10

Symmetrische Kryptografie

10.1 Das Problem von Alice und Bob

Was ist Kryptografie?

Der Begriff **Kryptografie** und die Bezeichnung für die verwandten Disziplinen **Kryptologie** und **Kryptoanalyse** stammen aus dem Griechischen. „Kryptos“ bedeutet so viel wie „geheim“ oder „verborgen“. „Graphein“ steht für „schreiben“. Die Endung „-analyse“ stammt von „analysein“, deutsch „entziffern“. „Logos“ bedeutet „Sinn“. Somit lassen sich die drei Disziplinen wie folgt unterteilen:

- Kryptografie: die Wissenschaft der Geheimschrift
- Kryptoanalyse: die Kunst, Geheimschrift (unbefugt) entziffern zu können, den Code zu brechen
- Kryptologie: die Wissenschaft, die Kryptografie und Kryptoanalyse miteinander vereint

Ziel der Kryptografie ist es, Nachrichten in eine Art „Geheimschrift“ zu übersetzen und diese so auf einem möglicherweise unsicheren Weg zum vorgesehenen Empfänger zu schicken. Die für die Nachricht gewählten Zeichen sollen so gewählt sein, dass nur der vorbestimmte Empfänger in der Lage sein sollte, den Inhalt der Nachricht wieder lesbar zu machen.

Die Verwendung von geheimen Zeichen ist für die Kryptografie nicht notwendig und auch nicht sinnvoll. Die Zeichen des **Klartextes** sind dieselben wie die des **Chiffretextes**. Bei den historischen Verschlüsselungsverfahren wurden die Zeichen des Alphabetes verwendet, bei den modernen, computergestützten Verfahren werden Gruppen binärer Zustände (meist Bytes oder Blöcke aus mehreren Bytes) eingesetzt.

Akteure in der Kryptologie

In der Kryptologie hat es sich eingebürgert, Problem- oder Protokollbeschreibungen nicht abstrakt vorzunehmen: „A möchte B eine verschlüsselte Nachricht zusenden, C versucht, diese Nachricht abzufangen und deren Inhalt zu entschlüsseln.“

Vielmehr werden in der kryptologischen Literatur Akteure mit Vornamen genannt, deren Anfangsbuchstabe Ihnen Aufschluss über die Rolle des Akteurs im jeweiligen Beispiel gibt. Aus den Hauptteilnehmern A und B werden auf diese Weise Alice und Bob.

Häufig verwendete Akteure

Alice	Hauptakteurin A, startet einen Vorgang
Bob	Hauptakteur B, ist Nutznießer des Vorgangs
Eve	Lauscherin, E = Eavesdropper
Mallory	Böswilliger, aktiver Angreifer, M = Malicious

Seltener vorkommende Akteure

Carol	Hauptakteurin C, dritte Teilnehmerin an einem kryptografischen Protokoll
Dave	Hauptakteur D, vierter Teilnehmer
Trent	Vertrauenswürdiger Vermittler, T = Trust
Walter	Ein Wächter, der Alice und Bob beschützt, W = Warden
Sara	Ein Server

Informationen und Schlüssel

Die grundlegende Herausforderung der Kryptografie ist es, eine Nachricht so zu verschlüsseln, dass deren Inhalt während des Transports vor der Kenntnisnahme durch Unbefugte geschützt ist. Der rechtmäßige Empfänger der Nachricht muss aber in der Lage sein, diese zu dechiffrieren, also die Verschlüsselung wieder rückgängig zu machen.

Damit dies möglich ist, benötigt der Empfänger eine Zusatzinformation – einen Schlüssel. Ein ideales Verschlüsselungsverfahren ist so sicher, dass es nur mithilfe des passenden Schlüssels möglich ist, an den Inhalt der Nachricht zu kommen.

Kryptografische Verfahren, bei denen Sender und Empfänger denselben Schlüssel zum Ver- wie auch zum Entschlüsseln verwenden, werden **symmetrische Verschlüsselungsverfahren** genannt.

Kerckhoffs' Forderungen

Der niederländische Philologe Kerckhoffs von Nieuwenhof (1835–1903) formulierte sechs Prinzipien für militärische Verschlüsselungsalgorithmen. Von diesen sind die wichtigsten:

- Das System selbst darf nicht geheim sein, es sollte kein Problem darstellen, wenn es in die Hände des Feindes gelangt.
- Der Schlüssel muss ausreichend klein, variabel und modifizierbar sein.

Vor allem die Implikationen der ersten Forderungen sind als das **Kerckhoffs'sche Prinzip** in der Öffentlichkeit bekannt. Ein sicheres System muss auch noch sicher sein, wenn der Algorithmus bekannt ist, mit dem gearbeitet wird. Die Sicherheit hängt also einzig und allein vom verwendeten Schlüssel ab.

Ein Verfahren, das sich an dieses Prinzip hält, genießt mehrere Vorteile:

- Zwischen Alice und Bob kann der Algorithmus offen bekannt gegeben werden, und auch Eve weiß, wie er funktioniert.
- Alice und Bob brauchen **nur einen** gemeinsamen geheimen Schlüssel, den sie niemand anderem mitteilen. Der Transport des Schlüssels ist relativ leicht, da er im Vergleich zur Nachricht oder den Instruktionen für den Algorithmus relativ klein ist.
- Dadurch, dass der Algorithmus öffentlich bekannt ist, haben viele Kryptoanalytiker die Möglichkeit, nach Schwachstellen und Angriffspunkten zu suchen. Sollte eine gefunden werden, kann über Lösungen diskutiert werden. Algorithmen, die öffentlich bekannt und anerkannt sind, gelten als relativ sicher, da viele Analytiker vergeblich versucht haben, eine Schwachstelle zu finden.
- Eve kann die Nachricht nur entziffern, wenn sie in den Besitz des Schlüssels gelangt.

Einige Institutionen und Firmen beherzigen dieses Prinzip trotz der offensichtlichen Vorteile nicht. Sie versuchen, ein höheres Sicherheitsniveau durch die Geheimhaltung der Algorithmen zu erreichen. Wenn die Algorithmen in Hardware implementiert sind, bedeutet das: Gelangt ein Angreifer in den Besitz der Verschlüsselungsmaschine, so ist das ein wesentlicher Schritt zum Knacken des Codes. Ein historisches Beispiel hierfür ist die kriegsentscheidende Erbeutung einer deutschen Enigma-Maschine durch die alliierten Streitkräfte.

Auch in der Software wird oft auf Geheimhaltung der Algorithmen gesetzt. Diese Verschleierungstaktik wird auch „Security through obscurity“ genannt und ist ebenfalls relativ wirkungslos. Jegliche Software kann disassembliert und somit auf Sourcecode-Ebene analysiert werden. Mit entsprechendem Zeit- und Arbeitsaufwand ist es also jedem Angreifer möglich, Kenntnis über die vermeintlich versteckten Algorithmen zu erlangen. Ein Beispiel hierfür ist die DVD-Verschlüsselung CSS, die gebrochen wurde, indem eine DVD-Playersoftware per Debugger analysiert wurde. Damit wurden Vendor-Key und Verschlüsselungsalgorithmus herausgefunden.

10.2 Einfache Verschlüsselungsmethoden

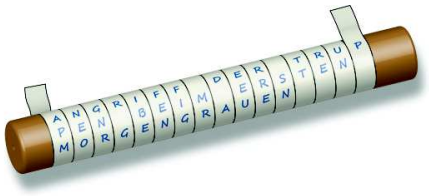
Stabchiffre

Eine der ältesten überlieferten Verschlüsselungsmethoden ist die von den Griechen eingesetzte sogenannte Stabchiffre (Skytale). Hierbei wird zur Übermittlung von Nachrichten zwischen dem Befehlshaber und seinen Truppen ein Band der Länge nach um einen Stab mit einem bestimmten, vorher festgelegten Durchmesser gewickelt. Die Nachricht wird entlang der Achse des Stabes niedergeschrieben. Zum Transport an den Empfänger wird das so beschriebene Band abgewickelt und zusammengerollt.

