DI G. Mittenecker • DI E. Neuherz

FH JOANNEUM
University of Applied Sciences

INFORMATIK

DEPARTMENT FÜR ANGEWANDTE INFORMATIK

KRYPTOGRAFIE I

für Studierende an der FH JOANNEUM

Die Autorinnen danken für jedes konstruktive Feedback. Verbesserungsvorschläge und Korrekturen von Anwenderinnen werden nach Möglichkeit zeitnah nach Bekanntwerden umgesetzt, d.h.:

„Bei der Generierung aktueller Versionen - automatische Generierung - werden die aktuellen Änderungen berücksichtigt. “

E-Mail: office@neo-lernhilfen.at

(Version: 2024-12-14 0:53)

FH JOANNEUM
University of Applied Sciences

KRYPTOGRAFIE I

für Studierende an der FH JOANNEUM

Studienjahr 2024/25

2024-11-18

v1.01

(Version: 2024-12-14 0:53)

(Download)

Verantwortlich für den Inhalt:

DI G. Mittenecker & DI E. Neuherz, Graz

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Einführung

1 Grundlagen der Mathematik

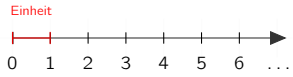
Inhaltsangabe

1.1 Zahlen

Das Wort Zahl kommt vom Verb zählen. Schon sehr früh in der Menschheitsgeschichte lernte der Mensch zu zählen. Die Null wurde dabei aber sehr spät als eigenes Zeichen 0 eingeführt.

Zahlen können auf einem **Zahlenstrahl** bzw. einer **Zahlengerade**[1] grafisch dargestellt werden.

1.1.1 Zahlenstrahl



Eine gerade Linie, auf der die natürlichen Zahlen $\mathbb{N}$ der Größe nach nach angeordnet sind, nennt man **Zahlenstrahl**.

Definition 1.1

Der Zahlenstrahl hat folgende Eigenschaften:

- Er hat einen Anfangspunkt aber keinen Endpunkt.
- Der Pfeil am Ende des Zahlenstrahls bedeutet, dass er unendlich weit verläuft.
- Er beginnt auf der linken Seite mit der Zahl Null.
- Auf ihm ist eine Ordnung erkennbar, d. h. in Richtung des Pfeils werden die Zahlen größer.

Die **Einheitsstrecke** ist der Abstand zwischen der 0 und dem ersten **Markierungsstrich**.

Definition 1.2

Man kann die Einheitsstrecke frei wählen. Üblicherweise wird für sie der Wert 1 verwendet.

1.1.2 Zahlengerade

Eine **Zahlengerade** wird durch die Zahl 0 in zwei Teile geteilt. Rechts von der 0 befinden sich die positiven und links davon die negativen Zahlen.

Definition 1.3

Die Zahlengerade hat folgende Eigenschaften:

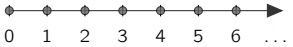
- Die Zahlengerade hat weder Anfangs- noch Endpunkt.
- Der Pfeil an beiden Enden der Zahlengerade bedeutet, dass er auf beiden Seiten unendlich weit verläuft.
- Zahlen in Richtung des rechten Pfeiles werden größer, in Richtung des linken Pfeiles kleiner.

1.2 Zahlenmengen

1.2.1 Menge $\mathbb{N}$ der natürlichen Zahlen

Zahlenstrahl

Die natürlichen Zahlen können auf einem Zahlenstrahl wie folgt dargestellt werden:



Der Pfeil auf der rechten Seite des Zahlenstrahls bedeutet, dass dieser unendlich weit verläuft, Die Menge der natürlichen Zahlen ist unendlich groß ist.

[1] unterscheide zwischen Zahlenstrahl und Zahlengerade

Mengendarstellung

Die natürlichen Zahlen $\mathbb{N}$ können als Menge wie folgt dargestellt werden[2]:

$$\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, \ldots\}$$

Drei Punkte auf der rechten Seite der Mengenklammer bedeuten die Menge ist unendlich groß.

Die Zuordnung der Null zu der Menge $\mathbb{N}$ ist nicht eindeutig definiert. Gemäß der Empfehlung der ÖNORM[3] ist die Null in $\mathbb{N}$ enthalten aber in $\mathbb{N}^*$ nicht!

$$\mathbb{N}^* = \{1, 2, 3, 4, 5, 6, \ldots\}$$

Teilmengen

Man kann die natürlichen Zahlen noch weiters unterteilen, z.B. in $\mathbb{N}^*$, $\mathbb{N}_g$, usw.:

Symbol	Verw.	Interpretation
$\mathbb{N}^*$	$\{1, 2, 3, \ldots\}$	die natürlichen Zahlen ohne Null
$\mathbb{N}_g$	$\{0, 2, 4, \ldots\}$	die geraden natürlichen Zahlen
$\mathbb{N}_u$	$\{1, 3, 5, \ldots\}$	die ungeraden natürlichen Zahlen

Tab. 1.1: *Teilmengen von* $\mathbb{N}$

Eigenschaften

Die Menge der natürlichen Zahlen ist geordnet[4] Auf dem Zahlenstrahl liegt die größere zwei Zahlen immer rechts von der anderen Zahl.

Eine natürliche Zahl heißt **gerade**, wenn sie ohne Rest durch zwei teilbar ist; andernfalls heißt sie **ungerade**.

Definition 1.4

Ungerade Zahlen hinterlassen bei Division durch 2 stets einen Rest von 1, gerade Zahlen den Rest 0.

Rechengesetze

Bei der **Addition** darf man die Summanden vertauschen.

Satz 1.1: Vertauschungsgesetz[5]

$$8 + 7 = 7 + 8$$

Bei der **Addition** darf man die Summanden beliebig zusammenfassen.

Satz 1.2: Verbindungsgesetz[6]

$$3 + 2 + 6 = 3 + (2 + 6) = (3 + 2) + 6$$

[2] Das Formelzeichen $\mathbb{N}$ wurde vom dt. Mathematiker Wedekind 1888 eingeführt

[3] ÖNORM A 6406 bzw. DIN 5473

[4] d.h. vergleicht man zwei verschiedene natürliche Zahlen dann ist eine immer kleiner (oder größer) als die andere Zahl. Für diesen Vergleich wird ein eigenes Symbol verwendet.

[5] Kommutativgesetz

[6] Assoziativgesetz

Bei der **Multiplikation** darf man die Faktoren vertauschen.

Satz 1.3: Vertauschungsgesetz

$$8 \cdot 2 = 2 \cdot 8$$

Bei der **Multiplikation** darf man die Faktoren beliebig zusammenfassen.

Satz 1.4: Verbindungsgesetz

$$3 \cdot 4 \cdot 4 = 3 \cdot (4 \cdot 5) = (3 \cdot 4) \cdot 5$$

Menge der Primzahlen

Von besonderer Bedeutung unter den natürlichen Zahlen sind die Primzahlen.

Eine **Primzahl**[7] ist eine natürliche Zahl **größer** als 1 und ausschließlich nur durch die Zahl **1** und **sich selbst** teilbar ist.

Satz 1.5

Die Menge der Primzahlen wird in der Regel mit dem Symbol $\mathbb{P}$ bezeichnet.

$$\mathbb{P} = \{2, 3, 5, 7, 11, 13, 17, \ldots\}$$

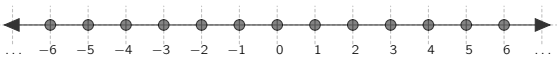
1.2.2 Menge $\mathbb{Z}$ der ganzen Zahlen

Beim Rechnen mit den natürlichen Zahlen ist es nicht möglich, die Differenz von z.B. $4 - 8$ zu berechnen. Dazu ist es notwendig den Zahlenraum um die negativen ganzen Zahlen zu erweitern.

Leonardo von Pisa[8] verwendete erstmals in Europa eine negative Zahl als Lösung einer Gleichung. Viele Mathematiker akzeptierten keine negative Zahlen. Erst im späten 19. Jahrhundert wurden sie allgemein anerkannt.

Zahlengerade

Die ganzen Zahlen können auf einer Zahlengerade wie folgt dargestellt werden:



Die Pfeile an beiden Enden der Zahlengerade bedeuten, dass er auf beiden Seiten unendlich weit verläuft, d.h. die Menge der ganzen Zahlen ist unendlich groß.

Mengendarstellung

Die Menge der ganzen Zahlen $\mathbb{Z}$ kann als Menge wie folgt dargestellt werden:

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

Die drei Punkte auf beiden Seite der Mengenklammer bedeuten, dass die Menge $\mathbb{Z}$ unendlich groß ist.

[7] **lateinisch:** *numerus primus* = erste Zahl
[8] 12. Jhd

Man kann auch die ganzen Zahlen noch weiters unterteilen, z.B. in $\mathbb{Z}^*$, $\mathbb{Z}^-$, usw.:

Symbol	Verw.	Interpretation
$\mathbb{Z}^*$	$\{\ldots -1, 1, 2 \ldots\}$	die ganzen Zahlen ohne Null
$\mathbb{Z}_g$	$\{\ldots -2, 0, 2 \ldots\}$	die geraden ganzen Zahlen
$\mathbb{Z}_u$	$\{\ldots -1, 1, \ldots\}$	die ungeraden ganzen Zahlen
$\mathbb{Z}^+$	$\{1, 2, 3, \ldots\}$	die positiven ganzen Zahlen
$\mathbb{Z}^-$	$\{\ldots, -2, -1\}$	die negativen ganzen Zahlen
$\mathbb{Z}_0^-$	$\{\ldots, -1, 0\}$	die nicht positiven ganzen Zahlen
$\mathbb{Z}_0^+$	$\{0, 1, 2, \ldots\}$	die nicht negativen ganzen Zahlen

Tab. 1.2: *Teilmengen von $\mathbb{Z}$*

Eigenschaften

Die Zahl 0 teilt die Zahlengerade in eine linke und rechte Seite (negative und positive Zahlen).

Eine ganze Zahl die kleiner als Null ist nennen wir eine **negative Zahl**.

Definition 1.5

Eine ganze Zahl die größer als Null ist nennen wir eine **positive Zahl**.

Definition 1.6

Damit bleibt die Frage offen, wohin gehört die Null? Da die Null zwischen den negativen und positiven ganzen Zahlen liegt gilt:

Die Zahl 0 (Null) ist die **einzige Zahl**, die weder negativ noch positiv ist. .

Satz 1.6

Vergleichen

Auch die Menge der ganzen Zahlen ist geordnet, d.h. vergleicht man zwei verschiedene ganze Zahlen, dann ist eine immer kleiner (oder größer) als die andere Zahl.

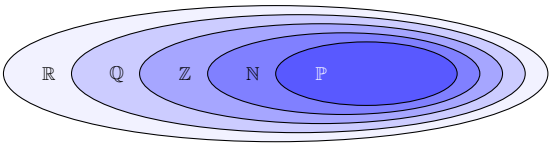
Auf der Zahlengerade liegt die größere Zahl immer rechts von der kleineren Zahl.

1.2.3 Menge $\mathbb{Q}$ der rationalen Zahlen

1.2.4 Menge $\mathbb{R}$ der reellen Zahlen

1.2.5 Zusammenfassung

In der Menge $\mathbb{R}$ sind alle bisher genannten Zahlenmengen enthalten. Folgendes Diagramm stellt diesen Zusammenhang anschaulich dar:



Dabei gilt:

$$\mathbb{P} \subset \mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$$

2 Grundlagen der Algebra

Inhaltsangabe

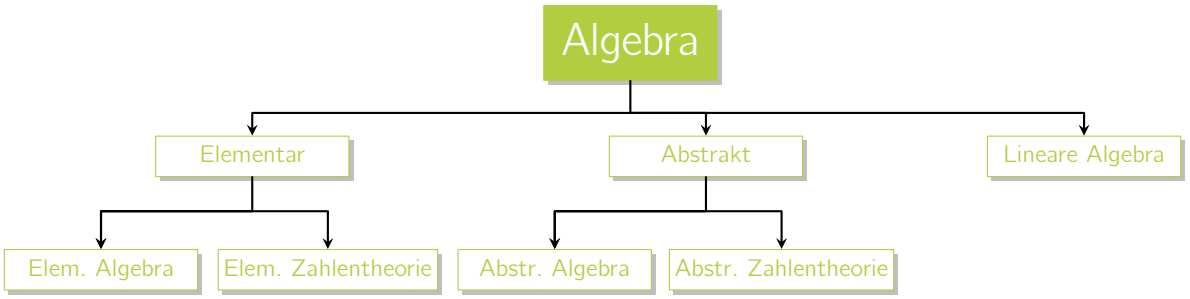


Abb. 2.1: *Algebra als Teilgebiet der Mathematik*

2.1 Elementare Algebra

Jener Bereich der Algebra, welche in der Schulmathematik gelehrt wird. Sie beschäftigt sich mit Zahlenmengen, den Grundrechenarten, den Rechenregeln und die Verwendung von Variablen.

2.1.1 Rechengesetze

Für die Addition und die Multiplikation gelten folgende Rechengesetze:

Bei der **Addition** (**bzw. Multiplikation**) darf man die Summanden (bzw. Faktoren) vertauschen, d.h.:

$$a + b = b + a \qquad \text{bzw.} \qquad a \cdot b = b \cdot a$$

Satz 2.1: Vertauschungsgesetz[1]

Bei der **Addition** (**bzw. Multiplikation**) darf man die Summanden (bzw. Faktoren) beliebig zusammenfassen, d.h.:

$$a + (b + c) = (a + b) + c \quad \text{bzw.} \quad a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

Satz 2.2: Verbindungsgesetz[2]

Zwischen der Addition und der Multiplikation gibt es noch folgenden Zusammenhang:

Eine Summe kann mit einer Zahl multipliziert werden, indem man jeden Summanden mit der Zahl multipliziert und die Produkte addiert, d.h.:

$$(a + b) \cdot c = a \cdot c + b \cdot c \quad \text{bzw.} \quad a \cdot (b + c) = a \cdot b + a \cdot c$$

Satz 2.3: Verteilungsgesetz[3]

2.1.2 Gleichungen und Variablen

Umgang mit Variablen und Gleichungen:

- Terme (Mathematische Ausdrücke)
- (Un)Gleichungen (Berechnung von Lösungen)
- Polynome (linear, quadratisch, kubisch, usw.)
- Bruch- und Wurzelgleichungen
- Exponential- und Logarithmengleichungen

2.1.3 Teiler

Ist eine Zahl z durch eine Zahl t ohne Rest teilbar, so nennt man t einen **Teiler** von z. Man schreibt[4]

$$t \mid z$$

Definition 2.1

Lösung 2.0:

a) Bewerte folgende Aussagen!

Aussage	richtig	falsch
3 | 9	☒	
3 | 0	☒	
5 | 14		☒
7 | 0	☒	
21 | 7		☒

Jede ganze Zahl $z \neq 0$ ist Teiler von 0, weil es bleibt bei der Division kein Rest.

Beobachtung 2.1

[1] Kommutativgesetz
[2] Assoziativgesetz
[3] Distributivgesetz
[4] man spricht: „t ist Teiler von z“

2.2 Elementare Zahlentheorie

2.2.1 Größter gemeinsamer Teiler

Die größte positive ganze Zahl die sowohl a als auch b teilt ($a, b \in \mathbb{N}^*$) bezeichnet man als **größten gemeinsamen Teiler**, kurz $ggT(a, b)$[5].

Definition 2.2

Für kleine Zahlen ist der ggT einfach durch Faktorisierung und Finden gemeinsamer Faktoren zu berechnen.

Der $ggT(r_0, r_1)$ ist das Produkt aller gemeinsamen Primfaktoren von r_0 und r_1.

Beobachtung 2.2

Für große Zahlen, wie sie in der Kryptografie verwendet werden, ist ein Faktorisierung praktisch nicht möglich. Ein effizienteres Verfahren ist der euklidische Algorithmus.

2.2.2 Euklidische Algorithmus

Euklidische Algorithmus [33]

Der gemeinsame Teiler $t = ggT(a, b)$ zweier Zahlen ist auch Teiler der Summe $a + b$ oder Differenz $a - b$ beider Zahlen.

Beobachtung 2.3

Beispiel

Der euklidische Algorithmus nutzt diese Eigenschaft und berechnet den ggT durch schrittweise Berechnung der Differenz zweier Zahlen (siehe Listing 17.2).

```
ggTv1(400,225,0)

400-225=175
    225-175=50
        175-50=125
            125-50=75
                75-50=25
                    50-25=25
                        25-25=0
25
```

Erweiterter euklidische Algorithmus

Für die Berechnung des ggT kann auch der erweiterte euklidische Algorithmus angewendet werden. Er ist im Vergleich zum euklidischen Algorithmus effizienter (siehe Listing 17.3).

```
x_euklid(400,225)

400 - 1 x 225 = 175
225 - 1 x 175 =  50
175 - 3 x  50 =  25
 50 - 2 x  25 =   0
25
```

Anderes Beispiel:

```
x_euklid(67,12)

 67 - 5 x  12 =   7
 12 - 1 x   7 =   5
  7 - 1 x   5 =   2
  5 - 2 x   2 =   1
  2 - 2 x   1 =   0
1
```

In der Kryptografie hat die Berechnung des ggT keine große praktische Bedeutung. Durch Modifikation des Algorithmus ist es aber möglich, in der Modulo-Arithmetik die Inverse einer Zahl zu berechnen. Dazu mehr im nächsten Abschnitt.

[5] **englisch:** *gcd* = great common divider

2.2.3 Modulare Arithmetik

Die modulare Arithmetik wurde von dem deutschen Mathematiker Carl Friedrich Gauß vor mehr als 200 Jahren entwickelt.

Einführendes Beispiel

Welche Eigenschaft verbindet folgende Zahlen 4, 7, 10, 13, 16 bei der Division durch 3?

$$\begin{aligned} 4:3 &= 1 && \text{Rest } 1 \\ 7:3 &= 2 && \text{Rest } 1 \\ 10:3 &= 3 && \text{Rest } 1 \\ 13:3 &= 4 && \text{Rest } 1 \\ 16:3 &= 5 && \text{Rest } 1 \end{aligned}$$

Sie alle haben den gleichen Rest von 1!

Kongruenz von Zahlen

Seien $a, r, m \in \mathbb{Z}$ und $m > 0$. Man schreibt[6]

$$a \equiv r \pmod{m}$$

wenn m ein Teiler von $a - r$ ist. [7]

Definition 2.3

Mit dieser Definition ist der Rest r nicht eindeutig bestimmt. Im Gegenteil, es gibt für einen gegebenen Modul m und einer natürlichen Zahl a unendlich viele Reste.

Restklassen

Restklassenringe [33]

Beim Rechnen mit modulo n wird durch das Modul n die Zahlenmenge $\mathbb{Z}$ in n Restklassen (Mengen) aufgeteilt.

Schreibweise [33]

Lösung 2.1:

Sind folgende Kongruenzen korrekt?

- a $12 \equiv 3 \pmod 9$ Ja, da $9 \mid (12-3) \implies 9 \mid (9)$
- a $12 \equiv 21 \pmod 9$ Ja, da $9 \mid (12-21) \implies 9 \mid (-9)$
- a $12 \equiv -6 \pmod 9$ Ja, da $9 \mid (12-(-6)) \implies 9 \mid (18)$

Wie aus dem Beispiel ersichtlich sind die Reste nicht eindeutig. Es gibt unendlich viele Reste die die Kongruenzen erfüllen.

Lösung 2.2:

Finde eine zu den Kongruenzen äquivalente Darstellung der Form $a = q \cdot m + r$?

- a $12 \equiv 3 \pmod 9$ $12 = 1 \cdot 9 + 3$
- a $12 \equiv 21 \pmod 9$ $12 = (-1) \cdot 9 + 21$
- a $12 \equiv -6 \pmod 9$ $12 = 2 \cdot 9 + (-6)$

Alle Zahlen die obige Kongruenzen erfüllen sind in folgender Restklasse enthalten:

$$R_3 = \{\ldots, -15, \boxed{-6}, \boxed{3}, 12, \boxed{21}, \ldots\}$$

Für den Modul $m = 9$ ergeben sich damit folgende 9 Restklassen (Mengen):

$$\begin{aligned} R_0 &= \{\ldots, -27, -18, -9, \boxed{0}, 9, 18, 27, \ldots\} \\ R_1 &= \{\ldots, -26, -17, -8, \boxed{1}, 10, 19, 28, \ldots\} \\ R_2 &= \{\ldots, -25, -16, -7, \boxed{2}, 11, 20, 29, \ldots\} \\ R_3 &= \{\ldots, -24, -15, -6, \boxed{3}, 12, 21, 30, \ldots\} \\ R_4 &= \{\ldots, -23, -14, -5, \boxed{4}, 13, 22, 31, \ldots\} \\ R_5 &= \{\ldots, -22, -13, -4, \boxed{5}, 14, 23, 32, \ldots\} \\ R_6 &= \{\ldots, -21, -12, -3, \boxed{6}, 15, 24, 33, \ldots\} \\ R_7 &= \{\ldots, -20, -11, -2, \boxed{7}, 16, 25, 34, \ldots\} \\ R_8 &= \{\ldots, -19, -10, -1, \boxed{8}, 17, 26, 35, \ldots\} \end{aligned}$$

Alle Elemente einer Restklasse verhalten sich äquivalent.

Beobachtung 2.4

Die Elemente einer Restklasse sind alle gleichberechtigt, deshalb kann man auch einen bevorzugten Repräsentanten wählen.

Für einen gegenen Modul m kann jede ganze Zahl $a \in \mathbb{Z}$ wie folgt dargestellt werden:

$$a = q \cdot m + r \qquad \text{mit} \qquad 0 \leq r < m$$

Beobachtung 2.5

Was ist damit gewonnen? Man ist nicht mehr genötigt mit den Restklassen, also mit unendlichen Mengen zu hantieren. In unserem Beispiel können wir mit 9 Zahlen {0,1,2,3,4,5,6,7,8} rechnen.

Eigentlich sollten diese Zahlen als eigene Symbole ihre Rolle als Repräsentant einer Restklasse (Menge) ausgezeichnet werden. Bei der Modulo-Arithmetik verzichtet man darauf und rechnet mit Repräsentanten wie mit ganzen Zahlen.

Berechnung einer Inversen

Durch Variation des erweiterte euklidische Algorithmus ist es möglich, in der der Modulo-Arithmetik die Inverse einer natürlichen Zahl zu berechnen (siehe Listing 17.4).

```
inv_euklid(67,12)

i         m         q         r         s         t
-------------------------------------------------------
 1        67         5         7         1        -5
 2        12         1         5        -1         6
 3         7         1         2         2       -11
 4         5         2         1        -5        28
28
```

Mit obigen Algorithmus berechnet sich die Inverse wie folgt:

$$12^{-1} = 28 \pmod{67}$$

Wenn eine Zahl mir ihrer Inversen multipliziert wird, ergibt dies laut Definition den Wert 1:

$$a \cdot a^{-1} \equiv 1 \pmod{67}$$

Damit ist es möglich das Ergebnis des Programmes mit $a = 12, a^{-1} = 28$ zu überprüfen:

$$\begin{aligned} 12 \cdot 28 &\equiv 12 \cdot 7 \cdot 4 \pmod{67} \\ 84 \cdot 4 &\equiv 17 \cdot 4 \pmod{67} \\ 68 &\equiv \underline{1} \pmod{67} \end{aligned}$$

[6] man spricht: „a ist kongruent r modulo m“

[7] Man bezeichnet m als Modul und r als den Rest

2.2.4 Sätze der Zahlentheorie

Die eulersche Phi-Funktion

Die Anzahl der ganzen Zahlen in $\mathbb{Z}_m$, die teilerfremd zu m sind, wird als Eulersche Phi-Funktion $\phi(m)$ bezeichnet.

Definition 2.4

Lösung 2.3:

Berechne die Anzahl der ganzen Zahlen in $\mathbb{Z}_m$, die teilerfremd zu m sind.

c) $m = 6$, $\mathbb{Z}_m = \mathbb{Z}_6 = \{0,1,2,3,4,5\}$

$ggT(0,6) = 6$ × $ggT(2,6) = 2$ × $ggT(4,6) = 2$ ×

$ggT(1,6) = 1$ ✓ $ggT(3,6) = 3$ × $ggT(5,6) = 1$ ✓

Genau zwei Zahlen (1 und 5) sind teilerfremd zu 6. Deshalb ergibt sich folgendes Wert für die Phi-Funktion $\phi(6) = 2$.

c) $m = 5$, $\mathbb{Z}_m = \mathbb{Z}_5 = \{0,1,2,3,4\}$

$ggT(0,5) = 5$ × $ggT(2,5) = 1$ ✓ $ggT(4,5) = 1$ ✓

$ggT(1,5) = 1$ ✓ $ggT(3,5) = 1$ ✓

Alle Zahlen außer Null sind teilerfremd zu 5. Deshalb ergibt sich folgendes Wert für die Phi-Funktion $\phi(5) = 4$.

Die Phi-Funktion kann effizient mit Hilfe des folgenden Satzes berechnet werden, wenn die Faktorisierung von m bekannt ist.

Sind für m die Primfaktoren p_i mit ihren Potenzen e_i bekannt, d.h. man kennt folgende Darstellung:

$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n}$ dann gilt: $\phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})$

Satz 2.4

Nur wenn die Faktorisierung von m bekannt ist, kann die eulersche Phi-Funktion effizient berechnet werden.

Beobachtung 2.6

Aus dem Satz folgt die Berechnung von der eulerschen Phi-Funktion für eine Primzahl p:

$$\phi(p) = \prod_{i=1}^{n} (p^1 - p^{1-1}) = p - p^0 = \underline{p - 1}$$

Der kleine fermatsche Satz

Der kleine fermatsche Satz wird vor allem für Primzahlentests und andere Bereiche der Kryptografie angewendet:

Seien $a \in \mathbb{Z}$ und p eine Primzahl, dann gilt:

$$a^p \equiv a \pmod{p}$$

Satz 2.5: Kleiner fermatsche Satz

Durch Umformung erhalten wir auch eine alternative Form des Satzes:

$$\begin{aligned} a^p &\equiv a \pmod{p} \quad \big| \cdot a^{-1} \\ a^p \cdot a^{-1} &\equiv a \cdot a^{-1} \pmod{p} \\ a^{p-1} &\equiv 1 \pmod{p} \end{aligned}$$

$$a^{p-1} \equiv 1 \pmod{p}$$

Der Satz von Euler

Seien $a, m \in \mathbb{Z}$ mit $ggT(a, m) = 1$, dann gilt:

$$a^{\phi(m)} \equiv 1 \pmod{m}$$

Satz 2.6: Satz von Euler

Der Satz von Euler ist eine Verallgemeinerung des kleinen fermatschen Satzes auf beliebige ganzzahlige Moduln die nicht prim sind:

Der kleine fermatsche Satz ist mit $m = p$ ein Spezialfall des Satzes von Euler.

$$\begin{aligned} a^{\phi(p)} &\equiv 1 \pmod{p} \quad \text{mit } \phi(p) = p - 1 \\ a^{p-1} &\equiv 1 \pmod{p} \end{aligned}$$

Der Satz ist auf Ringe ganzer Zahlen $\mathbb{Z}_m$ anwendbar.

2.2.5 Anwendung für RSA

Eine wichtige Anwendung ist der RSA-Algorithmus.

Gegeben sind zwei Primzahlen p_1, p_2 und die Zahlen n, d, e mit folgenden Eigenschaften:

$$\begin{aligned} &n = p_1 \cdot p_2 \Rightarrow \phi(n) = (p_1 - 1) \cdot (p_2 - 1) \\ &e = \text{beliebig} \quad \text{mit} \quad 1 < e < \phi(n) \\ &d \equiv e^{-1} \pmod{\phi(n)} \quad \text{und} \quad y \equiv x^e \pmod{n} \end{aligned}$$

dann gilt:

$$y^d \equiv (x^e)^d \equiv x^{e \cdot d} \pmod{n} \Rightarrow \underline{y^d \equiv x \pmod{n}}$$

Behauptung 2.7

Beweis

Wir unterscheiden zwei Fälle:

Fall I: $ggT(x, n) = 1$

$$\begin{aligned} d \cdot e &\equiv 1 \pmod{\phi(n)} && (1) \\ d \cdot e &= 1 + k \cdot \phi(n) && (2) \\ y^d &\equiv x^{1 + k \cdot \phi(n)} \equiv x^1 \cdot x^{k \cdot \phi(n)} \pmod{n} && (3) \\ y^d &\equiv x \left(x^{\phi(n)}\right)^k \equiv x \cdot (1)^k \pmod{n} && (4) \\ y^d &\equiv \underline{x \pmod{n}} \end{aligned}$$

Beweis 2.1

Hinweise:

(1) weil gilt $d = e^{-1} \Rightarrow d \cdot e = e^{-1} \cdot e = 1$

(2) nach Definition des Modul-Operators

(3) eingesetzt in $y^d \equiv x^{d \cdot e} \pmod{n}$

(4) weil gilt $ggT(x, n) = 1 \Rightarrow x^{\phi(n)} \equiv 1 \pmod{n}$

Fall II: $ggT(x, n) \neq 1$

$$\begin{aligned} x &= r \cdot p_2 && (5) \\ 1 &\equiv x^{\phi(p_1)} \pmod{p_1} \quad \big|^k && (6) \\ 1^k &\equiv \left(x^{\phi(p_1)}\right)^k \pmod{p_1} \\ \left(x^{\phi(n)}\right)^k &\equiv \left(x^{(p_1-1)(p_2-1)}\right)^k \equiv \left(x^{\phi(p_1)(p_2-1)}\right)^k && (7) \\ &\equiv \left(\left(x^{\phi(p_1)}\right)^k\right)^{p_2-1} \equiv \left(1^k\right)^{p_2-1} && (8) \\ &\equiv 1 \pmod{p_1} \\ \left(x^{\phi(n)}\right)^k &= 1 + u \cdot p_1 \quad \big| \cdot x && (9) \\ x \cdot \left(x^{\phi(n)}\right)^k &= x + x \cdot u \cdot p_1 = x + r \cdot p_2 \cdot u \cdot p_1 \\ &= x + r \cdot u \cdot n \equiv \underline{x \pmod{n}} \equiv y^d && (10) \end{aligned}$$

Beweis 2.2

Hinweise:

(5) da p_1, p_2 Primzahlen: p_1 oder p_2 ist Faktor von x

(6) weil $p_2 \mid x \Rightarrow ggT(x, p_1) = 1$

(7) weil $\phi(n) = \phi(p_1 \cdot p_2) = (p_1 - 1)(p_2 - 1)$

(8) siehe 3. Zeile

(9) nach Definition des Modul-Operators

(10) wegen (3)

2.3 Abstrakte Algebra

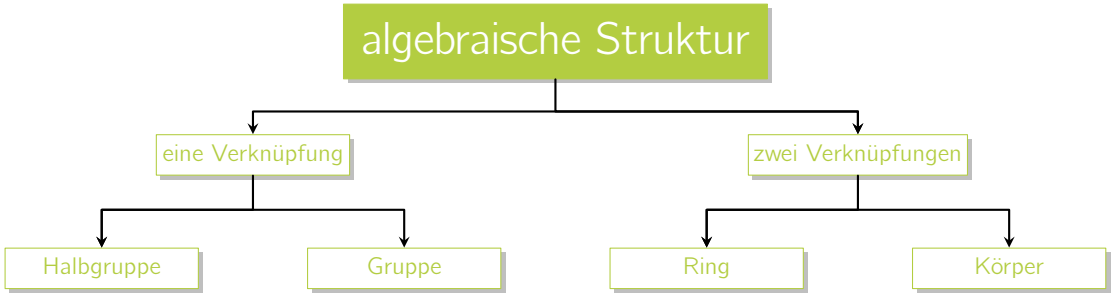


Abb. 2.2: *Algebraische Strukturen*

Im Gegensatz zur elementaren Algebra, beschäftigt sich die abstrakte Algebra mit den Strukturen von (Zahlen)Mengen, deren (inneren) Verknüpfungen (Rechenoperationen) und deren Eigenschaften.

In der Mathematik wird Verknüpfung als Oberbegriff für diverse Operationen gebraucht, wie z.B.:

- arithmetische Operationen
  (Grundrechenarten: Addition, Multiplikation, ...)
- geometrische Operationen
  (Spiegelung, Drehung, ...)
- logische Operationen

Eine zweistellige Verknüpfung ist eine Verknüpfung, die genau zwei Operanden besitzt.

2.3.1 Algebraische Strukturen

Eine algebraische Struktur besteht aus ein oder mehrere Mengen zusammen mit Verknüpfungen auf und zwischen diesen Mengen.

Nachfolgend werden die wichtigsten Strukturen beschrieben und Beispiele anhand der bekannten Zahlenmengen aus der Schulmathematik aufgezählt.

Gruppen

Eine **Gruppe** $\mathbb{G}$ ist eine Menge von Elementen zusammen mit einer Operation $\circ$, welche zwei Elemente von $\mathbb{G}$ verknüpft und folgende Eigenschaften erfüllt:

1. Die Gruppenoperation $\circ$ ist abgeschlossen.
2. Die Gruppenoperation $\circ$ ist assoziativ.
3. Es existiert genau ein neutrales Element 1.
4. Es existiert immer ein inverses Element a^{-1}.
5. Ist die Gruppenoperation $\circ$ kommutativ dann heißt die Gruppe $\mathbb{G}$ abelsch.

Definition 2.1

1. **Abgeschlossenheit** (Innere Verknüpfung)
   Für alle $a, b \in \mathbb{G}$ gilt $a \circ b = c \in \mathbb{G}$
2. **Assoziativität** (Verbindungsgesetz)
   Für alle $a, b, c \in \mathbb{G}$ gilt $a \circ (b \circ c) = (a \circ b) \circ c$
3. **Null-Element** (Neutrales Element)
   Es existiert **genau ein** neutrales Element $0 \in \mathbb{G}$ mit $0 \circ a = a \circ 0 = a$ für alle $a \in \mathbb{G}$.
4. **Inverses Element**
   Für jedes $a \in \mathbb{G}$ existiert ein inverses Element a^{-1} mit $a \circ a^{-1} = a^{-1} \circ a = 1$
5. **Kommutivität** (optional)
   Für alle $a, b \in \mathbb{G}$ gilt $a \circ b = b \circ a$

Vereinfacht formuliert ist eine Gruppe ein mathematisches Gebilde in der man mit den aus der Schulmathematik bekannten Rechenregeln rechnen kann.

Als Gruppenoperation haben wir dabei die Addition bzw. die Multiplikation kennengelernt.

Gruppeneigenschaften bekannter Zahlenmengen

Gruppe bzg. Addition						
Axiom	$\mathbb{N}$	$\mathbb{Z}$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{C}$	$\mathbb{Z}_p$
①	✓	✓	✓	✓	✓	✓
②	✓	✓	✓	✓	✓	✓
③	✓	✓	✓	✓	✓	✓
④	-	✓	✓	✓	✓	✓
Gruppe	-	✓	✓	✓	✓	✓
⑤	✓	✓	✓	✓	✓	✓
Abelsch	-	✓	✓	✓	✓	✓

Tab. 2.1: *Gruppeneigenschaft bzgl. Addition*

Gruppe bzgl. Multiplikation						
Axiom	$\mathbb{N}^*$	$\mathbb{Z}^*$	$\mathbb{Q}^*$	$\mathbb{R}^*$	$\mathbb{C}^*$	$\mathbb{Z}_p^*$
①	✓	✓	✓	✓	✓	✓
②	✓	✓	✓	✓	✓	✓
③	✓	✓	✓	✓	✓	✓
④	-	-	✓	✓	✓	✓
Gruppe	-	-	✓	✓	✓	✓
⑤	✓	✓	✓	✓	✓	✓
abelsch	-	-	✓	✓	✓	✓

Tab. 2.2: *Gruppeneigenschaft bzgl. Multiplikation*

Körper

Ein Körper $\mathbb{F}$ ist eine Menge von Elementen welche folgende Eigenschaften erfüllt:

1. Alle Elemente von $\mathbb{F}$ bilden mit $+$ und 0 eine abelsche Gruppe
2. Alle Elemente von $\mathbb{F}$ (mit Ausnahme 0) bilden mit $\cdot$ und 1 eine abelsche Gruppe
3. Die beiden Gruppenoperationen sind über das Distributivgesetz verknüpft.

