c	99	01100011	j	106	01101010	q	113	01110001	x	120	01111000
d	100	01100100	k	107	01101011	r	114	01110010	y	121	01111001
e	101	01100101	l	108	01101100	s	115	01110011	z	122	01111010
f	102	01100110	m	109	01101101	t	116	01110100	spatie	32	00100000
g	103	01100111	n	110	01101110	u	117	01110101	2	50	00110010

Klik op het plaatje om het in een ander venster te openen.

De binaire code zal dus worden:
01101101 01101111 01110010 01100111 01100101 01101110 00100000 01100010 01101001 01101010 00100000 01110000 01100001 01100001 01101100 00100000 00110010 00110010

We kiezen nu de sleutel: **crypto** (binair wordt dat:
01100011 01110010 01111001 01110000 01110100 01101111)
en tellen het codewoord er bij op, zoals we dat ook deden bij de Vigenère-code, maar nu met een binaire optelling.

De binaire optelling bepaalt dat **0+0=0; 0+1=1; 1+0=1** en **1+1=0**

Dat geeft het volgende effect:

c	r	y	p	t	o	c	r
01100011	**01110010**	**01111001**	**01110000**	**01110100**	**01101111**	**01100011**	**01110010**
m	o	r	g	e	n		b
01101101	**01101111**	**01110010**	**01100111**	**01100101**	**01101110**	**00100000**	**01100010**
y	p	t	o	c	r	y	p
01111001	**01110000**	**01110100**	**01101111**	**01100011**	**01110010**	**01111001**	**01110000**
i	j		p	a	a	l	
01101001	**01101010**	**00100000**	**01110000**	**01100001**	**01100001**	**01101100**	**00100000**
t	o						
01110100	**01101111**						
2	2						

00110010	00110010

Vervolgens tellen we de getallen op:

01100011	01110010	01111001	01110000	01110100	01101111	01100011	01110010
01101101	01101111	01110010	01100111	01100101	01101110	00100000	01100010
00001110	00011101	00001011	00010111	00010001	00000001	01000011	00010000
01111001	01110000	01110100	01101111	01100011	01110010	01111001	01110000
01101001	01101010	00100000	01110000	01100001	01100001	01101100	00100000
00010000	00011010	01010100	00011111	00000010	00010011	00010101	01010000
01110100	01101111						
00110010	00110010						
01000110	01011101						

Voor degene die het bericht ondervangt en vertaalt staat er nu onzin:

00001110	00011101	00001011	00010111	00010001	00000001	01000011	00010000
00010000	00011010	01010100	00011111	00000010	00010011	00010101	01010000
01000110	01011101						

Terugvertaald in ASCII-code staat er **14 29 11 23 17 1 67 16 16 26 84 31 2 19 21 80 70 93**
De ontvanger telt opnieuw de sleutel bij de binaire reeks:

01100011	01110010	01111001	01110000	01110100	01101111	01100011	01110010
00001110	00011101	00001011	00010111	00010001	00000001	01000011	00010000
01101101	01101111	01110010	01100111	01100101	01101110	00100000	01100010
01111001	01110000	01110100	01101111	01100011	01110010	01111001	01110000
00010000	00011010	01010100	00011111	00000010	00010011	00010101	01010000
01101001	01101010	00100000	01110000	01100001	01100001	01101100	00100000
01110100	01101111						
01000110	01011101						
00110010	00110010						

en vindt de binaire code van de klare tekst:

01101101 01101111 01110010 01100111 01100101 01101110 00100000 01100010 01101001 01101010 00100000 01110000 01100001 01100001 01101100 00100000 00110010 00110010

 En de inbreker dan?

Als inbreker op de code krijg je volgende rij letters te zien: **14 29 11 23 17 1 67 16 16 26 84 31 2 19 21 80 70 93**
Als je hier iets mee zou willen ondernemen, welke aanpak zou je dan kiezen?

klik hier

Opgave 6

Versleutel de klare tekst **hallo** met het sleutelwoord **david**. Welke cijferreeks vind je als je de versleutelde tekst probeert terug te vertalen naar ASCII-code?

9.5 Het systeem Lucifer

Het werken op deze manier is voor mensen geen prettige klus en de bovenstaande pagina met nulletjes en eentjes geeft ook al geen aanblik waar je direct vrolijk van wordt. Het grote voordeel van bovenstaande manier van werken is echter dat het volautomatisch door de computer kan worden uitgevoerd en dan weinig moeite hoeft te kosten. Het enige wat je moet doen is een programma schrijven wat de klare tekst omzet in binaire code via de ASCII-tabel. In het vervolg zullen we ons beperken tot bewerkingen met de ASCII-code.