Würde der Feind den Boten abfangen und das Band betrachten, so sähe er nur scheinbar sinnlose Buchstabenkombinationen, solange er das Verschlüsselungsprinzip und den Durchmesser des Stabes nicht kennt. Ein autorisierter Empfänger dieser Nachricht besitzt jedoch einen Stab mit identischem Durchmesser. Wickelt der Empfänger das Band um diesen Stab, so kann er die ursprüngliche Nachricht wieder lesen. Der Durchmesser des verwendeten Stabes ist hier also in gewisser Weise der Schlüssel für dieses System.

Schreibt der Feldherr z. B. einen Befehl für seine Truppen in dreizeiliger Form auf das Band:

```
Angriff der Trup
pen beim ersten
Morgengrauen
```



Eine Skytale

So ist auf dem abgewickelten Band Folgendes zu lesen:

```
ApMneognrr gibefenfig mrd aeeurre snTt re un p
```

Die Stabchiffre nimmt also nur eine Umverteilung der Buchstaben vor. Die Zeichen selbst und deren Häufigkeit bleibt erhalten, nur die Reihenfolge ändert sich (Transposition).

Caesar-Chiffre

Der römische Feldherr Caesar setzte zur Kommunikation mit seinen Generälen eine Methode ein, die später auch nach ihm benannt wurde. Cicero lobte Caesar in seinen Schriften dafür, dass er eine absolut sichere Methode gefunden habe, geheime Nachrichten zu verfassen.

Caesars Methode bestand darin, in einer Nachricht jeden Buchstaben durch denjenigen Buchstaben zu ersetzen, der drei Stellen später im Alphabet folgt. Wollte jemand eine derartige Nachricht lesen, so musste er im Chiffretext jeden Buchstaben durch den um drei Stellen vorgestellten Buchstaben ersetzen.

Caesar verschob also das Alphabet des Chiffretextes im Vergleich zum Klartextalphabet. Im Gegensatz zu einer Chiffre, die auf Änderung der Reihenfolge beruht (Transpositionschiffre), ist die Caesar-Chiffre also eine Rotationschiffre (ROT).

Die Zuordnung von Klartextbuchstaben zu Buchstaben im Chiffretext sieht nach Caesar so aus:

Klartext	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
Chiffre	**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**

Der Buchtitel **DE BELLO GALLICO** mit der Caesar-Chiffre verschlüsselt lautet: **GH EHOOR JDOOLFR**.

Die Caesar-Chiffre ist eigentlich ein Sonderfall. Caesar nahm zum Chiffrieren immer oben stehende Tabelle – also immer denselben Schlüssel.

Der etwas allgemeinere Fall ist die Rotationschiffre. Wenn Sie obige Tabelle betrachten, so können Sie die beiden Alphabete um 1 bis 25 Buchstaben gegeneinander verschieben. Sie haben also 25 verschiedene Schlüssel. Der Schlüssel 26 – eine Verschiebung um 0 Zeichen, würde keinen Sinn machen, da dann der Klar- und der Chiffretext identisch wären.

Eine Zahl zwischen 1 und 25 stellt also den Schlüssel dar, den Alice zu Bob übertragen muss, damit dieser eine entsprechend chiffrierte Nachricht wieder lesbar machen kann.

Kryptoanalyse von ROT-Verschlüsselung

Selbst wenn Sie sich nicht wie Caesar immer auf den Schlüssel 3 (ROT-3-Verschlüsselung) festlegen, sondern einen beliebigen Schlüssel zwischen 1 und 25 wählen, ist es relativ einfach, die Klartextnachricht aus dem Chiffretext zu ermitteln. Da es nur 25 mögliche Schlüssel gibt, reicht es, der Reihe nach sämtliche Schlüssel auf den Chiffretext anzuwenden (Brute-Force-Angriff). Nur in einer Zeile wird dann sinnvoller Text stehen.

Eve möchte wissen, welche Nachricht sich hinter der Chiffre **PUALYULADVYRPUNZPJOLYOLPA** versteckt, die mit ROT verschlüsselt wurde. Dazu erstellt sie eine Tabelle und probiert alle Schlüssel durch.

```
ROT-1:  OTZKXTKZCUXQOTMYOINKXNKOZ          ROT-14: BGMXKGXMPHKDBGZLBVAXKAXBM
ROT-2:  NSYJWSJYBTWPNSLXNHMJWMJNY          ROT-15: AFLWJFWLOGJCAFYKAUZWJZWAL
ROT-3:  MRXIVRIXASVOMRKWMGLIVLIMX          ROT-16: ZEKVIEVKNFIBZEXJZTYVIYVZK
ROT-4:  LQWHUQHWZRUNLQJVLFKHUKHLW          ROT-17: YDJUHDUJMEHAYDWIYSXUHXUYJ
ROT-5:  KPVGTPGVYQTMKPIUKEJGTJGKV          ROT-18: XCITGCTILDGZXCVHXRWTGWTXI
ROT-6:  JOUFSOFUXPSLJOHTJDIFSIFJU          ROT-19: WBHSFBSHKCFYWBUGWQVSFVSWH
ROT-7:  INTERNETWORKINGSICHERHEIT          ROT-20: VAGREARGJBEXVATFVPUREURVG
ROT-8:  HMSDQMDSVNQJHMFRHBGDQGDHS          ROT-21: UZFQDZQFIADWUZSEUOTQDTQUF
ROT-9:  GLRCPLCRUMPIGLEQGAFCPFCGR          ROT-22: TYEPCYPEHZCVTYRDTNSPCSPTE
ROT-10: FKQBOKBQTLOHFKDPFZEBOEBFQ          ROT-23: SXDOBXODGYBUSXQCSMROBROSD
ROT-11: EJPANJAPSKNGEJCOEYDANDAEP          ROT-24: RWCNAWNCFXATRWPBRLQNAQNRC
ROT-12: DIOZMIZORJMFDIBNDXCZMCZDO          ROT-25: QVBMZVMBEWZSQVOAQKPMZPMQB
ROT-13: CHNYLHYNQILECHAMCWBYLBYCN
```

Wie Sie sehen, ist die einzige Zeile, die einen Sinn ergibt, diejenige, in der von einer Verschlüsselung mit ROT–7 ausgegangen wurde. Der Klartext lautet „INTERNETWORKINGSICHERHEIT“. ROT ist wegen der geringen Anzahl möglicher Schlüssel („Größe des Schlüsselraums“) als nicht sicher einzustufen.

Dieses Beispiel zeigt, warum eine Brute-Force-Attacke für potenzielle Angreifer immer unrentabler wird, je größer der Schlüsselraum bei einer bestimmten Methode wird: Er müsste sehr viel Zeit in das Durchprobieren der Schlüssel investieren.

In der Regel gilt eine Methode für einen bestimmten Zweck dann als sicher, wenn davon ausgegangen werden kann, dass ein Brute-Force-Angriff extrem lange dauern würde. Übersteigt der Zeit- oder Kostenaufwand dann den Wert der verschlüsselten Information oder kann der Code erst zu einem Zeitpunkt geknackt werden, wenn die geschützte Nachricht bereits wertlos geworden ist, gilt das verwendete Verfahren als sicher. Beispielsweise ist die verschlüsselte Kommunikation bezüglich einer Firmenübernahme für einen Angreifer (z. B. für einen Aktienhändler, der Geschäfte mit diesen Insiderinformationen machen möchte) wertlos, wenn die Übernahme inzwischen stattgefunden hat.