Definition 2.2

1. **Additive abelsche Gruppe**
   Addition mit dem neutralem Element 0
2. **Multiplikative abelsche Gruppe**
   Multiplikation mit dem neutralem Element 1
3. **Punktrechnung vor Strichrechnung**
   Für alle $a, b, c \in \mathbb{G}$ gilt $a \cdot (b + c) = a \cdot b + a \cdot c$

Körper $\mathbb{F}$						
Axiom	$\mathbb{N}$	$\mathbb{Z}$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{C}$	$\mathbb{Z}_p$
①	-	✓	✓	✓	✓	✓
②	-	-	✓	✓	✓	✓
③	✓	✓	✓	✓	✓	✓
Körper	-	-	✓	✓	✓	✓

Tab. 2.3: *Zahlenmengen mit Körpereigenschaft?*

Ring

Ein **Ring** $\mathbb{R}$ ist eine Menge von Elementen welche folgende Eigenschaften erfüllt:

1. Alle Elemente von $\mathbb{R}$ bilden mit „+“ und 0 eine **abelsche Gruppe**
2. Alle Elemente von $\mathbb{R}^*$ bilden mit „·“ und 1 einen **Monoid**
3. Die beiden Operationen sind über das Distributivgesetz verknüpft.

Definition 2.3

Wenn man die Definitionen von einem Ring und einem Körper vergleicht, so sind sie fast ident.

In einem Ring wird, im Gegensatz zu einem Körper, die Existenz eines multiplikativen inversen Elementes nicht vorausgesetzt.

Beobachtung 2.1

2.3.2 Restklassenringe

Mit der Modulo-Operation kann die algebraische Struktur eines Restklassenringes wie folgt definiert werden:

Der **Restklassenring** $\mathbb{Z}_m$[8] besteht aus:

1. Der Menge $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$ sowie
2. den beiden Rechenoperation „+“ und „·“ sodass gilt:

$$\left.\begin{array}{l} a + b \equiv c \pmod m \\ a \cdot b \equiv d \pmod m \end{array}\right\} a, b, c, d \in \mathbb{Z}_m$$

Definition 2.4

Jede Kombination arithmetischer Operationen in diesem Restklassenring kann als Auswertung in den ganzen Zahlen mit einer **abschließenden** Modulo-Reduktion betrachtet werden. [3, S 19]

Als Endergebnis sind im Ring $\mathbb{Z}$ alle Repräsentanten erlaubt, im Restklassenring $\mathbb{Z}_m$ nur die Zahlen der angegebenen Menge $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$

Beobachtung 2.2

Verknüpfungstabellen in $\mathbb{Z}_3$

Eine Verknüpfungstabelle beschreibt, wie beliebige Elemente addiert oder multipliziert werden und welche die additiven und multiplikativen Inversen Elemente sind.

Mithilfe dieser Tabelle können wir jede Berechnung in dieser Menge ausführen, ohne die Modulararithmetik durchzuführen.

Bei der Verknüpfung in $\mathbb{Z}_3$ sind im Ergebnis nur die Elemente in $\{0, 1, 2\}$ erlaubt.

$(\mathbb{Z}_3, \oplus)$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Tab. 2.4: *Addition in* $\mathbb{Z}_3$

$(\mathbb{Z}_3, \odot)$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Tab. 2.5: *Multiplikation in* $\mathbb{Z}_3$

Man erkennt in den Tabellen das neutrale Element der Addition 0 und der Multiplikation 1. Die dazu korrespondierenden Werte sind zueinander invers.

Ber der Addition ist das zu „1“ inverse Element „2“. Bei der Multiplikation ist das zu „2“ inverse Element „2“.

[8] In der Literatur wird oft die Schreibweise $\mathbb{Z}/m\mathbb{Z}$ verwendet (sprich „Z modulo m“)

In jeder Zeile bzw. Spalte der Additionstabelle kommen alle Elemente von $\mathbb{Z}_3$ genau einmal vor (keine Doppelgänger).

Beobachtung 2.3

Verknüpfungstabellen in $\mathbb{Z}_4$

$(\mathbb{Z}_4, \oplus)$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Tab. 2.6: *Addition in* $\mathbb{Z}_4$

$(\mathbb{Z}_4, \odot)$	1	2	3
1	1	2	3
2	2	0	2
3	3	2	1

Tab. 2.7: *Multiplikation in* $\mathbb{Z}_4$

Verknüpfungstabellen in $\mathbb{Z}_5$

$(\mathbb{Z}_5, \oplus)$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Tab. 2.8: *Addition in* $\mathbb{Z}_5$

$(\mathbb{Z}_5, \odot)$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Tab. 2.9: *Multiplikation in* $\mathbb{Z}_5$

Verknüpfungstabellen in $\mathbb{Z}_6$

$(\mathbb{Z}_6, \oplus)$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

Tab. 2.10: *Addition in* $\mathbb{Z}_6$

$(\mathbb{Z}_6, \odot)$	1	2	3	4	5
1	1	2	3	4	5
2	2	4	0	2	4
3	3	0	3	0	3
4	4	2	0	4	2
5	5	4	3	2	1

Tab. 2.11: *Multiplikation in* $\mathbb{Z}_6$

Nullteiler

In $\mathbb{Z}$ folgt aus $a \cdot b = 0$ immer, dass mindestens einer der beiden Zahlen a, b gleich null sein muß. Anders im Restklassenring $\mathbb{Z}_4$. Betrachten wir die Multiplikationstabelle in $\mathbb{Z}_4$ so gilt $2 \cdot 2 = 0$. Eine Zahl wie 2 wird als **Nullteiler** bezeichnet.

In einem Restklassenring kann das Ergebnis einer Multiplikation gleich null sein, obwohl keiner der Faktoren gleich null ist.

Beobachtung 2.4

Restklassen

Eigenschaften

Nachfolgend vergleichen wir verschiedene Eigenschaften der betrachteten Restklassenringe von $\mathbb{Z}_3$ bis $\mathbb{Z}_6$:

Eigenschaften von $(\mathbb{Z}_m, \odot)$	$(\mathbb{Z}_3, \odot)$	$(\mathbb{Z}_4, \odot)$	$(\mathbb{Z}_5, \odot)$	$(\mathbb{Z}_6, \odot)$
Ring	✓	✓	✓	✓
Doppelgänger	✗	✓	✗	✓
Nullteiler	✗	✓	✗	✓
Inverse der $\odot$	✓	✗	✓	✗
$m \in \mathbb{P}$	✓	✗	✓	✗
Körper	✓	✗	✓	✗

Tab. 2.12: *Eigenschaften der Restklassenringe*

Unter bestimmten Voraussetzungen erfüllen die Restklassenringe $\mathbb{Z}_m$ auch die Eigenschaften eines Körpers.

Beobachtung 2.5

2.3.3 Primkörper

Sei p prim. Der Restklassenring $\mathbb{Z}_p$ wird als **Primkörper** $GF(p)$ oder **endlicher Körper** mit einer primen Anzahl von Elementen bezeichnet.

Satz 2.1

Alle Elemente von $GF(p)$ ungleich null haben eine multiplikative Inverse.

Rechenregeln für Kongruenzen

Bis jetzt haben wir nur die Rechenregeln für die Addition, Subtraktion und Multiplikation kennengelernt. Die Division ist bei der Berechnung von Kongruenzen nicht zulässig. Kongruenzen darf man wie Gleichungen mit Rechnungen kombinieren.

Sei $\mathbb{Z}_m$ ein Restklassenring mit $a, b, c, d \in \mathbb{Z}_m$ dann folgt aus:

$$\left.\begin{array}{l} a \equiv b \pmod m \\ c \equiv d \pmod m \end{array}\right\} \Longrightarrow \left\{\begin{array}{l} a + c \equiv b + d \pmod m \\ a - c \equiv b - d \pmod m \\ a \cdot c \equiv b \cdot d \pmod m \end{array}\right.$$

Satz 2.2

Nach dem Satz spielt es keine Rolle, ob man bei einer Addition bzw. Multiplikation zweier Zahlen:

- zuerst zu Restklassen übergeht, dann rechnet oder
- zuerst rechnet und dann zur Restklasse übergeht.

In beiden Fällen erhält man für die Kongruenz das identische Ergebnis. Auch ist bei der Multiplikation die erste Variante mit weniger Rechenaufwand verbunden.

Beispiele

Lösung 2.0:

Berechne und beachte dabei die Schreibweise:

c) $48 + 25 \pmod{11}$

Variante I:	Variante II:
$= (48 + 25) \bmod 11 =$	$= 48 \bmod 11 + 25 \bmod 11 =$
$= 73 \bmod 11 =$	$= 4 \bmod 11 + 3 \bmod 11 =$
$= 7 \bmod 11$	$= (4 + 3) \bmod 11 = 7 \bmod 11$

Lösung 2.1:

Berechne und beachte dabei die Schreibweise:

c) $48 - 25 \pmod{11}$

Variante I:	Variante II:
$= (48 - 25) \bmod 11 =$	$= 48 \bmod 11 - 25 \bmod 11 =$
$= 23 \bmod 11 =$	$= 4 \bmod 11 - 3 \bmod 11 =$
$= 1 \bmod 11$	$= (4 - 3) \bmod 11 = 1 \bmod 11$

Lösung 2.2:

Berechne und beachte dabei die Schreibweise:

c) $27 \cdot 11 \pmod 5$

Variante I:	Variante II:
$= (27 \cdot 11) \bmod 5 =$	$= 27 \bmod 5 \cdot 11 \bmod 5 =$
$= 297 \bmod 5 =$	$= 2 \bmod 5 \cdot 1 \bmod 5 =$
$= 2 \bmod 5$	$= (2 \cdot 1) \bmod 5 = 2 \bmod 5$

Lösung 2.3:

Berechne x für folgende Kongruenzen in $\mathbb{Z}_7 = \{0, 1, 2, \ldots, 6\}$.

a) $3^8 \equiv x \pmod 7$

$6561 \equiv 2 \pmod 7$ weil $6561 = 937 \cdot 7 + 2$

b) $3^4 \cdot 3^4 \equiv x \pmod 7$

$81 \cdot 81 \equiv 4 \cdot 4 \equiv 16 \equiv 2 \pmod 7$ weil $81 = 11 \cdot 7 + 4$

c) $(3^2)^4 \equiv x \pmod 7$

$9 \cdot 9 \cdot 9 \cdot 9 \equiv 2 \cdot 2 \cdot 2 \cdot 2 \equiv 16 \equiv 2 \pmod 7$ weil $9 = 1 \cdot 7 + 2$

2.3.4 Zusammenfassung

Axiome der Algebra

Wir haben verschiedene algebraische Strukturen kennengelernt. Die Algebra wird durch 11 Axiome begründet. Die Bezeichnung der algebraischen Struktur (Gruppe, etc.) ist davon abhängig, welche der folgenden 11 Axiome erfüllt werden und bilden damit die Basis für das Rechnen in dieser algebraischen Struktur:

Axiome der Addition

1. $\oplus$-**Abgeschlossenheit** (Innere Verknüpfung)
   Für alle $a, b \in \mathbb{G}$ gilt $a \oplus b = c \in \mathbb{G}$
2. $\oplus$-**Assoziativität** (Verbindungsgesetz)
   Für alle $a, b, c \in \mathbb{G}$ gilt $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
3. $\oplus$-**Null-Element** (Neutrales Element)
   Es existiert **genau ein** neutrales Element $0 \in \mathbb{G}$ mit $0 \oplus a = a \oplus 0 = a$ für alle $a \in \mathbb{G}$.
4. $\oplus$-**Inverses Element**
   Für jedes $a \in \mathbb{G}$ existiert ein inverses Element a_i mit $a \oplus a_i = a_i \oplus a = 0$
5. $\oplus$-**Kommutivität**
   Die Summe ist kommutativ, d.h. für alle $a, b \in \mathbb{G}$ gilt $a \oplus b = b \oplus a$

Axiome der Multiplikation

6. $\odot$-**Abgeschlossenheit** (Innere Verknüpfung)
   Für alle $a, b \in \mathbb{G}$ gilt $a \odot b = c \in \mathbb{G}$
7. $\odot$-**Assoziativität** (Verbindungsgesetz)
   Für alle $a, b, c \in \mathbb{G}$ gilt $a \odot (b \odot c) = (a \odot b) \odot c$
8. $\odot$-**Eins-Element** (Neutrales Element)
   Es existiert **genau ein** neutrales Element $1 \in \mathbb{G}$ mit $1 \odot a = a \odot 1 = a$ für alle $a \in \mathbb{G}$.
9. $\odot$-**Inverses Element**
   Für jedes $a \in \mathbb{G}^*$ $(a \neq 0)$ existiert ein inverses Element a^{-1} mit $a \odot a^{-1} = a^{-1} \odot a = 1$
10. $\odot$-**Kommutivität**
    Die Multiplikation ist kommutativ, d.h. für alle $a, b \in \mathbb{G}$ gilt $a \odot b = b \odot a$

Beziehung zwischen Addition und Multiplikation

11. **Distributivgesetz**
    Für alle $a, b, c \in \mathbb{G}$ gilt $a \odot (b \oplus c) = a \odot b \oplus a \odot c$

Das letzte Axiom beschreibt einen Zusammenhang zwischen Addition und Multiplikation (Rangfolge). Aus der Schulmathematik kennen wir: „Punktrechnung vor Strichrechnung“.

Beobachtung 2.6

Strukturen und ihre Eigenschaften

Nachfolgend sind die wichtigsten algebraischen Strukturen und ihre Eigenschaften in einer Tabelle dargestellt.

Algebraische Strukturen								
Axiom	$\mathbb{H}, \oplus$	$\mathbb{M}, \oplus$	$\mathbb{G}, \oplus$	$\mathbb{G}, \odot$	$\mathbb{G}^*, \odot$	$\mathbb{R}, \oplus, \odot$	$\mathbb{F}, \oplus, \odot$	$\mathbb{Z}_p, \oplus, \odot$
(1)	✓	✓	✓	-	-	✓	✓	✓
(2)	✓	✓	✓	-	-	✓	✓	✓
(3)	-	✓	✓	-	-	✓	✓	✓
(4)	-	-	✓	-	-	✓	✓	✓
(5)	-	-	-	-	-	✓	✓	✓
(6)	-	-	-	✓	✓	✓	✓	✓
(7)	-	-	-	✓	✓	✓	✓	✓
(8)	-	-	-	✓	✓	✓	✓	✓
(9)	-	-	-	-	✓	-	✓	-
(10)	-	-	-	-	-	✓	✓	✓
(11)	-	-	-	-	-	✓	✓	✓

Tab. 2.13: *Algebraische Strukturen*

Dabei werden folgende Bezeichnungen verwendet:

- $\mathbb{H}$ ... Halbgruppe
- $\mathbb{M}$ ... Monoid
- $\mathbb{G}$ ... Gruppe
- $\mathbb{R}$ ... Ring
- $\mathbb{F}$ ... Körper (Field)
- $\mathbb{Z}_p$ ... Primkörper

Beispiele von algebraischen Strukturen

Beispiele Algebraischer Strukturen						
Struktur	$\mathbb{N}^*, \oplus$	$\mathbb{N}, \oplus$	$\mathbb{Z}, \oplus$	$\mathbb{Z}, \odot$	$\mathbb{Q}^*, \odot$	$\mathbb{Z}, \ominus$
Halbgruppe	✓	✓	✓	✓	✓	✗
Monoid	✗	✓	✓	✓	✓	✗
Gruppe	✗	✗	✓	✗	✓	✗

Tab. 2.14: *Beispiele mit bekannten Zahlenmengen*

2.4 Abstrakte Zahlentheorie

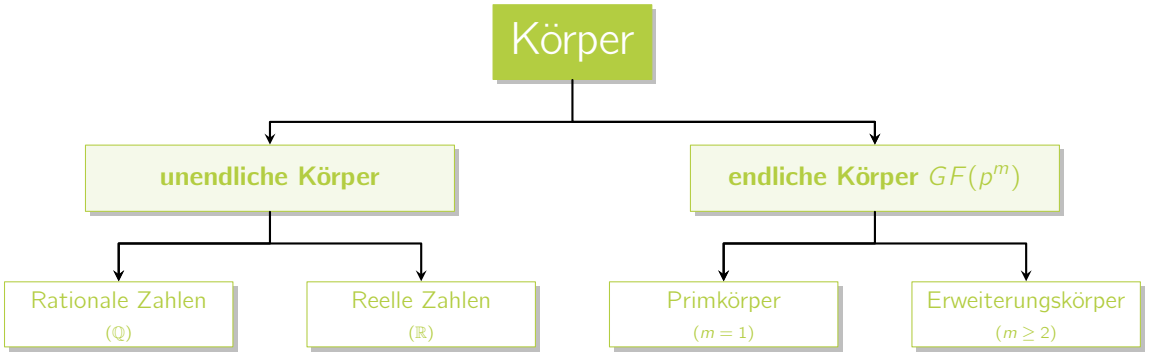


Abb. 2.3: *Körpererweiterung*

Aus der Definition des Körpers können alle notwendigen Voraussetzungen abgeleitet werden, dass, wie von der Schulmathematik gewohnt, in bekannter Weise die Grundrechenarten (Addition, Subtraktion, Multiplikation und Division) funktionieren.

Endliche Körper [15]

Wird die Rechenoperationen Subtraktion als inverse Operation der Addition, die Division als die inverse der Multiplikation interpretiert, kann der Körperbegriff auf andere Mengen und Verknüpfungen ausgeweitet werden.

2.4.1 Endliche Körper

Galoiskörper [33]

In der Kryptografie sind wir immer an Körpern mit einer endlichen Anzahl an Elementen interessiert.

Die Anzahl der Elemente in dem Körper nennt man die Ordnung oder die Kardinalität des Körpers.

Existenz von endlichen Körper

Für die Existenz von endlichen Körper gilt:

> Ein Körper mit der Ordnung q existiert nur, wenn q eine Primzahlpotenz ist, d. h. es gilt $q = p^m$, wobei m eine positive ganze Zahl und p eine Primzahl ist.
>
> *Satz 2.1*

Man nennt p die Charakteristik des endlichen Körpers.

2.4.2 Primkörper

Einfache Beispiele für endliche Körper sind Primkörper (siehe Kapitel 2.3.3) mit der Ordnung $m = 1$.

2.4.3 Erweiterungskörper

Im Kapitel 2.3.2 haben wir im Restklassenring $\mathbb{Z}_4$ mit 4 Elementen festgestellt, dass dieser nicht die Eigenschaften für einen Körper erfüllt.

Verknüpfungstabellen in $\mathbb{Z}_4$

$(\mathbb{Z}_4, \oplus)$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Tab. 2.15: *Addition in $\mathbb{Z}_4$*

$(\mathbb{Z}_4, \odot)$	1	2	3
1	1	2	3
2	2	0	2
3	3	2	1

Tab. 2.16: *Multiplikation in $\mathbb{Z}_4$*

Ist es möglich einen Körper mit 4 Elementen Nullteilfrei zu konstruieren der alle Voraussetzungen für einen Körper erfüllt? Die Antwort ist ja.

Verknüpfungstabellen in $GF(4)$

Nachfolgend die modifizierte Verknüpfungstabelle:

$(\mathbb{F}_4, \oplus)$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{1}$	$\overline{1}$	$\overline{0}$	$\overline{3}$	$\overline{2}$
$\overline{2}$	$\overline{2}$	$\overline{3}$	$\overline{0}$	$\overline{1}$
$\overline{3}$	$\overline{3}$	$\overline{2}$	$\overline{1}$	$\overline{0}$

Tab. 2.17: *Addition GF(4)*

$(\mathbb{F}_4, \odot)$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{1}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{2}$	$\overline{2}$	$\overline{3}$	$\overline{1}$
$\overline{3}$	$\overline{3}$	$\overline{1}$	$\overline{2}$

Tab. 2.18: *Multiplikation GF(4)*

Die Elemente von $\mathbb{Z}_4$ haben die Form[9]:

$$b_1 x + b_0 \qquad \text{mit} \qquad b_i \in \mathbb{Z}_2$$

Interpretiert man die Zahlen als Polynome wie folgt:

Polynomdarstellung			
Element	b_1	b_0	Polynom
$\overline{0}$	0	0	0
$\overline{1}$	0	1	1
$\overline{2}$	1	0	x
$\overline{3}$	1	1	$x+1$

Tab. 2.19: *Polynomdarstellung*

dann erhält man als Elemente:

$$GF(2^2) = GF(4) = \{0, 1, x, x+1\}$$

Verknüpfungstabellen in Polynomdarstellung

$GF(2^2), \oplus$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Tab. 2.20: *Addition GF(4)*

$GF(2^2), \odot$	1	x	$x+1$
1	1	x	$x+1$
x	x	$x+1$	1
$x+1$	$x+1$	1	x

Tab. 2.21: *Multiplikation GF(4)*

Die Reduktion von x^2 ergibt den Rest $x + 1$:

Polynom-Multiplikation

$$\frac{1 \cdot 1}{1} \qquad \frac{x \cdot 1}{x} \qquad \frac{(x+1) \cdot 1}{x+1}$$

* 1 ist das neutrale Element der Multiplikation

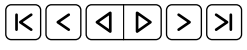


Abb. 2.4: *Rechnen in GF(4)*

[9] $\mathbb{Z}_2 = \{0,1\}$, mit dem irreduziblen Polynom $x^2 - x - 1$

2.4.4 Erweiterungskörper GF(256)

In der Kryptografie wird sehr oft mit einer Datenbreite von 8 Bits gearbeitet. Deshalb ist der endliche Körper $GF(2^8)$ mit 256 Elementen von besonderer Bedeutung.

Beobachtung 2.1

Elemente des GF(256)

Die Elemente des des $GF(2^8)$ haben die Form:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^3 + b_1x + b_0$$

mit $b_i \in \mathbb{Z}_2 = \{0, 1\}$. Als irreduzibles Polynom wurde $(x^8 + x^4 + x^3 + x + 1)$ gewählt.

Addition im GF(256)

Die Addition von zwei Elementen des $GF(2^8)$ erfolgt durch paarweise Addition mod 2 der Koeffizienten gleicher Potenzen (XOR-Verknüpfung).

Lösung 2.0:

Berechne und beachte: $c_i \in \mathbb{Z}_2$:

c $(x^7 + x^6 + x^5 + x^3 + x + 1) + (x^7 + x^5 + x^3 + x^2 + x) \equiv x^6 + x^2 + 1$

$$\begin{array}{llllll} & +x^7 & +x^6 & +x^5 & +x^3 & +x+1 \\ & +x^7 & & +x^5 & +x^3+x^2+x & \\ \hline & & +x^6 & & +x^2 & +1 \end{array}$$

Multplikation im GF(256)

Die Multiplikation von Elementen in $GF(2^8)$ erfolgt in zwei Schritten:

1. Gewöhnlichen Polynom-Multiplikation in $\mathbb{R}$
2. Berechnung des Restes (Reduktion)
   - entweder durch Polynomdivision
   - oder durch Substitution

Gewöhnliche Polynom-Multiplikation

Lösung 2.1:

Berechne und beachte: $c_i \in \mathbb{Z}_2$:

c $(x^6 + x + 1) \cdot (x^7 + x^5 + x^3 + x) \equiv x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^3 + x^2 + x$

Multiplikation:

$$\begin{array}{llllllll} +x^{13} & +x^{11} & +x^9 & & +x^7 & & & \\ & & & +x^8 & & +x^6 & +x^4 & +x^2 \\ & & & & +x^7 & & +x^5 & +x^3 & +x \\ \hline +x^{13} & +x^{11} & +x^9 & +x^8 & & +x^6+x^5 & & +x^3+x^2+x \end{array}$$

Reduktion durch Polynomdivision

Lösung 2.2:

Reduziere folgendes Polynom in $GF(2^8)$:

c $x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^3 + x^2 + x \equiv x^7 + x^6 + x^4 + x^2 + x$

Polynomdivision (Irreduzibles Polynom):

$$\begin{array}{llllll} (x^{13}+x^{11}+x^9+x^8 & & +x^6+x^5 & & +x^3+x^2+x) : (x^8+x^4+x^3+x+1) = x^5+x^3 \\ x^{13} & +x^9+x^8 & & +x^6+x^5 & & \\ \hline & +x^{11} & & & +x^3+x^2+x & \\ & +x^{11} & +x^7+x^6 & & +x^4+x^3 & \\ \hline & & +x^7+x^6 & & +x^4 & +x^2+x \end{array}$$

Reduktion durch Substitution

In $GF(2^8)$ können für Potenzen von x höher als 7 durch äquivalente Polynome (Restklassen) ausgedrückt werden.

Ein **irreduzibles Polynom** ist ein Polynom, das sich nicht als Produkt zweier einfacherer Polynome zerlegen lässt.

Definition 2.1

Irreduzibles Polynom [41]

Für das Irreduzible Polynom $x^8 + x^4 + x^3 + x + 1$ gilt:

$$\begin{aligned} 0 &\equiv x^8 + x^4 + x^3 + x + 1 \pmod{x^8 + x^4 + x^3 + x + 1} \\ -x^8 &\equiv x^4 + x^3 + x + 1 \pmod{x^8 + x^4 + x^3 + x + 1} \\ x^8 &\equiv x^4 + x^3 + x + 1 \pmod{x^8 + x^4 + x^3 + x + 1} \end{aligned}$$

Ein irreduzible Polynom kann aus vorhandenen Tabellen ausgewählt werden.

Beobachtung 2.2

Damit erhalten wir äquivalenten Polynome wie folgt;

$$\begin{aligned} x^8 &\equiv x^4 + x^3 + x + 1 \\ x^9 = x \cdot x^8 &\equiv x^5 + x^4 + x^2 + x \\ x^{10} = x \cdot x^9 &\equiv x^6 + x^5 + x^3 + x^2 \\ x^{11} = x \cdot x^{10} &\equiv x^7 + x^6 + x^4 + x^3 \\ x^{12} = x \cdot x^{11} &\equiv x^8 + x^7 + x^5 + x^4 = x^4 + x^3 + x + 1 + x^7 + x^5 + x^4 \\ &\equiv x^7 + x^5 + x^3 + x + 1 \\ x^{13} = x \cdot x^{12} &\equiv x^8 + x^6 + x^4 + x^2 + x = x^4 + x^3 + x + 1 + x^6 + x^4 + x^2 + x \\ &\equiv x^6 + x^3 + x^2 + 1 \\ x^{14} = x \cdot x^{13} &\equiv x^7 + x^4 + x^3 + x \end{aligned}$$

Lösung 2.3:

Reduziere folgendes Polynom in $GF(2^8)$:

c $x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^3 + x^2 + x \equiv x^7 + x^6 + x^4 + x^2 + x$

Substitution (Einsetzen):

$$\begin{array}{llllll} +x^{13} : & & +x^6 & & +x^3+x^2+x & \\ +x^{11} : & +x^7+x^6 & & +x^4+x^3 & & \\ +x^9 : & & +x^5+x^4 & & +x^2+x & \\ +x^8 : & & & +x^4+x^3 & & +x \\ \hline & +x^7+x^6 & & +x^4 & & +x^2+x \end{array}$$

Inversen Berechnung im GF(256)

Die Berechnung der Inversen erfolgt über den erweiterten Euklidischen Algorithmus.

2.4.5 Analogie Primkörper und GF

Nachfolgend werden Eigenschaften von Primkörper und Erweiterungskörper verglichen.

Exemplarisch verwenden wir dabei als Primkörper $\mathbb{Z}_7$ und als Erweiterungskörper $GF(2^8)$:

Analogie zw. Prim- und Erweiterungskörper		
Anmerkung	$\mathbb{Z}_7$	$GF(2^8)$
Anzahl Elemente	7	256
Reduktion durch	7	$x^8 + x^4 + x^3 + x + 1$
Modul ist	Primzahl	irreduzibles Polynom

Tab. 2.22: *Vergleich Primkörper und GF*

3 Grundlagen der Kryptologie

Inhaltsangabe

3.1 Teilbereiche

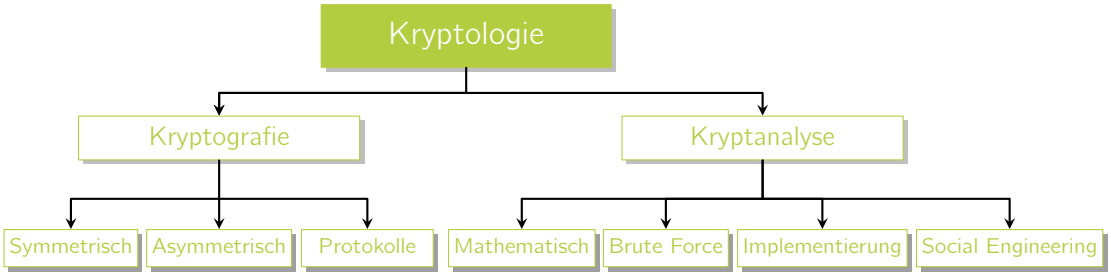


Abb. 3.1: *Überblick der Kryptologie*

3.1.1 Kryptologie

Die **Kryptologie**[1] als Wissenschaft beschäftigt sich mit dem Verschlüsseln und Entschlüsseln von Informationen.

Definition 3.1

Bis zur Einführung der Computer, wurden ausschließlich Verschlüsselungsverfahren betrachtet. Erst mit Einsatz des elektronischen Datenverkehrs wurde die Kryptologie um weitere Themenbereiche ergänzt.

3.1.2 Kryptografie

Die **Kryptografie**[2] beschäftigt sich mit der Absicherung von Daten, z. B. der Verschlüsselung von Daten

Definition 3.2

In der Kryptografie unterscheidet an weiters:

Bei **Symmetrische Algorithmen** verwenden zwei Parteien für die Ver- und Entschlüsselung einen gemeinsamen **geheimen Schlüssel**.

Definition 3.3

Die Klassische Kryptografie bis zum Jahr 1976 sind ein Teilgebiet der Symmetrischen Kryptografie.

Bei der **Asymmetrischen Kryptografie** besitzt ein Teilnehmer einen **privaten**[3] und einen **öffentlichen Schlüssel**[4].

Definition 3.4

Die Idee der Asymmetrischen Kryptografie wurde im Jahr 1976 von Whitfield Diffie, Martin Hellman und Ralph Merkle als neue Art der Kryptografie eingeführt.

Kryptografische **Protokolle** sind Anwendungen die auf kryptografische Algorithmen, sowohl symmetrisch als auch asymmetrisch, basieren.

Definition 3.5

3.1.3 Kryptanalyse

Die **Kryptanalyse** beschäftigt sich mit dem Brechen von Kryptosystemen. Sie ist von zentraler Bedeutung für die moderne Kryptografie.

Definition 3.6

Das Brechen von Codes ist eine eigene wissenschaftiche Disziplin, d. h. die viele Krypto-Analysten sind Wissenschaftler. [3, S 2]

Kryptologie [45]

Kryptographie [44]

Kryptoanalyse [43]

[1] **griechisch:** *kryptos* = versteckt, verborgen, geheim
[2] **griechisch:** *graphein* = schreiben (Geheimschrift)
[3] geheimen Schlüssel
[4] allgemein bekannten Schlüssel

3.2 Terminologie

3.2.1 Kryptografie

Klartext

Als **Klartext** wird die unverschlüsselte Nachricht bzw. Datenblock bezeichnet.

Definition 3.1

Geheimtext

Als **Geheimtext** (Chiffrat, Chiffre) wird ein Text bezeichnet, der durch die Verschlüsselung mithilfe eines kryptografischen Verfahrens in eine unverständliche Textfolge verändert wurde.

Definition 3.2

Verschlüsselung

Die **Verschlüsselung** (auch **Chiffrierung**) ist die Umwandlung von Klartext (Informationen) in einen Geheimtext (Chiffrat, Schlüsseltext).

Definition 3.3

Bei der Verschlüsselung wird ein geheimzuhaltender Schlüssel, der nur den befugten Personen bekannt sein darf, verwendet.

Entschlüsselung

In der Kryptologie ist die **Entschlüsselung** (auch **Dechiffrierung**) die Umkehrung der Verschlüsselung, d.h. ein Geheimtext wird wieder in den ursprünglichen Klartext umgewandelt.

Definition 3.4

Sprachlich wird die Entschlüsselung für die befugte Tätigkeit des legitimen Empfängers bezeichnet.

Entzifferung

Der Versuch einer dritten unbefugten Partei den Klartext zu gewinnen wird üblicherweise mit Entzifferung bezeichnet.

Bei der **Entzifferung** ist im Gegensatz zur Entschlüsselung, der Versuch aus einem Geheimtext ohne vorherige Kenntnis des Schlüssels den Klartext zu gewinnen.

Definition 3.5

Schlüssel

Als **Schlüssel** wird eine Information bezeichnet, die ein kryptografisches Verfahren benötigt um eine Nachricht zu ver- oder entschlüsseln.

Definition 3.6

Im einfachsten Fall ist der Schlüssel ein Kennwort, bei computerbasierten Verfahren eine Bitfolge.

Sicherer Kanal

Als **sicheren Kanal** (**geheimen Kanal**) bezeichnet man in der Kryptografie eine technisch (abhörsicher) oder organisatorisch (vertrauenswürdige Bote) abgesicherte Übertragung.

Definition 3.7

Unsicherer Kanal

Ein unsicherer Kanal kann von unauthorisierten Dritten abgehört werden.

Definition 3.8

3.2.2 Sicherheitsdienste

In der Literatur werden die Begriffe Sicherheitsdienste und Sicherheitsziele synonym verwendet.

Vertraulichkeit

Bei der **Geheimhaltung** (**Vertraulichkeit**) bekommen nur autorisierte Benutzer Zugang zu der Information.

Definition 3.9

Authentizität

Bei der **Authentizität** kann ein Anwender die Identität eines Teilnehmers oder Sender einer Nachricht zweifelsfrei feststellen.

Definition 3.10

Integrität

Bei der **Integrität** von Daten wird sichergestellt, das die Daten während der Übertragung nicht verändert wurden.

Definition 3.11

Beweisbarkeit

Bei der **Beweisbarkeit** (**Nichtzurückweisbarkeit**) kann der Sender einer Nachricht nicht abstreiten, dass die Nachricht von ihm stammt.

Definition 3.12

Identifikation

Bei der **Identifikation** kann die Identität eines Aktueurs (Person oder Gerät) eindeutig ermittelt werden.

Definition 3.13

Ressourcen-Zugriff

Nur dafür vorgesehene Akteure bekommen Zugriff auf bestimmte Ressourcen (z. B. Drucker).

Definition 3.14

Verfügbarkeit

Bei der **Verfügbarkeit** wird sichergestellt, dass ein elektronisches System verfügbar ist.

Definition 3.15

Physikalische Sicherheit

Die **Physikalische Sicherheit** bietet Schutz gegen physikalische Manipulation und/oder Aktionen, wenn physikalische Manipulation festgestellt wurden.

Definition 3.16

Anonymität

Bei der **Anonymität** ist eine Identität gegen Offenlegung und Missbrauch geschützt.