In 1947 werd de transistor uitgevonden, die een goedkoop alternatief bood voor de elektronenbuis en verdere ontwikkeling van de computer. In 1951 werden de eerste computers voor commercieel gebruik aangeboden en in 1953 bracht IBM zijn eerste computer op de markt. Vier jaar later werd de programmeertaal FORTRAN gelanceerd waarmee 'gewone' mensen zelf programma's voor de computer konden schrijven. In 1959 werd de IC (Integrated Circuit) uitgevonden waardoor computers veel krachtiger en sneller werden. Tegelijk werden computers niet alleen krachtiger maar ook goedkoper en steeds meer bedrijven schaften computers aan. De behoefte aan een standaard cijferschrift om bedrijven in staat te stellen met elkaar versleutelde berichten uit te wisselen, groeide. In de beginjaren zeventig ontwikkelde Horst Feistel een geducht algoritme en noemde het Lucifer. Hij moest voor zijn research tegen de stroom in werken, tegengewerkt door de NSA, de Amerikaanse National Security Agency, die het monopolie op cryptografische research opeist. Daarmee wilde de NSA voorkomen dat er cijferschriften in omloop kwamen die ze zelf niet konden ontcijferen. De NSA heeft meer wiskundigen in dienst, koopt meer hardware en onderschept meer berichten dan wie ook ter wereld.

Lucifer was zo sterk dat het zelfs voor de beste computers van de NSA van dat moment ondoenlijk werd de codes te ontcijferen. Men zegt dat onder druk van de NSA het aantal sleutels van het systeem Lucifer werd beperkt tot 56 bits, oftewel 100.000.000.000.000.000. De 56-bits encryptiestandaard is in 1977 onder de naam DES tot een federale standaard van de Verenigde Staten verheven. Het was op dat moment onmogelijk om met een burgercomputer een Brute Force Attack uit te voeren op de DES_standaard omdat deze teveel sleutelmogelijkheden telt.

Welbeschouwd is de ontwikkeling van de cryptografie met deze stap nog niet veel verder gekomen. Snellere computers zijn uiteindelijk weer in staat de DES-encryptie te breken, waarna DES opgevolgd zal worden door een sterkere standaard. Cryptografen en cryptoanalisten blijven op deze manier rondjes om de tafel rennen en het aloude probleem van de sleuteldistributie is hiermee nog altijd niet opgelost. In de jaren zeventig werden sleutels per ordonnans gedistribueerd. Ordonnansen waren mannen en vrouwen die als een vertrouwelijke koerier de hele wereld over reisden om sleutels naar klanten te brengen. Al gauw werden de bedrijfskosten onbetaalbaar naarmate het berichtenverkeer toenam. Regeringen en krijgsmachten hadden er alles voor over om te zorgen dat sleutels op een veilige manier werden gedistribueerd. De organisatie Communications Security (COMSEC) werd bijvoorbeeld speciaal in het leven geroepen om de tonnen aan VS-regeringssleutels veilig af te leveren bij de diverse ambassades. Het zou nog een paar jaar duren voordat er op dit terrein een doorbraak zou plaatsvinden die beschouwd mag worden als de grootste cryptografische ontwikkeling van de twintigste eeuw en het grootste succes sinds de uitvinding van het monoalfabetische cijferschrift, nu meer dan 2000 jaar geleden.

Opgave 7

Leg uit wat een 56-bits sleutel te maken heeft met het getal 100.000.000.000.000.000

 Reflectie

Waarom is het sleuteldistributieprobleem zo'n grote zorg voor bedrijven?

klik hier

De werking van het DES-algoritme wordt uitgelegd onder de kop 9.5A "Het DES-algoritme". De werking van het algoritme is tamelijk ingewikkeld. Voor het vervolg van de modul kan dit kopje zonder bezwaar worden overgeslagen.

9.5A Het DES-algoritme

Inhoud van de DES-pagina:

D.1 De techniek van het DES-algoritme
D.2 Wat doet de functie f met Ri en Ki?
D.3 Hoe ziet de deelsleutel Ki eruit?

D.1 - De techniek van het DES-algoritme

De behoefte aan een standaard cijferschrift om bedrijven in staat te stellen met elkaar versleutelde berichten uit te wisselen, groeide. In de beginjaren zeventig ontwikkelde Horst Feistel een geducht algoritme en noemde het Lucifer. Lucifer was gebaseerd op een 12-bits sleutel.
Het DES-algoritme (Data Encryption Standard) is door IBM ontwikkeld en in 1977 verheven tot federale standaard van de Verenigde Staten. Het vercijfert data-blokken van 64 bits en gebruikt daarbij een 64-bits sleutel, waarvan 8 controlebits, die niet meedoen met de vercijfering. Een opvolger van DES is AES (Advanced Encryption Standard), welke gebruik maakt van een grotere sleutel van 128, 192 of 256 bits.
Ook IDEA (International Data Encryption Algorithm), een algoritme uit 1990, gebruikt een 128-bits sleutel om data-blokken van 64 bits te vercijferen. Een toepassing van dit algoritme is het e-mail pakket Pretty Good Privacy (PGP), wat later nog ter sprake komt.

De genoemde algoritmes geven een **symmetrische** versleuteling, met andere woorden, de verzender en de ontvanger moeten beschikken over dezelfde sleutel. Ze zijn echter zeer gecompliceerd en zonder computer niet te gebruiken. We geven in deze paragraaf een toelichting van DES op hoofdlijnen. De vercijfering bestaat uit een reeks van permutaties en substituties. Onder een permutatie wordt verstaan een verwisseling en onder een substitutie een vervanging.