Häufigkeitsanalyse

Eine andere Methode, ROT-Code zu brechen, besteht darin, die Häufigkeitsverteilung der vorkommenden Buchstaben auszuwerten. Wenn die verschlüsselte Nachricht eine natürliche Sprache ist, so treten die einzelnen Buchstaben mit unterschiedlicher Häufigkeit auf. In der deutschen Sprache ist der Buchstabe E der häufigste. Will Eve nun den Schlüssel zu einer ROT-Nachricht errechnen, ohne auf einen Brute-Force-Angriff angewiesen zu sein, zählt sie die Häufigkeit jedes Buchstabens im Chiffretext. Wenn dieser ausreichend lang ist, ergibt sich eine Verteilung der Häufigkeiten, die annähernd der Verteilung der vermuteten Sprache entspricht. Ist der am häufigsten aufgetretene Buchstabe im Chiffretext ein R, so wird sie annehmen, dass E zu R verschlüsselt worden ist. Der Abstand von E zu R im Alphabet ist 13 – somit versucht sie eine Entschlüsselung mit der Annahme, dass mit ROT-13 verschlüsselt wurde. War die Annahme richtig, so kann der gesamte Text entschlüsselt werden.

Auftretenswahrscheinlichkeiten der Buchstaben in Prozent

Zeichen:	a	b	c	d	e	f	g	h	i	j	k	l	m
Deutsch	6.47	1.93	2.68	4.83	17.48	1.65	3.06	4.23	7.73	0.27	1.46	3.49	2.58
Englisch	8.04	1.54	3.06	3.99	12.51	2.30	1.96	5.49	7.26	0.16	0.67	4.14	2.53
Zeichen:	n	o	p	q	r	s	t	u	v	w	x	y	z
Deutsch	9.84	2.98	0.96	0.02	7.54	6.83	6.13	4.17	0.94	1.48	0.04	0.08	1.14
Englisch	7.09	7.60	2.00	0.11	6.12	6.54	9.25	2.71	0.99	1.92	0.19	1.73	0.09

ROT-13

Die Rotationsverschlüsselung mit dem Schlüssel 13 hat eine besondere Stellung bei den Rotationschiffren. Da das Alphabet 26 Buchstaben hat, bedeutet eine Verschiebung um 13 Stellen eine Verschiebung um genau die Hälfte des Alphabets. Wird ein so erzeugter Chiffretext erneut ROT-13 verschlüsselt, so wird wiederum um die Hälfte verschoben – was dazu führt, dass der Chiffretext wieder dem Klartext entspricht.

Die ROT-13-Chiffre ist in vielen gängigen E-Mail- und Newsreader-Programmen integriert und erlaubt es dem Benutzer, mit einem Tastendruck den gewählten Text zu verschlüsseln. Als Einsatzgebiet ist allerdings nicht die Vertraulichkeit vorgesehen – das wäre sinnlos, da jeder Unbefugte mit ROT-13-fähiger Software leicht in der Lage wäre, den Klartext zu lesen. ROT-13 wird verwendet, um versehentliches Lesen eines Textes beim Empfänger zu verhindern.

Vigenère-Chiffre

Das Problem der ROT-Chiffre ist, dass die Zeichen zwar durch andere ersetzt werden, die Häufigkeit von einzelnen Zeichen und die Reihenfolge des Chiffrealphabets aber erhalten bleibt. In der ROT-Chiffre gibt es nur ein Chiffrealphabet. Häufige Zeichen werden also immer durch dieselben anderen Zeichen ersetzt, die dann genauso häufig sind.

Ziel einer Verbesserung wäre es also, eine Verschlüsselung in der Art vorzunehmen, dass im Ergebnis alle Buchstaben gleich häufig vorkommen. Das kann z. B. durch Verwendung von mehreren Chiffrealphabeten erfolgen.

Der Benediktiner Johannes Heidenberg (1462–1516) hat zu diesem Zweck in seiner Tabula Recta alle 26 ROT-Verschiebungen aneinandergereiht. Diese Tabelle wird auch Vigenère-Quadrat genannt, benannt nach dem französischen Kryptologen Blaise de Vigenère (1523–1596).

Das Vigenère-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

Ein Text wird nun folgendermaßen verschlüsselt: Sender und Empfänger benötigen als Schlüssel ein Passwort. Dies wird dann über den Klartext geschrieben und wenn nötig so oft wiederholt, bis die Länge der Nachricht erreicht wurde.

```
Passwort: KRYPTOGRAFIEKRYPTOGRAFIEKRYPTOGRAFIE
Klartext: DIESENACHRICHTISTSTRENGGEHEIM
Chiffre:  NZCHXBGTHWQGRKGHMGZIESOKOYCXF
```

Alice verschlüsselt nun den ersten Buchstaben ihrer Nachricht, indem sie in der Spalte mit dem ersten Buchstaben des Passworts (Spalte K) die Zeile mit dem Klartextbuchstaben am Anfang sucht (Zeile D). Am Kreuzungspunkt dieser Zeile und Spalte befindet sich der Chiffre-Buchstabe N.

Anschließend wird der zweite Buchstabe der Nachricht mit dem zweiten Buchstaben des Passworts verschlüsselt usw., bis die komplette Nachricht verschlüsselt wurde.

Die Häufigkeiten der einzelnen Buchstaben sind nun gleichmäßiger verteilt. Der Buchstabe E kommt in obigem Beispiel 5-mal vor, aber anstatt 5-mal mit demselben Chiffrezeichen verschlüsselt zu werden, wird er durch 2-mal C, X, E und O ersetzt. Eine reine Häufigkeitsanalyse würde hier also ins Leere laufen.

Kryptoanalyse von Vigenère

Allerdings hat auch die Vigenère-Verschlüsselung leicht ausnutzbare Schwachstellen. Eine davon ist das Auftreten sogenannter Parallelstellen.

Werden längere Texte verschlüsselt, so kommen zwangsläufig im Klartext identische Zeichenketten vor. Dazu gehören häufig auftretende Wörter wie „der“, „die“, „das“ usw. Treffen derartige Zeichenfolgen bei der Verschlüsselung auf unterschiedliche Zeichen des Passwortes, so werden sie auch anders verschlüsselt.

```
Passwort: SECRETSECRETSECRETSECRETSECRET
Klartext: .....DAS...........DAS........
Chiffre:  .....WSW...........HCJ........
```

Allerdings besteht auch eine Wahrscheinlichkeit, dass bei der Verschlüsselung dieselben Zeichenfolgen des Klartexts auf dieselben Zeichen des Passwortes treffen:

```
Passwort: SECRETSECRETSECRETSECRETSECRET
Klartext: .....DAS..............DAS....
Chiffre:  .....WSW..............WSW....
```

In diesem Fall werden die Klartextpassagen auch mit denselben Zeichenfolgen im Chiffretext codiert. Der preußische Infanteriemajor Friedrich Wilhelm Kasiski (1805–1881) beschreibt in seinem Buch „Die Geheimschriften und die Dechiffrierkunst“ eine Methode, in der sämtliche derartigen Parallelstellen gesucht werden. Die Idee, die mit dieser Methode verfolgt wird, geht davon aus, dass identische Textpassagen im Chiffretext dadurch zustande kommen, dass derselbe Klartext auf dieselben Zeichen des Passwortes trifft – dies kann nur in einem ganzzahligen Vielfachen der Passwortlänge passieren.

Von allen gefundenen Parallelstellen werden also die Abstände zwischen den Parallelstellen ermittelt. Es gilt nun, die Länge des Passwortes zu ermitteln, die der größte gemeinsame Teiler der Abstände aller Parallelstellen ist. Die gefundenen Abstände müssen dazu in ihre Primfaktoren zerlegt werden. Diese Ermittlung ist jedoch nicht absolut eindeutig, da es auch zufällige Parallelstellen geben kann, die somit irrelevant sind.

