3. Kryptographie

3. Kryptographie

Kryptographie ist die Lehre von den Methoden zur Ver- und Entschlüsselung von Nachrichten

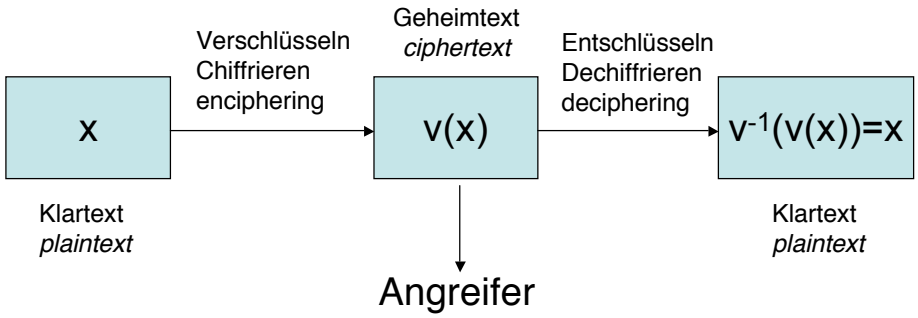
Kryptoanalyse ist die Wissenschaft von der den Methoden der Entschlüsselung von Nachrichten, ohne den Schlüssel zu haben.

Kryptologie umfasst Kryptographie und Kryptoanalyse

Kryptographie ist die Lehre von Methoden zur Ver- und Entschlüsselung von Daten/Nachrichten mit Hilfe mathematischer Verfahren. Dank der Kryptographie können vertrauliche Daten gespeichert oder über unsichere Netzte (z.B. das Internet) übertragen werden, so dass diese nur vom eigentlichen Empfänger gelesen werden können. *Kryptoanalyse* ist die Wissenschaft von der Analyse und vom Entschlüsseln verschlüsselter Daten. *Kryptologie* umfasst sowohl die Kryptographie als auch die Kryptoanalyse.

3. Kryptographie

Verschlüsselungsverfahren legt fest, wie Klartexte in Geheimtexte verschlüsselt und wieder entschlüsselt werden



Ein Verschlüsselungsverfahren legt fest, wie Klartexte in Geheimtexte transformiert (verschlüsselt) und wie Geheimtexte wieder in Klartexte zurück transformiert (entschlüsselt) werden. Das Ziel eines Verschlüsselungsverfahrens ist die Geheimhaltung der Nachricht gegenüber Dritten (Angreifern).

3. Kryptographie

Verschlüsselung dient zur

- Sicherung der Vertraulichkeit übertragener Information
- Sicherung der Vertraulichkeit gespeicherter Information
- Prüfung der Integrität einer übertragenen Nachricht (Message Authentication Code)
- Prüfung der Authentizität von Personen (digitale Unterschrift)
- Prüfung der Authentizität von Softwarekomponenten (Zertifikate)

Die Einsatzgebiete der Verschlüsselung liegen beispielsweise in der Sicherung übertragener und gespeicherter Information, in der Prüfung der Integrität einer übertragenden Nachricht oder Datei (Message Authentication Code) oder der Authentizität von Personen (digitale Unterschrift) oder von Softwarekomponenten (ausgedrückt durch Zertifikate).

3. Kryptographie

Symmetrische Verschlüsselungsverfahren
- Transposition
- Substitution
- DES, Triple-DES, AES,...

Asymmetrische Verschlüsselungsverfahren
- RSA,...

Kerckhoff-Prinzip: Sicherheit gegen Angriffe wird durch Geheimhaltung des Schlüssels – nicht des Verfahrens – erreicht!

Man kann zwei Klassen von Verschlüsselungsverfahren unterscheiden, nämlich *symmetrische* und *asymmetrische*. In einem symmetrischen Verfahren haben beide Kommunikationspartner einen gemeinsamen, geheimen Schlüssel. Das bedeutet, dass sie sich vorab über einen gemeinsamen Schlüssel verständigen müssen, was eine zusätzliche Unsicherheit bedeutet. Bei asymmetrischen Verfahren besitzt jeder Kommunikationspartner ein Schlüsselpaar bestehend aus persönlichem und geheimen Schlüssel. Der öffentliche Schlüssel wird öffentlich bekannt gemacht, zum Beispiel in einem Schlüsselverzeichnis im Internet. Im Gegensatz zu symmetrischen Verfahren müssen bei asymmetrischen Verfahren keine geheimen Schlüssel ausgetauscht werden. Ein bekanntes asymmetrisches Verfahren ist das RSA-Verfahren (nach R. RIVEST, A. SHAMIR und L. ADLEMAN), welches die Primfaktorenzerlegung natürlicher Zahlen verwendet.

Für Verschlüsselungsverfahren wird gefordert, dass die Sicherheit des Systems nicht von der Geheimhaltung der Algorithmen zur Ver- und Entschlüsselung abhängen, sondern von der Geheimhaltung der Schlüssel (Prinzip von Kerckhoff).

3.1 Symmetrische Verfahren

Transpositionsverfahren: Nachrichtenteile werden umgestellt (permutiert)

Substitutionsverfahren: Nachrichtenteile werden ersetzt

- Cäsar, Vigenere, Playfair...

Kombinierte Substitution/Transposition

- Data Encryption Standard (DES)

Bei Transpositionsverfahren wechselt jeder Buchstabe innerhalb des Klartextes seinen Platz, doch selbst bleibt er unverändert. Bei Substitutionsverfahren werden Nachrichtenteile ersetzt. Transposition und Substitution können auch in einem Verfahren kombiniert werden (z.B. im Data Encryption Standard).

3.1 Symmetrische Verfahren

Transpositionsverfahren

Beispiel 1: Permutation mit fester Periode 4

D E R S | C H A T | Z I S T

Schlüssel: 3 2 4 1

R E S D | A H T C | S I T Z

Bei einer Transposition (auch Permutation genannt) wird die Anordnung der Klartextzeichen vertauscht. Man fasst n Buchstaben zusammen und permutiert diese entsprechend eines Schlüssels.

3.1 Symmetrische Verfahren

Beispiel 2: Transpositionsmatrix

Schlüssel ergibt die Anzahl der Spalten

Geheimtext wird spaltenweise ausgelesen

D E R S	Schlüssel = 4
C H A T	Geheimtext = DCZEHIRASSTT
Z I S T	

Als Grundlage in diesem Transpositionsverfahren dient eine Matrix, in die das zu verschlüsselnde Wort zeilenweise eingetragen wird. Die Anzahl der Spalten ist der geheime Schlüssel. Daraus und der Gesamtlänge des Klartextes ergibt sich die Anzahl der Zeilen der Matrix. Der Geheimtext wird spaltenweise aus der Matrix ausgelesen. Mit der Kenntnis des Schlüssels und der Länge des Geheimtextes kann die Matrix wiederhergestellt werden (d.h. die Anzahl der Spalten kann berechnet werden) und ausgelesen werden.

3.1 Symmetrische Verfahren

Was steckt hier dahinter?

```
IDTECEONHRRBBDEAIOIRNKST
```

```
I C H B I N
D E R D O K
T O R E I S
E N B A R T
```

Schlüssel = 6

3.1 Symmetrische Verfahren

Substitutionsverfahren

Klartext-Buchstaben werden Geheimtext-Buchstaben zugeordnet

Monoalphabetische Substitution

Jeder Buchstabe des Klartext-Alphabets wird durch einen Geheimtext-Buchstaben ersetzt (z.B. Cäsar-Chiffre)

Polyalphabetische Substitution

Einem Klartextbuchstaben werden Geheimtext-Buchstaben mehrerer Alphabete zugeordnet (z.B. Vigenere-Chiffre)

Bei Substitutionsverfahren werden die Zeichen des Klartextes durch Zeichen des Geheimtextes ersetzt. Hierbei kann jedes Zeichen genau einem festen Symbol zugeordnet sein (Geheimtext verwendet nur ein Alphabet), oder einem Buchstaben können mehrere Symbole entsprechen (mehrere Alphabete für den Geheimtext). Im ersten Fall spricht man von einer monoalphabetischen Verschlüsselung, im zweiten Fall von einer polyalphabetischen Verschlüsselung.

3.1.1 Cäsar-Chiffre

Cäsar-Chiffre

Verschieben des Alphabets

Beispiel: verschieben um 3 Buchstaben

Klartext: a b c d e f g h i j k l m ...

Geheimtext: D E F G H I J K L M N O P ...

GLH VSLQQHQ GLH JDOOLHU =

Die spinnen die Gallier

Ein Beispiel für eine monoalphabetische Verschlüsselung ist der Verschlüsselungscode von Julius Cäsar, bei dem das Alphabet verschoben wird. Der Schlüssel ist hierbei die Anzahl der Zeichen, um die das Alphabet verschoben wird.

3.1.1 Cäsar-Chiffre

2 6 19 45 – 16 20 18 44 10 19 – 2 14 23 – 9 23 10 14 – 0 19 24 – 2 14 10 9 36 23 – 10 19 25 12 36 38 62 45 –

14 19 – 9 20 19 45 10 23 – 14 19 – 7 17 14 25 5 10 19 – 14 19 – 23 10 12 36 19

Restklassenkörper Z_{26}:

a => [0] = { x | x mod 26 = 0}

b => [1] = { x | x mod 26 = 1} .....

cgtt qusskt cox jxko aty cokjkx ktzmkmkt ot juttkx ot hrozfkt ujkx ot xkmkt

In diesem Beispiel ist jeder Buchstabe des Alphabetes genau einer Restklasse des Körpers Z_{26} zugeordnet: Dem Buchstaben a die Restklasse [0]={x | x mod 26 = 0}, der Buchstabe b der Restklasse [1] ={x | x mod 26 = 1},..., und der Buchstabe z der Restklasse [25] ={x | x mod 26 = 25}. Wendet man diese Zuordnung zur Entschlüsselung auf die obige Zahlenreihe an, erhält man die Buchstabentext am Ende der Folie. Dieser Text ist mit einer Cäsar-Chiffre verschlüsselt, bei der das Alphabet um fünf Stellen verschoben ist.

3.1.1 Cäsar-Chiffre

cgtt qusskt cox jxko aty cokjkx ktzmkmkt
ot juttkx ot hrozfkt ujkx ot xkmkt

Cäsar-Chiffre mit Verschiebung um 5 Buchstaben, d.h. a = F

wann kommen wir drei uns wieder entgegen
in donner in blitzen oder in regen

3.1.1. Cäsar-Chiffre

Spezialfall **ROT13** symmetrische Zuordnung der Zeichen (a = N, n = A, ...)

Cäsar-Chiffre hat 25 verschiedene Schlüssel.

Monoalphabetische Substitutionsverfahren mit einem Alphabet von 26 Zeichen haben 26! Schlüssel.

Alle Formen der monoalphabetischen Substitution können (bei natürlich sprachlichen Texten) durch Häufigkeitsanalysen gebrochen werden.

Die Substitutionsmethode mit A=N ist eine Sonderform des Cäsar-Codes. Sie zeichnet sich dadurch aus, dass die Zuordnung der Buchstaben symmetrisch ist, d.h. (A=N, N=A usw.). Dieses Verfahren ist auch als ROT13-Verfahren bekannt, weil die Buchstaben um 13 Stellen rotiert werden. Der Schlüsselraum bei der Cäsar-Chiffrierung besitzt nur 25 Schlüssel. Ein mit Cäsar verschlüsselter Text kann daher leicht durch eine Brute-Force-Attacke (d.h. das Durchsuchen des gesamten Schlüsselraumes) geknackt werden. Bei einer allgemeinen monoalphabetischen Substitution hat man schon einen Schlüsselraum der Größe 26!. Da bei monoalphabetischer Substitution jedoch die Buchstabenhäufigkeit erhalten bleibt, können diese Verfahren über eine *Häufigkeitsanalyse* geknackt werden.

