



Vorlesung Security by Design (SbD)

Prof. Dr. Hannes Federrath

Sicherheit in verteilten Systemen (SVS)

<http://svs.informatik.uni-hamburg.de>

Wegweiser durch die Vorlesung

- Kryptographie: Vertiefung
- Public Key Infrastructures (PKI)
- Sniffing, Spoofing, Denial of Service, Internet of Things and Security
- Mobile Security

Die bereitgestellten Kursmaterialien dienen ausschließlich dem persönlichen Gebrauch. Die Veröffentlichung, Vervielfältigung, Verbreitung oder Weitergabe, auch auszugsweise, ist nur mit schriftlicher Genehmigung des Verfassers erlaubt.

Organisatorisches

Hannes Federrath · Monina Schwarz · Pascal Wichmann

Sicherheit in verteilten Systemen (SVS)

<http://svs.informatik.uni-hamburg.de>

Ziele der vorlesungsbegleitenden Übungen

- **Verständnisprobleme lösen**
 - Offene Fragen der Vorlesung können im kleineren Kreis geklärt werden.
- **Vertiefung und Anwendung des Vorlesungsstoffs**
 - In der Vorlesung wird teilweise nur eine Auswahl des Stoffs präsentiert.
 - Bearbeitung von Aufgaben mit Bezug zur Vorlesung
 - Bearbeitung von Aufgaben, die über die Vorlesung hinausgehen
- **Vorstellung und Diskussion der Lösungen innerhalb der Übungsgruppe**
 - Wissenslücken können nur beseitigt werden, wenn sie bekannt sind.
 - Aktive Beteiligung ist notwendig und sinnvoll.
- **Übungsstoff ist auch klausurrelevant!**

Übungleistung

- Übungleistung wird online in Moodle erbracht
 - <https://lernen.min.uni-hamburg.de/course/>
 - Security by Design WS20/21
 - Einloggen mit Kennung und Passwort der Uniseiten
 - Einschreibeschlüssel: virtuell2020

- Bewertung
 - Tests müssen mit mindestens 25% der Punkte bestanden werden
 - Ergebnisse werden direkt angezeigt

Übungsaufgaben

- Übung bietet verschiedene Komponenten
 - Online-Test in Moodle: Einfaches Grundverständnis
 - Zweiwöchige Übungsblätter: Vertiefung der Inhalte

- Zeitplan für die Übungen
 - Freischaltung der Übungsblätter am Montag
 - Bearbeitungszeit je Blatt verschieden (in der Regel zwei Wochen)
 - Besprechungstermine auf dem Blatt angegeben

Übungstermine

■ Durchführung

- Während der Bearbeitungszeit: Konsultationstermine
 - Gelegenheit zum Stellen von Fragen
- Nach Ablauf der Bearbeitungszeit: Gemeinsame Besprechung und Diskussion der Übungen
 - Hintergründe zu den Übungsaufgaben werden erklärt

Bearbeiten der Übungen hilft dabei, den Vorlesungsstoff besser zu verstehen.

■ Es gibt zwei Übungsgruppen.

- Termine für das Wintersemester 2020/21:
 - Montag 10-12 Uhr
 - Dienstag 16-18 Uhr
- Konsultationstermin im Wintersemester 2020/21:
 - Dienstag 16-18 Uhr

Änderungen vorbehalten (siehe Moodle)

■ Formalia

- 60 Minuten Dauer
- mit Unterlagen («open book«)
- nicht-programmierbarer Taschenrechner erlaubt

■ Inhalte

- Inhalte der Vorlesung
- Inhalte der Übung

■ Klausurtermine zu finden unter

- <https://www.inf.uni-hamburg.de/studies/orga/dates.html>

- Bitte prüfen Sie bei Bedarf die Barrierefreiheit des Zugangs zu den Räumen und nehmen frühzeitig Kontakt zu uns auf, sollten Sie hierbei Probleme feststellen.

Probeklausur

- Probeklausur wird an einem der letzten Vorlesungstermine angeboten
 - per Videokonferenz
 - 60 Minuten Bearbeitungsdauer
 - Besprechung direkt im Anschluss
- Inhalte
 - entspricht in Art und Aufbau einer tatsächlichen Klausur

Kommunikation zur Vorlesung und Übung

■ Infos wo?

- Vorlesungsfolien, Ankündigungen, Verschiebungen in STiNE und/oder im Moodle
- Übungsblätter und Online-Tests in Moodle

■ Fragen?

- Zur Vorlesung: Prof. Dr. Hannes Federrath fragen
- Zu den Übungen: Übungsleiter des Vertrauens fragen
- Fragen Sie gerne auch per E-Mail!

■ Zur Vorlesung *und* zur Übung angemeldet (in STiNE)?

- Wenn nicht: umgehend nachholen!

Kryptographie: Vertiefung

Definition Konzelationssystem

■ Seien

- A und B Alphabete und K eine endliche Menge
- Klartext: $m \in A^n = M$ Schlüsseltext: $c \in B^l = C$ mit $n, l \in \mathbb{N}$

l/n : Expansionsfaktor
 $l/n=1$ längentreue Chiffre

- Dann ist eine Kryptofunktion eine Abbildung
derart, dass die Abbildung
definiert durch
für alle $k \in K$ injektiv ist.

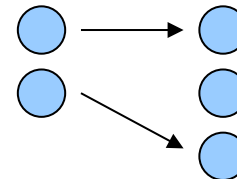
$$\begin{aligned} e: A^n \times K &\rightarrow B^l \\ e_k: A^n &\rightarrow B^l \\ e_k(m) &:= e(m, k) \end{aligned}$$

- injektiv: f^{-1} ist rechtseindeutig
- rechtseindeutig (auch: partiell):
 $\forall x. \forall y. y': ((f(x)=y \wedge f(x)=y') \rightarrow (y=y'))$

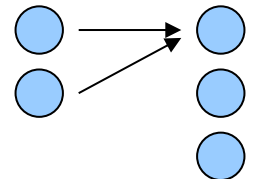
- Für jedes e_k existiert eine Umkehrfunktion $d_{k'}$.

■ Es gelten

- $c=e_k(m)$ und $m=d_{k'}(c)$, d.h. $m=d_{k'}(e_k(m))$ oder
- $c=e(m,k)$ und $m=d(c,k')$, d.h. $m=d(e(m,k),k')$



injektive Abbildung



nicht-injektive Abbildung

Klassische Chiffren: Systematik

■ Transpositionschiffren

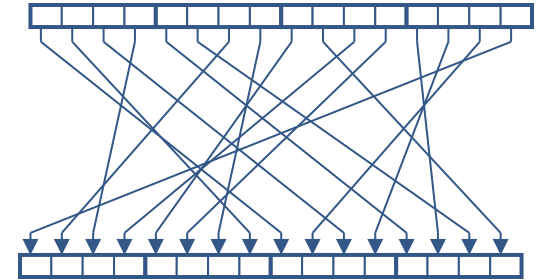
- Veränderung (Permutation) der Anordnung von Schriftzeichen

■ Substitutionschiffren

- Systematische Ersetzung von Schriftzeichen

■ Produktchiffren

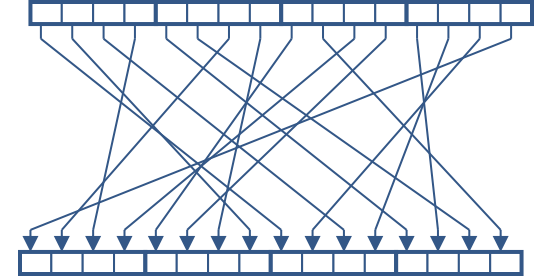
- Kombination von Transpositionen und Substitutionen
- Vorläufer der modernen symm. Kryptographie, bei denen Permutationen und Substitutionen (meist) iterativ angewendet werden



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

Klassische Chiffren: Konkrete Systeme

- **Transpositionschiffren**
 - Spalten-Transpositionen
 - freie Permutationen
- **Substitutionschiffren**
 - Schema von Polybios
 - Caesar-Chiffre
 - Vigenere-Chiffre
 - Vernam-Chiffre



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

■ Skytala

- ca. 2400 Jahre alte griechische Chiffre
- Zylinder mit gewickeltem Papierstreifen, schreiben, abwickeln
- Empfänger hat Zylinder mit gleichem Durchmesser



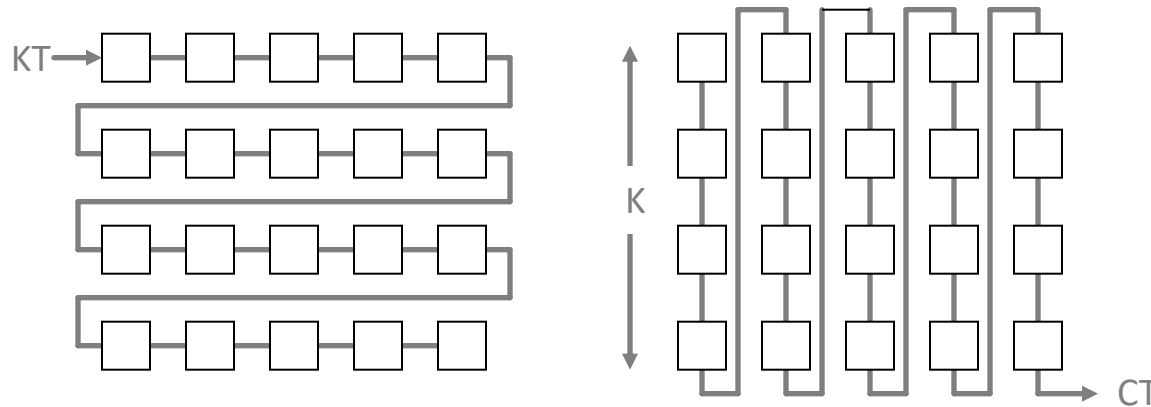
Bild: Wikipedia (CC BY-SA 3.0)

■ Kryptanalyse

- Heute: Durchprobieren (sehr kleiner Schlüsselraum)
- Statistische Analyse (Bigramme)

■ Skytala

- ca. 2400 Jahre alte griechische Chiffre
- Zylinder mit gewickeltem Papierstreifen, schreiben, abwickeln
- Empfänger hat Zylinder mit gleichem Durchmesser

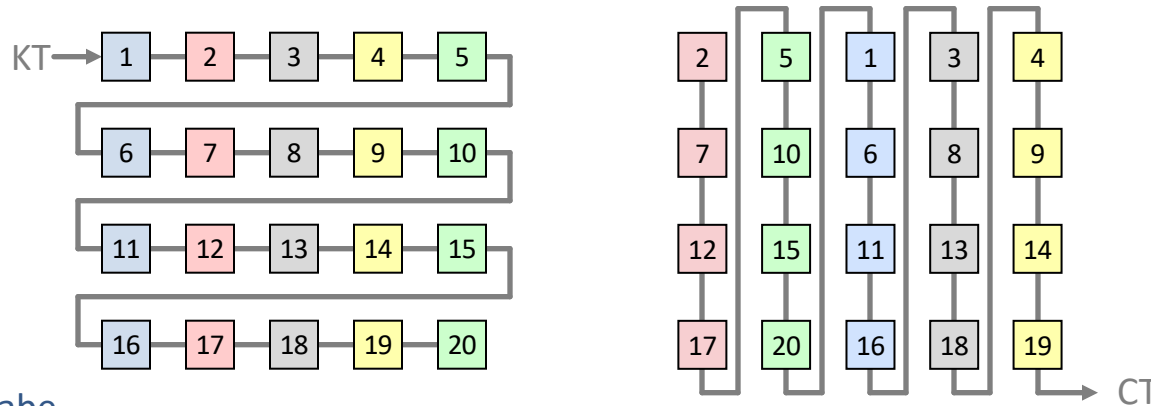


■ Kryptanalyse

- Heute: Durchprobieren (sehr kleiner Schlüsselraum)
- Statistische Analyse (Bigramme)

■ Variante

- zusätzlich die Spalten transponieren, d.h. deren Reihenfolge vertauschen bzw. permutieren



■ Übungsaufgabe

- Um welchen Faktor vergrößert sich der Schlüsselraum bei s Spalten?

Skytala: Statistische Kryptanalyse

■ Vorgehen

- Suche typ. Bigramme (z.B. EN, ER, CH, ...) und ermittle die Häufigkeit der Buchstabenabstände.
Beispiel:

■ Beispiel

- Klartext: VERSCHLUESSELNMACHTGROSSENSPASS

- k=5

VERSCHL
UESSELN
MACHTGR
OSSENSP
ASSXYZX

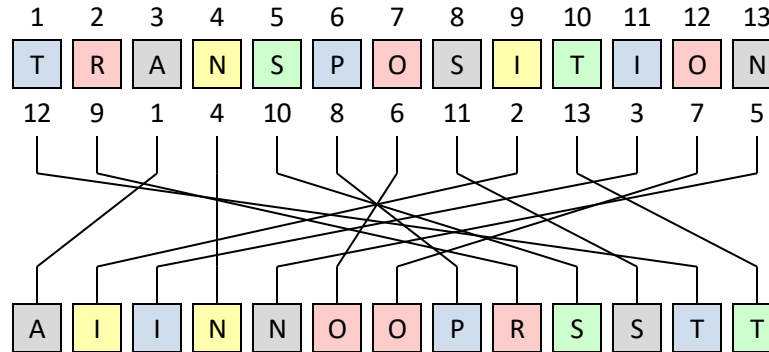
- Chiffretext: VUMOAEEASSRSCSSSSHEXCETNYHLGSZLNRPX

Bigramm	Chiffretext	Abstand/Häufigk.
EN:	VUMOAEEASSRSCSSSSHEXCETNYHLGSZLNRPX:	2/1, 5/1, ...
ER:	VUMOAEEASSRSCSSSSHEXCETNYHLGSZLNRPX:	5/1, 4/1, ...
EI:	VUMOAEEASSRSCSSSSHEXCETNYHLGSZLNRPX:	0
CH:	VUMOAEEASSRSCSSSSHEXCETNYHLGSZLNRPX:	5/2, ...

- Abstand 5 kommt am Häufigsten vor,
- teste, ob Text in 5 Zeilen sinnvoll → gebrochen

Freie Permutationen

- Idee
 - Zeichen werden nach einer Vorschrift vertauscht
- Beispiel

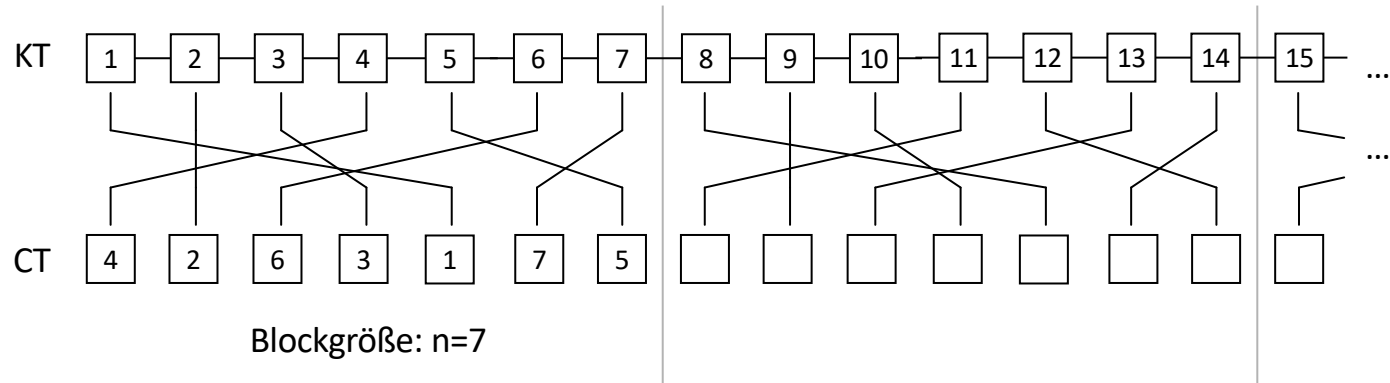


Zyklenschreibweise:
(1 12 7 6 8 11 3)
(2 9) (4) (5 10 13)

- Übungsaufgabe
 - Schreiben Sie eine Skytala mit 4 Zeilen und 3 Spalten in Zyklenschreibweise.

Block-Transposition

- Klartextzeichen in Blöcken fester Länge transponieren



- Übungsaufgabe
 - Größe des Schlüssels?