Hat der Angreifer nun eine vermutete Passwortlänge errechnet, so wird der Chiffretext in Gruppen eingeteilt, von denen man vermutet, dass sie mit demselben Buchstaben des Passwortes verschlüsselt wurden. Nimmt man also an, das Passwort sei 5 Zeichen lang, so bildet man 5 Gruppen. In der ersten Gruppe befindet sich der 1., 6., 11., 16. Buchstabe usw., in der zweiten Gruppe der 2., 7., 12., 17. in der dritten der 3., 8., 13., 18. usw.

Für jede dieser Gruppen kann nun getrennt eine Häufigkeitsanalyse durchgeführt und der häufigste Chiffrebuchstabe ermittelt werden – wurde Deutsch oder Englisch als Sprache der Nachricht verwendet, so kann wieder angenommen werden, dass der jeweils am häufigsten auftauchende Buchstabe ursprünglich ein E war. Führt dies allein noch nicht zum gewünschten Ergebnis, so kann für jede dieser Gruppen das komplette Häufigkeitsgebirge der vorkommenden Zeichen ermittelt werden. Dies erlaubt dann sehr zuverlässig die komplette Rekonstruktion des verwendeten Passwortes.

```
Chiffre: KYPQY PSWCE EMAZG FJMZI EYMLL EVV

Vermutete Länge des Passwortes: 5

Gruppen: KPEFEE, YSMJYV, PWAMMV, QCZZL, YEGIL
```

Neben dieser als Kasiski-Test bekannten Methode existiert eine noch effektivere Methode zur Ermittlung der Schlüssellänge: der Friedman-Test. Auf sie wird in diesem Buch aber nicht eingegangen.

Vernam

Wie Sie gesehen haben, garantiert auch die Verwendung mehrerer Chiffrealphabete (polyalphabetische Chiffre) keinen absoluten Schutz. Ist das Passwort in Relation zur versendeten Nachricht kurz, so treten Parallelstellen auf, die es Eve ermöglichen, Rückschlüsse auf die Länge des Passwortes zu ziehen. Auf diese Weise ist der Weg zu einem statistischen Angriff geebnet.

Die Konsequenz daraus wäre, eine mit Vigenère verschlüsselte Nachricht sicherer zu machen, indem das Passwort verlängert wird.

Der amerikanische Ingenieur Gilbert S. Vernam (1890–1960) erfand 1917 ein Chiffriersystem, das, basierend auf der Vigenère-Methode, perfekte Sicherheit garantieren konnte:

Ein Chiffriersystem ist dann perfekt, wenn jeder Klartext mit einem zum Chiffriersystem gehörenden Schlüssel auf jeden beliebigen Chiffretext abgebildet werden kann. Im Rahmen von Vigenère ist das möglich, wenn der Schlüssel genauso lang ist wie die Nachricht selbst.

Ein perfektes System bietet dem Angreifer keinerlei Anhaltspunkt für eine statistische Analyse, und ein Brute-Force-Angriff käme einem Ausprobieren sämtlicher möglicher Nachrichten gleich.

Wenn Eve diese Nachricht erhält: **HQAVUXGBCQMTIM**, so kann es bei einer Vernam-Chiffre sein, dass der Klartext **ANGRIFFUMFUENF** lautet, der mit dem Passwort **HDUEMSBHQLSPV** verschlüsselt wurde – genauso gut könnte es sich aber um den Klartext **FRUEHSTUECKSEI** handeln, der mit **CZGRNFNHYOCBEE** verschlüsselt wurde.

Die Vernam-Chiffre ist dann sicher, wenn die Buchstaben des Passwortes absolut zufällig gewählt werden und einmal benutzte Passwörter nie wieder verwendet werden. Früher wurden die Passwortbuchstaben auf die Blätter eines Abreißblocks geschrieben, weshalb eine Vernam-Chiffre auch One-Time-Pad (Abreißblock) genannt wird.

Was ist der Vorteil einer Vernam-Chiffre? Sender und Empfänger müssen das gleiche Passwort besitzen – und dieses ist ebenso lang wie die Nachricht selbst. Der Vorteil hierbei liegt in der Tatsache, dass Sender und Empfänger den Austausch des Passwortes (oder besser: lange zufällige Zeichenketten, die als Passwort auf Vorrat dienen) zu einem frei wählbaren Zeitpunkt durchführen können. Die Nachricht selbst muss aber meist zu einem bestimmten Zeitpunkt dringend abgeschickt werden.

Hier taucht das Bild eines Diplomaten auf, der sich mit einem an sein Handgelenk geketteten Koffer auf die Reise macht – er transportiert möglicherweise neue One-Time-Pad-Passwörter. Im Zweiten Weltkrieg und während der Ära des Kalten Krieges wurden für Nachrichten höchster Geheimhaltungsstufe von den beteiligten Parteien One-Time-Pads eingesetzt. Der „Heiße Draht“ zwischen Washington und Moskau wurde ebenfalls mit einem One-Time-Pad gesichert.

10.3 Symmetrische Verfahren

Blockchiffre/Stromchiffre

Im Computerzeitalter treten die klassischen Verschlüsselungsmethoden eher in den Hintergrund. Die Methoden, mit denen in der Informationstechnologie gearbeitet wird, erlauben aufgrund der Rechenleistung deutlich komplexere Operationen.

Grundsätzlich können moderne kryptografische Verfahren unterteilt werden in solche, die **symmetrisch** arbeiten (wenn Alice und Bob denselben Schlüssel benutzen), und in solche, die **asymmetrisch** arbeiten (hier besitzen Alice und Bob verschiedene Schlüssel).

Bei der Unterscheidung von Verschlüsselungsalgorithmen kann auch unterschieden werden, ob der Algorithmus die Daten als Blöcke mehrerer Bits auf einmal verschlüsselt oder jedes Bit einzeln. Erstere werden **Blockchiffren** genannt, letztere **Stromchiffren**.

DES – Data Encryption Standard

Um einen einheitlichen kryptografischen Algorithmus für die IT-Welt zu fördern, führte das NIST (National Institute of Standards and Technology, USA) im Jahre 1973 eine öffentliche Ausschreibung für einen kryptografischen Algorithmus mit folgenden Forderungen durch:

- Der Algorithmus muss einen hohen Grad an Sicherheit gewährleisten.
- Der Algorithmus muss vollständig spezifiziert und leicht nachzuvollziehen sein.

- Die Sicherheit des Algorithmus muss auf dem Schlüssel basieren, nicht auf der Geheimhaltung des Algorithmus.
- Der Algorithmus muss für alle Anwender zur Verfügung stehen.
- Der Algorithmus muss an verschiedene Anwendungen angepasst werden können.
- Der Algorithmus muss sich kostengünstig in elektronische Komponenten implementieren lassen.
- Der Algorithmus muss effizient in der Benutzung sein.
- Es muss möglich sein, den Algorithmus zu validieren.
- Der Algorithmus muss exportierbar sein.

Der aussichtsreichste Kandidat für diesen Algorithmus wurde von IBM eingereicht. Bevor er jedoch zum Standard deklariert wurde, wurde der Entwurf von IBM noch von der NSA (National Security Agency) modifiziert, die die Schlüssellänge als sichtbare Änderung von 128 Bit auf 56 Bit reduzierte. Über Gründe für Änderungen innerhalb des Algorithmus war die NSA zu keiner Stellungnahme zu bewegen, was viele Jahre für Bedenken in der Öffentlichkeit sorgte, die NSA habe eine Hintertür in den Ablauf des DES eingebaut, um diesen leichter entschlüsseln zu können.