3.1.2 Häufigkeitsanalyse

Zeichen	Englisch	Deutsch
A	8,04%	6,47%
B	1,54%	1,93%
C	3,06%	2,68%
D	3,99%	4,83%
E	12,51%	17,48%
F	2,30%	1,65%
G	1,96%	3,06%
H	5,49%	4,23%
I	7,26%	7,73%
J	0,16%	0,27%
K	0,61%	1,46%
L	4,14%	3,39%
M	2,53%	2,58%

Zeichen	Englisch	Deutsch
N	7,09%	9,84%
O	7,60%	2,98%
P	2,00%	0,96%
Q	0,11%	0,02%
R	6,12%	7,54%
S	6,54%	6,83%
T	9,25%	6,13%
U	2,71%	4,17%
V	0,99%	0,94%
W	1,92%	1,48%
X	0,19%	0,04%
Y	1,73%	0,08%
Z	0,09%	1,14%

Der Gelehrte Al-Kindi erkannte im 9. Jahrhundert die unterschiedliche Häufigkeit von Buchstaben in einer natürlichen Sprache. Natürliche Sprachen haben meist relativ wenige Buchstaben, die sehr unregelmäßige Häufigkeiten aufweisen. In einem monoalphabetisch verschlüsseltem Text können diese Strukturen gefunden werden und bilden einen Angriffspunkt für das Brechen des Geheimtextes. In der obigen Tabelle sind die Buchstabenhäufigkeiten der englischen und deutschen Sprache gezeigt.

3.1.2 Häufigkeitsanalyse

PR ISRSQ YSPUD SYOCREBS GPS NFRZB GSY NCYBVEYCWDPS SPRS ZVOUDS HVOONVQQSRDSPB, GCZZ GPS NCYBS SPRSY SPRMPESR WYVHPRM GSR YCFQ SPRSY ECRMSR ZBCGB SPRRCDQ FRG GPS NCYBS GSZ YSPUDZ GSR SPRSY WYVHPRM. QPB GSY MSPB ASTYPSGPEBSR GPSZS FSASYQCSZZPE EYVZZSR NCYBSR RPUDB OCSRESY, FRG QCR SYZBSOOBS SPRS NCYBS GSZ YSPUDZ, GPS ESRCF GPS EYVSZZS GSZ YSPUDZ DCBBS.

AVYESZ, HVR GSY ZBYSRES GSY JPZZSRZUDCTB

Auf den folgenden Folien ist die Anwendung der Häufigkeitsanalyse an einem Beispiel gezeigt. Das Beispiel stammt aus *Simon Singh, Geheime Botschaften, Hanser Verlag*.

Schritt 1: Häufigkeiten des Geheimtextes

Zeichen	Häufigkeit	in %		Zeichen	Häufigkeit	in %
A	3	0,9		N	7	2,1
B	20	6,1		O	7	2,1
C	18	5,5		P	30	9,1
D	11	3,3		Q	8	2,4
E	12	3,6		R	32	9,7
F	6	1,8		S	67	20,4
G	20	6,1		T	2	0,6
H	4	1,2		U	7	2,1
I	1	0,3		V	10	3,1
J	1	0,3		W	3	0,9
K	0	0		X	0	0,0
L	0	0		Y	29	8,8
M	5	1,5		Z	24	7,3

Schritt 2: Buchstaben-Zuordnung ermitteln

These1: häufigster Buchstabe S = e

These 2: R,P,Y und Z könnten n, i, s, r entsprechen

Bigramme (Zweierkombinationen von Buchstaben) betrachten

RS = 7 PS = 8

SR = 13 SP = 13

YS = 5 ZS = 4

SY = 11 SZ = 7

These 3: SR,SP,SY für er, en, ei

Bigramm	Häufigkeit
en	3,88%
er	3,75%
ch	2,75%
te	2,26%
de	2,00%
nd	1,99%
ei	1,88%
ie	1,79%
in	1,67%
es	1,52%

Trigramme (Dreierkombination)

Häufigstes Trigramm (mit e am Anfang) im Deutschen ist ein

These 4: SPR kommt im Geheimtext 7-mal vor, daher P=i und R=n

These 5: Häufigstes Wort im Deutschen ist die, daher GPS = die, d.h. G = d

These 6: Zweithäufigstes Wort ist der, GSY kommt 4-mal vor, GSZ 3-mal und SY kommt 11-mal, SZ 7-mal vor

→ Y=r und Z = s

Mit S=e, R=n, P=i, Y=r und Z=s ergibt sich...

in IeneQ reiUD erOCnEBe die NFnsB der NCrBVErCWDie eine sVOUDe HVOONVQQenDeiB, dCss die NCrBe einer einMiEen WrVHinM den rCFQ einer ECnMen sBCdB einnCDQ Fnd die NCrBe des reiUDs den einer WrVHinM. QiB der MeiB AeTriediEBen diese FeAerQCessiE ErVssen NCrBen niUDB OCenEer, Fnd QCn ersBeOOBe eine NCrBe des reiUDs, die EenCF die ErVesse des reiUDs DCBBe.

AVrEes, HVn der sBrenEe der JissensUDCTB

....Lösung des Rätsels....

In jenem Reich erlangte die Kunst der Kartographie eine solche Vollkommenheit, dass die Karte einer einzigen Provinz den Raum einer ganzen Stadt einnahm und die Karte des Reichs den einer Provinz. Mit der Zeit befriedigten diese uebermaessig grossen Karten nicht laenger, und man erstellte eine Karte des Reichs, die genau die Groesse des Reichs hatte.

Borges, Von der Strenge der Wissenschaft

3.1.3 Vigenere

Beispiel: Vigenere (1586)

polyalphabetisch

Vigenere-Quadrat

	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
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	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	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

Während die Cäsar-Verschlüsselung eine monoalphabetische Verschlüsselung ist, ist die Vigenere-Verschlüsselung ein Beispiel einer polyalphabetischen Verschlüsselung. Sie wurde im Jahre 1586 von dem französischen Diplomaten Blaise de Vigenère (1523 bis 1596) der Öffentlichkeit zugänglich gemacht. Der Hauptgedanke dieser Methode ist, verschiedene monoalphabetische Chiffrierungen im Wechsel zu benutzen. Um eine Nachricht zu verschlüsseln, wird ein sogenanntes Vigenere-Quadrat und ein Schlüsselwort benutzt. Das Vigenere-Quadrat besteht aus 26 Alphabeten, bei dem jedes um einen Buchstaben gegenüber dem vorigen verschoben ist.

3.1.3 Vigenere

Schlüsselwort: LICHT

Klartext: Truppenabzug nach Ost

l	i	c	h	t	l	i	c	h	t	l	i	c	h	t	l	i	c	h
t	r	u	p	p	e	n	a	b	z	u	g	n	a	c	h	o	s	t
E	Z	W	W	I	P	V	C	I	S	F	O	E	H	V	S	W	U	A

Geheimtext

Beim Vigenere-Verfahren wird jeder Buchstabe des Klartextes mittels einer anderen Zeile des Vigenere-Quadrats verschlüsselt. Hierzu wird ein Schlüsselwort verwendet, über das sich der Sender und Empfänger der Nachricht vorher einigen müssen. Das Schlüsselwort wird über den Klartext geschrieben und so lange wiederholt, bis jeder Buchstabe des Klartextes mit einem Buchstaben des Schlüsselwortes verknüpft ist. Der Geheimtext wird dann wie folgt ermittelt: Der Schlüsselbuchstabe, der über dem Klartextbuchstaben steht, bestimmt das Alphabet. Im Beispiel steht über dem ersten Klartextbuchstaben t ein l. Daher wird das Alphabet des Vigenere-Quadrats genommen, dass mit l anfängt. Hier wird dann geguckt, welcher Buchstabe für das t zur Verschlüsselung genommen werden muss. Im Beispiel des Buchstabens t ist es der Buchstabe E.

Ein Angriff mittels Häufigkeitsanalyse ist bei der Vignere-Verschlüsselung nicht sinnvoll, da die Häufigkeit der Buchstaben gleichmässiger verteilt ist. Aber auch diese Verschlüsselung ist mit heutigen Methoden leicht zu knacken.

3.1.3 Vigenere

Angriffe auf die Vigenere-Verschlüsselung

Häufigkeitsanalyse führt nicht zum Ziel
Ziel ist die Ermittlung der Schlüssellänge

Kasiski-Test

Misst den Abstand von wiederholten Zeichenketten mit mindestens drei Zeichen.

Nutzt aus, dass ein relative kurzes Schlüsselwort häufig wiederholt wird.

Friedman-Test

Statistisches Verfahren, welches auf der Wahrscheinlichkeit basiert, dass zwei zufällig aus einem Text gewählte Zeichen gleich sind.

Die Häufigkeitsanalyse führt bei polyalphabetischer Verschlüsselung nicht mehr zum Ziel, da die Buchstabenhäufigkeit im Geheimtext nicht mehr der Buchstabenhäufigkeit der natürlichen Sprache entspricht. Deshalb wurden andere Angriffsmethoden entwickelt. Bei der Analyse einer Vigenère-Chiffre ist das wichtigste Ziel die Bestimmung der Schlüssellänge *k*, denn wenn man diese bestimmt hat, so funktioniert der Rest der Entschlüsselung wie bei einer gewöhnlichen monoalphabetischen Verschlüsselung. Sowohl der Kasiski-Test als auch der Friedmann-Test sind Verfahren zur Bestimmung der Schlüssellänge. Der Kasisiki-Test sucht im Geheimtext nach Wiederholungen von Zeichenfolgen und misst deren Abstand. Das Verfahren stützt sich darauf, dass sich ein im Verhältnis zur Textlänge relativ kurzes Schlüsselwort ständig wiederholt. Der Friedman-Test ist ein statistisches Verfahren, dass auf der Wahrscheinlichkeit basiert, dass zwei zufällig aus einem Text gewählte Zeichen gleich sind.

3.1.3 Vigenere

Kasiski-Test

sucht nach Wiederholungen mit mindestens drei Zeichen

Beispiel

TFNMJ BDCRI TVVAF SRCJI FDPIN NNMKA PELIW TTPYQ
FUIWJ SJBIX ULMGP XRZWN FDQXO PICRS ALAER NVVKJ
HRVKJ OJQIM BKBIS TZKLZ FSMVW PSWXJ SLVXJ SYIPY
FERSW VEVLN FCBHF TDMRX DYTMH IVOIM JIVJZ FIMMS
FESSR QCQDN FIBIS DFUTZ UVZWT GZMAF SJQGM OZKLY
TFAMH IKMVT CJQII BQCWY JDUXJ FZVQJ OJKLR VJAXJ
EFKLR FYZWJ JEIPX FZVIR BJKLN OV

Der Kasiski-Test geht auf den preußischen Infanteriemajor Friedrich Wilhelm Kasiski (1805 bis 1881) zurück. Erfunden wurde dieser Test zwar 1854 von dem englischen Mathematiker Charles Babbage (1792 bis 1871), jedoch hat Babbage seinen Test nie veröffentlicht. Beim Kasiski-Test wird der Geheimtext nach Wiederholungen von Zeichenfolgen mit mindestens drei Zeichen untersucht und deren Abstand gemessen. Je länger die gefundenen Zeichenfolgen, desto größer die Wahrscheinlichkeit, dass der Abstand ein Vielfaches der Schlüssellänge ist. Wiederholt sich eine Zeichenfolge im Klartext mit einem Abstand als Vielfaches der Schlüssellänge, so wird die Wiederholung (bei einer Vigenère-Chiffre) gleich codiert. Es kann natürlich sein, dass im Chiffretext Wiederholungen auftreten, die rein zufällig sind und deren Abstand nicht das Vielfache der Schlüssellänge beträgt. Die Wahrscheinlichkeit für zufällige Wiederholungen ist aber viel kleiner als für Wiederholungen des Vielfachen der Schlüssellänge.