De klare tekst wordt in binaire vorm aangeboden en bestaat dus uit nullen en enen. Deze tekst wordt opgedeeld in blokjes van 64 bits. (In les 9 werd het binaire syteem uitgelegd). In de eerste stap worden de bits volgens een schema door elkaar geschud. Dit noemen we de Initiële permutatie (IP). De 64 bits worden vervolgens gescheiden in twee helften van 32 bits, de linkerhelft L_0 en de Rechterhelft R_0 (zie de afbeelding hiernaast). Dan volgen er 16 rondes waarbij steeds een deelsleutel K_i (met i=1,2,...16) wordt gebruikt voor een versleuteling van de rechterhelft. De cijfertekst wordt op een speciale manier opgeteld bij de linkerhelft en vormt de nieuwe rechterhelft. De oude rechterhelft wordt daarbij de nieuwe linkerhelft. Na de eerste iteratie is L_0 opgeteld bij de cijfertekst en niet meer herkenbaar. In de tweede iteratie wordt R_1 opnieuw vercijferd en is R_0, na een eerste verwisseling, opgeteld bij deze tweede vercijfering. Daarmee is R_0 ook niet meer herkenbaar. Ieder deel van de klare tekst wordt op deze manier 8 maal versleuteld.

Na 16 iteraties worden links en rechts niet verwisseld om aan te sluiten op de ontcijfering.

Na de 16^e iteratie vindt er opnieuw een permutatie plaats die de inverse is van de initiële permutatie aan het begin. Dit geeft de output van het eerste vercijferde blok van 64 bits.

Een paar vragen blijven over na dit eerste overzicht:

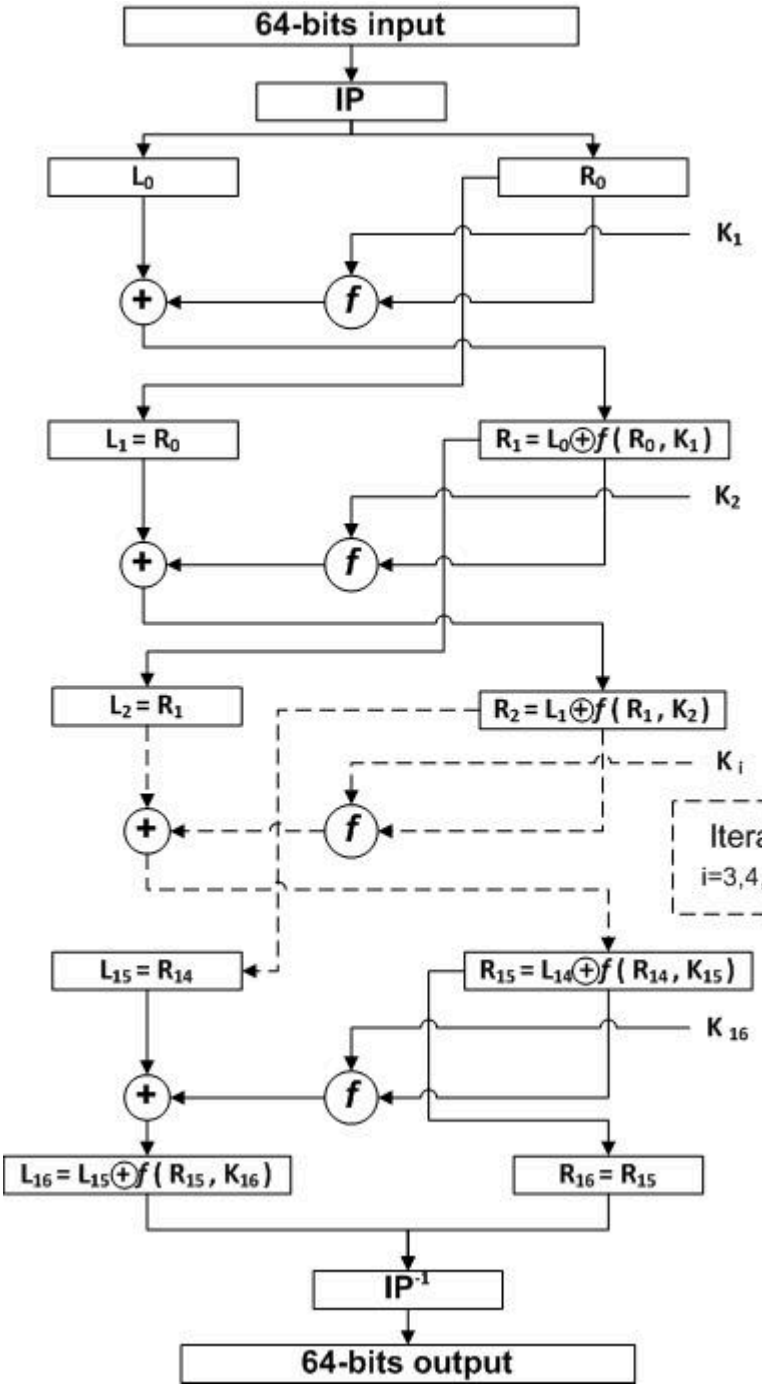
- hoe ziet de deelsleutel K_i eruit

- wat doet de functie *f* met R_i en K_i

- hoe werkt de vreemde optelling van *f* met L

De laatste vraag is het simpelst te beantwoorden. De optelling is een modulo 2 optelling, ook bekend als de exclusieve OR. De 32 bits van L_i en de 32 bits van $f(R_i,K_{i+1})$ worden onder elkaar gezet en paarsgewijs opgeteld. Daarbij is