Definition 3.17

Klartext [42]

Geheimtext [39]

Verschlüsselung [53]

Entschlüsselung [37]

Entschlüsselung [37]

Schlüssel [49]

4

Grundlagen der Informationstechnologie

Inhaltsangabe

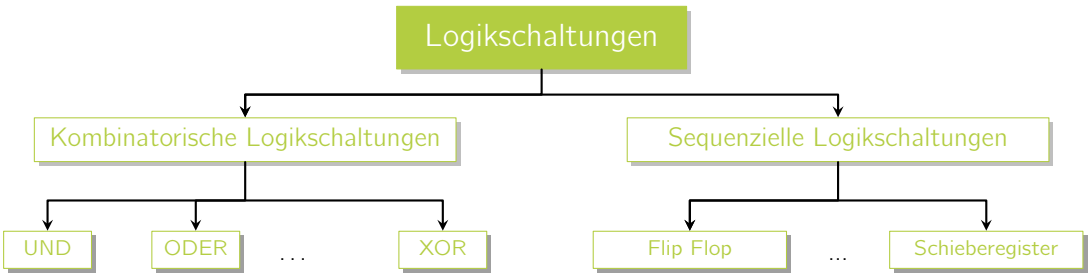


Abb. 4.1: *Einteilung von Logikschaltungen*

4.1 Logikschaltungen

Der erste funktionierende Bipolartransister wurde in den Bell Laboratories (USA), unter anderem von John Bardeen, William Shockley und Walter Brattain, entwickelt und firmenintern am 23. Dezember 1947 präsentiert.

Kombinatorische Logikschaltungen

Bei einer **kombinatorische Logikschaltung** (Schaltnetz) ist der Ausgangszustand nur vom Eingangszustand abhängig.

Definition 4.1

Kombinatorische Logikschaltungen bestehen aus Logikgattern und können auf drei Arten beschrieben werden:

- Logikdiagramm: Verdrahtung der Logik-Gatter
- Boolsche Algebra: Algebraischer Ausdruck
- Wahrheitstabelle: Liste aller Ausgangszustände

Sequenzielle Logikschaltungen

Bei einer **sequenziellen Logikschaltung** (Schaltwerk) ist mindestens ein Ausgang auf mindestens einen Eingang rückgekoppelt.

Durch die Rückkopplung erhält die Schaltung einen speichernden Charakter (Gedächnis).

Beobachtung 4.1

Bei einer **sequenziellen Logikschaltung** (Schaltnetz) ist der Ausgangszustand vom aktuellen und des vorherigen Zustandes abhängig.

Definition 4.2

Sequenziell bedeuted, dass die Verarbeitung in einer Sequenz nacheindander ablaufen. Dabei bestimmt ein Taktsignal den Zeitpunkt der Auswertung von Ein- und Ausgängen.

4.2 Logikgatter

Heutzutage bestehen Schaltungen für Logikgatter hauptsächlich aus Transistoren.

Ein **Transistor** ist ein elektronisches Bauelement zum Steuern oder Verstärken elektrischer Spannungen oder Ströme.

Definition 4.3

Hauptanwendungsgebiet der Transistoren ist der Einsatz in integrierten Schaltungen, wie z. B. RAM-/Flash-Speichern, Mikroprozessoren und Logikgattern.

Ein **integrierter Schaltkreis** ist eine auf einem dünnen, einige Millimeter großen Halbleiter-Material aufgebrachte **elektronische Schaltung**.

Definition 4.4

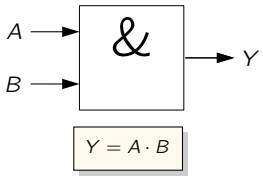
Eine **Wahrheitstabelle** ist eine tabellarische Darstellung der möglichen Ausgangszustände Y in Abhängigkeit von den Eingangszuständen.

Definition 4.5

4.2.1 UND-Gatter

Beim **Und-Gatter** wird der Ausgang logisch „1“ wenn am Eingang A **und** Eingang B logisch „1“ anliegt (und, alle Eingänge).

Definition 4.6



$Y = A \cdot B$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Tab. 4.1: *UND-Wahrheitstabelle*

Als Verknüpfungszeichen wird aus der Mathematik das Symbol für die Multiplikation „·“ verwendet.

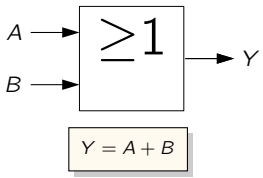
Den Ausgangszustand Y in der Wahrheitstabelle erhält man auch durch Multiplikation der beiden Eingänge A und B.

Beobachtung 4.2

4.2.2 ODER-Gatter

Beim **Oder-Gatter** wird der Ausgang logisch „1“ wenn am Eingang A **oder** Eingang B logisch „1“ anliegt (und/oder).

Definition 4.7



$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 4.2: *ODER-Wahrheitstabelle*

Als Verknüpfungszeichen wird aus der Mathematik das Plus-Symbol für die Addition „+“ verwendet.

Bei der Kombination von UND- und ODER-Gatter können auch die Rechengesetze aus der Mathematik verwendet werden.

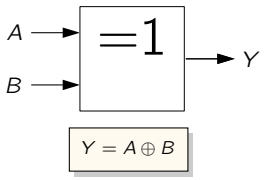
Beobachtung 4.3

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

4.2.3 XOR-Gatter

Beim **Exklusiv-Oder-Gatter**(**XOR**[1]) wird der Ausgang logisch „1“ wenn **entweder** am Eingang A **oder** B logisch „1“ anliegt (entweder/oder).

Definition 4.8



$Y = A \oplus B$

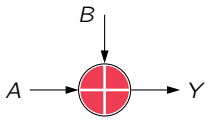
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 4.3: *XOR-Wahrheitstabelle*

4.3 Logikschaltungen

4.3.1 Addition modulo 2

Mathematisch entspricht die XOR-Wahrheitstabelle einer Addition modulo 2 und kann auch durch einen Kreis mit einem Plus-Symbol ⊕ dargestellt werden.



$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 4.4: *Modulo-2-Addition*

Sequentielle Logik

4.3.2 Halbaddierer

Ein **Halbaddierer** addiert zwei einstellige Binärzahlen. Das zweistellige Ergebnis ergibt für S die Summe und für C (carry) den Übertrag der Addition.

Definition 4.9

$S = X \oplus Y$ und $C = X \cdot Y$

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Tab. 4.5: *Halbaddierer*

Halbaddierer [40]

4.3.3 Volladdierer

Ein **Volladdierer** addiert drei einstellige Binärzahlen X,Y und C_{in}. Das zweistellige Ergebnis ergibt für S (sum) die Summe und für C_{out} (carry) den Übertrag der Addition.

Definition 4.10

$S = X \oplus Y \oplus C_{in}$ und

$C_{out} = (C_{in} \cdot (X \oplus Y)) + (X \cdot Y)$

C_{in}	X	Y	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tab. 4.6: *Volladdierer*

Volladdierer [54]

4.3.4 Flipflop

Ein **Flipflop** (**Flip-Flop**) ist eine elektronische Schaltung mit zwei stabilen Zuständen (bistabile Kippstufe), die als **Ein-Bit-Speicher** verwendet werden kann.

Definition 4.11

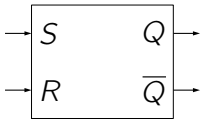
Flipflop [38]

[1] **englisch:** *XOR* = eXclusiveOR (exklusives Oder)

RS-Flipflop

Für das **RS-Flipflop** gilt: Bei $S = 1$ und $R = 0$ springt der Ausgang Q auf 1. Bei $S = 0$ und $R = 1$ spring Q auf 0. Bei $R = S = 0$ bleibt der zuvor gespeicherte Wert erhalten.

Definition 4.12



R	S	Q	$\overline{Q}$
0	0	Q^a	$\overline{Q}^a$
0	1	1	0
1	0	0	1
1	1	X^b	X^b

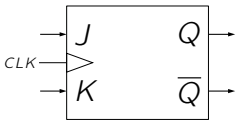
$Q = \overline{R + Q}$ (aus NOR-Gatter)

[a] Zustand unverändert
[b] Zustand verboten

Der Zustand $R = S = 1$ ist nicht erlaubt und auf jeden Fall zu vermeiden.

Beobachtung 4.4

JK-Flipflop



K	J	Q	$\overline{Q}$
0	0	Q^a	$\overline{Q}^a$
0	1	1	0
1	0	0	1
1	1	$\neg Q^b$	$\neg\overline{Q}^b$

$Q' = \left(J \cdot \overline{Q}\right) + \left(\overline{K} \cdot Q\right)$

[a] Zustand unverändert
[b] Zustand wechselt

Bei taktflankengesteuerter Realisierung kann der Eingang C für steigende oder fallende Flanken ausgelegt sein.

Der Zustand $J = K = 1$ ist erlaubt; in diesem Fall wechselt der Ausgangspegel mit jeder wirksamen Flanke des Taktsignals C.

Beobachtung 4.5

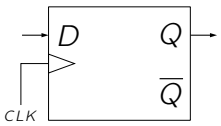
D-Flipflop

Das taktflankengesteuerte **D-Flipflop** besitzt einen Dateneingang D und einen dynamischen Eingang CLK. Das Flipflop übernimmt bei der aktiven Taktflanke den Zustand des Einganges D und gibt ihn an den Ausgang Q weiter.

Definition 4.13

Änderungen am Eingang D wirken sich erst aus wenn die aktive Taktflanke eintrifft. Es wird zwischen steigender Taktflanke ⎍ und fallender Taktflanke ⎏ unterschieden.

D-Flipflop mit Trigger auf steigende Taktflanke



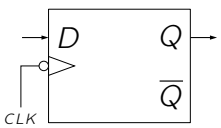
reagiert auf steigende Taktflanken

$Q' = D$

C	D	Q
⎍	0	0
⎍	1	1
0,1,⎏	X^a	Q^b

[a] beliebig (0 oder 1)
[b] Zustand unverändert

D-Flipflop mit Trigger auf fallende Taktflanke



reagiert auf fallende Taktflanken

$Q' = D$

C	D	Q
⎏	0	0
⎏	1	1
0,1,⎍	X^a	Q^b

[a] beliebig (0 oder 1)
[b] Zustand unverändert

D-Flipflop

4.4 Registerschaltungen

4.4.1 Synchronzähler

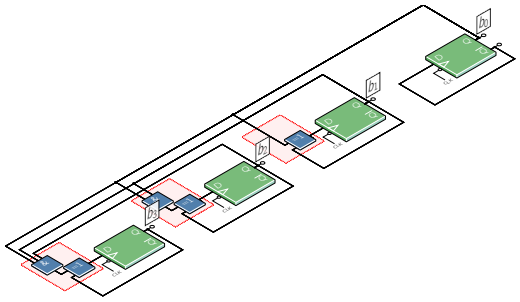


Abb. 4.2: *Vorwärtszähler aus D-Flipflops*

RS-Flipflop [38]

Ein **Synchronzähler** ist ein Bauelement welches eine Folge von Ereignissen **zählt** und den Wert bis zum nächsten Ereignis **speichert**.

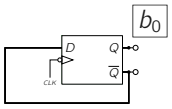
Definition 4.14

Synchroner Vorwärtszähler

Nachfolgend wird exemplarisch ein D-Flipflop für den synchronen Zähler verwendet.

Synchronzähler mit D-Flipflops

Rückkkopplung auf Flipflop Bit b_0



Nr	$D = \overline{b_0}$	b_0
0	1	0
1	0	1
⋮	⋮	0
⋮	⋮	1

Abb. 4.3: *Vorwärtszähler 1. Stelle b_0*

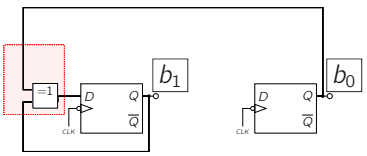
Wird beim D-Flipflop der inverse Ausgang $\overline{Q}$ auf den Eingang D rückgekoppelt, dann wechselt der Ausgang Q mit jeder Taktflanke seinen Zustand. Damit erhalten wir abwechselnd die Werte 0 und 1 und dies entspricht der Spalte b_0 in der Wertetabelle des Dualzählers (siehe Abb. 4.3).

Wir betrachten uns die Stellen b_1 bis b_3 und überlegen uns, wann sollen sie den Wert 1 annehmen?

- Stelle b_1: bei Zähler-Nr 2 und 3 (Abb. 4.4)
- Stelle b_2: bei Zähler-Nr 4, 5, 6 und 7 (Tab. 4.7)
- Stelle b_3: bei Zähler-Nr 8 bis 15 (Tab. 4.8)

Dabei muß der Eingang D für jedes Flipflop schon eine Zeile vorher auf 1 stehen, damit rechtzeitig mit der folgenden Taktflanke der Ausgang gesetzt wird.

Rückkkopplung auf Flipflop Bit b_1



Nr	D	b_1	b_0
0	0	0	0
1	1	0	1
2	1	1	0
3	0	1	1
		0	0
⋮	⋮	0	1
⋮	⋮	1	0
		1	1

Abb. 4.4: *Vorwärtszähler 2. Stelle b_1*

Dies wiederholt sich alle 4 Zeilen.

Rückkkopplung auf Flipflop Bit b_2

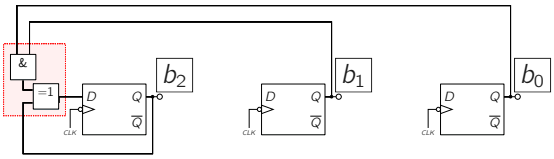


Abb. 4.5: *Vorwärtszähler 3. Stelle b_2*

	$b_0 \cdot b_1$	XOR	Flipflop		Vorgänger	
Nr	d_2	$d_2 \oplus b_2$	D	b_2	b_1	b_0
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	1	1	1	0	1	1
4	0	1	1	1	0	0
5	0	1	1	1	0	1
6	0	1	1	1	1	0
7	1	0	0	1	1	1

Tab. 4.7: *Wertetabelle inkl. Rückkopplung*

Dies wiederholt sich alle 8 Zeilen.

Rückkkopplung auf Flipflop Bit b_3

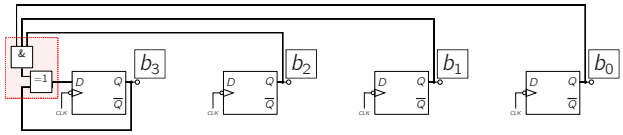


Abb. 4.6: *Vorwärtszähler 4. Stelle b_3*

	$b_0 \cdot b_1 \cdot b_2$	XOR	Flipflop		Vorgänger		
Nr	d_3	$d_3 \oplus b_3$	D	b_3	b_2	b_1	b_0
6	0	0	0	0	1	1	0
7	1	1	1	0	1	1	1
8	0	1	1	1	0	0	0
9	0	1	1	1	0	0	1
10	0	1	1	1	0	1	0
11	0	1	1	1	0	1	1
12	0	1	1	1	1	0	0
13	0	1	1	1	1	0	1
14	0	1	1	1	1	1	0
15	1	0	0	1	1	1	1

Tab. 4.8: *Wertetabelle inkl. Rückkopplung*

4.4.2 Register

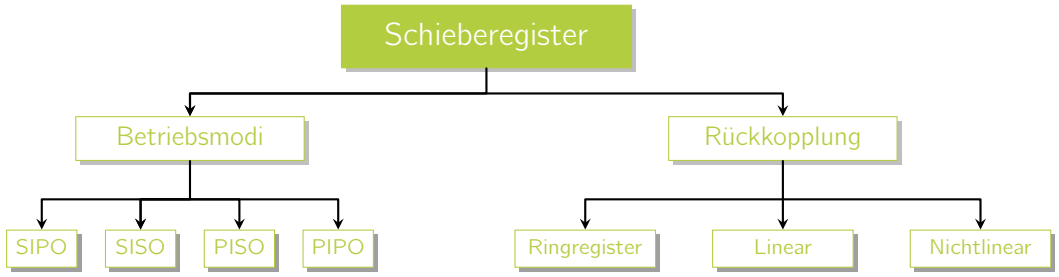


Abb. 4.7: *Einteilung von Schieberegister*

Schieberegister

Schieberegister sind in Serie geschaltete **1-Bit-Speicher** (Flipflops), die mehrstellige binäre Signale taktgesteuert aufnehmen, speichern und wieder abgeben können.

Definition 4.1

Schieberegisterarbeiten[1] nach dem **FIFO**[2] -Prinzip.

Je nach Anforderung bewegen sich dabei die Bits entweder nach links oder nach rechts.

Verschiebung nach rechts

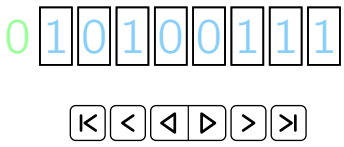


Abb. 4.8: *Schieberegister (Verschiebung nach rechts)*

Wenn ein Bit das Register auf der linken Seite betritt, bewegen sich alle Bits innerhalb des Schieberegisters um einen Platz nach rechts.

Das letzte (äußerste rechte) Bit in der Reihenfolge fällt aus dem Schieberegister (verschwindet).

Verschiebung nach links

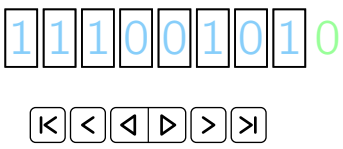


Abb. 4.9: *Schieberegister (Verschiebung nach links)*

Wenn ein Bit das Register auf der rechten Seite betritt, bewegen sich alle Bits innerhalb des Schieberegisters um einen Platz nach links.

Das letzte (äußerste linke) Bit in der Reihenfolge fällt aus dem Schieberegister (verschwindet).

Betriebsmodi

Man unterscheidet folgende grundlegende Bewegung der Bits durch ein Schieberegister:

- Serial Input Parallel Output (SIPO)
- Serial Input Serial Output (SISO)
- Parallel Input Serial Output (PISO)
- Parallel Input Parallel Output (PIPO)

Ringregister

Ringregister sind rückgekoppelte Schieberegister, d.h. das letzte Bit in der Reihenfolge fällt nicht aus dem Register sondern wird am Eingang wieder eingelesen.

Definition 4.2

Schiebe-register

Es ist die Sonderform eines Schieberegisters.

Das im Ringregister eingespeicherte Bitmuster bewegt sich zyklisch[3] durch die einzelnen Bit-Register.

Rotation nach rechts

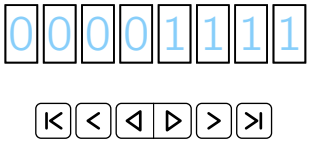


Abb. 4.10: *Ringregister (Rotation nach rechts)*

Beim Ringregister, bewegen sich alle Bits um einen Platz nach rechts. Das letzte (äußerste rechte) Bit wandert dabei wieder in das erste (äußerste linke) Bit.

Man sagt auch, die Bits bewegen sich zyklisch nach rechts bzw. rotieren nach rechts.

Rotation nach links

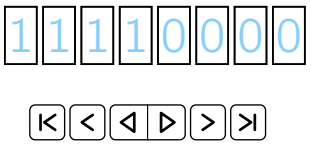


Abb. 4.11: *Ringregister (Rotation nach links)*

Beim Ringregister, bewegen sich alle Bits um einen Platz nach links. Das letzte (äußerste linke) Bit wandert dabei wieder in das erste (äußerste rechte) Bit.

Man sagt auch, die Bits bewegen sich zyklischnach links bzw. rotieren nach links.

[2] **englisch:** *FIFO* = First In First Out

[3] **lateinisch:** *cyclus* = Kreis, kreisförmig

Historische Entwicklung

5 Übersicht

Inhaltsangabe

5.1 Geschichte der Kryptografie

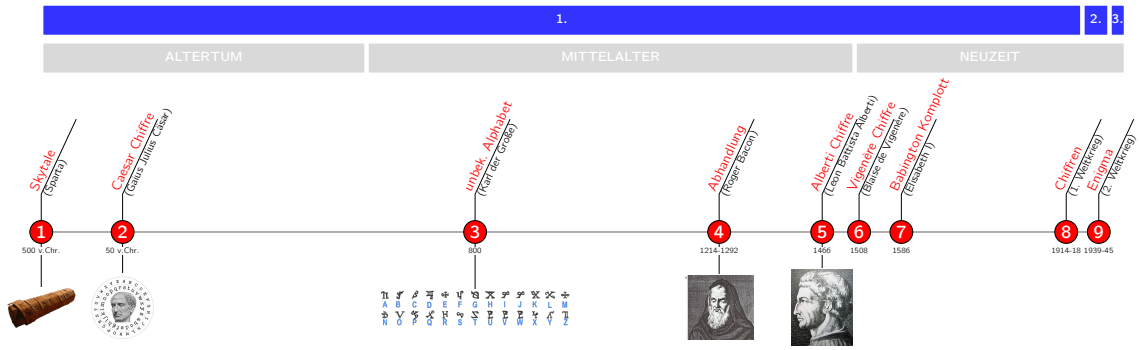


Abb. 5.1: *Meilensteine der Kryptologie*

Geschichte der Kryptografie

Die Geschichte der Kryptografie [a], lässt sich in drei Zeiträume aufteilen. Im ersten wurde per Hand („mit Papier und Bleistift“ oder mechanischen Scheiben) verschlüsselt, im zweiten (etwa 1920 bis 1970) mit spezielle Maschinen und im dritten (etwa seit 1970) mit Computer verschlüsselt.

[a] Verschlüsselung von Texten, Nachrichten oder Daten zum Zwecke der Geheimhaltung

5.1.1 Altertum

Skytale

Die **Skytale**[1] ist das älteste bekannte (militärische) Verschlüsselungsverfahren und wurde von den Spartanern vor etwa 2500 Jahren verwendet. *Definition 5.1*

Zur Verschlüsselung diente ein Stab mit einem definiertem Durchmesser (Skytale).

Um eine geheime Nachricht zu verfassen, wickelte der Absender einen Streifen (aus Pergament oder Leder) um die Skytale, schrieb die Botschaft längs auf das Band und wickelte es dann ab.

Fällt das Band in die falschen Hände, so kann die Nachricht nicht gelesen werden, da die Buchstaben scheinbar willkürlich auf dem Band angeordnet sind.

Caesar Chiffre

Skytale [50]

Als eines der einfachsten und unsichersten Verfahren dient es heute hauptsächlich dazu, Grundprinzipien der Kryptologie anschaulich darzustellen.

Der Einfachheit halber werden oftmals nur die 26 Buchstaben des lateinischen Alphabets ohne Unterscheidung von Groß- und Kleinbuchstaben als Alphabet für Klartext und Geheimtext verwendet und Sonderzeichen, Satzzeichen usw. nicht beachtet.

[1] **altgriechisch:** *skytale* = Stock

5.1.2 Mittelalter

Karl der Große

Karl der Große war von 768 bis 814 König des Fränkischen Reichs, dabei bis 771 gemeinsam mit seinem Bruder Karlmann.

Chiffre Karl des Großen

Karl der Große soll für seine Korrespondenz verschiedene Varianten einer Geheimschrift aus Symbolen benutzt haben.

Roger Bacon

Roger Bacon [48]

Der einzige europäische Gelehrte, von dem namentlich aus dieser Zeit eine Abhandlung über Kryptographie überliefert ist, war der englische Mönch und Universalgelehrte Roger Bacon (1214–1292 oder 1294).

5.1.3 Neuzeit

Chiffrierscheibe

Die bedeutendste Entwicklung aus dieser Zeit ist die Chiffrierscheibe, die 1466 von dem Italiener Leon Battista Alberti (1404–1472) beschrieben wurde.

Vigenère-Chiffre

Blaise de Vigenère [35]

Blaise de Vigenère (1523–1596) veröffentlichte die aus den durch den deutschen Benediktinerabt Johannes Trithemius (1462–1516) im Jahre 1508 im fünften Band seines in lateinischer Sprache geschriebenen sechsbändigen Werkes Polygraphiae libri sex (deutsch: Sechs Bücher zur Polygraphie) entnommene Tabula recta unter eigenem Namen.

Diese unter falschen Namen veröffentlichte Vigenère-Chiffre galt lange als unknackbar und wurde erst nach fast 300 Jahren von Charles Babbage systematisch entziffert.

Babington-Komplott

Babington-Komplott [34]

Die Babington-Verschwörung hat ihren Namen von Anthony Babington, der im Jahr 1586 gemeinsam mit einer Gruppe befreundeter Katholiken plante, die protestantische englische Königin Elisabeth I. zu ermorden und Maria Stuart aus dem Gefängnis zu befreien und sie auf den englischen Thron zu bringen.

Maria erhielt Briefe von ihren Anhängern, die mit einem Nomenklator verschlüsselt waren. Da die Briefe verschlüsselt waren, stellte der Sicherheitsminister von Elisabeth Francis Walsingham den erfahrenen Codeknacker Thomas Phelippes als Geheimsekretär ein, dem die Entzifferung der Nachrichten mit Hilfe der Häufigkeitsanalyse gelang.

Sezessionskrieg

Im amerikanischen Sezessionskrieg (1861–1865) wurde zwar bereits Telegrafie genutzt, doch die Bedeutung der Datenverschlüsselung und der Dechiffrierung wurden noch unterschätz.

Auf beiden Seiten des Konflikts gab es keine Koordination im Bereich der Kryptographie.

Obwohl beide Seiten nur geringen Aufwand in das Knacken der Codes der Gegenseite investierten, gelangen zahlreiche Entzifferungen.

Erster Weltkrieg

Der Erste Weltkrieg gilt als der erste Krieg, in dem die Möglichkeiten der Kryptoanalyse systematisch genutzt wurden. Der Aufwand, den die Kriegsparteien zur Entzifferung gegnerischer Funksprüche trieben, stieg im Verlauf des Kriegs deutlich an, nachdem einige Staaten zu Kriegsbeginn noch gar keine Entzifferungseinheiten betrieben hatten.

Die Entwicklung neuer Verschlüsselungsverfahren konnte mit dieser Entwicklung nicht Schritt halten, weshalb nahezu alle im Ersten Weltkrieg verwendeten Methoden mit vergleichsweise wenig Aufwand geknackt wurden.

One-Time-Pad

One-Time-Pad [47]

In die Zeit der ersten Maschinenentwicklungen fällt auch die Erfindung des One-Time-Pad (deutsch Einmalblock). Bei diesem Verfahren wird der Text zeichenweise mit einer zufälligen Zeichenfolge verschlüsselt, die nur einmal verwendet wird. Wenn es sich wirklich um eine Zufallsfolge handelt, ist jedes Verschlüsselungsergebnis gleich wahrscheinlich. Dann ist das Verfahren perfekt sicher.

Leon Battista Alberti [46]

Der Amerikaner Joseph O. Mauborgne (1881–1971) setzte diese Idee um und prägte den Begriff One-Time Pad (OTP).

Zweiter Weltkrieg

Die verheerenden Erfahrungen im Ersten Weltkrieg führten dazu, dass noch während des Kriegs sowie in den Jahren danach erste Maschinen zur Verschlüsselung entwickelt wurden. Diese boten eine deutlich höhere Sicherheit als die bis dahin üblichen manuellen Methoden.

Die zahlreichen Verschlüsselungsmaschinen, die nun eine neue Ära in der Kryptographie-Geschichte einläuteten, sollten jedoch nicht darüber hinwegtäuschen, dass (vor allem aus Kostengründen) vorläufig noch zahlreiche manuelle Verfahren eingesetzt wurden – wenn auch meist nur für weniger wichtige Zwecke.

Enigma

Eine erste Probemaschine wurde am 23. Februar 1918 von Arthur Scherbius zum Patent angemeldet. Die allgemeine militärische Aufrüstung ab 1933 trug zur intensiven militärischen Nutzung der Enigma I bei.

Enigma [36]

Sie kam schließlich im Zweiten Weltkrieg zu Zehntausenden zum Einsatz und galt auf deutscher Seite irrtümlicherweise als „unbrechbar“.

Am 15. Juli 1928 wurde zum ersten Mal einen mit einer Enigma chiffrierten deutschen Funkspruch abgefangen. Zur Jahreswende 1932/33 gelang dem Biuro Szyfrów (polnischer Chiffrierdienst) die ersten Entzifferungen.

Britischen Codebreakers um Alan Turing gelang es während des Zweiten Weltkriegs äußerst erfolgreich, die abgefangenen deutschen Funksprüche zu entziffern.

Dazu nutzten sie eine spezielle elektromechanische „Knack-Maschine“, genannt die Turing-Bombe.

Ab Januar 1940 wurde der deutsche Enigma-Funkverkehr mit nur wenigen Ausnahmen kontinuierlich „mitgelesen“

5.2 Verschlüsselungsklassen

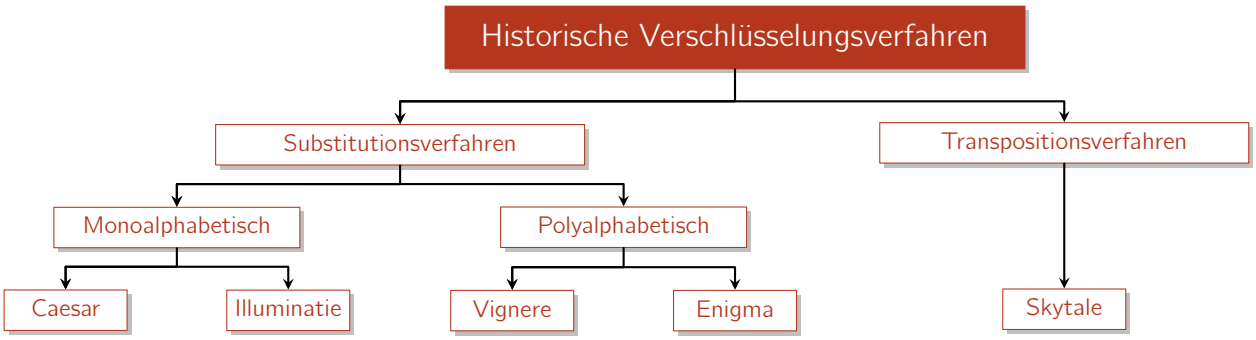


Abb. 5.2: *Einteilung von Historischen Verschlüsselungsverfahren*

Bei historischen Verfahren lassen sich zwei verschiedene Verschlüsselungsklassen unterscheiden:

5.2.1 Substitution

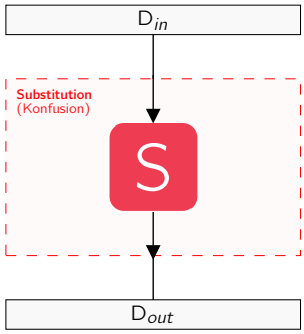


Abb. 5.3: *Substitution (Konfusion)*

Definition

Bei der **Substitution**[2] (**Konfusion**) werden Buchstaben (Zeichen) durch andere Zeichen, gennant Geheimtextzeichen, **ersetzt**.

Definition 5.1

Durch diesen Vorgang sind die Wörter, zumindest auf den ersten Blick, nicht mehr zu erkennen. Claude Shannon bezeichnete dies mit dem Wort **Konfusion**.

Bei der **Konfusion** soll zwischen Klar- und Chriffretext möglichst keine Beziehung erkennbar sein, die für einen Angriff ausgenutzt werden kann[3]

Definition 5.2

Implementierung

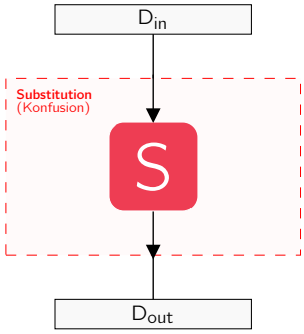


Abb. 5.4: *Implementierung durch Tabellen*

5.2.2 Transposition

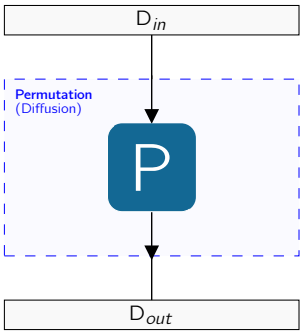


Abb. 5.5: *Transposition*

Definition

Bei der Transposition bleiben die Zeichen einer Botschaft unverändert erhalten, nur die Position, d.h. die Stellen an der sie stehen, werden verändert.

Bei der **Transposition**[4] (**Diffusion**, **Permutation**) werden Buchstaben (Zeichen) des Klartextes umsortiert, **versetzt**.

Definition 5.3

Bei der **Diffusion** sollen alle Zeichen des Klartextes und des Schlüssels möglichst viele Zeichen des Chiffretextes beinflussen.

Definition 5.4

Substitution [51]

Transposition [52]

Implementierung

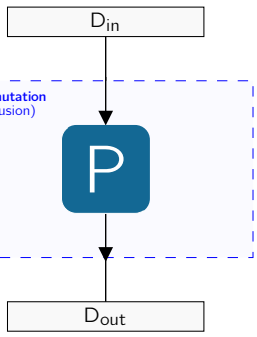


Abb. 5.6: *Implementierunt durch Verdrahtung*

[2] **lateinisch:** *substituere* = ersetzen
[3] insbesondere die statistische Verteilung der Zeichen

[4] **lateinisch:** *transponere* = versetzen

6 Monoalphabetische Chiffren

Inhaltsangabe

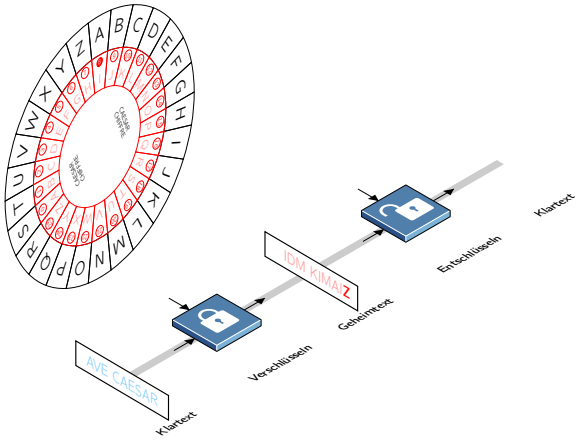


Abb. 6.1: *Caesar Chiffre*

Beim **monoalphabetischen**[1] Verfahren werden zur Verschlüsselung nur ein einziges (festes) Schlüsselalphabet verwendet.

Definition 6.1

6.1 Substitutions-Chiffre

Die Caesar-Verschlüsselung wird oft verwendet, die Grundlagen der Kryptografie anschaulich darzustellen.

Dabei werden einfachhalber nur die 26 Buchstaben des Alphabets (ohne Groß- und Kleinschreibung) verwendet. Sonderzeichen, Satzzeichen etc. werden ignoriert.

6.1.1 Caesar Chiffre

Bei der **Caesar-Verschlüsselung** wird jeder Buchstabe des Klartexts eines Alphabets um eine bestimmte Anzahl zyklisch nach rechts verschoben.

Definition 6.2

Der **Schlüssel** ist dabei die Anzahl der verschobenen Zeichen und bleibt für die Verschlüsselung unverändert.

Heute ist die Caesar-Verschlüsselung als **ROT13**[2] in Gebrauch, um Textinhalte wie Spoiler oder Pointen gegen unabsichtliches Lesen zu verschleiern.

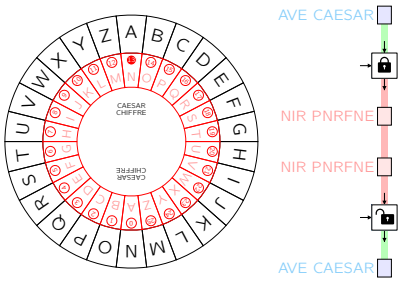


Abb. 6.2: *ROT13 Chiffre*

Bei der **ROT13** Verschlüsselung funktioniert das Verschlüsseln genauso wie das Entschlüsseln, d.h. mit dem Schlüssel 13 kann sowohl ver- als auch entschlüsselt werden. Diese Eigenschaft nennt man involutorisch.