Die am weitesten verbreitete Anwendung des DES war die Verschlüsselung des Geldautomatensystems. Die PIN einer EC-Karte wurde unter Verwendung der Bankleitzahl, Kontonummer, des Verfallsdatums und eines geheimen Schlüssels berechnet.

Aufbau des DES

Das Ziel dieses Buches ist nicht, den Leser zu befähigen, kryptografische Algorithmen selbst nachrechnen zu können. Sie sollten jedoch ein grundlegendes Verständnis für die Arbeitsweise von computerbasierten Verschlüsselungsalgorithmen besitzen, damit Sie Ähnlichkeiten zuordnen und möglicherweise auch Sicherheit und Sicherheitsrisiken selbst besser einschätzen können.

DES ist eine Blockchiffre und verschlüsselt jeweils Blöcke aus 64 Bit. Es wird ein Schlüssel mit der Länge von 64 Bit benötigt, von dem jedoch 8 Bits als Paritätsbits zur Fehlererkennung verwendet werden. Somit ist der Schlüssel effektiv 56 Bit lang.

DES besteht aus mehreren Runden (insgesamt 16), die durchlaufen werden. Soll ein Datenblock verschlüsselt werden, so durchläuft dieser zunächst die Initial Permutation (Eingangspermutation). Hier werden die Datenbits anhand einer fest vorgegebenen Tabelle in ihren Positionen vertauscht.

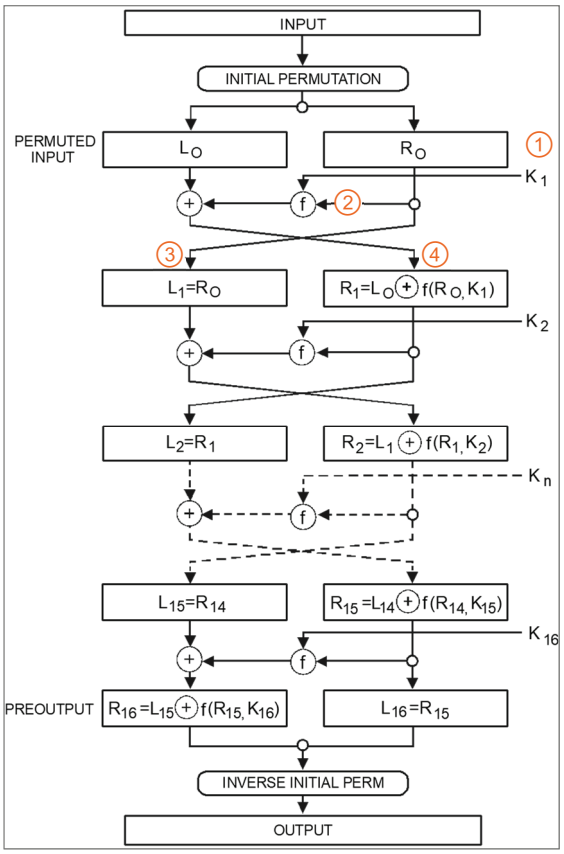
Anschließend werden die 64 Datenbits in zwei Blöcke zu jeweils 32 Bits aufgeteilt. Diese werden als L und R bezeichnet. Die nebenstehende Zahl in der Grafik zeigt jeweils an, um welche Runde es sich handelt.

Vor der ersten Runde existieren also die Blöcke L_0 und R_0 ①. Der DES berechnet, basierend auf dem 56-Bit-Schlüssel, für jede Runde einen sogenannten Round-Key (Rundenschlüssel), der in der Grafik mit K_1 bis K_{16} bezeichnet ist. Durch ein definiertes Verfahren werden in jeder Runde 48 aus den 56 vorhandenen Bits des Gesamtschlüssels ausgewählt.

Kern des DES ist die Verschlüsselungsfunktion f ②. Die erste Runde der Verschlüsselung besteht nun darin, dass die Funktion f auf R_0 mithilfe von K_1 angewandt wird. Das Ergebnis dieser Operation wird modulo 2 zu L_0 addiert (= XOR).

Bei einer bitweisen Addition modulo 2 werden 2 Bits addiert, aber der Überlauf wird verworfen. Das Ergebnis ist dasselbe wie bei einer XOR-Operation (exklusives ODER).

Für die nächste Runde ist die neue linke Hälfte L_1 gleich dem alten R_0 ③ und die neue rechte Hälfte gleich dem Ergebnis der modulo 2 Addition ④.



DES-Verschlüsselung

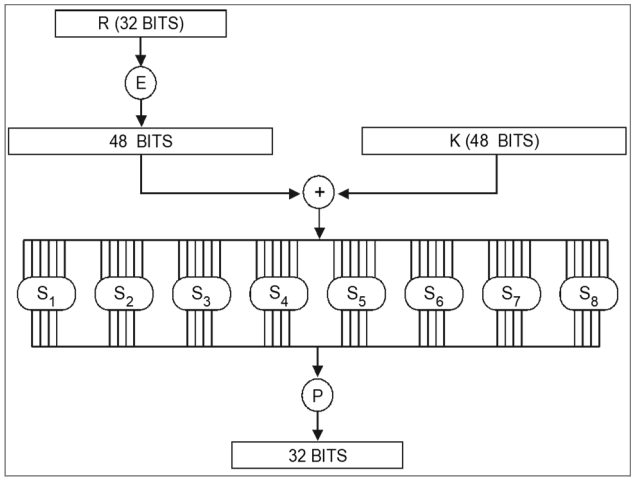
Dieser Vorgang wird nun für alle 16 Runden durchgeführt und durchläuft abschließend die Inverse Initial Permutation (Inverse Eingangspermutation). Das Ergebnis ist der mit dem Gesamtschlüssel verschlüsselte 64-Bit-Block.

0 + 0	0
0 + 1	1
1 + 0	1
1 + 1	0

Modulo 2 Addition, XOR

Funktion f

Die Funktion f, die in jeder Runde die Hälfte R mit dem jeweiligen Rundenschlüssel verschlüsselt, erhält 32 Datenbits und führt eine Expansionspermutation (E) durch. Dieser Vorgang sorgt anhand einer festgelegten Tabelle dafür, dass bestimmte Bits von R verdoppelt werden, sodass R auf eine Länge von 48 Bit anwächst. Nur bei gleicher Länge können diese Daten nämlich mit dem ebenfalls 48 Bit langen Rundenschlüssel modulo 2 addiert werden (=XOR).



DES-Funktion f

Nach der Addition muss das Ergebnis wieder auf eine Länge von 32 Bits gebracht werden, damit es im Algorithmus weiterverwendet werden kann. Jeweils 6 Bits wandern in eine der acht sogenannten S-Boxen (S wie Substitution, Ersetzen).

Anhand der fest in den jeweiligen S-Boxen definierten Tabellen werden die 6 Eingangsbits durch 4 Ausgangsbits substituiert. Alle Ergebnisbits werden dann wieder aneinandergefügt und stellen das Endergebnis der Funktion f in der aktuellen Runde dar.

S-Box 1	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	8
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

Sollen 6 Bits durch eine S-Box substituiert werden, so geben das erste und das letzte Bit als Dualzahl gelesen die Zeilennummer der S-Box an. Die 4 restlichen Bits in der Mitte bilden die Spaltennummer. Das Ergebnis ist der in der entsprechenden Zeile und Spalte stehende Wert.

Oben abgebildet befindet sich die S-Box 1. Beispielsweise kommen die 6 Bits 101110 (Dezimal 46) im Laufe der Bearbeitung des DES in diese S-Box. Das erste Bit ist 1, das letzte die 0. 10 als Dualzahl hat dezimal den Wert 2. Also kommt die Zeile mit der Nummer 2 zur Anwendung.