0+0 mod(2) = 0
0+1 mod(2) = 1



1+0 mod(2) = 1
1+1 mod(2) = 0

Op de eerste twee vragen gaan we nog wat uitgebreider in.


Reflectie

Tel de volgende twee rijen bij elkaar op met gebruik van de mod(2) optelling. Wat is het resultaat?

0 0 0 1 1 0 1 1 1 0 0 1 1 0 0 1
0 0 1 1 0 0 0 0 0 1 1 1 0 1 1 1

klik hier

Ontcijfering gebeurt in tegengestelde richting, waarbij de deelsleutels ook in omgekeerde volgorde worden gebruikt. Het begint met de initiële permutatie. De twee helften zijn nu L_{16} en R_{16}. Er geldt

$R_{15}=L_{16}$ en $L_{15}=R_{16} + f(L_{16},K_{16})$ mod(2)
De output L_{16} is dus de eerste input voor de functie *f* op de weg terug.

Samengevat geldt op de heenweg **$L_i=R_{i-1}$ en $R_i=L_{i-1} + f(R_{i-1},K_i)$ mod(2)**
en op de weg terug geldt dan
$R_{i-1}=L_i$ en $L_{i-1}=R_i + f(L_i,K_i)$ mod(2)

Opgave 1

Als je de weg terug neemt lijkt het logischer om R_{16} terug te zetten op de plek van R_{15}. Daar komt deze immers vandaan.
Dat zou inhouden dat L_{16} via de functie *f* teruggeleid zou moeten worden naar L_{15}. Waarom is dit niet mogelijk?

D.2 - Wat doet de functie *f* met R_i en K_i?

De tweede vraag die we willen beantwoorden is wat de functie *f* met R_i en K_i doet:
Allereerst wordt de 32 bits rechterhelft in een tabel met 4 kolommen van 8 rijen gezet en uitgebreid tot een 48-bits tabel. Daarbij worden de entries in de eerste en laatste kolom gekopieerd naar de 5^een 0^e kolom:

32	1	2	3	4	5
4	5	6	7	8	9

8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Je ziet hiernaast de 1^e tot en met de 32^e bit ingevuld in de middelste 4 kolommen. Verder zijn de bits van de eerste kolom
scheef gekopieerd naar de 5^e kolom en de bits van de 4^e kolom scheef gekopieerd naar de 0^e kolom. De getallen stellen dus
de nummers van de bits voor en niet de bits zelf.

De rechterhelft bestaat dus nu uit 48 bits en wordt modulo 2 opgeteld bij een 48 bits deelsleutel. Deze deelsleutel is op een speciale manier afgeleid van de 64 bits sleutel. We zullen straks zien hoe dat gedaan wordt.

Het resultaat is een tabel van 8 rijen en 6 kolommen.

Stel dat het resultaat van de bewerking op R_0 met deelsleutel K_1 er als volgt uitziet:

1	1	1	0	0	1
0	1	1	0	0	0
1	1	0	1	0	0
0	1	1	0	1	1
0	0	0	1	1	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	0	0	1	1

Dit levert 8 rijen met 6 bits. De eerste rij bestaat uit 1 1 1 0 0 1, de tweede uit 0 1 1 0 0 0, en zo verder.
Neem de eerste rij als voorbeeld: 1 1 1 0 0 1. De buitenste 2 bits vormen het binaire getal 11, wat gelijk is aan het decimale getal 3.
De binnenste 4 bits vormen het binaire getal 1100, wat gelijk is aan het decimale getal 12.

Doen we hetzelfde met de tweede rij, dan vinden we 0 1 1 0 0 0. De buitenste 2 bits vormen het binaire getal 00, wat hetzelfde is
als het decimale getal 0. De binnenste vier bits vormen het binaire getal 1100, wat opnieuw gelijk is aan het decimale getal 12.

Opgave 2

Doe hetzelfde voor de derde t/m de achtste rij en zet de resultaten in een tabel.

Met het resultaat van opgave 2 wordt nu de substitutietabel geraadpleegd, die hieronder is afgebeeld.

S1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	7
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Uit Saïd El Aoufi: Cryptografie en ICT

Deze tabel bestaat uit 8 deeltabellen S_1t/m S_8. De eerste rij leverde ons de getallen 3 en 12. In deeltabel S_1vinden we in de derde rij en de twaalfde kolom het binaire getal 10.
De eerste rij wordt nu vervangen door de binaire representatie van het decimale getal 10,
dus door 1 0 1 0.