[1] **griechisch:** *mono* = einzig

[2] ROT13 . . . Verschiebung um 13 Zeichen

7 Moderne Kryptografie

Inhaltsangabe

7.1 Übersicht

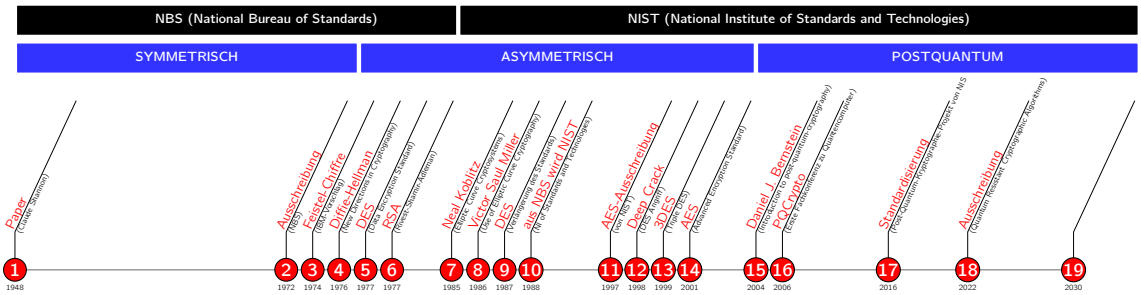


Abb. 7.1: *Meilensteine der modernen Kryptografie*

In den siebziger Jahren wandelte sich die Kryptographie von einer reinen Geheimwissenschaft zu einer Forschungsdisziplin, die auch öffentlich betrieben wurde. Dies ist vor allem darauf zurückzuführen, dass mit Aufkommen des Computers eine immer größere Nachfrage nach Datenverschlüsselung entstand.

7.2 Symmetrische Kryptografie

7.2.1 Data Encryption Standard

- **1972** (2): Ausschreibung für ein Verschlüsselungsverfahren
- **1974** (3): Vorschlag von IBM (Feistel-Chiffre)
- **1974-77**: Verifizierung/Verbesserung durch NSA
- **1977** (5): Veröffentlichung als DES

7.2.2 Advanced Encryption Standard

- **1997** (11): Ausschreibung von NIST für einen Nachfolger von DES
  Diese erfolgte öffentlich und alle Kryptologen aus der ganzen Welt konnten teilnehmen.
- **1998/1999**: Zwei Konferenzen/fünf Finalisten. (MARS, RC6, Rijndael, Serpent, Twofish).
- **2001** (14): Auswahl von Rijndael als Standard AES

7.3 Asymmetrische Kryptografie

Der Aufsatz von Diffie und Hellman stellte eine radikal neue Methode der Schlüsselverteilung vor und gab den Anstoß zur Entwicklung von Public-Key-Verfahren.

- **1976** [4]: „New Directions in Cryptography“ Artikel von Whitfield Diffie und Martin Hellman (Diffie-Hellman-Schlüsselaustausch)
- **1977** [6]: RSA-Kryptosystem (Ronald L. Rivest, Adi Shamir, Leonard Adleman)
- **1997**: Veröffentlichung vom britischen GCHQ (Government Communications Headquarters)

Der Schlüsselaustausch ist eines der fundamentalen Probleme der Kryptographie.

Vor dieser Entdeckung waren die Schlüssel symmetrisch, und der Besitz eines Schlüssels erlaubte sowohl das Verschlüsseln als auch das Entschlüsseln einer Nachricht.

Daher musste der Schlüssel zwischen den Kommunikationspartnern über einen sicheren Weg ausgetauscht werden, wie beispielsweise durch einen vertrauenswürdigen Kurier oder beim direkten Treffen der Kommunikationspartner.

Diese Situation wurde schnell unüberschaubar, wenn die Anzahl der beteiligten Personen anstieg.

Auch wurde ein jeweils neuer Schlüssel für jeden Kommunikationspartner benötigt.

Ein solches Verfahren wird als symmetrisch oder auch als „Secret-Key“-, „Shared-Secret“- oder „Private-Key“-Verfahren bezeichnet.

Bei der Public Key Cryptography wird ein Paar zusammenpassender Schlüssel eingesetzt.

Der eine ist ein öffentlicher Schlüssel, der – im Falle eines Verschlüsselungsverfahrens – zum Verschlüsseln von Nachrichten für den Schlüsselinhaber benutzt wird.

Der andere ist ein privater Schlüssel, der vom Schlüsselinhaber geheim gehalten werden muss und zur Entschlüsselung eingesetzt wird.

Ein solches System wird als asymmetrisch bezeichnet, da für Ver- und Entschlüsselung unterschiedliche Schlüssel verwendet werden.

Mit dieser Methode wird nur ein einziges Schlüsselpaar für jeden Teilnehmer benötigt, da der Besitz des öffentlichen Schlüssels die Sicherheit des privaten Schlüssels nicht aufs Spiel setzt.

7.3.1 RSA

Eines der bekanntesten Public Key Verfahren war **1977** das RSA-Kryptosystem, kurz RSA (von Ronald L. Rivest, Adi Shamir, Leonard Adleman).

Public-Key-Kryptographie wurde unter Geheimhaltung bereits vom Militär entwickelt, bevor die öffentliche Forschung dies erreichte.

Am 17. Dezember **1997** veröffentlichte das britische GCHQ (Government Communications Headquarters in Cheltenham) ein Dokument, in welchem sie angaben, dass sie bereits vor der Veröffentlichung des Artikels von Diffie und Hellman ein Public-Key-Verfahren gefunden hätten.

Verschiedene als geheim eingestufte Dokumente wurden in den 1960ern und 1970ern unter anderem von James H. Ellis, Clifford Cocks und Malcolm Williamson geschrieben, die zu Entwürfen ähnlich denen von RSA und Diffie-Hellman führten.

Die Sicherheit der faktorisierungsbasierten Public-Key-Kryptographie liegt in der Verwendung eines Produkts aus großen Primzahlen, welches als öffentlicher Schlüssel dient.

Der private Schlüssel besteht aus den dazugehörenden Primfaktoren bzw. davon abgeleiteten Werten.

Die Zerlegung eines hinreichend großen öffentlichen Schlüssels gilt aufgrund der mathematisch sehr aufwendigen Faktorisierung als nicht praktikabel.

7.3.2 Elliptische Kurven

Insbesondere nach der Einführung von Elliptic Curve Cryptography in den 1980er Jahren wurden fortgeschrittene zahlentheoretische Methoden in der Kryptographie angewandt.

7.4 Post-Quantum-Kryptografie

Quantenkryptographie ist ein kryptographisches Verfahren, das quantenmechanische Effekte bei Quantenkommunikation oder Quantencomputern verwendet.

- **2004** [15]: „Post-Quanten-Kryptographie“ Einführung des Begriffes von Daniel J. Bernstein
- **2006** [16]: Erste Fachkonferenz (PQCrypto)
- **2016** [17]: Post-Quantum-Kryptographie-Projekt NIST-Standardisierungsprozess für Verschlüsselungsverfahren
- **2022** [18]: Ausschreibung von NIST für „First Four Quantum-Resistant Cryptography“

Die bekanntesten Beispiele der Quantenkryptographie sind der Quantenschlüsselaustausch und der (noch nicht praktikable) Shor-Algorithmus zum Faktorisieren großer Zahlen.

Quantenkryptographie erlaubt das Entwickeln von Verfahren, die klassisch (d. h. ohne den Einsatz von Quanteneffekten) unmöglich sind.

Zum Beispiel kann bei einem Quantenkanal ein Lauscher entdeckt werden, weil seine Messung die gesendeten Daten beeinflusst.

Symmetrische Kryptografie

8 Designkriterien

Inhaltsangabe

8.1 Grundsätze der Kryptografie

Die erste wichtige Erkenntnis ist hierbei, dass die Stärke der Verschlüsselung nicht davon abhängt, wie geheim der Algorithmus gehalten wird. Im Gegenteil, einige der stärksten bekannten Verschlüsselungsalgorithmen sind vollständig veröffentlicht und für jedermann zugänglich.

Geheimhaltung an der richtigen Stelle Im Englischen heißt es: „There is no security by obscurity“. Das bedeutet in der Praxis, eine Verschlüsselung, die sich allein auf Unkenntlichkeit verlässt, ist wenig wirksam. Wenn Sie zum Beispiel an Ihrer Haustür einen Türöffner anbringen, der von außen nicht zu erkennen ist, dann ist Ihr Haus zunächst sicher. Sobald aber jemand den Zugangsmechanismus entdeckt, beispielsweise weil er Sie beobachtet, wie Sie ihn benutzen, kommt er genauso schnell ins Haus wie Sie.

Im Gegensatz dazu ist ein normales Türschloss von außen eindeutig erkennbar, es lässt sich aber ohne Aufwand nur mit dem passenden Schlüssel öffnen. Die Sicherheit hängt also davon ab, wie gut Sie Ihren Schlüssel verwahren, und davon, wie aufwändig das Öffnen ohne passenden Schlüssel ist.

8.1.1 Auguste Kerckhoff

Kerckhoff beschrieb 1883 in „La cryptographie militaire“[1] sechs Grundsätze zur Konstruktion eines sicheren Verschlüsselungsverfahrens:

Ein **Verschlüsselungssystem**

1. . . . muss im Wesentlichen unentzifferbar sein.
2. . . . darf keine Geheimhaltung erfordern.
3. . . . muss leicht übermittelbar sein und man muss sich die Schlüssel ohne schriftliche Aufzeichnung merken können.
4. . . . sollte mit telegraphischer Kommunikation kompatibel sein.
5. . . . muss transportabel sein und die Bedienung darf nicht mehr als eine Person erfordern.
6. . . . muss einfach anwendbar sein.

Das Kerckhoffs'sche Prinzip ist der zweite der sechs Grundsätze zur Konstruktion eines sicheren Verschlüsselungsverfahrens, die Kerckhoffs 1883 in einführt:

[1] **französisch:** *La cryptographie militaire* = Militärische Kryptografie

8.1.2 Claude Shannon

Für eine starke Verschlüsselung ist es, nach dem Begründer der modernen Informationstheorie Claude Shannon, zufolge notwendig zwei grundlegende Operationen zu realisieren:

- Konfusion (Substitution)
- Diffusion (Permutation)

8.2 Elementare Operationen

Zu den Designkriterien von Blockchiffen gehören auch Anforderungen an die prakische Einsatzfähigkeit, d.h. es muss möglich sein, Blockchiffren effizient zu programmieren. So ist es ineffizient für die Substitation oder Permutation eine Wertetabelle zu verwenden.

Anzahl Tabellen			Eingangsbreite			Ausgangsbreite			Speicherplatz		
	Bit	Byte	KByte	MByte	GByte	KByte	MByte	GByte	pro Tabelle	alle Tabellen	Gesamt
1×	2^{64}	2^{56}	2^{46}	2^{36}	2^{26}	2^{46}	2^{36}	2^{26}	$2 \cdot 2^{26}$ GByte	1×2^{26} GByte	$\approx 10^9$ GByte
2×	2^{32}	2^{24}	2^{14}	2^4	-	2^{14}	2^4	-	$2 \cdot 2^4$ MByte	2×2^5 MByte	≈ 64 MByte
4×	2^{16}	2^8	-	-	-	-	-	-	$2 \cdot 2^8$ Byte	4×2^9 Byte	≈ 2 kByte
8×	2^8	1	-	-	-	-	-	-	$2 \cdot 1$ Byte	8×2 Byte	≈ 16 Byte

Tab. 8.1: *Speicherplatz einer Tabelle abhängig von der Datenbreite*

8.2.1 Speicherplatz für Wertetabellen

Für eine Verschlüsselungsfunktion mit einer Blocklänge von 64 Bit müsste man den 2^{64} Möglichkeiten für den Klartext genausoviele für den Geheimtext bereitstellen. Insgesamt werden dabei 10^9 GByte benötigt.

Blockchiffren können also nicht durch Wertetabellen für die gesamte Datenbreite beschrieben werden, es muß eine kompaktere Implementierung verwendet werden.

8.2.2 Substitution (Konfusion)

Konfusion verschleiert die Beziehung zwischen Geheimtext und Schlüssel.

Definition 8.1

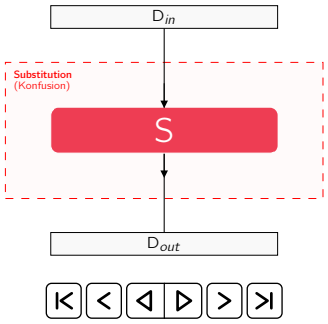


Abb. 8.1: *Substitution (S-Box)*

Sie wird oft in Form von einer Substitutionstabelle (**S-Box**) implementiert.

Dabei wird nicht eine S-Box für die gesamte Datenbreite verwendet, sondern die Datenblöcke auf mehrere S-Boxen aufgeteilt.

8.2.3 Permutation (Diffusion)

Diffusion dient dazu um statistische Eigenschaften des Klartextes zu verbergen.

Definition 8.2

Primitives [8]

Auch hier wird für die Permutation aus Effizienzgründen keine Tabelle verwendet, sondern die Eingangsdaten direkt mit dem Ausgang verdrahtet. [8]

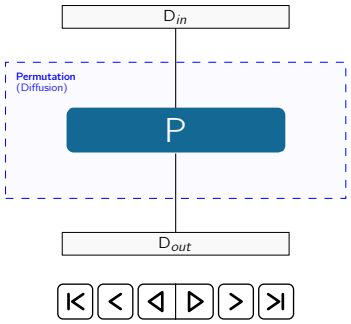


Abb. 8.2: *Permutation als Verdrahtung*

Historische Verschlüsselungsverfahren die **nur Konfusion** (z.B. bei Enigma, Schiebechiffren) oder **nur Diffusion** verwendeten, sind nicht sicher.

8.2.4 Kompression und Expansion

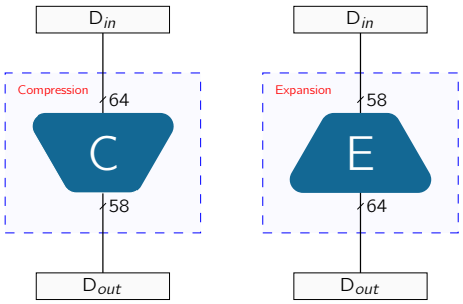


Abb. 8.3: *Kompression und Expansion*

8.3 Kombinierte Schaltungen

In verschiedenen Anwendungen wird die Permutation auch dann angwendet, wenn Ein- und Ausgänge verschiedene Busbreiten aufweisen.

Dann ist es notwendig am Ausgang die Anzahl der Bits zu verringern oder zu erhöhen (Kompression oder Expansion).

8.3.1 Erweiternde Permutation

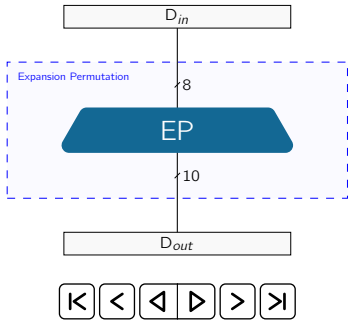


Abb. 8.4: *Expansion Permutation*

Die Erweiternde Permutation wird z. B. in DES bei der Implementierung der f-Funktion angewendet, um die Eingangsbusbreite von 32-Bits auf die Schlüsselbreite des Unterschlüssels von 48-Bits anzupassen.

8.3.2 Vermindernde Permutation

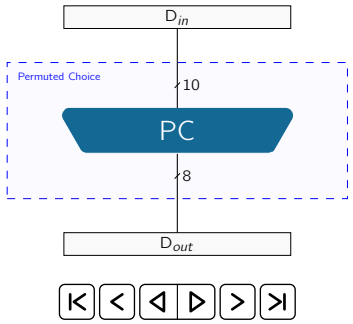


Abb. 8.5: *Permutated Choice*

Die Vermindernde Permutation wird z. B. in DES im Schlüsselpfad angewendet um die Eingangsschlüsselbreite von 64 Bits auf 56 Bits zu reduzieren.

Dabei wird jedes achte Bit des Eingangs, d.h. die Paritätsbits, nicht an den Ausgang weitergeleitet.

der Implementierung der f-Funktion angewendet, um die Eingangsbusbreite von 32-Bits auf die Schlüsselbreite des Unterschlüssels von 48-Bits anzupassen.

8.4 Produktchiffren

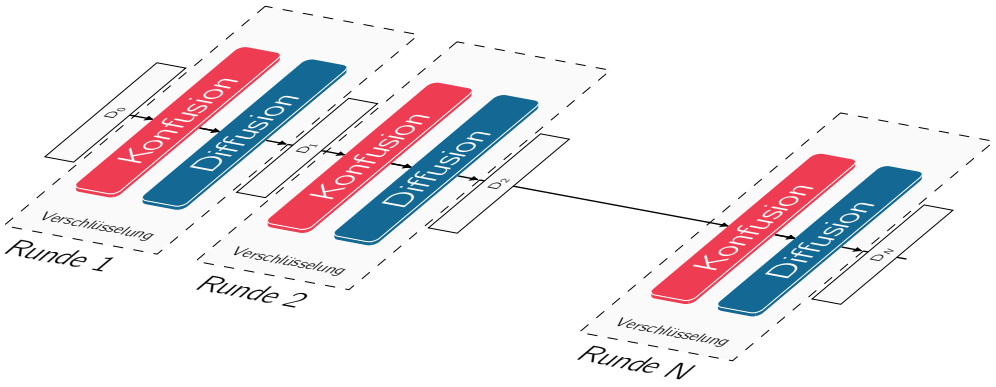


Abb. 8.6: *Prinzip einer Produkt-Chiffre*

Ein wichtiger Aspekt bei der Implementierung ist der praktische Einsatz, d.h. eine effiziente Programmierung.

Shannon schlug deshalb vor, für starke Chiffen ein Hintereinanderschalten von Diffusion und Konfusion zu verwenden.

Bei **Produktchiffren** wird der Verschlüsselungsalgorithmus durch das Hintereinanderschalten von Konfusion und Diffusion realisiert.

Definition 8.1

8.4.1 Standard-Chiffre

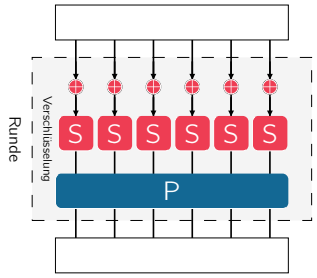


Abb. 8.7: *Standard SPN Chiffre (Verschlüsselung)*

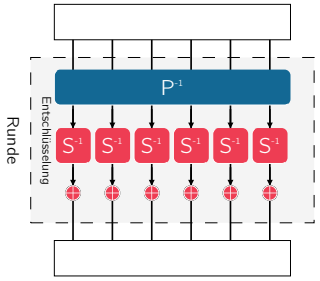


Abb. 8.8: *Standard SPN Chiffre (Entschlüsselung)*

8.4.2 Feistel-Chiffre

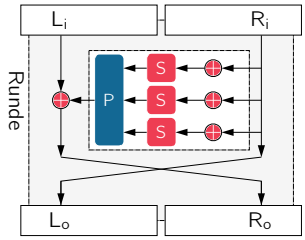


Abb. 8.9: *Feistel Chiffre*

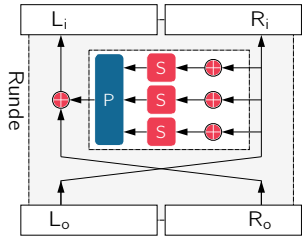


Abb. 8.10: *Symmetrie des Feistel Chiffres*

9 Stromchiffren

Inhaltsangabe

9.1 Übersicht

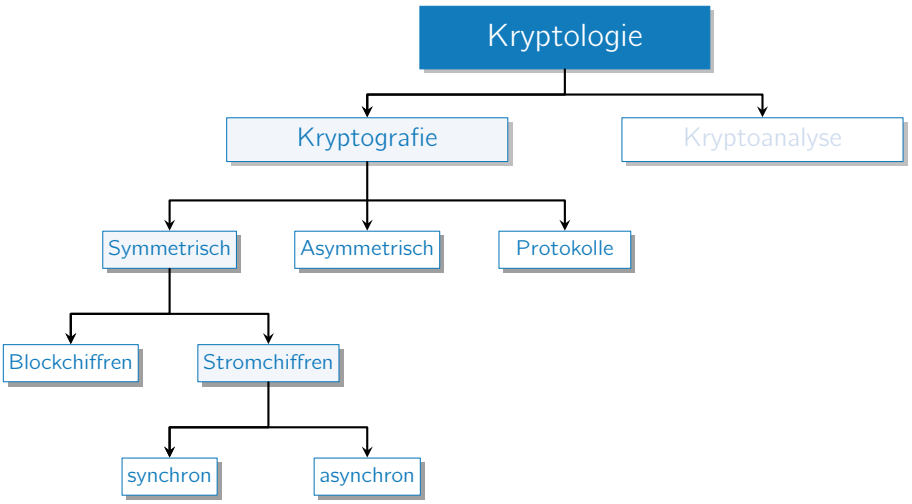


Abb. 9.1: *Überblick über die Kryptografie*

In der symmetrischen Kryptografie unterscheidet man zwischen Block- und Stromchiffren. Die Eingabe zu den Chiffren beträgt in beiden Fällen b Bit, wobei b auch die Eingangsbreite der Blockchiffre ist.

9.1.1 Blockchiffre

Prinzip

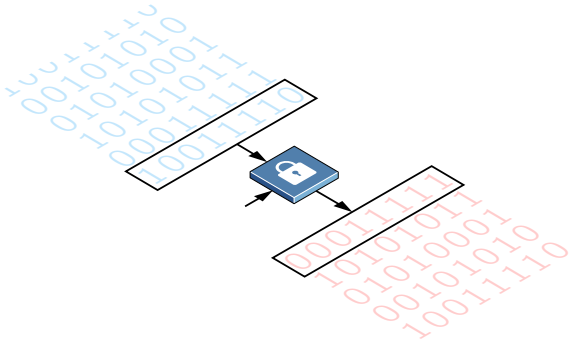


Abb. 9.2: *Blockchiffre: Verschlüsselung von b Bit*

Ein **Blockchiffre** verschlüsselt einen Block aus b Bit gleichzeitig mit dem gleichen Schlüssel. Dabei beeinflusst jedes Bit die Verschlüsselung jedes anderen Bits in dem Block.

Definition 9.1

Die allermeisten Blockchiffren haben entweder eine Blockbreite von 128 Bit oder 64 Bit[1].

9.1.2 Stromchiffre

Prinzip

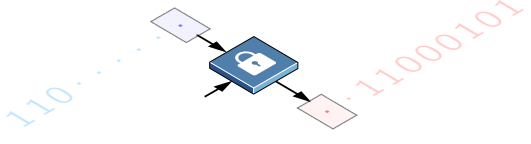


Abb. 9.3: *Stromchiffre: Verschlüsselung von b Bit*

Ein **Stromchiffre** verschlüsselt jedes **Bit** vom Klartext einzeln. Dabei wird zu jedem Bit vom Klartext ein **Bit** des **Schlüsselstroms** addiert.

Definition 9.2

Stromchiffren [15]

In der Praxis werden mehr synchrone als asynchrone Stromchiffren eingesetzt.

Ver- und Entschlüsselung

Bei der **Ver-** und **Entschlüsselung** mit **Stromchiffren** bestehen der Klartext x_i und der Schlüsselstrom s_i aus individuellen Bits $x_i, y_i, s_i \in \{0,1\}$.

Definition 9.3

Formale Schreibweise:

Verschlüsselung: $y_i = e_{s_i}(x_i) \equiv x_i + s_i \bmod 2$
Entschlüsselung: $x_i = d_{s_i}(x_i) \equiv y_i + s_i \bmod 2$

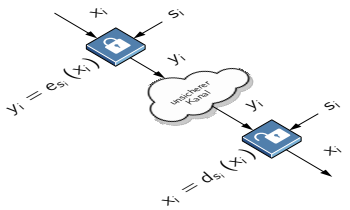


Abb. 9.4: *Ver- und Entschlüsselung mit einer Stromchiffre*

Da sowohl die Ver- als auch die Entschlüsselung „Additionen modulo 2“ sind, kann die grundsätzliche Funktionsweise auch durch ein XOR dargestellt werden[2].

Erzeugung des Schlüsselstroms

Die Generierung des Schlüsselstroms, der aus den Bits s_i besteht, bildet die zentrale Fragestellung bei Stromchiffren.

Die Sicherheit der Chiffre hängt vollständig von dem Schlüsselstrom ab und ob dieser für den Angreifer wie eine zufällig gewählte Bitfolge erscheint.

[1] z. B. AES mit 16 Byte oder DES, 3DES mit 8 Byte

[2] ein Kreis mit dem Plus-Symbol

9.2 Zufallszahlen

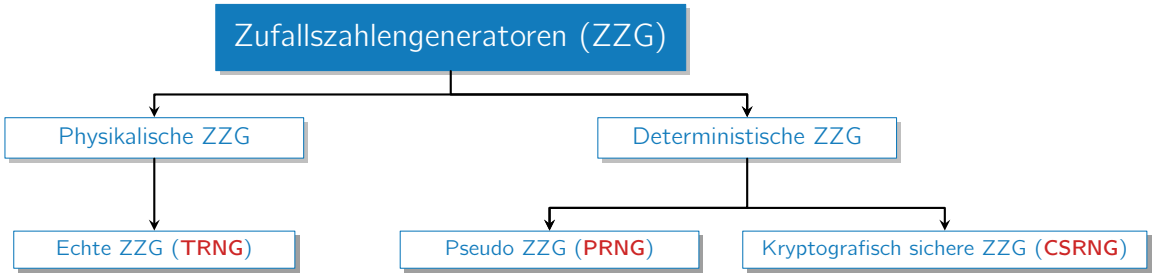


Abb. 9.5: *Einteilung von Zufallszahlengeneratoren [6, S 13]*

9.2.1 Zufallszahlengeneratoren

Die Sicherheit von Stromchiffren ist von der Qualität des Schlüsselstroms s_i, d.h. wie „zufällig" dieser gebildet werden kann, abhängig.

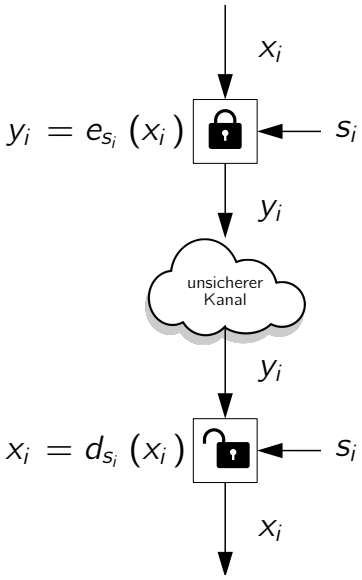


Abb. 9.6: *Schlüsselstrom s_i*

RNG [18]

Dabei stehen drei Arten von Zufallszahlengeneratoren (ZZG, **RNG**[3]) zur Auswahl.

Echte Zufallszahlengeneratoren

Echte Zufallszahlengeneratoren (**TRNG**[4]) produzieren Zahlen, die nicht reproduziert werden können und basieren auf physikalischen Prozessen[5]

Definition 9.1

In der Kryptografie werden Zufallszahlen für z.B. für die Erzeugung von Sitzungsschlüsseln benötigt.

Pseudozufallszahlengeneratoren

Ein **Pseudozufallszahlengenerator** (**PRNG**[6]) benutzt zur Erzeugung einer Sequenz von Werten ein Startwert (**Seed**). Jeder erzeugte Wert wird für die Berechnung des nächsten verwendet.

Definition 9.2

PRNG sind vollkommen deterministisch, d.h. ein identer Startwert erzeugt die gleiche Reihenfolge von „Zufallszahlen".

[3] **englisch:** *RNG* = Random Number Generators
[4] **englisch:** *TRNG* = True Random Number Generators
[5] z.B. Münzwurf, Würfeln, thermisches Rauschen von Halbleitern, radiaaktiver Zerfall etc.
[6] **englisch:** *PRNG* = Pseudo Random Number Generators

Kryptografisch sichere Generatoren

Kryptogtafisch sicheren Pseudozufallsgeneratoren (**CSPRNG**[7]) sind (rechentechnisch) nicht vorhersagbar.

Definition 9.3

Nicht vorhersagbar bedeutet es ist rechentechnisch unmöglich aus einem bekannten Teil des Schlüsselstroms nachfolgende oder vorherige Zahlen zu berechnen.

Gerade die **Nichtvorhersagbarkeit** von Zahlenfolgen ist in der Kryptografie eine unbedingt notwendige Voraussetzung.

Beobachtung 9.1

9.2.2 Das One-Time-Pad

Das **One-Time-Pad** (**OTP**[8]) ist ein symmetrisches Verschlüsselungsverfahren zur geheimen Kommunikation.

Definition 9.4

One-Time-Pad

Grundlegende Voraussetzungen für die Sicherheit des One-Time-Pad sind:

OTP [18]

Der Einmalschlüssel muss

- mindestens so lang sein wie die Nachricht,
- gleichverteilt zufällig gewählt werden,
- geheim bleiben und
- darf nicht wiederverwendet werden.

Ein Kryptoverfahren ist **Informationstheoretisch** oder **beweisbar sicher**, wenn es auch dann nicht gebrochen werden kann, wenn dem Angreifer **beliebige Rechenleistung** zur Verfügung steht.

Definition 9.5

9.3 Schieberegistern

Eine elegante Art, lange pseudozufällige Sequenzen für den Schlüsselstrom von Stromchiffren zu erzeugen, sind Linear rückgekoppelte Schieberegister.

Diese können in Hardware sehr einfach realisiert werden und können sehr lange (kryptografisch schwache) Pseudozufallssequenzen erzeugen.

Werden sie aber durch **nicht lineare** Komponenten miteinander kombiniert können sie für Stromchiffren krypografisch starke Pseudozufallssequenzen erzeugen.

[7] **englisch:** *CSPRNG* = Cryptographically secure PRNG
[8] **englisch:** OTP = One-Time-Pad (Einmalverschlüsselung oder Einmalschlüssel-Verfahren, wörtlich Einmal-Block)

Definition

Ein **Linear rückgekoppeltes Schieberegister** (**LFSR**[9]) ist ein durch die lineare logische XOR-Funktion rückgekoppeltes Schieberegister, das zur Erzeugung von streng deterministischen Pseudozufallszahlenfolge eingesetzt werden kann.

Definition 9.6

Der Startwert wird als Seed bezeichnet und bestimmt welche Reihenfolge die Zustände auftreten.

Mit dem Startwert wird der Schlüsselstrom am Ausgang des LFSR eindeutig bestimmt.

[9] **englisch:** LFSR = Linear Feedback Shift Register

9.3.1 Einleitendes Beispiel

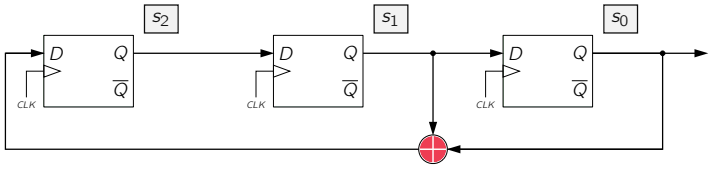


Abb. 9.7: *Beispiel eines LFSR mit 3 Flipflops*

Einleitend betrachten wir ein einfaches LFSR mit drei Flipflops FF_0, FF_1 und FF_2 und den in Abb. 9.7 dargestellten Rückkopplungspfad. Die Schaltung entspricht einem Schieberegister, d.h. bei jeder steigenden Taktflanke (Taktzyklus) werden die Bits um eine Stelle nach rechts verschoben.

Funktionsweise

Startwert 000
Wir betrachten das Verhalten des LSFR mit dem Startwert von 000

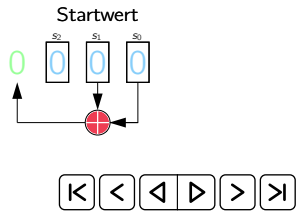


Abb. 9.8: *LFSR mit dem Startwert* 000

Haben alle Register den Wert 0, dann ändert sich durch die Rückkopplung der Wert nicht, d.h. am Eingang wird das Ergebnis 0 der XOR-Verknüpfung rückgekoppelt.

> Bei einem Startwert (seed) von 000 liefert das LFSR als Ergebnis einen Bitstrom der nur aus den Werten 0 besteht.
>
> *Beobachtung 9.1*

Deshalb ist ein Startwert dem Wert 000 für die Funktion nicht geeignet. Auch kann diese Bitkombination in keiner Sequenz länger als 1 auftreten.

Beliebiger Startwert
Wir betrachten das Verhalten des LSFR mit einem beliebigen Startwert ungleich 000 wie z. B. 100:

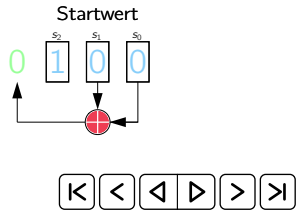


Abb. 9.9: *LFSR mit dem Startwert* 100

Für das erste, ganz links stehende Flipflop berechnet sich das Eingangsbit aus der XOR-Summe des Inhalts von FF_1 und FF_0.

> Bei einem LFSR mit 3 Flipflops (Speicherstellen) entsprechen die ersten 3 Bits s_0, s_1 und s_2 des Schlüsselstroms genau dem Startwert.
>
> *Beobachtung 9.2*

i	b_2	b_1	b_0	s_i
0	1	0	0	0
1	0	1	0	0
2	1	0	1	1
3	1	1	0	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	1	0	0	0

Tab. 9.1: *Schlüsselstrom des LFSR*

Wie wir der Tabelle 9.1 entnehmen können wiederholt sich der Startwert in der 8. Zeile (s_7) und es ergeben sich damit folgende periodische Ausgangssequenz:

$$0\,0\,1\,0\,1\,1\,1 \quad 0\,0\,1\,0\,1\,1\,1 \quad \ldots$$

Berechnung der Ausgangsbits
Jedes i-te Bit s_i des Schlüsselstroms lässt sich durch die XOR-Verknüpfung (als mod 2 Operator) der Vorgänger wie folgt berechnen:

in mathematische Schreibweise

$$\begin{aligned} s_3 &\equiv s_1 + s_0 \mod 2 \\ s_4 &\equiv s_2 + s_1 \mod 2 \\ \ldots &\equiv \quad \ldots \\ s_{i+3} &\equiv s_{i+1} + s_i \mod 2 \qquad i = 0, 1, 2, \ldots \end{aligned}$$

Abb. 9.10: *Berechnung der Ausgangsbits s_i*

LFSR [19]

Bitkombinationen (Zustände)

> Mit Ausnahme der Kombination 000 ergeben sich für ein LFSR mit 3 Flipflops eine maximale Anzahl von $2^3-1 = 7$ Zuständen.
>
> *Beobachtung 9.3*

Durch die endliche Anzahl von Möglichkeiten muß in den möglichen Kombinationen der Startwert wieder auftreten. Danach wiederholt sich die Sequenz der Bitkombinationen in derselben Reihenfolge.

Periodenlänge
Für das Beispiel (Abb. 9.7) wiederholen sich die Bitkombinationen nach 7 Verschiebungen. Die Periodenlänge für dieses LFSR ist deshalb 7.

> Für den ausgewählten Rückkopplungspfad des Beispiels erhalten wir die maximale Periodenlänge von $2^3 - 1 = 7$.
>
> *Beobachtung 9.4*

9.3.2 Beispiel LFSR mit $m = 4$

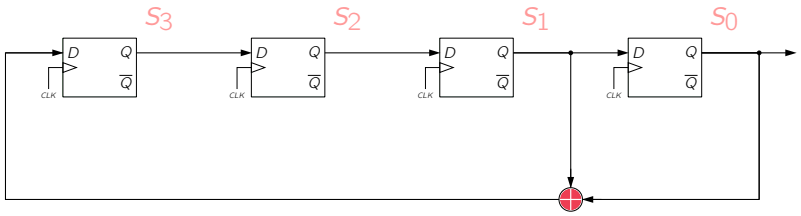


Abb. 9.11: *Beispiel eines LFSR mit* $m = 4$

Rückkopplungskoeffizienten p_i

Die Rückkopplung kann ersatzweise durch ein UND-Gatter realisert und in der Boolschen Algebra durch eine Multiplikation $p_i \cdot s_i$ dargestellt werden.