3.1.3 Vigenere

Vermutung:

Schlüssellänge 5 oder 2

Zeichenfolge	Abstand	Primfaktoren
AFS	175	5*5*7
VKJ	5	5
JOJ	145	5*29
JQI	125	5*5*5
BIS	80	2*2*2*2*5
ZKL	100	2*2*5*5
XJS	5	5
MHI	60	2*2*3*5
FZV	30	2*3*5
JKL	30	2*3*5
KLAR	10	2*5

Die Tabelle der Folie zeigt die gefundenen Zeichenketten und ihren Abstand. Man zerlegt die gefundenen Abstände in ihre Primfaktoren und vergleicht alle Faktoren für alle gefundenen Zeichenketten. Da der gesamte Text mit demselben Schlüsselwort verschlüsselt wurde, sucht man nun nach einem Faktor, der bei möglichst vielen Zeichenketten auftritt. In unserem Beispiel treten die Faktoren 5 und 2 häufig auf. Da ein Schlüssel der Länge 2 wenig wahrscheinlich ist (ein so kurzes Schlüsselwort verschleiert die Buchstabenhäufigkeit der natürlichen Sprache nur unzureichend), vermuten wir für die Länge des Schlüssels den Wert 5.

3.1.3 Vigenere

Friedman-Test

Untersucht, mit welcher Wahrscheinlichkeit zwei willkürlich aus einem Text herausgegriffene Buchstaben mit beliebigen Abstand gleich sind.

Koinzidenzindex bezeichnet diese Wahrscheinlichkeit

$$I = \frac{\sum_{i=1}^{26} n_i(n_i - 1)}{n(n-1)}$$

n = Anzahl der Zeichen im Text,
n_i = Anzahl des Zeichens i (z.B. i=A, B, C etc.)

Ein weiterer Test zum Bestimmen der Schlüssellänge ist der Friedman-Test. Der Friedman-Test wurde 1925 vom amerikanischen Kryptologen William Friedman (1891 bis 1969) entwickelt. Beim Friedman-Test wird untersucht, mit welcher Wahrscheinlichkeit zwei willkürlich aus einem Text herausgegriffene Buchstaben gleich sind. Der *Koinzidenzindex* ist diese Wahrscheinlichkeit, dass zwei zufällig gewählte Buchstaben mit beliebeigen Abstand gleich sind.

3.1.3 Vigenere

Wahrscheinlichkeit, zwei gleiche Buchstaben n_i aus Text der Länge n zu ziehen.

$$\frac{n_i(n_i-1)}{n(n-1)}$$

Wahrscheinlichkeit, zwei gleiche Buchstaben (a oder b oder c ...) aus Text der Länge n zu ziehen.

$$\frac{\sum_{i=1}^{26} n_i(n_i-1)}{n(n-1)}$$

Wie kommt man auf diese Formel? Sei n die Anzahl der Zeichen im Text, n_a die Anzahl des Zeichens a, n_b die Anzahl des Zeichens b etc. Dann ist die Wahrscheinlichkeit zwei Zeichen a zu ziehen $n_a(n_a -1)/n(n-1)$. Analog für die restlichen Zeichen b, c,…,z. Die Wahrscheinlichkeit zwei gleiche Buchstaben (zweimal a oder zweimal b oder ….) aus dem Text der Länge n ziehen ist dann die Anzahl der möglichen Buchstabenpaare geteilt durch die Anzahl der möglichen (auch unterschiedlichen) Paare.

3.1.3 Vigenere

Koinzidenzindex für
- zufällig generierte Texte I_r=0,03846
- deutsche Texte I_d etwa 0,0762
- englische Texte I_e etwa 0,066

Test-Idee:
1. Berechne Koinzidenzindex I für den Geheimtext
2. Wenn I ungefähr I_d dann monoalphabetisch,
   falls $I < I_d$ dann polyalphabetisch
   (im Beispiel I = 0,043423)
3. Bestimme die Schlüssellänge k

$$k = \frac{(I_d - I_r)n}{(n-1)I - I_r n + I_d}$$

Die Idee des Tests ist nun die folgende: Man berechnet zunächst den Koinzidenzindex. Wenn dieser (bei deutschen Texten) ungefähr 0,0762 beträgt, handelt es sich mit großer Wahrscheinlichkeit um eine monoalphabetische Verschlüsselung. Wenn der Index deutlich kleiner ist, dann ist der Text polyalphabetisch chiffriert. Der Wert wird kleiner, da sich die Buchstabenverteilung bei einer polyalphabetischen Verschlüsselung immer mehr einem zufälligen Alphabet (mit Index 0,03846) annähert. Im Beispiel des Geheimtextes der Kasiski-Test-Folie beträgt der Index 0,043423, dass heißt die Verteilung der Buchstaben ist schon „zufälliger“ als in allgemeinen deutschen Texten. Da der Wert nah an I_r ist, handelt es sich höchstwahrscheinlich um eine polyalphabetische Verschlüsselung.

3.1.3 Vigenere

- Teile den Geheimtext in k Zeilen der Länge n/k.
- Wähle Buchstaben x aus Geheimtext, dies zeichnet eine Zeile z aus.
- Wähle zweiten Buchstaben y
  - Kommt aus selber Zeile z und y = x

$$\frac{n/k - 1}{n - 1} I_d$$

  - Kommt aus anderer Zeile z‘ und y = x

$$\frac{n - n/k}{n - 1} I_r$$

Wie kommt man auf die Formel für die Schlüssellänge k? Wir nehmen einen Schlüssel der Länge k an und teilen den Geheimtext in k Zeilen auf, so dass in jeder Zeile alle Geheimtextbuchstaben mit demsleben Buchstaben des Schlüsselwortes verschlüsselt wurden. Die Zeilen haben dann die Länge n/k. Wähle nun einen Buchstaben aus dem Geheimtext. Dies bestimmt eine Zeile, in der der Buchstabe enthalten ist.

1. Die Wahrscheinlichkeit, dass der nächste gezogene Buchstabe ebenfalls in dieser Zeile ist, beträgt ((n/k)-1)/(n-1). Da dann beide Buchstaben aus einem Alphabet stammen (d.h. monoalphabetisch), kann man den Index I_d für normale deutsche Texte nehmen.
2. Die Wahrscheinlichkeit, dass der nächste gezogene Buchstabe aus einer anderen Zeile stammt, beträgt (n-(n/k))/n-1. Da in diesem Fall beide Buchstaben aus unterschiedlichen Alphabeten stammen, ist die Wahrscheinlichkeit, dass beide Buchstaben gleich sind dann etwa der Wahrscheinlichkeit für zufällig generierte Texte I_r.

Diese beiden Wahrscheinlichkeiten werden dann addiert zur Wahrscheinlichkeit I und die Formel nach der Schlüssellänge k umgestellt.

3.1.3 Vigenere

Friedman-Test ergibt für das Beispiel I = 0,043423

TFNMJ BDCRI TVVAF SRCJI FDPIN NNMKA PELIW TTPYQ
FUIWJ SJBIX ULMGP XRZWN FDQXO PICRS ALAER NVVKJ
HRVKJ OJQIM BKBIS TZKLZ FSMVW PSWXJ SLVXJ SYIPY
FERSW VEVLN FCBHF TDMRX DYTMH IVOIM JIVJZ FIMMS
FESSR QCQDN FIBIS DFUTZ UVZWT GZMAF SJQGM OZKLY
TFAMH IKMVT CJQII BQCWY JDUXJ FZVQJ OJKLR VJAXJ
EFKLR FYZWJ JEIPX FZVIR BJKLN OV

$$k = \frac{(I_d - I_r)n}{(n-1)I - I_r n + I_d} = \frac{(0{,}0762 - 0{,}03846)267}{266 \cdot 0{,}0434 - 0{,}03846 \cdot 267 + 0{,}0762} \approx 7{,}5$$

Für die Schlüssellänge des schon beim Kasisiki-Test benutzten Beispiels ergibt der Freidman-Test einen Wert von 7,5. Die Anzahl der Zeichen des Beispiel-Geheimtextes beträgt 267, der Koinzidenzindex hat einen Wert von 0,043423. Zusammen mit den Überlegungen des Kasiski-Tests, bei dem eine Schlüssellänge von 5 bzw. 2 vermutet wurde, zeigt sich, dass die Schlüssellänge 5 sein wird, da der Wert 7,5 viel näher an 5 als an 2 liegt.

3.1.3 Vigenere

Nachdem die Schlüssellänge bekannt ist, kann wie bei monoalphabetischer Verschlüsselung entschlüsselt werden.

TFNMJ BDCRI TVVAF SRCJI FDPIN NNMKA PELIW TTPYQ
FUIWJ SJBIX ULMGP XRZWN FDQXO PICRS ALAER NVVKJ
HRVKJ OJQIM BKBIS TZKLZ FSMVW PSWXJ SLVXJ SYIPY
FERSW VEVLN FCBHF TDMRX DYTMH IVOIM JIVJZ FIMMS
FESSR QCQDN FIBIS DFUTZ UVZWT GZMAF SJQGM OZKLY
TFAMH IKMVT CJQII BQCWY JDUXJ FZVQJ OJKLR VJAXJ
EFKLR FYZWJ JEIPX FZVIR BJKLN OV

1. TBTSFNPTF....	verschlüsselt mit Alphabet B
2. FBVRDBETUJLR.....	verschlüsselt mit Alphabet R
3. NCVCPMLP....	verschlüsselt mit Alphabet I
4. MRAJIKY.....	verschlüsselt mit Alphabet E
5. JIFINAWQJXP.....	verschlüsselt mit Alphabet F

Ist die Schlüssellänge bekannt, kann die komplette Nachricht genauso einfach geknackt werden wie eine monoalphabetische Verschlüsselung. Sei beispielsweise die Schlüssellänge 5, so wird jeder fünfte Buchstabe des Geheimtextes mit demselben Buchstaben des Schlüsselwortes (d.h. mit demselben Alphabet) verschlüsselt. Daher kann man hierfür wie bei der monoalphabetischen Verschlüsselung den Geheimtext mittels Häufigkeitsanalyse entschlüsseln. Führt man die Häufigkeitsanalyse für das Beispiel durch, erhält man das Schlüsselwort BRIEF....

3.1.3 Vigenere

Der ermittelte Klartext ist dann...

SOFIE AMUNDSEN WAR AUF DEM HEIMWEG VON DER SCHULE. DAS ERSTE STUECK WAR SIE MIT JORUNN ZUSAMMEN GEGANGEN. SIE HATTEN SICH UEBER ROBOTER UNTERHALTEN. .....

...und der vollständig entschlüsselte Geheimtext ergibt einen Auszug aus Jostein Gaarders Buch *Sofies Welt*.

3.1.3 Vigenere

- Vigenere-Schwäche ist das sich wiederholende Schlüsselwort
- Schlüssel mit bestimmter Ordnung bzw. mit erkennbaren Wörtern, der so lang wie der Klartext ist, garantiert auch keine absolute Sicherheit
- Schlüssel zufällig generieren und für jede Nachricht einen neuen Schlüssel benutzen

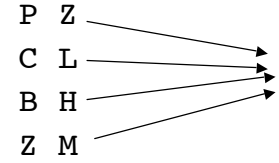
Die Schwäche der Vigenere-Verschlüsselung ist ihr zyklischer Charakter, der durch die Wiederholung des Schlüsselwortes entsteht. Um Zyklen zu verhindern, verlängert man den Schlüsseltext auf die Länge des Geheimtextes. Dann sind die Verfahren von Kasiski und Friedman nicht mehr anwendbar. Ein Schlüssel, der so lang wie der zu verschlüsselnde Text ist, garantiert jedoch noch keine absolute Sicherheit (z.B. können sinnvolle Wörter im Schlüsselwort als Angriffspunkte dienen). Um keine Ordnung oder erkennbare Wörter im Schlüsseltext zu haben, kann der Schlüsseltext zufällig generiert werden. Wird der Schlüssel dann nur einmal benutzt, gibt es auch keine Abhängigkeiten zwischen verschiedenen Nachrichten, die für eine Attacke ausgenutzt werden könnten. Erzeugte man nicht für jede Nachricht einen neuen Schlüssel, könnte man mehrere Nachrichten abfangen und hätte wieder eine Wiederholung des Schlüssels für eine Menge von Nachrichten.