Die restlichen Bits sind 0111. Als Dezimalzahl gelesen ergeben diese Bits den Wert 7. Das Ergebnis der S-Box 1 ist also in Zeile 2, Spalte 7 zu finden: 11. Binär liefert die S-Box also 1011 als Ergebnis.

Die S-Box-Substitution ist der wichtigste Teil des DES. Während alle anderen Operationen linear sind und sich leicht analysieren lassen, sorgen die S-Boxen mit ihrer Substitution für Nicht-Linearität. Beachten Sie, dass in der S-Box 1 (wie in den anderen 7 S-Boxen) in jeder Zeile die Werte 0 bis 15 jeweils nur einmal, aber in zufälliger Reihenfolge auftauchen.

Die Wiederholung dieser Vorgänge in jeweils 16 Runden führt dazu, dass sich der Zustand eines Bits, das an einer Stelle im Datenblock von einer Funktion geändert wurde, gleich einer Kettenreaktion auf möglichst viele andere Bits auswirken kann. Diese Kettenreaktion (Kaskadeneffekt) ist in der Kryptografie absolut erwünscht – so ist gewährleistet, dass nach Durchlaufen aller Runden jedes Bit des Schlüssels die Chance hatte, auf jedes Datenbit Einfluss zu nehmen. Das Ergebnis ist in so einem Fall ein Block aus scheinbar zufällig gesetzten Bits.

DES entschlüsseln

Soll ein mit DES verschlüsselter Datenblock wieder entschlüsselt werden, so wird derselbe Algorithmus durchlaufen. Der einzige Unterschied ist, dass die Runden in umgekehrter Reihenfolge, also von 16 bis 1, ausgeführt werden.

Dieses Rückwärts-Durchlaufen des DES-Algorithmus führt dazu, dass bei Eingabe des richtigen Schlüssels auch der Klartext wieder sichtbar wird.

Zukunft von DES

DES gilt heute als überholt. Die inzwischen aufgehobene Exportbeschränkung für kryptografische Verfahren und Produkte, die als „starke Kryptografie“ betrachtet wurden, und die Tatsache, dass DES im Rahmen eines internetbasierten Brute-Force-Angriffs geknackt wurde, machten den Bedarf für andere Verfahren deutlich.

War in den 60er-Jahren noch nicht ausreichend Rechenleistung vorhanden, um für einen Brute-Force-Angriff den Schlüsselraum von 2^{56} in ausreichender Zeit durchsuchen zu können, so ist dies heutzutage aufgrund der gestiegenen Rechenleistung und der Möglichkeit, die Brute-Force-Rechenleistung auf viele Rechner aufzuteilen, kein größerer Aufwand mehr.

Das Problem der Schlüssellänge wurde, bis ein neuer Algorithmus standardisiert wurde, zwischenzeitlich dadurch gelöst, dass die Daten mit DES dreimal hintereinander verschlüsselt wurden und dabei jedes Mal ein anderer 56-Bit-Schlüssel benutzt wurde. Nominell beträgt also die Schlüssellänge dieses als Triple DES (3DES) bekannten Verfahrens 168 Bits. Das Maß an Sicherheit ist jedoch nicht das Dreifache, sondern entspricht eher einer Schlüssellänge von 112 Bits. Aufgrund der Tatsache, dass beim 3DES der DES dreimal für jeden Datenblock aufgerufen wird, benötigt dieser auch das Dreifache an Rechenzeit.

Neben DES wurden einige weitere Algorithmen entwickelt, die das Problem des in die Jahre gekommenen DES lösen sollten. CAST, IDEA und Blowfish sind nur drei der zahlreichen Algorithmen, die neben vielen anderen auf dem Markt Beachtung gefunden haben.

AES – Advanced Encryption Standard

1997 forderte das NIST die Öffentlichkeit wiederum auf, Vorschläge für einen neuen Verschlüsselungsalgorithmus – den zukünftigen Advanced Encryption Standard – abzugeben. Der neue Standard sollte wie DES eine symmetrische Blockchiffre sein. Er musste Datenblöcke von 128 Bits verarbeiten und mit variablen Schlüssellängen von mindestens 128, 192 und 256 Bits arbeiten können. Die Implementierung sollte sowohl in Hardware als auch in Software möglich sein.

Aus vielen eingereichten Vorschlägen, unter denen sich auch der von der Deutschen Telekom entwickelte Algorithmus MAGENTA befand, wurden vom NIST fünf Finalisten ausgewählt:

- MARS (IBM)
- RC6 (RSA Laboratories)
- Rijndael (Joan Daemen und Vincent Rijmen)
- SERPENT (Ross Anderson, Eli Biham und Lars Knudsen)
- TWOFISH (Bruce Schneier und John Kelsey)

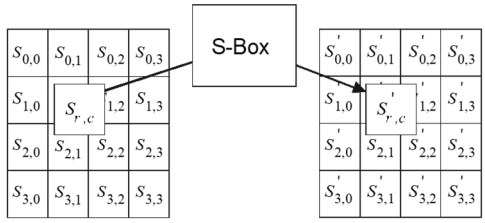
Am 2. Oktober 2000 stellte das NIST den Gewinner der Ausschreibung und somit den zukünftigen AES-Algorithmus vor: den aus Belgien stammenden Rijndael. Am 26. November 2001 wurde der AES als FIPS 197 verabschiedet (FIPS = Federal Information Processing Standards).

Wie die meisten Verschlüsselungsalgorithmen arbeitet Rijndael mit mehreren Runden, die Anzahl variiert aber mit der verwendeten Schlüssellänge zwischen 10, 12 und 14 Runden. Rijndael bearbeitet die 128 Bits als Matrix aus 16 Bytes, an denen pro Runde vorgegebene Substitutionsoperationen, Verschiebeoperationen, Spaltenoperationen und die Addition eines errechneten Rundenschlüssels vorgenommen werden.

Im Gegensatz zu DES betrachtet AES die zu verarbeitenden Daten nicht als Bitblöcke, sondern organisiert die Daten für die Verschlüsselung byteweise in einem State (Zustand). Soll ein 128-Bit-Block verschlüsselt werden, so wird dieser in 16 Bytes aufgeteilt. Diese 16 Bytes werden in eine 4-x-4-Matrix aufgeteilt, die den State darstellt.

Die Verschlüsselungsfunktion des AES besteht nun darin, in mehreren Runden durch Substitution, Verschiebung, Multiplikation und Addition von Rundenschlüsseln den State zu manipulieren.

Zu Beginn einer Runde werden die einzelnen Bytes des States durch die Funktion **ByteSub** (Byte Substitution) anhand einer S-Box durch andere Werte ersetzt.



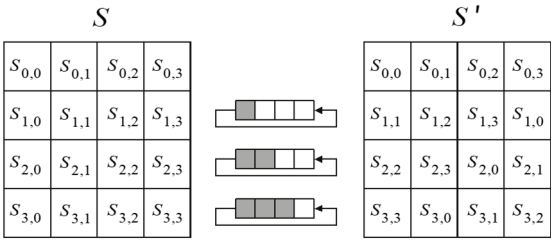
ByteSub anhand einer S-Box

Die Besonderheit beim AES im Gegensatz zum DES ist, dass die S-Box beim AES die einzige Tabelle ist, die der Algorithmus für seine Arbeit benötigt. Die Speicherung der Tabelle (und der inversen Tabelle für die Entschlüsselung) wäre sogar ganz vermeidbar, da die enthaltenen Werte auf einer Multiplikationsoperation basieren.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0e	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1c	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

AES S-Box

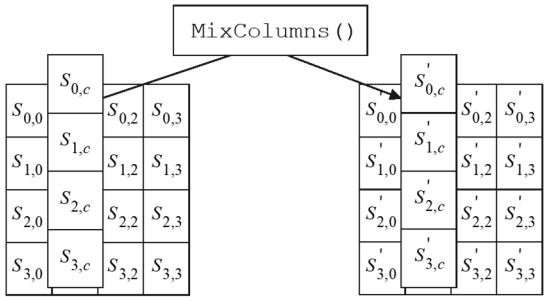
Nach dem **ByteSub** werden die Bytes durch **ShiftRow** zeilenweise um 0, 1, 2 oder 3 Stellen nach links verschoben. Die Bytes, die links aus der Matrix herausfallen, werden auf der rechten Seite wieder eingefügt.