LFSR [19]

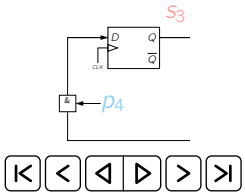


Abb. 9.12: *Rückkopplungskoeffizient* p_m

> Der erste Schalter p_m eines LFSR vom Grad m am Eingang hat immer den Wert 1 ansonsten gäbe es keine Rückkkopplung.
>
> *Beobachtung 9.1*

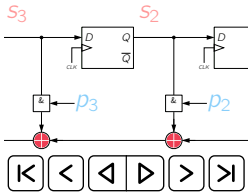


Abb. 9.13: *Inaktive Rückkopplung* (s_2, s_3) *(Schalter offen)*

> Hat der Rückkopplungskoeffizient p_i den Wert 0 ($p_i = 0$) dann ist die Rückkopplung für den Ausgang s_i inaktiv (Schalter offen).
>
> *Definition 9.1*

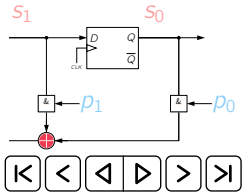


Abb. 9.14: *Aktive Rückkopplung* (s_0, s_1) *(Schalter geschlossen)*

> Hat der Rückkopplungskoeffizient p_i den Wert 1 ($p_i = 1$) dann ist die Rückkopplung für den Ausgang s_i aktiv (Schalter geschlossen).
>
> *Definition 9.2*

Beispiel mit Rückkopplungsvektor $p = (0\,0\,1\,1)$

Bsp (i): m=4 [19]

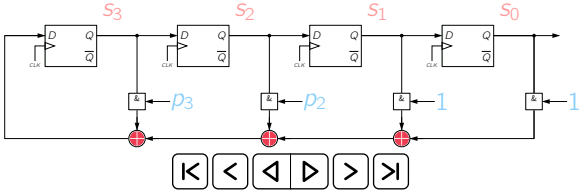


Abb. 9.15: *Beispiel mit* $m = 4$ *und* $p = (0\,0\,1\,1)$

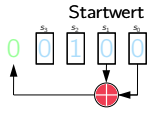


Abb. 9.16: *LFSR mit dem Startwert* 0100

Für den Rückkopplungspfad $p = (0\,0\,1\,1)$ ergibt sich die maximale, periodische Ausgangssequenz:

... 0010011010111110 0010011010111110 ...

Beispiel mit Rückkopplungsvektor $p = (1\,1\,1\,1)$

Bsp (ii): m=4 [19]

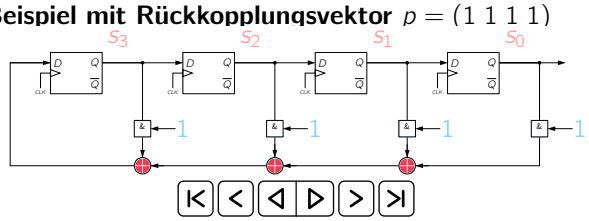


Abb. 9.17: *Beispiel mit* $m = 4$ *und* $p = (1\,1\,1\,1)$

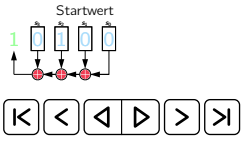


Abb. 9.18: *LFSR mit 3 Perioden (Startwerte:* 0100, 0101, 0111*)*

Für den Rückkopplungspfad $p = (1\,1\,1\,1)$ ergeben sich drei periodische Ausgangssequenzen mit je einer Periodenlänge von 5:

i	b_3	b_2	b_1	b_0
0	0	1	0	0
1	1	0	1	0
2	0	1	0	1
3	0	0	1	0
4	1	0	0	1
0	0	1	0	0

i	b_3	b_2	b_1	b_0
0	0	0	0	1
1	1	0	0	0
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1
0	0	0	0	1

i	b_3	b_2	b_1	b_0
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0
4	1	1	1	1
0	0	1	1	1

Tab. 9.2: *3 Perioden für Ausgangssequenzen*

Berechnung der Ausgangsbits

Die Ausgangssequenz des LFSR wird vom Rückkopplungspfad und dem Startwert (s_0 bis s_3) bestimmt.

Der Rückkopplungspfad kann durch einen Rückkopplungsvektor, für unser Beispiel $(p_3\,p_2\,p_1\,p_0) = (0\,0\,1\,1)$, beschrieben werden.

Die ersten 4 Ausgangsbits ($s_0, s_1, \ldots, s_3$) werden einzig allein nur durch den Startwert bestimmt. s_4 ist das erste Bit welches sich aus den Rückkopplungen wie folgt berechnen läßt.

$$s_4 \equiv s_3 \cdot p_3 + s_2 \cdot p_2 + s_1 \cdot p_1 + s_0 \cdot p_0 \mod 2$$
$$s_4 \equiv s_3 \cdot 0 + s_2 \cdot 0 + s_1 \cdot 1 + s_0 \cdot 1 \mod 2$$
$$s_4 \equiv s_1 + s_0 \mod 2$$

9.3.3 Eigenschaften von LFSR

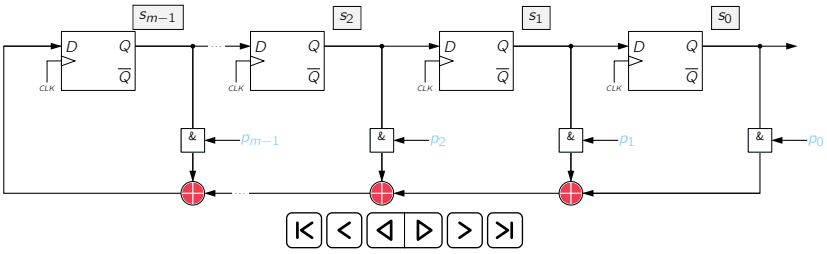


Abb. 9.19: *Prinzip Linear Feedback Shift Register (LFSR)*

Berechnung der Ausgangsbits

Die ersten m Ausgangsbits $(s_0, s_1, \ldots, s_{m-1})$ werden einzig allein durch den Startwert bestimmt. s_m ist das erste Bit welches sich wie folgt berechnen läßt.

$$\begin{aligned} s_m &\equiv s_{m-1} \cdot p_{m-1} + s_{m-2} \cdot p_{m-2} + \ldots + s_0 \cdot p_0 \mod 2 \\ s_{m+1} &\equiv s_m \cdot p_m + s_{m-1} \cdot p_{m-1} + \ldots + s_1 \cdot p_1 \mod 2 \end{aligned}$$

Allgemeine LFSR-Formel in Summenschreibweise

$$s_{m+i} \equiv \sum_{j=0}^{m-1} s_{i+j} \cdot p_j \mod 2$$

Alle Ausgangswerte sind lineare Summen von vorhergehenden Ausgangswerten.

Beobachtung 9.1

Grad eines LFSR

Der **Grad** m eines LFSR ist die Anzahl der verwendeten Speicher-Elemente (Flipflops) und damit die Anzahl der Bits.

Definition 9.1

Polynomdarstellung eines LFSR

Ein LFSR kann durch ein Polynom wie folgt beschrieben werden:

$$P(x) = x^m + p_{m-1} \cdot x^{m-1} + \ldots + p_1 \cdot x + p_0$$

oder in der Summenschreibweise:

$$P(x) = x^m + \sum_{i=0}^{m-1} p_i \cdot x^i$$

Periodenlänge eines LFSR

Die **maximale Länge einer Bitfolge**, die ein LFSR vom Grad m erzeugen kann, ist $2^m - 1$. Man spricht dann von einem LFSR mit **Maximalfolge**.

Satz 9.1

Nur bestimmte **Rückkopplungspfade** d.h. Konfigurationen von **Rückkopplungskoeffizienten** $(p_0, \ldots, p_{m-1})$ von ein und denselben LFSR erzeugen Folgen maximaler Länge erzeugen.

Beobachtung 9.2

Fazit

Darstellung eines LFSR

Ein LFSR kann auf folgende Arten eindeutig spezifiziert werden:

- Schaltung
- Koeffizientenvektor $p = (p_{m-1}, \ldots, p_0)$
- Polynomdarstellung $P(x)$

Eigenschaften [19]

Angriffe [19]

10 Der Data Encryption Standard (DES)

Inhaltsangabe

10.1 Symmetrie-Eigenschaften

Das Verstehen von **DES**[a] ist aus heutiger Sicht wichtig, da es sich um den am besten untersuchten symmetrischen Algorithmus handelt, dessen Design viele aktuelle Chiffren beeinflusst hat. [3, S 63]

In der Kryptografie sind die Symmetrieeigenschaften notwendig, da die Idee der Dechiffrierung darin liegt, die Verschlüsselung rundenweise rückgängig zu machen.

Nachfolgend werden nochmals kurz wichtige Eigenschaften von krpytografischen Grundschaltungen wiederholt:

[a] **englisch:** *DES* = Data Encryption Standard

10.1.1 XOR-Gatter

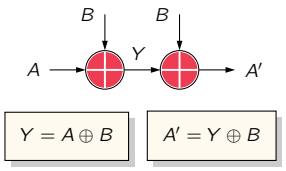
Das XOR-Gatter ist eine der wichtigsten Elemtente in der Kryptografie.

Eine Addition modulo 2 ist gleichbedeutend mit der XOR-Operation.

Beobachtung 10.1

XOR-Ver- und Entschlüsselung

Bei der Verwendung von XOR sind die Verschlüsselung und die Entschlüsselung die gleiche XOR-Operation.



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Y	B	A'
0	0	0
1	1	0
1	0	1
0	1	1

Diese zunächst überraschende Eigenschaft kann mathematisch leicht hergeleitet werden:

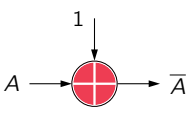
Zusammenhang zw. XOR und Modulo-Arithmetik

$$Y = A \oplus B = A + B \pmod 2$$
$$A' = Y \oplus B = Y + B \pmod 2$$

Kongruenz-Umformung

$$\begin{aligned} A' &\equiv Y + B \pmod 2 \equiv \\ &\equiv A + B + B \pmod 2 \\ &\equiv A + 2B \pmod 2 \\ &\equiv A + 0 \pmod 2 \\ &\equiv A \pmod 2 \end{aligned}$$
$$A' = \underline{\underline{A}}$$

XOR-Gatter als Inverter



A	B	$\overline{A}$
0	1	1
1	1	0

Weiters kann XOR auch als Inverter verwendet werden. Wird ein Eingang (z. B. B) konstant auf den Wert 1 gesetzt so wird der andere Eingang invertiert.

XOR Häufigkeitsverteilung

Die XOR-Funktion, ist im Gegensatz zu vielen anderen booleschen Gattern, perfekt ausbalanciert, d.h. bei gegebenen Ausgangswert sind alle Eingangswerte gleich wahrscheinlich.

10.1.2 Permutation

Eine **Permutation** ist eine Umordnung von Objekten von einer vorgegebenen Reihenfolge.

Definition 10.1

Bei einer Permutation wechselt jedes Bit des Datenblocks am Eingang seinen Platz, dabei bleibt der Wert unverändert, d.h. die Anzahl der Nullen und Einsen ändern sich nicht.

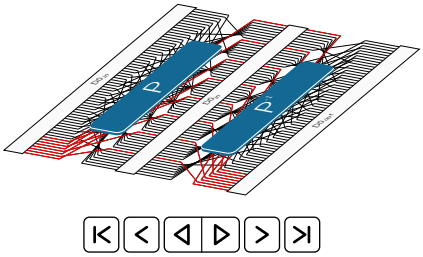


Abb. 10.1: *Permutation P und ihre Umkehrung* P^{-1}

10.1.3 Feistel-Chiffre

Viele moderne symmetrische Verschlüsselungsverfahren basieren auf Feistelnetzwerken, weil damit leicht die Entschlüsselung als Umkehrung der Verschlüsselung realisiert werden kann.

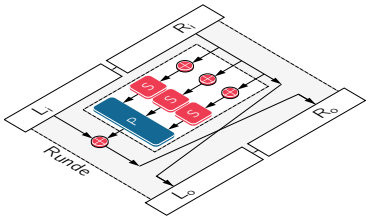


Abb. 10.2: *Feistel Cipher - Verschlüsselung*

Da im Feistelnetzwerk pro Runde immer nur eine Hälfte des Datenblockes verschlüsselt wird, kann mit den Ausgangsdaten und den gleichen Rundenschlüssel die Orginaldaten wieder rekunstruiert werden.

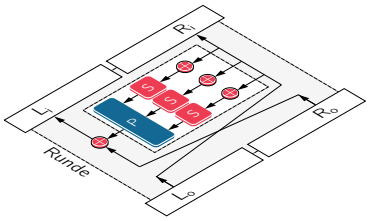


Abb. 10.3: *Feistel Cipher - Entschlüsselung*

Zur Entschlüsselung wird der gleiche Algorithmus verwendet, nur werden die Runden in umgekehrter Reihenfolge durchlaufen, d. h. die Rundenschlüssel in umgekehrter Reihenfolge verwendet.

Beobachtung 10.2

10.2 Übersicht über DES

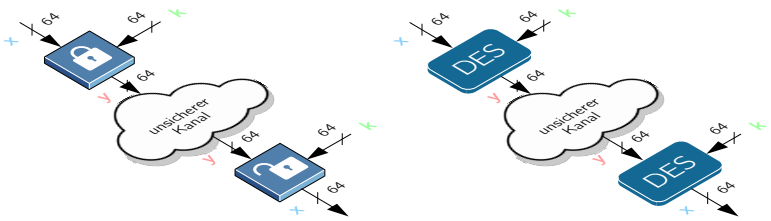


Abb. 10.4: *Ein- und Ausgangsparameter von DES (Symmetrie)*

10.2.1 Ein- und Ausgangsparameter

Beim DES-Algorithmus werden Blöcke von 64 Bit (Klartext x) mit einem 56-Bit-Schlüssel wieder zu Blöcken von 64 Bit (Geheimtext y) verschlüsselt.

DES [20]

DES verwendet folgende Eingangs- und Ausgangsparameter (siehe Abb. 10.4):

- x ... Klartext (64 Bit)
- k ... Schlüssel (56 Bit)
- y ... Geheimtext (64 Bit)

DES ist ein symmetrisches Verfahren, d. h. derselbe Schlüssel k wird sowohl für die Ver- als auch für die Entschlüsselung verwendet.
Beobachtung 10.1

Es gilt:

$$y = \mathbf{DES}_k(x)$$

Im inneren Aufbau von DES erfolgt die Verarbeitung der Daten auf zwei Pfaden:

- **Datenpfad**
  Hier werden in jeder Runde i im Feistel-Netzwerk die Eingangsdaten verschlüsselt.
- **Schlüsselpfad** (Schlüsselfahrplan)
  Hier wird für jede Runde i aus dem Hauptschlüssel k ein Rundenschlüssel k_i abgeleitet $(k_1, \ldots, k_{16})$.

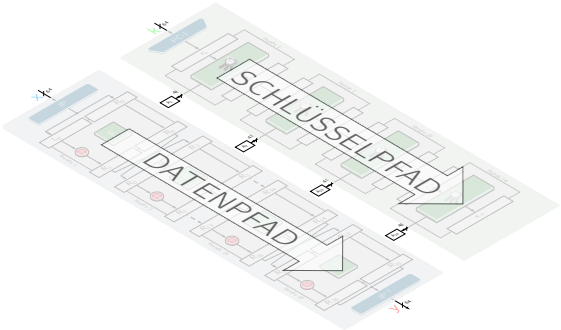


Abb. 10.5: *Daten- und Schlüsselpfad*

DES verwendet für die Ver- und Entschlüsselung einen iterativen[1] Algorithmus, d. h. für jeden Klartextblock wird die Ver- bzw. Entschlüsselung in 16 Runden durchgeführt, die alle identische Operationen ausführen.

10.2.2 Datenpfad (Feistel-Netzwerk)

Ein Vorteil vom Feistel-Netzwerk ist, dass Ver- und Entschlüsselung fast identisch sind.
Beobachtung 10.2

Feistel-NW [20]

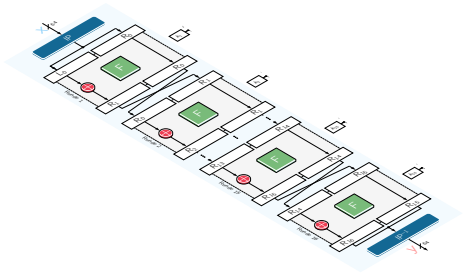


Abb. 10.6: *Datenpfad*

Der Datenpfad besteht aus folgenden Komponenten:

- Eingangspermutation (IP)
- Feistel-Netzwerk für jede Runde i
- Ausgangspermutation (IP^{-1})

Durch die Eingangs- bzw. Ausgangspermutation wird die Sicherheit von DES nicht erhöht.
Beobachtung 10.3

10.2.3 Schlüsselpfad (Transformation)

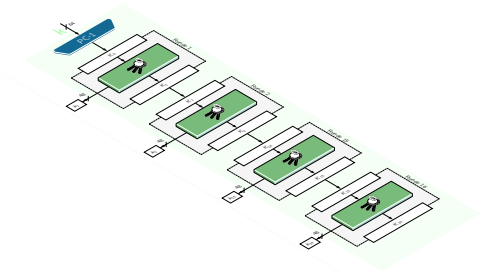


Abb. 10.7: *Schlüsselpfad*

Der Schlüsselpfad besteht aus folgenden Komponenten:

- Eingangspermutation $PC-1$ (Permuted Choice)
- Schlüsseltransformation k_i für jede Runde i

[1] **lateinisch:** *iterare* = wiederholen

10.3 DES-Algorithmus

10.3.1 Verschlüsselung

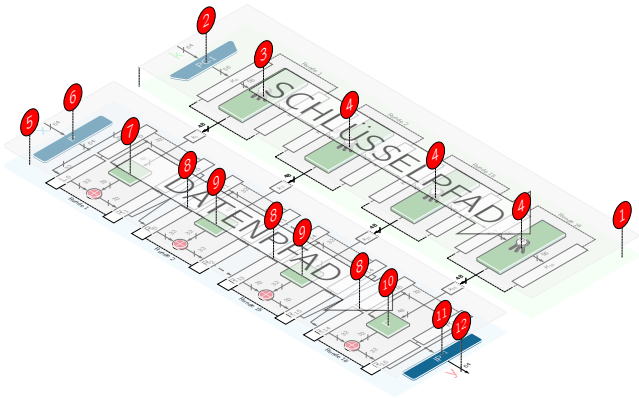


Abb. 10.8: *DES-Algorithmus (Verschlüsselung)*

Schlüsselfahrplan

Schlüsselfahrplan [21]

Aus dem Hauptschlüssel k werden in Folge die Rundenschlüssel $k_1, \ldots, k_{16}$ abgeleitet.

Beobachtung 10.1

1. **Schlüsselpfad**
   Transformation des Hauptschlüssels k im Schlüsselpfad
2. **Entfernung der Paritätsbits:**
   Durch die Permutation PC[2]-1 wird der 64-Bit-Eingangswert k auf 56 Bit K_0 reduziert.
3. **1. Runde der Schlüssel-Transformation:**
   Aus dem Hauptschlüssel K_0 wird durch eine Transformation der Rundenschlüssel k_1 abgeleitet.
4. **Iteration der Schlüssel-Transformation:**

Insgesamt wird die Schlüssel-Transformation 16× für die Rundenschlüssel $k_1, \ldots, k_{16}$ durchgeführt.

Feistelnetzwerk

5. **Datenpfad**
   Die Verschlüsselung der Daten erfolgt im Datenpfad
6. **Eingangspermutation:**
   Der 64-Bit-Klartext x wird nach der bitweisen Eingangspermutation IP in zwei Hälften $L0$ und $R0$ aufgeteilt.
7. **1. Runde im Feistel-Netzwerk:**
   Die rechte Hälfte $R0$ wird in der f-Funktion mit dem Rundenschlüssel k_1 verarbeitet. Das Ergebnis mit der linken Hälfte $L0$ XOR verknüpft und man erhält $R1$.
8. **Vertauschung:**
   $R1$ und $R0$ werden, vor der Verarbeitung der nächsten Runde im Feistel-Netzwerk, vertauscht.
9. **Iteration im Feistel-Netzwerk:**
   Insgesamt 16 Runden, d. h. mit den Rundenschlüssel $k_1, \ldots, k_{16}$.
10. **Letzte Runde:**
    Die Vertauschung entfällt. Über die Ausgangspermutation IP^{-1} erhält man den Geheimtext y.

Die Ausgangspermutation IP^{-1} ist die inverse Operation der Eingangspermutation IP.

Beobachtung 10.2

Schlüsselstrom

Im Feistelnetzwerk wird pro Runde immer nur eine Hälfte der Daten verschlüsselt.

Beobachtung 10.3

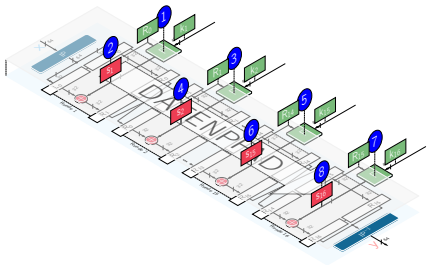


Abb. 10.9: *Schlüsselstrom s_i*

DES-Schlüsselfahrplan [15]

Man kann sich die f-Funktion auch als Schlüsselstromgenerator für die XOR-Verknüpfung vorstellen.

1. **f-Funktion (1. Runde)**
   In der f-Funktion wird R_0 und k_1 verarbeitet.
2. **Schlüsselstrom s_1**
   Als Ergebnis erhält man den Schlüsselstrom s_1.
3. **f-Funktionn (2. Runde)**
   In der f-Funktion wird R_1 und k_2 verarbeitet.
4. **Schlüsselstrom s_2**
   Als Ergebnis erhält man den Schlüsselstrom s_2.
5. **Iteration des Schlüsselstroms**
   In insgesamt 16 Runden werden die Schlüsselströme $s_1, \ldots, s_{16}$ generiert.
6. ...

Wir erhalten als Ergebnis (Abb. 10.9):

f-Funktion				
Runde		in	in	out
1	1.	R_0	k_1	s_1
3	2.	R_1	k_2	s_2
...		...	...	...
5	15.	R_{14}	k_{15}	s_{15}
7	16.	R_{15}	k_{16}	s_{16}

XOR Verschlüsselung				
Runde		in	in	$s_i \oplus L_i$
2	1.	s_1	L_0	R_1
4	2.	s_2	R_0	R_2
...		...	...	...
6	15.	s_{15}	R_{14}	R_{15}
8	16.	s_{16}	R_{15}	R_{16}

Tab. 10.1: *f-Funktion und XOR (Verschlüsselung)*

[2] **englisch:** PC = Permuted Choice (permutierte Auswahl)

10.3.2 Entschlüsselung

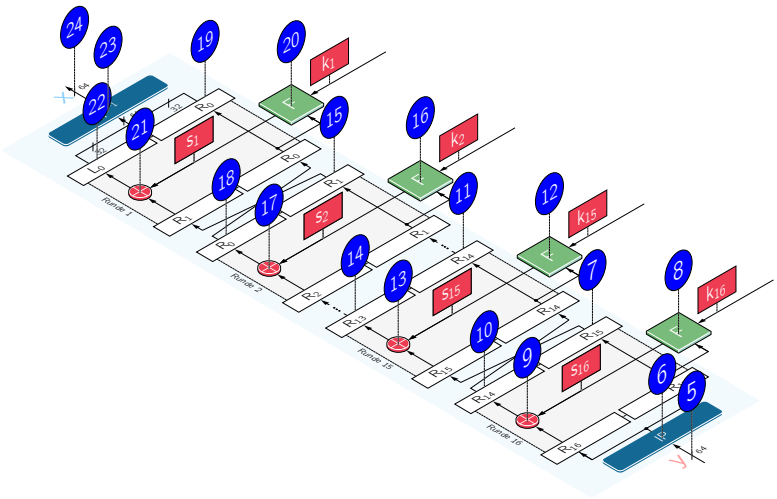


Abb. 10.10: *Umkehrung des Schlüsselstroms s_i*

Umgekehrter Schlüsselstrom

Für die Entschlüsselung werden im Feistelnetzwerk die Rundenschlüssel in umgekehrter Reihenfolge $k_{16}, \ldots, k_1$ benötigt.

Beobachtung 10.1

Entschlüsselung [21]

f-Funktion				
	Runde	in	in	out
8	1.	R_{15}	k_{16}	s_{16}
12	2.	R_{14}	k_{15}	s_{15}
	...	...		...
16	15.	R_1	k_2	s_2
20	16.	R_0	k_1	s_1

XOR Entschlüsselung				
	Runde	in	in	$s_i \oplus R_i$
9	1.	s_{16}	R_{16}	18 R_{14}
13	2.	s_{15}	R_{15}	14 R_{13}
	...	...		...
17	15.	s_2	R_2	19 R_0
21	16.	s_1	R_1	22 L_0

Tab. 10.2: *f-Funktion und XOR (Entschlüsselung)*

6 **Eingangspermutation:**
Der 64-Bit-Geheimtext y wird nach der bitweisen Eingangspermutation IP in zwei Hälften aufgeteilt.

7 **1. Runde (rechte Hälfte):**
Die rechte Hälfte $R15$ wird direkt in die nächste Runde übernommen.

8 **1. Runde (f-Funktion):**
Aus der rechten Hälfte $R15$ wird über die f-Funktion der Schlüsselstrom s_{16} abgeleitet.

9 **1. Runde (linke Hälfte):**
Die linke Hälfte $R16$ wird mit dem Schlüsselstrom s_{16} XOR verknüpft.

10 **1. Runde (linke Hälfte):**
Man erhält $R14$. $R14$ und $R15$ werden, vor der Verarbeitung in der nächsten Runde vertauscht.

11 ...

Insgesamt werden 16 Runden auf diese Art verarbeitet. Die Ergebnisse der f-Funktion und der XOR-Verknüpfung für die Entschlüsselung sind in der Tabelle Tab. 10.2 angeführt.

Die f-Funktion erzeugt für die selben Eingangswerte immer die selben Ausgangswerte.

Beobachtung 10.2

Bei der Entschlüsselung kann mit den Schlüsselstrom s_i in umgekehrter Reihenfolge die verschlüsselten Daten mit der XOR-Verknüpfung wieder entschlüsselt werden.

19 **16. Runde (rechte Hälfte):**
Die rechte Hälfte $R0$ wird direkt in die nächste Runde übernommen.

20 **16. Runde (f-Funktion):**
Aus der rechten Hälfte $R0$ wird über die f-Funktion der Schlüsselstrom s_1 abgeleitet.

21 **16. Runde (linke Hälfte):**
Die linke Hälfte $R1$ wird mit dem Schlüsselstrom s_1 XOR verknüpft.

22 **16. Runde (linke Hälfte):**
Man erhält $L0$.

23 **Ausgangspermutation:**
Über die Ausgangspermutation IP^{-1} erhält man wieder den Klartext x.

10.4 Struktur des Datenpfades

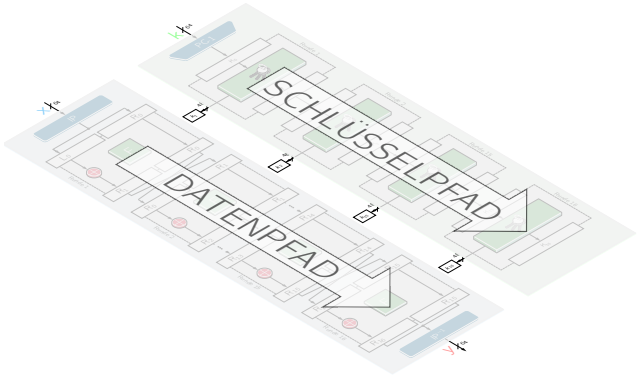


Abb. 10.11: *Daten- und Schlüsselpfad*

10.4.1 Eingangspermutation

Eingangspermutation *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

Tab. 10.3: *Eingangspermutation IP*

Eingangspermutation IP [20]

Die Tabelle besagt, dass das Eingangsbit an Position 58 (Tabellenindex) auf das Ausgangsbit an Position 1 abgebildet wird, das Eingangsbit 50 auf die zweite Position am Ausgang und so weiter.

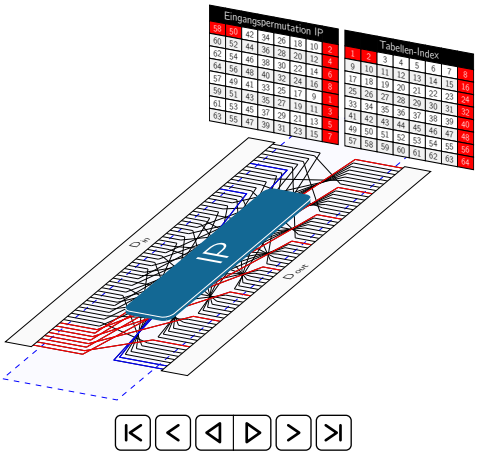


Abb. 10.12: *Eingangspermuation IP*

Eine Bitpermutation kann man sich als einfache Vertauschung von elektrischen Drähten vorstellen.

In Hardware können Bitpermutationen extrem einfach umgesetzt werden, ihre Softwarerealisierung ist etwas aufwendiger.

10.4.2 Ausgangspermutation

Ausgangspermutation 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

Tab. 10.4: *Ausgangspermutation* IP^{-1}

Die Tabelle besagt, dass das Eingangsbit an Position 40 (Tabellenindex) auf das Ausgangsbit an Position 1 abgebildet wird, das Eingangsbit 8 auf die zweite Position am Ausgang und so weiter.

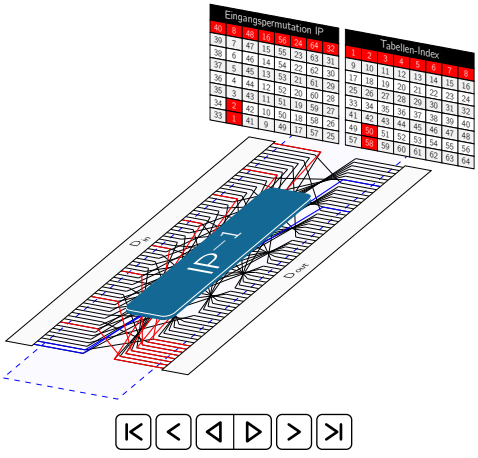


Abb. 10.13: *Ausgangspermutation* IP^{-1}

Die Ausgangspermutation ist die Umkehrung der Eingangspermutation, d.h. würde man die Eingangs- und Ausgangspermutation hintereinanderschalten, würden sich beide Operationen aufheben und es wäre keine Veränderung bemerkbar.

10.4.3 Die f-Funktion

Wie bereits erwähnt spielt die f-Funktion eine zentrale Rolle für die Sicherheit des DES. In Runde i verwendet sie als Eingabe die rechte Hälfte R_{i-1} der Ausgabe der vorherigen Runde und den aktuellen Rundenschlüssel k_i .

f-Funktion [20]

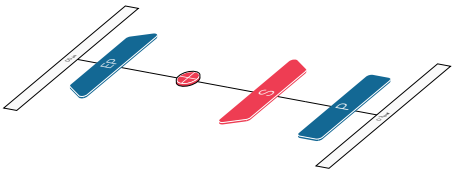


Abb. 10.14: *Struktur der f-Funktion*

Die Ausgabe der f-Funktion wird zur XOR-Verschlüsselung der linken Eingangs-Hälfte verwendet.

Expansion E

Expansion [20]

Zuerst werden die 32 Eingangsbits auf 48 Bits expandiert, indem jeweils Blöcke des Eingangsworts von vier Bits auf sechs Bits expandiert werden.

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

Tab. 10.5: *Tabelle der Expansion E*

Diese spezielle Art der Expansion passiert in der E-Box (Abb. 10.15).

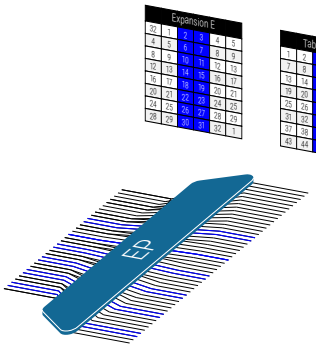


Abb. 10.15: *Expansion innerhalb der f-Funktion*

Substitution-Boxen

Substitution [20]

Die S-Boxen sind der eigentliche kryptografische Kern des DES. Sie sind die einzigen nichtlinearen Elemente in dem Algorithmus und erzeugen Konfusion.

S-Box S_1																
S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Tab. 10.6: *Tabelle der S-Box S_1*

Ohne ein nichtlineares Element könnte ein Angreifer den Zusammenhang zwischen DES- Eingang und -Ausgang durch ein System linearer Gleichungen beschreiben.

Solche Systeme können einfach und sehr effizient gelöst werden.

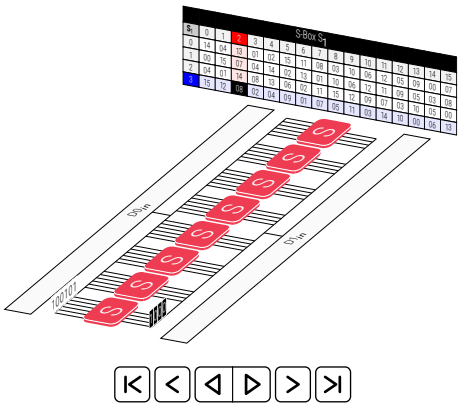


Abb. 10.16: *S-Boxen innerhalb der f-Funktion*

Permutation

Zuletzt wird der 32-Bit-Ausgang bitweise mithilfe der Permutation P verwürfelt (Tab. **??**).

Tabelle der 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

Tab. 10.7: *Tabelle der Permutation P*

Permutation [20]

Die Permutation P dient der Diffusion der Verschlüsselung, da die vier Ausgabebits jeder S-Box derart permutiert werden, dass sie jeweils mehrere S-Boxen in der nächsten Runde beeinflussen.

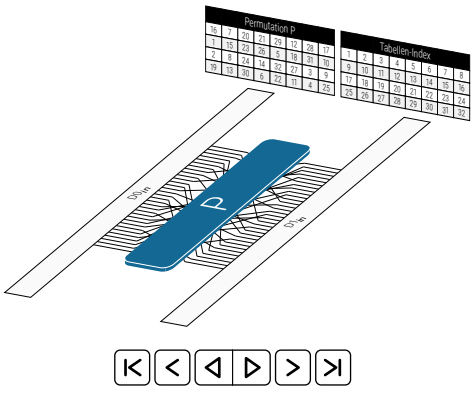


Abb. 10.17: *Permutation innerhalb der f-Funktion*

Durch die von der E-Box, den S-Boxen und der P-Permutation erzeugte Diffusion wird garantiert, dass jedes Bit am Ende der fünften Runde eine Funktion von jedem Bit des Klartexts und des Schlüssels ist.

Dieses Verhalten ist auch als Avalanche-Effekt (Lawineneffekt) bekannt.

11 Der Advanced Encryption Standard (AES)

Inhaltsangabe

11.1 Merkmale von AES

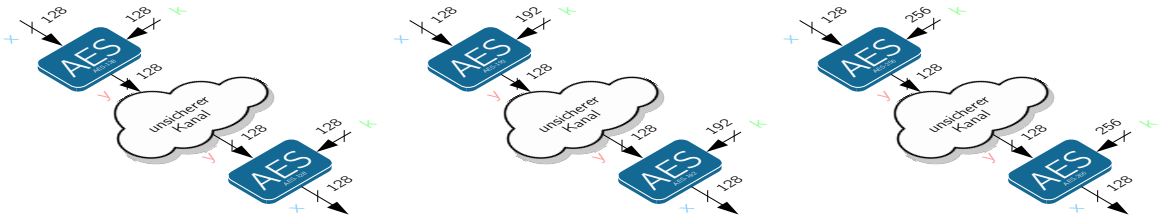


Abb. 11.1: *Ein- und Ausgangsparameter von AES-128, AES-192 und AES-256*

11.1.1 Übersicht

AES ist die heutzutage am meisten genutzte symmetrische Chiffre überhaupt. Die AES-Blockchiffre ist für zahlreiche behördliche und industrielle Anwendungen als Standard vorgeschrieben.

