Vorlesungsskript

Kryptologie 1

Wintersemester 03/04

Prof. Dr. Johannes Köbler
Humboldt-Universität zu Berlin
Lehrstuhl Komplexität und Kryptografie

16. März 2004

Inhaltsverzeichnis

1 Klassische Verfahren

1.1 Einführung

Kryptosysteme (Verschlüsselungsverfahren) dienen der Geheimhaltung von Nachrichten bzw. Daten. Hierzu gibt es auch andere Methoden wie z.B.

Physikalische Maßnahmen: Tresor etc.

Organisatorische Maßnahmen: einsamer Waldspaziergang etc.

Steganographische Maßnahmen: unsichtbare Tinte etc.

Andererseits können durch kryptographische Verfahren weitere **Schutzziele** realisiert werden.

- *Vertraulichkeit*
  - Geheimhaltung
  - Anonymität (z.B. Mobiltelefon)
  - Unbeobachtbarkeit (von Transaktionen)
- *Integrität*
  - von Nachrichten und Daten
- *Zurechenbarkeit*
  - Authentikation
  - Unabstreitbarkeit
  - Identifizierung
- *Verfügbarkeit*
  - von Daten
  - von Rechenressourcen
  - von Informationsdienstleistungen

In das Umfeld der Kryptographie fallen auch die folgenden Begriffe.

Kryptographie: Lehre von der Geheimhaltung von Informationen durch die Verschlüsselung von Daten. Im weiteren Sinne: Wissenschaft von der Übermittlung, Speicherung und Verarbeitung von Daten in einer von potentiellen Gegnern bedrohten Umgebung.

Kryptoanalysis: Erforschung der Methoden eines unbefugten Angriffs gegen ein Kryptoverfahren (Zweck: Vereitelung der mit seinem Einsatz verfolgten Ziele)

Kryptoanalyse: Analyse eines Kryptoverfahrens zum Zweck der Bewertung seiner kryptographischen Stärken bzw. Schwächen.

Kryptologie: Wissenschaft vom Entwurf, der Anwendung und der Analyse von kryptographischen Verfahren (umfasst Kryptographie und Kryptoanalyse).

1.2 Kryptosysteme

Es ist wichtig, Kryptosysteme von Codesystemen zu unterscheiden.

Codesysteme

- operieren auf semantischen Einheiten,
- starre Festlegung, welche Zeichenfolge wie zu ersetzen ist.

Beispiel 1 (Ausschnitt aus einem Codebuch der deutschen Luftwaffe)

xve	***Bis auf weiteres Wettermeldung gemäß Funkbefehl testen***
yde	***Frage***
sLk	***Befehl***
fin	***beendet***
eom	***eigene Maschinen***

Kryptosysteme

- operieren auf syntaktischen Einheiten,
- flexibler Mechanismus durch Schlüsselvereinbarung

Definition 2 (Alphabet)

Ein ***Alphabet*** ist eine geordnete endliche Menge $A = \{a_0, \dots, a_{m-1}\}$ von ***Zeichen***. Eine Folge $x = x_1 \dots x_n \in A^n$ heißt ***Wort*** (der ***Länge*** n). $A^* = \bigcup_{n \geq 0} A^n$.

Beispiel 3 *Das **lateinische Alphabet** A_{lat} enthält die 26 Buchstaben `A`, . . . , `Z`. Bei der Abfassung von Klartexten wurde meist auf den Gebrauch von Interpunktions- und Leerzeichen sowie auf Groß- und Kleinschreibung verzichtet ($\leadsto$ Verringerung der Redundanz im Klartext).*

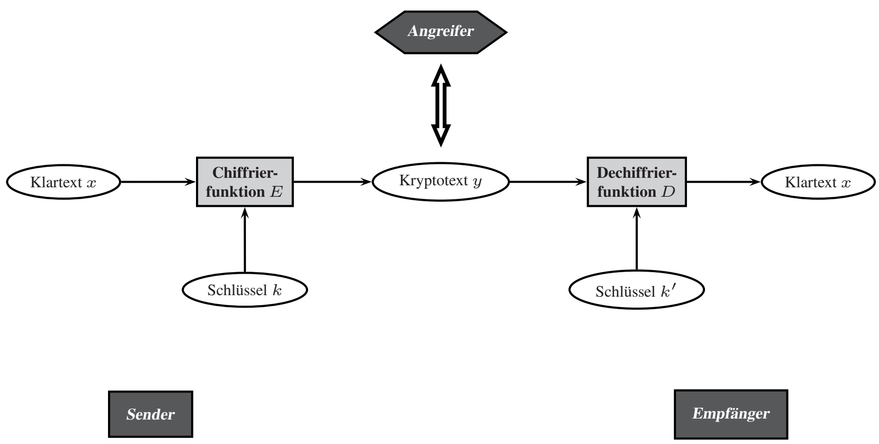
Definition 4 (**Kryptosystem**)

Ein **Kryptosystem** wird durch folgende Komponenten beschrieben:

- A, das **Klartextalphabet**,
- B, das **Kryptotextalphabet**,
- K, der **Schlüsselraum** (*key space*),
- $M \subseteq A^*$, der **Klartextraum** (*message space*),
- $C \subseteq B^*$, der **Kryptotextraum** (*ciphertext space*),
- $E : K \times M \to C$, die **Verschlüsselungsfunktion** (*encryption function*),
- $D : K \times C \to M$, die **Entschlüsselungsfunktion** (*decryption function*) und
- $S \subseteq K \times K$, eine Menge von Schlüsselpaaren (k, k') mit der Eigenschaft, dass für jeden Klartext $x \in M$ die Beziehung

$$D(k', E(k, x)) = x \tag{1}$$

gilt. Bei symmetrischen Kryptosystemen ist $S = \{(k, k) \mid k \in K\}$, weshalb wir auf die Angabe von S verzichten können.



Zu jedem Schlüssel $k \in K$ korrespondiert also eine **Chiffrierfunktion** $E_k : x \mapsto E(k, x)$ und eine **Dechiffrierfunktion** $D_k : y \mapsto D(k, y)$. Die Gesamtheit dieser Abbildungen wird auch **Chiffre** (englisch *cipher*) genannt. (Daneben wird der Begriff „Chiffre“ auch als Bezeichnung für einzelne Kryptotextzeichen oder kleinere Kryptotextsequenzen verwendet.)

Lemma 5 *Für jedes Paar $(k, k') \in S$ ist die Chiffrierfunktion $E_k : x \mapsto E(k, x)$ injektiv.*

Beweis: Angenommen, für zwei unterschiedliche Klartexte $x_1 \neq x_2$ ist $E(k, x_1) = E(k, x_2)$. Dann folgt

$$D(k', E(k, x_1)) = D(k', E(k, x_2)) \stackrel{(1)}{=} x_2 \neq x_1,$$

im Widerspruch zu (1). ■

1.3 Die affine Chiffre

Die Moduloarithmetik erlaubt es uns, das Klartextalphabet mit einer Addition und Multiplikation auszustatten.

Definition 6 (**teilt-Relation, modulare Kongruenz, ganzzahliger Rest**)

Seien a, b, m ganze Zahlen mit $m \geq 1$. Die Zahl a ***teilt*** b (kurz: $a|b$), falls ein $d \in \mathbb{Z}$ existiert mit $b = ad$. Teilt m die Differenz $a - b$, so schreiben wir hierfür

$$a \equiv_m b$$

(in Worten: a ist ***kongruent*** zu b modulo m). Weiterhin bezeichne

$$a \bmod m$$

den bei der Ganzzahldivision von a durch m auftretenden ***Rest***, also diejenige ganze Zahl $r \in \{0, \ldots, m-1\}$, für die eine ganze Zahl $d \in \mathbb{Z}$ existiert mit $a = dm + r$.

Die auf $\mathbb{Z}$ definierten Operationen

$$a \oplus_m b := (a + b) \bmod m$$

und

$$a \odot_m b := ab \bmod m.$$

sind abgeschlossen auf $\mathbb{Z}_m = \{0, \ldots, m-1\}$ und bilden auf dieser Menge einen kommutativen Ring mit Einselement, den sogenannten **Restklassenring** modulo m. Für $a \oplus_m -b$ schreiben wir auch $a \ominus_m b$.

Definition 7 (**Buchstabenrechnung**)

Sei $A = \{a_0, \ldots, a_{m-1}\}$ ein Alphabet. Für Indizes $i, j \in \{0, \ldots, m-1\}$ und eine ganze Zahl $z \in \mathbb{Z}$ ist

$$\begin{aligned} a_i + a_j &= a_{i \oplus_m j}, \quad & a_i - a_j &= a_{i \ominus_m j}, \quad & a_i a_j &= a_{i \odot_m j}, \\ a_i + z &= a_{i \oplus_m z}, \quad & a_i - z &= a_{i \ominus_m z}, \quad & z a_j &= a_{z \odot_m j}. \end{aligned}$$

Wir rechnen also mit Buchstaben, indem wir sie mit ihren Indizes identifizieren und die Rechnung modulo m ausführen. Mit Hilfe dieser Notation lässt sich die Verschiebechiffre, die auch als additive Chiffre bezeichnet wird, leicht beschreiben.

Definition 8 **(additive Chiffre)**

Bei der ***additiven Chiffre*** ist $A = B = M = C$ ein beliebiges Alphabet mit $m := \|A\| > 1$ und $K = \{1, \dots, m-1\}$. Für $k \in K$, $x \in M$ und $y \in C$ gilt

$$E(k, x) = x + k \text{ und } D(c, y) = y - k.$$

Im Fall des lateinischen Alphabets führt der Schlüssel $k = 13$ auf eine interessante Chiffrierfunktion, die in UNIX-Umgebungen auch unter der Bezeichnung ROT13 bekannt ist. Natürlich kann mit dieser Substitution nicht ernsthaft die Vertraulichkeit von Nachrichten geschützt werden. Vielmehr soll durch sie ein unbeabsichtigtes Mitlesen – etwa von Rätsellösungen – verhindert werden.

Tabelle 1 Werte der additiven Chiffrierfunktion ROT13 (Schlüssel $k = 13$).

x	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
$E(13, x)$	*N O P Q R S T U V W X Y Z A B C D E F G H I J K L M*

ROT13 ist eine ***involutorische*** – also zu sich selbst inverse – Abbildung, d.h. für alle $x \in A$ gilt

$$\text{ROT13}(\text{ROT13}(x)) = x.$$

Die Buchstabenrechnung legt folgende Modifikation der Caesar-Chiffre nahe: Anstatt auf jeden Klartextbuchstaben den Schlüsselwert k zu addieren, können wir die Klartextbuchstaben auch mit k multiplizieren. Allerdings erhalten wir hierbei nicht für jeden Wert von k eine injektive Chiffrierfunktion. So bildet etwa die Funktion $g : A_{lat} \to A_{lat}$ mit $g(x) = 2x$ sowohl A als auch N auf den Buchstaben $g(\texttt{A}) = g(\texttt{N}) = \texttt{A}$ ab. Um die vom Schlüsselwert k zu erfüllende Bedingung angeben zu können, führen wir folgende Begriffe ein.

Definition 9 **(ggT, kgV, teilerfremd)**

Seien $a, b \in \mathbb{Z}$. Für $(a, b) \neq (0, 0)$ ist

$$\text{ggT}(a, b) = \max\{d \in \mathbb{Z} \mid d \text{ teilt die beiden Zahlen } a \text{ und } b\}$$

der ***größte gemeinsame Teiler*** von a und b. Für $a \neq 0, b \neq 0$ ist

$$\text{kgV}(a, b) = \min\{d \in \mathbb{Z} \mid d \geq 1 \text{ und die beiden Zahlen } a \text{ und } b \text{ teilen } d\}$$

das ***kleinste gemeinsame Vielfache*** von a und b. Ist $\text{ggT}(a, b) = 1$, so nennt man a und b ***teilerfremd***.

Euklidscher Algorithmus: Der größte gemeinsame Teiler zweier Zahlen a und b lässt sich wie folgt bestimmen.

O. B. d. A. sei $a > b > 0$. Bestimme die natürlichen Zahlen (durch Divsision mit Rest):

$$r_{-1} = a > r_0 = b > r_1 > \cdots > r_n > r_{n+1} = 0 \quad \text{und} \quad d_1, d_2, d_3, \dots d_{n+1}$$

mit

$$r_{i-1} = d_{i+1} \cdot r_i + r_{i+1} \quad \text{für} \quad i = 0,\ 1,\ \ldots, n.^*$$

Hierzu sind $n+1$ Divisionsschritte erforderlich. Wegen

$$\mathrm{ggT}(r_{i-1}, r_i) = \mathrm{ggT}(r_i,\ \underbrace{r_{i-1} - d_{i+1} \cdot r_i}_{r_{i+1}})$$

folgt $\mathrm{ggT}(a, b) = r_n$.

Beispiel 10 *Für $a = 693$ und $b = 147$ erhalten wir z. B.*

i	r_{i-1}	$=$	d_{i+1}	$\cdot$	r_i	$+$	r_{i+1}
0	693	$=$	4	$\cdot$	147	$+$	105
1	147	$=$	1	$\cdot$	105	$+$	42
2	105	$=$	2	$\cdot$	42	$+$	21
3	42	$=$	2	$\cdot$	**21**	$+$	0

und damit $\mathrm{ggT}(693, 147) = r_3 = 21$.

Algorithmus 11 EUKLID$(a,\ b)$ (iterativ)

```
repeat r ← a mod b
  a ← b
  b ← r
until r = 0
return a
```

Algorithmus 12 EUKLID$(a,\ b)$ (rekursiv)

```
if b = 0 then
  return a
else
  return EUKLID(b, a mod b)
end
```

Zur Abschätzung von n verwenden wir die Folge der Fibonacci-Zahlen f_n:

$$f_n = \begin{cases} 0, & \text{falls } n = 0 \\ 1, & \text{falls } n = 1 \\ f_{n-1} + f_{n-2}, & \text{falls } n \geq 2 \end{cases}$$

Durch Induktion über i (mit $n \geq i \geq -1$) folgt $r_i \geq f_{n+1-i}$; also $a \geq f_{n+2}$. Wegen $f_n \geq \Re^n$ (wobei $\Re = \frac{1+\sqrt{5}}{2}$; Beweis durch Induktion) ist dann $a \geq \Re^n$, d. h. $n \leq \log_{\Re} a$.

Satz 13 *Der Euklidsche Algorithmus führt zur Berechnung von $\mathrm{ggT}(a, b)$ (unter der Annahme $a > b > 0$) höchstens $\lfloor \log_R a \rfloor + 1$ Divisionsschritte durch. Dies führt auf eine Zeitkomplexität von $O(n^3)$, wobei n die Länge der Eingabe in Binärdarstellung bezeichnet.*

---

*Also: $d_i = r_{i-2}$ div r_{i-1} und $r_i = r_{i-2} \bmod r_{i-1}$.

Erweiterter Euklidscher bzw. **Berlekamp-Algorithmus:** Der Euklidsche Algorithmus kann so modifiziert werden, dass er eine lineare Darstellung

$$\mathrm{ggT}(a,b) = \lambda \cdot a + \mu \cdot b \quad \text{mit} \quad \lambda,\ \mu \in \mathbb{Z}$$

des ggT liefert (Zeitkomplexität ebenfalls $O(n^3)$). Hierzu werden neben r_i und d_i (für $1 \le i \le n$) weitere Zahlen

$$p_i = p_{i-2} - d_i \cdot p_{i-1}, \quad \text{wobei} \quad p_{-1} = 1 \quad \text{und} \quad p_0 = 0,$$

und

$$q_i = q_{i-2} - d_i \cdot q_{i-1}, \quad \text{wobei} \quad q_{-1} = 0 \quad \text{und} \quad q_0 = 1,$$

bestimmt. Dann gilt für $i = -1$ und $i = 0$,

$$a \cdot p_i + b \cdot q_i = r_i,$$

und durch Induktion über i,

$$\begin{aligned} a \cdot p_{i+1} + b \cdot q_{i+1} &= a \cdot (p_{i-1} - d_{i+1} \cdot p_i) + b \cdot (q_{i-1} - d_{i+1} \cdot q_i) \\ &= a \cdot p_{i-1} + b \cdot q_{i-1} - d_{i+1} \cdot (a \cdot p_i + b \cdot q_i) \\ &= (r_{i-1} - d_{i+1} \cdot r_i) \\ &= r_{i+1} \end{aligned}$$

zeigt man, dass dies auch für $i = 1,\ 2,\ \ldots,\ n$ gilt.

Korollar 14 (Lemma von Bezout) *Der größte gemeinsame Teiler von a und b ist in der Form*

$$\mathrm{ggT}(a,b) = \lambda \cdot a + \mu \cdot b \quad \textit{mit} \quad \lambda,\ \mu \in \mathbb{Z}$$

darstellbar.

Beispiel 15 *Für $a = 693$ und $b = 147$ erhalten wir z. B. mit*

i	r_{i-1}	$=$	$d_{i+1} \cdot$	r_i	$+$	r_{i+1}	$p_i \cdot 693$	$+$	$q_i \cdot 147$	$=$	r_i
-1							$1 \cdot 693$	$+$	$0 \cdot 147$	$=$	693
0	693	$=$	4	$\cdot\, 147$	$+$	105	$0 \cdot 693$	$+$	$1 \cdot 147$	$=$	147
1	147	$=$	1	$\cdot\, 105$	$+$	42	$1 \cdot 693$	$+$	$(-4) \cdot 147$	$=$	105
2	105	$=$	2	$\cdot\, 42$	$+$	21	$(-1) \cdot 693$	$+$	$5 \cdot 147$	$=$	42
3	42	$=$	2	$\cdot\, \mathbf{21}$	$+$	0	$\mathbf{3} \cdot 693$	$+$	$(\mathbf{-14}) \cdot 147$	$=$	21

die lineare Darstellung $3 \cdot 693 - 14 \cdot 147 = 21$.

Aus der linearen Darstellbarkeit des größten gemeinsamen Teilers ergeben sich eine Reihe von nützlichen Schlussfolgerungen.

Korollar 16 *Der größte gemeinsame Teiler von a und b wird von allen gemeinsamen Teilern von a und b geteilt,*

$$x|a \ \wedge \ x|b \quad \Rightarrow \quad x|\,\mathrm{ggT}(a,b).$$

Korollar 17 (Lemma von Euklid) *Teilt a das Produkt $b \cdot c$ und gilt $\mathrm{ggT}(a, b) = 1$, so ist a auch Teiler von c,*

$$a|(b \cdot c) \ \wedge \ \mathrm{ggT}(a, b) = 1 \quad \Rightarrow \quad a|c.$$

Beweis: Wegen $\mathrm{ggT}(a, b) = 1$ existieren Zahlen $\mu, \lambda \in \mathbb{Z}$ mit $\mu \cdot a + \lambda \cdot b = 1$. Da a nach Voraussetzung das Produkt $b \cdot c$ teilt, muss a auch $c \cdot \mu \cdot a + c \cdot \lambda \cdot b = c$ teilen. ■

Korollar 18 *Wenn sowohl a als auch b zu einer Zahl $m \in \mathbb{Z}$ teilerfremd sind, so ist auch das Produkt $a \cdot b$ teilerfremd zu m,*

$$\mathrm{ggT}(a, m) = \mathrm{ggT}(b, m) = 1 \quad \Rightarrow \quad \mathrm{ggT}(a \cdot b,\ m) = 1.$$

Satz 19 (Chinesischer Restsatz) *Falls m_1 und m_2 teilerfremd sind, hat das System der linearen Kongruenzen*

$$\begin{aligned} x &\equiv_{m_1} b_1 \\ x &\equiv_{m_2} b_2 \end{aligned}$$

für alle $b_1, b_2 \in \mathbb{Z}$ genau eine Lösung modulo $m = m_1 m_2$.

Beweis: Wegen $\mathrm{ggT}(a, m) = \mathrm{ggT}(b, m) = 1$ existieren Zahlen $\mu, \lambda, \mu', \lambda' \in \mathbb{Z}$ mit

$$\mu \cdot a + \lambda \cdot m = \mu' \cdot b + \lambda' \cdot m = 1.$$

Mit $\mu \cdot a \cdot \mu' \cdot b = (1 - \lambda \cdot m) \cdot (1 - \lambda' \cdot m) = 1 - m \cdot (\lambda + \lambda' - m \cdot \lambda \cdot \lambda')$ ergibt sich $\mathrm{ggT}(a \cdot b,\ m) = 1$. ∎

Damit nun eine Abbildung $g : A \to A$ von der Bauart $g(x) = bx$ injektiv (oder gleichbedeutend, surjektiv) ist, muss es zu jedem Buchstaben $y \in A$ genau einen Buchstaben $x \in A$ mit $bx = y$ geben. Wie der folgende Satz zeigt, ist dies genau dann der Fall, wenn b und m teilerfremd sind.

Satz 20 *Sei $m \geq 1$. Die lineare Kongruenzgleichung $bx \equiv_m y$ besitzt genau dann eine eindeutige Lösung $x \in \{0, \dots, m-1\}$, wenn $\mathrm{ggT}(b, m) = 1$ ist.*

Beweis: Angenommen, $\mathrm{ggT}(b, m) = g > 1$. Dann ist mit x auch $x' = x + m/g$ eine Lösung von $bx \equiv_m y$ mit $x \not\equiv_m x'$. Gilt umgekehrt $\mathrm{ggT}(b, m) = 1$, so folgt aus den Kongruenzen

$$bx_1 \equiv_m y$$

und

$$bx_2 \equiv_m y$$

sofort $m|(bx_1 - y)$ und $m|(bx_2 - y)$, also $m|b(x_1 - x_2)$. Wegen $\mathrm{ggT}(b, m) = 1$ folgt mit dem Lemma von Euklid $m|(x_1 - x_2)$, also $x_1 \equiv_m x_2$. Dies zeigt, dass die Abbildung $f : \mathbb{Z}_m \to \mathbb{Z}_m$ mit $f(x) = bx \bmod m$ injektiv ist. Da jedoch Definitions- und Wertebereich von f identisch sind, muss f dann auch surjektiv sein. ∎

Korollar 21 *Im Fall $\mathrm{ggT}(b, m) = 1$ hat die Kongruenz $bx \equiv_m 1$ genau eine Lösung, die das **multiplikative Inverse** von b modulo m genannt und mit $b^{-1} \bmod m$ (oder einfach mit b^{-1}) bezeichnet wird. Die invertierbaren Elemente von $\mathbb{Z}_m$ werden in der Menge*

$$\mathbb{Z}_m^* = \{b \in \mathbb{Z}_m \mid \mathrm{ggT}(b, m) = 1\}$$

zusammengefasst.

Korollar 18 zeigt, dass $\mathbb{Z}_m^*$ unter der Operation $\odot_m$ abgeschlossen ist, und mit Korollar 21 folgt, dass $(\mathbb{Z}_m^*, \odot_m)$ eine multiplikative Gruppe bildet.

Das multiplikative Inverse von b modulo m ergibt sich aus der linearen Darstellung $\lambda a + \mu b = \mathrm{ggT}(b, m) = 1$ zu $b^{-1} = \mu \bmod m$. Bei Kenntnis von b^{-1} kann die Kongruenz $bx \equiv_m y$ leicht zu $x = yb^{-1}$ gelöst werden. Die folgende Tabelle zeigt die multiplikativen Inversen b^{-1} für alle $b \in \mathbb{Z}_{26}^*$.

b	1	3	5	7	9	11	15	17	19	21	23	25
b^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

Nun lässt sich die additive Chiffre leicht zur affinen Chiffre erweitern.

Definition 22 **(affine Chiffre)**

Bei der ***affinen Chiffre*** ist $A = B = M = C$ ein beliebiges Alphabet mit $m := \|A\| > 1$ und $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$. Für $k = (b, c) \in K$, $x \in M$ und $y \in C$ gilt

$$E(k, x) = bx + c \ \text{ und } \ D(k, y) = b^{-1}(y - c).$$

In diesem Fall liefert die Schlüsselkomponente $b = -1$ für jeden Wert von c eine involutorische Chiffrierfunktion $x \mapsto E(b, c; x) = c - x$ (**verschobenes komplementäres Alphabet**). Wählen wir für c ebenfalls den Wert -1, so ergibt sich die Chiffrierfunktion $h(x) = -x - 1$, die auch als **revertiertes Alphabet** bekannt ist:

x	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
$h(x) = -x - 1$	*Z Y X W V U T S R Q P O N M L K J I H G F E D C B A*

Wie man sieht, handelt es sich hierbei um eine echt involutorische Abbildung, d.h. kein Buchstabe wird auf sich selbst abgebildet (vorausgesetzt m ist gerade). Als nächstes illustrieren wir die Ver- und Entschlüsselung mit der affinen Chiffre an einem kleinen Beispiel.

Beispiel 23 (affine Chiffre)
Sei $A = \{\mathtt{A}, \ldots, \mathtt{Z}\} = B$, also $m = 26$. Weiter sei $k = (9, 2)$, also $b = 9$ und $c = 2$. Um den Klartextbuchstaben $x = \mathtt{F}$ zu verschlüsseln, berechnen wir

$$E(k, x) = bx + c = 9\mathtt{F} + 2 = V,$$

da der Index von $\mathtt{F}$ gleich 5, der von V gleich 21 und $9 \cdot 5 + 2 = 47 \equiv_{26} 21$ ist. Um einen Kryptotextbuchstaben wieder entschlüsseln zu können, benötigen wir das multiplikative Inverse von $b = 9$, das sich wegen

i	r_{i-1}	$=$	$d_{i+1} \cdot r_i$	$+$	r_{i+1}	$p_i \cdot 26$	$+$	$q_i \cdot 9$	$=$	r_i
-1						$1 \cdot 26$	$+$	$0 \cdot 9$	$=$	26
0	26	$=$	$2 \cdot 9$	$+$	8	$0 \cdot 26$	$+$	$1 \cdot 9$	$=$	9
1	9	$=$	$1 \cdot 8$	$+$	1	$1 \cdot 26$	$+$	$(-2) \cdot 9$	$=$	8
2	8	$=$	$8 \cdot \mathbf{1}$	$+$	0	$(-1) \cdot 26$	$+$	$3 \cdot 9$	$=$	$\mathbf{1}$

zu $b^{-1} = q_2 = 3$ ergibt. Damit erhalten wir für den Kryptotextbuchstaben $y = V$ den ursprünglichen Klartextbuchstaben

$$D(k, y) = b^{-1}(y - c) = 3(V - 2) = \mathtt{F}$$

zurück, da $3 \cdot 19 = 57 \equiv_{26} 5$ ist.