ShiftRow

Anschließend wird spaltenweise die Funktion **MixColumns** durchgeführt. Um die Spalten zu „mischen“, d. h., um dafür zu sorgen, dass jedes Byte einer Spalte Gelegenheit hat, auf jedes andere Byte derselben Spalte einen Einfluss auszuüben, wird jede Spalte als Vektor betrachtet und mit einer fest vorgegebenen Matrix multipliziert.

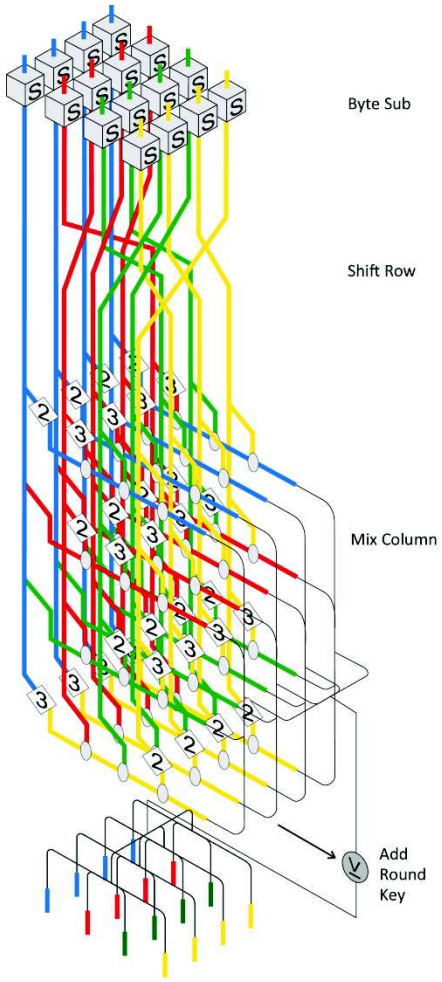
$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$



MixColumns

Abschließend wird der gesamte State bitweise modulo 2 mit dem Rundenschlüssel addiert. Der Rundenschlüssel für jede der 10 bis 14 nötigen Runden wird basierend auf dem Gesamtschlüssel vom AES errechnet. Sollten weitere Runden folgen, so beginnt AES wieder bei ByteSub.

Die Grafik rechts veranschaulicht den Ablauf einer AES-Runde dreidimensional.



AES – eine Runde

Verbreitung und Sicherheit von AES

Seit Oktober 2000 ist AES offizieller Nachfolger von DES und 3DES. Anwendungsfelder:

- WLAN mit WPA2/WPA3
- SSHv2/OpenSSH
- IPsec
- SFTP/FTPS/LFTP/OFTP
- Skype
- Dateisystemverschlüsselung (z. B. Apple macOS, ab Windows XP SP1)
- Dateikomprimierung (z. B. 7-Zip, RAR)
- PGP & GnuPG
- https usw.

Seine gründliche Analyse und die Tatsache, dass bisher keine Schwachstellen aufgedeckt wurden, fördern das Vertrauen in den neuen Algorithmus.

Während DES zwar 2^{56} mögliche Schlüssel akzeptiert, können nicht alle davon ohne Bedenken benutzt werden (sogenannte schwache Schlüssel). Bei AES sind derartige Schlüssel nicht bekannt. Zusätzlich hat der Anwender von AES die Auswahl aus mindestens 2^{128} verschiedenen Schlüsseln.

Der 1999 im Laufe eines Wettbewerbs mit verteilter Rechenleistung geknackte DES-Schlüssel wurde mit einer Brute-Force-Leistung von ca. 199.000.000.000 getesteten Schlüsseln pro Sekunde in ca. 22 Stunden gefunden.

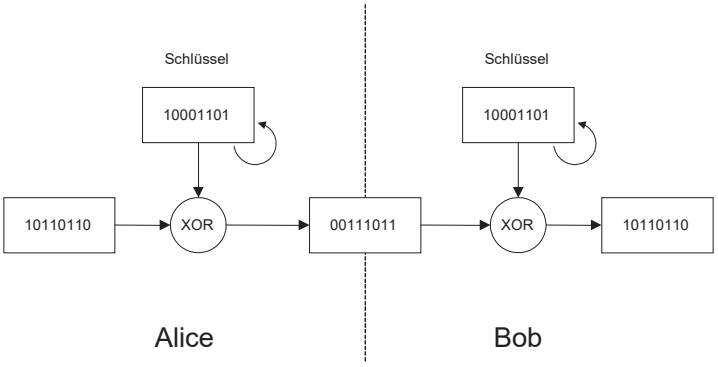
Hätte jemand dieselbe Rechenleistung zur Verfügung, um sämtliche 128-Bit-Schlüssel für AES durchprobieren zu können, so würde dies $1{,}709 * 10^{27}$ Sekunden dauern. Das sind $5{,}42 * 10^{19}$ Jahre. Das wäre also um ungefähr 387 Milliarden Mal länger als das zurzeit angenommene Alter des Universums (14 Mrd. Jahre = $14 * 10^{9}$).

RC4

RC4 wurde 1987 von Ronald L. Rivest entwickelt und wird heute nur noch bei PPTP eingesetzt. Im Gegensatz zu den bisher vorgestellten Algorithmen ist der RC4-Algorithmus keine Block-, sondern eine Stromchiffre. Stromchiffren eignen sich besonders gut, wenn eine blockweise Verschlüsselung der Daten (mit einhergehendem Zeitaufwand bzw. Latenzzeit) nicht geeignet ist, da die zu verschlüsselnden Daten zu einem (Echtzeit-)Datenstrom gehören.

Eine Stromchiffre betrachtet den Datenstrom bitweise und verschlüsselt diesen, indem der Datenstrom mit den Bits des Passwortstromes verknüpft wird (XOR).

Die Kernaufgabe einer Stromchiffre ist es, aus dem vorgegebenen Schlüssel, der für beide Parteien identisch ist, einen beliebig langen Bitstrom zu erzeugen, der pseudozufällig ist.



Stromchiffre

Die Verknüpfung der Schlüsselbits mit den Klartextbits macht den Chiffretext für Eve unlesbar, weil die Schlüsselbits scheinbar zufällige Modifikationen an den Datenbits vornehmen. Da Bob den korrekten Schlüssel besitzt, kann er auf seiner Seite denselben Pseudo-Zufallsprozess starten. Es wird also genau dieselbe Folge an Einsen und Nullen generiert, die mit dem Chiffretext verknüpft wird. Durch die Eigenschaften des XOR wird auf diese Weise der Klartext wieder sichtbar.

Die Qualität einer Stromchiffre hängt von der Qualität der Zufallszahlen ab, die der Generator basierend auf einem eingegebenen Schlüssel generiert. Ist die Bitfolge nicht vorhersehbar und wiederholt sie sich nicht, so ist die Chiffre sicher.