AES-Verschlüsselung [15]

AES basiert auf einem SP-Netzwerk[1] und verwendet sowohl für die Verschlüsselung als auch für die Entschlüsselung denselben grundlegenden Algorithmus.

Der AES-Algorithmus ist ein Byte-orientiertes Verfahren und arbeitet in jeder Phase mit Datenblöcke von 16 Bytes (128 Bits).

> Die Gesamtheit eines 16-Byte-Datenblockes (128-Bit-Datenpfad) wird als **Zustand** bezeichnet und in einer 4 × 4-Byte-Matrix dargestellt.
>
> *Definition 11.1*

11.1.2 Runden

Im Gegensatz zum DES hat AES keine Feistel-Struktur. In jeder Runde werden alle 128 Bits verschlüsselt.

> Ein Durchgang durch das SP-Netzwerk wird als **Runde** bezeichnet
>
> *Definition 11.2*

Die Runden-Anzahl hängt von der Schlüssellänge ab:

AES		
Schlüssellänge	Runden	Bezeichnung
128 Bit	10	AES-128
192 Bit	12	AES-192
256 Bit	14	AES-256

Tab. 11.1: *AES Schlüssellängen und Runden*

AES-Schlüsselfahrplan [15]

Dies ist ein Grund, warum AES im Vergleich zu DES weniger Runden hat.

11.1.3 Schichten

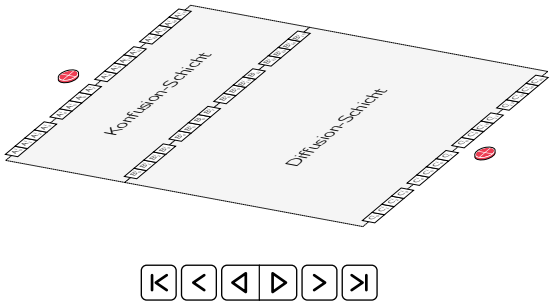


Abb. 11.2: *Schichten von AES*

Eine Runde besteht aus folgenden Schichten:

- **Key-Addition**
  Bitweise XOR-Verknüpfung zwischen Block und dem aktuellen 128-Bit-Rundenschlüssel.
- **Byte-Substitution**
  Jedes Byte des Zustands wird über eine nichtlineare Transformation[2] durch ein anderes Byte ersetzt.
- **ShiftRow (Unterschicht)**
  Hierbei wird der Datenblock auf Zeilen-Ebene byteweise permutiert.
- **MixColumn (Unterschicht)**
  Hierbei wird der Datenblock auf Spalten-Ebene bitweise miteinander verwürfelt.

11.1.4 Zustände

Den Datenpfad nennt man auch den Zustand des Algorithmus.

Mit Ausnahme der letzten Runde besteht jede Runde aus allen drei Schichten, wie in Abb. 11.3 dargestellt:

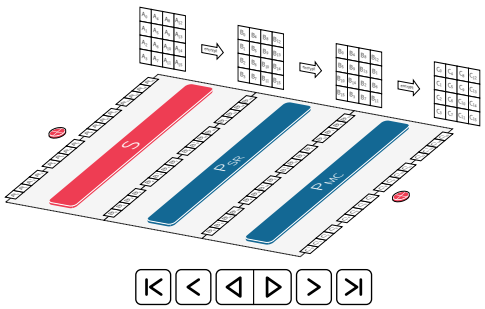


Abb. 11.3: *Zustände in den Schichten*

Der Klartext wird hierbei mit x bezeichnet, das Chiffrat mit y und die Anzahl der Runden mit nr.

Jedoch wird in der letzten Runde nicht die MixColumn-Transformation verwendet. Die AES-Ver- und -Entschlüsselung sind symmetrisch aufgebaut.

[1] Substitutions-Permutations-Netzwerk

[2] S-Box mit speziellen mathematischen Eigenschaften

11.2 Byte-Substitutions-Schicht

11.2.1 Verschlüsselung

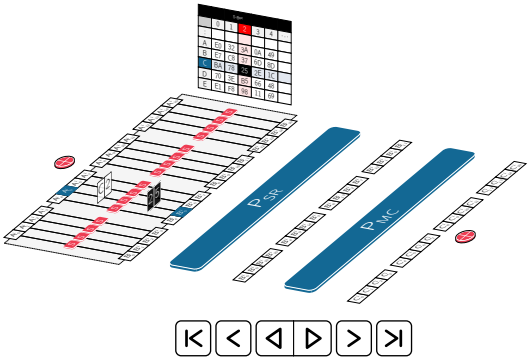


Abb. 11.4: *Byte-Substitutions-Schicht*

Bei der Verschlüsselung besteht die erste Schicht jeder Runde aus der Byte-Substitution-Schicht.

Sie besteht aus 16 parallelen S-Boxen, wobei jede S-Box 8 Ein- und Ausgabebits hat.

AES-S-Box S

	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

Abb. 11.5: *Tabelle der S-Box S*

Die Konfusion der Daten erfolgt in der Byte-Substitution-Schicht.

Im Gegensatz zu DES sind beim AES alle 16 S-Boxen identisch. Die Operation der S-BOX ist nicht linear.

Beobachtung 11.1

11.2.2 Entschlüsselung

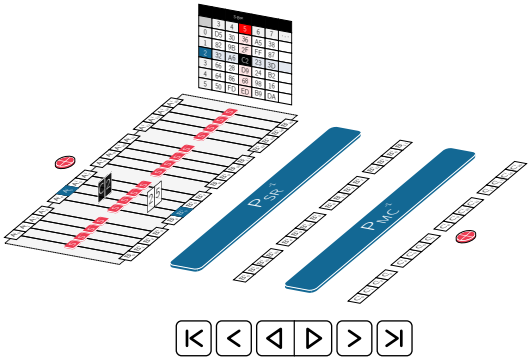


Abb. 11.6: *Inverse Byte-Substitutions-Schicht*

Bei der Entschlüsselung besteht die letzte Schicht jeder Runde aus der Byte-Substitution-Schicht.

Sie besteht aus 16 parallelen inverse S-Boxen, wobei jede inverse S-Box 8 Ein- und Ausgabebits hat.

inverse AES-S-Box S^{-1}

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Abb. 11.7: *Tabelle der inversen S-Box S^{-1}*

Die Idee der Dechiffrierung darin liegt, die Verschlüsselung rundenweise rückgängig zu machen.

Zur Entschlüsselung wird die Funktion der AES-S-Box umgedreht, d. h. die S-Box-Transformation wird rückgängig gemacht.

Beobachtung 11.2

11.3 ShiftRows-Unterschicht

Beim AES besteht die Diffusionsschicht aus zwei Teilen (Unterschichten), der **ShiftRow**-Transformation und der **MixColumn**-Transformation.

Bei der Shift-Row-Transformation erfolgt die Diffusion per Zeile und byteweise.

11.3.1 Verschlüsselung

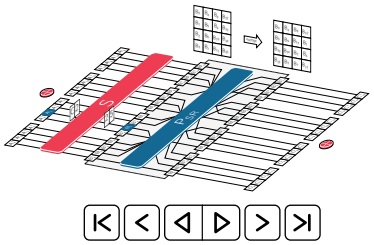


Abb. 11.8: *Funktionsweise der ShiftRows-Unterschicht*

Bei der ShiftRow-Transformation wird die Zustands-Matrix zeilenweise permutiert, d.h. zyklisch nach rechts verschoben (rotiert).

ShiftRows Transformation	
Zeile	Verschiebungen
1. Zeile	keine
2. Zeile	3 Byte nach rechts
3. Zeile	2 Byte nach rechts
4. Zeile	1 Byte nach rechts

Tab. 11.2: *ShiftRows Transformation*

Durch die ShiftRows-Operation wird die Diffusionseigenschaft des AES erhöht.

Beobachtung 11.1

11.3.2 Entschlüsselung

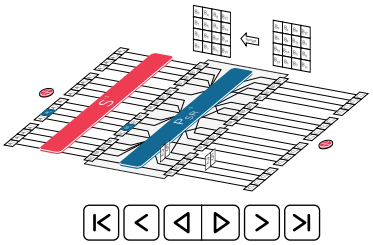


Abb. 11.9: *Funktionsweise der ShiftRows-Unterschicht*

Bei der inversen ShiftRow-Transformation wird die Zustands-Matrix zeilenweise permutiert, d.h. zyklisch nach links verschoben (rotiert).

ShiftRows Transformation	
Zeile	Verschiebungen
1. Zeile	keine
2. Zeile	3 Byte nach links
3. Zeile	2 Byte nach links
4. Zeile	1 Byte nach links

Tab. 11.3: *ShiftRows Transformation*

11.4 MixColum-Unterschicht

Bei der zweiten Diffusionsschicht (Unterschichten) erfolgt die Diffusion per Spalte und bitweise. Durch die zeilen- und spaltenweise Diffusion werden möglichst viele Bits des Zustandes beinflusst.

Der MixColumn-Schritt ist eine lineare Transformation, die die Bytes jeder Spalte der Zustandsmatrix untereinander verwürfelt.

11.4.1 Verschlüsselung

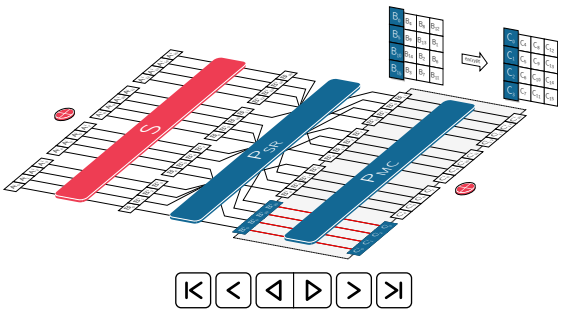


Abb. 11.10: *MixColum-Unterschicht*

Berechnung der 1. Spalte

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

Berechnung der 2. Spalte

$$\begin{pmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_4 \\ B_9 \\ B_{14} \\ B_3 \end{pmatrix}$$

Berechnung der 3. Spalte

$$\begin{pmatrix} C_8 \\ C_9 \\ C_{10} \\ C_{11} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_8 \\ B_{12} \\ B_2 \\ B_7 \end{pmatrix}$$

Berechnung der 4. Spalte

$$\begin{pmatrix} C_{12} \\ C_{13} \\ C_{14} \\ C_{15} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_{12} \\ B_1 \\ B_6 \\ B_{11} \end{pmatrix}$$

Bei der MixColumn-Operation wird erreicht, dass jedes der vier Bytes einer Spalte alle anderen Bytes der Spalte beeinflusst.

Beobachtung 11.1

11.4.2 Entschlüsselung

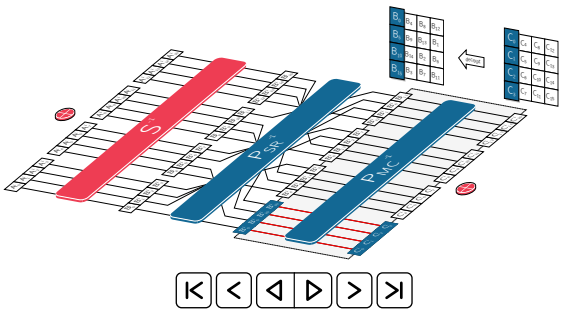


Abb. 11.11: *Inverse MixColum-Unterschicht*

Berechnung der 1. Spalte

$$\begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 0B & 0E & 09 & 0D \\ 0D & 0B & 0E & 09 \\ 09 & 0D & 0B & 0E \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

Berechnung der 2. Spalte

$$\begin{pmatrix} B_4 \\ B_9 \\ B_{14} \\ B_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 0B & 0E & 09 & 0D \\ 0D & 0B & 0E & 09 \\ 09 & 0D & 0B & 0E \end{pmatrix} \begin{pmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \end{pmatrix}$$

Berechnung der 3. Spalte

$$\begin{pmatrix} B_8 \\ B_{12} \\ B_2 \\ B_7 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 0B & 0E & 09 & 0D \\ 0D & 0B & 0E & 09 \\ 09 & 0D & 0B & 0E \end{pmatrix} \begin{pmatrix} C_8 \\ C_9 \\ C_{10} \\ C_{11} \end{pmatrix}$$

Berechnung der 4. Spalte

$$\begin{pmatrix} B_{12} \\ B_1 \\ B_6 \\ B_{11} \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 0B & 0E & 09 & 0D \\ 0D & 0B & 0E & 09 \\ 09 & 0D & 0B & 0E \end{pmatrix} \begin{pmatrix} C_{12} \\ C_{13} \\ C_{14} \\ C_{15} \end{pmatrix}$$

Die MixColumn-Operation ist das ausschlaggebende Diffusionselement des AES.

Beobachtung 11.2

11.5 Key-Addition-Schicht

11.5.1 Verschlüsselung

11.5.2 Entschlüsselung

Asymmetrische Kryptografie

12 Übersicht

Inhaltsangabe

12.1 Gegenüberstellung

12.1.1 Symmetrische Kryptografie

PKC [26]

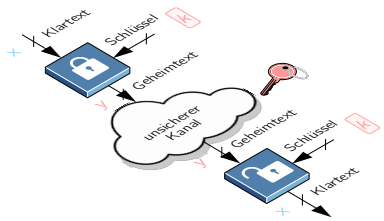


Abb. 12.1: *Symmetrisch: einzelner Schlüssel*

Schlüssel

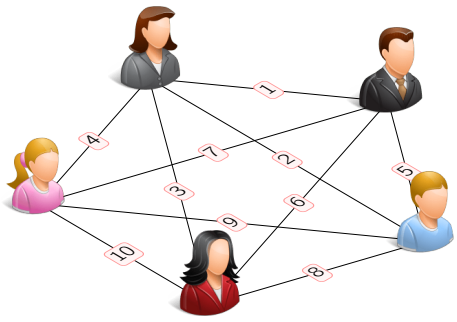
Bei symmetrischen Kryptoverfahren wird für je 2 Kommunikationsteilnehmer ein geheimer Schlüssel verwendet:

- **geheimer Schlüssel** k
  Die Besitzer des geheimen Schlüssels können damit Information ver- und entschlüsseln.
- **Schlüssel-Verteilung**
  Ein Kommunikationsteilnehmer muß für jeden Kommunikationspartner einen geheimen Schlüssel bereitstellen.

Bekannte Familien

- DES
- 3DES
- AES

Schlüsselverteilung



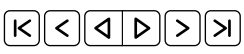


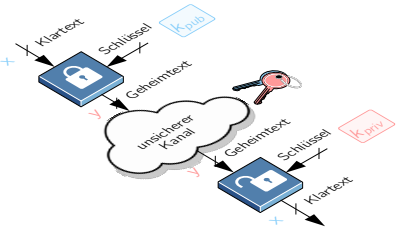
Abb. 12.2: *Schlüsselverteilung (symmetrisch)*

In einem Netz mit n Teilnehmern berechnet sich die Anzahl der N zu verteilenden Schlüssel wie folgt[1]:

$$N = \frac{n \cdot (n-1)}{2} \approx \frac{n^2}{2}$$

Effizienz

12.1.2 Asymmetrische Kryptografie



PKC [26]

Abb. 12.3: *Asymmetrisch: ein Schlüsselpaar*

Schlüssel

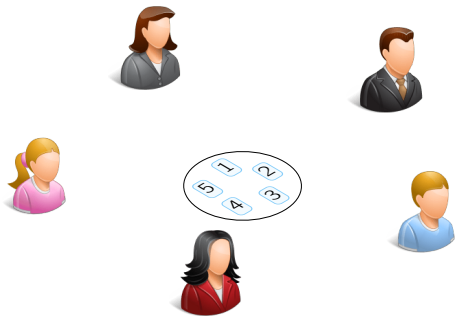
Bei asymmetrischen Kryptoverfahren besitzt jeder Kommunikationsteilnehmer ein Schlüsselpaar:

- **öffentlicher Schlüssel** k_{pub}
  Jeder kann damit Informationen für den Besitzer des privaten Schlüssels verschlüsseln.
- **privater Schlüssel** k_{priv}
  Der Besitzer des privaten Schlüssels kann, die mit dem öffentlichen Schlüssel verschlüsselte Informationen entschlüsseln.

Bekannte Familien

- RSA
- **DHKE**[2]
- **ECC**[3]

Schlüsselverteilung



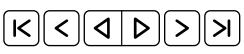


Abb. 12.4: *Schlüsselverteilung (asymmetrisch)*

In einem Netz mit n Teilnehmern berechnet sich die Anzahl der N zu verteilenden Schlüssel wie folgt:

$$N = n$$

Effizienz

[1] siehe auch Summenformel einer arithmetischen Folge

[2] **englisch:** *DHKE* = Diffie-Hellman-Key-Exchange (Diffie-Hellman-Schlüsselaustausch)

[3] **englisch:** *ECC* = Elliptic-Curve-Cryptography (Elliptische-Kurven-Kryptografie)

13 Das RSA-Kryptosystem

Inhaltsangabe

13.1 Mathematische Grundlagen

13.1.1 RSA Schlüsselgenerierung

Satz von Euler

Der Satz von Euler ist eine Verallgemeinerung des kleinen fermatschen Satzes auf beliebige ganzzahlige Moduln n die nicht prim sind:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

Sei $n = p \cdot q$ das Produkt zweier Primzahlen p und q. Dann gilt für jede natürliche Zahl k und jede natürliche Zahl $x \leq n$ mit $ggT(x, n) = 1$:

$$x^{k(p-1)(q-1)+1} \equiv x \pmod{n}$$

Behauptung 13.1

$$\begin{aligned}
x^{\phi(n)} &\equiv 1 \pmod{n} && \text{❶}\\
x^{(p-1)(q-1)} &\equiv 1 \pmod{n} \quad \big|^k && \text{❷}\\
\left(x^{(p-1)(q-1)}\right)^k &\equiv 1^k \pmod{n} && \text{❸}\\
x^{k(p-1)(q-1)} &\equiv 1 \pmod{n} \quad \big| \cdot x\\
x^{k(p-1)(q-1)} \cdot x &\equiv 1 \cdot x \pmod{n}\\
x^{k(p-1)(q-1)+1} &\equiv \underline{x \pmod{n}} && \text{❹}
\end{aligned}$$

Beweis 13.1

Hinweise:

❶ Satz von Euler (für $ggT(x, n) = 1$!!!)[1]

❷ weil $n = p \cdot q \quad \Rightarrow \quad \phi(n) = (p-1)(q-1)$

❸ Anwendung der Modular-Arithmetik

❹ Potenzregel $x^u \cdot x = x^{u+1}$

RSA Schlüsselpaar

Zur Realisierung des RSA-Algorithmus wählt man zwei natürliche Zahlen e und d, sodaß gilt[2]:

$$e \cdot d \equiv 1 \pmod{\phi(n)}$$

Die natürlichen Zahlen e und d werden beim RSA-Algorithmus als Schlüsselpaar, d. h. als öffentlichen bzw. privaten Schlüssel, verwendet.

Korrektheitsbeweis

Sei $e \cdot d \equiv 1 \pmod{\phi(n)}$. Dann gilt für jede natürliche Zahl $x \leq n$ und jede natürliche Zahl k:

$$(x^e)^d \equiv \left(x^d\right)^e \equiv x \pmod{n}$$

Behauptung 13.2

$$\begin{aligned}
e \cdot d &= 1 + k \cdot \phi(n) && \text{❺}\\
e \cdot d &= 1 + k \cdot (p-1) \cdot (q-1) && \text{❻}\\
(x^e)^d &\equiv \left(x^d\right)^e \equiv x^{e \cdot d} \equiv\\
&\equiv x^{1+k \cdot (p-1) \cdot (q-1)} \equiv \underline{x \pmod{n}} && \text{❼}
\end{aligned}$$

Beweis 13.2

Hinweise:

❺ Äquivalenz-Eigenschaft der Modulo-Arithmetik

❻ weil $\phi(n) = (p-1)(q-1)$

❼ siehe ❹

[1] Den Beweis für den Fall $ggT(x, n) \neq 1$ finden Sie auf Seite 10

[2] d ist das inverse Element von e und umgekehrt

13.1.2 RSA Algorithmus

RSA Schlüsselerzeugung

Folgende Schritte sind für die Berechnung des öffentlichen Schlüssels e und privaten Schlüssels d für das RSA-Kryptosystem notwendig:

① Wähle zwei große Primzahlen p und q

② Berechne das Produkt $n = p \cdot q$

③ Berechne $\phi(n) = (p-1) \cdot (q-1)$

④ Wähle e: $1 < e < \phi(n)$ mit $ggT(e, \phi(n)) = 1$

⑤ Berechne d mit $e \cdot d \equiv 1 \pmod{\phi(n)}$

Algorithmus 13.1

RSA-Algorithmus [26]

Ver- und Entschlüsselung

Wählt man e als öffentlichen, d als privaten Schlüssel dann gilt für einen Klartext x und den Geheimtext y:

Verschlüsselung

$$y \equiv x^e \pmod{n}$$

Entschlüsselung

$$x \equiv y^d \pmod{n}$$

Rechenbeispiel

Beispiel 13.1: RSA-Algorithmus

Alice möchte eine verschlüsselte Nachricht an Bob senden. Bob berechnet zuerst seine RSA-Parameter in den Schritten 1–5. Dann sendet er Alice seinen öffentlichen Schlüssel. Alice verschlüsselt die Nachricht ($x = 4$) und sendet das Chiffrat y an Bob. Bob entschlüsselt daraufhin y unter Verwendung seines privaten Schlüssels.

a RSA-Parameter von Bob (Schlüsselgenerierung in 5 Schritten).

b Verschlüsselung von Alice der Nachricht $x = 4$.

c Entschlüsselung von Bob der Nachricht y.

Lösung 13.1:

a **Bob**: Schlüsselgenerierung in 5 Schritten.

① Wähle Primzahlen: $p = 3$ und $q = 11$

② Berechne $n = p \cdot q = 3 \cdot 11 = 33$

③ Berechne $\phi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$

④ Wähle $e = 3$:

⑤ Berechne $d \equiv e^{-1} \equiv 7 \pmod{20}$

Lösung 13.1:

b **Alice**: Verschlüsselung des Klartextes $x = 4$.

$$y = x^e \equiv 4^3 \equiv 31 \pmod{33}$$

$$y = \underline{31}$$

Lösung 13.1:

b **Bob**: Entschlüsselung des Geheimtextes $y = 31$.

$$x = y^d \equiv 31^7 \equiv 4 \pmod{33}$$

$$x = \underline{4}$$

13.2 RSA-Verschlüsselung

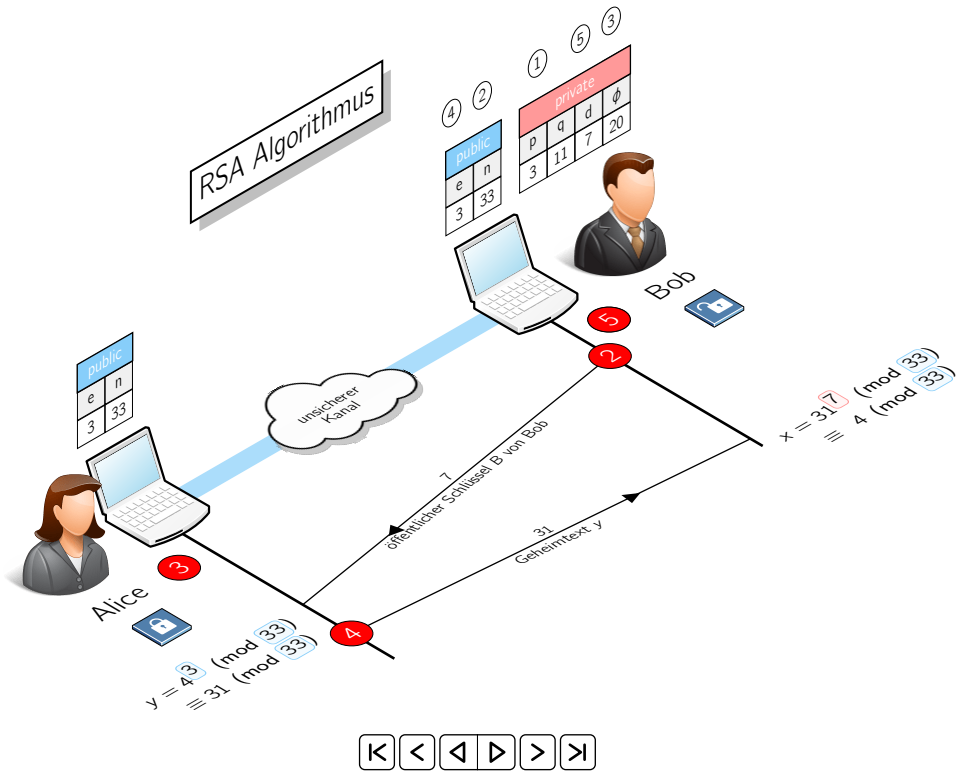


Abb. 13.1: *RSA Algorithmus*

13.2.1 RSA Algorithmus

RSA-Algorithmus [25]

1. **Bob**: 5 Schritte der Schlüsselgenerierung
   - ① Wähle Primzahlen: $p = 3$ und $q = 11$
   - ② Berechne $n = p \cdot q = 3 \cdot 11 = 33$
   - ③ Berechne $\phi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$
   - ④ Wähle $e = 3$:
   - ⑤ Berechne $d \equiv e^{-1} \equiv 7 \pmod{20}$
2. **Bob→Alice**: Schlüsselverteilung
   Bob sendet $k_{pub} = (33, 3)$ unverschlüsselt an Alice
3. **Alice**: Verschlüsseln der Nachricht $x = 4$
   Mit k_{pub}: $y \equiv 4^3 \equiv 31 \pmod{33}$.
4. **Alice→Bob**: Verschlüsselte Nachricht
   Senden des Geheimtextes 31
5. **Bob**: Entschlüsseln des Geheimtextes: $y = 31$
   Mit k_{priv}: $x \equiv 31^7 \equiv 4 \pmod{33}$.

14

Kryptosysteme basierend auf DL

Inhaltsangabe

14.1 Mathematische Grundlagen

14.1.1 Gruppen

Gruppen-Eigenschaft

Wir haben in Kapitel 2.3 verschiedene Strukturen der Algebra kennengelernt. In der Kryptographie wird vor allem mit endlichen Gruppen gearbeitet.

Multiplikative Gruppen

Für die Zahlenmengen $\mathbb{Q}$ und $\mathbb{R}$ haben wir die Menge ohne Null mit einem Stern, d. h. mit $\mathbb{Q}^*$ und $\mathbb{R}^*$, gekennzeichnet.

Es ist sinnvoll diese Schreibweise auch für endliche Mengen zu verwenden, aus der alle Elemente die keine Inverse besitzen, entfernt wurden.

> Die Menge $\mathbb{Z}_n^*$ aus den ganzen Zahlen $i = 1, \ldots, n-1$ mit $ggT(i, n) = 1$, bildet eine abelsche Gruppe mit der Gruppenoperation Multiplikation modulo n. Die Identität ist $e = 1$.
>
> *Satz 14.1*

Gruppe $\mathbb{Z}_5^*$ [27]

Entfernt man aus $\mathbb{Z}_9$ alle (Zeilen und Spalten der) nicht invertierbaren Elemente so erhhält man $\mathbb{Z}_9^*$:

$(\mathbb{Z}_9, \odot)$	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	2	4	6	8	1	3	5	7
3	3	6	0	3	6	0	3	6
4	4	8	3	7	2	6	1	5
5	5	1	6	2	7	3	8	4
6	6	3	0	6	3	0	6	3
7	7	5	3	1	8	6	4	2
8	8	7	6	5	4	3	2	1

Tab. 14.1: *Menge* $\mathbb{Z}_9$

$(\mathbb{Z}_9^*, \odot)$	1	2	4	5	7	8
1	1	2	4	5	7	8
2	2	4	8	1	5	7
4	4	8	7	2	1	5
5	5	1	2	7	8	4
7	7	5	1	8	4	2
8	8	7	5	4	2	1

Tab. 14.2: *Menge* $\mathbb{Z}_9^*$

> Alle Mengen $\mathbb{Z}_p$ mit $p \in \mathbb{P}$ (Primzahl) bilden immer eine abelsche Gruppe.
>
> *Beobachtung 14.1*

Ordnung einer Gruppe

In der Kryptographie werden vor allem mit endliche Mengen gearbeitet.

> Eine Gruppe $(G, \circ)$ ist **endlich**, wenn sie eine endliche Anzahl an Elementen hat.
>
> *Definition 14.1*

Für endliche Mengen gilt für die Anzahl n ihrer Elemente immer $n \in \mathbb{N}$.

> Die **Ordnung**[1] $|G|$ einer **Gruppe** G bezeichnet die Anzahl n ihrer Elemente.
>
> *Definition 14.2*

> Aus $\mathbb{Z}_p^* = \mathbb{Z}_p \backslash \{0\}$ folgt immer $|\mathbb{Z}_p^*| = |\mathbb{Z}_p| - 1$, d. h. $\mathbb{Z}_p^*$ hat ein Element weniger als $\mathbb{Z}_p$.
>
> *Beobachtung 14.2*

Zwischen $\mathbb{Z}_n^*$ und $\phi(n)$ gilt folgender Zusammenhang:

$$|\mathbb{Z}_n^*| = \phi(n) \quad \text{bzw.} \quad |\mathbb{Z}_p^*| = p - 1$$

[1] oder auch **Kardinalität**

Ordnung eines Elements

> Die **Ordnung** $ord(a)$ eines **Elements** a einer Gruppe $(G, \circ)$ ist die kleinste positive ganze Zahl k mit
>
> $$a^k = \underbrace{a \circ a \circ \cdots \circ a}_{k-mal} = 1$$
>
> wobei 1 die Identiät von G ist.
>
> *Definition 14.3*

Beispiel

$(\mathbb{Z}_{11}^*, \odot)$	1	2	3	4	5	6	7	8	9	10
1	1	2	4	3	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

Tab. 14.3: *Multiplikationstabelle für* $\mathbb{Z}_{11}^*$

Beispiel 14.1: Potenzen von 3

Gegeben ist die mulitiplikative Gruppe $(\mathbb{Z}_{11}^*, \odot)$ und ein Element $a = 3$ dieser Gruppe.

a) Berechne alle Potenzen von a^1 bis a^5.

b) Was passiert wenn wir die Berechnung fortsetzen?

c) Berechne die Ordnung von a.

Beispiel I [25]

Lösung 14.1:

a) Berechne alle Potenzen von a^1 bis a^5.

$a^1 = 3$

$a^2 = a \cdot a = 3 \cdot 3 = 9$

$a^3 = a^2 \cdot a = 9 \cdot 3 \equiv 5 \pmod{11}$

$a^4 = a^3 \cdot a = 5 \cdot 3 \equiv 4 \pmod{11}$

$a^5 = a^4 \cdot a = 4 \cdot 3 \equiv 1 \pmod{11}$

b) Was passiert wenn wir die Berechnung fortsetzen?

$a^6 = 3$

$a^7 = a^6 \cdot a = 3 \cdot 3 = 9$

$a^8 = a^7 \cdot a = 9 \cdot 3 \equiv 5 \pmod{11}$

$a^9 = a^8 \cdot a = 5 \cdot 3 \equiv 4 \pmod{11}$

$a^{10} = a^9 \cdot a = 4 \cdot 3 \equiv 1 \pmod{11}$

c) Berechne die Ordnung von a.

Aus der vorletzten Zeile folgt: $a^5 \Longrightarrow k = 5 \Longrightarrow ord(3) = 5$.

> Bei Berechnung der Potenzen von 3 werden in den Ergebnissen **nicht alle** Elemente der Gruppe erfasst (Zykluslänge 5).
>
> *Beobachtung 14.3*

Beispiel 14.2: Potenzen von 2

Gegeben ist die mulitiplikative Gruppe $(\mathbb{Z}_{11}^*, \odot)$ und ein Element $a = 2$ dieser Gruppe.

a) Berechne alle Potenzen von a.

Beispiel II [25]

Lösung 14.2:

a) Berechne alle Potenzen von a.

$a^1 = 2$

$a^2 = a \cdot a = 2 \cdot 2 = 4$

$a^3 = a^2 \cdot a = 4 \cdot 2 = 8$

$a^4 = a^3 \cdot a = 8 \cdot 2 \equiv 5 \pmod{11}$

$a^5 = a^4 \cdot a = 5 \cdot 2 = 10 \pmod{11}$

$a^6 = a^5 \cdot a = 10 \cdot 2 \equiv 9 \pmod{11}$

$a^7 = a^6 \cdot a = 9 \cdot 2 \equiv 7 \pmod{11}$

$a^8 = a^7 \cdot a = 7 \cdot 2 \equiv 3 \pmod{11}$

$a^9 = a^8 \cdot a = 3 \cdot 3 \equiv 6 \pmod{11}$

$a^{10} = a^9 \cdot a = 6 \cdot 3 \equiv 1 \pmod{11}$

> Bei Potenzen von 2 werden **alle** Elemente der Gruppe erzeugt (Zykluslänge 10).
>
> *Beobachtung 14.4*

14.1.2 Zyklische Gruppen

Definition

Zyklische Gruppen [25]

Eine Gruppe $(G, \circ)$ mit mindestens ein Element α mit maximalen Ordnung $ord(\alpha) = |G|$, nennt man **zyklisch**[2].

Definition 14.4

Primitive Elemente (Generatoren)

Alle Elemente mit maximaler Ordnung nennt man **primitive Elemente**, **Generatoren** oder **Erzeuger**.

Definition 14.5

Eigenschaften zyklischer Gruppen

Die für kryptografischen Anwendungen wichtigste Eigenschaften sind:

Für jede Primzahl p ist $(\mathbb{Z}_p^*, \circ)$ eine endliche abelsche **zyklische Gruppe**.

Satz 14.2

Daraus folgt: **jede** multiplikative Gruppe eines Primzahlkörpers $\mathbb{Z}_p^*$ ist zyklisch.

Eigenschafteb [25]

Für eine endliche Gruppe G gilt für jedes $a \in G$:

- $a^{|G|} = 1$
- $ord(a)$ ist Teiler von $|G|$

Satz 14.3

Da gilt $ord(a)$ ist Teiler von $|G|$ folgt daraus immer für jedes Element $a^{|G|} = 1$.

Beobachtung 14.5

Beispiel I [25]

Beispiel 14.3: Potenzen von 3

Gegeben ist die mulitiplikative Gruppe $(\mathbb{Z}_{11}^*, \odot)$. Berechne für folgende Elemente $a \in 1,2,3$ die Zyklus-Elemente a_i und überprüfe damit obige Aussage.

a) $a = 1$ b) $a = 2$ c) $a = 3$

Lösung 14.3:

a) $a = 1$

$1, 1, 1, 1, 1, 1, 1, 1, 1, 1$

b) $a = 2$

$\underbrace{2, 4, 8, 5, 10, 9, 7, 3, 6, 1}_{\text{Zyklus } 10}$

c) $a = 3$

$\underbrace{3, 9, 5, 4, 1}_{\text{Zyklus } 5}, \underbrace{3, 9, 5, 4, 1}_{\text{Zyklus } 5}$

a	$ord(a)$	Zyklus	Erzeuger
1	1	1	
2	10	2, 4, 8, 5, 10, 9, 7, 3, 6, 1	✓
3	5	3, 9, 5, 4, 1	
4	5	4, 5, 9, 3, 1	
5	5	5, 3, 4, 9, 1	
6	10	6, 3, 7, 9, 10, 5, 8, 4, 2, 1	✓
7	10	7, 5, 2, 3, 10, 4, 6, 9, 8, 1	✓
8	10	8, 9, 6, 4, 10, 3, 2, 5, 7, 1	✓
9	5	9, 4, 3, 5, 1	
10	2	10, 1,	

Tab. 14.4: *Die Ordnung aller Elemente von* $\mathbb{Z}_{11}^*$

Beispiel I [25]

Die Ordnung der Gruppe $(\mathbb{Z}_{11}^*, \odot)$ ist $|G| = 10$. Mögliche Teiler von 10 sind {1,2,5,10} und entspricht den möglichen Werten von $ord(a)$.