Eine wichtige Rolle spielt die Funktion

$$\varphi : \mathbb{N} \to \mathbb{N} \quad \text{mit} \quad \varphi(m) = \|\mathbb{Z}_m^*\| = \|\{a \mid 0 \le a \le m - 1,\ \mathrm{ggT}(a, m) = 1\}\|,$$

die sogenannte *Eulersche φ-Funktion*.

m	1	2	3	4	5	6	7	8	9
$\mathbb{Z}_m^*$	$\{0\}$	$\{1\}$	$\{1,2\}$	$\{1,3\}$	$\{1,2,3,4\}$	$\{1,5\}$	$\{1,\cdots,6\}$	$\{1,3,5,7\}$	$\{1,2,4,5,7,8\}$
$\varphi(m)$	1	1	2	2	4	2	6	4	6

Wegen

$$\mathbb{Z}_{p^e} - \mathbb{Z}_{p^e}^* = \{0, p, 2p, \ldots, (p^{e-1}-1)p\}$$

folgt sofort

$$\varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p-1).$$

Um hieraus für beliebige Zahlen $m \in \mathbb{N}$ eine Formel für $\varphi(m)$ zu erhalten, genügt es, $\varphi(a \cdot b)$ im Fall $\mathrm{ggT}(a,b) = 1$ in Abhängigkeit von $\varphi(a)$ und $\varphi(b)$ zu bestimmen. Hierzu betrachten wir die Abbildung $f : \mathbb{Z}_{ab} \to \mathbb{Z}_a \times \mathbb{Z}_b$ mit

$$f(x) := (x \bmod a, x \bmod b).$$

Beispiel 24 *Sei $a = 3$ und $b = 7$. Dann erhalten wir die Funktion $f : \mathbb{Z}_{21} \to \mathbb{Z}_3 \times \mathbb{Z}_7$ mit*

x	0	$\mathbf{1}$	$\mathbf{2}$	3	$\mathbf{4}$	$\mathbf{5}$	6	7	$\mathbf{8}$	9
$f(x)$	$(0,0)$	$(\mathbf{1},\mathbf{1})$	$(\mathbf{2},\mathbf{2})$	$(0,\mathbf{3})$	$(\mathbf{1},\mathbf{4})$	$(\mathbf{2},\mathbf{5})$	$(0,\mathbf{6})$	$(\mathbf{1},0)$	$(\mathbf{2},\mathbf{1})$	$(0,\mathbf{2})$

$\mathbf{10}$	$\mathbf{11}$	12	$\mathbf{13}$	14	15	$\mathbf{16}$	$\mathbf{17}$	18	$\mathbf{19}$	$\mathbf{20}$
$(\mathbf{1},\mathbf{3})$	$(\mathbf{2},\mathbf{4})$	$(0,\mathbf{5})$	$(\mathbf{1},\mathbf{6})$	$(\mathbf{2},0)$	$(0,\mathbf{1})$	$(\mathbf{1},\mathbf{2})$	$(\mathbf{2},\mathbf{3})$	$(0,\mathbf{4})$	$(\mathbf{1},\mathbf{5})$	$(\mathbf{2},\mathbf{6})$

*Die **fett** gedruckten Werte gehören zu $\mathbb{Z}_{21}^*$, $\mathbb{Z}_3^*$ bzw. $\mathbb{Z}_7^*$. Man beachte, dass ein x-Wert genau dann in $\mathbb{Z}_{21}^*$ ist, wenn beide Komponenten von $f(x)$ zu $\mathbb{Z}_3^*$ bzw. $\mathbb{Z}_7^*$ gehören.* $\triangleleft$

Der Chinesische Restsatz, den wir im nächsten Abschnitt beweisen, besagt, dass f im Fall $\mathrm{ggT}(a,b) = 1$ bijektiv und damit invertierbar ist.

f^{-1}	0	1	2	3	4	5	6
0	0	15	9	3	18	12	6
1	7	1	16	10	4	19	13
2	14	8	2	17	11	5	20

Wegen

$$\begin{aligned} \mathrm{ggT}(x,ab) = 1 \quad &\Leftrightarrow \quad \mathrm{ggT}(x,a) = \mathrm{ggT}(x,b) = 1 \\ &\Leftrightarrow \quad \mathrm{ggT}(x \bmod a, a) = \mathrm{ggT}(x \bmod b, b) = 1 \end{aligned}$$

ist daher die Einschränkung von f auf den Bereich $\mathbb{Z}_{ab}^*$ eine Bijektion zwischen $\mathbb{Z}_{ab}^*$ und $\mathbb{Z}_a^* \times \mathbb{Z}_b^*$, d.h. es gilt

$$\varphi(ab) = \|\mathbb{Z}_{ab}^*\| = \|\mathbb{Z}_a^* \times \mathbb{Z}_b^*\| = \|\mathbb{Z}_a^*\| \cdot \|\mathbb{Z}_b^*\| = \varphi(a)\varphi(b).$$

Satz 25 *Die Eulersche φ-Funktion ist multiplikativ, d. h. für teilerfremde Zahlen a und b gilt $\varphi(ab) = \varphi(a)\varphi(b)$.*

Korollar 26 *Sei $m = \prod_{i=1}^{k} p_i^{e_i}$ die Primfaktorzerlegung von m. Dann gilt*

$$\varphi(m) = \prod_{i=1}^{k} p_i^{e_i - 1}(p_i - 1).$$

Beweis: Es gilt

$$\varphi(\prod_{i=1}^{k} p_i^{e_i}) = \prod_{i=1}^{k} \varphi(p_i^{e_i}) = \prod_{i=1}^{k}(p_i^{e_i} - p_i^{e_i-1}) = \prod_{i=1}^{k} p_i^{e_i-1}(p_i - 1).$$

■

Der Chinesische Restsatz

Die beiden linearen Kongruenzen

$$\begin{aligned} x &\equiv_3 0 \\ x &\equiv_6 1 \end{aligned}$$

besitzen je eine Lösung, es gibt aber kein x, das beide Kongruenzen gleichzeitig erfüllt. Der nächste Satz zeigt, dass unter bestimmten Voraussetzungen gemeinsame Lösungen existieren, und wie sie berechnet werden können.

Satz 27 (Chinesischer Restsatz) *Falls $m_1, \ldots, m_k$ paarweise teilerfremd sind, dann hat das System*

$$\begin{aligned} x &\equiv_{m_1} b_1 \\ &\vdots \\ x &\equiv_{m_k} b_k \end{aligned} \tag{2}$$

genau eine Lösung modulo $m = \prod_{i=1}^{k} m_i$.

Beweis: Da die Zahlen $n_i = {}^{m}/_{m_i}$ teilerfremd zu m_i sind, existieren Zahlen μ_i und λ_i mit

$$\mu_i n_i + \lambda_i m_i = \mathrm{ggT}(n_i, m_i) = 1.$$

Dann gilt

$$\mu_i n_i \equiv_{m_i} 1$$

und

$$\mu_i n_i \equiv_{m_j} 0$$

für $j \neq i$. Folglich erfüllt $x = \sum_{j=1}^{k} \mu_i n_i b_i \bmod m$ die Kongruenzen

$$x \equiv_{m_i} \mu_i n_i b_i \equiv_{m_i} b_i$$

für $i = 1, \ldots, k$. Dies zeigt, dass (2) lösbar, also die Funktion

$$f : \mathbb{Z}_m \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k}$$

mit $f(x) = (x \bmod m_1, \ldots, x \bmod m_k)$ surjektiv ist. Da der Definitions- und der Wertebereich von f die gleiche Mächtigkeit haben, muss f jedoch auch injektiv sein, d.h. (2) ist sogar eindeutig lösbar. ■

Man beachte, dass der Beweis des Chinesischen Restsatzes konstruktiv ist und die Lösung x unter Verwendung des erweiterten Euklidschen Algorithmus' effizient berechenbar ist.

1.4 Die Hill-Chiffre

Die von Hill im Jahr 1929 publizierte Chiffre ist eine Erweiterung der multiplikativen Chiffre auf Buchstabenblöcke, d.h. der Klartext wird nicht zeichenweise, sondern blockweise verarbeitet. Sowohl der Klartext- als auch der Kryptotextraum enthält alle Wörter x über A einer festen Länge l. Zur Chiffrierung wird eine $(l \times l)$-Matrix $k = (k_{ij})$ mit Koeffizienten in $\mathbb{Z}_m$ benutzt, die einen Klartextblock $x = x_1 \ldots x_l \in A^l$ in den Kryptotextblock $y_1 \ldots y_l \in A^l$ transformiert, wobei

$$y_i = x_1 k_{1i} + \cdots + x_l k_{li},\ i = 1, \ldots, l$$

ist (hierbei machen wir von der Buchstabenrechnung Gebrauch). y entsteht also durch Multiplikation von x mit der Schlüsselmatrix k:

$$(x_1, \cdots, x_l) \begin{pmatrix} k_{11} & \ldots & k_{1l} \\ \vdots & \ddots & \vdots \\ k_{l1} & \ldots & k_{ll} \end{pmatrix} = (y_1, \cdots, y_l)$$

Wir bezeichnen die Menge aller $(l \times l)$-Matrizen mit Koeffizienten in $\mathbb{Z}_m$ mit $\mathbb{Z}_m^{l \times l}$. Als Schlüssel können nur invertierbare Matrizen k benutzt werden, da sonst der Chiffriervorgang nicht injektiv ist. k ist genau dann invertierbar, wenn die Determinante von k teilerfremd zu m ist (siehe Übungen).

Definition 28 (**Determinante**)

Sei $A = (a_{ij})$ eine $l \times l$-Matrix. Für $1 \leq i, j \leq l$ sei A_{ij} die durch Streichen der i-ten Zeile und j-ten Spalte aus K hervorgehende Matrix. Die ***Determinante*** von A ist dann $\det(A) = a_{11}$, falls $l = 1$, und

$$\det(A) = \sum_{j=1}^{l} (-1)^{i+j} a_{i,j} \det(A_{ij}),$$

wobei $i \in \{1, \cdots, l\}$ (beliebig wählbar) ist.

Für die Dechiffrierung wird die zu k inverse Matrix k^{-1} benötigt, wofür effiziente Algorithmen bekannt sind (siehe Übungen).

Satz 29

Sei A ein Alphabet und sei $k \in \mathbb{Z}_m^{l\times l}$ ($l \geq 1$, $m = \|A\|$). Die Abbildung $f : A^l \to A^l$ mit

$$f(x) = xk,$$

ist genau dann injektiv, wenn $\mathrm{ggT}(\det(k), m) = 1$ ist.

Beweis: Siehe Übungen. ■

Definition 30 **(Hill-Chiffre)**

Sei $A = \{a_0, \ldots, a_{m-1}\}$ ein beliebiges Alphabet und für eine natürliche Zahl $l \geq 2$ sei $M = C = A^l$. Bei der **Hill-Chiffre** ist $K = \{k \in \mathbb{Z}_m^{l\times l} \mid \mathrm{ggT}(\det(k), m) = 1\}$ und es gilt

$$E(k, x) = xk \;\text{ und }\; D(k, y) = yk^{-1}.$$

Beispiel 31 (Hill-Chiffre) *Benutzen wir zur Chiffrierung von Klartextblöcken der Länge $l = 4$ über dem lateinischen Alphabet A_{lat} die Schlüsselmatrix*

$$k = \begin{pmatrix} 11 & 13 & 8 & 21 \\ 24 & 17 & 3 & 25 \\ 18 & 12 & 23 & 17 \\ 6 & 15 & 2 & 15 \end{pmatrix},$$

so erhalten wir beispielsweise für den Klartext HILL *wegen*

$$(\mathtt{H}, \mathtt{I}, \mathtt{L}, \mathtt{L}) \begin{pmatrix} 11 & 13 & 8 & 21 \\ 24 & 17 & 3 & 25 \\ 18 & 12 & 23 & 17 \\ 6 & 15 & 2 & 15 \end{pmatrix} = (N, E, R, X)$$

bzw.

$$\begin{aligned} 11\,\mathtt{H} + 24\,\mathtt{I} + 18\,\mathtt{L} + 6\,\mathtt{L} &= N \\ 13\,\mathtt{H} + 17\,\mathtt{I} + 12\,\mathtt{L} + 15\,\mathtt{L} &= E \\ 8\,\mathtt{H} + 3\,\mathtt{I} + 23\,\mathtt{L} + 2\,\mathtt{L} &= R \\ 21\,\mathtt{H} + 25\,\mathtt{I} + 17\,\mathtt{L} + 15\,\mathtt{L} &= X \end{aligned}$$

den Kryptotext $E(k, \mathtt{HILL}) = NERX$.

1.5 Die Vigenère-Chiffre und andere Stromsysteme

Bei der nach dem Franzosen Blaise de Vigenère (1523–1596) benannten Chiffre werden zwar nur einzelne Buchstaben chiffriert, aber je nach Position im Klartext unterschiedlich.

Definition 32 (**Vigenère-Chiffre**)

Sei $A = B$ ein beliebiges Alphabet. Die **Vigenère-Chiffre** chiffriert unter einem Schlüssel $k = k_0 \dots k_{d-1} \in K = A^*$ einen Klartext $x = x_0 \dots x_{n-1}$ beliebiger Länge zu

$$E(k, x) = y_0 \dots y_{n-1}, \quad \text{wobei } y_i = x_i + k_{(i \bmod d)} \text{ ist,}$$

und dechiffriert einen Kryptotext $y = y_0 \dots y_{n-1}$ zu

$$D(k, y) = x_0 \dots x_{n-1}, \quad \text{wobei } x_i = y_i - k_{(i \bmod d)} \text{ ist.}$$

Beispiel 33 (Vigenère-Chiffre) *Verwenden wir das lateinische Alphabet A_{lat} als Klartextalpabet und wählen wir als Schlüssel das Wort $k =$ `WIE`, so ergibt sich für den Klartext* `VIGENERE` *beispielsweise der Kryptotext*

$$\begin{aligned} E(\mathit{WIE}, \mathtt{VIGENERE}) &= \underbrace{\mathtt{V}+\mathit{W}}_{\mathit{R}}\underbrace{\mathtt{I}+\mathit{I}}_{\mathit{Q}}\underbrace{\mathtt{G}+\mathit{E}}_{\mathit{K}}\underbrace{\mathtt{E}+\mathit{W}}_{\mathit{A}}\underbrace{\mathtt{N}+\mathit{I}}_{\mathit{V}}\underbrace{\mathtt{E}+\mathit{E}}_{\mathit{I}}\underbrace{\mathtt{R}+\mathit{W}}_{\mathit{N}}\underbrace{\mathtt{E}+\mathit{I}}_{\mathit{M}} \\ &= \mathit{RQKAVINM} \end{aligned}$$

Um einen Klartext x zu verschlüsseln, wird also das Schlüsselwort $k = k_0 \dots k_{d-1}$ so oft wiederholt, bis der dabei entstehende **Schlüsselstrom** $\hat{k} = k_0, k_1, \dots, k_{d-1}, k_0 \dots$ die Länge von x erreicht. Dann werden x und $\hat{k}$ zeichenweise addiert, um den zugehörigen Kryptotext y zu bilden. Aus diesem kann der ursprüngliche Klartext x zurückgewonnen werden, indem man den Schlüsselstrom $\hat{k}$ wieder subtrahiert.

Beispiel 33 (Vigenère-Chiffre, Fortsetzung)

Chiffrierung:		*Dechiffrierung:*	
`VIGENERE`	(*Klartext* x)	*RQKAVINM*	(*Kryptotext* y)
$+$ *WIEWIEWI*	(*Schlüsselstrom* $\hat{k}$)	$-$ *WIEWIEWI*	(*Schlüsselstrom* $\hat{k}$)
RQKAVINM	(*Kryptotext* $y = x + \hat{k}$)	`VIGENERE`	(*Klartext* $x = y - \hat{k}$)

Die Chiffrierarbeit lässt sich durch Benutzung einer Additionstabelle erleichtern (auch als **Vigenère-Tableau** bekannt).

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

Um eine involutorische Chiffre zu erhalten, schlug Sir Francis Beaufort, ein Admiral der britischen Marine, vor, den Schlüsselstrom nicht auf den Klartext zu addieren, sondern letzteren von ersterem zu subtrahieren.

Beispiel 34 (Beaufort-Chiffre) *Verschlüsseln wir den Klartext* BEAUFORT *beispielsweise unter dem Schlüsselwort $k =$ WIE, so erhalten wir den Kryptotext XMEQNSNB. Eine erneute Verschlüsselung liefert wieder den Klartext* BEAUFORT*:*

Chiffrierung:

	WIEWIEWI	(*Schlüsselstrom* $\hat{k}$)
−	BEAUFORT	(*Klartext* x)
	XMEQNSNB	(*Kryptotext* $y = \hat{k} - x$)

Dechiffrierung:

	WIEWIEWI	(*Schlüsselstrom* $\hat{k}$)
−	XMEQNSNB	(*Kryptotext* y)
	BEAUFORT	(*Klartext* $x = \hat{k} - y$)

Bei den bisher betrachteten Chiffren wird aus einem Schlüsselwort $k = k_0 \dots k_{d-1}$ ein **periodischer Schlüsselstrom** $\hat{k} = \hat{k}_0 \dots \hat{k}_{n-1}$ erzeugt, das heißt, es gilt $\hat{k}_i = \hat{k}_{i+d}$ für alle $i = 0, \dots, n - d - 1$. Da eine kleine Periode das Brechen der Chiffre erleichtert, sollte entweder ein Schlüsselstrom mit sehr großer Periode oder noch besser ein **fortlaufender Schlüsselstrom** zur Chiffrierung benutzt werden. Ein solcher nichtperiodischer Schlüsselstrom lässt sich beispielsweise ohne großen Aufwand erzeugen, indem man an das Schlüsselwort den Klartext oder den Kryptotext anhängt (sogenannte ***Autokey*-Chiffrierung**).[†]

[†]Die Idee, den Schlüsselstrom durch Anhängen des Klartextes an ein Schlüsselwort zu bilden, stammt von Vigenère, während er mit der Erfindung der nach ihm benannten Vigenère-Chiffre „nichts zu tun“ hatte. Diese wird vielmehr Giovan Batista Belaso (1553) zugeschrieben.

Beispiel 35 (*Autokey*-Chiffre) *Benutzen wir wieder das Schlüsselwort* `WIE`*, um den Schlüsselstrom durch Anhängen des Klar- bzw. Kryptotextes zu erzeugen, so erhalten wir für den Klartext* `VIGENERE` *folgende Kryptotexte:*

Klartext-Schlüsselstrom:

$$\begin{array}{rll} & \mathtt{VIGENERE} & (\textit{Klartext}\,) \\ + & \underline{\underline{\mathtt{WIE}}\mathit{VIGEN}} & (\textit{Schlüsselstrom}) \\ & \mathit{RQKZVKVR} & (\textit{Kryptotext}\,) \end{array}$$

Kryptotext-Schlüsselstrom:

$$\begin{array}{rll} & \mathtt{VIGENERE} & (\textit{Klartext}\,) \\ + & \underline{\underline{\mathtt{WIE}}\mathit{RQKVD}} & (\textit{Schlüsselstrom}) \\ & \mathit{RQKVDOMH} & (\textit{Kryptotext}\,) \end{array}$$

Auch die Dechiffrierung ist in beiden Fällen einfach. Bei der ersten Alternative kann der Empfänger durch Subtraktion des Schlüsselworts den Anfang des Klartextes bilden und gleichzeitig den Schlüsselstrom verlängern, so dass sich auf diese Weise Stück für Stück der gesamte Kryptotext entschlüsseln lässt. Noch einfacher gestaltet sich die Dechiffrierung im zweiten Fall, da sich hier der Schlüsselstrom vom Kryptotext nur durch das vorangestelle Schlüsselwort unterscheidet.

1.6 Der One-Time-Tape

Es besteht auch die Möglichkeit, eine Textstelle in einem Buch als Schlüssel zu vereinbaren und den dort beginnenden Text als Schlüsselstrom zu benutzen (Lauftextverschlüsselung). Besser ist es jedoch, aus einem relativ kurzen Schlüssel einen möglichst zufällig erscheinenden Schlüsselstrom zu erzeugen. Hierzu können beispielsweise Pseudozufallsgeneratoren eingesetzt werden. Absolute Sicherheit wird dagegen erreicht, wenn der Schlüsselstrom rein zufällig erzeugt und nach einmaliger Benutzung wieder vernichtet wird.[‡] Ein solcher „Wegwerfschlüssel“ (*One-Time-Pad* oder *One-Time-Tape*, im Deutschen auch als **individueller Schlüssel** bezeichnet) lässt sich allerdings nur mit großem Aufwand generieren und verteilen, weshalb diese Chiffre nur wenig praktikabel ist. Dennoch wurde diese Methode beispielsweise beim „heißen Draht“, der 1963 eingerichteten, direkten Fernschreibverbindung zwischen dem Weißen Haus in Washington und dem Kreml in Moskau, angewandt.

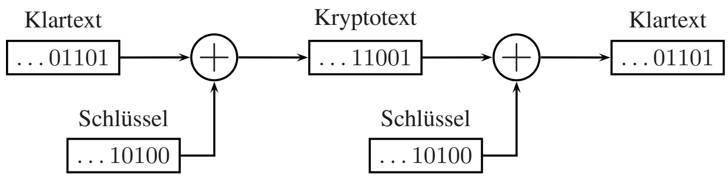
Beispiel 36 (*One-Time-Pad*) *Sei* $A = \{a_0, \ldots, a_{m-1}\}$ *ein beliebiges Klartextalphabet. Um einen Klartext* $x = x_0 \ldots x_{n-1}$ *zu verschlüsseln, wird auf jeden Klartextbuchstaben* x_i *ein neuer, zufällig generierter Schlüsselbuchstabe* k_i *addiert,*

$$y = y_0 \ldots y_{n-1}, \text{ wobei } y_i = x_i + k_i.$$

Der Klartext wird also wie bei einer additiven Chiffre verschlüsselt, nur dass der Schlüssel nach einmaligem Gebrauch gewechselt wird. Dies entspricht dem Gebrauch einer Vigenère-Chiffre, falls als Schlüssel ein zufällig gewähltes Wort von der Länge des

[‡]Diese Art der Schlüsselerzeugung schlug der amerikanische Major Joseph O. Mauborgne im Jahr 1918 vor, nachdem ihm ein von Gilbert S. Vernam für den Fernschreibverkehr entwickeltes Chiffriersystem vorgestellt wurde.

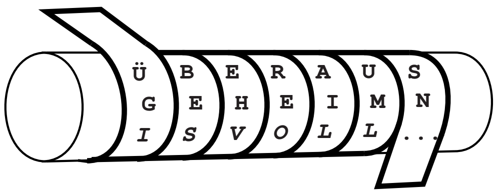
Klartextes benutzt wird. Wie diese ist der *One-Time-Pad* im Binärfall also involutorisch.



1.7 Klassifikation von Kryptosystemen

Die bisher betrachteten Chiffrierfunktionen handelt es sich um **Substitutionen**, d.h. sie erzeugen den Kryptotext aus dem Klartext, indem sie Klartextzeichen – einzeln oder in Gruppen – durch Kryptotextzeichen *ersetzen*. Dagegen verändern **Transpositionen** lediglich die *Reihenfolge* der einzelnen Klartextzeichen.

Beispiel 37 (Skytale-Chiffre) *Die älteste bekannte Verschlüsselungstechnik stammt aus der Antike und wurde im 5. Jahrhundert v. Chr. von den Spartanern entwickelt: Der Sender wickelt einen Papierstreifen spiralförmig um einen Holzstab (die sogenannte* ***Skytale****) und beschreibt ihn in Längsrichtung mit der Geheimbotschaft.*



ÜBERAUS GEHEIMNISVOLL...
$\rightsquigarrow$ *ÜGI...BES...EHV...REO...AIL...UML...SN...*

Besitzt der Empfänger eines auf diese Weise beschrifteten Papierstreifens einen Stab mit dem gleichen Umfang, so kann er ihn auf dieselbe Art wieder entziffern.

Als Schlüssel fungiert hier also der Stabumfang bzw. die Anzahl k der Zeilen, mit denen der Stab beschrieben wird. Findet der gesamte Klartext x auf der Skytale Platz und beträgt seine Länge ein Vielfaches von k, so geht x bei der Chiffrierung in den Kryptotext

$$E(k, x_1 \cdots x_{km}) = x_1 x_{m+1} x_{2m+1} \cdots x_{(k-1)m+1} x_2 x_{m+2} x_{2m+2} \cdots x_{(k-1)m+2} \cdots x_m x_{2m} x_{3m} \cdots x_{km}$$

über. Dasselbe Resultat stellt sich ein, wenn wir x zeilenweise in eine $k \times m$-Matrix schreiben und spaltenweise wieder auslesen (sogenannte **Spaltentransposition**):

$$\begin{array}{|llll|}
\hline
x_1 & x_2 & \cdots & x_m \\
x_{m+1} & x_{m+2} & \cdots & x_{2m} \\
x_{2m+1} & x_{2m+2} & \cdots & x_{3m} \\
\vdots & \vdots & \ddots & \vdots \\
x_{(k-1)m+1} & x_{(k-1)m+2} & \cdots & x_{km} \\
\hline
\end{array}$$

Ist die Klartextlänge kein Vielfaches von k, so kann der Klartext durch das Ein- bzw. Anfügen von sogenannten **Blendern** (Füllzeichen) verlängert werden. Damit der Empfänger diese Füllzeichen nach der Entschlüsselung wieder entfernen kann, ist lediglich darauf zu achten, dass sie im Klartext leicht als solche erkennbar sind.

Von der Methode, die letzte *Zeile* nur zum Teil zu füllen, ist dagegen abzuraten. In diesem Fall würden nämlich auf dem abgewickelten Papierstreifen Lücken entstehen, aus deren Anordnung man Schlüsse auf den benutzten Schlüssel k ziehen könnte. Andererseits ist nichts dagegen einzuwenden, dass der Sender die letzte *Spalte* der Skytale nur zum Teil beschriftet.

Bevor wir weitere Beispiele für Transpositionen betrachten, wenden wir uns der Klassifikation von Substitutionschiffren zu. Ein wichtiges Unterscheidungsmerkmal ist z.B. die Länge der Klartexteinheiten, auf denen die Chiffre operiert.

Monographische Substitutionen ersetzen Einzelbuchstaben.

Polygraphische Substitutionen ersetzen dagegen aus mehreren Zeichen bestehende Klartextsegmente auf einmal.

Eine polygraphische Substitution, die auf Buchstabenpaaren operiert, wird **digraphisch** genannt. Das älteste bekannte polygraphische Chiffrierverfahren wurde von Giovanni Porta im Jahr 1563 veröffentlicht. Dabei werden je zwei aufeinanderfolgende Klartextbuchstaben durch ein einzelnes Kryptotextzeichen ersetzt.

Beispiel 38 *Bei der **Porta-Chiffre** werden 400 (!) unterschiedliche von Porta für diesen Zweck entworfene Kryptotextzeichen verwendet. Diese sind in einer* 20×20*-Matrix* $M = (y_{ij})$ *angeordnet, deren Zeilen und Spalten mit den* 20 *Klartextbuchstaben* $\mathtt{A}, \ldots, \mathtt{I}, \mathtt{L}, \ldots, \mathtt{T}, \mathtt{V}, \mathtt{Z}$ *indiziert sind. Zur Ersetzung des Buchstabenpaars* $a_i a_j$ *wird das in Zeile* i *und Spalte* j *befindliche Kryptotextzeichen*

$$E(M, a_i a_j) = y_{ij}$$

benutzt.

Eine Substitution heißt **monopartit**, falls sie die Klartextsegmente durch Einzelzeichen ersetzt, sonst **multipartit**. Wird der Kryptotext aus Buchstabenpaaren zusammengesetzt, so spricht man von einer **bipartiten** Substitution.