- Vorgehen: Rekonstruktion von Bi- und Trigrammen
- Beispiel: Chiffretext in vermuteter Sprache Deutsch; Test auf typ. Bi und Trigramme wie ER und SCH

Suche nach SCH im Chiffretext: CESVLRHEESUUSLLGANOSGMIHRSTU

Test auf Blocklänge 5:

Test auf Blocklänge 6:

Test auf Blocklänge 7:

Feststellung: SCH passt
nicht in Block

1234567 1234567 1234567 1234567

CESVLRH EESUUSL LGANOSG MIHRSTU

SCH SEL ALG HMU

Suche nach ER im Chiffretext:

CESVLRH EESUUSL LGANOSG MIHRSTU

ER ES GS IT

Transposition ergibt auch
in den anderen Blöcken
halbwegs sinnvolle Bi- und
Trigramme

Weiteres Probieren führt zu:

CESVLRH EESUUSL LGANOSG MIHRSTU

5241736 5241736 5241736 5241736

VERSCHL UESSELU NGSALGO RITHMUS

Entschlüsselungsschlüssel lautet:

(1 5 7 6 3 4) (2)

Klassische Substitutionschiffren

- Monoalphabetische Substitution

- Jedem Zeichen bzw. jeder Zeichenfolge über A ist eindeutig ein Zeichen bzw. eine Zeichenfolge über B zugeordnet.

- Polyalphabetische Substitution

- Jedem Zeichen bzw. jeder Zeichenfolge über A ist eindeutig ein Zeichen bzw. eine Zeichenfolge über B_1, B_2, \dots, B_n zugeordnet.

- Monographische Substitution

- Es werden einzelne Zeichen ersetzt.

- Polygraphische Substitution

- Es werden Zeichenfolgen ersetzt.

Schema von Polybios

■ Def.

- $A = \{A:Z\}$
- $B = \{ (i,j) \mid i,j \in \{1:5\} \}$
- e/d:

i \ j	j →				
	1	2	3	4	5
1 ↓	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V	W	X/	Z

Beispiel:

Chiffretext: 223515452315

Klartext:

■ Eigenschaften

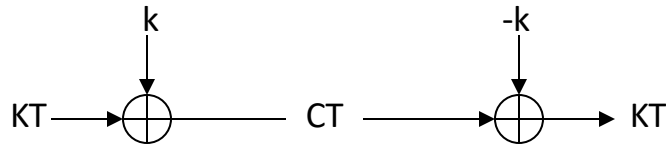
- monoalphabetische/monographische Substitution
- arbeitet »schlüssellos«

■ Ableitungen

- Tabelle mit Zeichen (Freimaurer-Chiffre, Friedhofschiffre)

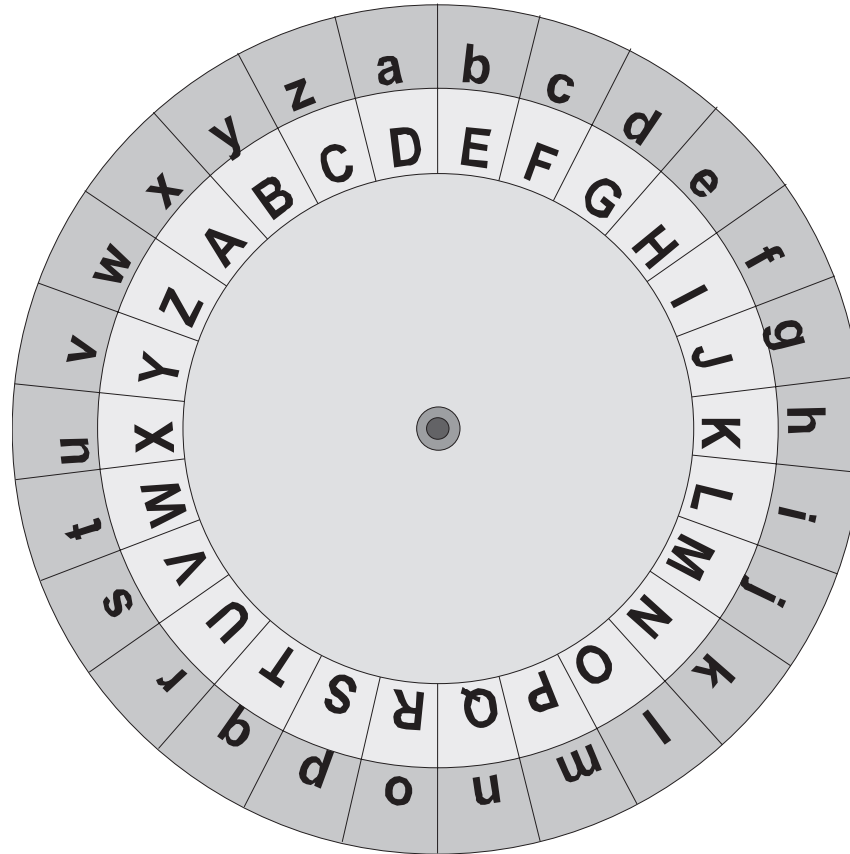
Verschiebechiffre

- auch: Caesar-Chiffre
 - Caesar, röm. Kaiser und Feldherr (100-44 v.Chr)
- Def.
 - $A=B=\{A:Z\}$
 - $K=\{A:Z\}$ oder allg. $K=\{0:n-1\}$ mit $n \leq \text{card}(\{A:Z\})$
 - e: $c=(m+k) \bmod n$
 - d: $m=(c-k) \bmod n$



$$(x+y) \bmod 26$$

		Klartext																									
Schlüssel	A	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																									
	A	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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



Verschiebechiffre

- auch: Caesar-Chiffre

- Caesar, röm. Kaiser und Feldherr (100-44 v.Chr)

- Def.

- $A=B=\{A:Z\}$
 - $K=\{A:Z\}$ oder allg. $K=\{0:n-1\}$ mit $n \leq \text{card}(\{A:Z\})$
 - e: $c=(m+k) \bmod n$
 - d: $m=(c-k) \bmod n$

- Eigenschaften

- monoalphabetische/monographische Substitution
 - additive Chiffre
 - Buchstabenhäufigkeiten bleiben erhalten und bilden Ansatz zur Kryptanalyse

»Historisches« Beispiel für Verschiebechiffre – Spiegel Plus

Quelle:

<http://andreas-zeller.blogspot.de/2016/06/spiegel-online-nutzt-unsichere-casar.html>

The screenshot shows a web browser window with the URL `spiegel.de`. The article text is partially visible, with several words highlighted in red boxes: **SPIEGEL:**, **Cryan**, **Dszbo:**, and **TQJFHFM:**. A semi-transparent overlay box is positioned over the text, containing the following information:

Cryan	SPIEGEL	Klartext
Dszbo	TQJFHFM	Schlüsseltext
ABCDEFGHIJKLMNOPQRSTUVWXYZ		Verschiebung um 1 Position
BCDEFGHIJKLMNOPQRSTUVWXYZA		

The article text continues with:

SPIEGEL: Wie wird die Deutsche Bank auf die Brexit-Entscheidung reagieren?

Dszbo: Ebt xjse ebwpo bciãohfo- xjf ejf Wfsiboemvohfo wfsmbvgo/ Xjs tjoe eb tfis gmfyjcfm- eb xjs tpxim jo Mpoepo bmt bvdi jo Gsbolgvsu tubsl wfsusufu tjoe/ Ebevsdi xfsefo xjs gýs fvspqãjtdif Voufsofinfo vn timer xjdiumhfs- hfsbef jo ejftfs Qibtf efs Votjdisifju bo efo Lbqjubmnãslufo/

TQJFHFM: Xfsefo Tjf Ufjmf eft Hftdiãgut bo efo Nbjo wfsmbhfsfo@

Dszbo: Tpmuf ft ubutãdimjdi {v fjofn Bvtusjuu nfjoft Ifjnbumboeft bvt efs FV lppnfo- eboo xjse ebt Mpoepo tdixãdifo voe Gsbolgvsu tuãslfo/ Xbt ebt bcfs hfobv gýs ejf Djuz voe gýs vot ifjäu- mãttu tjdì opdi ojdiu wpsifstbhfo/

Verschiebechiffre als Webanwendung

Cäsar Verschiebechiffre bzw. Cäsar Verschlüsselung

Original

Cryan; Das wird davon abhängen- wie die Verhandlungen verlaufen/ Wir sind da sehr flexibel- da wir sowohl in London als auch in Frankfurt stark vertreten sind/ Dadurch werden wir für europäische Unternehmen umso wichtiger- gerade in dieser Phase der Unsicherheit an den Kapitalmärkten/

SPIEGEL; Werden Sie Teile des Geschäfts an den Main verlagern@

Cryan; Sollte es tatsächlich {u einem Austritt meines Heimatlandes aus der EU kommen- dann wird das London

Verschiebung

1

Kodiert

Dszbo; Ebt xjse ebwpo bciäohfo- xjf ejf Wfsiboemvohfo wfsmbvgfo/ Xjs tjoe eb tfis gmfyjcfm- eb xjs tpxpim jo Mpoepo bmt bvdi jo Gsbolgvsu tubsl wfsusfufo tjoe/ Ebevsdi xfsefo xjs gys fvspqäjt dif Voufsofinfo vn timer xjdiujhfs- hfsbef jo ejftfs Qibtf efs Votjdifsifju bo efo Lbqjubmnäslufo/

TQJFHFH; Xfsefo Tjf Ufjmf eft Hftdiägut bo efo Nbjo wfsmbhfso@

Dszbo; Tpmuf ft ubutädimjdi {v fjoen Bvtusjuu nfjoft Ifjnbumboeft bvt efs FV lpnnfo- eboo xjse ebt Mpoepo

Methode:

Cäsar Verschiebechiffre

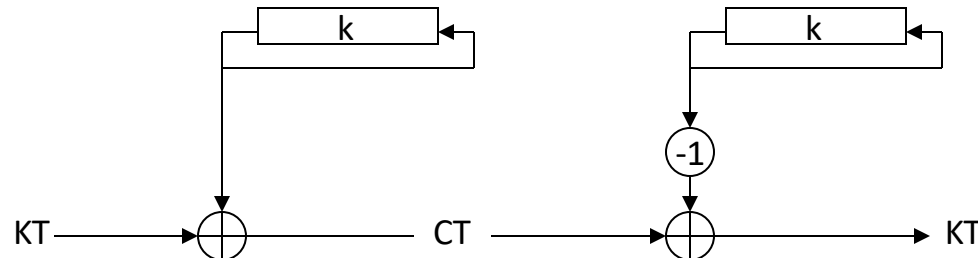
Hilfe: A-Z,a-z werden um die gewünschte Anzahl von Positionen im Alphabet zyklisch nach rechts oder links verschoben, alle anderen Zeichen bleiben unverändert. ROT13 ist eine Sonderform der Cäsar Verschiebechiffre mit einer Verschiebung um 13 Positionen. Die Umwandlung funktioniert in beide Richtungen. Bei einem Verschiebewert von '0' werden alle Verschiebemöglichkeiten von 1-25 ausgegeben.

Encode ▼ Decode ▲ ▼▲

<https://gc.de/gc/caesar/>

Vigenère-Chiffre

- nach: Blaise de Vigenère, 1586, französischer Kryptologe
- Idee
 - Gleiche Klartextzeichen auf unterschiedlichen Chiffretextzeichen abbilden, um Häufigkeitsanalyse zu erschweren (polyalphabetische Substitution)
- Def.
 - $A=B=\{A:Z\}$
 - $K=\{(k_1, k_2, k_3, \dots, k_r) \mid k_i \in \{A:Z\}, r \in \{1, 2, \dots\}\}$
 - r : Periodenlänge des Schlüssels
 - e, d : für jedes k_i analog Caesar-Chiffre



Beispiel: Verwendung des Vigenere-Tableaus

Schlüssel:

HUGO

Klartext:

VIGENERECHIFFRE

Verschlüsselung:

VIGENERECHIFFRE

HUGOHUGOHUGOHUG

CCM...

?

Entschlüsselung
entsprechend

		Klartext																											
Schlüssel	A	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		
	A	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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		

Vigenère-Chiffre

■ Vigenère-Chiffre: Beispiel

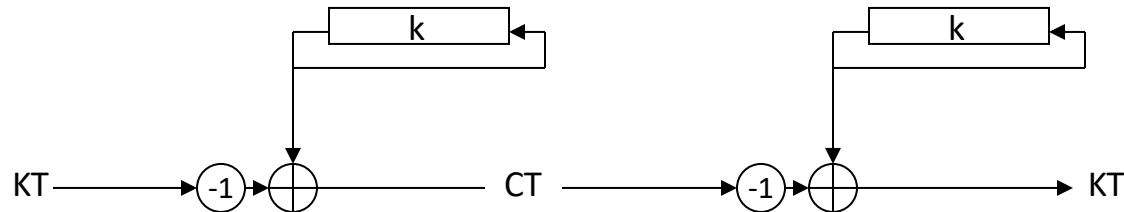
- VIGENERECHIFFRE = Klartext
- HUGOHUGOHUGOHUG = Schlüssel mit $r=4$
- CCM = Chiffretext
- Kryptanalyse:
 - Periodenlänge ermitteln, dann weiter wie Caesar-Chiffre

■ Variante: Autokey-Verfahren

- Klartext als Teil des Schlüssels verwenden:
 - VIGENERECHIFFRE = Klartext
 - HUGOVIGENERECHI = Schlüssel

Beaufort-Chiffre

- nach: Francis Beaufort (1857)
 - jedoch bereits 1710 von Giovanni Sesti vorgeschlagen
 - Variante der Vigenère-Chiffre
 - involutorische Chiffre ($e=d$)



- Beispiel für die $e=d$ -Eigenschaft
 - $c=(-1 \cdot m+k) \bmod n$ und $m=(-1 \cdot c+k) \bmod n$
 - $FED=m$, $k=3$
 - $FGA=c$
 - $FED=m$

$i=0$	1	2	3	4	5	6	($n=7$)
A	B	C	D	E	F	G	

Chiffrieren nach Beaufort mit Vigenere-Tableau

Das Chiffretextzeichen zum Klartextzeichen a ist durch die Zeile gegeben, die das Schlüsselzeichen z in der Spalte a enthält. (Fumy, S. 53)

Schlüssel:

HUGO

Klartext:

VIGENERECHIFFRE

Verschlüsselung:

VIGENERECHIFFRE

HUGOHUGOHUGOHUG

?

Entschlüsselung
entsprechend

		Klartext																									
Schlüssel	A	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
	A	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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

Vernam-Chiffre (One-Time-Pad)

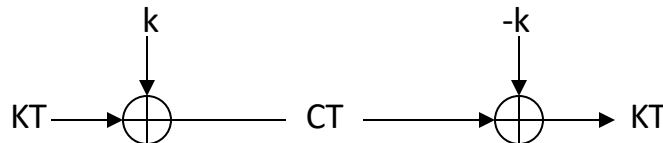
■ Def.