Beobachtung 14.6

[2] **lateinisch:** *cyclus* = Kreis, kreisförmig (wiederholend)

14.1.3 Das diskrete Logarithmusproblem

Definition

Das **DLP**[3], als Basis für das DHKE-Protokoll, kann direkt mithilfe von zyklischen Gruppen erklärt werden.

DLP [28]

Gegeben sind die endliche zyklische Gruppe $\mathbb{Z}_p^*$ der Ordnung $p-1$, ein primitives Element $\alpha \in \mathbb{Z}_p^*$ und ein weiteres Element $\beta \in \mathbb{Z}_p^*$. Das **DLP** ist:

Geg.: α, β und p
Ges.: $x \in \mathbb{Z}_p^*$ mit $1 \leq x \leq p-1$, sodass:

$$\alpha^x \equiv \beta \pmod{p}$$

Definition 14.6

Die Schulmathematik würde eine Gleichung mit der gesuchten Variablen im Exponenten wie folgt lösen:

$$\alpha^x \equiv \beta \pmod{p} \quad \Big| \ \log_\alpha()$$
$$x \equiv \log_\alpha(\beta) \pmod{p}$$

In $\mathbb{Z}_{11}^*$ gibt es aber nur ganze Zahlen.

In der Reihenfolge der Zyklus-Elemente ist keine Regelmäßigkeit erkennbar (scheint zufällig). Es ist praktisch unmöglich eine Lösung des diskreten Logarithmusproblem mit bekannten Methoden der Mathematik zu berechnen.

Beobachtung 14.7

Beispiel

Beispiel 14.4: Das diskrete Logarithumusproblem

Gegeben sind die endliche zyklische Gruppe $\mathbb{Z}_{47}^*$, ein primitives Element $\alpha = 5$ und $\beta = 41$.

a) $x \in \mathbb{Z}_{47}^*$ mit $1 \leq x \leq 46$, sodass gilt:
$5^x \equiv 41 \pmod{p}$

Selbst für kleine Zahlen ist die des diskreten Logarithmus in $\mathbb{Z}_{47}^*$ nicht einfach.

Lösung 14.4:

a) $x \in \mathbb{Z}_{47}^*$ mit $1 \leq x \leq 46$, sodass gilt:
$5^x \equiv 41 \pmod{p}$

Durch Ausprobieren aller Werte erhalten wir als Lösung $x =$ 15.

Beispiel

Im nachfolgenden Beispiel wird für α ein nicht primitives Element von $\mathbb{Z}_{47}^*$ ausgewählt. Deshalb ist es hier notwendig die entsprechende Untergruppe von $\mathbb{Z}_{47}^*$ zu betrachten.

Beispiel 14.5: Das diskrete Logarithumusproblem

Gegeben sind die endliche zyklische Gruppe $\mathbb{Z}_{47}^*$, ein primitives Element $\alpha = 2$ der Untergruppe mit 23 Elementen und $\beta = 36$.

a) $x \in \mathbb{Z}_{47}^*$ mit $1 \leq x \leq 46$, sodass gilt:
$2^x \equiv 36 \pmod{p}$

Lösung 14.5:

a) $x \in \mathbb{Z}_{47}^*$ mit $1 \leq x \leq 46$, sodass gilt:
$2^x \equiv 36 \pmod{p}$

Durch Ausprobieren aller Werte erhalten wir als Lösung $x =$ 17.

Es ist möglich das DLP auf auf andere algebraische Strukturen zu erweiteren. Dazu mehr im Kapitel 15.

[3] **englisch:** DLP = Discrete-Logarithm-Problem (Diskrete LogarithmusProblem)

14.1.4 DHKE Algorithmus

Das Problem der klassischen Kryptographie ist die (kryptografisch unsichere) Weitergabe des geheimen Schlüssels an den Kommunikationspartner.

Mit der Methode von Diffie-Hellman war es erstmals möglich über einen nicht-geheimen, öffentlichen Kanal ein Geheimnis zu übertragen. [2, S 42]

Dabei handelt es sich eigentlich um eine **Schlüsselvereinbarung** und nicht um einen Schlüsseltausch, da beide Parteien an der Erzeugung des geheimen Wertes $k_(A,B)$ beteiligt sind.

Beobachtung 14.8

Diffie-Hellman-Schlüsselaustausch

Der **DHKE**[4] besteht aus zwei Protokollen:

- **Set-up**

  ① Wähle eine große Primzahl p
  ② Wähle eine ganze Zahl α mit $\alpha \in \{2,3,\ldots,p\,1\}$
  ③ Veröffentliche p und α

  Algorithmus 14.7

- **Schlüsselaustausch**

  ④ Wähle privaten Schlüssel $\tilde{a}$ von A
  ⑤ Berechne öffentl. Schlüssel $a = \alpha^{\tilde{a}} \pmod p$ von A
  ⑥ Wähle privaten Schlüssel $\tilde{b}$ von B
  ⑦ Berechne öffentl. Schlüssel $b = \alpha^{\tilde{b}} \pmod p$ von B
  ⑧ A sendet a an B
  ⑨ B sendet b an A
  ⑩ Berechne Sitzungsschlüssel
  A: $k_{AB} = b^{\tilde{a}} \pmod p$
  B: $k_{AB} = a^{\tilde{b}} \pmod p$

  Algorithmus 14.8

DHKE [27]

Korrektheitsbeweis

Teilnehmer A und B berechnen den gleichen Sitzungsschlüssel, d. h. es gilt:

$$b^{\tilde{a}} \equiv a^{\tilde{b}} \pmod p$$

Behauptung 14.4

$$b^{\tilde{a}} \equiv a^{\tilde{b}} \pmod p$$
$$\left(\alpha^{\tilde{b}}\right)^{\tilde{a}} \equiv \left(\alpha^{\tilde{a}}\right)^{\tilde{b}}$$
$$\alpha^{\tilde{a}\cdot\tilde{b}} \equiv \alpha^{\tilde{a}\cdot\tilde{b}} \pmod p$$

Beweis 14.1

Korrektheitsbeweis [27]

Berechnung des geheimen Schlüssels

Teilnehmer A und B berechnen unabhängig voneinander den gleichen, geheimen Sitzungsschlüssel wie folgt (Schlüsselvereinbarung):

Teilnehmer A	**Teilnehmer B**
$k_{AB} = b^{\tilde{a}} \pmod p$	$k_{AB} = a^{\tilde{b}} \pmod p$

Anmerkungen

- Die Berechnungen, die für den DHKE notwendig sind, ähneln stark denen von RSA.
- Die Zahl α muss eine spezielle Eigenschaft erfüllen: Sie muss ein sog. primitives Element sein.
- Die Wahl der privaten Schlüssel $\tilde{a}$ und $\tilde{b}$ sollten aus einem echten Zufallszahlengenerator stammen, um zu verhindern, dass ein Angreifer diese erraten kann.
- Die öffentlichen Schlüssel werden üblicherweise vorausberechnet. Während eines Schlüsselaustauschs besteht der Hauptrechenaufwand daher aus der Exponentiation für den Sitzungsschlüssel.

[3, S 238]

Das Protokoll kann verallgemeinert werden, insbesondere auf Gruppen von Punkten auf elliptischen Kurven. Man spricht hier von **ECC**[5]

Beobachtung 14.9

Rechenbeispiel

Beispiel 14.6: DHKE Protokoll

Gegeben sind die Diffie-Hellman-Domain-Parameter sind $p = 29$ und $\alpha = 2$. .

a Berechne für Bob und Alice einen geheimen Sitzungschlüssel mit Hilfe des DHKE-Protokolls.

Lösung 14.6:

a Berechne für Bob und Alice einen geheimen Sitzungschlüssel mit Hilfe des DHKE-Protokolls.

① **Alice** wählt privaten Schlüssel:
$\tilde{a} = k_{pr,A} = 5$
② **Alice** berechnet öffentlichen Schlüssel:
$a = k_{pub,A} = 2^5 \equiv 3 \pmod{29}$
③ **Bob** wählt privaten Schlüssel:
$\tilde{b} = k_{pr,B} = 12$
④ **Bob** berechnet öffentlichen Schlüssel:
$b = k_{pub,B} = 2^{12} \equiv 7 \pmod{29}$
⑤ **Alice→Bob**: öffentlichen Schlüssel von Alice
⑥ **Bob→Alice**: öffentlichen Schlüssel von Bob
⑦ **Alice** $k_{A,B} = b^{\tilde{a}} = 7^5 \equiv 16 \pmod{29}$
⑧ **Bob** $k_{A,B} = a^{\tilde{b}} = 3^{12} \equiv 16 \pmod{29}$

[4] **englisch:** *DHKE* = Diffie-Hellman-Key-Exchange (Diffie-Hellman-Schlüsselaustausch)

[5] **englisch:** *ECC* = Elliptic-Curve-Cryptography (Elliptische-Kurven-Kryptografie)

14.2 Diffie-Hellman-Schlüsselaustausch

Der DHKE wurde von Whitfield Diffie und Martin Hellman im Jahr 1976 vorgestellt und war das erste veröffentlichte asymmetrische Verfahren überhaupt.

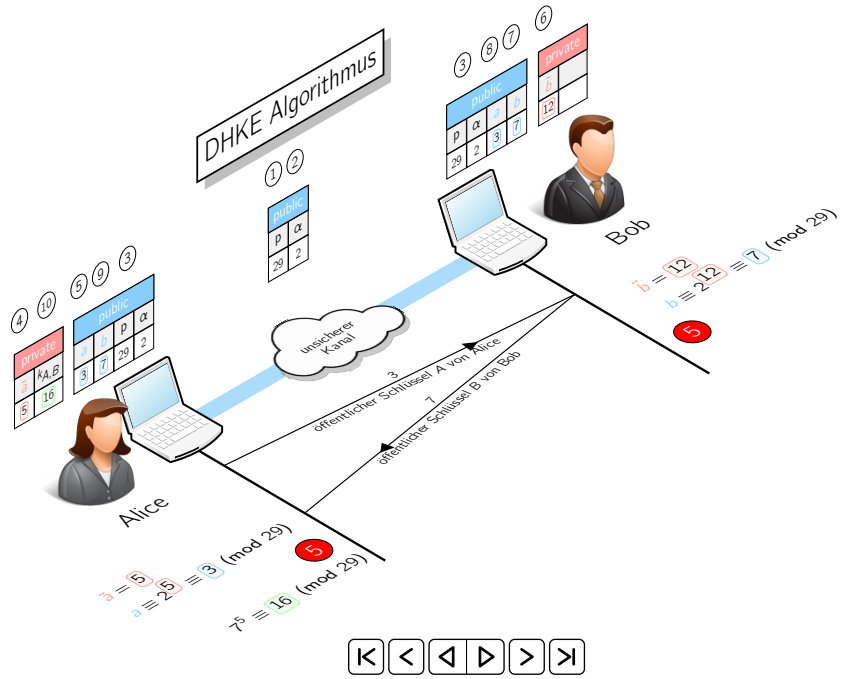


Abb. 14.1: *Diffie-Hellman-Schlüsselaustausch*

14.2.1 DHKE Algorithmus

- **Diffie-Hellman-Set-up**
  - 1 **Domain**: Parametergenerierung
    - ① Wähle eine große Primzahl:
      $p = 29$
    - ② Wähle eine ganze Zahl
      $\alpha = 2$ mit $\alpha \in \{2, 3, \ldots, p-1\}$
    - ③ Veröffentliche p und α
- **Diffie-Hellman-Schlüsselaustausch**
  - 2 **Alice**: Schlüsselgenerierung
    - ④ **Alice** Wähle privaten Schlüssel:
      $\tilde{a} = k_{pr,A} = 5$
    - ⑤ **Alice** Berechne öffentlichen Schlüssel:
      $a = k_{pub,A} = 2^5 \equiv 3 \pmod{29}$
  - 3 **Bob**: Schlüsselgenerierung
    - ⑥ **Bob** Wähle privaten Schlüssel:
      $\tilde{b} = k_{pr,B} = 12$
    - ⑦ **Bob** Berechne öffentlichen Schlüssel:
      $b = k_{pub,B} = 2^{12} \equiv 7 \pmod{29}$
  - 4 **Alice→Bob**: Schlüsselaustausch
    - ⑧ **Alice→Bob**: öffentlichen Schlüssel von Alice
    - ⑨ **Bob→Alice**: öffentlichen Schlüssel von Bob
  - 5 **Geheimer Sitzungsschlüssel**
    - ⑩ **Alice** $k_{A,B} = b^{\tilde{a}} = 7^5 \equiv 16 \pmod{29}$
    - ⑪ **Bob** $k_{A,B} = a^{\tilde{b}} = 3^{12} \equiv 16 \pmod{29}$

15

Kryptosyteme mit elliptische Kurven

Inhaltsangabe

15.1 Mathematische Grundlagen

15.1.1 Elliptische Kurven

Einführung EC [29]

Eine **elliptische Kurve** ist eine spezielle Polynomgleichung. Um sie in der Kryptografie anwenden zu können, muss das Polynom anstatt über den reellen Zahlen, über einem endlichen Körper betrachtet werden.

Definition

In der Praxis werden am häufigsten elliptische Kurven über Primkörpern benutzt, d. h. alle Berechnungen werden modulo p durchgeführt.

Die **elliptische Kurve** über $\mathbb{Z}_p$, $p > 3$, ist die Menge der Punkte (x, y) mit $x, y \in \mathbb{Z}_p$, die die folgende Gleichung erfüllen:

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

wobei folgende Bedingungen gelten müssen:

$$a, b \in \mathbb{Z}_p \qquad \text{und} \qquad 4 \cdot a^3 + 27 \cdot b^2 \neq 0$$

Zu der elliptischen Kurve gehört des Weiteren auch der imaginäre **Punkt im Unendlichen** $\mathbb{O}$

Definition 15.1

Elliptische Kurve als Gruppe

EC als Gruppe [29]

Um elliptische Kurven in der Kryptografie zu verwenden ist es notwendig darauf eine zyklische Gruppe als algebraische Struktur abzubilden, d. h. man benötigt:

- **Gruppenelemente**: Punkte auf der Kurve $P = (x_1, y_1)$, $Q = (x_2, y_2)$, $R = (x_3, y_3)$,
- **Gruppenoperation**: Addition zweier Punkte $P \oplus Q = R$ bzw. $P \oplus P = 2 \cdot P = R$

Gruppenoperation

Gruppenperationen [29]

Als Gruppenoperation wird als Symbol $\oplus$ verwendet und ist geometrisch wie folgt definiert:

Punktaddition

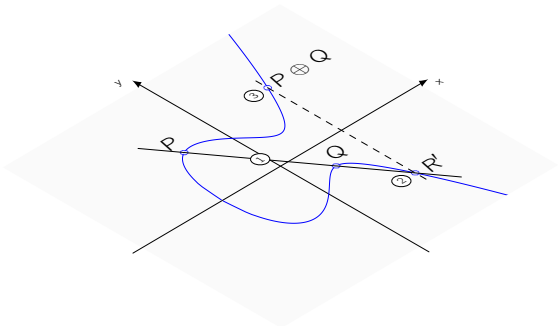


Abb. 15.1: *Punktaddition*

Die **Punktaddition** $P \oplus Q$ zweier ungleichen Punkte P und Q ist geometrisch wie folgt definiert:

1. Gerade durch die Punkte P und Q.
2. Die Gerade schneidet die elliptische Kurve in einen dritten Punkt R'.
3. Durch Spiegelung von R' erhält man den Punkt R als Ergebnis der Addition.

Definition 15.2

Punktverdopplung

Sind beide Punkte ident, dann spricht man von einer Punktverdopplung und diese ist wie folgt definiert:

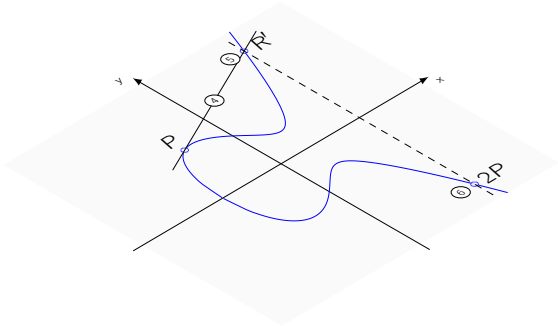


Abb. 15.2: *Punktverdopplung*

Die **Punktverdopplung** $P \oplus P = 2P$ von P ist geometrisch wie folgt definiert:

4. Tangente durch den Punkt P.
5. Die Tangente schneidet die elliptische Kurve in einen zweiten Punkt R'.
6. Durch Spiegelung von R' erhält man den Punkt R als Ergebnis der Verdopplung.

Definition 15.3

Verdopplung [29]

Neutrales Element $\mathcal{O}$

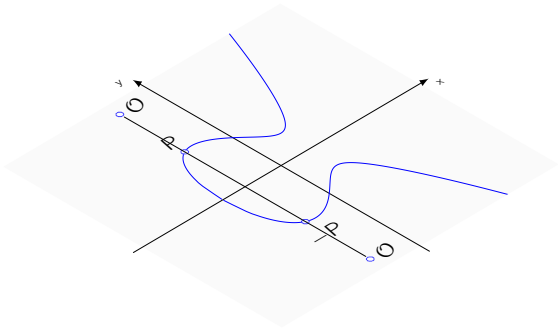


Abb. 15.3: *Neutrales Element*

Um die Gruppeneigenschaft zu vervollständigen benötigt man noch ein neutrales Element $\mathcal{O}$. Es muss für alle Punkte P auf der Kurve folgende Eigenschaft erfüllen:

$$P \oplus \mathcal{O} = P$$

Neutrales Element [29]

Kein geometrischer Punkt auf der elliptischen Kurve erfüllt diese Eigenschaft. Deshalb gilt:

Das **neutrale Element** $\mathcal{O}$ wird durch einen imaginären Punkt im Unendlichen definiert.

Definition 15.4

Inverse Element

Das **inverse Element** $-P$ wird durch die Spiegelung von P an der x-Achse bestimmt und es gilt:

$$P \oplus (-P) = \mathcal{O}$$

Definition 15.5

15.1.2 Berechnung der Koordinaten

Die Gruppenoperationen werden natürlich nicht geometrisch konstruiert sondern wie folgt berechnet:

x- und y-Koordinate

$$x_3 \equiv s^2 - x_1 - x_2 \pmod p \qquad y_3 \equiv s \cdot (x_1 - x_3) - y_1 \pmod p$$

Für den Parameter s (Steigung) gilt:

falls $P \neq Q$ (Punktaddition)

$$s \equiv \frac{y_2 - y_1}{x_2 - x_1} \pmod p$$

falls $P = Q$ (Punktverdopplung)

$$s \equiv \frac{3x_1^2 + a}{2y_1} \pmod p$$

Beweis für die Formeln

Die Koordinaten $R = (x_3, y_3)$ berechnet sich wie folgt:

$$x_3 \equiv s^2 - x_1 - x_2$$
$$y_3 \equiv s \cdot (x_1 - x_3) - y_1$$
$$\text{wobei gilt } s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{falls } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{falls } P = Q \end{cases}$$

Behauptung 15.1

Beweis für $P \neq Q$:

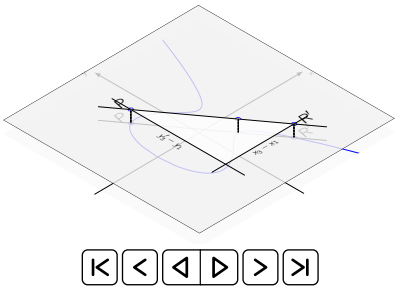


Abb. 15.4: *Beweis für $P \neq Q$*

$$y = s \cdot x + d \quad \text{mit } s = \frac{y_2 - y_1}{x_2 - x_1} \qquad (1)$$
$$y^2 = x^3 + a \cdot x + b$$
$$\Rightarrow \quad (s \cdot x + d)^2 = x^3 + b \cdot x + c \qquad (2)$$
$$0 = x^3 - s^2 \cdot x^2 - 2 \cdot s \cdot x \cdot d + b \cdot x + c - d^2 \qquad (3)$$
$$0 = (x - x_1)(x - x_2)(x - x_3) \qquad (4)$$
$$0 = x^3 - (x_1 + x_2 + x_3)x^2 + \ldots$$
$$\ldots + (x_1 \cdot x_2 + x_1 \cdot x_3 + x_2 \cdot x_3)x - x_1 \cdot x_2 \cdot x_3$$
$$s^2 = x_1 + x_2 + x_3 \Rightarrow x_3 = s^2 - x_1 - x_2 \qquad (5)$$
$$s = \frac{y_3' - y_1}{x_3 - x_1} \Rightarrow y_3' = s \cdot (x_3 - x_1) + y_1 \qquad (6)$$
$$\text{Spiegelung} : \Rightarrow y_3 = -y_3' = s \cdot (x_1 - x_3) - y_1 \qquad (7)$$

Beweis 15.1

Hinweise:

1. Geradengleichung der Sekante mit der Steigung s
2. Geradengleichung in elliptische Kurve eingesetzt
3. Null setzen durch Umformung
4. Gegebene Punkte x_1, x_2, x_3 müssen Gleichung erfüllen
5. Ausmultiplizieren und Lösen durch Koeffizientenvergleich
6. Steigungsdreieck gilt auch für Punkte P und R'
7. Gesuchter Punkt ist Spiegelung an der x-Achse $y_3 = -y_3'$

Beweis für $P = Q$:

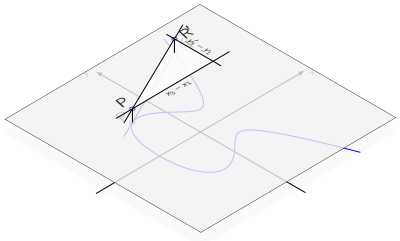


Abb. 15.5: *Beweis für $P \neq= Q$*

$$y^2 = x^3 + a \cdot x + b \qquad (8)$$
$$2 \cdot y \cdot y' = 3 \cdot x^2 + b \Rightarrow y' = s = \frac{3 \cdot x^2 + b}{2 \cdot y} \ldots \qquad (9)$$
$$s^2 = x_1 + x_2 + x_3 \Rightarrow x_3 = s^2 - x_1 - x_2$$
$$s = \frac{y_3' - y_1}{x_3 - x_1} \Rightarrow y_3' = s \cdot (x_3 - x_1) + y_1$$
$$\text{Spiegelung} : \Rightarrow y_3 = -y_3' = s \cdot (x_1 - x_3) - y_1$$

Beweis 15.2

Hinweise:

8. Tangentensteigung s durch implizite Ableitung
9. es folgt Beweiskette analog zum Fall $P \neq Q$

15.1.3 Rechenbeispiel

Tabelle aller Inversen (siehe Listing 17.33):

e	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
e^{-1}	9	6	13	7	3	5	15	2	12	14	10	4	11	8	16

Tab. 15.1: **Hilfstabelle**: *Die Inversen in $\mathbb{Z}_{17}^*$*

Beispiel 15.1: DHKE Protokoll

a) Bestimme alle Gruppenelemente der elliptischen Kurve

$$E : y^2 \equiv x^3 + 2 \cdot x + 2 \pmod{17} \quad \text{und } P = (5,1)$$

Beispiel [29]

Lösung 15.1:

a) Bestimme alle Gruppenelemente der elliptischen Kurve

$$E : y^2 \equiv x^3 + 2 \cdot x + 2 \pmod{17} \quad \text{und } P = (5,1)$$

$$P = (5,1) \qquad \Rightarrow s \equiv (3 \cdot 5^2 + 2)(2)^{-1} \equiv 9 \cdot 9 \equiv 13$$
$$x_3 \equiv 13^2 - 5 - 5 \equiv 159 \equiv 6$$
$$y_3 \equiv 13\,(5 - 6) - 1 \equiv 4 - 1 \equiv 3$$

(Berechnung aller Gruppen-Elemente siehe Listing 17.36):

Beispiel [30]

1P	2P	3P	4P	5P	6P	7P	8P	9P	10P
(5,1)	(6,3)	(10,6)	(3,1)	(9,16)	(16,13)	(0,6)	(13,7)	(7,6)	(7,11)
11P	12P	13P	14P	15P	16P	17P	18P	19P	
(13,10)	(0,11)	(16,4)	(9,1)	(3,16)	(10,11)	(6,14)	(5,16)	$\mathcal{O}$	

Tab. 15.2: *Zyklus mit Erzeuger $P = (5,1)$*

Beispiel 15.2: DHKE Protokoll

a) Bestimme alle Gruppenelemente der elliptischen Kurve

$$E : y^2 \equiv x^3 + 2 \cdot x + 2 \pmod{17} \quad \text{und } P = (13,7)$$

Berechnung aller Gruppen-Elemente (siehe Listing 17.36):

Lösung 15.2:

a) Bestimme alle Gruppenelemente der elliptischen Kurve

$$E : y^2 \equiv x^3 + 2 \cdot x + 2 \pmod{17} \quad \text{und } P = (13,7)$$

$$P = (13,7) \qquad \Rightarrow s \equiv (3 \cdot 13^2 + 2)(14)^{-1} \equiv 16 \cdot 11 \equiv 6$$
$$x_3 \equiv 6^2 - 13 - 13 \equiv 10 \equiv 10$$
$$y_3 \equiv 6\,(13 - 10) - 7 \equiv 18 - 7 \equiv 11$$

(Berechnung aller Gruppen-Elemente siehe Listing 17.36):

1P	2P	3P	4P	5P	6P	7P	8P	9P	10P
(13, 7)	(10, 11)	(9, 16)	(16, 4)	(6, 3)	(7, 11)	(5, 16)	(0, 6)	(3, 16)	(3, 1)
11P	12P	13P	14P	15P	16P	17P	18P	19P	
(0, 11)	(5, 1)	(7, 6)	(6, 14)	(16, 13)	(9, 1)	(10, 6)	(13, 10)	$\mathcal{O}$	

Tab. 15.3: *Zyklus mit Erzeuger $P = (13,7)$*

15.1.4 DHKE mit elliptischen Kurven

Das DLP über elliptische Kurven

In den oben stehenden Abschnitten wurden die Gruppenoperation (Punktaddition bzw. -verdopplung) und das neutrale Element eingeführt.

Es wurde auch gezeigt, dass jedes Gruppenelement eine Inverse hat. Es gilt der folgende Satz: [3, S 280]

DLP über EC [30]

Die Punkte einer **elliptischen Kurve** zusammen mit $\mathcal{O}$ haben zyklische Untergruppen. Unter bestimmten Bedingungen bilden *alle* Punkte einer elliptischen Kurve eine **zyklische Gruppe**.

Satz 15.2

Dieser Satz wird hier ohne Beweis angegeben. Die Aussage des Satzes ist extrem hilfreich, da wir wissen, wie man mit zyklischen Gruppen Kryptoverfahren konstruiert.

Der Satz stellt sicher, dass ein primitives Element existiert, dessen Vielfache alle Gruppenelemente erzeugen.

Schlüsselaustausch mit elliptischen Kurven

Der Diffie-Hellman-Schlüsseltausch mit elliptischen Kurven (**ECDH**[1]) besteht aus zwei Protokollen:

- **ECDH-Set-up**

Beispiel [30]

① Wähle Primzahl p, eine elliptische Kurve $E : y^2 \equiv x^3 + a \cdot x + b \pmod p$

② Wähle ein primitives Element $P = (x_1, y_1)$

③ Veröffentliche: Primzahl p, Punkt P und Kurvenparameter a, b

Algorithmus 15.6

- **Schlüsselaustausch**

④ Wähle privaten Schlüssel $\tilde{a}$ von A mit $\tilde{a} \in \{2,3,\ldots,|E|-1\}$

⑤ Berechne öffentl. Schlüssel $a = \tilde{a}\ P = (x_a, y_a)$ von A

⑥ Wähle privaten Schlüssel $\tilde{b}$ von B mit $\tilde{b} \in \{2,3,\ldots,|E|-1\}$

⑦ Berechne öffentl. Schlüssel $b = \tilde{b}\ P = (x_b, y_b)$ von B

⑧ A sendet a an B

⑨ B sendet b an A

⑩ Berechne Sitzungsschlüssel
A: $T_{AB} = \tilde{a}\ b$
B: $T_{AB} = \tilde{b}\ a$

Algorithmus 15.7

Korrektheitsbeweis

Korrektheitsbeweis [30]

Teilnehmer A und B berechnen den gleichen Sitzungsschlüssel, d. h. es gilt:

$$T_{AB} = \tilde{a}\ b = \tilde{b}\ a$$

Behauptung 15.3

$$\tilde{a}\ b = b\ a$$
$$\tilde{a}\ (\tilde{b}\ P) = \tilde{b}\ (\tilde{a}\ P)$$
$$\tilde{a}\ \tilde{b}\ P = \tilde{b}\ \tilde{a}\ P$$

Beweis 15.3

Berechnung des geheimen Schlüssels

Teilnehmer A und B berechnen unabhängig voneinander den gleichen, geheimen Sitzungsschlüssel wie folgt (Schlüsselvereinbarung):

Teilnehmer A	**Teilnehmer B**
$T_{AB} = \tilde{a}\ b$	$T_{AB} = \tilde{b}\ a$

Anmerkungen

- Die Bestimmung einer geeigneten elliptischen Kurve ist rechenaufwändig. Sie muß gewisse mathematische Eigenschaften besitzen, damit das DLP sicher ist.
- Das eigentliche Schlüsselaustauschprotokoll ist nun vollkommen analog zum Diffie-Hellmann-Protokoll über endlichen Körpern.
- Die Punkteaddition ist assoziativ (Gruppeneigenschaft)
- Die öffentliche Schlüssel und das gemeinsamen Geheimnis T_{AB} sind Kurvenpunkte (geometrische Koordinaten)
- Aus dem gemeinsamen Geheimnis T_{AB} kann ein Sitzungsschlüssel abgeleitet werden (meist die x-Koordinate).

[3, S 284]

Das Protokoll kann verallgemeinert werden, insbesondere auf Gruppen von Punkten auf elliptischen Kurven. Man spricht hier von **ECC**[2]

Beobachtung 15.1

Rechenbeispiel

Beispiel 15.3: DHKE Protokoll

Geben sei ein ECDH mit den folgenden Domain-Parametern:

- **Elliptische Kurve**: $E : y^2 \equiv x^3 + 2x + 2 \pmod{17}$ (bildet eine zyklische Gruppe der Ordnung $|E| = 19$)
- **Primitives Element**: $P = (5,1)$

a Berechne für Bob und Alice einen geheimen Sitzungschlüssel mit Hilfe des ECDH-Protokolls.

Lösung 15.3:

a Berechne für Bob und Alice einen geheimen Sitzungschlüssel mit Hilfe des ECDH-Protokolls.

① **Alice** wählt privaten Schlüssel:
$\tilde{a} = k_{pr,A} = 3$

② **Alice** berechnet öffentlichen Schlüssel:
$a = k_{pub,A} = 3\ P = (10,6)$

③ **Bob** wählt privaten Schlüssel:
$\tilde{b} = k_{pr,B} = 10$

④ **Bob** berechnet öffentlichen Schlüssel:
$b = k_{pub,B} = 10\ P = (7,11)$

⑤ **Alice→Bob**: öffentlichen Schlüssel von Alice

⑥ **Bob→Alice**: öffentlichen Schlüssel von Bob

⑦ **Alice** $T_{A,B} = \tilde{a}\ b = 3\ (7,11) = 3\ 10\ P = 30\ P = (13,10)$

⑧ **Bob** $T_{A,B} = \tilde{b}\ a = 10\ (10,6) = 10\ 3\ P = 30\ P = (13,10)$
(Es gilt[3]: $30\ P \equiv 11\ P \pmod{19}$)

[1] **englisch:** *ECDH* = Elliptic-Curve-Diffie-Hellman- (Diffie-Hellman-Schlüsselaustausch)

[2] **englisch:** *ECC* = Elliptic-Curve-Cryptography (Elliptische-Kurven-Kryptografie)

[3] da Ordnung (Kardinalität) $|E| = 19$, d. h. 19 Zyklus-Elemente

15.2 Diffie-Hellman-Schlüsselaustausch mit ECC

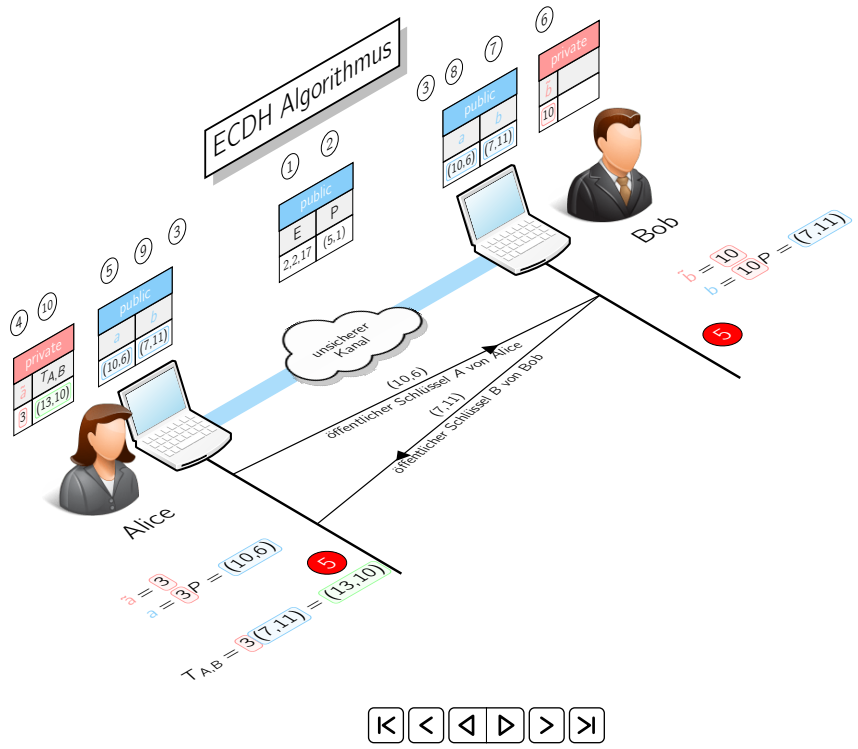


Abb. 15.6: *Diffie-Hellman-Schlüsselaustausch mit ECC*

15.2.1 ECDH Algorithmus

- **Diffie-Hellman-Set-up**
  - 1 **Domain**: Parametergenerierung
    - ① Wähle Primzahl und eine elliptische Kurve E:
      $a = 2$, $b = 2$, $p = 17$,
      $E : y^2 \equiv x^3 + 2x + 2 \pmod{17}$
    - ② Wähle ein primitives Element
      $P = (5,1)$
    - ③ Veröffentliche p, a, b und P
- **Diffie-Hellman-Schlüsselaustausch**
  - 2 **Alice**: Schlüsselgenerierung
    - ④ **Alice** Wähle privaten Schlüssel:
      $\tilde{a} = k_{pr,A} = 3$
    - ⑤ **Alice** Berechne öffentlichen Schlüssel:
      $a = k_{pub,A} = 3P = (10,6)$
  - 3 **Bob**: Schlüsselgenerierung
    - ⑥ **Bob** Wähle privaten Schlüssel:
      $\tilde{b} = k_{pr,B} = 10$
    - ⑦ **Bob** Berechne öffentlichen Schlüssel:
      $b = k_{pub,B} = 10P = (7,11)$
  - 4 **Alice→Bob**: Schlüsselaustausch
    - ⑧ **Alice→Bob**: öffentlichen Schlüssel von Alice
    - ⑨ **Bob→Alice**: öffentlichen Schlüssel von Bob
  - 5 **Geheimer Sitzungsschlüssel**
    - ⑩ **Alice** $T_{A,B} = \tilde{a}\ b = 3\ (7,11) = (13,10)$
    - ⑪ **Bob** $T_{A,B} = \tilde{b}\ a = 10\ (10,6) = (13,10)$

ECDH Algorithmus [30]

Anhang

16 Personenregister

Inhaltsangabe

16.1 Pioniere der Kryptologie

16.1.1 Claude Shannon

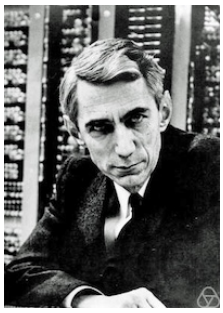


Abb. 16.1: *Claude Shannon*

Cl. Shannon

Claude Elwood Shannon (1916-2001) war ein US-amerikanischer Mathematiker und Elektrotechniker. Er gilt als Begründer der Informationstheorie.