De tweede rij leverde ons de getallen 0 en 12. In deeltabel S_2vinden we in de 0^e rij, in de 12^ekolom het decimale getal 12.
De tweede rij wordt nu vervangen door de binaire representatie van het decimale getal 12,
dus door 1 1 0 0.

Zo gaan we rij voor rij langs en vervangen iedere rij door een binair getal van 4 cijfers.
Het resultaat is een 32 bits getal. We noemen dit $f(R_0,K_1)$.

Opgave 3

Stel met behulp van het resultaat van opgave 2 de 32-bits tabel samen van $f(R_0,K_1)$.

Uit opgave 3 volgt dat $f(R_0,K_1)$ nu het 32-bits getal
10101100001010100001011100001100 is.
Dit getal wordt modulo 2 opgeteld bij de 32-bits van L_0.
Het resultaat van deze bewerking is R_1.

D.3 - Hoe ziet de deelsleutel Ki eruit?

De laatste vraag die nu nog beantwoord moet worden is de vraag waaruit de 16 deelsleutels K_1 t/m K_{16}

bestaan en hoe die gevonden worden. In feite is het een eenvoudige bewerking in een spreadsheet.
Uitgangspunt is een 64-bits sleuteltabel van 8 rijen en 8 kolommen.
Van deze tabel worden volgens een schema 56 bits geselecteerd.
De 56 bits worden gesplitst in 2 rijtjes van 28 bits.
De volgorde binnen deze 28-bits rijtjes draait bij iedere iteratie 1 of 2 plaatsen door. Dit worden *SHIFT*s genoemd.
Hierbij worden de eerste bit of eerste 2 bits de laatste en de andere komen 1 of 2 plaatsen naar voren.
De twee helften worden teruggezet in een tabel en vervolgens worden er volgens een vast schema 48 bits uitgekozen. Bij iedere iteratie zijn de bits van plaats gewisseld bij het doordraaien, zodat iedere deelsleutel anders zal zijn. De schema's om 56 bits te selecteren en bij iedere iteratie 48 bits te selecteren staan vast. Het aantal SHIFTs staat ook per ronde vast.

Hieronder een rekenvoorbeeld waarin dit wordt toegelicht.

We kiezen een 64-bits sleutel, bijvoorbeeld:

1	1	0	1	0	0	1	0
1	1	0	1	1	1	1	0
1	1	0	0	1	0	1	0
1	0	1	0	1	0	0	1
0	1	0	1	0	0	1	1
1	1	0	0	0	1	0	0
1	0	1	0	1	0	0	1
0	0	1	0	0	1	0	1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Vervolgens wordt er een keuze gemaakt volgens het rechterschema. Hierin is te zien dat eerst de 57^{e} bit wordt geselecteerd (linksonder kleur oranje), vervolgens de 49^{e} (recht daarboven kleur geel) etcetera.
Zetten we deze op volgorde in 2 rijen van 28 bits dan krijgen we het volgende resultaat:
0 1 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 1

0 0 1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1 1

Zoals je kunt zien bestaat de laatste kolom uit niet-gebruikte bits. Dit worden controlebits of pariteitsbits genoemd.

We kijken nu naar het aanal SHIFTs. In onderstaand schema is dit vastgelegd

Ronde	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
SHIFTs	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Voor de eerste ronde schuiven alle bits 1 plaats op en de eerste gaan naar de laatste plaats:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
1 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0

29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
0 1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 1 1 0

Vervolgens worden er 48 bits gekozen volgens het linkerschema:

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

1	1	1	0	1	1
0	0	1	1	0	1
0	0	0	1	0	0
1	1	1	1	1	1
0	0	1	1	0	1
1	1	1	0	1	0
0	0	0	0	0	1
1	1	0	1	0	0

Het resultaat levert de rechtertabel voor K_1:

In de tweede iteratie verandert de volgorde binnen de rijen als gevolg van de SHIFT waardoor de matrix

K_2 verschilt van de matrix K_1. Hetzelfde geldt voor de hieropvolgende deelsleutels.

Opgave 4

Bereken K_2 .

De manier om het DES-algoritme te kraken is met behulp van een Brute-Force-Attack, oftwel een brute aanval. Het komt er op neer dat je gewoon alle 56-bits sleutels moet uitproberen en los moet laten op het algoritme. Dat lijkt simpel, maar er zijn 2^{56} mogelijke sleutels, oftewel 72.057.594.037.900.000. Met een snelle computer zou je wellicht 1 miljoen mogelijkheden per seconde uit kunnen proberen. Daarmee kun je de computer gemiddeld ruim 1000 jaar bezighouden.
In 1997 is het een groep wetenschappers onder leiding van Rocke Verser gelukt het DES-algoritme te kraken in een challenge die uitgeschreven was door het bedrijf RSA Data Security. Het bedrijf stelde een beloning van $10.000.= in het vooruitzicht voor degene die het algoritme als eerste zou weten te kraken. Daarvoor gebruikten zij niet 1 computer maar een hele serie computers. Daarmee duurde het 4 maanden om het algoritme te kraken. Op dat moment was pas 25% van alle sleutels uitgeprobeerd. De ontcijferde tekst luidde "Strong cryptography makes the world a safer place".