- $A=B=\{A:Z\}$
- $K=\{(k_1, k_2, k_3, \dots, k_l) \mid k_i \in \{A:Z\}, i=[1, l]\}$
- Die k_i werden unabhängig und zufällig erzeugt
- Der Schlüssel $k=(k_1, k_2, k_3, \dots, k_l)$ hat die gleiche Länge wie der Klartext $m=(a_1, a_2, a_3, \dots, a_l)$
- e: $c=(m+k) \bmod n$ (zeichenweise)
- d: $m=(c-k) \bmod n$

■ Beispiel

- KRYPTOMACHTSPASS = Klartext
- VABZEQTAWPNRTLKB = Schlüssel

■ Kryptanalyse: unmöglich, da informationstheoretisch sicher



Vernam-Chiffre (One-Time-Pad)

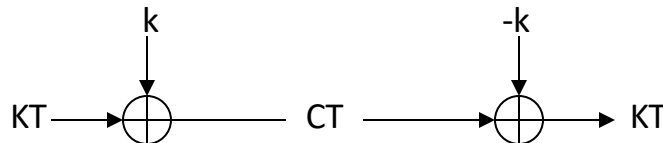
■ Informationstheoretisch sichere Verschlüsselung

- Egal, was der Angreifer a priori an Information über den Klartext hat, er gewinnt durch die Beobachtung des Schlüsseltextes keine Information hinzu.


$$\forall s \in S \exists const \in N \forall x \in X : |\{k \in K \mid k(x) = s\}| = const$$

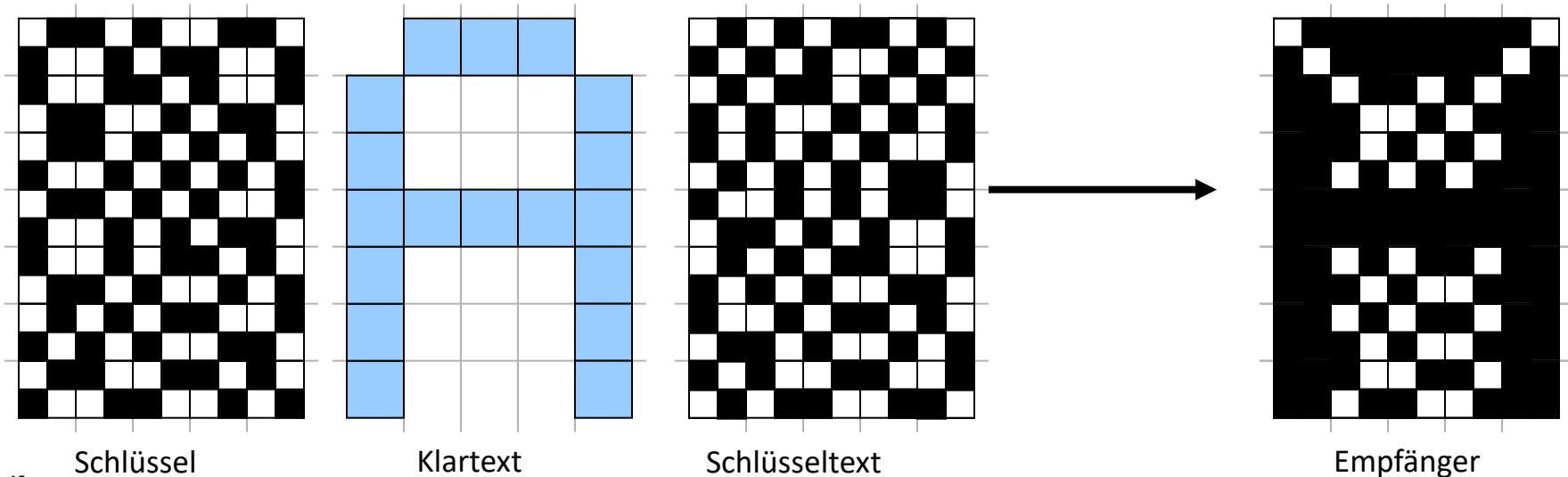
Für alle Schlüsseltexte s existiert eine konstante Anzahl von Schlüsseln k , die jeweils alle Klartexte x derart verschlüsseln, dass aus x jeder Schlüsseltext entstehen kann. $N = \{1, 2, 3, \dots\}$

- »Hinter jedem Schlüsseltext kann sich jeder Klartext verbergen.«



Visuelle Kryptographie

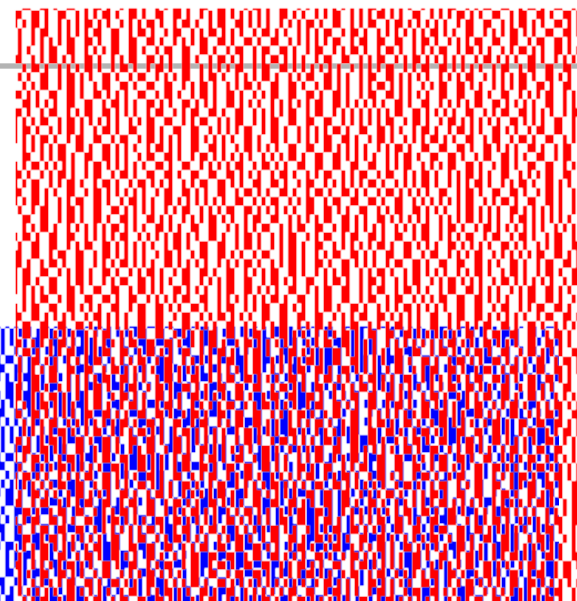
- Symmetrisches Verfahren: Sender und Empfänger erzeugen sich Zufallsmuster aus zwei komplementären Basismustern:  Zufallsmuster nur für genau eine Botschaft verwenden!
- Visuelle Botschaft:
 - Sender verwendet negiertes Muster für schwarze Bildpunkte
 - Für »weiße« Bildpunkte: keine Veränderung



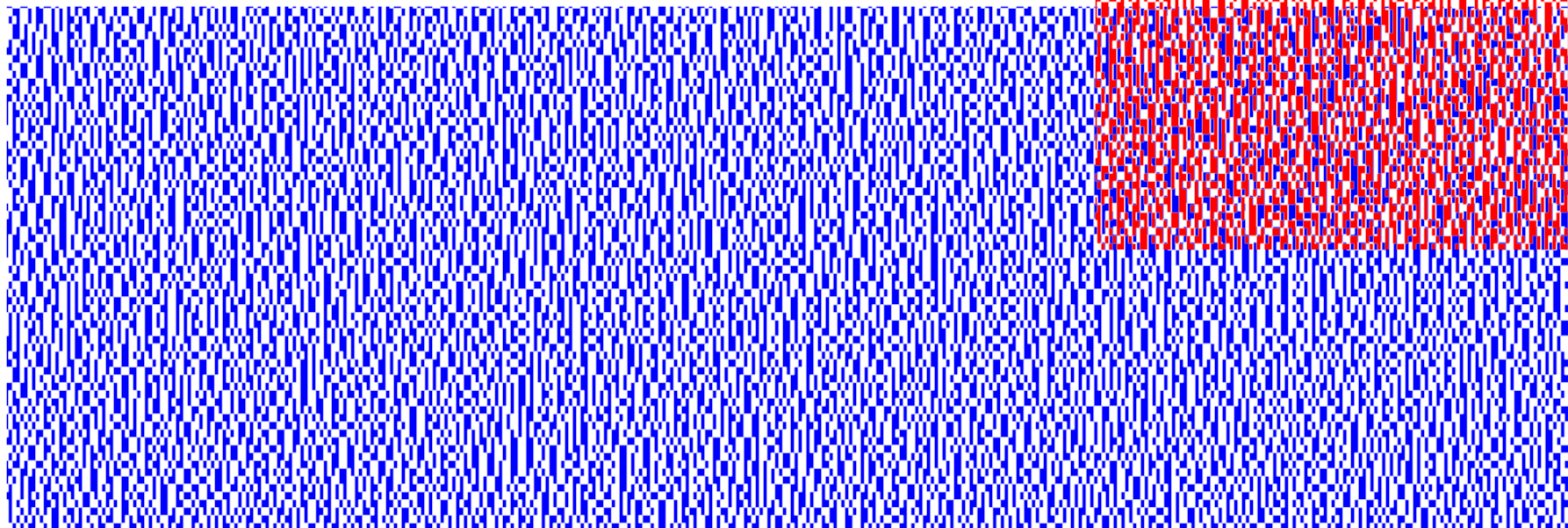
Visuelle Kryptographie: Demo



Schlüssel



Botschaft



Moderne Kryptosysteme

■ Symmetrische Systeme

- One-Time-Pad (mod 2)
- Symmetrische Authentifikationscodes
- DES (Data Encryption Standard)
- IDEA (International Data Encryption Algorithm)
- AES (Advanced Encryption Standard)

■ Praktischer Einsatz

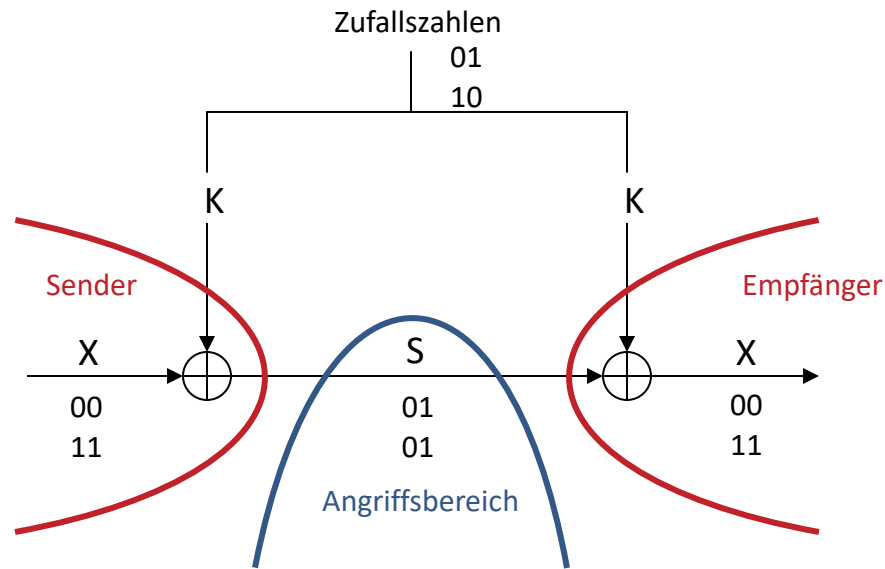
- Betriebsarten von Blockchiffren

■ Asymmetrische Systeme

- Diffie-Hellman-Key-Exchange
- ElGamal Kryptosystem
- RSA zur Konzelation und Signatur
- Blinde Signaturen mit RSA
- Kryptosysteme auf Basis elliptischer Kurven

One-Time-Pad (mod 2)

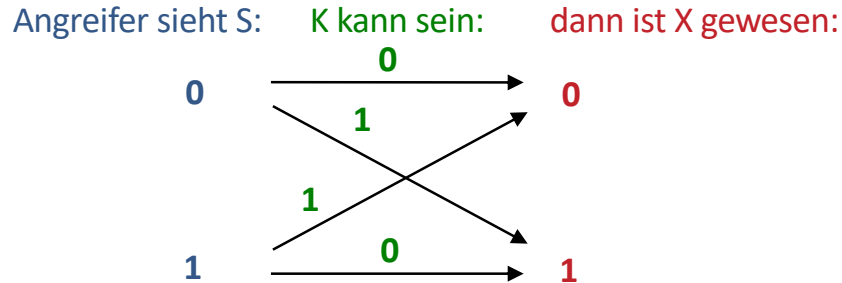
- Jedes Schlüsselbit darf nur einmal verwendet werden
- Bits von K sind zufällig und unabhängig
- Schlüssel genauso lang wie Klartext



$X \oplus K = S$		
0	0	0
0	1	1
1	0	1
1	1	0

One-Time-Pad (mod 2)

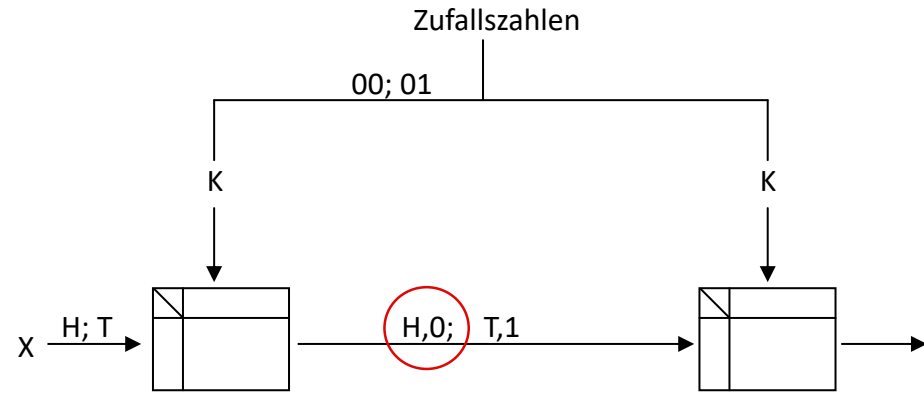
- Jedes Schlüsselbit darf nur einmal verwendet werden
- Bits von K sind zufällig und unabhängig
- Schlüssel genauso lang wie Klartext



X	\oplus K	= S
0	0	0
0	1	1
1	0	1
1	1	0

- Der Angreifer kann alle 4 Varianten durchrechnen, erhält dadurch aber keine zusätzliche Information über den Klartext.
- Die Wahrscheinlichkeit, ein Kartextbit richtig zu raten, verändert sich durch die Beobachtung des Schlüsseltextes nicht, sondern bleibt $\text{const} = 0,5$.

Informationstheoretisch sichere symmetrische Authentifikationscodes



H: Heads (Kopf)
T: Tails (Zahl)

x, MAC k \	H,0	H,1	T,0	T,1
00	H	–	T	–
01	H	–	–	T
10	–	H	T	–
11	–	H	–	T

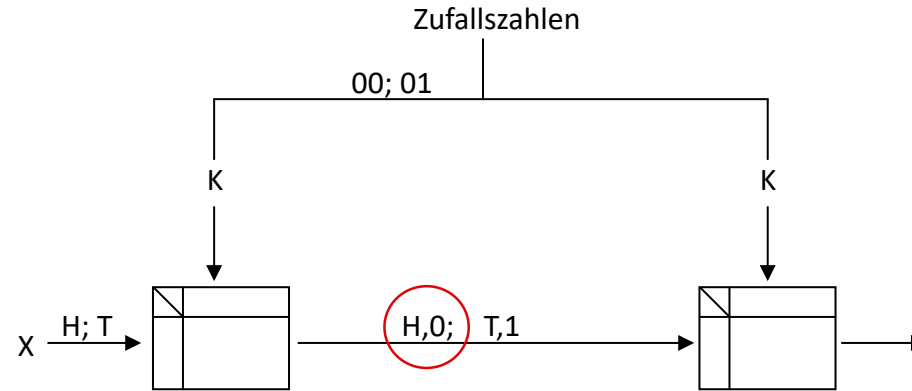
oder

$k \backslash x$	H	T
00	0	0
01	0	1
10	1	0
11	1	1

MAC

H := "0"
T := "1"

Informationstheoretisch sichere symmetrische Authentifikationscodes



H: Heads (Kopf)
T: Tails (Zahl)

- **Angriff 1: (blind)**
 - Angreifer will T senden
 - erwischt richtigen MAC mit Wkt = 0,5
- **Angriff 2: (sehend)**
 - Angreifer will H,0 in T ändern
 - weiß: $k \in \{00,01\}$
 - wenn $k = 00$ war, muss er T,0 senden
 - wenn $k = 01$ war, muss er T,1 senden
 - Wkt. ist immernoch 0,5

k \ x	x	
	H	T
00	0	0
01	0	1
10	1	0
11	1	1

MAC

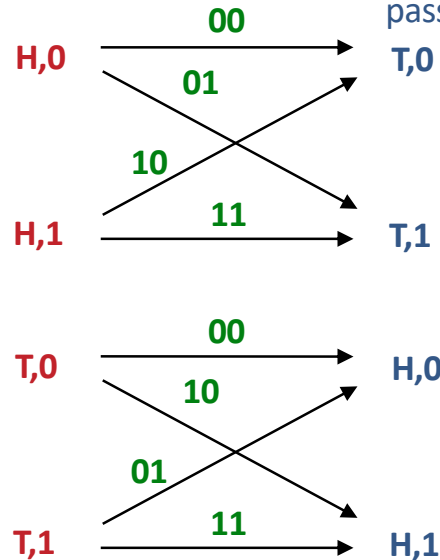
Informationstheoretisch sichere symmetrische Authentifikationscodes

- informationstheoretisch sicher

Angreifer sieht:

K kann sein:

Angreifer will x fälschen und sucht passenden MAC



Wkt., dass Angreifer den richtigen MAC für das Bit wählt, ist 0,5 (d.h. „Raten“)

H: Heads (Kopf)
T: Tails (Zahl)

k \ x	x	
	H	T
00	0	0
01	0	1
10	1	0
11	1	1

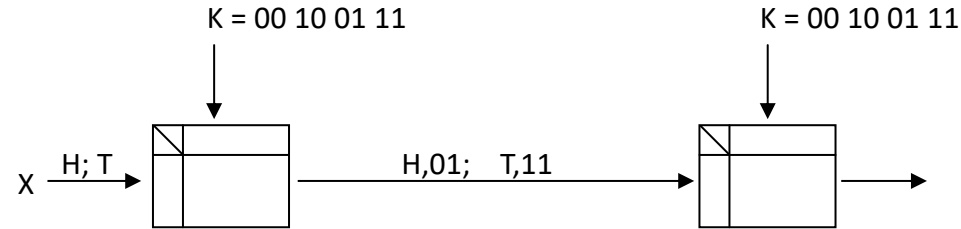
MAC

- Leseempfehlung: Baumann, Franz, Pfitzmann: Kryptographische Systeme. Springer Vieweg, 2014, S. 182

Informationstheoretisch sichere symmetrische Authentifikationscodes

■ Erweiterung auf **r-Bit MAC** zur Senkung der Ratewahrscheinlichkeit

- verwende pro Nachrichtenbit r Bit ($r > 1$) für den MAC
- Beispiel für $r=2$:



H: Heads (Kopf)
T: Tails (Zahl)