Ein frühes (monographisches) Beispiel einer bipartiten Chiffriermethode geht auf Polybios (circa 200 – 120 v. Chr.) zurück:

M	*0*	*1*	*2*	*3*	*4*
0	A	B	C	D	E
1	F	G	H	I	J
2	K	L	M	N	O
3	P	Q	R	S	T
4	U	V	W	X/Y	Z

POLYBIOS $\rightsquigarrow$ *30 24 21 43 01 13 24 33*

Bei der **Polybios-Chiffre** dient eine 5×5-Matrix, die aus sämtlichen Klartextbuchstaben gebildet wird, als Schlüssel.[§] Die Verschlüsselung des Klartextes erfolgt buchstabenweise, indem man einen in Zeile i und Spalte j eingetragenen Klartextbuchstaben

[§]Da nur 25 Plätze zur Verfügung stehen, muss bei Benutzung des lateinischen Alphabets entweder ein Buchstabe weggelassen oder ein Platz mit zwei Buchstaben besetzt werden.

durch das Koordinatenpaar ij ersetzt. Der Kryptotextraum besteht also aus den Ziffernpaaren $\{\texttt{00}, \texttt{01}, \ldots, \texttt{44}\}$.

Die Frage, ob bei der Ersetzung der einzelnen Segmente des Klartextes eine einheitliche Strategie verfolgt wird oder ob diese von Segment zu Segment verändert wird, führt uns auf ein weiteres wichtiges Unterscheidungsmerkmal bei Substitutionen.

Monoalphabetische Substitutionen ersetzen die einzelnen Klartextsegment unabhängig von ihrer Position im Klartext.

Polyalphabetische Substitutionen verwenden dagegen eine variable Ersetzungsregel, auf die sich auch die bereits verarbeiteten Klartextsegmente auswirken.

Die Bezeichnung „monoalphabetisch" bringt zum Ausdruck, dass der Ersetzungsmechanismus auf einem einzelnen Alphabet beruht (sofern wir das Klartextalphabet als bekannt voraussetzen). Die von Caesar benutzte Chiffriermethode kann beispielsweise vollständig durch Angabe des Ersetzungsalphabets

$$\{\texttt{D,E,F,G,W,...,Y,Z,A,B,C}\}$$

beschrieben werden. Auch im Fall, dass nicht einzelne Zeichen, sondern ganze Buchstabengruppen auf einmal ersetzt werden, genügt im Prinzip ein einzelnes Alphabet zur Beschreibung. Hierzu sortiert man die Klartexteinheiten, auf denen der Ersetzungsmechanismus operiert, und bildet die Folge (sprich: das Alphabet) der zugeordneten Kryptotextsegmente.

Monoalphabetische Chiffrierverfahren ersetzen meist Texteinheiten einer festen Länge $l \geq 1$ durch Kryptotextsegmente derselben Länge.

Definition 39 (**Blockchiffre**)

Sei A ein beliebiges Alphabet und es gelte $M = C = A^l$, $l \geq 1$. Eine ***Blockchiffre*** realisiert für jeden Schlüssel $k \in K$ eine bijektive Abbildung g auf A^l und es gilt

$$E(k, x) = g(x) \text{ und } D(k, y) = g^{-1}(y)$$

für alle $x \in M$ und $y \in C$. Im Fall $l = 1$ spricht man auch von einer ***einfachen Substitutionschiffre***.

Polyalphabetische Substitutionen greifen im Wechsel auf verschiedene Ersetzungsalphabete zurück, so dass unterschiedliche Vorkommen eines Zeichens (oder einer Zeichenkette) auch auf unterschiedliche Art ersetzt werden können. Welches Ersetzungsalphabet wann an der Reihe ist, wird dabei in Abhängigkeit von der Länge oder der Gestalt des bereits verarbeiteten Klartextes bestimmt.

Fast alle polyalphabetischen Chiffrierverfahren operieren – genau wie monoalphabetische Substitutionen – auf Klartextblöcken einer festen Länge l, die sie in Kryptotextblöcke einer festen Länge l' überführen, wobei meist $l = l'$ ist. Da diese Blöcke jedoch vergleichsweise kurz sind, kann der Klartext der Chiffrierfunktion ungepuffert zugeführt werden. Man nennt die einzelnen Klartextblöcke in diesem Zusammenhang auch nicht ‚Blöcke' sondern ‚Zeichen' und spricht von **sequentiellen Chiffren** oder von **Stromchiffren**.

Definition 40 (**Stromchiffre**)

Sei A ein beliebiges Alphabet und sei $M = C = A^l$ für eine natürliche Zahl $l \geq 1$. Weiterhin seien K und $\hat{K}$ Schlüsselräume. Eine **Stromchiffre** wird durch eine Verschlüsselungsfunktion $E : \hat{K} \times M \to C$ und einen Schlüsselstromgenerator $g : K \times A^* \to \hat{K}$ beschrieben. Der Generator g erzeugt aus einem externen Schlüssel $k \in K$ für einen Klartext $x = x_0 \ldots x_{n-1}$, $x_i \in M$, eine Folge $\hat{k}_0, \ldots, \hat{k}_{n-1}$ von internen Schlüsseln $\hat{k}_i = g(k, x_0 \ldots x_{i-1}) \in \hat{K}$, unter denen x in den Kryptotext

$$E_g(k, x) = E(\hat{k}_0, x_0) \ldots E(\hat{k}_{n-1}, x_{n-1})$$

überführt wird.

Der interne Schlüsselraum kann also wie bei der Blockchiffre eine maximale Größe von $(m^l)!$ annehmen (im häufigen Spezialfall $l = 1$ also $m!$). Die Aufgabe des Schlüsselstromgenerators g besteht darin, aus dem externen Schlüssel k und dem bereits verarbeiteten Klartext $x_0 \ldots x_{i-1}$ den aktuellen internen Schlüssel $\hat{k}_i$ zu berechnen. Die bisher betrachteten Stromchiffren benutzen z.B. die folgenden Schlüsselstromgeneratoren.

Stromchiffre	Chiffrierfunktionen	Schlüsselstromgenerator
Vigenère	$E(\hat{k}, x) = x + \hat{k},\ \hat{k} \in A$	$g(k_0 \ldots k_{d-1}, x_0 \ldots x_{i-1}) = k_{(i \bmod m)}$
Beaufort	$E(\hat{k}, x) = \hat{k} - x,\ \hat{k} \in A$	$g(k_0 \ldots k_{d-1}, x_0 \ldots x_{i-1}) = k_{(i \bmod m)}$
Autokey[a]	$E(\hat{k}, x) = x + \hat{k},\ \hat{k} \in A$	$g(k_0 \ldots k_{d-1}, x_0 \ldots x_{i-1}) = \begin{cases} k_i, & i < d \\ x_{i-d}, & i \geq d \end{cases}$
Autokey[b]	$E(\hat{k}, x) = x + \hat{k},\ \hat{k} \in A$	$g(k_0 \ldots k_{d-1}, x_0 \ldots x_{i-1}) = \begin{cases} k_i, & i < d \\ y_{i-d}, & i \geq d \end{cases}$ $= k_{(i \bmod d)} + \sum_{j=1}^{\lfloor i/d \rfloor} x_{i-jd}$

[a]mit Klartext-Schlüsselstrom
[b]mit Kryptotext-Schlüsselstrom

Bei der Vigenère- und Beaufortchiffre hängt der Schlüsselstrom nicht vom Klartext, sondern nur vom externen Schlüssel k ab, d.h. sie sind **synchron**. Die *Autokey*-Chiffren sind dagegen **asynchron** (und aperiodisch).

Gespreizte Substitutionen

Bei den bisher betrachteten Substitutionen haben die einzelnen Blöcke, aus denen der Kryptotext zusammengesetzt wird, eine einheitliche Länge. Es liegt nahe, einem Gegner die unbefugte Rekonstruktion des Klartextes dadurch zu erschweren, dass man

Blöcke unterschiedlicher Länge verwendet. Man spricht hierbei auch von einer **Spreizung** (*straddling*) des Kryptotextalphabets. Ein bekanntes Beispiel für diese Technik ist die sogenannte Spionage-Chiffre, die vorzugsweise von der ehemaligen sowjetischen Geheimpolizei NKWD (*Naródny Komissariàt Wnutrennich Del*; zu deutsch: Volkskommissariat des Innern) benutzt wurde.

Beispiel 41 *Bei der* ***Spionage-Chiffre*** *wird in die erste Zeile einer* 3×10*-Matrix ein Schlüsselwort* w *geschrieben, welches keinen Buchstaben mehrfach enthält und eine Länge von 6 bis 8 Zeichen hat (also zum Beispiel* `SPIONAGE`*). Danach werden die anderen beiden Zeilen der Matrix mit den restlichen Klartextbuchstaben (etwa in alphabetischer Reihenfolge) gefüllt.*

	4	1	9	6	0	3	2	7	5	8
	S	*P*	*I*	*O*	*N*	*A*	*G*	*E*		
8	B	C	D	F	H	J	K	L	M	Q
5	R	T	U	V	W	X	Y	Z		

GESPREIZT
$\rightsquigarrow$ *274154795751*

Man überzeugt sich leicht davon, dass sich die von der Spionage-Chiffre generierten Kryptotexte wieder eindeutig dechiffrieren lassen, da die Kryptotextsegmente `1, 2, ..., 8, 01, 02, ..., 08, 91, 92, ..., 98`, die für die Klartextbuchstaben eingesetzt werden, die **Fano-Bedingung** erfüllen: Keines von ihnen bildet den Anfang eines anderen. Da die Nummern `5` und `8` der beiden letzten Spalten der Matrix auch als Zeilennummern verwendet werden, liefert dies auch eine Erklärung dafür, warum keine Schlüsselwortbuchstaben in die beiden letzten Spalten eingetragen werden dürfen.

Verwendung von Blendern und Homophonen

Die Verwendung von gespreizten Chiffren zielt offenbar darauf ab, die „**Fuge**“ zwischen den einzelnen Kryptotextsegmenten, die von unterschiedlichen Klartextbuchstaben herrühren, zu verdecken, um dem Gegner eine unbefugte Dechiffrierung zu erschweren. Dennoch bietet die Spionage-Chiffre noch genügend Angriffsfläche, da im Klartext häufig vorkommende Wortmuster auch im Kryptotext zu Textwiederholungen führen.

Eine Möglichkeit, diese Muster aufzubrechen, besteht darin, Blender in den Klartext einzustreuen. Abgesehen davon, dass das Entfernen der Blender auch für den rechtmäßigen Empfänger mit Mühe verbunden ist, muss für den Zugewinn an Sicherheit auch mit einer Expansion des Kryptotextes bezahlt werden.

Ist man bereit, dies in Kauf zu nehmen, so gibt es auch noch eine wirksamere Methode, die Übertragung struktureller und statistischer Klartextmerkmale auf den Kryptotext abzumildern. Die Idee dabei ist, zur Chiffrierung der einzelnen Klartextzeichen a nicht nur jeweils eines, sondern eine Menge $H(a)$ von Chiffrezeichen vorzusehen, und daraus für jedes Vorkommen von a im Klartext eines auszuwählen (am besten zufällig). Da alle Zeichen in $H(a)$ für dasselbe Klartextzeichen stehen, werden sie auch **Homophone** genannt.

Definition 42 (**homophonen Substitutionschiffre**)

Sei A ein Klartextalphabet und sei $M = A$. Weiter sei C ein Kryptotextraum der Größe $\|C\| > \|A\| = m$. In einer (einfachen) ***homophonen Substitutionschiffre*** beschreibt jeder Schlüssel $k \in K$ eine Zerlegung von C in m disjunkte Mengen $H(a)$, $a \in A$.

Um ein Zeichen $a \in A$ unter k zu chiffrieren, wird nach einer bestimmten Methode ein Homophon y aus der Menge $H(a)$ gewählt und für a eingesetzt.

Durch den Einsatz einer homophonen Substitution wird also erreicht, dass verschiedene Vorkommen eines Klartextzeichens auch auf unterschiedliche Weise ersetzt werden können. Damit der Empfänger den Kryptotext auch wieder eindeutig dechiffrieren kann, dürfen sich die Homophonmengen zweier verschiedener Klartextzeichen aber nicht überlappen. Daher kann es nicht vorkommen, dass zwei verschiedene Klartextbuchstaben durch dasselbe Geheimtextzeichen ersetzt werden. Man beachte, dass der Chiffriervorgang $x \mapsto E(k, x)$ nicht durch eine Funktion beschreibbar ist, da derselbe Klartext x in mehrere verschiedene Kryptotexte y übergehen kann.

Durch eine geringfügige Modifikation der Polybios-Chiffre lässt sich die folgende bipartite homophone Chiffre erhalten.

Beispiel 43 (homophone Substitution)
Sei $A = \{\mathtt{A}, \ldots, \mathtt{Z}\}$, $B = \{\mathit{0}, \ldots, \mathit{9}\}$ *und* $C = \{\mathit{00}, \ldots, \mathit{99}\}$.

M	*1,0*	*2,9*	*3,8*	*4,7*	*5,6*
1,6	A	F	K	P	U
2,7	B	G	L	Q	V
3,8	C	H	M	R	W
4,9	D	I	N	S	X/Y
5,0	E	J	O	T	Z

HOMOPHON $\rightsquigarrow$ *82 03 88 53 17 32 08 98*

Genau wie bei Polybios wird eine 5×5*-Matrix* M *als Schlüssel benutzt. Die Zeilen und Spalten von* M *sind jedoch nicht nur mit jeweils einer, sondern mit zwei Ziffern versehen, so dass jeder Klartextbuchstabe* x *über vier verschiedene Koordinatenpaare ansprechbar ist. Der Kryptotextraum wird durch* M *also in* 25 *Mengen* $H(a)$, $a \in A$, *mit je* 4 *Homophonen partitioniert.*

Wie wir noch sehen werden, sind homophone Chiffrierungen auch deshalb schwerer zu brechen, weil durch sie die charakteristische Häufigkeitsverteilung der Klartextbuchstaben zerstört wird. Dieser Effekt kann dadurch noch verstärkt werden, dass man für häufig vorkommende Klartextzeichen a eine entsprechend größere Menge $H(a)$ an Homophonen vorsieht. Damit lässt sich erreichen, dass die Verteilung der im Geheimtext auftretenden Zeichen weitgehend nivelliert wird.

Beispiel 44 (homophone Substitution, verbesserte Version)
Ist $p(a)$ *die Wahrscheinlichkeit, mit der ein Zeichen* $a \in A$ *in der Klartextsprache auftritt, so sollte* $\|H(a)\| \approx 100 \cdot p(a)$ *sein.*

a	$p(a)$	$H(a)$
A	*0.0647*	{*15, 26, 44, 59, 70, 79*}
B	*0.0193*	{*01, 84*}
C	*0.0268*	{*13, 28, 75*}
D	*0.0483*	{*02, 17, 36, 60, 95*}
E	*0.1748*	{*04, 08, 12, 30, 43, 46, 47, 53, 61, 67, 69, 72, 80, 86, 90, 92, 97*}
⋮	⋮	⋮

Da der Buchstabe A *im Deutschen beispielsweise mit einer Wahrscheinlichkeit von* $p(\mathtt{A}) = 0{,}0647$ *auftritt, sind für ihn sechs verschiedene Homophone vorgesehen.*

Um den Suchaufwand bei der Dechiffrierung zu reduzieren, empfiehlt es sich, eine 10×10-Matrix anzulegen, in der jeder Klartextbuchstabe a an allen Stellen vorkommt, deren Koordinaten in $H(a)$ enthalten sind.

M'	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*0*
1	N	E	C	S	A	O	D	X	I	N
2	R	G	S	N	N	A	U	C	H	Y
3	T	L	I	O	U	D	Z	M	N	E
4	H	R	E	A	N	E	E	S	I	T
5	N	I	E	T	P	H	S	L	A	R
6	E	U	M	F	R	J	E	N	E	D
7	N	E	K	S	C	T	I	T	A	A
8	H	N	I	B	R	E	U	G	V	E
9	T	E	L	S	D	R	E	O	S	E
0	B	D	W	E	Q	I	F	E	I	R

HOMOPHON $\leadsto$ *56 98 63 34 55 29 16 68*

Offenbar kann man diese Matrix auch zur Chiffrierung benutzen, was sogar den positiven Nebeneffekt hat, dass dadurch eine zufällige Wahl der Homophone begünstigt wird.

Transpositionen

Eng verwandt mit der Skytale-Chiffre ist die Zick-Zack-Transposition.

Beispiel 45 *Bei Ausführung einer* ***Zick-Zack-Transposition*** *wird der Klartext in eine Zick-Zack-Linie geschrieben und horizontal wieder ausgelesen. Die Höhe der Zick-Zack-Linie kann als Schlüssel vereinbart werden.*

```
Z       Z       L       E
  I   K   A   K   I   I
    C       C       N
```

ZICKZACKLINIE $\leadsto$ *ZZLEIKAKIICCN*

Bei einer Zick-Zack-Transposition werden Zeichen im vorderen Klartextbereich bis fast ans Ende des Kryptotextes verlagert und umgekehrt. Dies hat den Nachteil, dass für die Generierung des Kryptotextes der gesamte Klartext gepuffert werden muss. Daher werden meist **Blocktranspositionen** verwendet, bei denen die Zeichen nur innerhalb fester Blockgrenzen transponiert werden.

Definition 46 **(Blocktranspositionschiffre)**

> Sei $A = B$ ein beliebiges Alphabet und für eine natürliche Zahl $l \geq 2$ sei $M = C = A^l$. Bei einer ***Blocktranspositionschiffre*** wird durch jeden Schlüssel $k \in K$ eine Permutation π beschrieben, so dass für alle Zeichenfolgen $x_1 \cdots x_l \in M$ und $y_1 \cdots y_l \in C$
>
> $$E(k, x_1 \cdots x_l) = x_{\pi(1)} \cdots x_{\pi(l)}$$
>
> und
>
> $$D(k, y_1 \cdots y_l) = y_{\pi^{-1}(1)} \cdots y_{\pi^{-1}(l)}$$
>
> gilt.

Eine Blocktransposition mit Blocklänge l lässt sich durch eine Permutation $\pi \in S_l$ (also auf der Menge $\{1, \ldots, l\}$) beschreiben.

Beispiel 47 *Eine Skytale, die mit 4 Zeilen der Länge 6 beschrieben wird, realisiert beispielsweise folgende Blocktransposition:*

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
$\pi(i)$	1	7	13	19	2	8	14	20	3	9	15	21	4	10	16	22	5	11	17	23	6	12	18	24

Für die Entschlüsselung muss die zu π **inverse Permutation** π^{-1} benutzt werden.

Beispiel 48

i	1	2	3	4	5	6
$\pi(i)$	4	6	1	3	5	2

i	1	2	3	4	5	6
$\pi^{-1}(i)$	3	6	4	1	5	2

Wird π durch Zyklen $(i_1\ i_2\ i_3\ \ldots\ i_n)$ dargestellt, wobei i_1 auf i_2, i_2 auf i_3 usw. und schließlich i_3 auf i_1 abgebildet wird, so ist π^{-1} sehr leicht zu bestimmen.

Beispiel 49 *Obiges π hat beispielsweise die Zyklendarstellung*

$$\pi = (1\ 4\ 3)\ (2\ 6)\ (5) \text{ oder } \pi = (1\ 4\ 3)\ (2\ 6),$$

wenn, wie allgemein üblich, Einerzklen weggelassen werden. Daraus erhalten wir unmittelbar π^{-1} zu

$$\pi^{-1} = (3\ 4\ 1)\ (6\ 2) \text{ oder } (1\ 3\ 4)\ (2\ 6),$$

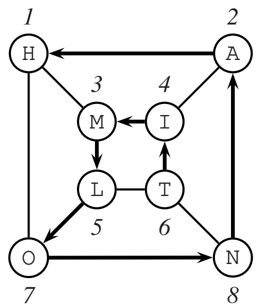
wenn wir jeden Zyklus mit seinem kleinsten Element beginnen lassen und die Zyklen nach der Größe dieser Elemente anordnen.

Beispiel 50 *Bei der **Matrix-Transposition** wird der Klartext zeilenweise in eine $k \times m$-Matrix eingelesen und der Kryptotext spaltenweise gemäß einer Spaltenpermutation π, die als Schlüssel dient, wieder ausgelesen. Für $\pi = (1\ 4\ 3)\ (2\ 6)$ wird also zuerst Spalte $\pi(1) = 4$, dann Spalte $\pi(2) = 6$ und danach Spalte $\pi(3) = 1$ usw. und zuletzt Spalte $\pi(6) = 2$ ausgelesen.*

3	6	4	1	5	2
D	I	E	S	E	R
K	L	A	R	T	E
X	T	I	S	T	N
I	C	H	T	S	E
H	R	L	A	N	G

DIESER KLARTEXT IST NICHT SEHR LANG
⇝ *SRSTA RENEG DKXIH EAIHL ETTSN ILTCR*

Beispiel 51 *Bei der* ***Weg-Transposition*** *wird als Schlüssel eine Hamiltonlinie in einem Graphen mit einer vorgegebenen Knotennummerierung benutzt. (Eine Hamiltonlinie ist eine Anordnung aller Knoten des Graphen, in der je zwei aufeinanderfolgende Knoten durch eine Kante verbunden sein müssen.) Der Klartext wird gemäß der Knotennummerierung in den Graphen eingelesen und der Kryptotext entlang der Hamiltonlinie wieder ausgelesen.*



HAMILTON ⇝ *TIMLONAH*

Es ist leicht zu sehen, dass sich jede Blocktransposition durch eine Hamiltonlinie in einem geeigneten Graphen realisieren lässt. Der Vorteil, eine Hamiltonlinie als Schlüssel zu benutzen, besteht offenbar darin, dass man sich den Verlauf einer Hamiltonlinie bildhaft vorstellen und daher besser einprägen kann als eine Zahlenfolge.

Sehr beliebt ist auch die Methode, eine Permutationen in Form eines **Schlüsselworts** (oder einer aus mehreren Wörtern bestehenden **Schlüsselphrase**) im Gedächtnis zu behalten. Aus einem solchen Schlüsselwort lässt sich die zugehörige Permutation σ leicht rekonstruieren, indem man das Wort auf Papier schreibt und in der Zeile darunter für jeden einzelnen Buchstaben seine Position i innerhalb des Wortes vermerkt.

Schlüsselwort für σ	*C A E S A R*
i	1 2 3 4 5 6
$\sigma(i)$	3 1 4 6 2 5
Zyklendarstellung von σ	(1 3 4 6 5 2)

DIE BLOCKLAENGE IST SECHS ⇝
EDBOIL LCANKE IGSSET EXCSYH

Die Werte $\sigma(i)$, die σ auf diesen Nummern annimmt, werden nun dadurch ermittelt, dass man die Schlüsselwort-Buchstaben in alphabetischer Reihenfolge durchzählt. Dabei werden mehrfach vorkommende Buchstaben gemäß ihrer Position im Schlüsselwort an die Reihe genommen. Alternativ kann man auch alle im Schlüsselwort wiederholt vorkommenden Buchstaben streichen, was im Fall des Schlüsselworts *CAESAR* auf eine Blocklänge von 5 führen würde.

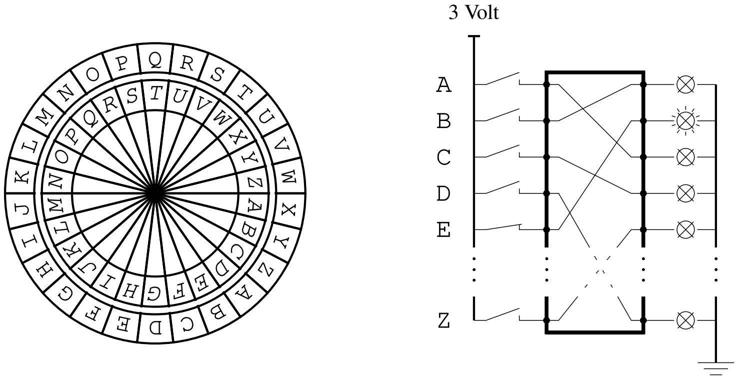
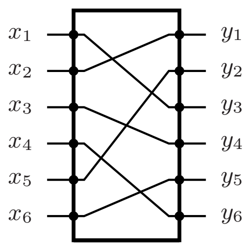


Abbildung 1 Realisierung von einfachen Substitutionen mit einer Drehscheibe und mit Hilfe von Steckverbindungen.

1.8 Realisierung von Blocktranspositionen und einfachen Substitutionen

Abschließend möchten wir eine einfache elektronische Realisierungsmöglichkeit von Blocktranspositionen erwähnen, die auf binär kodierten Klartexten operieren (d.h. $A = \{0, 1\}$). Um einen Binärblock $x_1 \cdots x_l$ der Länge l zu permutieren, müssen die einzelnen Bits lediglich auf l Leitungen gelegt und diese gemäß π in einer sogenannten **Permutationsbox** (kurz **P-Box**) vertauscht werden.



Die Implementierung einer solchen P-Box kann beispielsweise auf einem VLSI-Chip erfolgen. Allerdings kann hierbei für größere Werte von l aufgrund der hohen Zahl von Überkreuzungspunkten ein hoher Flächenbedarf anfallen.

Blocktranspositionen können auch leicht durch Software als eine Folge von Zuweisungen

$$\mathtt{Y1} := \mathtt{X2}; \quad \mathtt{Y2} := \mathtt{X5}; \quad \ldots \quad \mathtt{Y6} := \mathtt{X4};$$

implementiert werden. Bei großer Blocklänge und sequentieller Abarbeitung erfordert diese Art der Implementierung jedoch einen relativ hohen Zeitaufwand.

Von Alberti stammt die Idee, das Klartext- und Kryptotextalphabet auf zwei konzentrischen Scheiben unterschiedlichen Durchmessers anzuordnen. In Abbildung 1 ist gezeigt, wie sich mit einer solchen Drehscheibe beispielsweise die additive Chiffre realisieren lässt. Zur Einstellung des Schlüssels k müssen die Scheiben so gegeneinander verdreht werden, dass der Schlüsselbuchstabe a_k auf der inneren Scheibe mit dem Klartextzeichen $a_0 = \texttt{A}$ auf der äußeren Scheibe zur Deckung kommt. Auf der Drehscheibe in Abbildung 1 ist beispielsweise der Schlüssel $k = 3$ eingestellt, das heißt, $a_k = \textit{D}$. Die Verschlüsselung geschieht nun durch bloßes Ablesen der zugehörigen Kryptotextzeichen auf der inneren Scheibe, so dass von der Drehfunktion der Scheiben nur bei einem Schlüsselwechsel Gebrauch gemacht wird.

Aufgrund ihrer engen Verwandtschaft mit der Klasse der Blocktranspositionen lassen sich einfache Substitutionen auch mit Hilfe einer P-Box realisieren (vergleiche Abbildung). Hierfür können beispielsweise zwei Steckkontaktleisten verwendet werden. Der aktuelle Schlüssel wird in diesem Fall durch Verbinden der entsprechenden Kontakte mit elektrischen Kabeln eingestellt (siehe Abbildung 1). Um etwa den Klartextbuchstaben E zu verschlüsseln, drückt man auf die entsprechende Taste, und das zugehörige Kryptotextzeichen *B* wird im selben Moment durch ein aufleuchtendes Lämpchen signalisiert.

Schließlich lassen sich Substitutionen auch leicht durch Software realisieren. Hierzu wird ein Feld (*array*) deklariert, dessen Einträge über die Klartextzeichen $x \in A$ adressierbar sind. Das mit x indizierte Feldelemente enthält das Kryptotextzeichen, durch welches x beim Chiffriervorgang zu ersetzen ist.