Opgave 5

Maak een schatting van het aantal computers dat de groep gebruikt heeft.

Opgave 6

Bereken hoeveel jaar het algoritme met een 128 bits sleutel stand zou hebben gehouden tegen een Brute Force Attack onder identieke voorwaarden als in bovenstaand voorbeeld.

Samenvatting deel 1

In de cryptografie spreken we over de ***klare tekst*** en de ***cijfertekst***. Het omzetten van de klare tekst in cijfertekst heet ***versleutelen*** of ***encrypten.*** Het systeem om tekst te encrypten en ***decrypten*** (***ontcijferen***) wordt het ***encryptiesysteem*** of het ***cijfer*** genoemd en een ***algoritme*** beschrijft het proces van versleuteling. Het systeem maakt gebruik van ***sleutels*** die alleen bekend zijn aan de verzender en ontvanger van het bericht. Als een derde het bericht onderschept kan hij proberen de sleutel te achterhalen en daarmee de code te ontcijferen. We noemen dit het ***kraken van de code***. Het systeem is over het algemeen goed beschreven en bekend, maar de sleutel blijft geheim. Eenvoudige systemen beschikken over een klein aantal sleutels. Het Caesar-cijfer kent bijvoorbeeld maar 26 sleutels. Ingewikkelde systemen beschikken over onvoorstelbaar veel sleutels. Zo telt het DES-algoritme 72.057.594.037.900.000 sleutels. Hoe meer sleutels hoe moeilijker het wordt om de juiste te vinden en de code te kraken.
Eigenlijk is het woord ***code*** bedoeld om in het geheim te spreken over een bepaalde gebeurtenis of afspraak, zoals operatie *Overlord* stond voor de invasie in Normandië. Een codewoord kan ook een symbool zijn dat bijvoorbeeld een persoon of een woord voorstelt. In het taalgebruik wordt ***coderen*** echter ook gebruikt op de plaats van ***vercijferen.***

De eerste vormen van geheimschrift waren voorbeelden van ***steganografie***, de kunst van het verbergen van een boodschap. Een nadeel van deze methode is, dat als een boodschap wordt gevonden, de onderschepper van het bericht de boodschap direct kan lezen. ***Cryptografie*** is begonnen met het verwisselen van de letters in de klare tekst. Dit principe staat bekend als *transpositie* of *verwisseling*. Een bekend voorbeeld daarvan is de scutalè, die al in de vijfde eeuw voor Christus door de Spartanen gebruikt werd. Een ander voorbeeld is de hekvercijfering (Railfence).
De eerste goed beschreven vorm van militaire encryptie is het ***Caesar-cijfer***. Dit is het simpelste voorbeeld van substitutie. Het is een ***schuifsysteem,*** dat wil zeggen dat iedere letter een vast aantal plaatsen opschuift in het alfabet. De a wordt dan bijvoorbeeld vervangen door de D en de b door de E, de c door de F, etcetera. Een ander voorbeeld is het ***Kamasutra-cijfer*** waarin tweetallen van letters worden verwisseld, zoals bijvoorbeeld de e door de O en de o door de E, of het varkenshokcijfer waarin iedere letter wordt vervangen door een symbool.
Een meer algemene vorm van substitutie herschikt het alfabet op een willekeurige manier. Dit ***mono-alfabetische*** substitutiesysteem heeft ruime toepassing gevonden en heeft een onuitputtelijke hoeveelheid sleutels. Het was de methode die tot in de zestiende eeuw gebruikelijk was. De betrouwbaarheid van deze methode nam af toen analisten gebruik gingen maken van de ***frequentieanalyse.*** Dit was in de Arabische wereld al in de 9^{e} eeuw bedacht door Al-Kindi, maar drong pas aan het eind van de 15^{e} eeuw door tot Europa. Voor die tijd werd nog het ***Atbash*** door monniken gebruikt, een variatie op het Kamasutra-cijfer.
Frequentieanalyse is gebaseerd op het idee dat bepaalde letters in de taal vaker voorkomen dan andere. Door in een tekst de frequenties van de voorkomende letters te tellen ontdek je vrij snel waar sommige letters uit het cijferalfabet voor staan. Dit biedt de mogelijkheid om delen van de tekst te raden en zo stap voor stap de code te kraken.
Aanvankelijk stelden de cryptografen zich tegen de frequentieanalyses teweer door ***nieten*** in te voeren in de cijfertekst. Dit zijn symbolen die geen betekenis hebben maar alleen bedoeld zijn om de frequentieanalyse te frustreren. Ook het gebruik van ***codewoorden*** en de foute spelling van woorden was een krachtig middel. Een cryptosysteem dat gebruik maakt van een sleutel en van codewoorden wordt een ***Nomenclator*** genoemd. Een beroemd voorbeeld daarvan is het systeem wat Mary Stuart gebruikte en uiteindelijk tot haar ondergang leidde.