Anzahl der notwendigen Schlüsselbits pro Nachrichtenbit	r-Bit MAC	Ratewahrscheinlichkeit
2	1	$1/2$
4	2	$1/4$
6	3	$1/8$
8	4	$1/16$
$2 \cdot r$	r	2^{-r}

k \ x	x	
	H	T
00	0	0
01	0	1
10	1	0
11	1	1

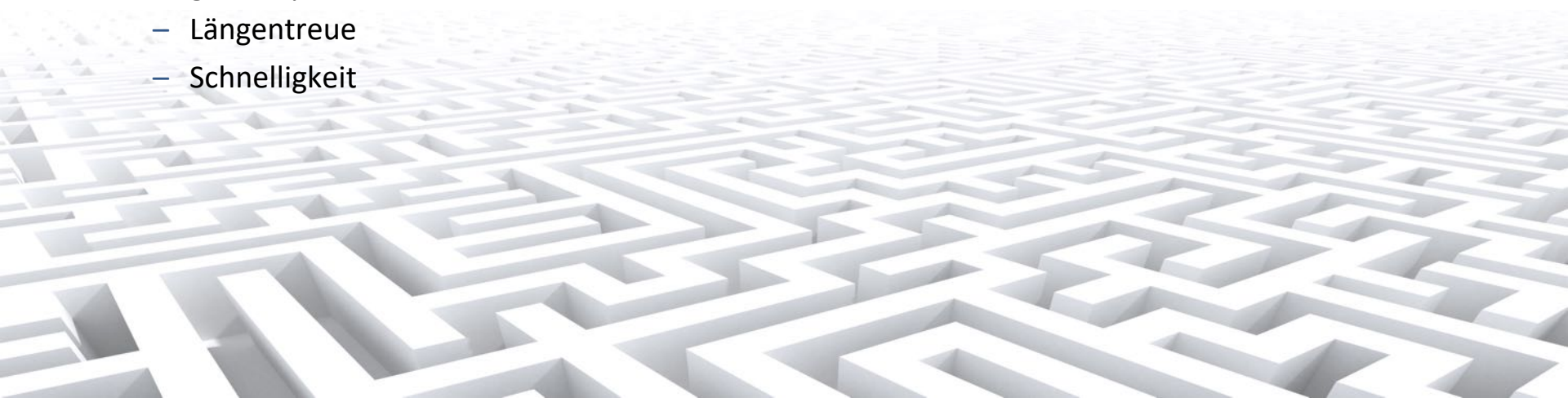
MAC

DES (Data Encryption Standard)

- Amerikanischer Verschlüsselungsstandard
 - 1977 vom National Bureau of Standards (NBS) der USA standardisiert
- Blockchiffre
 - operiert auf Blöcken von jeweils 64 Bit
- Feistel-Chiffre
 - iterierte Anwendung eines Verschlüsselungsschemas aus Permutationen, Substitutionen und Expansionen
- $n=16$ Runden
 - mit jeweils unterschiedlichen Teilschlüsseln K_i
- Schema ist selbstinvers
 - d.h. Entschlüsselung wie Verschlüsselung, jedoch umgekehrte Reihenfolge der Teilschlüssel

Gütekriterien für gute moderne symmetr. Chiffren

- **Höchstmaß an**
 - Vollständigkeit
 - Avalanche
 - Nichtlinearität
 - Korrelationsimmunität
- **weitere Kriterien**
 - gute Implementierbarkeit
 - Längentreue
 - Schnelligkeit



Gütekriterien

■ Vollständigkeit

- Def.: Eine Funktion $F:\{0,1\}^n \rightarrow \{0,1\}^m$ ist dann vollständig, wenn jedes Bit des Outputs von jedem Bit des Inputs abhängt.
- Beispiel:
 - $y_1 = x_1x_2 + x_1x_3 + x_2x_3 + x_2 + x_3 + 1$
 - $y_2 = x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_3 + 1$
 - $y_3 = x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_2 + 1$

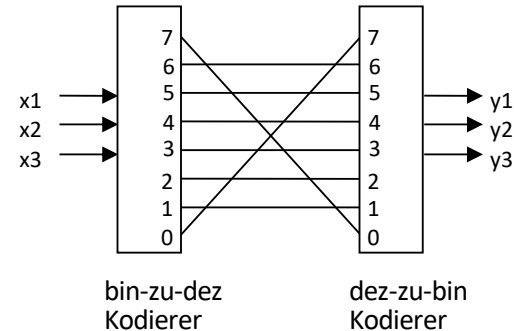
■ Avalanche

- Def.: Eine Funktion $F:\{0,1\}^n \rightarrow \{0,1\}^m$ besitzt dann den **Avalanche-Effekt**, wenn die Änderung eines Input-Bits im Mittel die Hälfte aller Output-Bits ändert.
- Wird durch Änderung eines Input-Bits jedes Output-Bit mit einer Wahrscheinlichkeit von 50% verändert, so erfüllt F das **strikte Avalanche-Kriterium**.
- Satz: Erfüllt F das strikte Avalanche-Kriterium, so ist F stets **vollständig**.

■ Linearität

- Def.: Eine Funktion $F:\{0,1\}^n \rightarrow \{0,1\}^m$ ist dann linear, wenn jedes Output-Bit y_j linear von den Input-Bits x_i abhängt.
- Wenn wenigstens ein Output-Bit linear von den Input-Bits abhängt, bezeichnet man F als **partiell linear**.
- Beispiel: (siehe Vollständigkeit)

X_{dez}	x_3	x_2	x_1	y_3	y_2	y_1	Y_{dez}
0	0	0	0	1	1	1	7
1	0	0	1	0	0	1	1
2	0	1	0	0	1	0	2
3	0	1	1	0	1	1	3
4	1	0	0	1	0	0	4
5	1	0	1	1	0	1	5
6	1	1	0	1	1	0	6
7	1	1	1	0	0	0	0

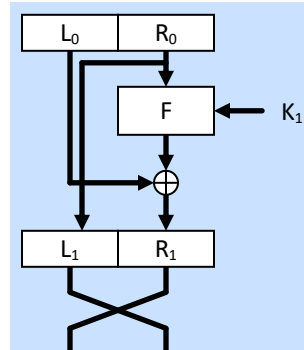


■ Korrelationsimmunität

- Sei $f(x_1, \dots, x_n)$ eine boolesche Funktion in n Variablen.
- f ist dann **k-korrelationsimmun**, wenn man aus Kenntnis einer beliebigen Menge von k Eingangswerten keine Informationen über den resultierenden Ausgangswert erhalten kann und umgekehrt.
- Bedeutung:
 - Jede Teilmenge der Output-Vektoren, die Rückschlüsse auf Teilmengen der Input-Vektoren zulässt, verringert den Aufwand für das vollständige Durchsuchen des Schlüsselraumes.

Feistel-Prinzip (1 Runde)

Verschlüsselung



$$L_1 = R_0 \quad (1)$$

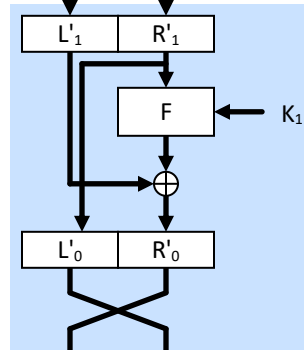
$$R_1 = f(R_0) \oplus L_0 \quad (2)$$

$$L'_1 = R_1 \quad (3)$$

$$R'_1 = L_1 \quad (4)$$

Schlüssel-
text

Entschlüsselung



$$L'_0 = R'_1 \quad (5)$$

$$R'_0 = f(R'_1) \oplus L'_1 \quad (6)$$

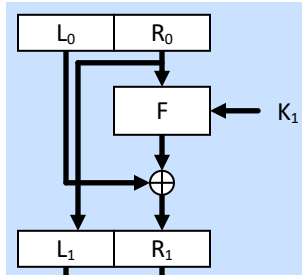
Funktion F kann
Einwegfunktion sein

Klartext

Feistel-Prinzip (n Runden)

Verschlüsselung

Runde 1



$$L_n = R_{n-1}$$

$$R_n = f(R_{n-1}) \oplus L_{n-1}$$

Runde 2..n

nach letzter Runde

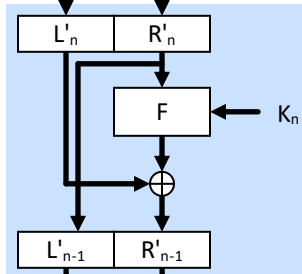
Schlüssel-
text

$$L'_n = R_n$$

$$R'_n = L_n$$

Entschlüsselung

Runde n



$$L'_{n-1} = R'_n$$

$$R'_{n-1} = f(R'_n) \oplus L'_n$$

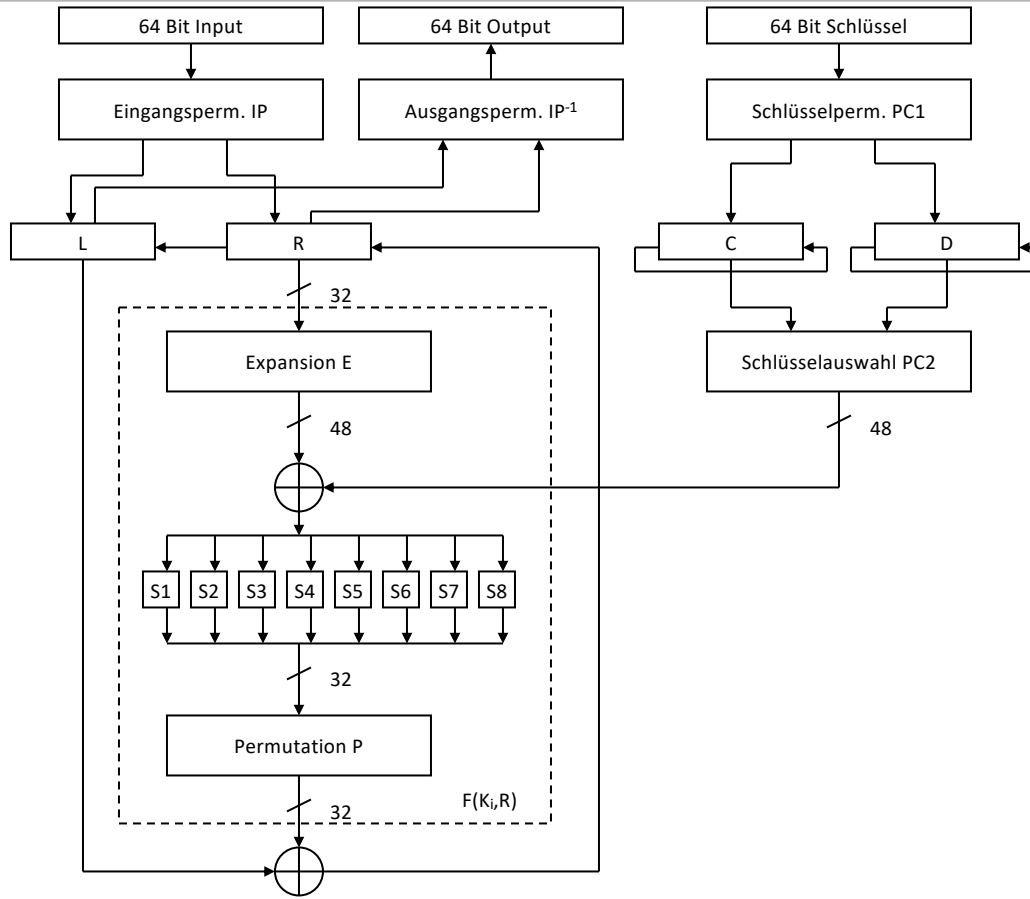
Runde n-1..1

L und R vertauschen

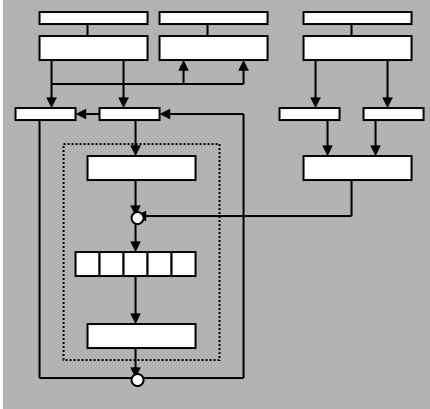
Klartext

Data Encryption Standard (DES)

Fumy, Rieß: Kryptographie, 1988



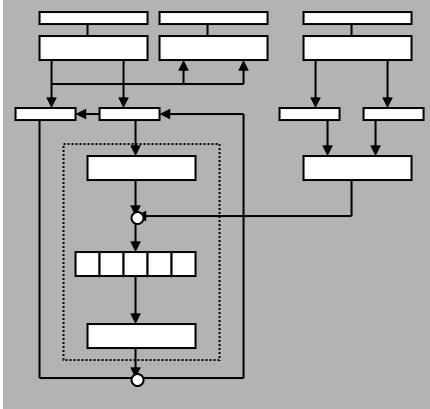
DES (Data Encryption Standard)



■ Symmetrische Blockchiffre

- $M \in \{0,1\}^{64}$, $K \in \{0,1\}^{56}$
- Feistel-Chiffre
- $n = 16$ Runden
- Schlüssel besteht aus 56 Bit + 8 Paritätsbits
- Teilschlüssel $K_1 \dots K_{16}$ (jeweils 48 Bit) werden aus einem 56-Bit Schlüssel gewonnen
- Vor der ersten und nach der letzten Runde durchläuft der Datenblock eine Permutation IP bzw. IP^{-1} , die kryptographisch irrelevant ist.

DES (Data Encryption Standard)



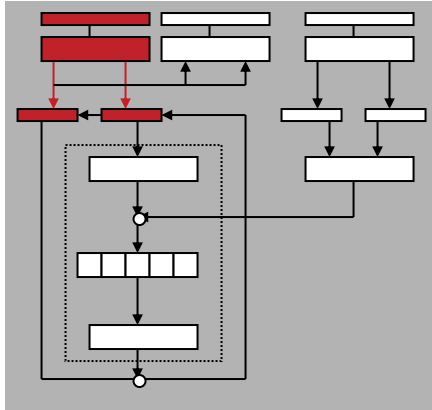
■ Funktion $F(K_i, R_{i-1})$

- Expansionsabbildung von 32 auf 48 Bit
- 8 S-Boxen, jede S-Box: 6-Bit-Input, 4-Bit-Output
- 32-Bit-Permutation

■ Teilschlüsselgenerierung

- Permuted Choice 1 (Schlüsselpermutation)
- Zyklische Schiebeoperationen auf Registern C und D in Abhängigkeit der Rundennummer
- Permuted Choice 2 (Schlüsselauswahl 48 aus 56 Bit)

DES: IP

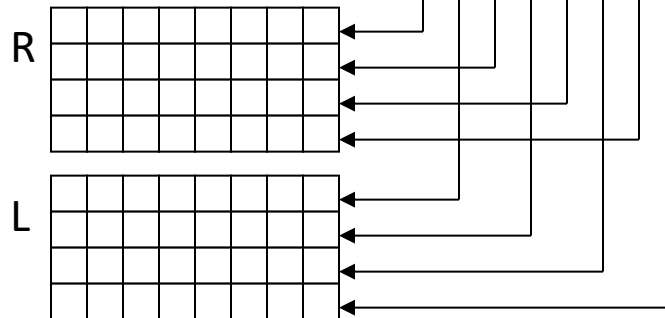


Inputpermutation IP

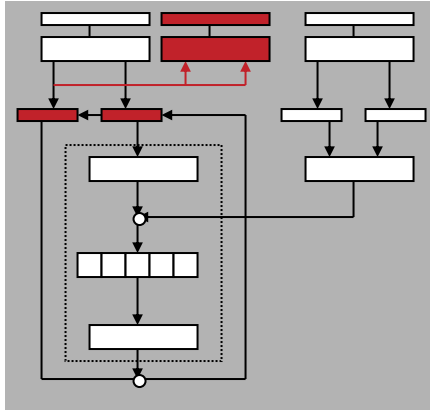
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

1	2	3	4	5	6	7	8
9	10	11	12	...			16
17							24
25							32
33							40
41							48
49							56
57	58	59	60	61	62	63	64

Dateninput/-output



DES: IP-1

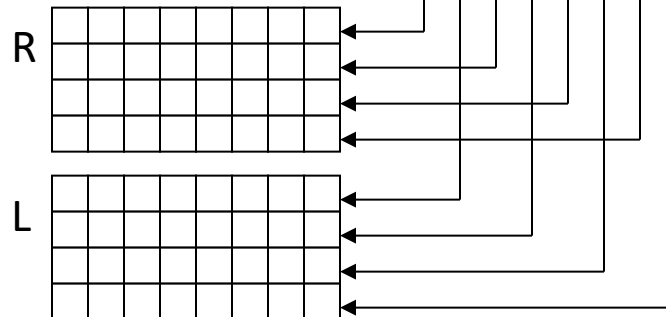


Outputpermutation IP-1

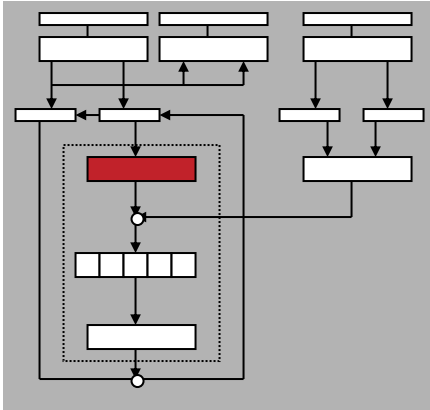
40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