3.1.4 Perfekte Verschlüsselung

One-Time-Pad

Schlüssel ist **zufällige** Zeichenkette, die genauso lang wie der Klartext ist und nur einmal verwendet wird

Klartext	Schlüssel	Geheimtext
T I M	X P Z	
E V A	M C L	Q X L
U W E	W B H	
X Y Z	T Z M	



Durchsuchung des Schlüsselraums macht hier keinen Sinn!

Das *One-Time-Pad* beruht auf einem Zufallsschlüssel, der dieselbe Länge hat wie der zu verschlüsselnde Text selbst und jeder Schlüssel darf nur ein einziges Mal verwendet werden. Dieses Verschlüsselungsverfahren ist beweisbar sicher, da ein Angreifer selbst bei einem Durchprobieren aller möglichen Schlüssel nicht die richtige Nachricht ermitteln kann. Er würde zwar die richtige Botschaft finden, zugleich findet er aber auch alle falschen, aber sinnvollen Texte. Der Kryptoanalytiker kann nicht zwischen der richtigen und allen anderen, die ebenfalls Sinn machen, unterscheiden.

3.1.4 Perfekte Verschlüsselung

Beweisbar sicher durch Zufälligkeit des Schlüssels (keine Ordnung, keine Muster)

Probleme in der Praxis

- Generierung der großen Mengen von „echten“ Zufallsschlüsseln
- Verteilung der Schlüssel
- Koordination, wann welcher Schlüssel benutzt wird

Die Sicherheit des One-Time-Pad beruht auf der Zufälligkeit des Schlüssels, der keine Muster, keine innere Ordnung, nichts enthält an dem sich ein Angreifer orientieren könnte. Das One-Time-Pad ist mathematisch beweisbar sicher, d.h. eine mit dem One-Time-Pad verschlüsselte Nachricht kann nicht geknackt werden. Das One-Time-Pad-Verfahren weist allerdings einige praktische Nachteile auf: das Generieren von großen Mengen von Zufallsschlüsseln ist nicht einfach, die Schlüssel müssen verteilt und die Schlüsselnutzung koordiniert werden. Aus diesen Gründen wird das One-Time-Pad in der Praxis so gut wie nicht eingesetzt.

3.1.5 Block- und Stromchiffren

Verschlüsselung im Rechner:

Klartext sind Folgen aus Binärzahlen. Er wird in Einheiten (Zeichen oder Blöcke) aufgeteilt.

Stromverschlüsselung *(stream cipher):*

kleine Klartexteinheiten (Bytes, Bits)
einfache Verknüpfungsoperationen zur Ver- und Entschlüsselung (z.B. XOR)

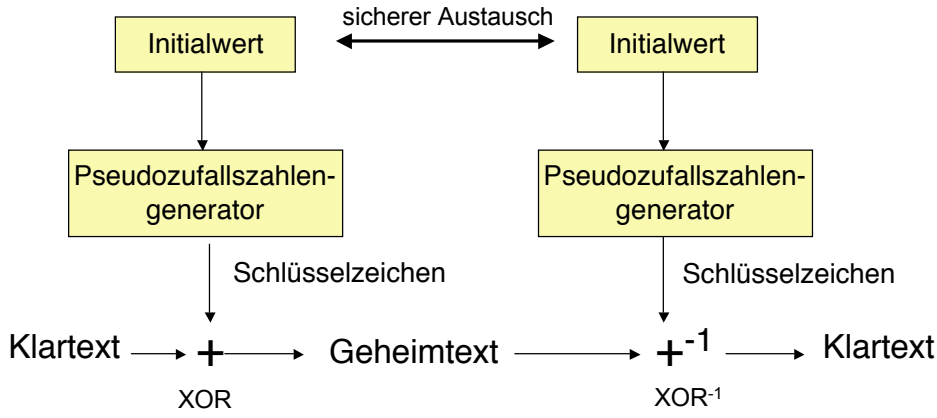
Blockverschlüsselung *(block cipher):*

große Klartexteinheiten fester Länge (z.B. Blöcke von 64 Bits)
werden separat mit derselben Funktion *sicher* verschlüsselt
Vorteile: schnell *und* sicher, Direktzugriff möglich
Nachteil: Block-Wiederholungen im Geheimtext sichtbar

Mit dem Aufkommen von Rechnern sind die Anforderungen an Verschlüsselungsverfahren extrem gestiegen, da die Analysemöglichkeiten durch die Geschwindig- und Leistungsfähigkeit moderner Rechner stark gestiegen sind. Ein weiterer Unterschied zu den bisherigen Verfahren ist, dass Rechner Folgen aus Binärzahlen und keine Buchstaben eines Alphabets verarbeiten. Zur Verschlüsselung werden die Binärfolgen des Klartextes in Einheiten fester Länge eingeteilt. Man unterscheidet hierbei Block- und Stromchiffren. Stromchiffren sind sequentielle Chiffren, die eine Folge von kleinen Klartexteinheiten mit einer variierenden Funktion verschlüsseln. Gängige Einheiten sind 1 Bit oder 1 Byte. Blockchiffren verarbeiten Eingabeblöcke fester Länge, wobei jeder Block mit der gleichen Funktion verschlüsselt wird. Eine typische Blockgröße beträgt hier 64-Bit.

3.1.5 Block- und Stromchiffren

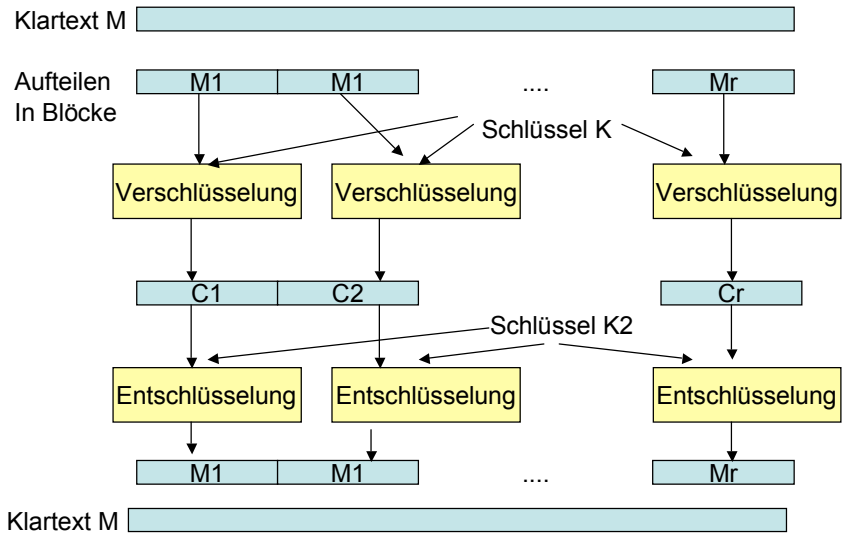
Stromverschlüsselung (z.B. A5-Algorithmus im Mobilfunk)



Die Arbeitsweise einer Stromchiffre ist die folgende: Ein Pseudozufallszahlengenerator generiert durch einen deterministischen Prozess unter Verwendung eines Initialwertes eine Pseudozufallszahlenfolge. Zum Ver- und Entschlüsseln muss derselbe Generator mit demselben Initialwert zur Erzeugung der Pseudozufallszahlenfolge verwendet werden. Im Gegensatz zu Blockchiffren werden bei Stromchiffren in der Regel sehr einfache Verknüpfungsoperationen verwendet (z.B. XOR zur Verschlüsselung und das negierte XOR zur Entschlüsselung). Für die kryptographische Sicherheit ist der Zufallszahlengenerator ausschlaggebend. Beispiel einer Stromchiffre ist der europaweit standardisierte A5-Algorithmus, der im GSM-Standard für mobile Telephonie verwendet wird, um Gespräche zu verschlüsseln. Stromchiffren verwendet man meist, wenn der vollständige Klartext nicht von Anfang an vorliegt und es nicht praktikabel ist, mit der Übertragung solange zu warten, bis genug Zeichen für einen ganzen Block vorhanden sind (z.B. verschlüsselte Sprachübertragung).

3.1.5 Block- und Stromchiffren

Blockverschlüsselung (z.B. DES)



Blockchiffren sind dadurch charakterisierbar, dass sie Eingabeblöcke fester Länge verarbeiten. Jeder Block wird dann mit derselben Verschlüsselungsfunktion verschlüsselt. Eine typische Blockgröße beträgt 64-Bit. Für die Entschlüsselung werden die Geheimtextblöcke wiederum separat mit demselben Entschlüsselungsschlüssel entschlüsselt. Blockchiffren verwendet man meist dann, wenn bereits vor Beginn der Verschlüsselung der Klartext vollständig vorliegt (E-Mail, Festplatte).

3.1.5 Block- und Stromchiffren

Elementare Blockverschlüsselung (*electronic code book*, ECB) hat Nachteile

- Klartextblöcke werden mit demselben Schlüssel verschlüsselt
- Wiederholungen von Klartextblöcken im Geheimtext erkennbar
- Unabhängige Verarbeitung der Blöcke (erleichtert entfernen, modifizieren, einspielen von Blöcken)

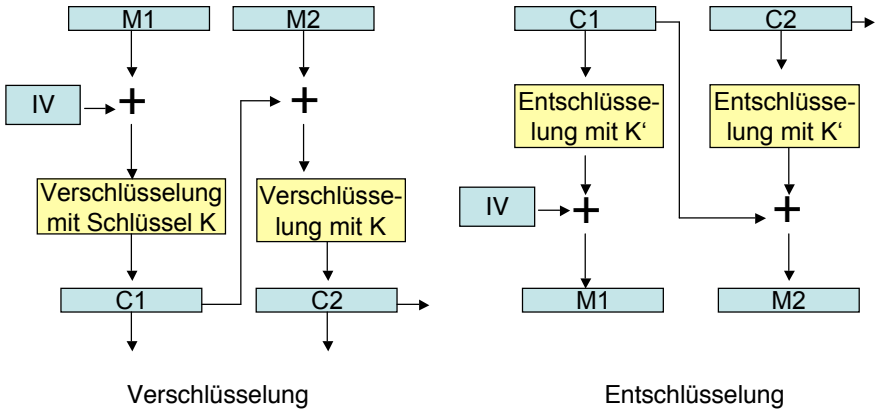
Abhilfe: Schlüssel laufend durch den Text selbst modifizieren durch Verkettung, z.B.

- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)

Die auf der vorigen Folie vorgestellte Blockchiffrierung ist der einfachste Fall und wird als *Electronic Codebook* (*ECB*) bezeichnet. Ein Nachteil dieses Verfahrens ist, dass die Klartextblöcke alle mit demselben Schlüssel verschlüsselt werden. Somit werden zwei identische Klartextblöcke auf identische Schlüsseltextblöcke abgebildet. Bei längeren Texten vereinfacht das einen Angriff (ähnlich wie beim Vigenere-Verfahren). ECB eignet sich daher nur für kurze Nachrichten (z.B. Übertragung eines Schlüssels). Außerdem basiert der ECB auf der unabhängigen Verarbeitung der einzelnen Blöcke, wodurch ein Angreifer leicht einen Block entfernen, einen Block modifizieren oder einen eigenen Block einspielen kann. Abhilfe verschaffen Verfahren der Verkettung, bei denen der vorangegangene Block die Verschlüsselung des nachfolgenden Blocks beeinflusst: *Cipher Block chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB)*.

3.1.5 Block- und Stromchiffren

Cipher Block Chaining (CBC) benutzt bei der Verschlüsselung eines Klartextblocks den vorangegangenen Geheimtextblock



Der CBC-Modus führt eine Verkettung der Blöcke durch, indem jeder Klartextblock vor der Verschlüsselung mit dem vorhergehenden verschlüsselten Block XOR-verknüpft wird. Um auch den ersten Klartextblock mit einem anderen Block zu verketten, nimmt man einen Initialisierungsvektor (IV). Entschlüsselt wird indem der entschlüsselte Block mit dem vorangegangenen Geheimtextblock XOR-verknüpft wird. Es wird derselbe Initialisierungsvektor für den ersten Block verwendet. Im CBC-Modus ist jeder Geheimtextblock vom dazugehörigen Klartextblock und dem vorangegangenen Geheimtextblock abhängig.