Ein Nachteil hierbei ist, dass das Feld nach jedem Schlüsselwechsel neu beschrieben werden muss. Um dies zu umgehen, kann ein zweidimensionales Feld deklariert werden, dessen Einträge zusätzlich über den aktuellen Schlüsselwert k adressierbar sind. Ist genügend Speicherplatz vorhanden, um für alle $x \in A$ und alle $k \in K$ die zugehörigen Kryptotextzeichen $E(k, x)$ abspeichern zu können, so braucht das Feld nur einmal initialisiert und danach nicht mehr geändert werden.

Schlüssel-wert	**Klartextbuchstabe**			
	A	B	...	Z
0	*U*	*H*	...	*C*
1	*E*	*H*	...	*A*
⋮	⋮	⋮	⋱	⋮
63	*Y*	*F*	...	*W*

Die Tabelle zeigt ein Feld, dessen Einträge mit (zufällig gewählten) Kryptotextzeichen $E(k, x) \in B = \{\textit{A}, \ldots, \textit{Z}\}$ initialisiert wurden, wobei k einen beliebigen Wert in dem Schlüsselraum $K = \{0, 1, \ldots, 63\}$ annehmen kann und x alle Klartextbuchstaben des Alphabets $A = \{\texttt{A}, \ldots, \texttt{Z}\}$ durchläuft.

1.9 Klassifikation von Angriffen gegen Kryptosysteme

Die Erfolgsaussichten eines Angriffs gegen ein Kryptosystem hängen sehr stark davon ab, wie gut die Ausgangslage ist, in der sich der Gegner befindet. Prinzipiell sollte man die Fähigkeiten des Gegners genauso wenig unterschätzen wie die Unvorsichtigkeit der Anwender von Kryptosystemen. Bereits vor mehr als einem Jahrhundert postulierte Kerckhoffs, dass die Frage der Sicherheit keinesfalls von irgendwelchen obskuren Annahmen über den Wissensstand des Gegners abhängig gemacht werden darf.

Goldene Regel für Kryptosystem-Designer (auch „Kerckhoffs' Prinzip" genannt)
Unterschätze niemals den Kryptoanalytiker. Gehe insbesondere immer von der Annahme aus, dass dem Gegner das angewandte System bekannt ist.¶

In der folgenden Liste sind eine Reihe von Angriffsszenarien mit zunehmender Gefährlichkeit aufgeführt. Auch wenn nicht alle Eventualitäten eines Angriffs vorhersehbar sind, so vermittelt diese Aufstellung doch eine gute Vorstellung davon, welchen unterschiedlichen Bedrohungen ein Kryptosystem im praktischen Einsatz ausgesetzt sein kann.

Angriff bei bekanntem Kryptotext (*ciphertext-only attack*)
Der Gegner fängt Kryptotexte ab und versucht, allein aus ihrer Kenntnis Rückschlüsse auf die zugehörigen Klartexte oder auf die benutzten Schlüssel zu ziehen.

Angriff bei bekanntem Klartext (*known-plaintext attack*)
Der Gegner ist im Besitz von einigen zusammengehörigen Klartext-Kryptotext-Paaren. Hierdurch wird erfahrungsgemäß die Entschlüsselung weiterer Kryptotexte oder die Bestimmung der benutzten Schlüssel wesentlich erleichtert.

Angriff bei frei wählbarem Klartext (*chosen-plaintext attack*)
Der Angriff des Gegners wird zusätzlich dadurch erleichtert, dass er in der Lage ist (oder zumindest eine Zeit lang war), sich zu Klartexten seiner Wahl die zugehörigen Kryptotexte zu besorgen. Kann hierbei die Wahl der Kryptotexte in Abhängigkeit von zuvor erhaltenen Verschlüsselungsergebnissen getroffen werden, so spricht man von einem **Angriff bei adaptiv wählbarem Klartext** (***adaptive chosen-plaintext attack***).

Angriff bei frei wählbarem Kryptotext (*chosen-ciphertext attack*)
Vor der Beobachtung des zu entschlüsselnden Kryptotextes konnte sich der Gegner zu Kryptotexten seiner Wahl die zugehörigen Klartexte besorgen, ohne dabei jedoch in den Besitz des Dechiffrierschlüssels zu kommen (**Mitternachtsattacke**). Das dabei erworbene Wissen steht ihm nun bei der Durchführung seines Angriffs zur Verfügung. Auch in diesem Fall können sich die Erfolgsaussichten des Gegners erhöhen, wenn ein **Angriff bei adaptiv wählbarem Kryptotext** (***adaptive chosen-ciphertext attack***) möglich ist, also der Kryptotext in Abhängigkeit von den zuvor erzielten Entschlüsselungsergebnissen wählbar ist.

Angriff bei frei (oder adaptiv) wählbarem Text (*chosen-text attack*)
Sowohl Klartexte als auch Kryptotexte sind frei (oder sogar adaptiv) wählbar.

¶Diese Annahme ergibt sich meist schon aus der Tatsache, dass die Prinzipien fast aller heute im Einsatz befindlichen Kryptosysteme allgemein bekannt sind.

Ohne Frage ist ein Kryptosystem, das bereits bei einem Angriff mit bekanntem Kryptotext Schwächen erkennen lässt, für den praktischen Einsatz vollkommen ungeeignet. Tatsächlich müssen aber an ein praxistaugliches Kryptosystem noch weit höhere Anforderungen gestellt werden. Denn häufig unterlaufen den Anwendern sogenannte **Chiffrierfehler**, die einen Gegner leicht in eine sehr viel günstigere Ausgangsposition versetzen als dies sonst der Fall wäre. So ermöglicht beispielsweise das Auftreten stereotyper Klartext-Formulierungen einen Angriff bei bekanntem Klartext, sofern der Gegner diese Formulierungen kennt oder auch nur errät. Begünstigt durch derartige Unvorsichtigkeiten, die im praktischen Einsatz nicht vollständig vermeidbar sind, können sich selbst winzige Konstruktionsschwächen eines Kryptosystems sehr schnell zu einer ernsthaften Bedrohung der damit verfolgten Sicherheitsinteressen auswachsen. Die Geschichte der Kryptographie belegt sehr eindrucksvoll, dass es häufig die Anwender eines Kryptosystems selbst sind, die – im unerschütterlichen Glauben an seine kryptographische Stärke – dem Gegner zum Erfolg verhelfen.

Zusammenfassend lässt sich also festhalten, dass die Gefährlichkeit von Angriffen, denen ein Kryptosystem im praktischen Einsatz ausgesetzt ist, kaum zu überschätzen ist. Andererseits kann selbst das beste Kryptosystem keinen Schutz vor einer unbefugten Dechiffrierung mehr bieten, wenn es dem Gegner etwa gelingt, in den Besitz des geheimen Schlüssels zu kommen – sei es aus Unachtsamkeit der Anwender oder infolge einer Gewaltandrohung des Gegners (**kompromittierte Schlüssel**).

1.10 Kryptoanalyse von Blocktranspositionen

Mit Hilfe von Bigrammhäufigkeiten, die manchmal auch als Kontakthäufigkeiten bezeichnet werden, lassen sich Blocktranspositionen sehr leicht brechen, sofern genügend Kryptotext vorliegt. Ist die Blocklänge l bekannt, so trägt man hierzu den Kryptotext zeilenweise in eine Matrix $S = (s_{ij})$ mit l Spalten $S_1, \dots, S_l$ ein. Da jede Zeile dieser Matrix aus dem zugehörigen Klartextblock mit derselben Permutation π erzeugt wurde, müssen die Spalten S_j jetzt nur noch in die „richtige" Reihenfolge gebracht werden, um den gesuchten Klartext zu erhalten. Der Nachfolger S_k von S_j (bzw. der Vorgänger S_j von S_k) kann sehr gut anhand der Werte von $\hat{p}(S_j, S_k) = \sum_i p(s_{ij}, s_{ik})$ bestimmt werden.

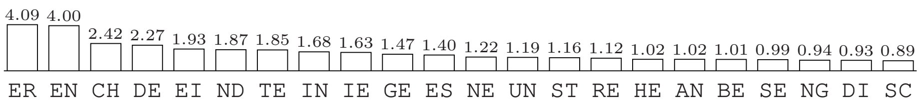


Abbildung 2 Die häufigsten Bigramme im Deutschen (Angaben in %).

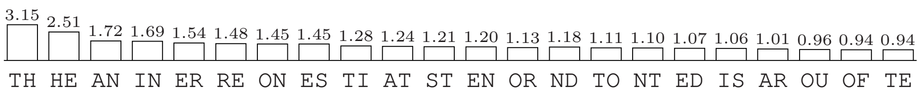


Abbildung 3 Die häufigsten Bigramme im Englischen (in %; nach O.P. Meaker, 1939).

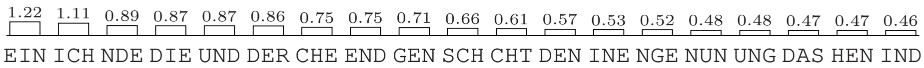


Abbildung 4 Die häufigsten Trigramme im Deutschen (in %).

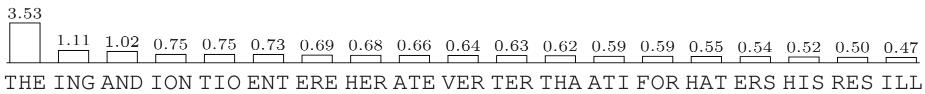


Abbildung 5 Die häufigsten Trigramme im Englischen (in %).

Beispiel 52 (Häufigkeitsanalyse von Bigrammen)
Für den mit einer Blocktransposition (mit vermuteter Blocklänge 5) erzeugten Kryptotext

IHEHR BWEAN RNEII NRKEU ELNZK RXTAE VLOTR ENGIE

erhalten wir eine Matrix S mit den folgenden fünf Spalten.

S_1	S_2	S_3	S_4	S_5
I	H	E	H	R
B	W	E	A	N
R	N	E	I	I
N	R	K	E	U
E	L	N	Z	K
R	X	T	A	E
V	L	O	T	R
E	N	G	I	E

Um die richtige Vorgänger- oder Nachfolgerspalte von S_1 zu finden, bestimmen wir für jede potentielle Spalte S_j, $j = 2, \ldots, 5$, wieviele der Bigramme $s_{ij}s_{i1}$ (bzw. $s_{i1}s_{ij}$) zu den 20 häufigsten (aus Abbildung 2) gehören.

				↓		↓		
S_2	S_3	S_4	S_5	S_1	S_2	S_3	S_4	S_5
H	**E**	H	R	**I**	H	**E**	H	R
W	E	A	N	**B**	W	**E**	A	N
N	**E**	I	I	**R**	N	**E**	I	I
R	K	**E**	**U**	**N**	R	K	**E**	U
L	**N**	Z	K	**E**	L	**N**	Z	K
X	T	A	**E**	**R**	X	T	A	**E**
L	O	T	R	V	L	O	T	R
N	**G**	**I**	E	**E**	**N**	G	**I**	E

Da die beiden Spaltenpaare (S_3, S_1) und (S_1, S_3) jeweils vier häufige Bigramme bilden, können wir annehmen, dass im Klartext S_1 auf S_3 oder S_3 auf S_1 folgen muss. Entscheiden wir uns für die zweite Möglichkeit, so sollten wir als nächstes die Spaltenpaare (S_j, S_1) und (S_3, S_j), $j = 2, 4, 5$ betrachten.

S_2	S_4	S_5	S_1	↓ S_3	S_2	S_4	↓ S_5
H	H	R	I	**E**	H	H	**R**
W	A	N	B	**E**	W	A	**N**
N	I	I	R	**E**	**N**	**I**	**I**
R	**E**	**U**	**N**	K	R	E	U
L	Z	K	E	N	L	Z	K
X	A	**E**	**R**	**T**	X	A	**E**
L	T	R	V	O	L	T	R
N	**I**	E	**E**	**G**	N	I	**E**

Aufgrund des hohen Wertes von $\hat{p}(S_3, S_5)$ können wir annehmen, dass auf S_3 die Spalte S_5 folgt. Im nächsten Schritt erhalten wir daher die folgende Tabelle.

S_2	↓ S_4	↓ S_1	S_3	↓ S_5	↓ S_2	S_4
H	H	I	E	R	H	H
W	A	B	E	N	W	A
N	I	R	E	**I**	**N**	I
R	**E**	**N**	K	U	R	E
L	Z	E	N	K	L	Z
X	A	R	T	E	X	A
L	T	V	O	R	L	T
N	**I**	**E**	G	**E**	**N**	**I**

Diese lässt die Spaltenanordnung S_4, S_1, S_3, S_5, S_2 vermuten, welche tatsächlich auf den gesuchten Klartext führt:

S_4	S_1	S_3	S_5	S_2
H	I	E	R	H
A	B	E	N	W
I	R	E	I	N
E	N	K	U	R
Z	E	N	K	L
A	R	T	E	X
T	V	O	R	L
I	E	G	E	N

1.11 Kryptoanalyse von einfachen Substitutionschiffren

Manche der bisher betrachteten Chiffrierverfahren verwenden einen so kleinen Schlüsselraum, dass ohne großen Aufwand eine vollständige Schlüsselsuche ausgeführt werden kann.

Beispiel 53 (vollständige Schlüsselsuche)
Es sei bekannt, dass das Kryptotextstück $y =$ SAXP mit einer additiven Chiffre erzeugt wurde ($K = A = B = A_{lat}$). Entschlüsseln wir y probeweise mit allen möglichen Schlüsselwerten, so erhalten wir folgende Zeichenketten.

k	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
$D(k, y)$	RZWO	QYVN	PXUM	OWTL	NVSK	MURJ	LTQI	KSPH	JROG	IQNF	HPME	GOLD
N	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
FNKC	EMJB	DLIA	CKHZ	BJGY	AIFX	ZHEW	YGDV	XFCU	WEBT	VDAS	UCZR	TBYQ

Unter diesen springen vor allem die beiden Klartextkandidaten $x =$ GOLD (Schlüsselwert $k =$ M) und $x =$ WEBT ($k =$ W) ins Auge.

Ist $s = \|K\|$ die Größe des Schlüsselraums, so kann der Gegner bei bekanntem Kryptotext y die Suche nach dem zugehörigen Klartext x auf eine Menge von maximal s Texten $x_1, \ldots, x_s$ beschränken. Daneben hat der Gegner ein gewisses *a priori* Wissen über den Klartext, wie zum Beispiel dass er in deutscher Sprache verfasst ist, das es ihm gestattet, einen Großteil der Texte x_i auszuschließen. Ferner erscheinen aufgrund dieses Hintergrundwissens manche der übrig gebliebenen Klartextkandidaten plausibler als andere (sofern nicht nur ein einziger übrig bleibt). Mit jedem Text x_i, der nicht als Klartext in Frage kommt, kann auch mindestens ein Schlüssel ausgeschlossen werden. Sind noch mehrere Schlüsselwerte möglich, so kann weiteres Kryptotextmaterial Klarheit bringen. Manchmal hilft aber auch eine Inspektion der verbliebenen Schlüsselwerte weiter, etwa wenn der Schlüssel nicht rein zufällig erzeugt wurde, sondern aus einem einprägsamen Schlüsselwort ableitbar ist.

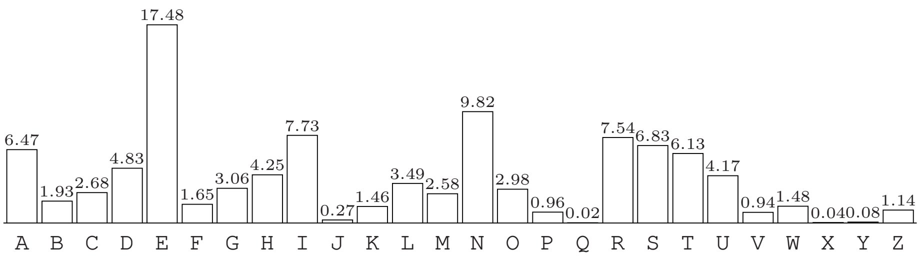


Abbildung 6 Häufigkeitsverteilung von Einzelbuchstaben im Deutschen (in %).

Meist kennt der Gegner zumindest die Sprache, in der der gesuchte Klartext abgefasst ist. Mit zunehmender Länge gleichen sich die Häufigkeitsverteilungen der Buchstaben in natürlichsprachigen Texten einer „Grenzverteilung“ an, die in erster Linie von der benutzten Sprache und nur in geringem Umfang von der Art des Textes abhängt. Diese Verteilungen weisen typischerweise eine sehr starke Ungleichmäßigkeit auf, was darauf zurückzuführen ist, dass in natürlichen Sprachen relativ viel Redundanz enthalten

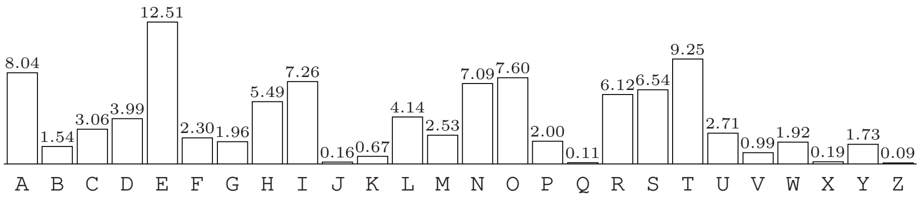


Abbildung 7 Häufigkeitsverteilung von Einzelbuchstaben im Englischen (in %).

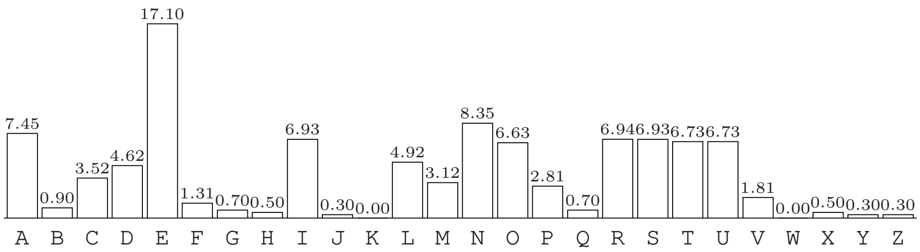


Abbildung 8 Häufigkeitsverteilung von Einzelbuchstaben im Französischen (in %).

ist. Die Abbildungen 6, 7 und 8, zeigen typische Verteilungen von Einzelbuchstaben in der deutschen, englischen und französischen Sprache (ohne Berücksichtigung von Interpunktions- und Leerzeichen). Ein typischer deutscher Text besteht demnach zu 62% aus den sieben häufigsten Zeichen `E`, `N`, `I`, `R`, `S`, `A`, `T` (das sind nicht einmal 27% der Klartextzeichen).

Bei additiven Chiffren reicht es oftmals, den häufigsten Buchstaben im Kryptotext zu bestimmen, und davon den häufigsten Buchstaben der Klartextsprache zu subtrahieren, um den Schlüssel k zu erhalten. Bei affinen Chiffren müssen gewöhnlich nur die beiden häufigsten Buchstaben bestimmt werden; dadurch erhält man zwei Verschlüsselungsgleichungen. Dieses Gleichungssystem muss gelöst werden, und man erhält das gesuchte Schlüsselpaar.

Beispiel 54 (Analyse einer affinen Chiffre mittels Buchstabenhäufigkeiten)
Es sei bekannt, dass sich hinter dem Kryptotext

```
laoea ehoap hwvae ixobg jcbho thlob lokhe ixope vbcix ockix
qoppo boapo mohqc euogk opeho jhkpl eappj seobe ixoap opmcu
```

ein deutscher Klartext verbirgt, der mit einer additiven Chiffre verschlüsselt wurde. Berechnen wir für jedes Chiffrezeichen y die (absolute) Häufigkeit $H(y)$ seines Auftretens,

y	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
$H(y)$	7	6	5	0	10	0	2	8	5	3	4	4	2	0	19	11	2	0	1	1	2	2	1	5	0	0

so liegt die Vermutung nahe, dass das am häufigsten vorkommende Chiffrezeichen `P` *für das Klartextzeichen* `E` *und das am zweithäufigsten vorkommende* `O` *für* `N` *steht. Unter* *dieser Annahme kann der gesuchte Schlüssel $k = (b, c)$ als Lösung der beiden Gleichungen*

$$\begin{aligned} b \cdot \mathtt{E} + c &= \mathit{O} \\ b \cdot \mathtt{N} + c &= \mathit{P} \end{aligned}$$

bestimmt werden. Subtrahieren wir nämlich die erste von der zweiten Gleichung, so erhalten wir die Kongruenz $9 \cdot b \equiv_{26} 1$, woraus sich $b = 3$ und damit $c = 2$ ergibt. Tatsächlich weist der Schlüssel $k = (3, 2)$ nicht nur für die beiden Paare $(\mathtt{E}, \mathit{O})$ und $(\mathtt{N}, \mathit{P})$, sondern auch für alle übrigen Paare (x, y) eine gute Übereinstimmung zwischen der Häufigkeit $H(y)$, mit der $y = E(k, x)$ im Kryptotext vorkommt, und der erwarteten Häufigkeit $E(x)$ auf, mit der x in einem typischen deutschen Text der Länge 100 vorkommt (die Tabelle zeigt die Werte von $E(x)$ gerundet):

x	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
$E(x)$	6	2	3	5	17	2	3	4	8	0	1	3	3	10	3	1	0	8	7	6	4	1	1	0	0	1
$y = E(k, x)$	*C*	*F*	*I*	*L*	*O*	*R*	*U*	*X*	*A*	*D*	*G*	*J*	*M*	*P*	*S*	*V*	*Y*	*B*	*E*	*H*	*K*	*N*	*Q*	*T*	*W*	*Z*
$H(y)$	5	0	5	4	19	0	2	5	7	0	2	3	2	11	1	2	0	6	10	8	4	0	2	1	1	0

Durch eine Häufigkeitsanalyse können insbesondere einfache Substitutionen g leicht gebrochen werden, sofern die einzelnen Buchstaben a in der benutzten Klartextsprache mit voneinander differierenden Häufigkeiten $p(a)$ auftreten (vergleiche Tabelle 2). Selbst wenn, was insbesondere bei kurzen Texten zu erwarten ist, die tatsächliche Häufigkeitsverteilung nur in etwa der vom Gegner angenommenen Verteilung entspricht, reduziert sich dadurch die Zahl der in Frage kommenden einfachen Substitutionen ganz erheblich. Berechnet man die relativen Häufigkeiten h der Kryptotextbuchstaben im Kryptotext, so gilt $p(a) \approx h(g(a))$ (vorausgesetzt der Kryptotext ist genügend lang). Für die Schilderung einer nach dieser Methode durchgeführten Kryptoanalse sei auf die Erzählung „Der Goldkäfer" von Edgar Allan Poe verwiesen.

Tabelle 2 Einteilung von Buchstaben in Cliquen mit vergleichbaren Häufigkeitswerten.

	Deutsch	Englisch	Französisch
sehr häufig	E	E	E
häufig	N | I R S | A T	T | A O I N | S R H	N | A R S I T U
durchschnittlich	D H U | L G O | C M	L D | C U M F	L D | C M P
selten	B F W K Z | P V	P G W Y B | V K	V | F B G Q H X
sehr selten	J Y X Q	X J Q Z	J Y Z K W

1.12 Kryptoanalyse von polygraphischen Chiffren

Blocksysteme mit kleinem k (beispielsweise bigraphische Systeme) lassen sich ähnlich wie einfache Substitutionen durch Häufigkeitsanalysen brechen. Wird bei Hill-Chiffren k sehr groß gewählt, so ist eine solche statistische Analyse nicht mehr möglich. Das Hill-System kann dann zwar einem Kryptotextangriff widerstehen, jedoch kaum einem

Angriff mit bekanntem Klartext und schon gar nicht einem Angriff mit gewähltem Klartext.

Angriff mit gewähltem Klartext O. B. d. A. sei $A = \{0, 1, \ldots, m-1\}$. Bei einem GK-Angriff verschafft sich der Gegner den Kryptotext zu $100\ldots0$, $010\ldots0$, $\ldots$, $0\ldots001 \in A^l$:

$$\begin{array}{rcl} g(100\ldots0) & = & k_{1\,1}\,k_{1\,2}\ldots k_{1\,l} \\ g(010\ldots0) & = & k_{2\,1}\,k_{2\,2}\ldots k_{2\,l} \\ & \vdots & \\ g(0\ldots001) & = & k_{l\,1}\,k_{l\,2}\ldots k_{l\,l} \end{array}$$

und erhält damit die Schlüsselmatrix k.

BK-Angriff (bekannter Klartext). Sind bei einem BK-Angriff ausreichend geeignete Klartext-Kryptotextpaare bekannt, so kann das Hill-System folgendermaßen gebrochen werden: Sind x_i, y_i $(i = 1, \ldots, \mu)$ Paare mit $x_i k = y_i$ und gilt $\mathrm{ggT}(\det X, m) = 1$ für eine aus l Blöcken x_i, $i \in I$, als Zeilen gebildete Matrix X, so lässt sich die Schlüsselmatrix k zu $k = YX^{-1}$ bestimmen (Y ist die aus den Blöcken y_i, $i \in I$, gebildete Matrix).

1.13 Kryptoanalyse von polyalphabetischen Chiffren

Die Vigenère-Chiffre galt bis ins 19. Jahrhundert als sicher. Da der Schlüsselstrom bei der Vigenère-Chiffre periodisch ist, lassen sie sich mit statistischen Methoden ebenfalls leicht brechen, insbesondere wenn der Kryptotext im Verhältnis zur Periode d (Länge des Schlüsselwortes) genügend lang ist.

Bestimmung der Schlüsselwortlänge

Wir haben gesehen, dass es recht einfach ist, eine Vigenère-Chiffre zu brechen, wenn man die Länge des Schlüsselwortes hat. Diese ist im allgemeinen aber nicht bekannt, so dass man vor der Anwendung des eben beschriebenen Verfahrens zuerst einmal die Schlüsselwortlänge in Erfahrung bringen muss. Dazu betrachten wir zwei Vorgehensweisen: den Kasiski-Test und die Koinzidenzindex-Untersuchung.

Der Kasiski-Test. Die früheste generelle Methode zur Bestimmung der Periode bei der Vigenère-Chiffre stammt von Friedrich W. Kasiski (1860). Kommt ein Wort an zwei verschiedenen Stellen im Kryptotext vor, so kann es sein, dass die gleiche Klartextsequenz zweimal auf die gleiche Weise, d. h. mit der gleichen Schlüsselsequenz, verschlüsselt wurde. In diesem Fall ist die Entfernung δ der beiden Vorkommen ein Vielfaches der Periode d. Werden mehrere Paare mit verschiedenen Entfernungen δ_i

gefunden, so liegt die Vermutung nahe, dass d gemeinsamer Teiler aller (oder zumindest vieler) δ_i ist, was die Anzahl der noch in Frage kommenden Werte für d stark einschränkt.