1	2	3	4	5	6	7	8
9	10	11	12	...			16
17							24
25							32
33							40
41							48
49							56
57	58	59	60	61	62	63	64

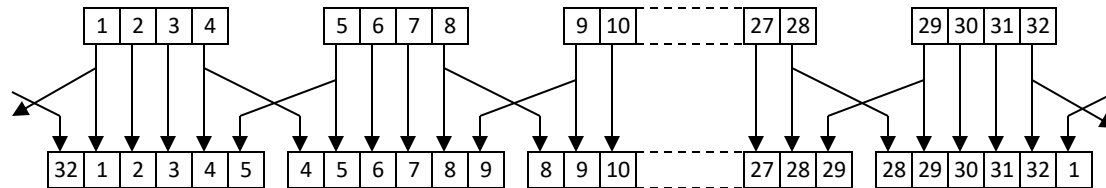
Dateninput/-output



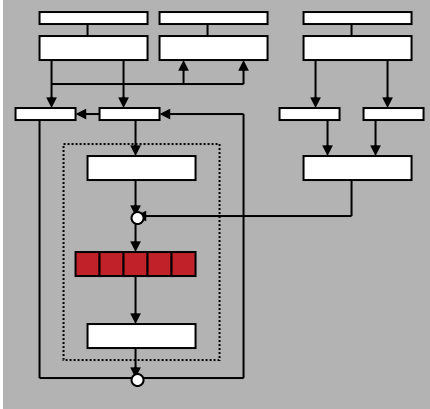
DES: Expansion E



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

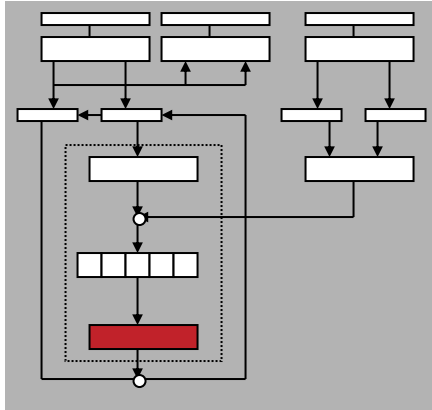


DES: S-Boxen S1 bis S8

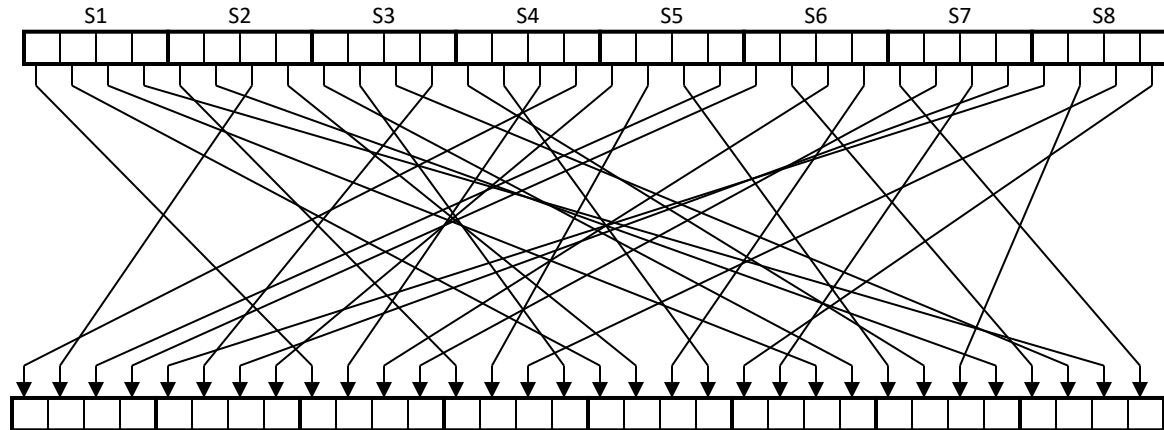


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
S1:	0:	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1:	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2:	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3:	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2:	0:	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1:	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2: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

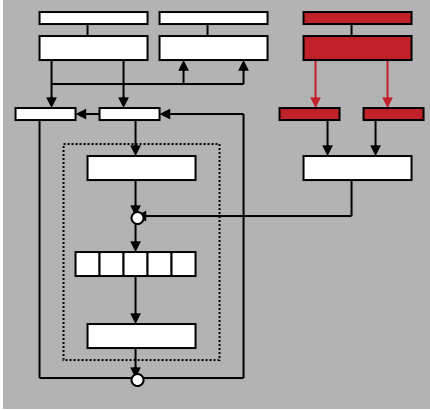
DES: Permutation P



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



DES: PC1



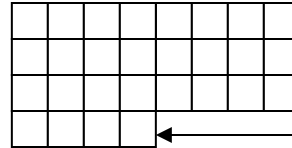
Schlüsselpermutation PC1							
57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36				
63	55	47	39	31	23	15	7
62	54	46	38	30	22	14	6
61	53	45	37	29	21	13	5
				28	20	12	4

Externer Schlüssel

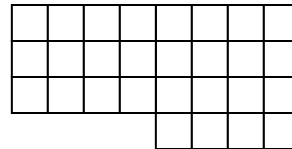
MSB				LSB			
1	2	3	4	5	6	7	8
9	10	11	12	...			16
17							24
25							32
33							40
41							48
49							56
57	58	59	60	61	62	63	64

Paritätsbits

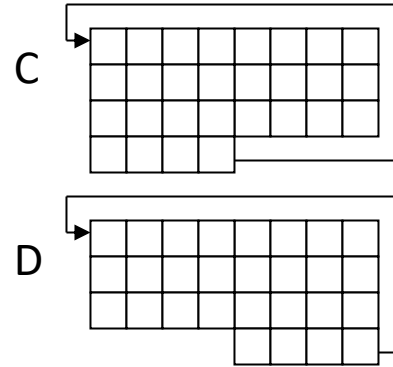
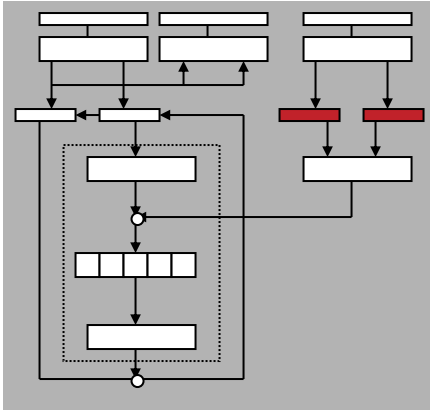
C



D



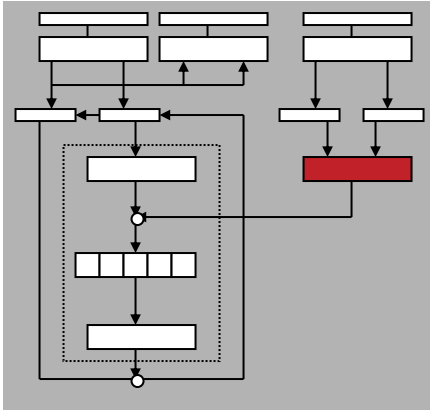
DES: Shifts bei Chiffrierung und Dechiffrierung



Anzahl der Shifts bei der Chiffrierung bzw. Deciffrierung

Rundennummer:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Links-Shifts:	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1 (Ver)
Rechts-Shifts:	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1 (Ent)

DES: PC2



Schlüsselauswahl (Permuted Choice 2, PC2)

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

Eigenschaften des DES

- Der DES ist vollständig: Jedes Output-Bit hängt von jedem Input-Bit ab.
- Der DES ist derart komplex, dass keinerlei analytische Abhängigkeit zwischen Input und Output oder Schlüssel und Output feststellbar ist.
- Der DES ist invariant gegenüber Komplementbildung, d.h.

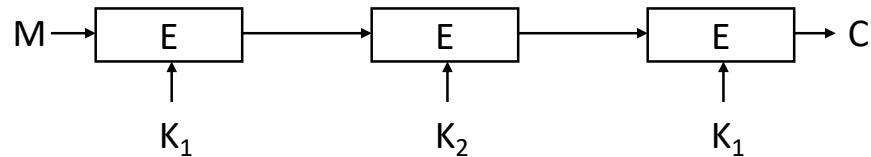
$$\overline{\text{DES}(K, M)} = \text{DES}(\overline{K}, \overline{M})$$

- Vier der 2^{56} Schlüssel sind schwach, d.h. $\text{DES}(K, \text{DES}(K, M)) = M$.

Externer Schlüssel	C-Register	D-Register
01 01 01 01 01 01 01 01	0000000	0000000
1F 1F 1F 1F 0E 0E 0E 0E	0000000	FFFFFFF
E0 E0 E0 E0 F1 F1 F1 F1	FFFFFFF	0000000
FE FE FE FE FE FE FE FE	FFFFFFF	FFFFFFF

Kritikpunkte

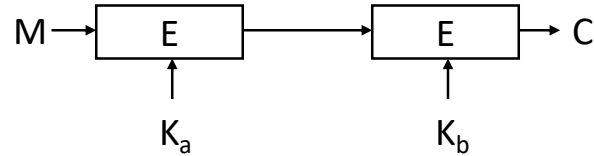
- Designkriterien wurden nicht offengelegt (inzwischen bekannt)
- nur ineffizient in Software implementierbar (wg. Permutationen)
- wirksame Schlüssellänge heute viel zu gering (56 Bit)
 - Ausweg: 3-DES (Triple-DES)
 - Verbesserung der Sicherheit durch 3-fache Anwendung



$$C = \text{DES}(K_1, \text{DES}(K_2, \text{DES}(K_1, M)))$$

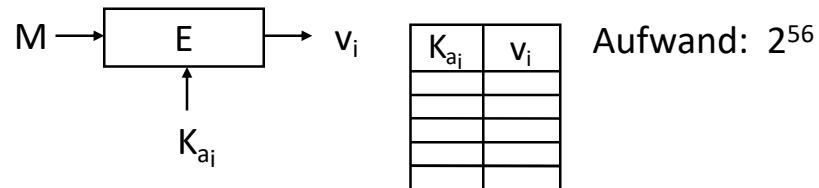
Möglicher Angriff bei 2-DES

■ Ausgangssituation



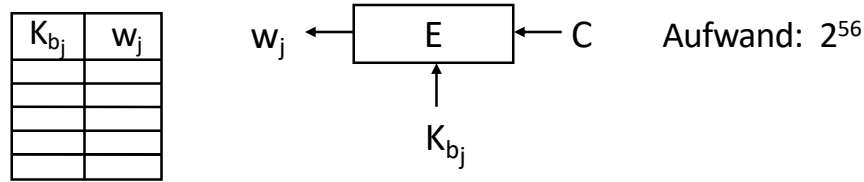
■ Known-plaintext-attack

1. Verschlüsse M für alle möglichen K_a und speichere die Schlüsseltexte v_i in einer Tabelle: $v_i = E(M, K_{a_i})$



Möglicher Angriff bei 2-DES

2. Entschlüssele C für alle möglichen K_b und speichere die Klartexte w_j ebenfalls in einer Tabelle: $w_j = D(C, K_{b_j})$



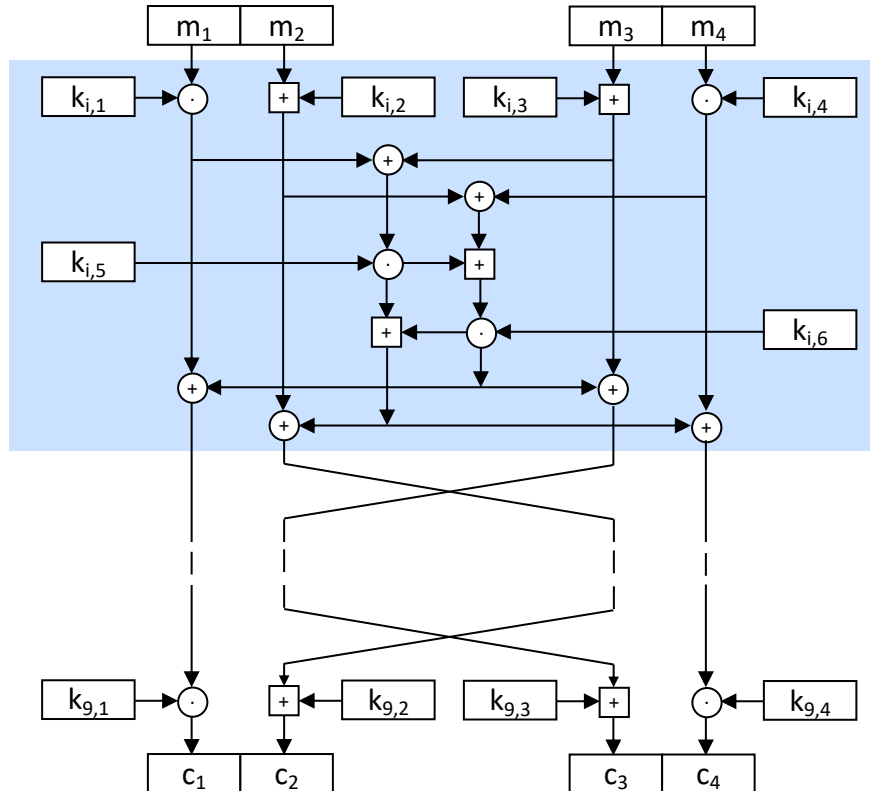
3. Falls $v_i == w_j$ für ein bestimmtes Paar i und j , sind K_{a_i} und K_{b_j} die gesuchten Schlüssel, ggf. Probe mit weiteren M/C-Paaren machen!

■ Aufwand

- $2^{56} + 2^{56} = 2 \cdot 2^{56} = 2^{57}$
- Sicherheitsgewinn wäre nur 1 Bit

- Symmetrische Blockchiffre mit $M \in \{0,1\}^{64}$, $K \in \{0,1\}^{128}$
- Operationen:
 - ⊕ bitweise Addition mod 2
 - ⊞ Addition mod 2^{16}
 - ⊙ Multiplikation mod $2^{16} + 1$ (0 durch 2^{16} dargestellt)
- Ablauf
 - M wird in vier 16-Bit-Operanden $m_1 \dots m_4$ zerlegt.
 - Es werden $i=1\dots 8$ Runden durchlaufen.
 - Aus K werden sechs 16-Bit-Operanden $k_{i,1} \dots k_{i,6}$ erzeugt.
- Teilschlüsselgenerierung
 - $K \rightarrow k_{1,1} \dots k_{1,6}, k_{2,1}, k_{2,2}$ (K wird in 8 Teile zerlegt.)
 - $\text{shiftLeft}(K, 25) \rightarrow k_{2,3} \dots k_{2,6}, k_{3,1} \dots k_{3,4}$
 - $\text{shiftLeft}(K, 25) \rightarrow k_{3,5}, k_{3,6}, k_{4,1} \dots k_{4,6}$
 - u.s.w
 - Nach jeder Erzeugung zyklische Linksverschiebung von K um 25 Bitstellen.

International Data Encryption Algorithm (IDEA)



1. Runde,
selbstinvers,
insg. 8 Runden

- \oplus bitweise Addition mod 2
- \boxplus Addition mod 2^{16}
- \odot Multiplikation mod $2^{16} + 1$
(0 wird durch 2^{16} dargestellt)

International Data Encryption Algorithm (IDEA)

■ Entschlüsselung

- k_j sei Teilschlüssel zum Verschlüsseln in Runde j
- d_j sei Teilschlüssel zum Entschlüsseln in Runde j
- r_{\max} sei Rundenzahl (hier $r_{\max} = 8$)
- $z = r_{\max} + 2$

$$d_{j,1} = (k_{z-j,1})^{-1} \bmod 2^{16}+1 \quad \text{mit } 1 \leq j \leq r_{\max} + 1$$

$$d_{j,4} = (k_{z-j,4})^{-1} \bmod 2^{16}+1 \quad \text{mit } 1 \leq j \leq r_{\max} + 1$$

$$d_{j,2} = (k_{z-j,2})^{-1} \bmod 2^{16} \quad \text{mit } j = 1, j = r_{\max} + 1$$

$$d_{j,2} = (k_{z-j,3})^{-1} \bmod 2^{16} \quad \text{mit } 1 < j < r_{\max} + 1$$

$$d_{j,3} = (k_{z-j,3})^{-1} \bmod 2^{16} \quad \text{mit } j = 1, j = r_{\max} + 1$$

$$d_{j,3} = (k_{z-j,2})^{-1} \bmod 2^{16} \quad \text{mit } 1 < j < r_{\max} + 1$$

$$d_{j,5} = (k_{z-(j+1),5}) \quad \text{mit } 1 \leq j \leq r_{\max} + 1$$

$$d_{j,6} = (k_{z-(j+1),6}) \quad \text{mit } 1 \leq j \leq r_{\max} + 1$$

International Data Encryption Algorithm (IDEA)

■ Designkriterien/Eigenschaften

- Mischen verschiedenartiger Grundoperationen soll hohe Komplexität bereits nach wenigen Runden erreichen
- Grundoperationen bewusst »inkompatibel« gewählt (erfüllen z.B. in keiner Kombination ein Distributiv- oder Assoziativgesetz)
- hoher Grad an Immunität gegenüber differentieller Kryptanalyse (nach vier Runden immun)
- bereits nach 1 Runde bzgl. der Inputbits vollständig, nach 2 Runden vollständig bzgl. der Schlüsselbits

■ Praktischer Einsatz

- sehr gut in Hard- und Software implementierbar
- sehr effizient
- Für kommerzielle Anwendungen fallen Lizenzgebühren an.