Een nieuwe weg werd ingeslagen door Alberti in de vijftiende eeuw met de ***dubbele encryptie***. Volgens dit systeem worden de letters om en om versleuteld met twee verschillende sleutels of cijferalfabetten. Het was het begin van de ***polyalfabetische substitutie***. Dit systeem maakt gebruik van een codewoord. Iedere letter van het codewoord geeft een ander schuifsysteem en dus een ander cijferalfabet aan. Neem je het codewoord SUPER, dan wordt gebruik gemaakt van 5 schuifsystemen die afwisselend worden gehanteerd om de klare tekst te vercijferen. Dit systeem werd in de zestiende eeuw bedacht door Belaso en kennen we als het ***Vigenère-systeem***, genoemd naar de man die het systeem verbeterde. Het systeem is echter zeer bewerkelijk waardoor cryptografen liever gebruik maakten van andere instrumenten om de frequentieanalyse onschadelijk te maken,zoals het ***homofone syteem***, dat men in de zeventiende eeuw bedacht. Voor veelkomende letters worden hierin eenvoudig meerdere

symbolen gebruikt waardoor de verschillende frequenties worden geneutraliseerd. Een vervolg op het Vigenère-systeem werd aan het einde van de Eerste Wereldoorlog bedacht door Amerikaanse wetenschappers en wordt het ***Autokey-systeem***genoemd. Dit is pas echt ongevoelig voor frequentieanalyse.

Een cijfer dat het tot 1890 volhield en bedacht was door Rossignol, was het ***Grand Chiffre***. Het vervangt lettergrepen door getallen. Uiteindelijk wist Bazeries deze code te kraken. Het is feitelijk een opgevoerd monoalfabetisch systeem.

De automatisering van de cryptografie begon in primitieve vorm met een apparaat bedacht door Alberti in de 15^{e} eeuw, maar de industrialisatie in de 19^{e} eeuw maakte het pas mogelijk om het systeem van Vigenère op grote schaal in te voeren. De opkomst van de telegraaf in het begin van de 19^{e} eeuw maakte het ook noodzakelijk om een veiligere encryptiemethode te gaan gebruiken. In de tweede helft van de 19^{e} eeuw werd de Vigenère code gekraakt door Kasiski en Babbage. Ze vonden een methode om de sleutellengte te achterhalen en dit maakte het mogelijk om de letters van de cijfertekst op te splitsen in groepen die volgens hetzelfde systeem waren versleuteld, waardoor deze weer gevoelig waren voor frequentieanalyse.
Inmiddels waren in de diverse landen door de regeringen groepen cryptografen en cryptoanalisten gevormd, de zogenaamde ***Zwarte Kamers***. Deze moesten ervoor zorgen dat de regeringen op een veilige manier boodschappen konden overbrengen en onderschepte boodschappen van andere regeringen konden lezen.
In de tweede helft van de negentiende eeuw kreeg het grote publiek belangstelling voor geheimschriften. Een veelgebruikte methode was de speldenprik-encryptie, wat eigenlijk een vorm van steganografie is. Ook in boeken, zoals in het boek van Jules Verne, werd gespeeld met versleutelde stukken tekst. Ook Sherlock Holmes toonde zich een bedreven cryptoanalist door het***Dancing Man Cipher*** te breken, een vorm van monoalfabetische substitutie.

Aan het einde van de negentiende eeuw werd het ***boekcijfer*** bedacht, waarvan het ***Beale-cijfer***een voorbeeld is. Dit cijfer is eigenlijk niet te breken omdat het voortdurend verwijst naar letters in een boek. De onderschepper van een bericht weet niet om welk boek het gaat en kan onmogelijk alle boeken en geschriften uitproberen. Een andere vorm van het boekcijfer is het ***eenmalig blokcijfer***, in 1918 geïntroduceerd door Mauborgnes. Dit is feitelijk onbreekbaar maar het lastigste in deze methode van encryptie is de distributie. Een andere encryptiemethode uit de negentiende eeuw is het ***Playfair-cijfer***, vernoemd naar zijn bedenker. In deze methode wordt steeds een tweetal letters volgens een kruistabel vervangen door twee andere letters. De volgorde van de letters in de kruistabel (de sleutel) wordt bepaald door een sleutelwoord.

De uitvinding van de radio door Marconi aan het einde van de negentiende eeuw, zorgde ervoor dat op nog grotere schaal gecommuniceerd kon worden. De behoefte aan veilige encryptiemethoden werd steeds groter. De Duitsers bedachten in de Eerste Wereldoorlog het ***ADFGVX-systeem***, gekraakt door Painvin in juni 1918. In 1916 onderschepte de Britse zwarte kamer ***Kamer 40*** een telegram van de Duitse minister Zimmermann, wat van grote invloed zou blijken op het verdere verloop van de oorlog en waardoor Amerika bij de oorlog betrokken werd.