1948 veröffentlichte Claude E. Shannon seine bahnbrechende Arbeit A Mathematical Theory of Communication (dt. Mathematische Grundlagen in der Informationstheorie).

In diesem Aufsatz konzentrierte er sich auf das Problem, unter welchen Bedingungen eine von einem Sender kodierte und durch einen gestörten Kommunikationskanal übermittelte Information am Zielort wiederhergestellt, also ohne Informationsverlust dekodiert werden kann.

Dabei konnte er den aus der Physik bekannten Begriff der Entropie erfolgreich in der Informationstheorie anwenden.

16.1.2 Phil Zimmermann

Abb. 16.2: *Phil Zimmermann*

Ph. Zimmermann

Philip R. Zimmermann (1954) ist Softwareentwickler und Erfinder der E-Mail-Verschlüsselungssoftware Pretty Good Privacy (PGP).

Er ist Mitbegründer und Chefentwickler von Silent Circle, einem Unternehmen für verschlüsselte Kommunikation.

Zimmermann studierte Informatik an der Florida Atlantic University und arbeitete anschließend als Softwareentwickler in Boulder, Colorado.

Mit seinem Programm PGP war er der erste, der die asymmetrische Kryptographie (auch Public-Key-Kryptographie genannt) als Software der Allgemeinheit leicht zugänglich machte.

„Wenn Privatsphäre gesetzlich verboten wird, haben nur Gesetzlose Privatsphäre."
(Phil Zimmermann)

Zitat 16.1

16.1.3 Whitfield Diffie

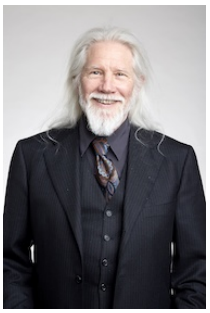


Abb. 16.3: *Whitfield Diffie*

Wh. Diffie [55]

Whitfield „Whit“ Diffie (1944) ist ein US-amerikanischer Experte für Kryptographie.

Er gehört gemeinsam mit Martin Hellman zu den Wegbereitern der Public-Key-Kryptographie (Verschlüsselung mit öffentlich zugänglichen Schlüsseln und asymmetrischen Verschlüsselungssystemen).

Zimmermann studierte Informatik an der Florida Atlantic University und arbeitete anschließend als Softwareentwickler in Boulder, Colorado.

Er beendete sein Mathematik-Studium am Massachusetts Institute of Technology 1965 (Bachelor-Abschluss) und war dort im Umfeld des Artificial Intelligence Lab, ab den 1960er Jahren ein führendes Forschungszentrum zur Künstlichen Intelligenz, an dem neben Wissenschaftlern auch viele Hacker aktiv waren.

Ab 1969 war er an der Stanford University, wo seine Zusammenarbeit mit Hellman zur Kryptographie begann. Sein Interesse dafür stammte wie das von Hellman insbesondere aus der Lektüre des 1967 erschienenen Klassikers The Codebreakers von David Kahn.

Nach seiner Zeit in Stanford arbeitete er an Sicherheits-Software bei Northern Telecom, wo er Manager für Secure Systems Research war und die Schlüssel-Verteilungs-Architektur für X.25-Netzwerke in deren PDSO-System entwickelte.

Ab 1991 arbeitete Diffie bei der Firma Sun Microsystems in Menlo Park in Kalifornien als Distinguished Engineer und als Chief Security Officer auf Vizepräsidentenebene.

Er ist Sun Fellow. Er war unter anderem Gastprofessor am Royal Holloway College der Universität London, war Fellow des Isaac Newton Institute in Cambridge und der Marconi Foundation.

16.1.4 Martin Hellman

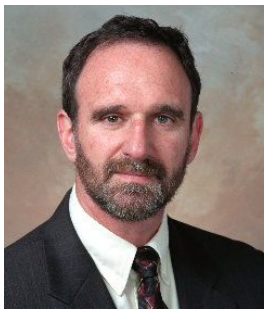


Abb. 16.4: *Martin Hellman*

M. Hellman [55]

Martin Edward Hellman (2. Oktober 1945 in New York City) ist ein US-amerikanischer Kryptologe, bekannt als einer der Entwickler der Public-Key-Kryptographie.

Hellman studierte Elektrotechnik an der New York University (Bachelorabschluss 1966) und der Stanford University (Master-Abschluss 1967), wo er 1969 promovierte.

Als Post-Doc war er 1968/69 bei IBM an deren IBM Thomas J. Watson Research Center in Yorktown Heights.

1969 bis 1971 war er Assistenzprofessor für Elektrotechnik am Massachusetts Institute of Technology und war ab 1971 zunächst Assistenzprofessor und später Professor in Stanford, wo er 1996 emeritierte.

Zurzeit ist er Professor Emeritus an der Stanford University.

16.1.5 James Henry Hellis

J. H. Hellis

James Henry Ellis (1924-1997) war ein britischer Ingenieur und Kryptograph, der Anfang der 1970er Jahre ein asymmetrisches Kryptosystem vorschlug, das aus Gründen der Geheimhaltung aber zunächst nicht veröffentlicht wurde.

Ende der 1960er Jahre bat deshalb das **GCHQ**[1] seinen Mitarbeiter Ellis um einen Vorschlag zur Kostensenkung. Beim Literaturstudium dazu stieß er auf Shannons Arbeit **Communication in the Presence of Noise**[2].

Ellis, der kein Mathematiker war, wusste nicht recht, wie er seinen Einfall implementieren sollte. Das erledigten 1973–1974 zwei junge Mathematiker – Clifford Cocks und Malcolm Williamson.

Die Arbeit der drei Forscher am GCHQ blieb zunächst geheim.

1976 publizierten Diffie und Hellman eine Vorarbeit zur Public-Key-Kryptographie, die das Schlüsselübermittlungsproblem löste und ein Jahr später bei dem RSA-Kryptosystem genutzt wurde.

Das GCHQ hatte Williamson 1976 eine Veröffentlichung untersagt.

16.1.6 Clifford Cocks

C. Cocks

Clifford Christopher Cocks, (1950) ist ein britischer Mathematiker und Kryptograph am GCHQ. Er entdeckte den weit verbreiteten Verschlüsselungsalgorithmus, der nun unter dem Namen RSA bekannt ist, etwa drei Jahre bevor er unabhängig von Ronald Rivest, Adi Shamir und Leonard Adleman am MIT entwickelt wurde.

Diese Errungenschaft wird ihm nicht allgemein anerkannt, da seine Arbeit per definitionem einer Geheimhaltungsstufe unterlag und deshalb damals nicht veröffentlicht wurde.

16.1.7 Malcolm Williamson

Williamson

Malcolm J. Williamson (1950-2015) war ein britischer Kryptologe.

Williamson war 1974 und 1975 als Mitarbeiter des britischen GCHQ an der Entwicklung eines asymmetrischen Verschlüsselungsverfahren beteiligt.

Die theoretischen Vorarbeiten hatte ab 1969 James H. Ellis, ebenfalls Mitarbeiter des GCHQ geleistet. Der Durchbruch gelang 1973 Clifford Cocks.

Da die Verschlüsselungsmethode der Geheimhaltung unterlag, wurde das Verfahren 1975 bis 1977 von amerikanischen Wissenschaftlern als RSA-Kryptosystem ein zweites Mal erfunden.

16.1.8 Bruce Schneier

Schneier

Bruce Schneier (1963) ist ein US-amerikanischer Experte für Kryptographie und Computersicherheit und Autor verschiedener Bücher über Computersicherheit.

Schneier ist Fellow des Berkman Center for Internet & Society und Lecturer für Public Policy an der Harvard Kennedy School.

Er wurde von der britischen Zeitung The Guardian in das Redaktionsteam geholt, das auf Basis der Enthüllungen von Edward Snowden die Überwachungs- und Spionageaffäre 2013 aufgedeckt hat.

Er ist zudem Vorstandsmitglied der Electronic Frontier Foundation.

[1] **englisch:** *GCHQ* = britischen Government Communications Headquarters

[2] **englisch:** *Communication in the Presence of Noise* = Nachrichtenübermittlung bei auftretenden Rauschsignalen

17 Algorithmen

Inhaltsangabe

17.1 Python Grundlagen

17.2 Algebra

17.2.1 Euklidischer Algorithmus

Berechnung vom ggT

```
import math

def print_euklid(a,b,a_diff,cnt,dig):
    if b>a: a,b = b, a
    print(" "*cnt + f"{a:{dig}d} - {b:{dig}d
        } = {a_diff:{dig}d}")

def ggTv1(a,b,cnt=0,dig=3):
    str_tab = cnt+1+int(math.log(max(a,b)))
    a_diff, a_min = abs(a-b), min(a,b)
    print_euklid(a,b,a_diff,str_tab,dig)
    if a!=b:
        return ggTv1(a_min,a_diff,cnt=
            str_tab)
    else:
        return a
```

Listing 17.1: *Euklidischer Algorithmus*

```
ggTv1(400,225,0)

400-225=175
    225-175=50
        175-50=125
            125-50=75
                75-50=25
                    50-25=25
                        25-25=0
25
```

```
# Euklidische Algorithmus
def ggT(a,b):
    diff, a_min = abs(a-b), min(a,b)
    return ggT(diff,a_min) if a != b else a
```

Listing 17.2: *Euklidischer Algorithmus*

```
ggT(400,225)

25
```

17.2.2 Erweiterter Eukl. Algorithmus

Berechnung vom ggT

```
def x_euklid(a,b):
    r1, r2 = divmod(a,b)
    c = (max(a,b)-r2)//r1
    print(f"{a} - {r1} x {c} = {r2} ")
    return x_euklid(c,r2) if r2 != 0 else c
```

Listing 17.3: *Erweiterter Euklidischer Algorithmus*

```
x_euklid(400,225)

400 - 1 x 225 = 175
225 - 1 x 175 =  50
175 - 3 x  50 =  25
 50 - 2 x  25 =   0
25
```

Berechnung der Inversen

```
def init_global_variables(i):
    global m, s, t
    if i==1: m, s, t = [], [1,0], [0,1]

def print_table_header(i,dig,spc,head="mqrst
    "):
    line = [f"{spc}{h}" for h in head]
    print(" i" + ''.join(line))
    print("-"*(dig*len(head)+3))

def print_lines(vals,i,dig):
    line = [f"{v:{dig}d}" for v in vals]
    print(f"{i:2d}" + ''.join(line))

def print_table(values,i,dig):
    '''print results per step i'''
    if i==1: print_table_header(i,dig," "*(
        dig-1))
    print_lines(values,i,dig)

def calc_next_in_list(l,qi):
    l.append(l[-2] - qi * l[-1])
    return l[-1]

def inv_euklid(a,r,i=1,dig=8):
    '''returns inverse of r mod a'''
    init_global_variables(i)
    m.append(a)
    qi, ri = divmod(a,r)
    c = (max(a,r)-ri)//qi
    si = calc_next_in_list(s,qi)
    ti = calc_next_in_list(t,qi)
    print_table((a,qi,ri,si,ti),i,dig)
    if ri > 1:
        return inv_euklid(c,ri,i=i+1)
    else:
        return ti if ti>0 else ti+m[0]
```

Listing 17.4: *Berechnung der Inversen (Teil I)*

17.3 AES-Algorithmus

17.3.1 Consolen-Ausgabe

```
import numpy as np
import re
from functools import reduce
import pprint
import galois
import termcolor
from termcolor import colored
```

Listing 17.5: *Python Imports*

```
def stringToByteMatrix(string):
    a = [int('0x' + val,0) for val in re.
        findall('.{1,2}',string)]
    return np.reshape(np.matrix(a),(4,4))

def stringToByteArray(string):
    result = np.array([int('0x' + val,0) for
         val in re.findall('.{1,2}',string)
        ])
    return result

def byteMatrixToString(matrix,b_max=16):
    result = matrix.transpose().reshape
        (1,16).flatten()
    l_str = [ "%02X" % n for n in result]
    return ''.join(l_str)
```

Listing 17.6: *String Converting*

```
GF_256 = galois.GF(2**8)

def byteToGaloisPoly(v_hex, g_field=GF_256):
    b_list = hexToBinaryList(v_hex)
    return galois.Poly(b_list,field=g_field)

def polyToByteArray(poly, g_field=GF_256):
    res = np.array([ polyToValue(p,'int')
        for p in poly.flatten() ])
    return res.reshape(4,4)

def byteMatrixToGaloisPoly(matrix,b_max=4):
    ufunc = np.frompyfunc(byteToGaloisPoly
        ,1,1)
    g_poly = ufunc(matrix)
    return g_poly
```

Listing 17.7: *GF Convering*

```
def bytesToState(bytes, row_max=4):
    state = np.array(bytes).reshape(row_max,
        row_max).transpose()
    return state

def byteArrayToState(b_array):
    states = [ bytesToState(b) for b in
        b_array]
    return states
```

Listing 17.8: *Byte Converting*

```
def pprint_sbox(sbox, row_max=16):
    out = []
    for i,n in enumerate(sbox):
        out.append(str("%02X " % n))
        if (i+1) % row_max == 0:
            out.append("\n")
    print(''.join(out))

def print_hex(res):
    pprint.pp(np.vectorize(hex)(res))


def print_ln(p):
    print()
    print(p)
```

Listing 17.9: *Console Output*

```
KEY_COLORS = ['green']*16
PRV_COLORS = [None,'green',None]*2

BYTE_COLORS = ['blue']*16
RND_COLORS = ['red','blue','blue','green',
    None]
OUT_COLORS = ['red','blue','white','green',
    None]

REV_RND_COLORS = ['blue','blue','red','green
    ',None]
REV_OUT_COLORS = ['white','blue','red','
    green',None]
```

Listing 17.10: *Farben*

17.3.2 Schichten von AES

```python
import numpy as np
import galois
from functools import wraps

from aes_lib.aes_print import *
```

Listing 17.11: *Python Imports*

```python
GF_256  = galois.GF(2**8)
IRR_256 = galois.Poly([1,0,0,0,1,1,0,1,1],
    field=GF_256)
```

Listing 17.12: *GF Konstanten*

```python
def processlayer(func):
    '''
    Decorater that handle and print current
        states
    '''
    @wraps(func)
    def wrapper(*args,**kwargs):
        layers, _ = args
        new = func(*args,**kwargs)
        layers.append(new)
        return
    return wrapper
```

Listing 17.13: *Decorator Consolen Output*

Vorrunde (Pre-Round)

```python
def pre_round(state:list, key:list,show=
    False):
    a_state = np.array(state).reshape(4,4).
        transpose()
    a_key = np.array(key).reshape(4,4)
    xor_state = np.bitwise_xor(a_state,a_key
        )

    if show:
        print_matrices([a_state,a_key,
            xor_state])
    return xor_state
```

Listing 17.14: *Pre-Round-Key-Schicht*

Add-Round-Key-Schicht

```python
def add_round_key(state:list, key:list,show=
    False):
    a_state = np.array(state).reshape(4,4)
    a_key = np.array(key).reshape(4,4)
    xor_state = np.bitwise_xor(a_state,a_key
        )

    if show:
        print_matrices([a_state,a_key,
            xor_state])
    return xor_state
```

Listing 17.15: *Add-Round-Key-Schicht*

Byte-Substitution-Schicht

```python
def byte_substitution(state:list,use_sbox,
    show=False):
    a_state = np.array(state)
    ufunc = np.frompyfunc(use_sbox,1,1)
    s_state = ufunc(a_state)

    if show:
        print_matrices([a_state,s_state])
    return s_state
```

Listing 17.16: *Byte-Substitution-Schicht*

ShiftRow-Schicht

```python
def shift_rows(state:list,r_dir:int,show=
    False):
    a_state = np.array(state)
    b_state = np.array(state)

    b_state[1] = np.roll(b_state[1],r_dir*3)
    b_state[2] = np.roll(b_state[2],r_dir*2)
    b_state[3] = np.roll(b_state[3],r_dir*1)

    if show:
        print_matrices([state,b_state])
    return b_state
```

Listing 17.17: *ShiftRow-Schicht*

Mix-Columns-Schicht

```python
def mix_columns(state:list,poly_mc,show=
    False):
    poly_b = byteMatrixToGaloisPoly(state)
    mul_result = np.matmul(poly_mc,poly_b)
    modulo_result = np.mod(mul_result,
        IRR_256)

    new_state = polyToByteArray(
        modulo_result)

    if show:
        print_matrices([state,new_state])
    return new_state
```

Listing 17.18: *Mix-Columns-Schicht*

17.3.3 Verschlüsselung

Python Imports

```
import numpy as np
import galois

from aes_lib.aes_print import *
from aes_lib.aes_layers import *
```

Listing 17.19: *Python Imports*

Python Konstante (MC)

```
MC = np.array([
    0x02, 0x03, 0x01, 0x01,
    0x01, 0x02, 0x03, 0x01,
    0x01, 0x01, 0x02, 0x03,
    0x03, 0x01, 0x01, 0x02,
]).reshape(4,4)

POLY_MC = byteMatrixToGaloisPoly(MC)
```

Listing 17.20: *MC-Matrix und Inverse*

Byte Substitution (inverse S-BOX)

```
def sbox(x:int=None):
    if x is None:
        return print_matrix(S_BOX.reshape
            (16,16),row_max=16)

    return S_BOX[x]
```

Listing 17.21: *S-Box*

Consolen Ausgabe (Print)

```
def print_encryption_process(aes_states):
    for i, states in enumerate(aes_states):
        if i==0:
            print('Vorrunde')
            print_matrices(states, c_cols=
                PRV_COLORS)
        elif i==10:
            print('%2d. Runde:' % i)
            print_matrices(states[1:],
                c_cols=OUT_COLORS)
        else:
            print('%2d. Runde:' % i)
            print_matrices(states[1:])
```

Listing 17.22: *Output States*

Schichten der Verschlüsselung

```
# Schichten (Layers)
# 1. BS ... ByteSubstitution-Schicht
# 2. SR ... ShiftRows-Schicht
# 3. MC ... MixColumns-Unterschicht
# 4. KA ... Key-Additiion-Schicht

@processlayer
def layer_BS(layers,i):
    return byte_substitution(layers[-1],sbox
        )

@processlayer
def layer_SR(layers,i):
    return shift_rows(layers[-1],+1)

@processlayer
def layer_MC(layers,i):
    return mix_columns(layers[-1],POLY_MC)
        if i<9 else layers[-1]

@processlayer
def layer_KA(layers,r_key):
    new = add_round_key(layers[-1], r_key)
    layers.append(r_key)
    return new
```

Listing 17.23: *MC-Matrix und Inverse*

Vollständige Verschlüsselung

```
def aes_encryption(s_text, keys,show=False):
    text = np.array(s_text).reshape(4,4).
        transpose()
    states = []
    r_state = pre_round(s_text, keys[0])
    states.append([text, keys[0], r_state])
    for i, r_key in enumerate(keys[1:]):
        layers = [states[-1][-1]]
        layer_BS(layers,i)
        layer_SR(layers,i)
        layer_MC(layers,i)
        layer_KA(layers,r_key)
        states.append(layers)
    if show:
        print_encryption_process(states)
```

Listing 17.24: *AES Verschlüsselung*

17.3.4 Entschlüsselung

Python Imports

```
import numpy as np
import galois
from functools import wraps

from aes_lib.aes_print import *
from aes_lib.aes_layers import *
```

Listing 17.25: *Python Imports*

Python Konstante (MC, IMC)

```
I_MC = np.array([
    0x0E, 0x0B, 0x0D, 0x09,
    0x09, 0x0E, 0x0B, 0x0D,
    0x0D, 0x09, 0x0E, 0x0B,
    0x0B, 0x0D, 0x09, 0x0E,
]).reshape(4,4)

POLY_I_MC = byteMatrixToGaloisPoly(I_MC)
```

Listing 17.26: *MC-Matrix und Inverse*

Byte Substitution (inverse S-BOX)

```
def i_sbox(x:int=None):
    if x is None:
        return print_matrix(IS_BOX.reshape
            (16,16),row_max=16)

    return IS_BOX[x]
```

Listing 17.27: *Inverse S-Box*

Consolen Ausgabe (Print)

```
def print_decryption_process(states):
    for i, states in enumerate(states):
        if i==0:
            print('Vorrunde')
            print_matrices(states, c_cols=
                PRV_COLORS)
        elif i==1:
            print('%2d. Runde:' % i)
            print_matrices(states[1:],
                c_cols=REV_OUT_COLORS)
        else:
            print('%2d. Runde:' % i)
            print_matrices(states[1:],
                c_cols=REV_RND_COLORS )
```

Listing 17.28: *Output States*

Schichten der Entschlüsselung

```
# inverse Schichten (Layers)
# 1. BS ... ByteSubstitution-Schicht
# 2. SR ... ShiftRows-Schicht
# 3. MC ... MixColumns-Unterschicht
# 4. KA ... Key-Additiion-Schicht

@processlayer
def inv_layer_BS(layers,i):
    return byte_substitution(layers[-1],
        i_sbox)

@processlayer
def inv_layer_SR(layers,i):
    return shift_rows(layers[-1],-1)

@processlayer
def inv_layer_MC(layers,i):
    return mix_columns(layers[-1],POLY_I_MC)
        if i>0 else layers[-1]

@processlayer
def inv_layer_KA(layers,r_key):
    new = add_round_key(layers[-1], r_key)
    layers.append(r_key)
    return new
```

Listing 17.29: *MC-Matrix und Inverse*

Vollständige Entschlüsselung

```
def aes_decryption(bytes, keys,show=False):
    states = []
    chiffre = bytesToState(bytes)
    r_state = pre_round(bytes, keys[0])
    states.append([chiffre, keys[0], r_state
        ])
    for i, rnd_key in enumerate(keys[1:]):
        layers = [states[-1][-1]]
        inv_layer_MC(layers,i)
        inv_layer_SR(layers,i)
        inv_layer_BS(layers,i)
        inv_layer_KA(layers,rnd_key)
        states.append(layers)
    if show:
        print_decryption_process(states)
```

Listing 17.30: *AES Entschlüsselung*

17.4 Elliptische Kurven

17.4.1 Berechnung der Inversen

```python
from math import log

def init_global_variables(i):
    global m, s, t
    if i==1: m, t = [], [0,1]

def calc_next_in_list(l,qi):
    l.append(l[-2] - qi * l[-1])
    return l[-1]

def inverse(a,r,i=1,dig=8):
    '''returns inverse of r mod a'''
    if r==1: return 1
    init_global_variables(i)
    m.append(a)
    if r<0: r = r+a
    qi, ri = divmod(a,r)
    c = (max(a,r)-ri)//qi
    ti = calc_next_in_list(t,qi)
    if ri > 1:
        return inverse(c,ri,i=i+1)
    else:
        return ti if ti>0 else ti+m[0]
```

Listing 17.31: *Python ECC*

17.4.2 Konsolen-Ausgabe

Pretty Print mit Rahmen.

```python
C, M = "\u2500","\u2502"
TL,TC,TR = "\u250c","\u252c","\u2510"
ML,MC,MR = "\u251c","\u253c","\u2524"
BL,BC,BR = "\u2514","\u2534","\u2518"

def pptable(lists,d=2):
    table = []
    t_len = len(lists[0])
    i_frm = "%s%s%s" % ("%",str(d),"d")
    tops = TL+TC.join([C*d]*t_len)+TR+"\n"
    mids = ML+MC.join([C*d]*t_len)+MR+"\n"
    bots = BL+BC.join([C*d]*t_len)+BR+"\n"
    for lst in lists:
        data = [i_frm % i for i in lst]
        table.append(M+M.join(data)+M+"\n")
    print(tops+mids.join(table)+bots)
```

Listing 17.32: *Pretty Print*

```python
def table_inverse(p):
    num = list(range(2,p))
    inv = [inverse(p,i) for i in num]
    dig = int(log(p,10)+1)
    k = p // 34
    for i in range(k+1):
        a, b = i*34, (i+1)*34
        pptable([num[a:b],inv[a:b]],d=dig)

    x1,y1 = P
    R, LP = (0,0), (x1,p-y1)
    all_pts = [P]
    while LP != R:
        nP = all_pts[-1]
        R = add_points(nP,P,p,a)
        all_pts.append(R)
    all_pts.append("O")
    return all_pts
```

Listing 17.33: *Tablelle aller Inversen*

17.4.3 Punkteaddition und -verdopplung

```python
def calc_slope(P,Q,p,a):
    x1, y1 = P
    x2, y2 = Q

    if P!=Q:
        nom, den = y2-y1, x2-x1
    else:
        nom, den = 3*x1**2 + a, 2*y1

    s = nom * inverse(p,den) % p

    return s
```

Listing 17.34: *Steigung s*

Punkteverdopplung bzw. -addition

```python
def add_points(P,Q,p,a):
    x1, y1 = P
    x2, y2 = Q

    s = calc_slope(P,Q,p,a)
    x3 = (s**2 - x1 - x2)  % p
    y3 = (s * (x1 - x3) - y1)  % p
    return (x3,y3)
```

Listing 17.35: *Punkteaddition,-verdopplung*

```
# Punkteverdopplung
P = (5,1)
add_points(P,P,17,2)

(6, 3)
```

```
# Punkteaddition (abelsche Gruppe, kommutativ!!!)
P, Q = (5,1), (6,3)
add_points(P,Q,17,2)
add_points(Q,P,17,2)

(10, 6)
(10, 6)
```

17.4.4 Zyklus-Elemente

Berechnung aller Gruppenelemente einer EC.

```python
# cyklus elements of elliptic curve
def pp_elements(plist):
    pp_e = []
    frm = "%3sP: %8s"
    for i,j in enumerate(plist):
        if (i) % 4 == 0:
            pp_e.append("\n"+frm % (i+1,j))
        else:
            pp_e.append(frm % (i+1,j))
    print(",        ".join(pp_e))

def all_elements(P,p,a):
    R, LP = (0,0), (P[0],p-P[1])
    all_pts = [P]
    while LP != R:
        nP = all_pts[-1]
        R = add_points(nP,P,p,a)
        all_pts.append(R)
    all_pts.append("O")
    return all_pts
```

Listing 17.36: *Zyklus-Elemente*

```
P = (5,1)
pts = all_elements(P,17,2)

pp_elements(pts)

  1P:   (5, 1),    2P:   (6, 3),    3P:  (10, 6),
  4P:   (3, 1),    5P:  (9, 16),    6P: (16, 13),
  7P:   (0, 6),    8P:  (13, 7),    9P:   (7, 6),
 10P:  (7, 11),   11P: (13, 10),   12P:  (0, 11),
 13P:  (16, 4),   14P:   (9, 1),   15P:  (3, 16),
 16P: (10, 11),   17P:   (6, 14),   18P:  (5, 16),
 19P:         O
```

```
P = (13,7)
pts = all_elements(P,17,2)

pp_elements(pts)

  1P:  (13, 7),    2P: (10, 11),    3P:   (9, 16),
  4P:  (16, 4),    5P:   (6, 3),    6P:   (7, 11),
  7P:  (5, 16),    8P:   (0, 6),    9P:   (3, 16),
 10P:   (3, 1),   11P:  (0, 11),   12P:   (5, 1),
 13P:   (7, 6),   14P:  (6, 14),   15P:  (16, 13),
 16P:   (9, 1),   17P:  (10, 6),   18P:  (13, 10),
 19P:         O
```

Quellenangaben

Literatur

[1] Albrecht Beutelspacher, Heike B. Neumann und Thomas Schwarzpaul. Kryptografie in Theorie und Praxis. 2. Auflage. Vieweg+Teubner GWV Fachverlage GmbH, Wiesbaden, 2010. ISBN: 978-3-8348-0977-3.

[2] Albrecht Beutelspacher, Jörg. Schwenk und Klaus-Dieter Wolfenstetter. Moderne Verfahren der Kryptographie. 9. Auflage. Springer-Verlag GmbH, 2022. ISBN: 978-3-662-65718-8.

[3] Christof Paar und Jan Pelzl. Kryptografie verständlich. Ein Lehrbuch für Studierende und Anwender. Springer-Verlag Berlin, 2016. ISBN: 978-3-662-49296-3.

[4] Christof Paar und Jan Pelzl. Understanding Cryptography. Solutions Handbook (Odd numbered Problems). Springer Vieweg, 2016.

[5] Christof Paar, Jan Pelzl und Tim Güneysu. Understanding Cryptography. From Established Symmetric and Asymmetric Ciphers to Post-Quantum Algorithms. Springer Vieweg, 2024. ISBN: 978-3-662-69006-2.

[6] Matthias Peter und Werner Schindler. A Proposal for Functionality Classes for Random Number Generators. Bundesamt für Sicherheit in der Informationstechnologie (BSI). This dokument proposes an evaluation methology for true and deterministic random number generators. 2022.

[7] Joachim Swoboda, Stephan Spitz und Michael Pramateftakis. Kryptografie und IT-Sicherheit. 1. Auflage. Vieweg+Teubner GWV Fachverlage GmbH, Wiesbaden, 2008. ISBN: 978-3-8348-0248-4.

Videos (IAIK)

[8] Maria Eichsleder. Cryptography WS 2020/21. Youtube. 2020.
URL: https://www.youtube.com/playlist?list=PLfX_2xRB__0NMha7KevY04wv4j77gf7yM.

[9] Maria Eichsleder. **VO 01**: Cryptography - Introduction. Youtube. 2020.
URL: https://www.youtube.com/watch?v=DYe56V5dTrk.

[10] Maria Eichsleder. **VO 02**: Symmetric Primitives and ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=hMoWNQbe5SI.

[11] Maria Eichsleder. **VO 03**: Stream Ciphers and lw. ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=3ffxvfl8Yrw.

[12] Maria Eichsleder. **VO 04**: Block Cipher Design and ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=Pk-Kci_NIKI.

[13] Maria Eichsleder. **VO 05**: Symmetric Authentication. Youtube. 2020.
URL: https://www.youtube.com/watch?v=3F6bfCXmZUc.

[14] Maria Eichsleder. **VO 06**: Symmetric Encryptionm ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=ARp5tUVWkWA.

Videos (PAAR)

[15] Christof Paar. Einführung in die Kryptographie. Youtube. 2020.
URL: https://www.youtube.com/@einfuhrungindiekryptograph621/videos.

[16] Christof Paar. **VO 01**: Einführung in die Kryptografie. Youtube. 2020.
URL: https://www.youtube.com/watch?v=nUwJzSSZu3Q.

[17] Christof Paar. **VO 02**: Modulare Arithmetik und ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=DPC5-IR4jRM.

[18] Christof Paar. **VO 03**: Stromchiffren, Zufallszahlen ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=y9hlEqzhVxU.

[19] Christof Paar. **VO 04**: Stromchiffren und LFSR und ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=xkucoLGg9UA.

[20] Christof Paar. **VO 05**: Die DES-Verschlüsselung. Youtube. 2020.
URL: https://www.youtube.com/watch?v=H7bvLU-2JUI.

[21] Christof Paar. **VO 06**: Der DES-Schlüsselfahrplan. Youtube. 2020.
URL: https://www.youtube.com/watch?v=QTmmOXDM1to.

[22] Christof Paar. **VO 07**: Einführung in endliche Körper. Youtube. 2020.
URL: https://www.youtube.com/watch?v=cumjreKJMBO.

[23] Christof Paar. **VO 08**: Die AES-Verschlüsselung. Youtube. 2020.
URL: https://www.youtube.com/watch?v=t0sg9DvB-PA.

[24] Christof Paar. **VO 09**: Der AES-Schlüsselfahrplan. Youtube. 2020.
URL: https://www.youtube.com/watch?v=tvYHcS57Lq4.

[25] Christof Paar. **VO 12**: Zahlentheorie für PKC und EEA. Youtube. 2020.
URL: https://www.youtube.com/watch?v=87HW_KU_fmY.

[26] Christof Paar. **VO 13**: Das RSA-Kryptosystem und ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=GkS4xkiOvgk.

[27] Christof Paar. **VO 14**: Der Diffie-Hellman-Schlüsselaustausch. Youtube. 2020.
URL: https://www.youtube.com/watch?v=A7Pw9w8wUYI.

[28] Christof Paar. **VO 15**: Das generalisierte DLP und ... Youtube. 2020.
URL: https://www.youtube.com/watch?v=nCHTRnVBY2E.

[29] Christof Paar. **VO 17**: Elliptische-Kurven-Kryptografie I. Youtube. 2020.
URL: https://www.youtube.com/watch?v=ikeA6SZ83W8.

[30] Christof Paar. **VO 18**: Elliptische-Kurven-Kryptografie II. Youtube. 2020.
URL: https://www.youtube.com/watch?v=NeM12tNQ59g.

Videos (WEITZ)

[31] Edmund Weitz. Konkrete Mathematik (nicht nur) für Mathematiker. Kongruenz bzgl. eines Moduls. 2020.
URL: https://www.youtube.com/watch?v=KxuRForrRoU.

[32] Edmund Weitz. Konkrete Mathematik (nicht nur) für Mathematiker.
URL: https://weitz.de/haw-videos.

[33] Edmund Weitz. Mathematik- und Informatikvideos. 2020.
URL: https://www.youtube.com/@WeitzHAWHamburg.

Wikipedia

[34] Wikipedia. Babington-Verschwörung. Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/Babington-Verschw%C3%B6rung.

[35] Wikipedia. Blaise de Vigenere. Online; Stand 20. November 2024. 2024.

[36] Wikipedia. Enigma (Maschine). Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/Enigma_(Maschine).

[37] Wikipedia. Entschlüsselung. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Entschl%C3%BCsselung.

[38] Wikipedia. Flipflop. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Flipflop.

[39] Wikipedia. Geheimtext. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Geheimtext.

[40] Wikipedia. Halbaddierer. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Halbaddierer.

[41] Wikipedia. Irreduzibles Polynom. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Irreduzibles_Polynom.

[42] Wikipedia. Klartext. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Klartext_(Kryptographie).

[43] Wikipedia. Kryptoanalyse. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Kryptoanalyse.

[44] Wikipedia. Kryptographie. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Kryptographie.

[45] Wikipedia. Kryptologie. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Kryptologie.

[46] Wikipedia. Leon Battista Alberti. Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/Leon_Battista_Alberti.

[47] Wikipedia. One-Time-Pad. Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/One-Time-Pad.

[48] Wikipedia. Roger Bacon. Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/Roger_Bacon.

[49] Wikipedia. Schlüssel (Kryptologie). Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Schl%C3%BCssel_(Kryptologie).

[50] Wikipedia. Skytale. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Skytale.

[51] Wikipedia. Substitution (Kryptographie). Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/Substitution_(Kryptographie).

[52] Wikipedia. Transposition (Kryptographie). Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/Transposition_(Kryptographie).

[53] Wikipedia. Verschlüsselung. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Verschl%C3%BCsselung.

[54] Wikipedia. Volladdierer. Online; Stand 20. November 2024. 2024.
URL: https://de.m.wikipedia.org/wiki/Volladdierer.

[55] Wikipedia. Whitfiield Diffie. Online; Stand 20. November 2024. 2024.
URL: https://de.wikipedia.org/wiki/Whitfield_Diffie.