3.1.5 Block- und Stromchiffren

Es ergibt sich beim Ver- und wieder Entschlüsseln derselbe Klartext

Verschlüsselung des Klartextblocks M_i

$C_i = E(K, M_i \text{ xor } C_{i-1})$

Entschlüsselung des Geheimtextblocks C_i

$$\begin{aligned} M_i &= D(K', C_i) \text{ xor } C_{i-1} \\ &= D(K', E(K, M_i \text{ xor } C_{i-1})) \text{ xor } C_{i-1} \\ &= M_i \text{ xor } C_{i-1} \text{ xor } C_{i-1} \\ &= M_i \end{aligned}$$

Das beim CBC-Modus die Entschlüsselung wieder die Verschlüsselung ergibt, zeigt die obige Folie.

3.1.5 Block- und Stromchiffren

Output Feedback (OFB) und Cipher Feedback (CFB)

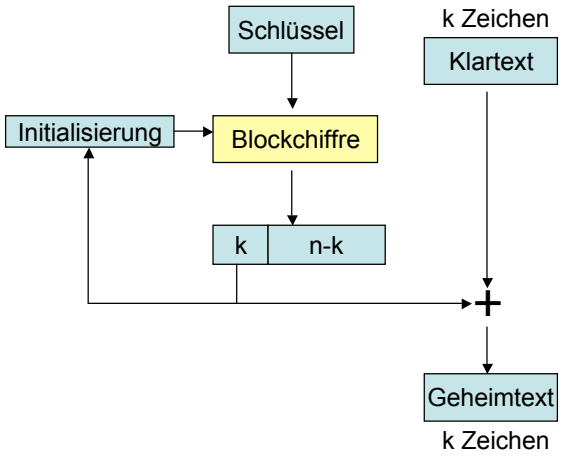
Benutzen Blockverschlüsselung für eine Stromverschlüsselung mit Rückkopplung

Blockverschlüsselung dient als Pseudozufallsfolgengenerator für den Schlüsselstrom (zum Ver- und Entschlüsseln)

Der Cipher-Feedback Modus (CFB) bzw. der Output Feedback Modus (OFB) nutzen die Blockchiffre als Pseudozufallsfolgengenerator für eine Stromchiffre. Hierbei dient die Blockchiffre sowohl beim Ver- und Entschlüsseln nur zur Erzeugung des Schlüsselstroms. Der OFB und der CFB unterscheiden sich, wie die Eingabeblöcke erzeugt werden, die die Blockchiffre zur Schlüsselgenerierung benötigt. Beim Output Feedback ist dies der Ausgabeblock der Blockchiffre, beim Cipher Feedback der erzeugte Geheimtextblock.

3.1.5 Block- und Stromchiffren

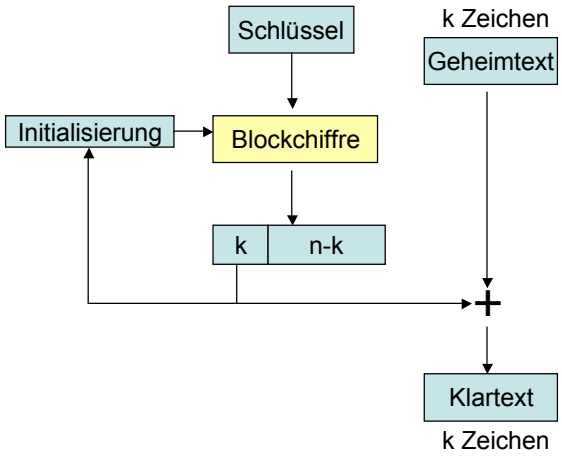
Output Feedback Modus (Verschlüsselung)



Sollen beim OFB-Modus jeweils k-Bit des Klartextes verschlüsselt werden, so wird zunächst eine Verschlüsselungsoperation der Blockchiffre, die n-Bit Blöcke verschlüsselt, durchgeführt. k Bits des resultierenden Ausgabeblocks werden mit den k Klartextbits XOR-verknüpft. Gleichzeitig ersetzen sie auch k Bits des Eingabeblocks für die Blockchiffre, der zu Beginn der Verschlüsselung den Initialisierungsvektor enthält. Die Blockchiffre wird nur zum Verschlüsseln, nie zum Entschlüsseln genutzt, da die Blockchiffre nur die Pseudozufallsfolge generiert. Zum Ver- und Entschlüsseln wird nur derselbe Schlüssel und derselbe Initialisierungsvektor benötigt.

3.1.5 Block- und Stromchiffren

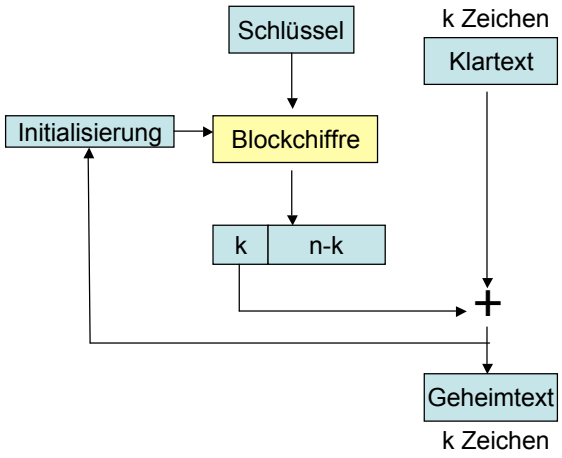
Output Feedback Modus (Entschlüsselung)



Zur Entschlüsselung wird von der Blockchiffre wieder der Zufallsfolgenstrom erzeugt. Damit diese gleich derjenigen bei der Verschlüsselung ist, muss derselbe Schlüssel und derselbe Initialisierungsvektor benutzt werden.

3.1.5 Block- und Stromchiffren

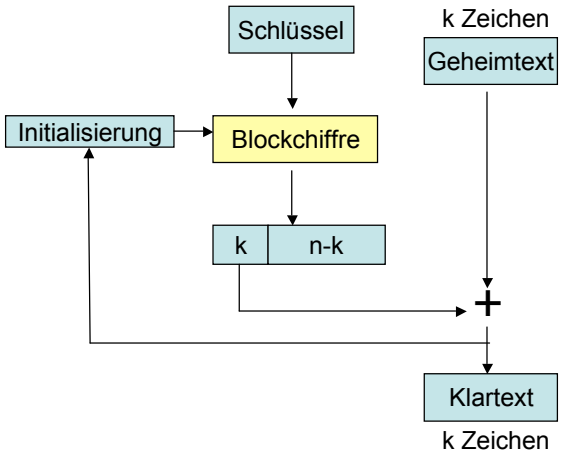
Cipher Feedback Modus (Verschlüsselung)



Der CFB-Modus arbeitet ähnlich dem OFB-Modus. Er ersetzt die k-Bits des Eingabeblocks für die Blockchiffre jedoch nicht aus dem Ausgabeteil der Blockchiffre, sondern aus dem erzeugten Geheimtext.

3.1.5 Block- und Stromchiffren

Cipher Feedback Modus (Entschlüsselung)



Die Entschlüsselung nimmt dann die k Bits für die Eingabe der Blockchiffre aus dem Klartext.

3.1.6 Data Encryption Standard (DES)

1973-77 Zwei Ausschreibungen, ein geeigneter Kandidat (LUCIFER von IBM) wurde nach Überarbeitung DES

Ab 1977 Kritik an der Schlüssellänge, 1994 wurde DES Schlüssel in 50 Tagen geknackt, 1998 in 56 Stunden, 2001 in 22 Stunden

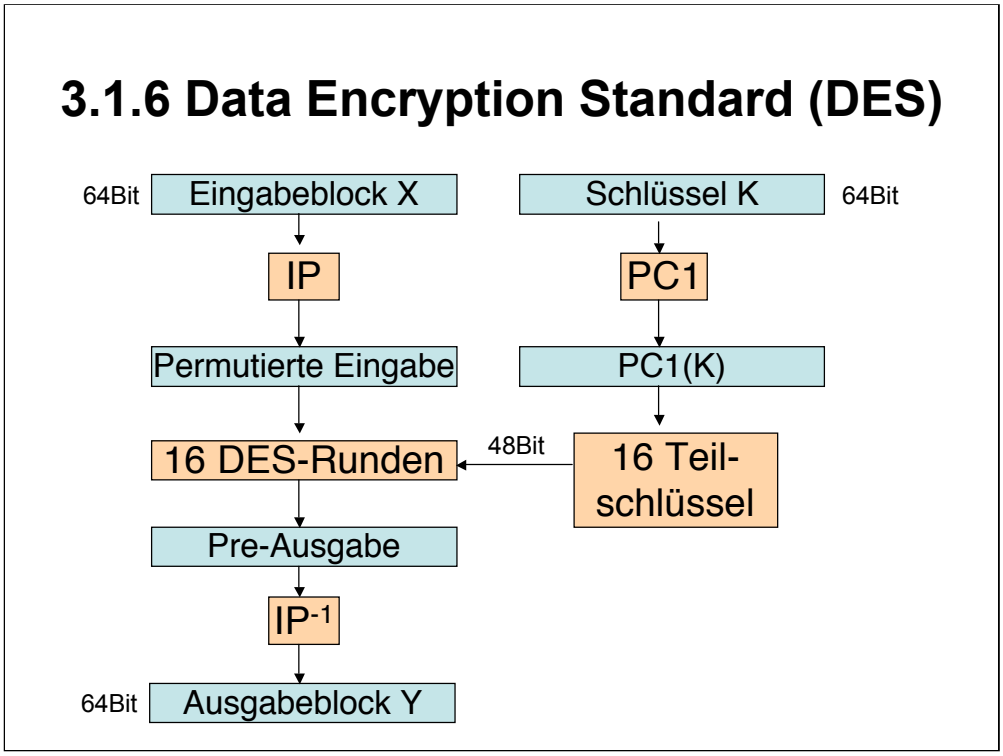
Der Klartext (Bitfolge beliebiger Art) wird in 64 bit Blöcken zerlegt, die in 16 Phasen gesteuert durch einen 64 Bit (56 + 8 Paritätsbits) langen Schlüssel durcheinander gewürfelt werden.

Schlüsselraum von 2^{56} verschiedenen Schlüsseln

Das bekannteste symmetrische Verschlüsselungsverfahren ist der DES-Algorithmus (Data Encryption Standard). 1973 erfolgte eine Ausschreibung im Federal Register, der nur unzureichende Ergebnisse brachte. Nach der zweiten Ausschreibung 1974 reichte IBM den 128-Bit LUCIFER Algorithmus ein. Der modifizierte LUCIFER-Algorithmus wurde 1975 von der NSA veröffentlicht. Eine wesentliche Modifikation war die Reduzierung der Schlüssellänge auf 56-Bit. 1977 wurde dieser Algorithmus als Data Encryption Standard (DES) durch das National Bureau of Standards zum US-Verschlüsselungsstandard genormt. Schon ab 1977 gab es vielfache Kritik an der zu kurzen Schlüssellänge. Deutlich wurde diese Schwäche durch Angriffe, die den Schlüsselraum systematisch durchsuchen, z.B. 1994 als ein DES-Schlüssel von 12 Workstations in 50 Tagen ermittelt wurde. 1998 wurde ein Spezialcomputer vorgestellt, der einen DES-Schlüssel in 56 Stunden entschlüsseln konnte. Durch paralleles Internet-Computing gelang 2001 ein Angriff in nur 22 Stunden.

Der DES verschlüsselt einen 64-Bit Eingabeblock mit einem Schlüssel von 64-Bit, bei dem 56-Bit frei wählbar sind, die restlichen 8 Bit sind Paritätsbits. Damit besitzt der DES einen Schlüsselraum von 2^{56} unterschiedlichen Schlüsseln.