Na de Eerste Wereldoorlog werd gezocht naar nieuwe methoden om encryptie te vergemakkelijken. De Duitser Scherbius bedacht de ***Enigma***, een apparaat dat op zeer ingenieuze wijze de klare tekst versleutelt. Meerdere draaiende schijven, *scramblers*, zorgen voor een meervoudige substitutie van de klare tekst en een schakelbord voegt daar een verwisseling van een deel van de letters aan toe. De manier waarop de scramblers de letters verwisselde was in het begin geheim maar werd verraden door Hans-Thilo Schmidt. De sleutel bestond uit de volgorde van de scramblers, de beginstand van de scramblers en de instelling van het schakelbord. De Pool Rejewski, die dankzij het verraad van Schmidt de werking van de scramblers kon nabootsen wist patronen te ontdekken in de werking van de scramblers. Daarmee legde hij een catalogus aan waarmee het mogelijk werd om de sleutel te breken. Hij maakte daarbij gebruik van zogenaamde ***Bombes***, wat geautomatiseerd een ***Brute Aanval*** op de gecodedeerde tekst uit kan voeren.

De Enigma werd verder uitgebreid met meer scramblers en een uitgebreider schakelbord bij aanvang van de Tweede Wereldoorlog. In de jaren veertig richtten de Britten een groot centrum in ***Bletchley Park*** op als opvolger van Kamer 40. Voortbouwend op de resultaten van Rejewski wisten de Britten de werking van de Enigma lange tijd te ontcijferen, voornamelijk omdat de Duitsers hun bericht altijd vooraf

lieten gaan door een code die direct herhaald werd. Zogenaamde ***Cilly's***, voorspelbare berichtsleutels, en andere voorspelbare zaken wisten het breken van de code vaak nog te bespoedigen. Alan Turing bedacht een methode die niet afhankelijk was van de dubbele code aan het begin van het bericht omdat hij vermoedde dat de Duitsers daar mee zouden stoppen als ze in de gaten kregen wat het gevaar daarvan was. Hij bouwde een eigen type ***Bombe*** en maakte gebruik van ***spiekers***, die hij afleidde uit de bestudering van oude ontcijferde berichten. Zo wist hij dat om 6 uur 's morgens altijd een weerbericht werd uitgezonden. Hij kon daardoor delen van de tekst raden en dit bracht hem verder. Hij neutraliseerde de werking van de scramblers door op een slimme manier meerdere Enigma's aan elkaar te koppelen.
De Amerikanen maakten in de Tweede Wereldoorlog gebruik van ***Navajo-indianen*** die de communicatie tussen schepen verzorgden. Het Navajo is een taal die geen invloeden kent van andere talen waardoor het volstrekt onbegrijpelijk was voor de meeluisterende vijand.

De uitvinding van de computer leidde tot meer wiskundige modellen, waarin de letters worden omgezet in getallen waarna er berekeningen op losgelaten kunnen worden in een cryptografisch substitutiesysteem. De ***ENIAC*** staat bekend als de eerste gerealiseerde computer, maar in de Tweede Wereldoorlog had Tommy Flowers al een idee van Max Newman uitgewerkt en de ***Colossus***gebouwd. Deze computer werkte op 1500 elektronenbuizen.
Uitvinding van de ***transistor*** in 1947 maakte het mogelijk op commerciële basis computers te produceren. De uitvinding van het ***Integrated Circuit*** in 1959 verhoogde de rekenkracht van de computers waardoor het mogelijk werd om zeer ingewikkelde algoritmes te automatiseren. Wat handmatig ondoenlijk is wordt met computerprogramma's eenvoudig gemaakt. Zo bedacht Feistel in de jaren zeventig het ***Lucifer-systeem***. Op basis van dit systeem ontwikkelde IBM de 56-bits encryptiestandaard ***DES,*** die in 1977 in Amerika werd verheven tot *Federal Encryption Standard*. Men vermoedt dat onder invloed van de Amerikaanse veiligheidsdienst NSA de sleutel werd beperkt tot 56-bits omdat daarmee de snelste computers nog in staat bleken in een Brute Aanval de sleutel te achterhalen. Alleen de NSA beschikte over een computer met deze snelheid. DES hield stand tot het in 1997 gebroken werd door een groep onder leiding van Rocke Verser. Vergroting van het aantal sleutels wordt toegepast in ***AES***. Een andere opvolger van DES is ***IDEA***.

Al deze encryptiesystemen vallen onder het type van de symmetrische versleuteling en kennen het grote probleem van de ***sleuteldistributie***.

Over dit lesmateriaal

Colofon

Auteurs	Bètapartners
Team	Wikiwijs Maken Auteurs
Laatst gewijzigd	18 december 2014 om 14:11
Licentie	De Nederlandse Creative Commons 3.0 licentie waarbij de gebruiker het werk mag kopiëren, verspreiden en doorgeven en afgeleide werken mag maken onder de voorwaarden: Naamsvermelding en Gelijk Delen, zie http://creativecommons.org/licenses/by-sa/3.0/nl/. Meer informatie over de CC Naamsvermelding-GelijkDelen 3.0 Nederland licentie licentie.

Aanvullende informatie over dit lesmateriaal

Van dit lesmateriaal is de volgende aanvullende informatie beschikbaar:

Leerniveaus	HAVO 5
Leerinhoud en doelen	Wiskunde D, Inzicht en handelen
Eindgebruiker	leerling/student
Trefwoorden	e-klassen rearrangeerbaar