Beispiel 55 (Kasiski-Test)

$$\begin{array}{rll} & \texttt{DERERSTEUNDLETZTEVERS...} & \textit{(Klartext } x\textit{)} \\ + & \underline{\texttt{KASKASKASKASKASKASKAS...}} & \textit{(Schlüsselstrom } \hat{k}\textit{)} \\ & \texttt{NEJORKDEMXDDOTRDENORK...} & \textit{(Kryptotext } y\textit{)} \end{array}$$

Dass die Textstücke `ORK`*, bzw.* `DE` *im Kryptotext in den Entfernungen* $\delta_1 = 15$ *und* $\delta_2 = 9$ *vorkommen, führt auf die richtige Vermutung, dass die Periode* $d = \mathrm{ggT}(9, 15) = 3$ *ist.*

Koinzidenzindex-Untersuchungen. Zur Bestimmung der Periode d gibt es neben heuristischen Methoden auch folgenden statistischen Ansatz, der erstmals von William Frederick Friedman im Jahr 1920 beschrieben wurde. Er basiert auf der Beobachtung, dass eine längere Periode eine zunehmende *Glättung* der Buchstabenhäufigkeiten im Kryptotext bewirkt.

Definition 56 **(Koinzidenzindex)**

Der ***Koinzidenzindex*** (engl. ***i****ndex of* ***c****oincidence*) eines Textes y der Länge n über dem Alphabet $\mathcal{B}$ ist definiert als

$$IC(y) = \frac{1}{n \cdot (n-1)} \cdot \sum_{a \in \mathcal{B}} H(a) \cdot (H(a) - 1).$$

Hierbei ist $H(a)$ die absolute Häufigkeit des Buchstabens a im Text y.

$IC(y)$ gibt also die Wahrscheinlichkeit an, mit der man im Text y an zwei zufällig gewählten Positionen den gleichen Buchstaben vorfindet. Er ist umso größer, je ungleichmäßiger die Häufigkeiten $H(a)$ sind (siehe unten).

Um die Periode d einer Vigenère-Chiffre zu bestimmen, schreibt man den Kryptotext y für $d = 1, 2, 3, \ldots$ in eine Matrix mit d Spalten und berechnet für jede Spalte y_i den Koinzidenzindex. Für genügend lange Kryptotexte ist dasjenige d, welches das maximale arithmetische Mittel der Spaltenindizes liefert mit hoher Wahrscheinlichkeit die gesuchte Periode.

Ist der Koinzidenzindex IC_L der Klartextsprache bekannt, so kann der Suchraum für den Wert der Periode d erheblich eingeschränkt werden. Da der Erwartungswert $\mathrm{E}(IC)$ des Koinzidenzindexes für den Kryptotext stark von der verwendeten Periode abhängt, kann nämlich die Periode d in erster Näherung durch Vergleich von $IC(y)$ mit dem Erwartungswert $\mathrm{E}(IC)$ abgeschätzt werden. Untenstehende Tabelle gibt den Erwartungswert $\mathrm{E}(IC)$ des Koinzidenzindexes für Kryptotexte der Länge $n = 100$ in Abhängigkeit von der Periodenlänge d einer Vigenère-Chiffre wieder (Klartext ist ein zufällig gewählter Text der englischen Sprache).

d	1	2	3	4	5	6	8	10	∞
$\mathrm{E}(IC)$	0,0687	0,0535	0,0484	0,0459	0,0444	0,0434	0,0422	0,0414	0,0389

Beispiel 57 *Berechnet sich der Koinzidenzindex eines vorliegenden Kryptotextes (mit der Länge* 100*) zu* $0{,}045$*, so liegt die Vermutung nahe, dass das verwendete Schlüsselwort die Länge vier oder fünf hat.*

Allgemein gilt

$$\mathrm{E}(IC) = \frac{n-d}{d\cdot(n-1)}\cdot IC_L + \frac{n\cdot(d-1)}{d\cdot(n-1)}\cdot {}^{1}\!/\!_{\|A\|},$$

wobei IC_L der Koinzidenzindex der Klartextsprache L über dem Alphabet A ist (ohne Beweis).

Definition 58 (**Koinzidenzindex einer Sprache**)

Der ***Koinzidenzindex*** IC_L ***einer Sprache*** mit Buchstabenverteilung $p: A \to [0,1]$ ist definiert als

$$IC_L = \sum_{a\in A} p(a)^2.$$

IC_L ist die Wahrscheinlichkeit, in einem zufälligen Text der Sprache L an zwei zufällig gewählten Positionen den gleichen Buchstaben vorzufinden, während $IC(x)$ die Wahrscheinlichkeit für das Auftreten dieses Ereignisses im Text x ist.

Man beachte, dass der Koinzidenzindex bei Transpositionschiffren sowie bei einfachen Substitutionen erhalten bleibt. Erstere lassen sich von letzteren dadurch unterscheiden, dass bei ihnen sogar die Buchstabenhäufigkeiten unverändert bleiben.

IC_L ist zudem ein Maß für die Rauheit der Verteilung p:

Definition 59 (**Rauheitsgrad; Measure of Roughness**)

Der ***Rauheitsgrad*** MR_L einer Sprache L mit Einzelbuchstabenverteilung p ist

$$MR_L = \sum_{a\in A}\left(p(a) - {}^{1}\!/\!_{m}\right)^2 = \sum_{a\in A} p(a)^2 - {}^{1}\!/\!_{m} = IC_L - {}^{1}\!/\!_{m},$$

wobei $m = \|A\|$ ist.

Beispiel 60 *Für die englische Sprache (*$m = 26$*) gilt beispielsweise* $IC_{\mathrm{Englisch}} \approx 0{,}0687$ *und* $MR_{\mathrm{Englisch}} \approx 0{,}0302$.

Ist d bekannt, so schreibt man den Kryptotext in Blöcke der (Zeilen-) Länge d untereinander. Verfahrensbedingt wurden die einzelnen Spalten $y_1,\ldots,y_d$ durch eine monoalphabetische Substitution (genauer: durch eine Verschiebechiffre) verschlüsselt. Sie können daher einzeln wie eine additive Chiffre gebrochen werden (Häufigkeitsanalyse); jede Spalte y_i liefert einen Buchstaben k_i des Schlüsselwortes der Vigenère-Chiffre.

Zur Bestimmung des Schlüsselwortes bei bekannter Periode d kann auch wie folgt vorgegangen werden. Man schreibt den Kryptotext y in Spalten y_i auf und berechnet für $a \in A$ und $i = 1, \ldots, d$ die relativen Häufigkeiten $h_i(a)$ von a in y_i. Da y_i aus dem Klartext durch Addition von k_i entstanden ist, kommt die Verteilung

$$h_i(a+k), a \in A$$

für $k = k_i$ der Klartextverteilung $p(a), a \in A$ näher als für $k \neq k_i$. Da

$$\alpha_i(k) := \sum_{a \in A} p(a) h_i(a+k)$$

ein Maß für die Ähnlichkeit der beiden Verteilungen $p(a)$ und $h_i(a+k)$ ist (siehe Übungen), wird der Wert von $\alpha_i(k)$ wahrscheinlich für $k = k_i$ maximal werden.

Beispiel 61 *Der folgende Kryptotext* y

```
HUDS KUAE ZGXR AVTF PGWS WGWS ZHTP PBIL LRTZ PZHW
LOIJ VFIC VBTH LUGI LGPR KHWM YHTI UAXR BHTW UCGX
OSPW AOCH IMCS YHWQ HWCF YOCG OGTZ LBIL SWBF LOHX
ZWSI ZVDS ATGS THWI SSUX LMTS MHWI KSPX OGWI HRPF
LSAM USUV VAIL LHGI LHWV VIVL AVTW OCIJ PTIC MSTX
VII
```

der Länge 203 *wurde von einer Vigenère-Chiffre mit Schlüssellänge* $d = 4$ *aus englischem Klartext erzeugt. Schreiben wir den Kryptotext in vier Spalten* $y_1, \ldots, y_4$ *der Länge* $|y_1| = |y_2| = |y_3| = 51$ *und* $|y_4| = 50$*, so ergeben sich folgende Werte für* $\alpha_i(k)$*.*

k	$\alpha_1(k)$	$\alpha_2(k)$	$\alpha_3(k)$	$\alpha_4(k)$
0	0,0367	0,0445	0,0478	0,0381
1	0,0319	0,0411	0,0417	0,0405
2	0,0315	0,0407	0,0480	0,0270
3	0,0458	0,0514	0,0371	0,0412
4	0,0380	0,0410	0,0496	**0,0650**
5	0,0261	0,0319	0,0400	0,0474
6	0,0422	0,0372	0,0350	0,0288
7	**0,0735**	0,0433	0,0302	0,0344
8	0,0448	0,0347	0,0485	0,0390
9	0,0267	0,0283	0,0321	0,0339
10	0,0362	0,0362	0,0257	0,0352
11	0,0474	0,0266	0,0428	0,0364
12	0,0306	0,0288	0,0312	0,0307
13	0,0321	0,0431	0,0265	0,0306
14	0,0360	**0,0681**	0,0436	0,0486
15	0,0292	0,0451	**0,0769**	0,0449
16	0,0283	0,0354	0,0377	0,0359
17	0,0394	0,0277	0,0317	0,0424
18	0,0487	0,0424	0,0392	0,0476
19	0,0421	0,0433	0,0453	0,0384
20	0,0425	0,0408	0,0354	0,0397
21	0,0390	0,0359	0,0346	0,0348
22	0,0424	0,0306	0,0371	0,0275
23	0,0425	0,0246	0,0263	0,0380
24	0,0355	0,0314	0,0303	0,0367
25	0,0310	0,0459	0,0256	0,0370

Da $\alpha_1(k)$ für $k = 7 =$ `H`*, $\alpha_2(k)$ für $k = 14 =$* `O`*, $\alpha_3(k)$ für $k = 15 =$* `P` *und $\alpha_4(k)$ für $k = 4 =$* `E` *einen Maximalwert annimmt, lautet das Schlüsselwort* `HOPE`*. Damit ergibt sich folgender Klartext*

```
A GOOD GLASS IN THE BISHOPS HOSTEL IN THE DEVILS
SEAT FORTYONE DEGREES AND THIRTEEN MINUTES NOR-
THEAST AND BY NORTH MAIN BRANCH SEVENTH LIMB EAST
SIDE SHOOT FROM THE LEFT EYE OF THE DEATHS HEAD
A BEE LINE FROM THE TREE THROUGH THE SHOT FIFTY
FEET OUT
```

Zur Bestimmung des Schlüsselwortes kann man auch die Methode des *gegenseitigen Koinzidenzindexes* verwenden. Dabei ist die verwendete Klartextsprache (und somit deren Häufigkeitsverteilung) irrelevant, da die Spalten – wie der Name schon sagt – gegenseitig in Relation gesetzt werden. Aber zuerst die Definition.

Definition 62 **(Gegenseitiger Koinzidenzindex)**

Der ***gegenseitge Koinzidenzindex*** von zwei Texten y und y' mit den Längen n und n' über dem Alphabet $\mathcal{B}$ ist definiert als

$$IC(y, y') = \frac{1}{n \cdot n'} \cdot \sum_{a \in \mathcal{B}} H(a) \cdot H'(a).$$

Hierbei ist $H(a)$ bzw. $H'(a)$ die absolute Häufigkeit des Buchstabens a im Text y bzw. y'.

$IC(y, y')$ ist also die Wahrscheinlichkeit, dass bei zufälliger Wahl einer Position in y und einer Position in y' der gleiche Buchstabe vorgefunden wird. $IC(y, y')$ ist umso größer, je besser die Häufigkeitsverteilung von y und y' (d. h. H und H') übereinstimmen.

Ist nun y ein Kryptotext, der mit einem Schlüsselwort bekannter Länge d erzeugt wurde, und sind y_i, $i = 1, \ldots, d$ die zugehörigen Spalten, so gibt der gegenseitige Koinzidenzindex der Spalten y_i und $y_j + \delta$ (für $1 \leq i < j \leq d$) die Wahrscheinlichkeit an, dass man bei zufälliger Wahl einer Position in y_i und in $y_j + \delta$ denselben Buchstaben vorfindet, wobei δ eine Verschiebung von Spalte y_j relativ zur Spalte y_i ist (mit $0 \leq \delta \leq 25$).

Mit großer Wahrscheinlichkeit nimmt also $IC(y_i, y_j + \delta)$ für $\delta = \delta_{ij} = k_j - k_i$ einen relativ großen Wert an, während für $\delta \neq \delta_{ij}$ mit kleinen Werten zu rechnen ist.

Beispiel 63 *Betrachten wir den Kryptotext aus vorigem Beispiel, so ergeben sich folgende Werte für $IC(y_i, y_j + \delta)$.*

Δ	$IC(\dot{c}_1, \dot{c}_2)$	$IC(\dot{c}_1, \dot{c}_3)$	$IC(\dot{c}_1, \dot{c}_4)$	$IC(\dot{c}_2, \dot{c}_3)$	$IC(\dot{c}_2, \dot{c}_4)$	$IC(\dot{c}_3, \dot{c}_4)$
0	0,040369	0,026913	0,038039	0,050750	0,046275	0,049804
1	0,031526	0,047290	0,040000	**0,085736**	0,053725	0,036863
2	0,025375	0,025759	0,029412	0,049596	0,040392	0,038039
3	0,038062	0,021146	0,031373	0,021530	0,037255	0,060784
4	0,025375	0,047674	0,035686	0,028451	0,051765	0,036863
5	0,021530	0,032680	0,024314	0,035755	0,042353	0,025098
6	0,046521	0,018070	0,032941	0,024990	0,029804	0,034510
7	**0,074587**	0,049596	0,058431	0,034218	0,023529	0,019216
8	0,050365	**0,091503**	0,042745	0,046905	0,024314	0,029020
9	0,033833	0,042291	0,032549	0,025375	0,032549	0,042745
10	0,031526	0,027297	0,044314	0,024606	0,040784	0,041569
11	0,044214	0,051519	0,050588	0,027297	0,055294	0,033333
12	0,043060	0,045367	0,043922	0,059208	0,038039	0,054510
13	0,034987	0,031911	0,039608	0,050365	0,031373	0,027843
14	0,031911	0,029604	0,031373	0,050365	0,032941	0,036078
15	0,028451	0,032295	0,020784	0,053057	0,045098	**0,078824**
16	0,024606	0,023837	0,034510	0,051134	**0,067843**	0,047843
17	0,031526	0,029220	0,036863	0,024221	0,049412	0,025490
18	0,044214	0,027682	0,030980	0,022299	0,025098	0,029804
19	0,045367	0,039600	0,040000	0,026528	0,027451	0,033725
20	0,037678	0,045752	0,045490	0,043829	0,029804	0,027059
21	0,048443	0,046136	0,024706	0,036909	0,029412	0,028627
22	0,064975	0,039600	0,042745	0,035755	0,034510	0,047059
23	0,044214	0,058439	**0,078431**	0,032680	0,037255	0,032941
24	0,025375	0,044214	0,047843	0,024221	0,038431	0,027843
25	0,031911	0,024606	0,022353	0,034218	0,035294	0,054510

Also ist (mit großer Wahrscheinlichkeit)

$$\Delta_{1\,2} = 7,\ \Delta_{1\,3} = 8,\ \Delta_{1\,4} = 23,\ \Delta_{2\,3} = 1,\ \Delta_{2\,4} = 16,\ \Delta_{3\,4} = 15.$$

Wir können nun alle Spalten relativ zur ersten Spalte so verschieben, dass der ganze Text eine einheitliche Verschiebung Δ hat, also die zweite Spalte um -7, die dritte um -8 und die vierte um -23. Für die Bestimmung von Δ, muss man nur den häufigsten Buchstaben in dem auf diese Weise erzeugten Text bestimmen. Dieser ist L *$(16,3\%)$. Also ist $\Delta = \mathtt{L} - \mathtt{E} = H = 7$ und das Schlüsselwort lautet HOPE ($H + 7 = O$, $H + 8 = P$, $H + 23 = E$).*

Analyse der Lauftextverschlüsselung

Zum Brechen einer Stromchiffre mit Klartextschlüsselstrom kann man so vorgehen: Man geht zunächst davon aus, dass jeder Kryptotextbuchstabe durch Summation eines Klartext- und Schlüsselstrombuchstabens mit jeweils mittlerer bis hoher Wahrscheinlichkeit entstanden ist. Dies sind beispielsweise im Englischen die Buchstaben E, T, A, O, I, N, S, R, H. Zu einem Teilwort w des Kryptotextes bestimmt man dann alle Paare von Wörtern $(w_1,\ w_2)$ mit $w_1 + w_2 = w$ und $w_1, w_2 \in \{\mathtt{E}, \mathtt{T}, \mathtt{A}, \mathtt{O}, \mathtt{I}, \mathtt{N}, \mathtt{S}, \mathtt{R}, \mathtt{H}\}$. In der Regel ergeben sich nur sehr wenige sinnvolle Paare, aus denen durch Kontextbetrachtungen und Erweitern von w nach links und rechts der Kryptotext entschlüsselt werden kann. Wird die Analyse durch ein Computerprogramm durchgeführt, kann an die Stelle der Kontextbetrachtungen auch die Häufigkeitsverteilung von n-Grammen der Sprache treten. Das Programm wählt dann solche Wortpaare (w_1, w_2), die eine hohe Wahrscheinlichkeit haben.

Beispiel 64 *Gegeben ist der Kryptotext MOQKTHCBLMWXF... Wir beginnen die Untersuchung mit einer Wortlänge von vier Buchstaben, also w = MOQK. Der erste Buchstabe M kann nur auf eine der folgenden Arten zustande gekommen sein:*

$$\begin{array}{rll} & \mathtt{ABCDE...I...T...Z} & \textit{(Klartextzeichen)} \\ + & \mathtt{MLKJI...E...T...N} & \textit{(Schlüsselzeichen)} \\ \hline = & \mathtt{MMMMM...M...M...M} & \textit{(Kryptotextzeichen)} \end{array}$$

Es ergeben sich als wahrscheinliche Paare für die Einzelbuchstaben von w:

M:	(E,*I*)	*O:*	(A,*O*)	*Q:*	(I,*I*)	*K:*	(R,*T*)
	(I,*E*)		(H,*H*)				(S,*S*)
	(T,*T*)		(O,*A*)				(T,*R*)

und damit (w_1, w_2) zu

w_1	EAIR	EAIS	EAIT	EHIR	...	THIS	...	TOIT
w_2	*IOIT*	*IOIS*	*IOIR*	*IHIT*	...	*THIS*	...	*TAIR*

Als sinnvoll stellt sich also nur die Wahl $w_1 = w_2 =$ THIS *heraus.*

Autokey Chiffren

Kryptotextschlüsselstrom. Diese Systeme bieten eigentlich keinen großen kryptographischen Schutz, da sie ohne Kenntnis des Schlüsselwortes sehr leicht entschlüsselt werden können (falls die Länge des Schlüsselwortes im Verhältnis zur Länge des Kryptotextes relativ kurz ist). Man subtrahiert dazu den Kryptotext y für $d = 1, 2, \ldots$ von dem um d Positionen verschobenen Kryptotext – also $y_{1+\delta}\, y_{2+\delta}\, y_{3+\delta} \ldots$ minus $y_1\, y_2\, y_3 \ldots$ –, bis sinnvoller (Klar-) Text erscheint:

$$\begin{array}{rrl} & \mathtt{DUMSQMOZKFN...} & \text{(Kryptotext } y\text{)} \\ - & \mathtt{DUMSQMO...} & \text{(„Kryptotextschlüsselstrom“)} \\ \hline = & \mathtt{....NSCHUTZ...} & \text{(Klartext } x\text{)} \end{array}$$

Klartextschlüsselstrom. Neben der oben beschriebenen Analyse der Lauftextverschlüsselung kann das Brechen der *Autokey*-Systeme mit Klartextschlüsselstrom auch analog zur Kasiski-Methode erfolgen: Sei d die Länge des Schlüsselwortes $k_1 \dots k_d$. Falls im Klartext die gleiche Buchstabenfolge $x_i \dots x_{i+\delta-1}$ im Abstand $2d$ auftritt (beispielsweise $d = 3$ und $\delta = 2$),

$$\begin{array}{c|cccccccccccccccl}
 & & & & & \downarrow & \downarrow & & & & & \downarrow & \downarrow & & & & \\
 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_5 & x_6 & x_{11} & x_{12} & \dots & \text{(Klartext } x\text{)} \\
+ & k_1 & k_2 & k_3 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_5 & \dots & \text{(Klartextschlüsselstrom } km\text{)} \\
\hline
= & y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 & y_{10} & y_8 & y_9 & y_{11} & y_{12} & \dots & \text{(Kryptotext } y\text{)}
\end{array}$$

so tritt im Kryptotext die gleiche Buchstabenfolge im Abstand d auf, d. h. d kann auf diese Art unter Umständen leicht bestimmt werden. Ist d bekannt, so können die Buchstaben $k_1 \dots k_d$ des Schlüsselwortes der Reihe nach bestimmt werden: Da durch k_i die Klartextzeichen an den Positionen $i,\ d+i,\ 2d+i,\ \dots$ eindeutig festgelegt sind, kann jedes einzelne k_i unabhängig von den anderen Schlüsselwortbuchstaben durch eine statistische Analyse bestimmt werden.

Bei der dritten Methode *Autokey*-Chiffren mit Klartextschlüsselstrom zu brechen, „rechnet man den Schlüssel aus dem Kryptotext heraus“. Doch zuerst einige allgemeine Betrachtungen zu den Eigenschaften einer *Autokey*-Chiffre mit Klartextschlüsselstrom.

Betrachten wir den Klartext `HIERKOMMTEINMUSTER`, welcher mit dem Schlüssel *`DER`* verschlüsselt werden soll:

$$\begin{array}{cll}
 & \mathtt{HIERKOMMTEINMUSTER} & \text{(Klartext)} \\
+ & \underline{\mathit{DER}}\mathit{HIERKOMMTEINMUS} & \text{(Schlüsselstrom)} \\
\hline
= & \mathit{KMVYSSDWHQUGQCFFYJ} & \text{(Kryptotext)}
\end{array}$$

Für eine kryptoanalytische Untersuchung des Geheimtextes nehmen wir nun an, dass die Schlüsselwortlänge $d = 3$ bekannt ist und dechiffrieren den Kryptotext mit einem beliebigen Schlüssel dieser Länge (beispielsweise *`AAA`*).

$$\begin{array}{cll}
 & \mathit{KMVYSSDWHQUGQCFFYJ} & \text{(Kryptotext)} \\
- & \underline{\mathit{AAA}}\mathit{KMVOGXPQKBEWPYJ} & \text{(Schlüsselstrom)} \\
\hline
= & \mathtt{KMVOGXPQKBEWPYJQAA} & \text{(modifizierter Klartext)}
\end{array}$$

Der modifizierte Klartext wird nun mit dem ursprünglichen Klartext verglichen. Dabei fällt auf, dass sich die Differenzen der Buchstaben im Abstand der doppelten Schlüsselwortlänge wiederholen:

h	i	e	r	k	o	m	m	t	e	i	n	m	u	s	t	e	r
K	M	V	O	G	X	P	Q	K	B	E	W	P	Y	J	Q	A	A
3	4	17	23	22	9	3	4	17	23	22	9	3	4	17	23	22	9

Diese Wiederholung besagt aber nichts anderes, als dass bei dem modifizierten Klartext eine gewöhnliche Vigenère-Chiffre vorliegt, die mit einem Schlüssel der Länge $2d$ chiffriert wurde.

Als kryptoanalytische Vorgehensweise ergibt sich daraus folgende Strategie:

Da die Länge d des Schlüsselwortes nicht bekannt ist, erzeugt man (wie oben beschrieben) unter Verwendung von unterschiedlichen Werten für $d = 1, 2, 3, \ldots$ aus dem Kryptotext eine Anzahl modifizierter Klartexte, die dann einer Häufigkeitsanalyse mit *doppelter* Schlüssellänge unterzogen werden. Bei dieser Untersuchung sind nur die ersten d Stellen von Bedeutung; sie ergeben das Schlüsselwort, mit dem der ursprünglich gegebene Kryptotext entschlüsselt werden kann.[‖]

1.14 Informationstheoretische Sicherheitsanalyse

Claude E. Shannon untersuchte die Sicherheit kryptographischer Systeme auf informationstheoretischer Basis (1945, freigegeben 1949). Seinen Untersuchungen liegt das Modell einer Nachrichtenquelle zugrunde, die einzelne Nachrichten unter einer bestimmten Wahrscheinlichkeitsverteilung aussendet.

Bei der Betrachtung der informationstheoretischen Eigenschaften von Kryptosystemen gehen wir von einer Wahrscheinlichkeitsverteilung auf den Paaren $(k, x) \in K \times M$ aus, d. h. $p(k, x)$ gibt die Wahrscheinlichkeit an, dass der Klartext x mit dem Schlüssel k verschlüsselt wird. Dabei setzen wir voraus, dass nach jeder Verschlüsselung einer Nachricht x ein neuer Schlüssel gewählt wird. Dies bedeutet, dass beispielsweise bei der additiven Chiffre für $M = A^n$ zu setzen ist (und nicht $M = A$ wie in Definition 4), falls mit dem selben Schlüssel eine Folge von n Buchstaben chiffriert wird.

Weiterhin nehmen wir an, dass der Schlüssel unabhängig vom Klartext gewählt wird. D.h. es ist $p(k, x) = p(k)p(x)$, wobei

$$p(k) = \sum_{x \in M} p(k, x)$$

die Wahrscheinlichkeit für den Schlüssel k und

$$p(x) = \sum_{k \in K} p(k, x)$$

die Wahrscheinlichkeit für den Klartext x ist. O.B.d.A. sei $p(x) > 0$ für alle Klartexte $x \in M$, da wir andernfalls x aus M entfernen können. Für einen Kryptotext y berechnet sich die Wahrscheinlichkeit zu

$$p(y) = \sum_{k,x:E(k,x)=y} p(k, x)$$

[‖] Unter Verwendung des ursprünglichen Kryptosystems: *Autokey*-Chiffre mit Klartextschlüsselstrom!

und für einen fest vorgegebenen Kryptotext y ist

$$p(x|y) = \sum_{k:E(k,x)=y} \frac{p(k,x)}{p(y)}$$

die (bedingte) Wahrscheinlichkeit dafür, dass y aus dem Klartext x entstanden ist. Wir nehmen o.B.d.A. an, dass für alle $y \in C$ $p(y) > 0$ ist (andernfalls kann y aus C entfernt werden).

Definition 65 (**informationstheoretisch sicher**)

Ein Kryptosystem heißt ***absolut sicher*** (***informationstheoretisch sicher***), falls für alle $x \in M$ und alle $y \in C$ gilt:

$$p(x) = p(x|y).$$

Bei einem absolut sicheren Kryptosystem ist demnach die *a posteriori* Wahrscheinlichkeit $p(x|y)$ einer Klartextnachricht x gleich der *a priori* Wahrscheinlichkeit $p(x)$, d.h. die Wahrscheinlichkeit von x sie ist unabhängig davon, ob der Kryptotext y bekannt ist oder nicht. Die Kenntnis von y erlaubt somit keinerlei Rückschlüsse auf die gesendete Nachricht x. Dies bedeutet, dass es dem Gegner nicht möglich ist – auch nicht mit unbegrenzten Rechenressourcen – das System zu brechen. Wie wir sehen werden, lässt sich diese Art der Sicherheit nur mit einem extrem hohen Aufwand realisieren. Daher begnügt man sich meist mit schwächeren Sicherheitsanforderungen.