Der effektivste Angriff besteht aus dem Durchprobieren (fast) aller möglichen Schlüssel, bis der richtige gefunden wird (brute-force-attack). Aufgrund der relativ kurzen Schlüssellänge von effektiv 56 Bits (64 Bits, die allerdings 8 Paritätsbits enthalten), sind in der Vergangenheit schon mehrfach mit dem DES verschlüsselte Nachrichten gebrochen worden, so dass er heute als nur noch bedingt sicher anzusehen ist. Symmetrische Alternativen zum DES sind zum Beispiel die Algorithmen IDEA oder Triple-DES.



Die Funktionsweise des Standard DES besitzt folgendes Grundschema: Der Eingabeblock durchläuft zunächst die Initialpermutation IP, die die permutierte Eingabe erzeugt (siehe nächste Folie). Der permutierte Eingabeblock durchläuft dann 16 identische Chiffrierschritte, die als Runden bezeichnet werden. In jeder dieser Runden wird ein bestimmter Teilschlüssel des symmetrischen Schlüssels verwendet. Diese Teilschlüssel werden aus einem 56-Bit Wert erzeugt, der seinerseits aus dem ursprünglichen 64-Bit Schlüssel durch eine permutierte Auswahl (PC1) entsteht. Das Ergebnis der sechzehn Runden durchläuft abschließend die inverse Initialpermutation IP^{-1}.

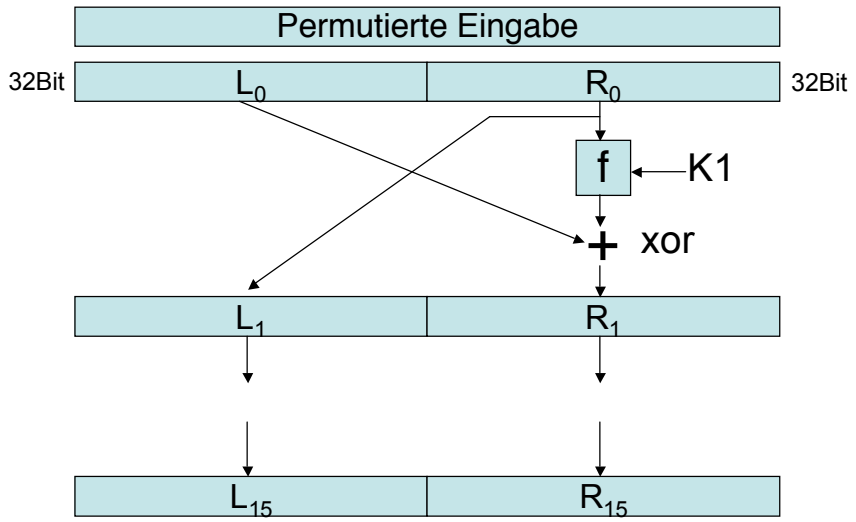
3.1.6 Data Encryption Standard (DES)

Permutationstabelle IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

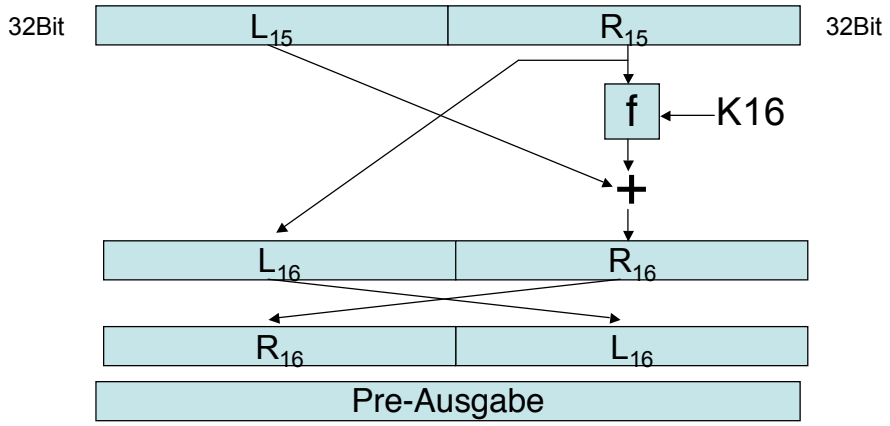
Die Permutationstabelle wird für die Initialpermutation IP verwendet. Liest man die Einträge einer Zeile von links nach rechts, und die Zeilen von oben nach unten, so geben die Einträge die Nummer desjenigen Bits im Eingabeblock an, das an diese Stelle gesetzt wird. Der Eingabeblock X=(x1,x2,...,x64) ergibt somit einen permutierten Eingabeblock IP(X)=(x58,x50,...,x7).

3.1.6 Data Encryption Standard (DES)

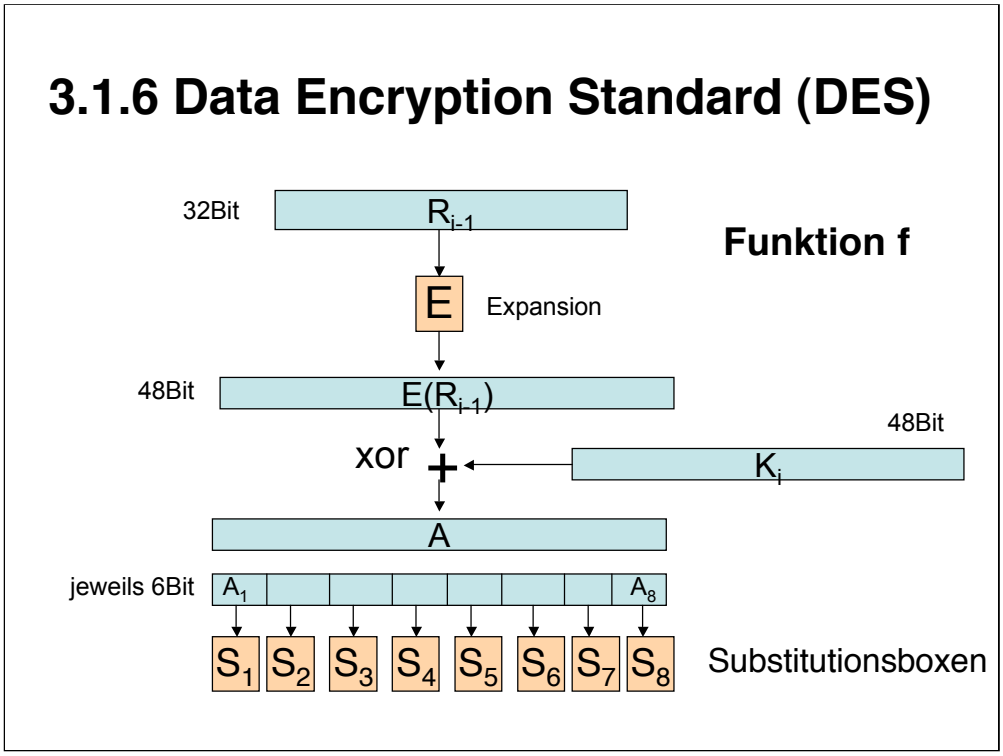


Der permutierte Eingabeblock wird in die Hälften L_0 und R_0 mit je 32 Bit zerlegt. Es gilt für i=1,...,16: $L_i = R_{i-1}$ und $R_i = L_{i-1}$ xor $f(R_{i-1},K_i)$, wobei K_i der i-te Teilschlüssel ist und xor die exclusiv-oder Operation bezeichnet.

3.1.6 Data Encryption Standard (DES)

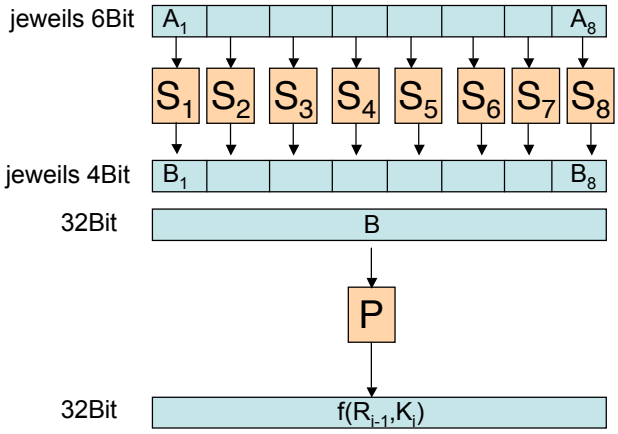


Die beiden Hälften $L_{16}R_{16}$ des Ergebnisses der letzten Runde werden getauscht zu $R_{16}L_{16}$ und ergeben die Pre-Ausgabe.



Die Funktion f bildet den Kern des DES. Ihre wesentlichen Elemente sind die acht Substitutionsboxen (S-Boxen). Mit diesen wird eine schlüsselabhängige Substitution einzelner Teile des Eingangswertes durchgeführt. Bevor der 32-Bit Wert R_{i-1} mit dem 48-Bit Teilschlüssel K_i verknüpft wird, muss R_{i-1} durch eine Expansionsabbildung E auf einen 48-Bit $E(R_{i-1})$ erweitert werden. Die XOR-Verknüpfung von K_i und $E(R_{i-1})$ ergibt einen 48-Bit Wert A, der in acht 6-Bit Blöcke A_1, A_2,...., A_8 zerlegt wird. Jeder Block A_j (j=1...8) dient als Eingabe für die S-Box S_j.

3.1.6 Data Encryption Standard (DES)



Permutation P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Jede S-Box S_j erzeugt einen 4-Bit Ausgabeblock B_j. Die Konkatenation der Blöcke B_1,..., B_8 wird von einer Permutation P permutiert und liefert den Ausgabewert $f(R_{i-1},K_i)$.

3.1.6 Data Encryption Standard (DES)

Beispiel S_3-Box

	0	**1**	**2**	**...**	**13**	**14**	**15**
0	10	0	9	...	4	2	8
1	13	7	0	...	11	15	1
2	13	6	4	...	**10**	14	7
3	1	10	13	...	5	2	12

A_3 = (111010) , 10 = 2= Zeile, 1101=13=Spalte

B_3 = 1010 = 10

Eine S-Box ist eine Matrix, bestehend aus 4 Zeilen und 16 Spalten. Die Substitution wählt aus der S-Box S_j ein Element aus, indem der Spalten- und Zeilenindex aus dem Eingabeblock A_j berechnet wird. Dazu bilden das linke und das rechte Bit von A_j den Zeilenindex und die mittleren 4 Bits den Spaltenindex.

3.1.6 Data Encryption Standard (DES)

Permutationstabelle IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Die Permutation IP^{-1} ist die inverse Operation zur Permutation IP.

3.1.6 Data Encryption Standard (DES)

Berechnung der Teilschlüssel K_i

- *Permuted Choice* PC1 entfernt jedes achte Bit und permutiert die 56 restlichen Bits
- 56-Bit Schlüssel wird in C_0 und D_0 (jeweils 28-Bit) zerlegt
- 16 Runden, in denen C_i und D_i mittels eines zyklichen Links-Shifts um ein oder zwei Bits aus C_{i-1} und D_{i-1} erzeugt werden.
- Aus jedem 56-Bit Wert C_i,D_i wird mittels einer weiteren Permutation ein 48-Bit Schlüssel K_i erzeugt.

Ein 64-Bit Schlüssel durchläuft zunächst eine permutierende Auswahl PC1 (Permuted Choice). Diese entfernt jedes achte Bit und permutiert die verbleibenden 56 Bits. Die ausgelassenen Bits sind als Paritätsbits gedacht, haben aber auf das Verfahren keinen Einfluss. Der entstandene 56-Bit Schlüssel wird in die Hälften C_0 und D_0 zu je 28-Bit aufgespalten. Diese durchlaufen nun 16 Runden, in denen C_i und D_i mittels eines zyklichen Links-Shifts um ein oder zwei Bits aus C_{i-1} und D_{i-1} erzeugt werden. Die Anzahl der Positionen, um die in der jeweiligen Runde geshiftet wird, ist festgelegt. Aus jedem 56-Bit Wert C_i,D_i wird mittels einer weiteren Permutation ein 48-Bit Schlüssel K_i erzeugt.