Advanced Encryption Algorithm (AES)

- **Nachfolger des DES**
 - Januar 1997 vom National Institute of Standards and Technology (NIST) als Nachfolger für DES initiiert
 - öffentliche internationale Ausschreibung

- **Neue Blockchiffre sollte folgende Kriterien erfüllen:**
 - symmetrische Blockchiffre mit einer Blockgröße von 128 Bit und variabler Schlüssellänge von 128, 192 und 256 Bit.
 - AES soll für mindestens 30 Jahre Sicherheit bieten.
 - Weder Algorithmus noch Implementierung dürfen patentiert sein.

- **August 1998 wurden 15 Kandidaten der Öffentlichkeit zur Begutachtung vorgelegt.**

Advanced Encryption Algorithm (AES)

- August 1999 wurden die 5 Finalisten vorgestellt:
 - MARS – IBM
 - RC6 – RSA Labs
 - Rijndael – Joan Daemen (Proton World Intl.), Vincent Rijmen (Katholieke Universiteit Leuven, Belgien)
 - Serpent – Ross Anderson (Univ of Cambridge), Eli Biham (Technion), Lars Knudsen (UC San Diego)
 - Twofish – Bruce Schneider, John Kelsey, Niels Ferguson (Counterpane Internet Security), Doug Whiting (Hi/fn, Inc.), David Wagner (UC Berkeley), Chris Hall (Princeton Univ.)
- Oktober 2000:
 - Rijndael wird ausgewählt.
- Begründung für Rijndael
 - Beste Kombination von Sicherheit, Leistungsfähigkeit, Effizienz und Implementierbarkeit sowohl in Software als auch in Hardware.

Rijndael (AES)

■ Rijndael (sprich: Rein-dahl)

- Blockchiffre
- keine Feistel-Chiffre, arbeitet aber in Runden
- Rundentransformation besteht aus drei invertierbaren Transformationen
- variable Blocklänge und variable Schlüssellänge, jeweils unabhängig wählbar aus {128 Bit, 192 Bit, 256 Bit}.
- Blockbreite {Nachrichtenblock, Schlüssel} in Bit
= {**Nb**, **Nk**} · 8 Bit · 4 rows
- Beispiel: Nb = 6 und Nk = 4

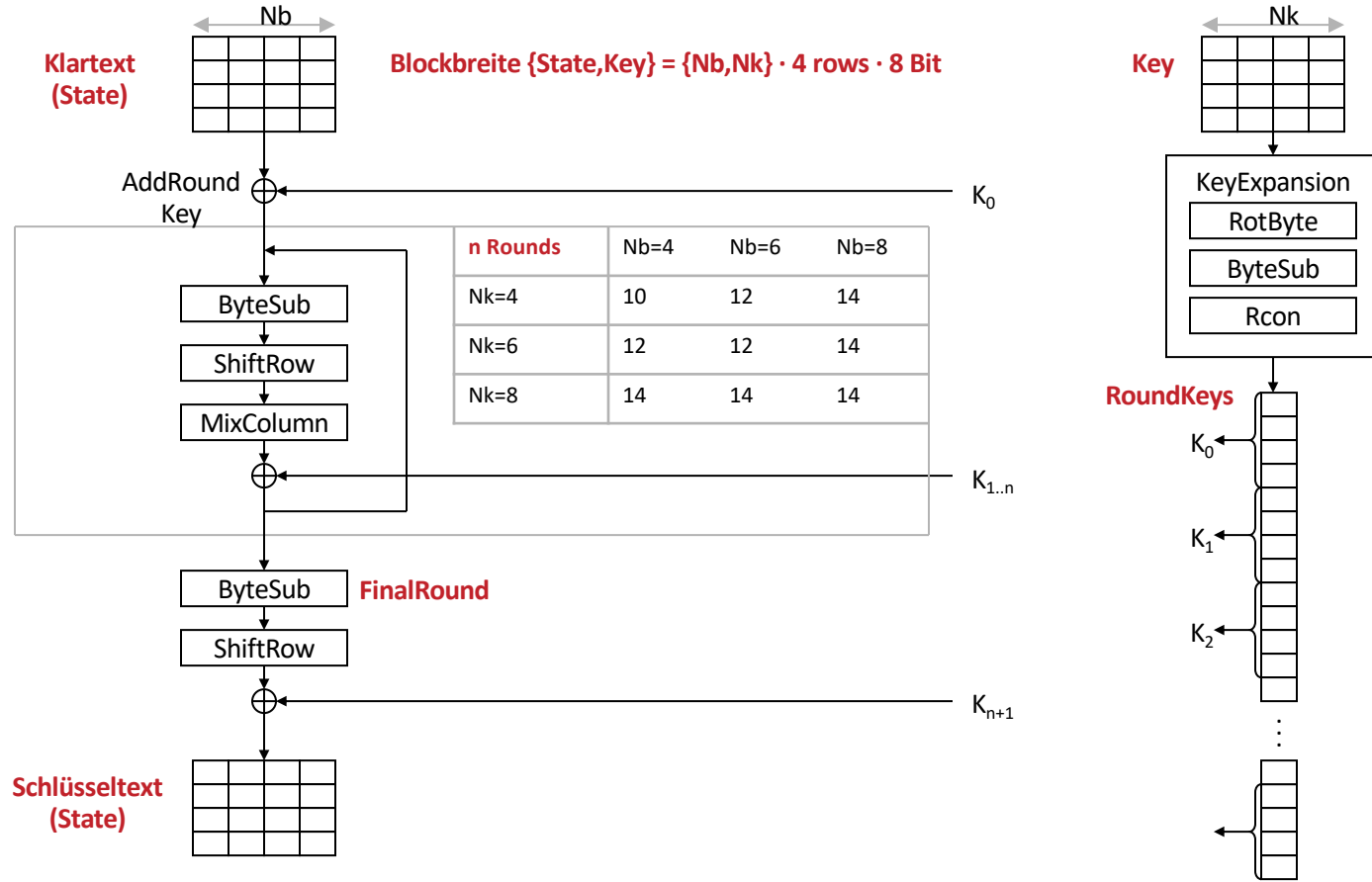
State

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

Cipher Key

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Rijndael (AES)



Rijndael (AES)

- Rundenzahl Nr ist eine Funktion von Nb und NK

Nr	Nb=4	Nb=6	Nb=8
Nk=4	10	12	14
Nk=6	12	12	14
Nk=8	14	14	14

```
Rijndael(State,CipherKey) {  
    KeyExpansion(CipherKey,ExpandedKey);  
    AddRoundKey(State,ExpandedKey);  
    For(i=1;i<Nr;i++)  
        Round(State,ExpandedKey+Nb*i); // Pointer !  
    FinalRound(State,ExpandedKey+Nb*Nr); // Pointer !  
}
```

Rijndael (AES)

■ Rundentransformationen

```
Round(State, RoundKey) {  
    ByteSub(State);  
    ShiftRow(State);  
    MixColumn(State);  
    AddRoundKey(State, RoundKey);  
}  
  
FinalRound(State, RoundKey) {  
    // wie Round, aber ohne MixColumn  
    ByteSub(State);  
    ShiftRow(State);  
    AddRoundKey(State, RoundKey);  
}
```

Rijndael (AES)

■ ByteSub

- operiert auf jedem Byte von State unabhängig
- ist eine S-Box-Transformation

1. berechne das Multiplikative Inverse in $GF(2^8)$

2. berechne:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

- Umkehroperation: Inverse Tabelle und anschließend Berechnung des Multiplikativen Inversen in $GF(2^8)$
- ByteSub kann als Tabelle vorberechnet werden.

Rijndael (AES)

■ ByteSub

- ByteSub kann als Tabelle vorberechnet werden:

Input unteres Halbbyte →

↓
Input
oberes
Halbbyte

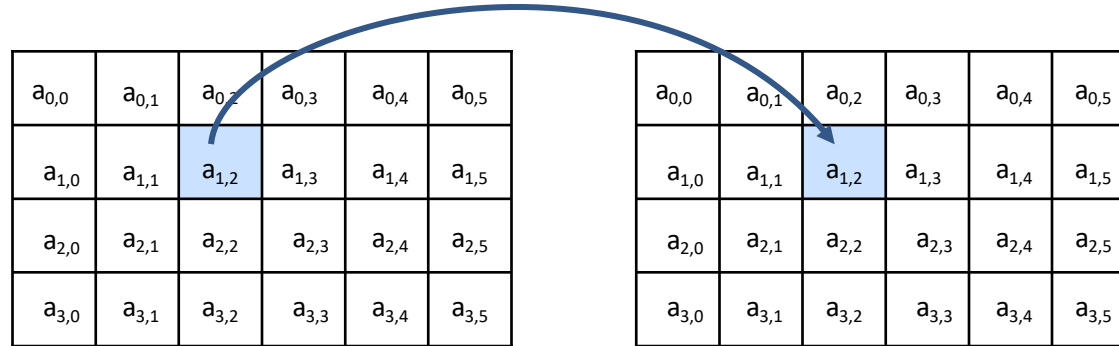
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	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	0B	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	1D	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

Darstellung in
Hexadezimalzahlen

Rijndael (AES)

■ ByteSub

- substituiert die Bytes von State unabhängig voneinander



Rijndael (AES)

■ ShiftRow

- Anzahl der zyklischen Linksshifts in Abhängigkeit von Nb

	row 0	row 1	row 2	row 3
Nb=4	0	1	2	3
Nb=6	0	1	2	3
Nb=8	0	1	3	4

- Beispiel: Nb=6

row 0: no shift	a _{0,0}	a _{0,1}	a _{0,2}	a _{0,3}	a _{0,4}	a _{0,5}	a _{0,0}	a _{0,1}	a _{0,2}	a _{0,3}	a _{0,4}	a _{0,5}
row 1: 1 shift	a _{1,0}	a _{1,1}	a _{1,2}	a _{1,3}	a _{1,4}	a _{1,5}	a _{1,1}	a _{1,2}	a _{1,3}	a _{1,4}	a _{1,5}	a _{1,0}
row 2: 2 shift	a _{2,0}	a _{2,1}	a _{2,2}	a _{2,3}	a _{2,4}	a _{2,5}	a _{2,2}	a _{2,3}	a _{2,4}	a _{2,5}	a _{2,0}	a _{2,1}
row 3: 3 shift	a _{3,0}	a _{3,1}	a _{3,2}	a _{3,3}	a _{3,4}	a _{3,5}	a _{3,3}	a _{3,4}	a _{3,5}	a _{3,0}	a _{3,1}	a _{3,2}
	vorher						nachher					

Rijndael (AES)

■ MixColumn

- operiert auf allen Spalten von State
- Berechne in $GF(2^8)$:

$$b(x) = a(x) \otimes c(x) \bmod x^4 + 1$$

mit $c(x) = '03' x^3 + '01' x^2 + '01' x + '02'$

- d.h.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	$b_{0,4}$	$b_{0,5}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$b_{1,5}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$	$b_{2,4}$	$b_{2,5}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$	$b_{3,4}$	$b_{3,5}$

Rijndael (AES)

■ MixColumn

– Inverse Operation:

$$a(x) = b(x) \otimes d(x) \bmod x^4+1$$

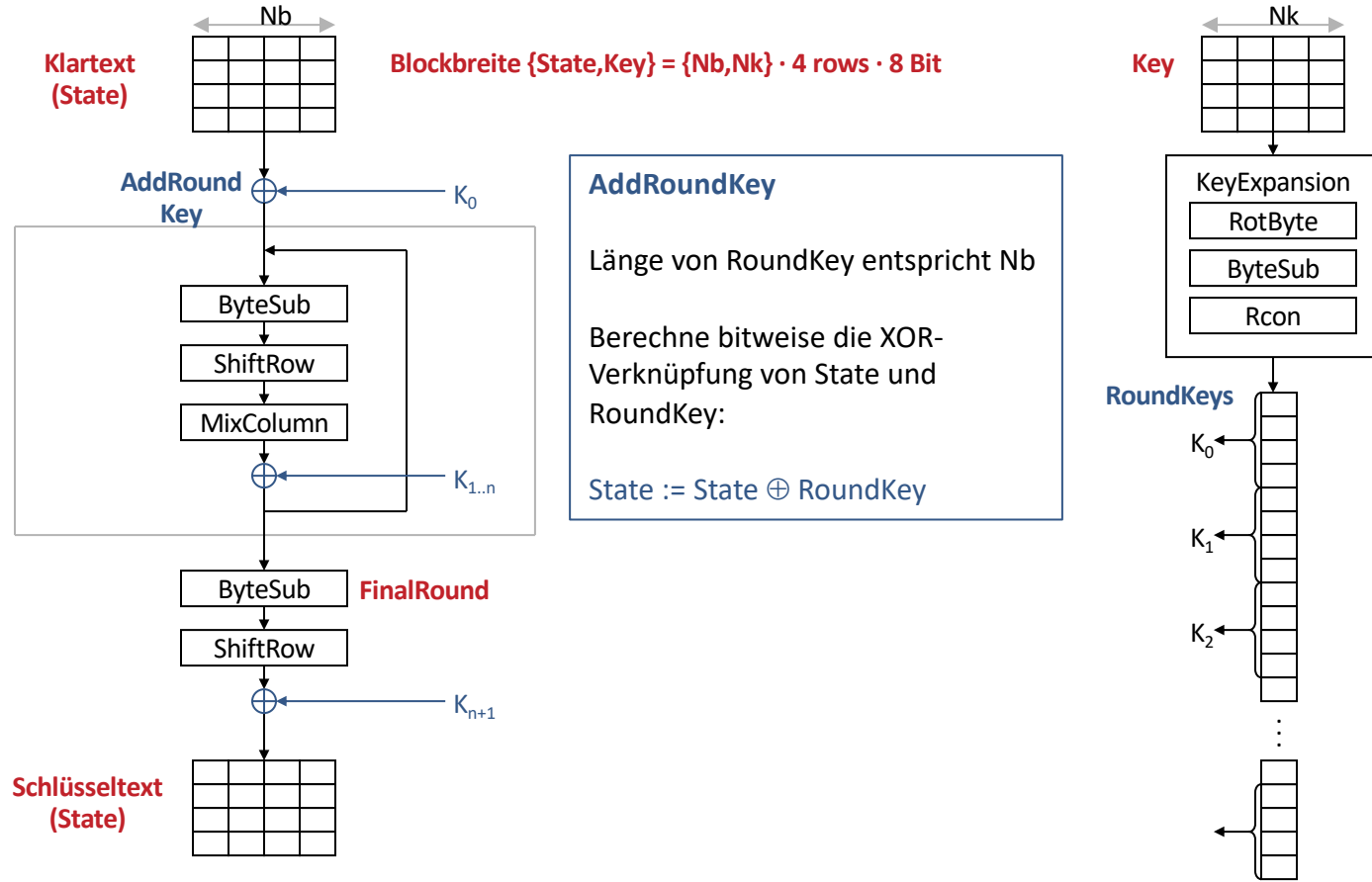
mit $d(x) = '0B' x^3 + '0D' x^2 + '09' x + '0E'$,

da $('03' x^3 + '01' x^2 + '01' x + '02') \otimes d(x) = '01'$
(neutrales Element bzgl. Multiplikation)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$	$b_{0,4}$	$b_{0,5}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,4}$	$b_{1,5}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$	$b_{2,4}$	$b_{2,5}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$	$b_{3,4}$	$b_{3,5}$

Rijndael (AES)



Rijndael (AES)

■ KeyExpansion

– für $Nk \leq 6$:

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)]){  
    for(i = 0; i < Nk; i++)  
        W[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);  
    for(i = Nk; i < Nb * (Nr + 1); i++) {  
        temp = W[i - 1];  
        if (i % Nk == 0)  
            temp = ByteSub(RotByte(temp)) ^ Rcon[i / Nk];  
        W[i] = W[i - Nk] ^ temp;  
    }  
}
```

– für $Nk > 6$:

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)]) {  
    for(i = 0; i < Nk; i++)  
        W[i] = (key[4*i],key[4*i+1],key[4*i+2],key[4*i+3]);  
    for(i = Nk; i < Nb * (Nr + 1); i++) {  
        temp = W[i - 1];  
        if (i % Nk == 0)  
            temp = ByteSub(RotByte(temp)) ^ Rcon[i / Nk];  
        else if (i % Nk == 4)  
            temp = ByteSub(temp);  
        W[i] = W[i - Nk] ^ temp;  
    }  
}
```

Rijndael (AES)

- **RotByte**: zyklische Schiebeoperation (byteweise links)
- **ByteSub** (wie bei Rundentransformation)
- **Rcon**[i] = (RC[i], 0x00, 0x00, 0x00) mit

$$\text{RC}[1] = 1$$

$$\text{RC}[i] = 2 \cdot \text{RC}[i-1] \quad \text{für } i > 1 \text{ und } \text{RC}[i-1] < 0x80$$

$$\text{RC}[i] = 2 \cdot \text{RC}[i-1] \text{ XOR } 0x11 \quad \text{für } i > 1 \text{ und } \text{RC}[i-1] \geq 0x80$$

i	1	2	3	4	5	6	7	8	9	10	(dez)
RC[i]	01	02	04	08	10	20	40	80	1B	36	(hex)

■ RoundKey Selection

- fortlaufende Auswahl
- Beispiel für $N_b = 6$ und $N_k = 4$:

W0 W1 W2 W3 W4 W5 W6 W7 W8 W9 W10 W11 W12 W13 W14 ...