- Ein Kryptosystem gilt als **komplexitätstheoretisch sicher** oder als **berechnungssicher** (***computationally secure***), falls es dem Gegner nicht möglich ist, das System mit einem für ihn lohnenswerten Aufwand zu brechen. Das heißt, die Kosten eines erfolgreichen Angriffs (sofern dieser überhaupt möglich ist), würden den Nutzen bei weitem übersteigen.
- Ein Kryptosystem gilt als **nachweisbar sicher** (***provably secure***), wenn seine Sicherheit mit bekannten komplexitätstheoretischen Hypothesen verknüpft werden kann, deren Gültigkeit gemeinhin akzeptiert ist.
- Als **praktisch sicher** (***practically secure***) werden dagegen Kryptosysteme eingestuft, die über mehrere Jahre hinweg jedem Versuch einer erfolgreichen Kryptoanalyse widerstehen konnten, obwohl sie bereits eine weite Verbereitung gefunden haben und allein schon deshalb ein lohnenswertes Ziel für einen Angriff darstellen.

Die komplexitätstheoretische Analyse eines Kryptosystems ist äußerst schwierig. Dies hängt damit zusammen, daß der Aufwand eines erfolgreichen Angriffs unabhängig davon abgeschätzt werden muß, welche Strategie dabei verfolgt wird. Das heißt, es genügt nicht, alle derzeit bekannten kryptoanalytischen Ansätze in Betracht zu ziehen, sondern es müssen alle *möglichen* berücksichtigt werden. Dabei darf sich die Aufwandsanalyse auch nicht ausschließlich an einer vollständigen Rekonstruktion des Klartextes orientieren. Denn auch die Kenntnis winziger Klartextfragmente kann für den Gegner einen großen Vorteil bedeuten.

Aus den genannten Gründen ist es bis heute noch für kein praktikables Kryptosystem gelungen, seine komplexitätstheoretische Sicherheit mathematisch zu beweisen. Damit ist auch nicht so schnell zu rechnen, zumindest nicht solange der Status fundamentaler komplexitätstheoretischer Fragen wie etwa des berühmten $P \stackrel{?}{=} NP$-Problems offen ist. Dagegen gibt es eine ganze Reihe praktikabler Kryptosysteme, die als nachweisbar sicher oder praktisch sicher gelten. Zunächst wollen wir uns jedoch mit de Bedingung für absolute Sicherheit etwas näher beschäftigen.

Wegen $p(x|y)p(y) = p(x,y) = p(y|x)p(x)$ ist

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

(Satz von Bayes) und daher ist die Bedingung $p(x) = p(x|y)$ gleichbedeutend mit $p(y) = p(y|x)$.

Beispiel 66 *Sei* (M, C, E, D, K) *ein Kryptosystem mit* $M = \{x_1, \ldots, x_4\}$, $K = \{k_1, \ldots, k_4\}$, $C = \{y_1, \ldots, y_4\}$ *und*

E	x_1	x_2	x_3	x_4
k_1	y_1	y_4	y_3	y_2
k_2	y_2	y_1	y_4	y_3
k_3	y_3	y_2	y_1	y_4
k_4	y_4	y_3	y_2	y_1

Weiter sei $p(x_1) = 1/2$, $p(x_2) = p(x_3) = p(x_4) = 1/6$, *sowie* $p(k_1) = 1/2$, $p(k_2) = 1/4$ *und* $p(x_3) = p(x_4) = 1/8$. *Dann ergibt sich folgende Verteilung auf* C*:*

$$\begin{aligned}
p(y_1) &= 1/2 \cdot 1/2 + (1/4 + 1/8 + 1/8) \cdot 1/6 = 1/3 \\
p(y_2) &= 1/4 \cdot 1/2 + (1/8 + 1/8 + 1/2) \cdot 1/6 = 1/4 \\
p(y_3) &= 1/8 \cdot 1/2 + (1/8 + 1/2 + 1/4) \cdot 1/6 = 5/24 \\
p(y_4) &= 1/8 \cdot 1/2 + (1/2 + 1/4 + 1/8) \cdot 1/6 = 5/24
\end{aligned}$$

Die bedingten Wahrscheinlichkeiten $p(x|y_1)$ *berechnen sich wie folgt:*

$$\begin{aligned}
p(x_1|y_1) &= p(k_1, x_1)/p(y_1) = (1/2)(1/2)/(1/3) = 3/4 \\
p(x_2|y_1) &= p(k_2, x_2)/p(y_1) = (1/4)(1/6)/(1/3) = 1/8 \\
p(x_3|y_1) &= p(k_3, x_3)/p(y_1) = (1/8)(1/6)/(1/3) = 1/16 \\
p(x_4|y_1) &= p(k_4, x_4)/p(y_1) = (1/8)(1/6)/(1/3) = 1/16
\end{aligned}$$

Wegen $p(x_1) = 1/2 \neq 3/4 = p(x_1|y_1)$ *ist das System unter dieser Verteilung nicht absolut sicher.*

Die Bedingung $p(x) = p(x|y)$ *ist nach dem Satz von Bayes genau dann erfüllt, wenn* $p(y) = p(y|x)$ *ist. Da jedoch für jedes Paar* (x, y) *genau ein Schlüssel* $k = k_{x,y} \in K$ *mit* $E(k, x) = y$ *existiert, also* $p(y|x) = p(k_{x,y})$ *ist, ist dies äquivalent zu* $p(y) = p(k_{x,y})$. *Für* $y = y_1$ *bedeutet dies, dass alle Schlüssel* $k_i = k_{x_i,y_1}$ *die gleiche Wahrscheinlichkeit* $p(k_i) = 1/4$ *haben müssen. Eine leichte Rechnung zeigt, dass unter dieser*

Schlüsselverteilung $p(y_i) = {}^1\!/_4$ *für* $i = 1, \ldots, 4$ *ist. Somit ist das Kryptosystem genau dann absolut sicher, wenn der Schlüssel unter Gleichverteilung gewählt wird (dies gilt unabhängig von der Klartextverteilung).*

Verwendet man beim *One-Time-Pad* nur Klartexte einer festen Länge n, d. h. $M \subseteq A^n$, so ist dieser absolut sicher (vorausgesetzt, der Schlüssel wird rein zufällig, also unter Gleichverteilung gewählt). Bezeichne $k_{x,y}$ den (eindeutig bestimmten) Schlüssel, der den Klartext x auf den Kryptotext y abbildet. Wegen $K = A^n$ und $p(k) = \|A\|^{-n}$ für alle $k \in K$ folgt zunächst für jeden Klartext x und Kryptotext y

$$p(y) = \sum_{k,x:E(k,x)=y} p(k,x) = \sum_x p(x) \cdot p(k_{x,y}) = p(k_{x,y})$$

und damit

$$p(x|y) = \frac{\sum_{k:E(k,x)=y} p(k,x)}{p(y)} = \frac{p(x) \cdot p(k_{x,y})}{p(y)} = p(x).$$

Variiert die Klartextlänge, so kann ein Gegner aus y nur die Länge des zugehörigen Klartextes x ableiten. Wird jedoch derselbe Schlüssel k zweimal verwendet, so kann aus den Kryptotexten die Differenz der zugehörigen Klartexte ermittelt werden:

$$\left.\begin{array}{lcl} y_1 & = & E(x_1, k) = x_1 + k \\ y_2 & = & E(x_2, k) = x_2 + k \end{array}\right\} \rightsquigarrow y_1 - y_2 = x_1 - x_2$$

Sind die Klartexte natürlichsprachig, so können aus $y_1 - y_2$ die beiden Nachrichten x_1 und x_2 ähnlich wie bei der Analyse einer Lauftextverschlüsselung (siehe Abschnitt 1.13) rekonstruiert werden.

Da jeder Schlüsselbuchstabe beim *One-Time-Pad* nur einmal verwendet werden darf, ist der Aufwand extrem hoch. Zunächst muss ein genügend langer Schlüsselstrom k durch einen Zufallsgenerator erzeugt werden, welcher dann auf einem sicheren Kanal zwischen den Kommunikationspartnern ausgetauscht werden muss. Wird keine absolute Sicherheit angestrebt, so kann der Schlüsselstrom auch von einem Pseudo-Zufallsgenerator erzeugt werden. Dieser erhält als Eingabe eine Zufallsfolge s_0 (den sogenannten *Keim*) und erzeugt daraus eine lange Folge $v_0\, v_1 \ldots$ von Pseudo-Zufallszahlen. Als Schlüssel muss jetzt nur noch das Wort s_0 ausgetauscht werden.

In der Informationstheorie wird die Unsicherheit, mit der eine durch X beschriebene Quelle ihre Nachrichten aussendet, nach ihrer Entropie bemessen. Das heißt, die Unsicherheit über X entspricht genau dem Informationsgewinn, der sich aus der Beobachtung der Quelle X ziehen lässt. Dabei wird die in einer einzelnen Nachricht (***m**essage*) m steckende Information um so höher bemessen, je seltener m auftritt. Tritt eine Nachricht m mit einer positiven Wahrscheinlichkeit $p(m) = Prob[X = m] > 0$ auf, dann ist

$$Inf_X(m) = \log_2({}^1\!/_{p(m)})$$

der **Informationsgehalt** von m. Ist dagegen $p(m) = 0$, so sei $Inf_X(m) = 0$. Dieser Wert des Informationsgehalts ergibt sich zwangsläufig aus den beiden folgenden Forderungen:

- Der gemeinsame Informationsgehalt $\mathit{\textbf{Inf}}_{X,Y}(m, m')$ von zwei Nachrichten m und m', die aus stochastisch unabhängigen Quellen X und Y stammen, sollte gleich $\mathit{\textbf{Inf}}_X(m) + \mathit{\textbf{Inf}}_Y(m')$ sein;
- der Informationsgehalt einer Nachricht, die mit Wahrscheinlichkeit $1/2$ auftritt, soll genau 1 (bit) betragen.

Die Entropie von X ist nun der erwartete Informationsgehalt einer von X stammenden Nachricht.

Definition 67 **(Entropie)**

Sei X eine Zufallsvariable mit Wertebereich $W(X) = \{m_1, \ldots, m_n\}$ und sei $p_i = \mathit{Prob}[X = m_i]$. Dann ist die **Entropie** von X definiert als

$$H(X) = \sum_{i=1}^{n} p_i \mathit{\textbf{Inf}}_X(m_i).$$

Die Entropie nimmt also im Fall $p_1 = \cdots p_n = 1/n$ den Wert $\log_2(n)$ an. Für jede andere Verteilung $p_1, \ldots, p_n$ gilt dagegen $H(X) < \log_2(n)$ (Beweis später). Generell ist die Unsicherheit über X um so kleiner, je ungleichmäßiger X verteilt ist. Bringt X nur einen einzigen Wert mit positiver Wahrscheinlichkeit hervor, dann (und nur dann) nimmt $H(X)$ den Wert 0 an.

Eine wichtige Eigenschaft der Entropie ist, dass sie eine untere Schranke für die mittlere Codewortlänge von Binärcodes bildet. Ein **Binärcode** für X ist eine (geordnete) Menge $C = \{x_1, \ldots, x_n\}$ von binären Codewörtern x_i für die Nachrichten m_i mit der Eigenschaft, dass die Abbildung $c : M^* \to \{0,1\}^*$ mit $c(m_{i_1} \cdots m_{i_k}) = x_{i_1} \cdots x_{i_k}$ injektiv ist. Die mittlere Codewortlänge von C unter X ist

$$L(C) = \sum_{i=1}^{n} p_i \cdot |x_i|.$$

C heißt **optimal**, wenn kein anderer Binärcode für X eine kürzere mittlere Codewortlänge besitzt. Für einen optimalen Binärcode C für X gilt

$$H(X) \leq L(C) < H(X) + 1.$$

Beispiel 68 (Entropie) *Sei X eine Zufallsvariable mit der Verteilung*

m_i	*sonnig*	*leicht bewölkt*	*bewölkt*	*stark bewölkt*	*Regen*	*Schnee*	*Nebel*
p_i	$1/4$	$1/4$	$1/8$	$1/8$	$1/8$	$1/16$	$1/16$

Dann ergibt sich die Entropie von X zu

$$H(X) = 1/4 \cdot (2+2) + 1/8 \cdot (3+3+3) + 1/16 \cdot (4+4) = 2{,}625.$$

Betrachten wir die beiden Codes $C_1 = \{\mathtt{001}, \mathtt{010}, \mathtt{011}, \mathtt{100}, \mathtt{101}, \mathtt{110}, \mathtt{111}\}$ und $C_2 = \{\mathtt{00}, \mathtt{01}, \mathtt{100}, \mathtt{101}, \mathtt{110}, \mathtt{1110}, \mathtt{1111}\}$, so erhalten wir für die mittlere Codewortlänge von C_1 den Wert $L(C_1) = 3$, während C_2 wegen $|x_i| = \log_2(1/p_i)$ den Wert $L(C_2) = H(X)$ erreicht und somit optimal ist.

Die Redundanz eines Codes für eine Zufallsvariable X ist um so höher, je größer seine mittlere Codewortlänge im Vergleich zur Entropie von X ist. Um auch Codes über unterschiedlichen Alphabeten miteinander vergleichen zu können, ist es notwendig, die Codewortlänge in einer festen Einheit anzugeben. Hierzu berechnet man die **Bitlänge** eines Wortes x über einem Alphabet A mit $m > 2$ Buchstaben zu $|x|_2 = |x| \log_2(m)$. Beispielsweise ist die Bitlänge von GOLD (über dem lateinischen Alphabet) $|\texttt{GOLD}|_2 = 4 \log_2(26) = 18,8$. Entsprechend berechnet sich für einen Code $C = \{x_1, \ldots, x_n\}$ unter einer Verteilung $p_1, \ldots, p_n$ die mittlere Codewortlänge (in bit) zu

$$L_2(C) = \sum_{i=1}^{n} p_i \cdot |x_i|_2.$$

Damit können wir die Redundanz eines Codes als den mittleren Anteil der Codewortbuchstaben definieren, die keine Information tragen.

Definition 69 (**Redundanz**)

Die (relative) **Redundanz** eines Codes C für X ist definiert als

$$\mathcal{R}(C) = \frac{L_2(C) - H(X)}{L_2(C)}.$$

Beispiel 68 (Entropie, Fortsetzung) *Während eine von X generierte Nachricht im Durchschnitt $H(X) = 2{,}625$ bit an Information enthält, haben die Codewörter von C_1 eine Bitlänge von 3. Der Anteil an „überflüssigen" Zeichen pro Codewort beträgt also*

$$\mathcal{R}(C_1) = \frac{3 - 2{,}625}{3} = 12,5\%,$$

wogegen C_2 keine Redundanz besitzt.

Auch Schriftsprachen wie Deutsch oder Englisch und Programmiersprachen wie C oder PASCAL können als eine Art Code aufgefasst werden. Um die statistischen Eigenschaften einer solchen Sprache L zu erforschen, erweist es sich als zweckmäßig, die Textstücke der Länge n (n-Gramme) von L für unterschiedliche n getrennt voneinander zu betrachten. Sei also L_n die Zufallsvariable, die die Verteilung aller n-Gramme in L beschreibt. Interpretieren wir diese n-Gramme als Codewörter einer einheitlichen Codewortlänge n, so ist

$$\mathcal{R}(L_n) = \frac{n \log_2 m - H(L_n)}{n \log_2 m}$$

die Redundanz dieses Codes. Es ist zu erwarten, dass eine Sprache umso mehr Redundanz aufweist, je restriktiver die Gesetzmäßigkeiten sind, unter denen in ihr Worte und Sätze gebildet werden.

Definition 70 (**Entropie einer Sprache**)

Für eine Sprache L über einem Alphabet A mit $\|A\| = m$ und n-Gramm-Verteilung L_n ist $H(L_n)/n$ die ***Entropie von*** L_n (pro Buchstabe). Falls dieser Wert für n gegen ∞ gegen einen Grenzwert

$$H(L) = \lim_{n\to\infty} H(L_n)/n$$

konvergiert, so wird dieser Grenzwert als die ***Entropie von*** L bezeichnet. In diesem Fall konvergiert $\mathcal{R}(L_n)$ gegen den Grenzwert

$$\lim_{n\to\infty} \mathcal{R}(L_n) = 1 - \frac{H(L)}{\log_2 m},$$

der als die Redundanz $\mathcal{R}(L)$ von L bezeichnet wird.

Für eine Reihe von natürlichen Sprachen wurden die Redundanzen $\mathcal{R}(L_n)$ der n-Gramme (für nicht allzu große Werte von n) empirisch bestimmt, so dass sich $\mathcal{R}(L)$ näherungsweise bestimmen lässt.

Beispiel 71 *Im Deutschen haben die Einzelbuchstaben eine Entropie von* $H(L_1) = 4,1$ *bit, während sich für die Bi- und Trigramme Entropiewerte von 3,86 und 3,61 bit pro Buchstabe ergeben. Mit wachsender Länge sinkt die Entropie von deutschsprachigen Texten weiter ab und strebt gegen einen Grenzwert* $H(L)$ *von 1,6 bit pro Buchstabe.*

n	$H(L_n)$	$H(L_n)/n$
1	$4,10$	$4,10$
2	$7,72$	$3,86$
3	$10,82$	$3,61$
$\vdots$	$\vdots$	$\vdots$
∞	∞	$1,6$

Für die Redundanz $\mathcal{R}(L)$ *ergibt sich hieraus (wegen log(26) = 4,76) ein Wert von ca. 67%, so dass sich ein deutscher Text bei optimaler Kodierung auf circa* $1/3$ *seiner ursprünglichen Länge komprimieren lässt.*

Wir betrachten nun den Fall, dass mit einem Kryptosystem Klartexte der Länge n verschlüsselt werden, ohne dass dabei der Schlüssel gewechselt wird. D. h.

$$E_n : K \times A^n \to C_n$$

wobei C_n die Menge der zugehörigen Kryptotexte ist. Der Einfachheit halber nehmen wir $\|C_n\| = \|A^n\| = m^n$ an. Ist y ein abgefangener Kryptotext, so ist

$$S(y) = \{k \in K \mid \exists x \in A^n : E_n(k, x) = y \wedge p(x) > 0\}$$

die Menge aller in Frage kommenden Schlüssel für y. $S(y)$ besteht aus einem „echten" (d. h. dem zur Generierung von y tatsächlich benutzten) und $\|S(y)\| - 1$ so genannten „unechten" Schlüsseln. Aus informationstheoretischer Sicht ist das Kryptosystem desto unsicherer, je kleiner die erwartete Anzahl

$$\bar{s}_n = \sum_{y \in C_n} p(y) \cdot (\|S(y)\| - 1) = \sum_{y \in C_n} p(y) \cdot \|S(y)\| - 1$$

der unechten Schlüssel ist. Ist $\bar{s}_n$ gleich 0, so liefert der abgefangene Kryptotext y dem Gegner genügend Information, um den zugehörigen Klartext eindeutig bestimmen zu können (sofern er über unbegrenzte Ressourcen an Rechenkraft und Zeit verfügt).

Definition 72 (**Eindeutigkeitsdistanz**)

Die **Eindeutigkeitsdistanz** n_0 eines Kryptosystems ist der kleinste Wert von n, für den $\bar{s}_n = 0$ wird.

Als nächstes wollen wir eine untere Schranke für n_0 herleiten. Hierzu benötigen wir den Begriff der bedingten Entropie $H(X|Y)$ von X, wenn Y bereits bekannt ist.

Definition 73 (**bedingte Entropie**)

Seien X, Y Zufallsvariablen. Dann ist die **bedingte Entropie** von X unter Y definiert als

$$H(X|Y) = \sum_{y \in W(Y)} p(y) \cdot H(X|y),$$

wobei $X|y$ die Zufallsvariable mit der Verteilung $Prob[X|y = x] = p(x|y) = Prob[X = x \mid Y = y]$ ist (d.h. $H(X|y) = \sum_{x \in W(X)} p(x|y) \cdot \log_2(1/p(x|y))$).

Als weiteres Hilfsmittel aus der Analysis benötigen wir die Jensensche Ungleichung, mit der sich obere Schranken für die Entropie von Zufallsvariablen herleiten lassen.

Definition 74 (**konkav**)

Eine reellwertige Funktion f ist **konkav** auf einem Intervall I, falls für alle $x \neq y \in I$ gilt:

$$f\left(\frac{x+y}{2}\right) \quad \geq \quad \frac{f(x) + f(y)}{2}$$

Gilt sogar „$>$" anstelle von „$\geq$", so heißt f **streng konkav** auf I.

Satz 75 (Jensensche Ungleichung) *Sei f eine streng konkave Funktion auf I und seien $0 \leq a_1, \dots, a_n \leq 1$ reelle Zahlen mit $\sum_{i=1}^n a_i = 1$. Dann gilt*

$$\sum_{i=1}^{n} a_i f(x_i) \quad \leq \quad f\left(\sum_{i=1}^{n} a_i x_i\right).$$

Die beiden Seiten der Ungleichung sind genau dann gleich, wenn $x_1 = \cdots = x_n$ ist.

Beispiel 76 *Die Funktion $f(x) = \log_2(x)$ ist konkav auf $(0, \infty)$.*

Satz 77 *Sei X eine Zufallsvariable mit Wertebereich $W(X) = \{x_1, \dots, x_n\}$ und Verteilung $Prob[X = x_i] = p_i$, $i = 1, \dots, n$. Dann gilt $H(X) \leq \log_2(n)$, wobei Gleichheit genau im Fall $p_i = 1/n$ für $i = 1, \dots, n$ eintritt.*

Beweis: Es gilt

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} p_i \log_2(1/p_i) \\ &\leq \log_2 \sum_{i=1}^{n} p_i/p_i \\ &= \log_2 n. \end{aligned}$$

Nach obigem Satz tritt Gleichheit genau im Fall $1/p_1 = \cdots = 1/p_n$ ein, was mit $p_i = 1/n$ für $i = 1, \dots, n$ gleichbedeutend ist. ■

Satz 78 $H(X, Y) = H(Y) + H(X|Y)$.

Beweis: s. Übungen. ■

Satz 79 $H(X, Y) \leq H(X) + H(Y)$, *wobei Gleichheit genau dann eintritt, wenn X und Y stochastisch unabhängig sind.*

Beweis: s. Übungen. ■

Korollar 80 $H(X|Y) \leq H(X)$, *wobei Gleichheit genau dann eintritt, wenn X und Y stochastisch unabhängig sind.*

Satz 81 *In jedem Kryptosystem gilt für die Klartextentropie $H(X)$, die Schlüsselentropie $H(K)$ und die Kryptotextentropie $H(Y)$*

$$H(K|Y) = H(K) + H(X) - H(Y).$$

Beweis: Zunächst ist $H(K|Y) = H(K,Y) - H(Y)$. Es reicht also zu zeigen, dass

$$H(K,Y) = H(K) + H(X)$$

ist. Da bei Kenntnis des Schlüssels der Wert von X bereits eindeutig durch Y und der Wert von Y eindeutig durch X festgelegt ist, folgt unter Berücksichtigung der gemachten Annahme, dass X und K unabhängig sind,

$$H(K,Y) = H(K,X,Y) = H(K,X) = H(K) + H(X).$$

■

Jetzt verfügen wir über alle Hilfsmittel, um die erwartete Anzahl

$$\bar{s}_n = \sum_{y \in C_n} p(y) \cdot \|S(y)\| - 1$$

unechter Schlüssel nach unten abschätzen zu können. Seien X_n und Y_n die Zufallsvariablen, die die Verteilungen der n-Gramme der Klartextsprache und der zugehörigen Kryptotexte beschreiben. Mit Satz 81 folgt

$$H(K|Y_n) = H(K) + H(X_n) - H(Y_n).$$

Die Klartextentropie $H(X_n)$ lässt sich durch

$$H(X_n) \geq nH(L) = n(1 - R(L)) \log_2 m$$

abschätzen, wobei $m = \|A\|$ ist. Zudem lässt sich die Kryptotextentropie $H(Y_n)$ wegen $W(Y_n) = C_n$ und $\|C_n\| = m^n$ durch

$$H(Y_n) \leq n \log_2 m$$

abschätzen. Somit ist

$$H(K|Y_n) = H(K) + \underbrace{H(X_n) - H(Y_n)}_{\geq -nR(L)\log_2 m}.$$

Andererseits gilt (unter Verwendung der Jensenschen Ungleichung)

$$\begin{aligned}
H(K|Y_n) &= \sum_{y \in C_n} p(y) \cdot H(K|y) \\
&\leq \sum_{y \in C_n} p(y) \cdot \log_2 \|S(y)\| \\
&\leq \log_2 \sum_{y \in C_n} p(y) \cdot \|S(y)\| \\
&= \log_2(\bar{s}_n + 1).
\end{aligned}$$

Zusammen ergibt sich also

$$\log_2(\bar{s}_n + 1) \geq H(K) - nR(L) \log_2 m.$$

Im Fall, dass der Schlüssel unter Gleichverteilung gezogen wird, erreicht $H(K)$ den maximalen Wert $\log_2 \|K\|$, was auf die gesuchte Abschätzung für $\bar{s}_n$ führt. Wir fassen zusammen.

Satz 82 *Werden mit einem Kryptosystem Klartexte $x \in A^n$ der Länge n mit einem unter Gleichverteilung gezogenen Schlüssel $k \in K$ verschlüsselt, und ist $\|C_n\| = \|A^n\| = m^n$ für den zugehörigen Kryptotextraum $C_n = \{E(k,x) \mid k \in K, x \in A^n\}$, so gilt für die erwartete Anzahl $\bar{s}_n$ der unechten Schlüssel,*

$$\bar{s}_n \geq \frac{\|K\|}{m^{nR(L)}} - 1.$$

Setzen wir in obiger Abschätzung $\bar{s}_n = 0$, so erhalten wir folgende untere Schranke für die Eindeutigkeitsdistanz n_0 des Kryptosystems:

$$n_0 \geq \frac{\log_2 \|K\|}{R(L)\log_2 m} = \frac{\log_2 \|K\|}{\log_2 m - H(L)} = \log_2 \|K\|/R_{abs}(L),$$

wobei $R_{abs}(L) = \log_2 m - H(L)$ die absolute Redundanz $R_{abs}(L)$ der Klartextsprache bezeichnet.

Beispiel 83 *Für Transpositionen bei deutschsprachigem Klartext ergeben sich die folgenden Schätzwerte $\log_2 \|K\|/R_{abs}(L)$ für die Eindeutigkeitsdistanz n_0 (wobei wir $R_{abs}(L) = 3{,}2$ annehmen, was einer relativen Redundanz von $R(L) = 3,2/4,76 \approx 67\%$ entspricht):*

Kryptosystem	*Schlüsselanzahl* $\|K\|$	$\log_2 \|K\|$	$\log_2 \|K\|/R_{abs}(L)$
Verschiebechiffre	26	4,7	$\frac{4{,}7}{3{,}2} \approx 1{,}5$
Affine Chiffre	$12 \cdot 26 = 312$	8,3	2,6
einfache Substitution	26!	88,4	27,6
Vigenère-Chiffre	26^d	$4{,}7 \cdot d$	$1{,}5 \cdot d$
DES-Algorithmus	2^{56}	56	17,5

Dagegen erhalten wir für Blocktranspositionen folgende Schätzwerte für die Menge an Kryptotext, die zur eindeutigen Bestimmung des Schlüssels benötigt wird:

Analyse auf Basis von	$R_{abs}(L)$	*Blocklänge l* 10	20	50	100	1 000
n-Grammen	3,20	*7*	*19*	*67*	*164*	*2 665*
Trigrammen	1,15	*24*	*65*	*226*	*553*	*9 473*
Bigrammen	0,90	*40*	*111*	*390*	*954*	*15 502*
Einzelzeichen	0,66	*59*	*165*	*578*	*1415*	*22986*

Auch wenn die Schätzwerte für n_0 bei der Analyse auf Basis von Einzelzeichen endlich sind, ist in dieseen Fällen $n_0 = \infty$. Denn wie wir wissen, führt eine solche Analyse nicht zum Ziel, und zwar unabhängig davon, über wie viel Kryptotext der Gegner verfügt.