3.1.6 Data Encryption Standard (DES)

***Beh**.:* Der Chiffrier-Algorithmus *dechiffriert*, wenn die Teilschlüssel *Ki* in *umgekehrter* Reihenfolge verwendet werden.

Beweis

1. Die Permutation IP der Bits des Geheimtextes liefert $R_{16}L_{16}$.

Der DES wird sowohl zum Ver- und Entschlüsseln verwendet. Die zur Entschlüsselung benötigten Teilschlüssel werden in umgekehrter Reihenfolge eingesetzt, d.h. anstelle der Reihenfolge K_1, K_2,...,K_{16} wird bei der Entschlüsselung die Reihenfolge K_{16}, K_{15},...,K_1 verwendet. Es wird also in der Runde i der Teilschlüssel K_{17-i} verwendet. Das die Entschlüsselung wieder den Klartext zeigt, demonstriert der Beweis auf der Folie.

2. Ein Schritt *i, i=16,..,1*, erzeugt aus $R_i\, L_i$ den Wert $l\, r$ mit

   $l = L_i = R_{i-1}$ und

   $r = R_i \text{ xor } f(L_i, K_i)$
   $= (L_{i-1} \text{ xor } f(R_{i-1}, K_i)) \text{ xor } f(L_i, K_i)$
   $= L_{i-1}$ wegen $L_i = R_{i-1}$

   also den Wert $R_{i-1}\, L_{i-1}$.

3. Die Vertauschung der $R_0\, L_0$ liefert $L_0\, R_0$, und die Rückpermutation liefert den Klartext.

3.1.6 Data Encryption Standard (DES)

DES ist ein kombiniertes Transpositions-/Substitutions-verfahren

DES besitzt die wünschenswerte Eigenschaft, dass minimale Änderungen des Schlüssels bzw. minimale Änderungen des Klartextes erhebliche Auswirkung auf den verschlüsselten Text besitzen

DES ist gut in Hardware implementierbar – Chiffrierraten in der Größenordnung von 100 MB/sec werden erreicht.

DES wird z.B. zur Passwortverschlüsselung in Unix verwendet (siehe Kapitel Zugangskontrolle)

Der DES kombiniert Transposition (d.h. Permutation) und Substitution (in den Subsititutions-Boxen). Der Algorithmus hat die wünschenswerte Eigenschaft, dass eine minimale Änderung des Schlüssels bzw. des Klartextes erhebliche Auswirkung auf den Geheimtext haben. Diese Eigenschaft erschwert eine Kryptoanalyse durch lineare Approximation des Verhaltens des Verschlüsselungverfahrens. Desweiteren ist der DES gut in Hardware implementierbar und wird beispielsweise bei der Passwortverschlüsselung in UNIX-Systemen eingesetzt.

3.1.6 Data Encryption Standard (DES)

Schlüsselgröße von 56 Bits ist heutzutage zu klein.
Dennoch viel sicherer als Substitutionsverfahren, weil Beziehung zwischen Klartext und Geheimtext ungleich komplizierter.

Tripel-DES (drei Schlüssel)

Verschlüsselung $DES(DES^{-1}(DES(X,K1),K2),K3)$

Entschlüsselung $DES^{-1}(DES(DES^{-1}(X,K3),K2,K1)$

effektive Schlüssellänge 108 Bit

Wie schon erwähnt, ist die Schlüsselgröße von 56-But heutzutage zu klein, um vor Angriffen zu schützen. Fortschritte lassen sich durch die Technik des Tripel-DES mit drei Schlüsseln K1, K2 und K3 erzielen. Die Ver- und Entschlüsselung erfolgt durch die auf der Folie angegebene Reihenfolge. Der zu leistende Vreschlüsselungsaufwand wird beim Triple-DES verdreifacht, die effektive Schlüssellänge ist jedoch nur 108 bit anstelle von 168 Bit. Die zweifache Verschlüsselung ergibt keine große Verbesserung, da aufgrund einer Meet-in-the-middle Attacke nur $2*2^{56}$ ausprobiert werden müssen (siehe C.Eckart, IT Sicherheit).

3.1.6 Data Encryption Standard (DES)

Meet-in-the-middle Angriff beim Double-DES

Klartext X und zugehöriger Geheimtext $c = DES(K_b, DES(K_a,X))$
liegen vor – gesucht sind K_a und K_b .

Für alle Schlüssel K_i aus der Menge $\{0,...,2^{56}\}$ den einfach verschlüsselten Klartext erstellen:

$$c_i = DES(K_i,x) .$$

Für alle Schlüssel K_j Geheimtext einfach entschlüsseln,

$$y_j = DES^{-1}(Kj,c) ,$$

und jeweils prüfen, ob das y_j unter den c_i vorkommt.

Bei einem solchen Angriff geht man von einem Klartext/Geheimtextpaar aus, welches mit $DES(K_b, DES(K_a,X))$ verschlüsselt ist. Man versucht die verwendeten Schlüssel K_a, K_b zu brechen. Im ersten Schritt wird der Klartext X mit allen möglichen Schlüsseln verschlüsselt. Im nächsten Schritt werden der Geheimtext mit allen möglichen Schlüsseln entschlüsselt und es wird geprüft, ob y_j unter den c_i vorkommt.

3.1.6 Data Encryption Standard (DES)

Falls $y_j = c_i$ für ein bestimmtes j und i,

„*Ki* und *Kj* sind die gesuchten Schlüssel“ vermuten
und mit weiterem Klartext/ Geheimtext-Paar überprüfen.

v Höchstens $2*2^{56}$ Schlüssel werden durchprobiert.

Ist dies der Fall, stellt man die Hypotese auf , dass K_i und K_j die gesuchten Schlüssel sind. Diese Hypothese wird mit weiteren Klartext/Geheimtextblöcken überprüft. Ein zweifach-DES ist mit einem effektiven Schlüsselraum von 2^{57} Schlüsseln nicht wesentlich stärker als der einfache DES.

3.1.7 IDEA

International Data Encryption Algorithm (IDEA),
Lai/Massey 1990

verschlüsselt 64-Bit-Blöcke bei einer Schlüssellänge von 128 Bits,

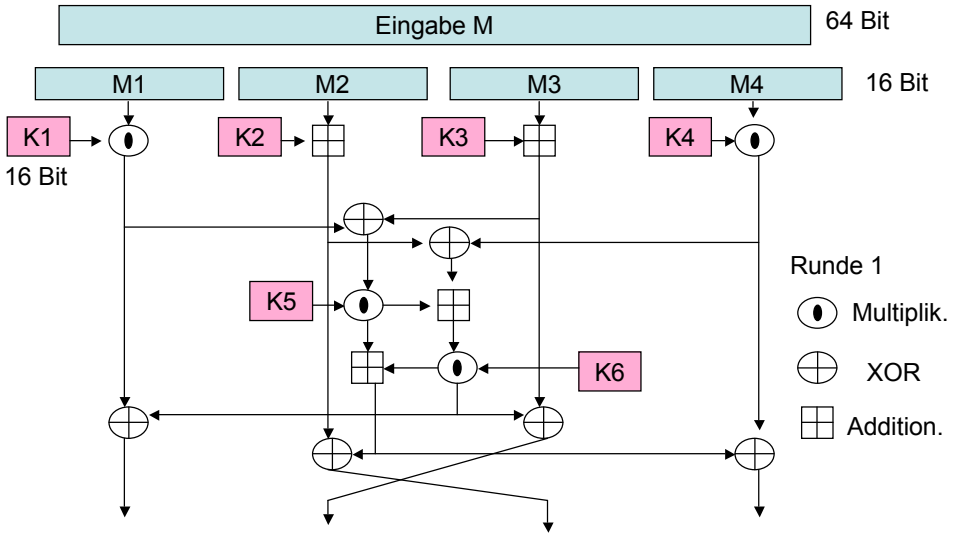
arbeitet mit 8 (statt 16) Runden,

verwendet nur xor, Addition und Multiplikation, d.h. keine Permutation oder Substitution

ist schneller als DES

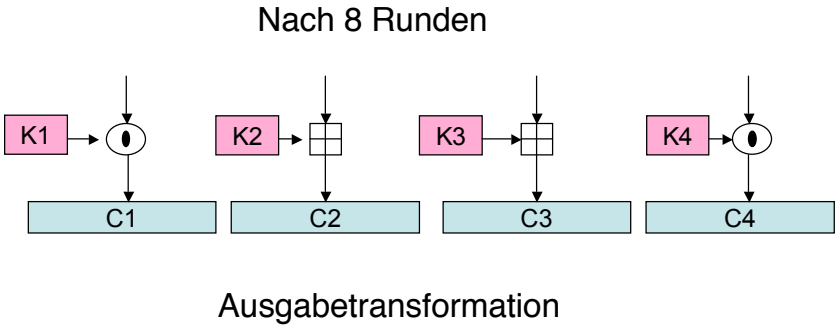
Eine Alternative zum DES ist der 1990 von Xuejia Lai und James L. Massey entwickelte International Data Encryption Algorithm (IDEA). IDEA ist eine Blockchiffre, die einen 64-Bit Eingabeblock mit einem 128-Bit langen Schlüssel verschlüsselt. Der IDEA wird ebenfalls sowohl zum Ver- als auch Entschlüsseln verwendet. Eine Verschlüsselungsoperation besteht aus acht Runden gefolgt von einer Ausgabetransformation. Um schnell in Software implementierbar zu sein, wurde auf die Verwendung von Permutationen und Substitutionen verzichtet. Die eingesetzten Operationen beschränken sich auf XOR, Addition und Multiplikation.

3.1.7 IDEA



Die 64-Bit Eingabewerte werden in vier 16-Bit Blöcke M1,...,M4 aufgeteilt, die die Eingabe der ersten Runde bilden. In jeder Runde *r* werden (beeinflusst von sechs Teilschlüsseln $K1^r$,...,$K6^r$ der Länge 16 Bit) verschiedene Operationen durchgeführt, deren Ergebnisse von der nächsten Runde weiterverarbeitet werden.

3.1.7 IDEA



Nach den acht Runden gibt es noch eine einfache Ausgabetransformation.

3.1.7 IDEA

52 Schlüssel werden benötigt, da es 6 Teilschlüssel in jeder der 8 Runden gibt und 4 Schlüssel für die Ausgabetransformation benutzt werden

Teilschlüsselerzeugung

Teile 128-Bit Schlüssel in acht 16-Bit Schlüssel. Danach mache einen Links-Shift um 25 Bit und entnehme die nächsten acht 16-Bit Schlüssel usw.

Insgesamt benötigt der Algorithmus für jede Runde sechs Schlüssel, das heißt bei acht Runden 48 Schlüssel. Dazu kommen noch die 4 Schlüssel für die Ausgabetransformation. Die Schlüssel haben eine Länge von 16-Bit. Die Teilschlüssel werden aus dem 128-bit Schlüssel generiert, indem dieser in acht 16-Bit Blöcke aufgeteilt wird, welche die ersten 8 Teilschlüssel bilden. Danach wird ein Links-Shift um 25 Bits vorgenommen und die nächsten acht Schlüssel genommen. In dieser Weise werden dann die restlichen Teilschlüssel erzeugt.

3.1.7 IDEA

Entschlüsselung

gleicher Algorithmus wie für die Verschlüsselung, aber

Teilschlüssel in umgekehrter Reihenfolge anwenden

einige Teilschlüssel modifizieren (für Addition und Multiplikation)

Sicherheit des IDEA

gilt als sicherer als DES aufgrund der höheren Schlüssellänge von 128 Bit

Erzeugt komplexe Abhängigkeit des Geheimtextes vom Klartext und dem Schlüssel

Die Entschlüsselung erfolgt im Wesentlichen analog zur Verschlüsselung, wobei sich die benötigten Teilschlüssel unterscheiden. Zum einen werden sie in umgekehrter Reihenfolge verwendet, zum anderen sind einige Teilschlüssel zu verändern. Dies sind jene, die für die Multiplikation und Addition verwendet wurden. Da XOR selbstinvers ist, ist dies für die beim XOR verwendeten Schlüssel nicht nötig.