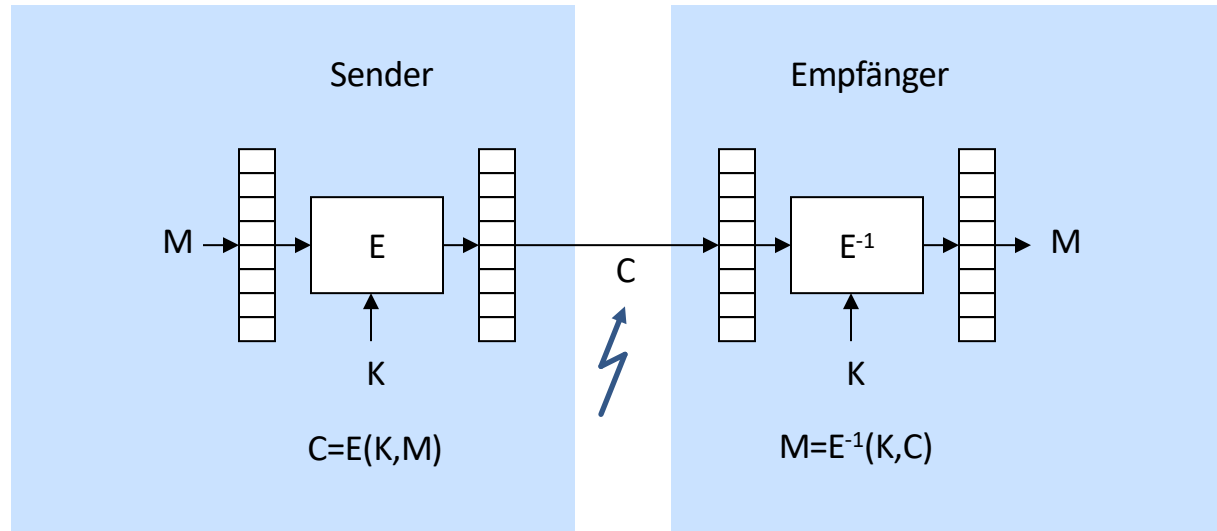
Round Key 0

Round Key 1

...

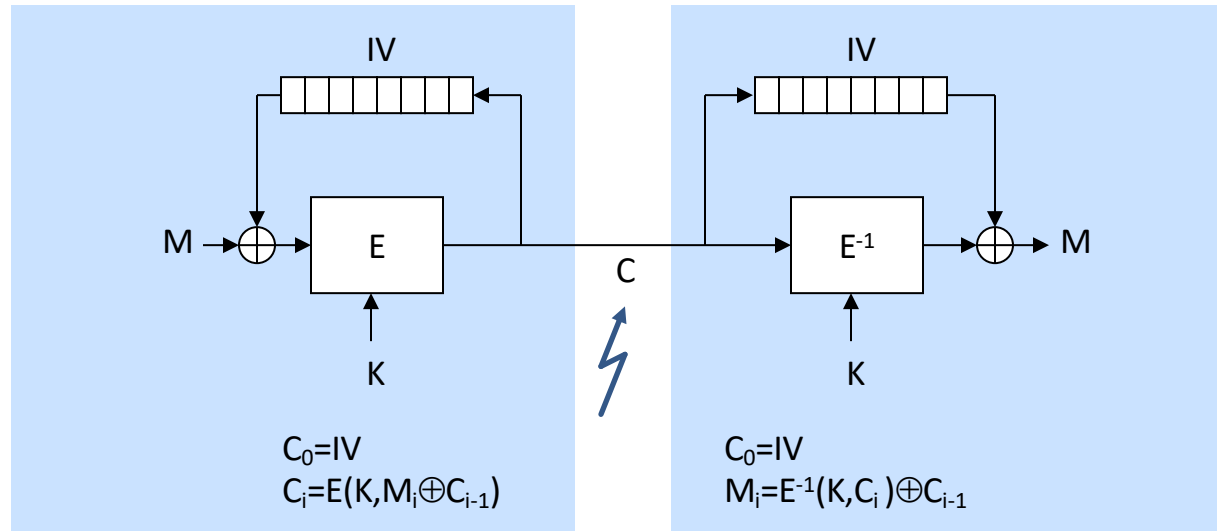
Betriebsarten von Blockchiffren

- Electronic Code Book (ECB)

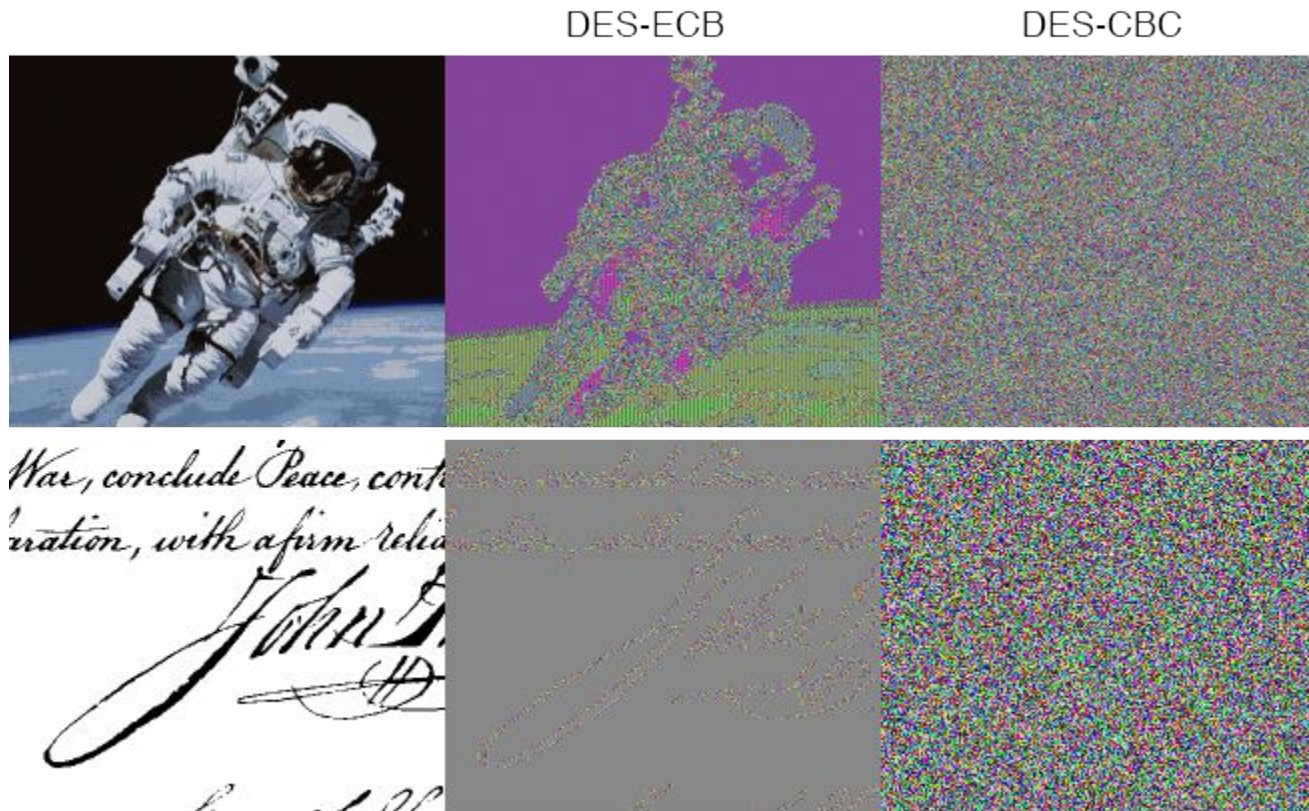


Betriebsarten von Blockchiffren

- Cipher Block Chaining (CBC)



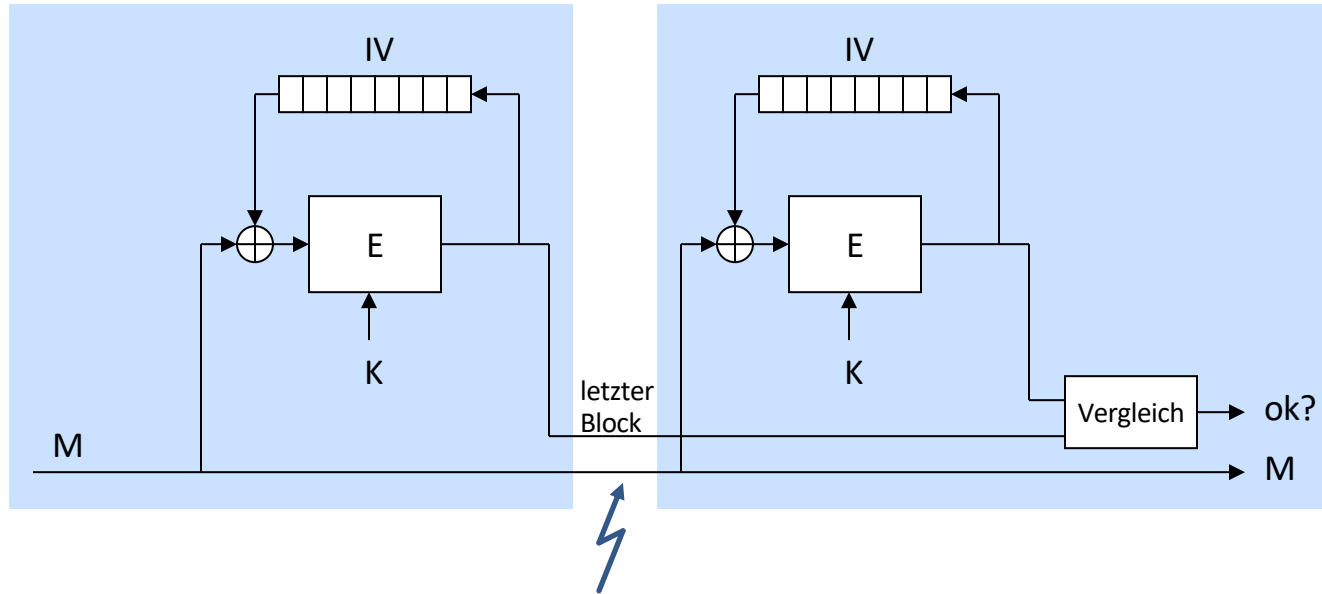
ECB und CBC im visuellen Vergleich



<http://gustlik.wordpress.com/2008/10/15/importance-of-block-cipher-modes/>

Betriebsarten von Blockchiffren

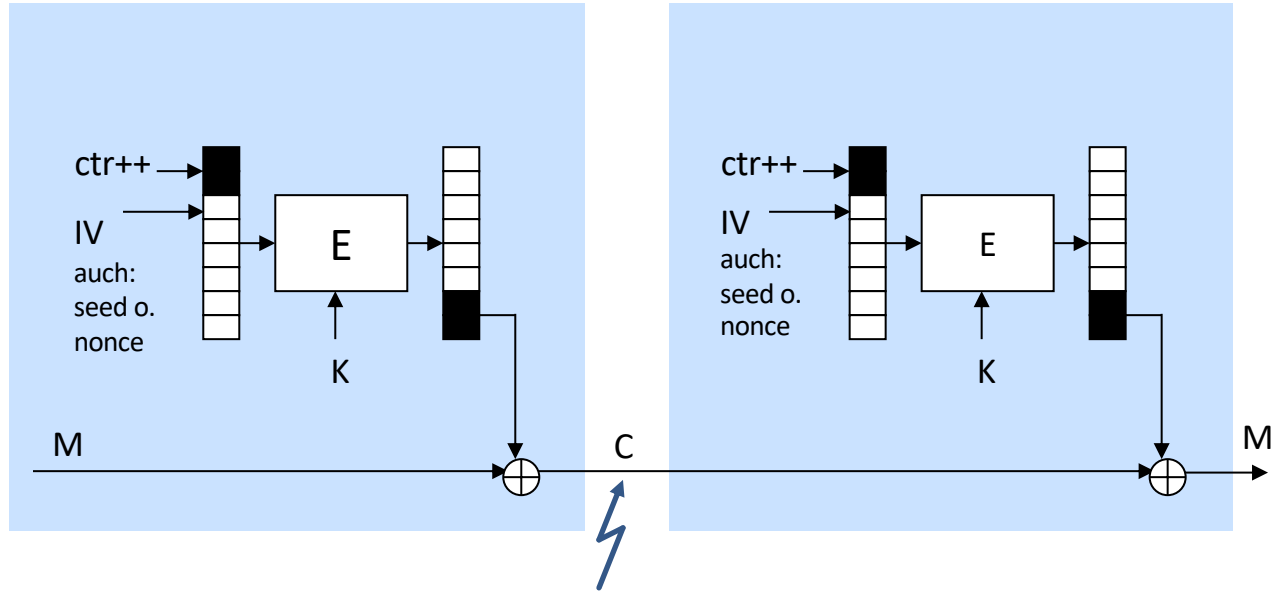
- CBC zur Authentifikation (auch: CBCAuth, CBC-MAC)



- Nur anwenden auf Nachrichten fester Länge, da ansonsten Length Extension Attack möglich

Betriebsarten von Blockchiffren

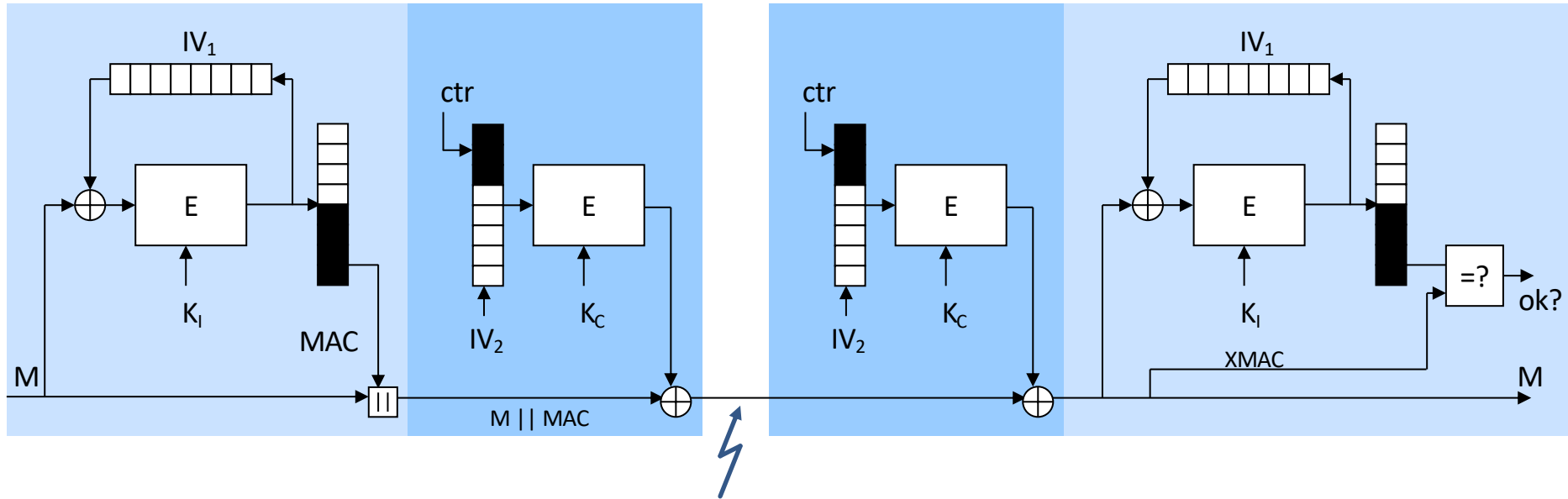
■ Counter Mode



■ IV stets zufällig wählen!