2 Symmetrische Kryptosysteme

2.1 Produktchiffren

Produktchiffren erhält man durch die sequentielle Anwendung mehrerer Verschlüsselungsverfahren. Sie können extrem schwer zu brechen sein, auch wenn die einzelnen Komponenten leicht zu brechen sind.

Definition 84 (**Produktkryptosystem**)

Seien $S_1 = (M_1, C_1, E_1, D_1, K_1)$ und $S_2 = (M_2, C_2, E_2, D_2, K_2)$ Kryptosysteme mit $C_1 = M_2$. Dann ist das **Produktkryptosystem** von S_1 und S_2 definiert als $S_1 \times S_2 = (M_1, C_2, E, D, K_1 \times K_2)$ mit

$$E(k_1, k_2; x) = E_2(k_2, E_1(k_1, x)) \text{ und } D(k_1, k_2; y) = D_1(k_1, D_2(k_2, y))$$

für alle $x \in M_1$, $y \in C_2$ und $(k_1, k_2) \in K_1 \times K_2$.

Der Schlüsselraum von $S_1 \times S_2$ umfasst also alle Paare (k_1, k_2) von Schlüsseln $k_1 \in K_1$ und $k_2 \in K_2$, wobei wir voraussetzen, dass die Schlüssel unabhängig gewählt werden (d.h. es gilt $p(k_1, k_2) = p(k_1)p(k_2)$).

Beispiel 85 *Sei $A = \{a_0, \ldots, a_{m-1}\}$. Man sieht leicht, dass die affine Chiffre $S = (M, C, K, E, D)$ mit $M = C = \mathcal{A}$ und $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$ das Produkt $S = S_1 \times S_2$ der multiplikativen Chiffre $S_1 = (M, C, K_1, E_1, D_1)$ mit der additiven Chiffre $S_2 = (M, C, K_2, E_2, D_2)$ ist, da für jeden Schlüssel $k = (k_1, k_2) \in K = \mathbb{Z}_m^* \times \mathbb{Z}_m$ gilt:*

$$E(k, x) = k_1 x + k_2 = E_2(k_2, E_1(k_1, x)).$$

Für $S' = S_2 \times S_1$ erhalten wir das Kryptosystem $S' = (M, C, K', E', D')$ mit $K' = \mathbb{Z}_m \times \mathbb{Z}_m^*$ und

$$E'(k_1, k_2; x) = k_2(x + k_1) = k_2 x + k_2 k_1 = E(k_2, k_2 k_1; x)$$

für jeden Schlüssel $(k_1, k_2) \in K'$. Da die Abbildung

$$(k_1, k_2) \mapsto (k_2, k_2 k_1)$$

eine Bijektion zwischen den Schlüsselräumen K' und K ist und die Schlüssel unter Gleichverteilung gewählt werden, sind die Kryptosysteme $S = S_1 \times S_2$ und $S' = S_2 \times S_1$ als gleich (besser: äquivalent, siehe übungen) anzusehen, d.h. S_1 und S_2 kommutieren. ◁

Definition 86 (**endomorph, idempotent**)

Ein Kryptosystem $S = (M, C, K, D, E)$ mit $M = C$ heißt **endomorph**.
Ein endomorphes Kryptosystem S heißt **idempotent**, falls $S \times S = S$ ist.

Beispiel 87 *Eine leichte Rechnung zeigt, dass die additive, die multiplikative und die affine Chiffre idempotent sind. Ebenso die Blocktransposition sowie die Vigenère- und Hill-Chiffre.*

Will man durch mehrmalige Anwendung (Iteration) derselben Chiffriermethode eine höhere Sicherheit erreichen, so darf diese nicht idempotent sein. Man kann beispielsweise versuchen, ein nicht idempotentes System S durch die Kombination $S = S_1 \times S_2$ zweier idempotenter Verfahren S_1 und S_2 zu erhalten. Wegen

$$\begin{aligned}(S_1 \times S_2) \times (S_1 \times S_2) &= S_1 \times (S_2 \times S_1) \times S_2 \\ &= S_1 \times (S_1 \times S_2) \times S_2 \\ &= (S_1 \times S_1) \times (S_2 \times S_2) \\ &= S_1 \times S_2\end{aligned}$$

dürfen hierbei S_1 und S_2 jedoch nicht kommutieren.

Im Rest dieses Kapitels werden wir nur noch das Binäralphabet $A = \{0, 1\}$ als Klar- und Kryptotextalphabet benutzen und auch der Schlüsselraum wird von der Form $\{0, 1\}^k$ sein, wobei k die Schlüssellänge bezeichnet.

Eine iterierte Blockchiffre wird typischerweise durch eine Rundenfunktion (*round function*) g und einen Schlüsselgenerator (*key schedule algorithm*) f beschrieben. Ist N die Rundenzahl, so erzeugt f bei Eingabe eines Schlüssels K eine Folge $f(K) = (K^1, \ldots, K^N)$ von N Rundenschlüsseln K^i für g. Mit diesen wird ein Klartext $x = w^0$ durch N-malige Anwendung der Rundenfunktion g zu einem Kryptotext $y = w^N$ verschlüsselt:

$$\begin{aligned}&w^1 \leftarrow g(K^1, w^0)\\ &\quad\vdots\\ &w^N \leftarrow g(K^N, w^{N-1})\end{aligned}$$

Um y wieder zu entschlüsseln, muss die inverse Rundenfunktion g^{-1} mit umgekehrter Rundenschlüsselfolge $K^N, \ldots, K^1$ benutzt werden:

$$\begin{aligned}&w^{N-1} \leftarrow g^{-1}(K^N, w^N)\\ &\quad\vdots\\ &w^0 \leftarrow g^{-1}(K^1, w^1)\end{aligned}$$

Beispiele für iterierte Chiffren sind der aus 16 Runden bestehende DES-Algorithmus und der AES mit einer variablen Rundenzahl $N \in \{10, 12, 14\}$, die wir in späteren Abschnitten behandeln werden.

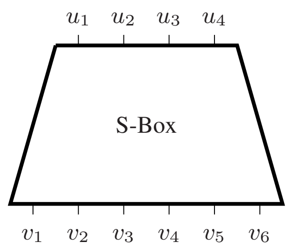
2.2 Substitutions-Permutations-Netzwerke

In diesem Abschnitt betrachten wir den prinzipiellen Aufbau von iterierten Blockchiffren. Als Basisbausteine für die Rundenfunktion eignen sich Substitutionen und Transpositionen besonders gut. Aus Effizienzgründen sollten die Substitutionen nur eine relativ kleine Blocklänge l haben.

Definition 88 (**Teilwort**)

Für ein Wort $u = u_1 \cdots u_n \in \{0,1\}^n$ und Indizes $1 \leq i \leq j \leq n$ bezeichne $u[i,j]$ das Teilwort $u_i \cdots u_j$ von u. Im Fall $n = lm$ bezeichnen wir das Teilwort $u[(i-1)l+1, il]$ auch einfach mit $u_{(i)}$, d.h. es gilt $u = u_{(1)} \cdots u_{(m)}$, wobei $|u_{(i)}| = l$.

Sei $\pi_S : A^l \to A^{l'}$ eine Substitution, die Binärblöcke u der Länge l in Binärblöcke $v = \pi_S(u)$ der Länge l' überführt (engl. auch als **S-Box** bezeichnet).



Durch parallele Anwendung von m dieser S-Boxen erhalten wir folgende Substitution $S : A^{lm} \to A^{l'm}$,

$$S(u_1 \cdots n_{lm}) = \pi_S(u_{(1)}) \cdots \pi_S(u_{(m)}).$$

Um eine S-Box $\pi_S : \{0,1\}^l \to \{0,1\}^{l'}$ durch Software zu implementieren, werden $l'2^l$ bit Speicherplatz benötigt (im Fall $l = l'$ also $l2^l$ bit). Für $l = l' = 16$ wären dies beispielsweise 2^{20} bit, was bereits Smartcard-Anwendungen ausschließen würde.

Für eine Transposition P auf A^{lm} bezeichnen wir die zugehörige Permutation auf $\{1, \ldots, lm\}$ mit π_P, d.h.

$$P(u_1 \cdots u_{lm}) = u_{\pi_P(1)} \cdots u_{\pi_P(m)}.$$

Definition 89 (**Substitutions-Permutations-Netzwerk**)

Sei $A = \{0,1\}$ und sei $M = C = A^{lm}$ für natürliche Zahlen $l, m \geq 1$. Ein **Substitutions-Permutations-Netzwerk** (SPN) wird durch Permutationen $\pi_S : \{0,1\}^l \to \{0,1\}^l$ und $\pi_P : \{1, \ldots, lm\} \to \{1, \ldots, lm\}$ sowie durch einen Schlüsselgenerator $f : \{0,1\}^k \to \{0,1\}^{lm(N+1)}$ beschrieben. Der Generator f erzeugt aus einem (externen) Schlüssel $K \in \{0,1\}^k$ eine Folge $f(K) = (K^1, \ldots, K^{N+1})$ von $N+1$ Rundenschlüsseln K^r, unter denen ein Klartext $x \in \{0,1\}^{lm}$ gemäß folgendem Algorithmus in einen Kryptotext $y = E_{f,\pi_S,\pi_P}(K, x) \in \{0,1\}^{lm}$ überführt wird:

Algorithmus 90 $E_{f,\pi_S,\pi_P}(K,x)$

$w^0 \leftarrow x$
for $r \leftarrow 1$ **to** $N-1$ **do**
  $u^r \leftarrow w^{r-1} \oplus K^r$
  $v^r \leftarrow S(u^r)$
  $w^r \leftarrow P(v^r)$
end
$u^N \leftarrow w^{N-1} \oplus K^N$
$v^N \leftarrow S(u^N)$
$y \leftarrow v^N \oplus K^{N+1}$

Zu Beginn jeder Runde $r \in \{1,\ldots,N\}$ wird w^{r-1} zunächst einer XOR-Operation mit dem Rundenschlüssel K^r unterworfen (dies wird *round key mixing* genannt), deren Resultat u^r den S-Boxen zugeführt wird. Auf die Ausgabe v^r der S-Boxen wird in jeder Runde $r \leq N-1$ die Transposition P angewendet, was die Eingabe w^r für die nächste Runde $r+1$ liefert.

Am Ende der letzten Runde $r = N$ wird nicht die Transposition P angewandt, sondern der Rundenschlüssel K^{N+1} auf v^N addiert. Durch diese (*whitening* genannte) Vorgehensweise wird einerseits erreicht, dass auch für den letzten Chiffrierschritt der Schlüssel benötigt und somit der Gegner von einer partiellen Entschlüsselung des Kryptotexts abgehalten wird. Zum Zweiten ermöglicht dies eine (legale) Entschlüsselung nach fast demselben Verfahren (siehe Übungen).

Beispiel 91 *Sei $l = m = N = 4$ und sei $k = 32$. Für f wählen wir die Funktion $f(K) = (K^1, \ldots, K^5)$ mit $K^r = K[4(r-1)+1, 4(r-1)+16]$. Weiter seien $\pi_S : \{0,1\}^4 \to \{0,1\}^4$ und $\pi_P : \{1,\ldots,16\} \to \{1,\ldots,16\}$ die folgenden Permutationen (wobei die Argumente und Werte von π_S hexadezimal dargestellt sind; siehe auch Abbildung 1):*

z	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\pi_S(z)$	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

und

z	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi_P(z)$	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Für den Schlüssel $K = 0011\,1010\,1001\,0100\,1101\,0110\,0011\,1111$ liefert f beispielsweise die Rundenschlüssel $f(K) = (K^1, \ldots, K^5)$ mit

$$\begin{aligned}
K^1 &= 0011\,1010\,1001\,0100,\\
K^2 &= 1010\,1001\,0100\,1101,\\
K^3 &= 1001\,0100\,1101\,0110,\\
K^4 &= 0100\,1101\,0110\,0011,\\
K^5 &= 1101\,0110\,0011\,1111,
\end{aligned}$$

unter denen der Klartext $x = 0010\,0110\,1011\,0111$ die folgenden Chiffrierschritte durchläuft:

$$\begin{aligned}
x &= 0010\,0110\,1011\,0111 = w^0\\
w^0 \oplus K^1 &= 0001\,1100\,0010\,0011 = u^1\\
S(u^1) &= 0100\,0101\,1101\,0001 = v^1\\
P(v^1) &= 0010\,1110\,0000\,0111 = w^1\\
&\vdots\\
P(v^3) &= 1110\,0100\,0110\,1110 = w^3\\
w^3 \oplus K^4 &= 1010\,1001\,0000\,1101 = u^4\\
S(u^4) &= 0110\,1010\,1110\,1001 = v^4\\
u^4 \oplus K^5 &= 1011\,1100\,1101\,0110 = y.
\end{aligned}$$

2.3 Lineare Kryptoanalyse von SPNs

Zuerst stellen wir eine Reihe von Ergebnissen aus der Wahrscheinlichkeitstheorie bereit. Für eine Zufallsvariable X mit Wertebereich $W(X) = \{0,1\}$ und $p = Prob[X = 0]$ bezeichne $\varepsilon(X)$ den Wert $\varepsilon(X) = p - 1/2$ (auch *bias* von X genannt). Sind X_1, X_2 unabhängige Zufallsvariablen mit Wertebereich $W(X_i) = \{0,1\}$ und bias $\varepsilon_i = \varepsilon(X_i)$,

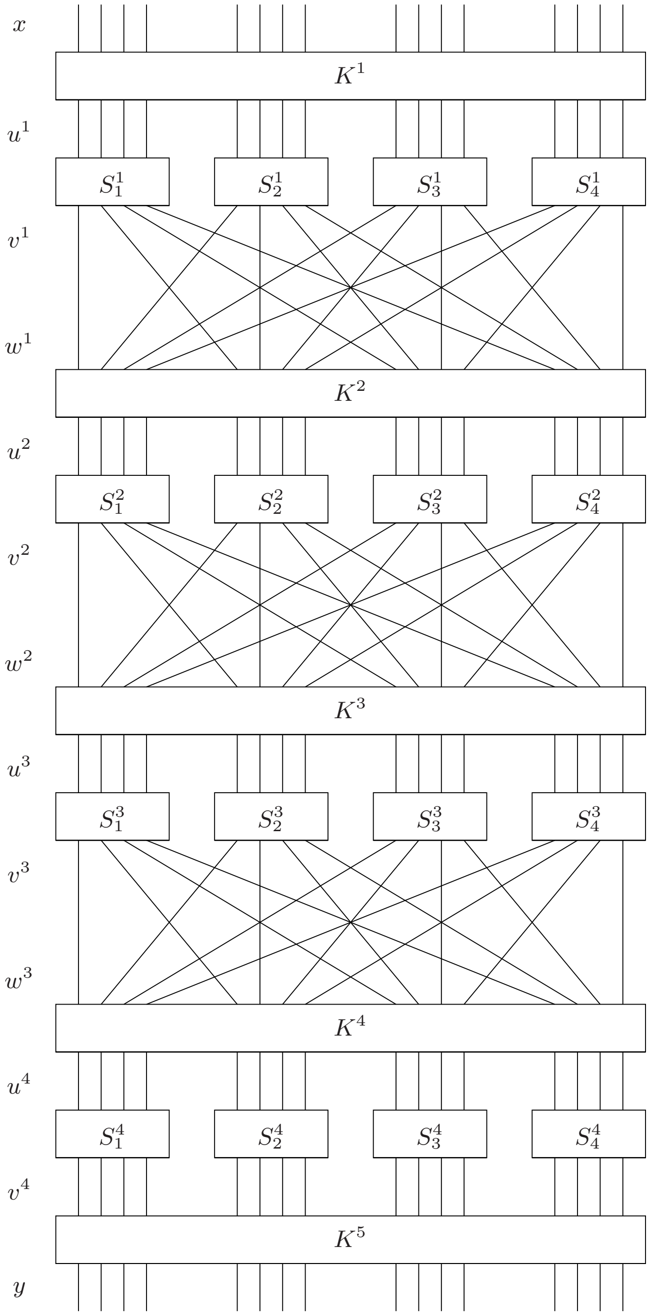


Abbildung 1 Ein Substitutions-Permutations-Netzwerk.

dann ist

$$\begin{aligned} Prob[X_1 \oplus X_2 = 0] &= Prob[X_1 = X_2 = 0] + Prob[X_1 = X_2 = 1] \\ &= (1/2 + \varepsilon_1)(1/2 + \varepsilon_2) + (1/2 - \varepsilon_1)(1/2 - \varepsilon_2) \\ &= 1/2 + 2\varepsilon_1\varepsilon_2 \end{aligned}$$

und

$$\begin{aligned} Prob[X_1 \oplus X_2 = 1] &= Prob[X_1 = 0, X_2 = 1] + Prob[X_1 = 1, X_2 = 0] \\ &= (1/2 + \varepsilon_1)(1/2 - \varepsilon_2) + (1/2 - \varepsilon_1)(1/2 + \varepsilon_2) \\ &= 1/2 - 2\varepsilon_1\varepsilon_2, \end{aligned}$$

d.h. es gilt $\varepsilon(X_1 \oplus X_2) = 2\varepsilon_1\varepsilon_2$.

Diese Beobachtung lässt sich leicht verallgemeinern.

Lemma 92 (Piling-up-Lemma) *Seien $X_1, \ldots, X_n$ unabhängige Zufallsvariablen mit $W(X_i) = \{0, 1\}$. Dann gilt*

$$\varepsilon(X_1 \oplus \cdots \oplus X_n) = 2^{n-1} \prod_{i=1}^{n} \varepsilon(X_i).$$

Beweis: Wir führen den Beweis durch Induktion über n.

Induktionsanfang ($n = 1$): Klar.

Induktionsschritt ($n \rightsquigarrow n + 1$): Seien $X_1, \ldots, X_{n+1}$ unabhängige Zufallsvariablen mit $W(X_i) = \{0, 1\}$. Dann gilt nach IV $\varepsilon(Z) = 2^{n-1}\varepsilon(X_1) \cdots \varepsilon(X_n)$ für die Zufallsvariable $Z = X_1 \oplus \cdots \oplus X_n$ und daher ist

$$\begin{aligned} \varepsilon(X_1 \oplus \cdots \oplus X_{n+1}) &= \varepsilon(Z \oplus X_{n+1}) \\ &= 2\varepsilon(Z)\varepsilon(X_{n+1}) \\ &= 2^n\varepsilon(X_1) \cdots \varepsilon(X_{n+1}) \end{aligned}$$

■

Beispiel 93 *Seien X_1, X_2, X_3 Zufallsvariablen mit $\varepsilon(X_i) = 1/4$ für $i = 1, 2, 3$. Dann gilt nach obigem Lemma $\varepsilon_{i,j} = \varepsilon(X_i \oplus X_j) = 1/8$ für $1 \le i < j \le 3$. Man beachte, dass die Zufallsvariablen $X_1 \oplus X_2$ und $X_2 \oplus X_3$ nicht unabhängig sind, und daher das Piling-up-Lemma nicht anwendbar ist. Dieses würde nämlich für die Zufallsvariable*

$$(X_1 \oplus X_2) \oplus (X_2 \oplus X_3) = X_1 \oplus X_3$$

einen bias von $\varepsilon = 2(1/8)^2 = 1/32$ ergeben, was dem tatsächlichen Wert $\varepsilon(X_1 \oplus X_3) = \varepsilon_{1,3} = 1/8$ widersprechen würde.

Lineare Approximationen

Sei $f : \{0,1\}^l \to \{0,1\}^{l'}$ eine Abbildung. Wählen wir für f eine zufällige Eingabe $U = U_1 \cdots U_l$ unter Gleichverteilung, so gilt für die zugehörige Ausgabe $V = V_1 \cdots V_{l'} = f(U)$

$$Prob[V = v \mid U = u] = \begin{cases} 1 & \pi_S(u) = v, \\ 0 & \text{sonst} \end{cases}$$

für alle $u \in \{0,1\}^l$ und $v \in \{0,1\}^{l'}$. Wegen $Prob[U = u] = 2^{-l}$ folgt

$$Prob[V = v, U = u] = \begin{cases} 2^{-l} & \pi_S(u) = v, \\ 0 & \text{sonst.} \end{cases}$$

Ist f linear, so sind die Zufallsvariablen V_i in der Form

$$V_i = U_{j_1} \oplus \cdots \oplus U_{j_k} \oplus c$$

für geeignete Indizes $1 \le j_1 < \cdots < j_k \le l$ und eine Konstante $c \in \{0,1\}$ darstellbar. Die Idee hinter der linearen Kryptoanalyse ist nun, Gleichungen der Form

$$V_{i_1} \oplus \cdots \oplus V_{i_k} = U_{j_1} \oplus \cdots \oplus U_{j_{k'}} \oplus c$$

zu finden, die mit großer WK gelten. Definieren wir für $a \in \{0,1\}^l$ und $b \in \{0,1\}^{l'}$ die Zufallsvariablen

$$U_a = \oplus_{i=1}^{l} a_i U_i \text{ und } V_b = \oplus_{i=1}^{l'} b_i V_i,$$

so sind wir also an solchen Werten für a, b und c interessiert, für die das Ereignis $V_b = U_a \oplus c$ (oder gleichbedeutend: $U_a \oplus V_b \oplus c = 0$) eine möglichst große Wahrscheinlichkeit besitzt. Dies bedeutet, dass die Zufallsvariable $U_a \oplus V_b \oplus c$ einen möglichst großen bias $\varepsilon(U_a \oplus V_b \oplus c)$ haben sollte. Wegen

$$\varepsilon(U_a \oplus V_b \oplus 1) = -\varepsilon(U_a \oplus V_b)$$

ist die durch a und b beschriebene lineare Approximation $U_a \oplus V_b$ um so besser, je größer $|\varepsilon(U_a \oplus V_b \oplus 1)|$ ist.

Beispiel 94 *Wir betrachten die S-Box $\pi_S : \{0,1\}^4 \to \{0,1\}^4$ aus Beispiel 91, so nimmt die Zufallsvariable $(U_1, \ldots, U_4, V_1, \ldots, V_4)$ jeden der folgenden Werte mit*

Wahrscheinlichkeit $2^{-4} = {}^1\!/_{16}$ *an:*

U_1	U_2	U_3	U_4	V_1	V_2	V_3	V_4	$U_3 \oplus U_4 \oplus V_1 \oplus V_4$
0	0	0	0	1	1	1	0	1
0	0	0	1	0	1	0	0	1
0	0	1	0	1	1	0	1	1
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	1	1	1	1	1
0	1	1	0	1	0	1	1	1
0	1	1	1	1	0	0	0	1
1	0	0	0	0	0	1	1	1
1	0	0	1	1	0	1	0	0
1	0	1	0	0	1	1	0	1
1	0	1	1	1	1	0	0	1
1	1	0	0	0	1	0	1	1
1	1	0	1	1	0	0	1	1
1	1	1	0	0	0	0	0	1
1	1	1	1	0	1	1	1	1

Um nun $\varepsilon(U_a \oplus V_b)$ *zu berechnen, müssen wir nur die Anzahl* $L(a,b)$ *der Zeilen bestimmen, für die* $U_a = V_b$ *ist. Dann gilt* $Prob[U_a \oplus V_b = 0] = Prob[U_a = V_b] = L(a,b)/16$ *und somit*

$$\varepsilon(U_a \oplus V_b) = L(a,b)/16 - 1/2 = (L(a,b) - 8)/16.$$

Für $a = 0011$ *und* $b = 1001$ *gibt es z.B.* $L(a,b) = 2$ *Zeilen (Zeile 5 und Zeile 10) mit* $U_a = U_3 \oplus U_4 = V_b = V_1 \oplus V_4$*, d.h.* $\varepsilon(U_3 \oplus U_4 \oplus V_1 \oplus V_4) = (L(a,b) - 8)/16 = {}^{-3}\!/_8$*. Die folgende Tabelle zeigt für alle Werte von* a *und* b *(hexadezimal dargestellt) die Anzahlen* $L(a,b)$.

	b															
a	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	6	6	8	8	6	14	10	10	8	8	10	10	8	8
2	8	8	6	6	8	8	6	6	8	8	10	10	8	8	2	10
3	8	8	8	8	8	8	8	8	10	**2**	6	6	10	10	6	6
$\vdots$									$\vdots$							
F	8	6	4	6	6	8	10	8	8	6	12	6	6	8	10	8

Lineare Kryptoanalyse eines SPN

Wir betrachten nun das SPN aus Beispiel 91 und führen eine lineare Kryptoanalyse durch. Dabei handelt es sich um einen Angriff bei bekanntem Klartext, d.h. es steht eine Menge M von t Klartext-Kryptotext-Paaren (x, y) zur Verfügung, die alle mit dem gleichen unbekannten Schlüssel K erzeugt wurden.

Seien $K^1, \ldots, K^5$ die zu K gehörigen Rundenschlüssel. Das Ziel besteht zunächst einmal darin, eine lineare Approximation für die Abbildung $x \mapsto u^4$ zu finden, bei der

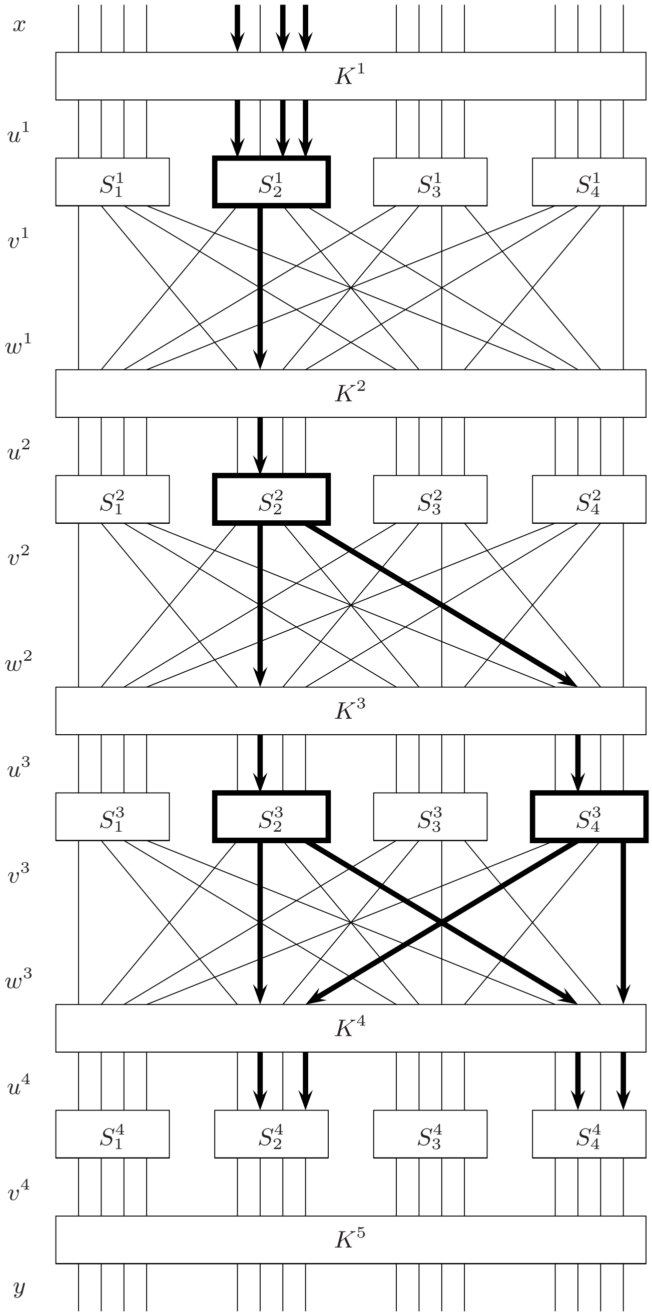


Abbildung 2 Eine lineare Approximation an ein Substitutions-Permutations-Netzwerk.

die Rundenschlüssel $K^1, \ldots, K^4$ benutzt werden (siehe Abbildung 2). Hierzu benutzen wir die beiden folgenden linearen Approximationen an die S-Box S:

$$T = U_1 \oplus U_3 \oplus U_4 \oplus V_2$$

mit einem bias von $\varepsilon(T) = (L(B,4) - 8)/16 = (12 - 8)/16 = {}^1\!/_4$ und

$$T' = U_2 \oplus V_2 \oplus V_4$$

mit einem bias von $\varepsilon(T') = (L(4,5) - 8)/16 = (4 - 8)/16 = {}^{-1}\!/_4$.