IDEA hat im Gegensatz zum DES einen größeren Schlüssel der Länge 128 Bit. Dadurch vergrößert sich der Schlüsselraum (d.h. die Menge aller möglichen Schlüssel) und Brute-Force Attacken können effektiv nicht durchgeführt werden. Außerdem erreicht der Algorithmus schon nach wenigen Runden eine komplexe Abhängigkeit des Geheimtextes vom Klartext und dem Schlüssel.

3.1.8 Advanced Encryption Standard (AES)

Offizieller Nachfolger von DES ist der AES, für den 1997 eine öffentliche Ausschreibung vom NIST (National Institute of Standards and Technology) stattfand.

Anforderungen:

- Symmetrische Verschlüsselung, und zwar Blockchiffre
- Blockverschlüsselung: mindestens128-bit-Blöcke
- Schlüssellänge:128/192/256 bit
- Leicht in Hardware und Software implementierbar
- Gute Performance in Hard- und Software
- Widerstand gegen bekannte Krytoanalysemöglichkeiten
- Geringer Ressourcenverbrauch (für Einsatz in Smartcards)
- Weltweite freie Verfügbarkeit

Der DES Algorithmus kann nicht mehr als sicher angesehen werden, wie mehrere erfolgreiche Angriffe gezeigt haben. Auch der Triple-DES bietet keine zufriedenstellende Lösung, da er recht langsam ist und die verwendete Blocklänge von 64 Bit ein potenzielles Sicherheitsrisiko darstellt. Daher hat das US-amerikanische National Institute of Standards and Technology (NIST) Anfang 1997 zu einem offenen Wettbewerb aufgerufen, dessen Sieger als Advanced Encryption Standard (AES) festgelegt werden soll. Dabei wurden folgende Kriterien aufgestellt, die von den Algorithmen zu erfüllen sind:

AES muss ein symmetrischer Algorithmus sein, und zwar eine Blockchiffre.

AES muss mindestens 128 Bit lange Blöcke verwenden und Schlüssel von 128, 192 und 256 Bit Länge einsetzen können.

AES soll gleichermaßen leicht in Hard- und Software zu implementieren sein.

AES soll in Hardware wie Software eine überdurchschnittliche Performance haben.

AES soll allen bekannten Methoden der Kryptoanalyse widerstehen können.

Speziell für den Einsatz in Smartcards sollen geringe Ressourcen erforderlich sein (kurze Codelänge, niedriger Speicherbedarf).

Der Algorithmus muss frei von patentrechtlichen Ansprüchen sein und darf von jedermann unentgeltlich genutzt werden.

3.1.8 Advanced Encryption Standard (AES)

August 1998: 15 Vorschläge gehen beim NIST ein

April 1999: fünf Kandidaten

- MARS (IBM)
- RC6 (RSA Laboratories)
- Rijndael (Daemen und Rijmen)
- Serpent (Ross Anderson, Eli Biham, Lars Knudsen)
- Twofish (Bruce Schneier et. al)

kommen in die nächste Runde

Im August 1998 sind 15 Algorithmen beim NIST eingegangen und wurden öffentlich diskutiert und auf die Erfüllung der genannten Kriterien geprüft. Die engere Wahl wurde im April 1999 beendet und die fünf besten Kandidaten (MARS, RC6, Rijndael, Serpent, Twofish) kamen in die nächste Runde. Alle fünf Kandidaten erfüllen die Forderungen. Daher werden ab nun weitere Kriterien hinzugezogen: Die Algorithmen sollen auf theoretische Schwachstellen überprüft werden, durch die ein Algorithmus möglicherweise irgendwann einmal unsicher werden kann.

3.1.8 Advanced Encryption Standard (AES)

Oktober 2000:

- Sieger Rijndael der belgischen Mathematiker Daemen und Rijmen

- Hard- und Softwareimplementierung überdurchschnittlich schnell
- Einfach (500 Zeilen C-Code)

Im Mai 2000 wurden die Analysen und öffentlichen Diskussionen abgeschlossen und schließlich am 2. Oktober 2000 der Sieger bekannt gegeben: Der belgische Algorithmus Rijndael wird neuer Standard. Aber Rijndael hat insbesondere durch seine Einfachheit (die Referenz-Implementierung umfasst weniger als 500 Zeilen C-Code) und Performance überzeugt. Als einziger Algorithmus ist Rijndael als Hardware- und Software-Implementierung überdurchschnittlich schnell. Andere Kandidaten haben jeweils in unterschiedlichen Bereichen kleinere Schwächen.

3.1.8 Advanced Encryption Standard (AES)

Ablauf des AES

Blocklänge 128/ 192/ 256 Bit

Schlüssellänge 128/ 192/ 256 Bit

Zerlegen der Blöcke in Tabellen mit vier Zeilen, dessen Zellen 1 Byte groß sind

(d.h. 4 Spalten bei 128 bit, 6 Spalten bei 192 bit, 8 Spalten bei 256 bit)

Jeder Block wird Transformationen unterzogen. Die Anzahl der Runden variiert und ist von der Schlüssellänge k und Blockgröße b abhängig.

Runden

	b=128	b=192	b=256
k=128	10	12	14
k=192	12	12	14
k=256	14	14	14

AES (d.h. der Algorithmus Rijndael) ist eine Blockchiffre, dessen Blocklänge und Schlüssellänge unabhängig voneinander die Werte 128, 192 oder 256 Bit erhalten kann. Jeder Block wird zunächst in eine zweidimensionale Tabelle mit vier Zeilen geschrieben, dessen Zellen ein Byte groß sind. Die Anzahl der Spalten variiert somit je nach Blockgröße von 4 (128 Bit) bis 8 (256 Bit). Jeder Block wird nun nacheinander bestimmten Transformationen unterzogen. Aber anstatt jeden Block einmal mit dem Schlüssel zu Chiffrieren, wendet AES verschiedene Teile des Schlüssels nacheinander auf den Klartext-Block an. Die Anzahl dieser Runden (r) variiert und ist von Schlüssellänge (k) und Blockgröße (b) abhängig (siehe Tabelle auf der Folie).

3.1.8 Advanced Encryption Standard (AES)

Erzeugung der Rundenschlüssel aus dem Eingabeschlüssel:

- erster Rundenschlüssel ist der Eingabeschlüssel selbst, restliche Schlüssel durch Substitution (S-Boxen) und XOR-Operationen

Verschlüsselungsrunden bestehen aus 4 Schritten

1. Substitution()
2. ShiftRow()
3. MixColumn()
4. KeyAddition()

Der Ablauf des AES-Algorithmus ist wie folgt, wobei wir hier nicht auf alle Details eingehen. Zunächst werden die Rundenschlüssel auf Basis des Eingabeschlüssels erzeugt. Der Rundenschlüssel für die erste Runde ist der Eingabeschlüssel selbst, die restlichen Schlüssel ergeben sich durch Substitution und XOR-Verknüpfungen.

Die Verschlüsselungsrunden bestehen aus 4 Schritten:

Substitution()

ShiftRow()

MixColumn()

KeyAddition()

3.1.8 Advanced Encryption Standard (AES)

1. **Substitution**

   Jedes Byte der Tabelle für den Klartext wird mit einer S-Box verschlüsselt.

   S-Box dient zur Vermischung der Bytes

2. **Shift Row**

   Zeilen werden um eine bestimmte Anzahl von Spalten nach links verschoben

	b=128	b=192	b=256
Zeile 0	0	0	0
Zeile 1	1	1	1
Zeile 2	2	2	2
Zeile 3	3	3	4

Substitution

Im ersten Schritt jeder Runde wird für jedes Byte im Block ein Äquivalent in der S-Box gesucht. Somit werden die Daten monoalphabetisch verschlüsselt. Eine Substitutionsbox (S-Box) dient als Basis für eine monoalphabetische Verschlüsselung. Sie ist meist als Array implementiert und gibt an, welches Byte wie getauscht wird. Die S-Box in AES basiert auf einem mathematischen Zusammenhang und ist somit fest im Algorithmus implementiert. Doch dies ist kein Schwachpunkt, da die S-Box lediglich zur Vermischung der Bytes in Kombination mit weiteren Operationen und dem Schlüssel genutzt wird.

ShiftRow

Wie oben erwähnt, liegt ein Block in Form einer zweidimensionalen Tabelle mit vier Zeilen vor. In diesem Schritt werden die Zeilen um eine bestimmte Anzahl von Spalten nach links verschoben. Überlaufende Zellen werden von rechts fortgesetzt. Die Anzahl der Verschiebungen ist zeilen- und blocklängenabhängig:

3.1.8 Advanced Encryption Standard (AES)

3. **Mix Column**

   Jede Spalte wird mit einer Matrix multipliziert (Galois-Feld)

4. **Key Addition**

   XOR-Verknüpfung zwischen zu verschlüsselnden Block und Block für den Rundenschlüssel

MixColumn

Schließlich werden die Spalten vermischt. Über jeder Spalte wird eine Matrizenmultiplikation durchgeführt, und zwar über einem Galois-Feld GF(2^8). Die Bytes werden sozusagen nicht als Nummern, sondern als Polynome gesehen.

KeyAddition

Bei der Key Addition wird eine bitweise XOR-Verknüpfung zwischen dem Block und dem aktuellen Rundenschlüssel vorgenommen. Dies ist die einzige Funktion in AES, die den Algorithmus vom Benutzerschlüssel abhängig macht.

3.1.8 Advanced Encryption Standard (AES)

Entschlüsselung

Algorithmus wird rückwärts ausgeführt

1. KeyAddition()
2. MixColumn()
3. ShiftRow()
4. Substitution()

Zu beachten: andere S-Boxen (aus original S-Box berechenbar) müssen verwendet werden und Zeilenverschiebungen erfolgen in die andere Richtung

Anwendung des AES z.B. bei Verschlüsselungsprotokoll für WLAN oder SSH

Entschlüsselung

Bei der Entschlüsselung von Daten wird genau rückwärts vorgegangen. Die Daten werden zunächst wieder in zweidimensionale Tabellen gelesen und die Rundenschlüssel generiert. Allerdings werden alle Funktionen in jeder Runde in der umgekehrten Reihenfolge aufgerufen. Durch die vielen XOR-Verknüpfungen unterscheiden sich die meisten Funktionen zum Entschlüsseln nicht von denen zum Verschlüsseln. Jedoch muss eine andere S-Box genutzt werden (die sich aus der original S-Box berechnen lässt) und die Zeilenverschiebungen erfolgen in die andere Richtung.

Anwendung

AES wird u.a. von WPA, dem Verschlüsselungsprotokoll für WLAN und bei SSH genutzt.

3.1.9 Fazit Symmetrische Verfahren

Vorteil symmetrischer Verfahren
- Hohe Geschwindigkeit

Nachteil symmetrischer Verfahren
- Schlüsselmanagement
- Spontane Kommunikation zwischen unbekannten Personen schwierig

Lösung: **asymmetrische Verfahren**
(Idee Diffie/Hellman 1976)

Vorteile von symmetrischen Algorithmen sind die hohe Geschwindigkeit, mit denen Daten ver- und entschlüsselt werden. Ein Nachteil ist das Schlüsselmanagement. Um miteinander vertraulich kommunizieren zu können, müssen Sender und Empfänger vor Beginn der eigentlichen Kommunikation über einen sicheren Kanal einen Schlüssel ausgetauscht haben. Spontane Kommunikation zwischen Personen, die sich vorher noch nie begegnet sind, scheint so nahezu unmöglich. Soll in einem Netz mit n Teilnehmern jeder mit jedem zu jeder Zeit spontan kommunizieren können, so muss jeder Teilnehmer vorher mit jedem anderen der n - 1 Teilnehmer einen Schlüssel ausgetauscht haben. Insgesamt müssen also n(n - 1)/2 Schlüssel ausgetauscht werden.

Die Lösung stellten erstmals Diffie und Hellman 1976 vor, die ein Schlüsselpaar aus öffentlichem und privatem Schlüssel vorschlugen.