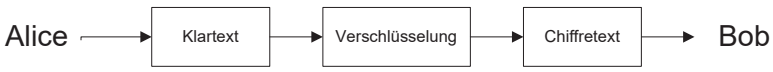
Stromchiffre weist eine hohe Ähnlichkeit zur Vernam-Chiffre auf – das ist kein Zufall. Der Unterschied zwischen Vernam- und der Stromchiffre liegt lediglich darin, dass bei Vernam das Passwort in der Länge der Nachricht vor der Übertragung ausgetauscht werden muss. Bei einer Stromchiffre muss vorher nur ein kurzes Passwort ausgetauscht werden, das dann zur Generierung eines beliebig langen binären „Vernam-Passwortes“ dient.

RC4 ist ein von Ron Rivest in den 80er-Jahren entwickelter Stromchiffre-Algorithmus, der inzwischen schon gebrochen wurde, aber dennoch sehr weit verbreitet ist. Die häufigste Anwendung ist die Verschlüsselung des Datenstromes im Webbrowser, wenn eine gesicherte Verbindung mit SSL hergestellt wird. RC4 wurde von Herstellern in Amerika gerne in Software implementiert, da RC4 zu den Zeiten der Kryptoexport-Kontrolle mit einer Schlüssellänge von 40 Bit ohne besondere Genehmigung durch die Regierung exportiert werden durfte.

Operationsmodi

Generell unterscheidet man, in welchen Operationsmodi kryptografische Algorithmen auf Daten angewandt werden. Der einfachste dieser Modi ist der ECB-Modus.

Beim **ECB**, dem **Electronic Codebook**, wird ein Datenblock gemäß dem Algorithmus mit dem gleichen Schlüssel chiffriert und an den Empfänger geschickt. Anschließend wird der nächste Block verschlüsselt usw.



Electronic Codebook

Die Datenblöcke werden also unabhängig voneinander verschlüsselt – analog zu einem Codebuch, in dem nachgeschlagen werden muss, welche Wörter der Nachricht durch welche Chiffrewörter ersetzt werden müssen.

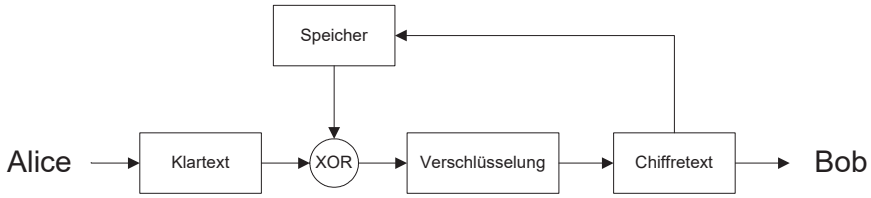
Diese Methode hat einen gravierenden Nachteil: Identische Passagen der Klartextblöcke werden mit stets den gleichen Chiffretextblöcken codiert.

Dies würde Eve wiederum erlauben, eine Häufigkeitsanalyse zu erstellen. Ist die Art der übermittelten Daten bekannt (Text, Bilddokumente, Audio etc.), so können, ähnlich wie bei der Häufigkeitsverteilung der Buchstaben in natürlicher Sprache, Rückschlüsse auf den Klartext gezogen werden.

Die anderen Betriebsmodi führen aus diesem Grund eine Rückkopplung der Art ein, dass ein verschlüsselter Datenblock die Verschlüsselung für nachfolgende Datenblöcke beeinflusst. Auf diese Weise erzeugen auch lange identische Datenblöcke im Klartext keine identischen Datenblöcke im Chiffretext.

Beim **CBC**, dem **Cipher Block Chaining**, wird ein Klartextblock vor der Verschlüsselung mit dem vorherigen Chiffretextblock XOR-verknüpft. Der nach der Verschlüsselung erhaltene Chiffretextblock wird gespeichert.

Mit diesem wird dann der nächste Klartextblock mit dem Chiffretextblock XOR-verknüpft usw.

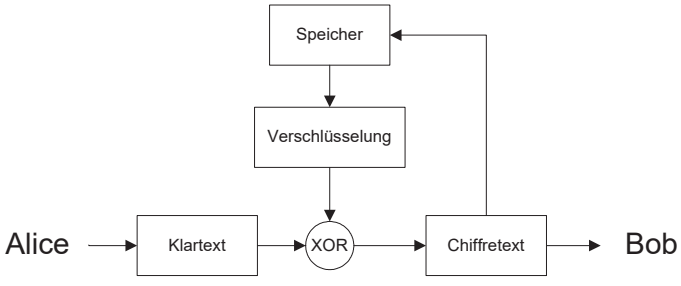


Cipher Block Chaining

Da bei der Verschlüsselung des ersten Blockes noch kein vorhergehender Block gespeichert wurde, kommt hier ein IV, ein Initialisierungsvektor, zum Einsatz.

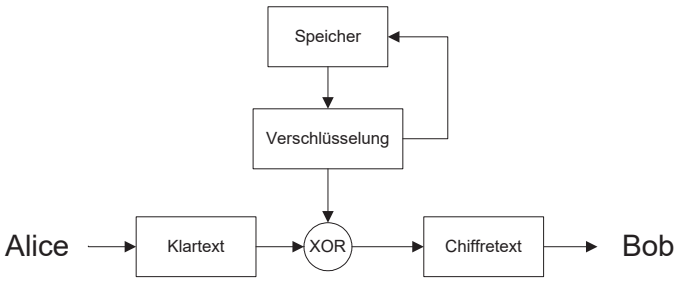
Im **CFB** Mode, dem **Cipher Feedback**, wird zuerst der IV verschlüsselt und dieser dann per XOR mit dem Klartext verknüpft. Das Ergebnis wird gespeichert und beim nächsten Block verschlüsselt, bevor es per XOR mit dem neuen Block verknüpft wird usw. CBF eignet sich gut für Fälle, in denen Daten verschlüsselt werden sollen, die kleiner sind als die vom Algorithmus unterstützte Blockgröße.

Soll jeweils nur ein einzelnes Byte verschlüsselt übertragen werden, die Blockgröße ist jedoch 8 Bytes (64 Bit), so wird der IV aus dem Zwischenspeicher verschlüsselt, aber die XOR-Operation nur mit dem 1. Byte des Ergebnisses am Klartextbyte durchgeführt. Das Ergebnisbyte wird dann am Ende des Zwischenspeichers angehängt, während das erste Byte gelöscht wird. Nachfolgende Bytes werden ebenfalls verschlüsselt, indem der Zwischenspeicher verschlüsselt wird, aber nur das erste Byte mit dem Klartext mit dem Chiffretextblock XOR verknüpft wird usw.



Cipher Feedback

Der **Output Feedback** Mode (**OFB)** ist dem CFB Mode sehr ähnlich, nur wird hier nicht das Ergebnis in die Warteschlange gestellt, sondern der gerade zuvor verschlüsselte Zwischenspeicher selbst. In diesem Modus bildet die Kryptografieeinheit einen in sich geschlossenen Kreis, der vom Schlüssel angestoßen wird. Die Verschlüsselung der Klartextdaten selbst findet über die Verknüpfung mit XOR statt. Mit diesem Betriebsmodus ist ein beliebiger kryptografischer Algorithmus als Stromchiffre einsetzbar.



Output Feedback

Ein auf diese Weise betriebener guter kryptografischer Algorithmus ist so auch als guter Zufallszahlengenerator verwendbar.

10.4 Übung

Fragen zur symmetrischen Kryptografie

Übungsdatei: -- **Ergebnisdatei: *uebung10.pdf***

1. Welche Aussage trifft auf die symmetrische Verschlüsselung zu?

a		Wie bei der asymmetrischen Verschlüsselung existieren ein Private Key und ein Public Key.
b		Die symmetrische Verschlüsselung beruht auf dem Prinzip, dass Sender und Empfänger über den gleichen geheimen Schlüssel verfügen.
c		Der Austausch des geheimen Schlüssels muss selbst über einen verschlüsselten Kanal getätigt werden.