Betriebsarten von Blockchiffren

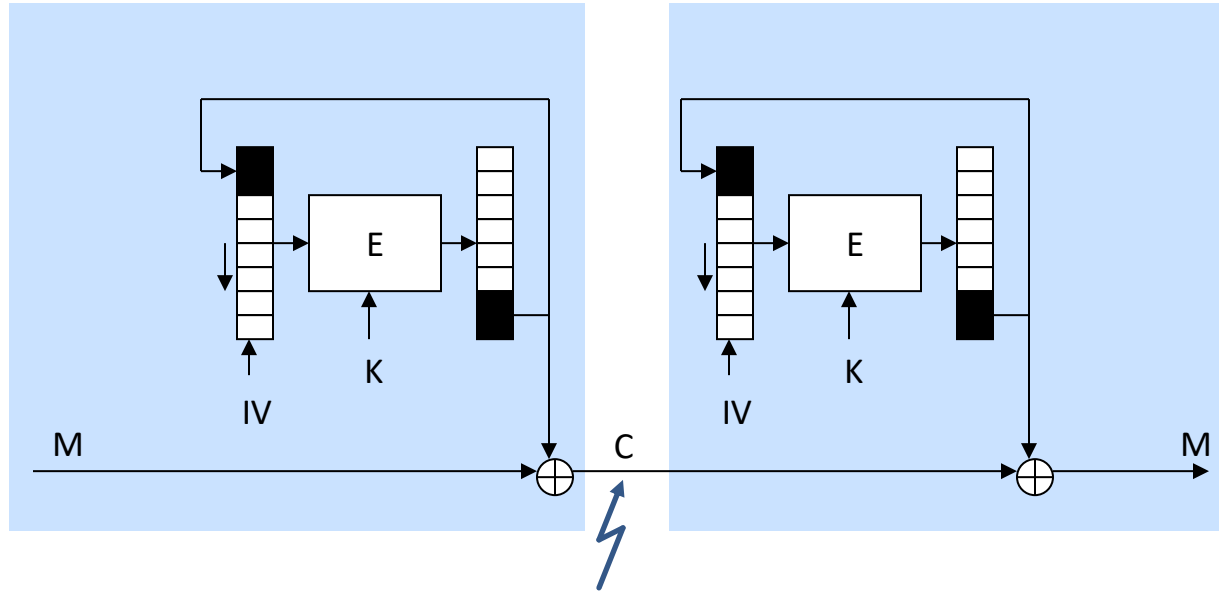
■ CCM (Counter with CBC-MAC)



- CCM als Kombination von CBC-MAC und Counter Mode
- angewendet bei WPA2 mit AES als Blockchiffre und $|MAC|=64\text{Bit}$

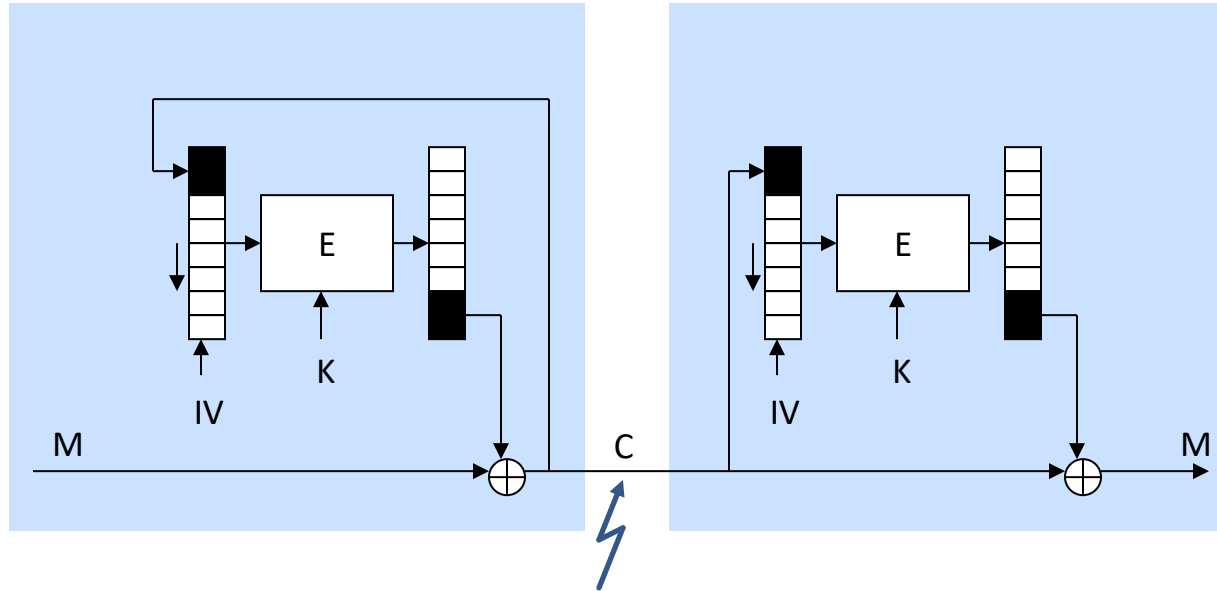
Betriebsarten von Blockchiffren

- Output Feedback (OFB)



Betriebsarten von Blockchiffren

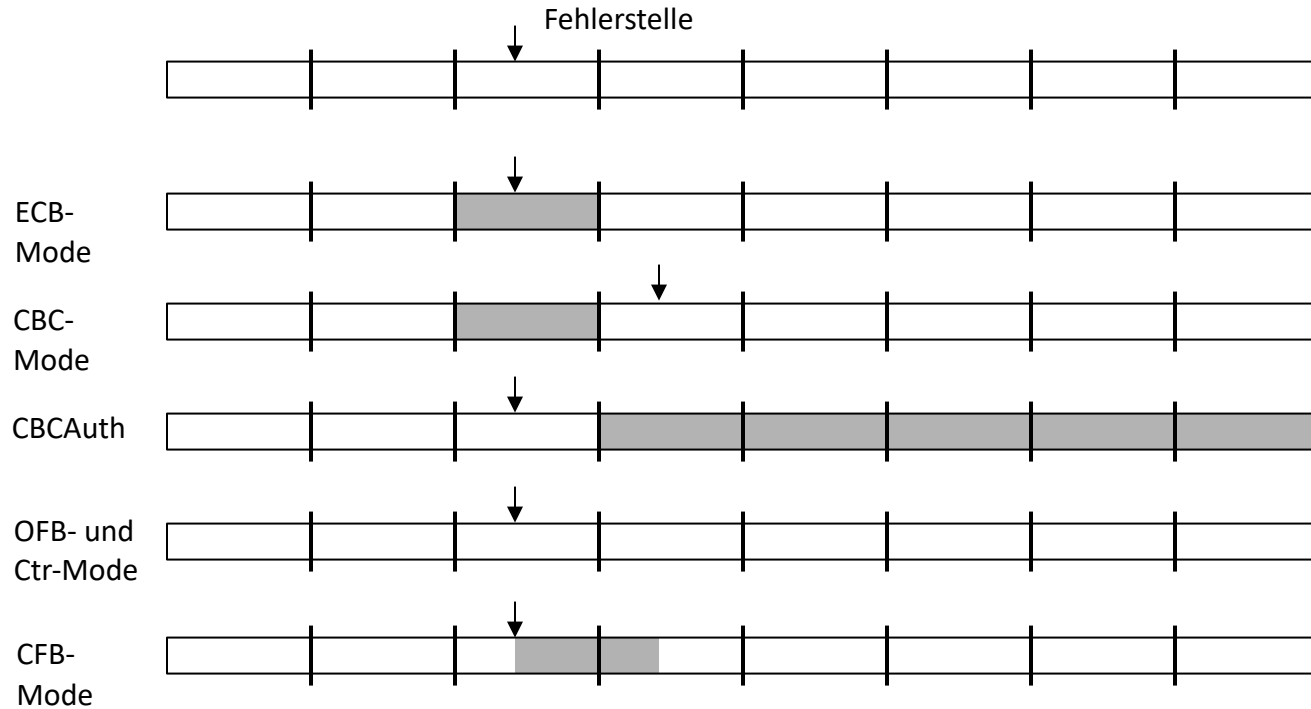
- Cipher Feedback (CFB)



- CFB hat heute eher historische Bedeutung.

Betriebsarten von Blockchiffren

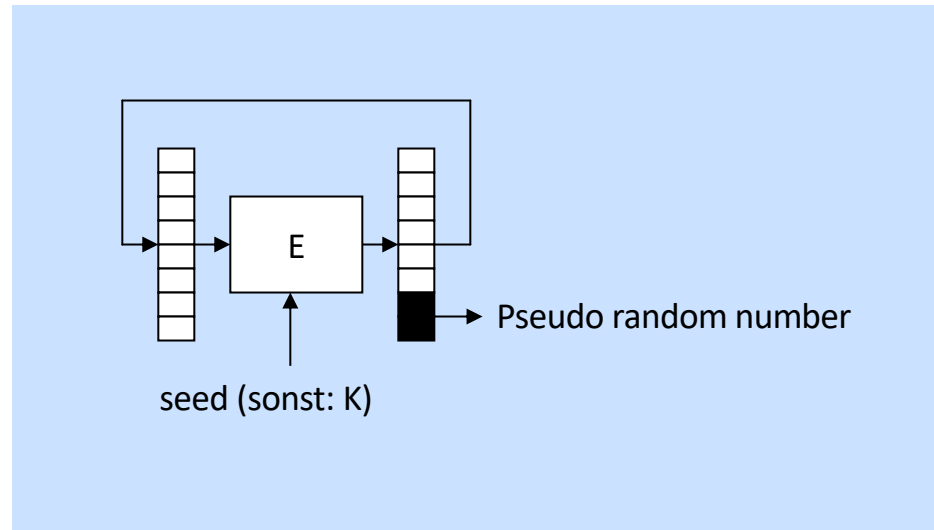
■ Fehlerfortpflanzung



Modus	Vorteile	Nachteile
ECB	<ul style="list-style-type: none"> • Direktzugriff möglich • keine Fehlerfortpflanzung bei additiven Fehlern 	<ul style="list-style-type: none"> • Fehlerfortpflanzung in alle nachfolgenden Blöcke bei Synchronisationsfehlern • unerkennbare additive Veränderungen möglich • gezieltes Einfügen und Entfernen von Blöcken möglich • gleiche Klartextblöcke liefern gleiche Chiffretextblöcke • Codebuchanalyse möglich
CBC	<ul style="list-style-type: none"> • gleiche Klartextblöcke liefern unterschiedliche Chiffretextblöcke • Manipulationen sind erkennbar (CBC-Auth) • Kryptanalyse erschwert gegenüber ECB-Modus 	<ul style="list-style-type: none"> • Fehlerfortpflanzung in alle nachfolgenden Blöcke bei Synchronisationsfehlern • Fehlerfortpflanzung in den Folgeblock bei additiven Fehlern • kein Direktzugriff möglich
OFB, Counter	<ul style="list-style-type: none"> • keine Fehlerfortpflanzung bei additiven Fehlern 	<ul style="list-style-type: none"> • Fehlerfortpflanzung in alle nachfolgenden Bits bei Synchronisationsfehlern • unerkennbare additive Veränderungen möglich • kein Direktzugriff möglich (nur OFB)
CFB	<ul style="list-style-type: none"> • Schlüsselstrom abhängig von Klartextstrom • Kryptanalyse erschwert gegenüber OFB-Modus • Manipulationen sind erkennbar • selbstsynchronisierender Modus 	<ul style="list-style-type: none"> • Fehlerfortpflanzung in den Folgeblock bei additiven Fehlern • kein Direktzugriff möglich

Konstruktionen aus einer symmetrischen Blockchiffre

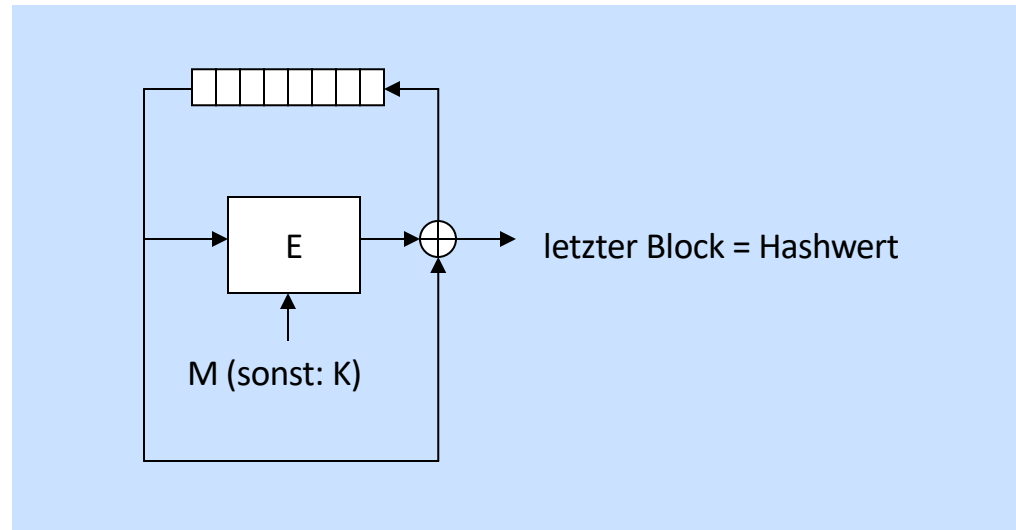
- Pseudozufallszahlengenerator



Konstruktionen aus einer symmetrischen Blockchiffre

■ Hashfunktion

- Aus Sicherheitsgründen sollte die Schlüssellänge nicht wesentlich länger sein als die Blocklänge

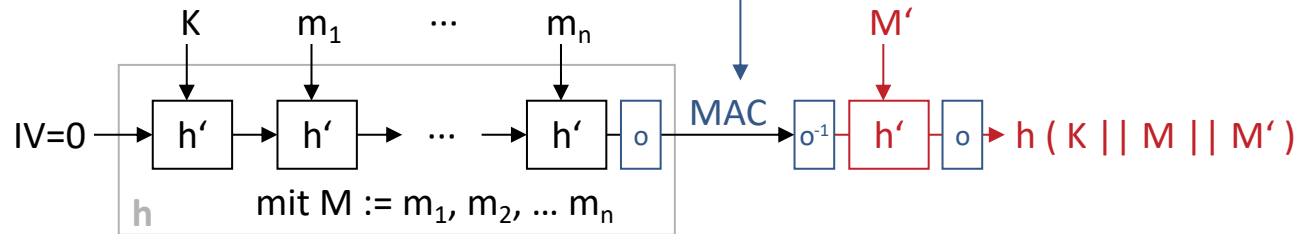


Unsichere Konstruktion von MACs aus Hashfunktionen

- Häufig verwendet, aber bei Verwendung von iterierten Hashfunktionen wie MD5, SHA-1, SHA-256/512 unsicher:

- Length Extension Attack (Skizze)

- existenzieller Angriff
- Gegeben sind MAC, M.
- Angreifer berechnet ohne Kenntnis von K: $h(K || M || M')$



Aufgrund der iterierten Anwendung einer (internen) Struktur von h lässt sich mit einem M' »weiterrechnen« und so ein gültiger MAC erzeugen.

Unter https://github.com/iagox86/hash_extender und <https://github.com/bwall/HashPump> sind Beispielimplementierungen für die Length Extension Attack zu finden.

(Un)sichere Konstruktion von MACs aus Hashfunktionen

- Naive, aber u.U. bereits sichere Abhilfe:

- K nicht nur M voranstellen, sondern auch M nachstellen:

$$\text{MAC} = h(K \parallel M \parallel K)$$

- Length Extension Attack sowohl am Anfang als auch am Ende von M wird erschwert

- Restproblem:

Zumindest für $\text{MAC} = h(M \parallel K)$ wurde gezeigt, dass, wenn ein Angreifer eine Kollision für zwei ungleiche Nachrichten M1 und M2 mit $h(M1)=h(M2)$ findet, dann auch für $h(M1 \parallel K) = h(M2 \parallel K)$ leicht eine Kollision konstruiert werden kann.

M. Bellare, R. Canetti, H. Krawczyk: Keying hash functions for message authentication. Proc. Crypto 96, LNCS 1109, Springer, 1996, 1-15

Sichere Konstruktion von MACs aus Hashfunktionen: HMAC

- (Keyed)-Hash MAC (HMAC):

$$\text{MAC} = h((K \oplus \text{opad}) \parallel h((K \oplus \text{ipad}) \parallel M))$$

mit $\text{ipad} = 0x36 \dots 0x36$ und $\text{opad} = 0x5c \dots 5c$ nach RFC 2104