Konkret benutzen wir die lineare Approximation T für die S-Box S_2^1,

$$T_1 = U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus V_6^1$$

und die lineare Approximation T' für die S-Boxen S_2^2, S_2^3, S_4^3,

$$\begin{aligned} T_2 &= U_6^2 \oplus V_6^2 \oplus V_8^2, \\ T_3 &= U_6^3 \oplus V_6^3 \oplus V_8^3, \\ T_4 &= U_{14}^3 \oplus V_{14}^3 \oplus V_{16}^3. \end{aligned}$$

Indem wir nun die linearen Approximationen $T_1, \ldots, T_4$ der S-Boxen S_2^1, S_2^2, S_2^3 und S_4^3 „zusammen schalten“, erhalten wir für ein $c' \in \{0,1\}$ die gesuchte lineare Approximation

$$X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \quad = \quad T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus c' \qquad (3)$$

von $x \mapsto u^4$.

An dieser Stelle ergeben sich folgende drei Fragen.

1. Warum gilt (3)?
2. Wie gut ist die lineare Approximation $X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4$?
3. Wie können wir mit ihrer Hilfe den Schlüssel bestimmen?

Wir gehen zunächst auf Frage 1 ein. Drücken wir die U-Variablen, die in $T_1, \ldots, T_4$ vorkommen, durch X-Variablen und V-Variablen sowie durch Schlüsselbits K_i^r aus,

$$\begin{array}{rcl} U_5^1 & = & X_5 \oplus K_5^1, \\ U_7^1 & = & X_7 \oplus K_7^1, \\ U_8^1 & = & X_8 \oplus K_8^1, \\ U_6^2 & = & W_6^1 \oplus K_6^2 = V_6^1 \oplus K_6^2, \\ U_6^3 & = & W_6^2 \oplus K_6^3 = V_6^2 \oplus K_6^3, \\ U_{14}^3 & = & W_{14}^2 \oplus K_{14}^3 = V_8^2 \oplus K_{14}^3, \end{array}$$

so führt dies mit $c = K_5^1 \oplus K_7^1 \oplus K_8^1 \oplus K_6^2 \oplus K_6^3 \oplus K_{14}^3$ auf die Darstellung

$$T_1 \oplus \cdots \oplus T_4 = X_5 \oplus X_7 \oplus X_8 \oplus V_6^3 \oplus V_8^3 \oplus V_{14}^3 \oplus V_{16}^3 \oplus c.$$

Jetzt müssen wir nur noch die verbliebenen V^3-Variablen durch U^4-Variablen und Schlüsselbits ersetzen,

$$\begin{array}{rcl} V_6^3 & = & U_6^4 \oplus K_6^4, \\ V_8^3 & = & U_{14}^4 \oplus K_{14}^4, \\ V_{14}^3 & = & U_8^4 \oplus K_8^4, \\ V_{16}^3 & = & U_{16}^4 \oplus K_{16}^4, \end{array}$$

um mit $c' = c \oplus K_6^4 \oplus K_8^4 \oplus K_{14}^4 \oplus K_{16}^4$ die gesuchte Darstellung zu erhalten:

$$T_1 \oplus \cdots \oplus T_4 = X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \oplus c'.$$

Nun zu Frage 2: Wären die Zufallsvariablen $T_1, \ldots, T_4$ unabhängig, so würde uns das Piling-up-Lemma den Wert $2^3(1/4)(-1/4)^3 = -1/32$ für den bias von $T_1 \oplus \cdots \oplus T_4$ liefern. Obwohl dies nicht der Fall ist, stellt sich in der Praxis heraus, dass sich der tatsächliche Wert $\varepsilon = \varepsilon(T_1 \oplus \cdots \oplus T_4)$ nicht zu sehr von diesem “hypothetischen” Wert unterscheidet, und somit

$$|\varepsilon(X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4)| \approx {}^1\!/_{32}$$

ist.

Und schließlich zu Frage 3: Wir wissen bereits, dass die Wahrscheinlichkeit, dass ein zufälliger Klartext X zu einem Zwischenresultat U^4 mit

$$X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 = 0 \tag{4}$$

führt, hoch oder niedrig ist.

Gehen wir also davon aus, dass M eine repräsentative Auswahl von Klartext-Kryptotext-Paaren (x, y) darstellt, so wird die Anzahl der Paare (x, y) in M, die (4) erfüllen, ebenfalls eine Mehrheit oder eine Minderheit in M bilden. Man beachte, dass sich aus einem Kryptotext y die zugehörigen Werte für die Variablen U_6^4, U_8^4, U_{14}^4 und U_{16}^4 unter Benutzung von π_S^{-1} leicht berechnen lassen, sofern die 8 Schlüsselbits in $K_{(2)}^5$ und $K_{(4)}^5$ bekannt sind.

Die Idee besteht nun darin, für jeden Subschlüssel-Kandidaten (engl. *candidate subkey*) (L_1, L_2) für $(K_{(2)}^5, K_{(4)}^5)$ die Anzahl $\alpha(L_1, L_2)$ aller Paare (x, y) in M zu bestimmen, die Gleichung (4) bei Benützung des Schlüssel-Kandidaten (L_1, L_2) erfüllen. Für den richtigen Kandidaten wird diese Anzahl ungefähr bei $t/2 \pm t/32$ liegen, wogegen bei Benutzung eines falschen Subschlüssels mit einer Anzahl von circa $t/2$ zu rechnen ist. Ist t genügend groß, so lassen sich auf diese Weise 8 bit von K^5 (und damit von K) bestimmen.

Algorithmus 95 LINEARATTACK(M, t, π_S^{-1})

for $(L_1, L_2) \leftarrow (0, 0)$ **to** (F,F) **do**
  $\alpha(L_1, L_2) \leftarrow 0$ **end**
for each $(x, y) \in M$ **do**
  for $(L_1, L_2) \leftarrow (0, 0)$ **to** (F, F) **do**
    $v_{(2)}^4 \leftarrow L_1 \oplus y_{(2)}$
    $v_{(4)}^4 \leftarrow L_2 \oplus y_{(4)}$
    $u_{(2)}^4 \leftarrow \pi_S^{-1}(v_{(2)}^4)$
    $u_{(4)}^4 \leftarrow \pi_S^{-1}(v_{(4)}^4)$
    if $x_5 \oplus x_7 \oplus x_8 \oplus u_6^4 \oplus u_8^4 \oplus u_{14}^4 \oplus u_{16}^4 = 0$ **then**
      $\alpha(L_1, L_2) \leftarrow \alpha(L_1, L_2) + 1$
  end
end
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0, 0)$ **to** (F, F) **do**
  $\beta(L_1, L_2) \leftarrow |\alpha(L_1, L_2) - t/2|$
  if $\beta(L_1, L_2) > max$ **then**
    $max \leftarrow \beta(L_1, L_2)$
    $maxkey \leftarrow (L_1, L_2)$
end
Ausgabe: $maxkey$

Im allgemeinen werden für eine erfolgreiche lineare Attacke circa $t \approx c\varepsilon^{-2}$ Klartext-Kryptotext-Paare benötigt, wobei c eine „kleine" Konstante ist (im Beispielfall reichen $t \approx 8000$ Paare, d.h. $c \approx 8$, da $\varepsilon^{-2} = 1024$ ist.

2.4 Differentielle Kryptoanalyse von SPNs

Bei der differentiellen Kryptoanalyse handelt es sich um einen Angriff bei frei wählbarem Klartext. Genauer gesagt, basiert der Angriff auf einer Menge M von t Klartext-Kryptotext-Doppelpaaren (x, x^*, y, y^*) mit der Eigenschaft, dass alle Klartext-Paare (x, x^*) die gleiche Differenz $a' = x \oplus x^*$ bilden.

Definition 96 (**Eingabe- und Ausgabedifferenz**)

Seien $x, x^* \in \{0,1\}^l$ zwei Eingaben für eine S-Box $\pi_S : \{0,1\}^l \to \{0,1\}^{l'}$ und seien $y = \pi_S(x)$ und $y^* = \pi_S(x^*)$ die zugehörigen Ausgaben. Dann wird $x' = x \oplus x^*$ die **Eingabedifferenz** (engl. input-x-or) und $y' = \pi_S(x) \oplus \pi_S(x^*)$ die **Ausgabedifferenz** (engl. output-x-or) des Paares (x, x^*) genannt. Für eine vorgegebene Eingabedifferenz $a' \in \{0,1\}^l\}$ sei weiter

$$\begin{aligned} \Delta(a') &= \{(x, x^*) \mid x, x^* \in \{0,1\}^l, x \oplus x^* = a'\} \\ &= \{(x, x \oplus a') \mid x \in \{0,1\}^l\} \end{aligned}$$

die Menge aller Eingabepaare, die die Differenz x' realisieren.

Berechnen wir für alle Eingabepaare $(x, x^*) \in \Delta(a')$ die zugehörigen Ausgabedifferenzen, so verteilen sich diese mehr oder weniger gleichmäßig auf die $2^{l'}$ möglichen Werte in $\{0,1\}^{l'}$. Je ungleichmäßiger die auftretenden Ausgabedifferenzen verteilt sind, je leichter lässt sich eine differentielle Kryptoanalyse durchführen. Man beachte, dass im Fall einer linearen S-Box nur die Ausgabedifferenz $\pi_S(a')$ auftritt, da dann $\pi_S(x) \oplus \pi_S(x^*) = \pi_S(x \oplus x^*)$ ist.

Definition 97 (**Differential, Weitergabequotient**)

Sei $a' \in \{0,1\}^l$ eine Eingabe- und sei $b' \in \{0,1\}^{l'}$ eine Ausgabedifferenz für eine S-Box π_S. Dann heißt (a', b') **Differential**. Die Anzahl der Eingabepaare (x, x^*), die die Eingabedifferenz a' in die Ausgabedifferenz b' überführen, bezeichnen wir mit $D(a', b')$, d.h.

$$D(a', b') = \|\{(x, x^*) \in \Delta(a') \mid \pi_S(x) \oplus \pi_S(x^*) = b'\}\|.$$

Der **Weitergabequotient** (engl. propagation ratio) von π_S für ein Differential (a', b') ist

$$Q(a', b') = \frac{D(a', b')}{2^l}.$$

$Q(a', b')$ ist also die (bedingte) Wk

$$Prob[\pi_S(x) \oplus \pi_S(x^*) = b' \mid x \oplus x^* = a'],$$

dass zwei zufällig gewählte Eingaben die Ausgabedifferenz b' erzeugen, wenn sie die Eingabedifferenz a' bilden.

Beispiel 98 *Betrachten wir die S-Box* $\pi_S : \{0,1\}^4 \rightarrow \{0,1\}^4$ *aus Beispiel 91, so erhalten wir für die Eingabedifferenz* $a' = 1011$ *die Menge*

$$\Delta(a') = \{(0000, 1011), \ldots, (1111, 0100)\}$$

von möglichen Eingabepaaren, die auf folgende Ausgabedifferenzen $y' = y \oplus y^* = \pi_S(x) \oplus \pi_S(x^*)$ *führen:*

x	x^*	y	y^*	y'
0000	1011	1110	1100	0010
0001	1010	0100	0110	0010
0010	1001	1101	1010	0111
0011	1000	0001	0011	0010
0100	1111	0010	0111	0101
0101	1110	1111	0000	1111
0110	1101	1011	1001	0010
0111	1100	1000	0101	1101
1000	0011	0011	0001	0010
1001	0010	1010	1101	0111
1010	0001	0110	0100	0010
1011	0000	1100	1110	0010
1100	0111	0101	1000	1101
1101	0110	1001	1011	0010
1110	0101	0000	1111	1111
1111	0100	0111	0010	0101

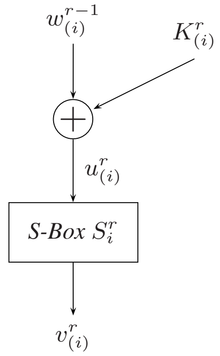
Die Ausgabedifferenz $b' = 0010$ *kommt also* $D(a', 0010) = 8$ *Mal vor, während die Differenzen* 0101, 0111, 1101 *und* 1111 *je zwei Mal und die übrigen Werte überhaupt nicht vorkommen (siehe Zeile* B *in nachfolgender Tabelle). Führen wir diese Berechnungen für jede der* $2^4 = 16$ *Eingabedifferenzen* $a' \in \{0,1\}^4$ *aus, so erhalten wir die folgenden Werte für die Häufigkeiten* $D(a', b')$ *der Ausgabedifferenz* b' *bei Eingabedifferenz* a' *(*a' *und* b' *sind hexadezimal dargestellt):*

	b'															
a'	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
$\vdots$									$\vdots$							
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
$\vdots$									$\vdots$							
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Wie können wir nun die ungleichmäßige Verteilung der Ausgabedifferenzen zur Bestimmung von Schlüsselbits ausnützen? Eine Antwort auf diese Frage geben die folgenden zwei Beobachtungen.

Beobachtung 99 *Für die Eingabe $u^r_{(i)}$ der S-Box S^r_i in Runde r gilt*

$$u^r_{(i)} = w^{r-1}_{(i)} \oplus K^r_{(i)}.$$



Daher hängt die Eingabedifferenz

$$(u^r_{(i)})' = u^r_{(i)} \oplus (u^r_{(i)})^* = (w^{r-1}_{(i)} \oplus K^r_{(i)}) \oplus ((w^{r-1}_{(i)})^* \oplus K^r_{(i)}) = w^{r-1}_{(i)} \oplus (w^{r-1}_{(i)})^*$$

von S^r_i nicht von den Schlüsselbits in Runde r ab, während die Ausgabedifferenz

$$(v^r_{(i)})' = v^r_{(i)} \oplus (v^r_{(i)})^* = \pi_S(w^{r-1}_{(i)} \oplus K^r_{(i)}) \oplus \pi_S((w^{r-1}_{(i)})^* \oplus K^r_{(i)})$$

neben der Eingabedifferenz $(u^r_{(i)})'$ auch noch von $w^{r-1}_{(i)} \oplus K^r_{(i)}$ abhängt.

Können wir nun in einem SPN für bestimmte S-Boxen S_i Differentiale (a'_i, b'_i) finden, so dass die Eingabedifferenz dieser Differentiale mit der (permutierten) Ausgabedifferenz der Differentiale in der jeweils vorhergehenden Runde übereinstimmt (siehe Abbildung 3), so können wir diese Differentiale zu einer so genannten **Differentialspur** (engl. differential trail) zusammen setzen. Könnten wir annehmen, dass die beteiligten S-Boxen S_i (diese werden auch als **aktiv** bezeichnet) die Differenzen unabhängig voneinander gemäß (a'_i, b'_i) weitergeben, so wäre der Weitergabequotient der Spur gleich dem Produkt der Weitergabequotienten aller beteiligten Differentiale. (Obwohl diese Annahme nicht richtig ist, wird der daraus resultierende Wert durch die Praxis bestätigt).

Beispiel 98 (Fortsetzung) *Betrachten wir das SPN aus Beispiel 91, so lassen sich folgende Differentiale zu einer Spur für die Abbildung $x \mapsto u^4$ kombinieren (siehe auch Abbildung 3):*

Für S^1_2: *das Differential* $(1011, 0010) = (B, 2)$ *mit* $Q(B, 2) = 1/2$*,*
für S^2_3: *das Differential* $(0100, 0110) = (4, 6)$ *mit* $Q(4, 6) = 3/8$ *und*
für S^3_2 und S^3_3: *das Differential* $(0010, 0101) = (2, 5)$ *mit* $Q(2, 5) = 3/8$.

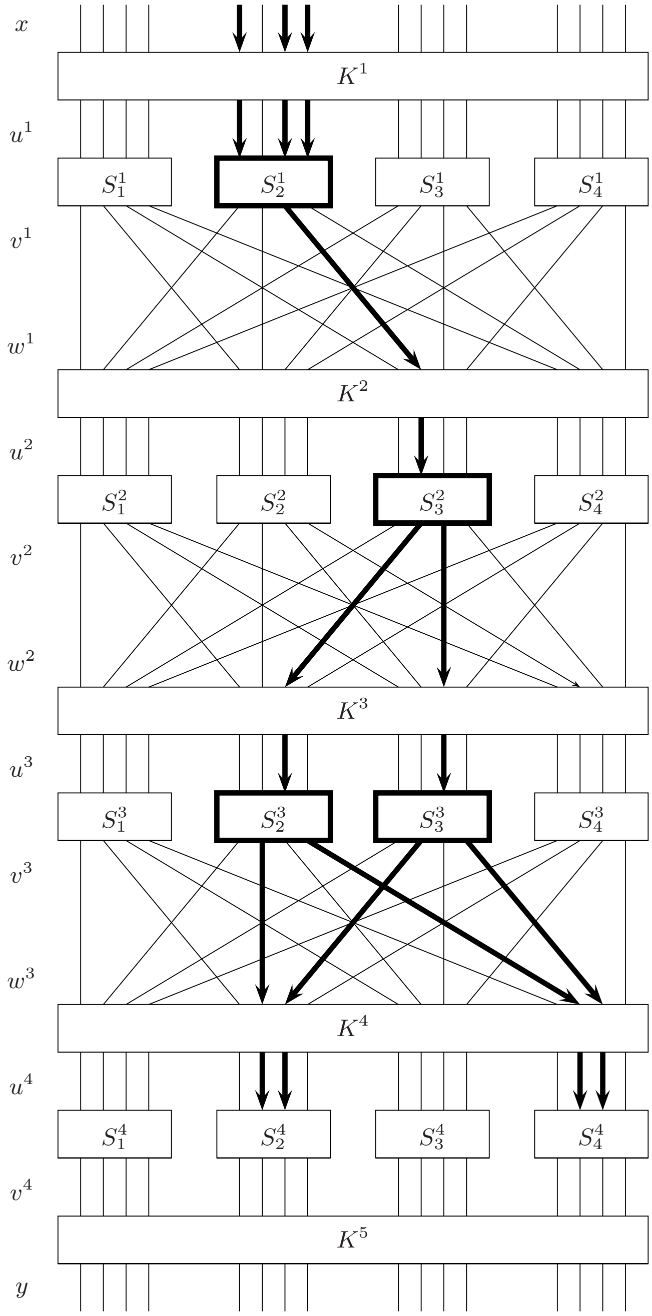


Abbildung 3 Eine Differentialspur für ein Substitutions-Permutations-Netzwerk.

Gemäß dieser Spur führt also eine Eingabedifferenz

$$x' = 0000\,1011\,0000\,0000$$

mit hypothetischer Wk $1/2(3/8)^3 = 27/1024 \approx 0,026$ *auf die Differenz*

$$(v^3)' = 0000\,0101\,0101\,0000,$$

welche wiederum mit Wk 1 auf die Differenz

$$(u^4)' = 0000\,0110\,0000\,0110$$

führt. D.h. wir erhalten für die Abbildung $x \mapsto u^4$ *die Spur*

$$(a', b') = (0000\,1011\,0000\,0000, 0000\,0110\,0000\,0110)$$

mit einem hypothetischen Weitergabequotienten von $Q(a', b') = 27/1024$.

Sei nun (a', b') eine Spur für die Abbildung $x \mapsto u^4$ mit einem hypothetischen Weitergabequotienten $Q(a', b') = \varepsilon$. Weiter sei M eine Menge von t Klartext-Kryptotext-Doppelpaaren (x, x^*, y, y^*), die alle mit dem gleichen unbekannten Schlüssel K erzeugt wurden und zusätzlich die Eigenschaft haben, dass die Klartextdifferenz $x' = x \oplus x^* = a'$ ist. Dann wird ca. ein ε-Anteil dieser Doppelpaare der Spur (a', b') folgen und daher bei Verschlüsselung mit K Zwischenergebnisse u^4 und $(u^4)^*$ liefern, die die Differenz

$$(u^4)' = u^4 \oplus (u^4)^* = b'$$

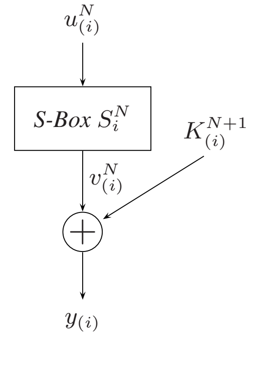
aufweisen. Doppelpaare mit dieser Eigenschaft werden **richtige Doppelpaare** (für die Spur (a', b')) genannt.

Beobachtung 100 *Für die Ausgabe* $v^r_{(i)}$ *der S-Box* S^r_i *in Runde* $r = N$ *gilt*

$$v^N_{(i)} = y_{(i)} \oplus K^{N+1}_{(i)}$$

und die Eingabe $u^N_{(i)}$ *der S-Box* S^N_i *in Runde* N *ist*

$$u^N_{(i)} = \pi^{-1}_{S^N_i}(v^N_{(i)}) = \pi^{-1}_{S^N_i}(y_{(i)} \oplus K^{N+1}_{(i)})$$



Ist S_i^N nichtlinear, so hängt die aus den Kryptotextblöcken $y_{(i)}$ und $(y_{(i)})^*$ zurückgerechnete Eingabedifferenz

$$(u_{(i)}^N)' = u_{(i)}^N \oplus (u_{(i)}^N)^* = \pi_{S_i^N}^{-1}(y_{(i)} \oplus K_{(i)}^{N+1}) \oplus \pi_{S_i^N}^{-1}((y_{(i)})^* \oplus K_{(i)}^{N+1})$$

von den Schlüsselbits in $K_{(i)}^{N+1}$ ab. Ist also (x, x^*, y, y^*) ein richtiges Doppelpaar, so sind sowohl die Eingabedifferenz $b'_{(i)} = (u_{(i)}^N)'$ von S_i^N als auch die Kryptotextblöcke $y_{(i)}$ und $y_{(i)}^*$ bekannt. Folglich kommen nur solche Subkey-Werte k für $K_{(i)}^{N+1}$ in Frage, für die

$$\pi_{S_i^N}^{-1}(y_{(i)} \oplus k) \oplus \pi_{S_i^N}^{-1}(y_{(i)}^* \oplus k) = b'_{(i)} \tag{5}$$

ist. Erfüllt k Gleichung (5), so sagen wir auch, k ist mit dem Doppelpaar (x, x^*, y, y^*) ***konsistent****.*

Gemäß Beobachtung 100 kann jedes richtige Doppelpaar dazu benutzt werden, die in Frage kommenden Werte für bestimmte Blöcke des Rundenschlüssels K^5 einzuschränken. Ist M hinreichend groß, so wird sich schließlich der richtige Teilschlüssel als derjenige heraus kristallisieren, der mit den meisten Doppelpaaren konsistent ist. Wir benutzen nun die in Beispiel 98 gefundene Spur für einen Angriff mittels differentieller Analyse.

Beispiel 98 (Fortsetzung) *Wegen*

$$(u_{(i)}^4)' = 0000 \;\Rightarrow\; (v_{(i)}^4)' = 0000 \;\Rightarrow\; (y_{(i)})' = 0000$$

kann (x, x^*, y, y^*) nur dann ein richtiges Doppelpaar für die Spur (a', b') sein, wenn $y'_{(1)} = y'_{(3)} = 0000$ ist. Da falsche Doppelpaare nichts zur Bestimmung des Schlüssels beitragen (im Wesentlichen erzeugen sie nur „Hintergrundrauschen"), werden sie am besten ausgefiltert (zumindest soweit sie als solche erkennbar sind).

In Algorithmus 98 wird nun für jeden Subschlüssel-Kandidaten (L_1, L_2) für $(K_{(2)}^5, K_{(4)}^5)$ die Anzahl $\gamma(L_1, L_2)$ aller Doppelpaare (x, x^*, y, y^*) in M bestimmt, die nicht als falsch erkannt werden und mit (L_1, L_2) konsistent sind. Ausgegeben wird der Kandidat (L_1, L_2) mit dem größten γ-Wert.

Algorithmus 99 DIFFERENTIALATTACK(M, t, π_S^{-1})

for $(L_1, L_2) \leftarrow (0, 0)$ ***to (F,F) do***
  $\gamma(L_1, L_2) \leftarrow 0$ ***end***
for each $(x, x^*, y, y^*) \in M$ ***do***
  if $y_{(1)} = y_{(1)}^*$ *und* $y_{(3)} = y_{(3)}^*$ ***then***
    for $(L_1, L_2) \leftarrow (0, 0)$ ***to*** (F, F) ***do***
      $v_{(2)}^4 \leftarrow L_1 \oplus y_{(2)}$
      $v_{(4)}^4 \leftarrow L_2 \oplus y_{(4)}$
      $u_{(2)}^4 \leftarrow \pi_S^{-1}(v_{(2)}^4)$

$\quad\quad u^4_{(4)} \leftarrow \pi_S^{-1}(v^4_{(4)})$
$\quad\quad (v^4_{(2)})^* \leftarrow L_1 \oplus y^*_{(2)}$
$\quad\quad (v^4_{(4)})^* \leftarrow L_2 \oplus y^*_{(4)}$
$\quad\quad (u^4_{(2)})^* \leftarrow \pi_S^{-1}((v^4_{(2)})^*)$
$\quad\quad (u^4_{(4)})^* \leftarrow \pi_S^{-1}((v^4_{(4)})^*)$
$\quad\quad (u^4_{(2)})' \leftarrow u^4_{(2)} \oplus (u^4_{(2)})^*$
$\quad\quad (u^4_{(4)})' \leftarrow u^4_{(4)} \oplus (u^4_{(4)})^*$
$\quad\quad$ ***if*** $(u^4_{(2)})' = 0110$ ***und*** $(u^4_{(4)})' = 0110$ ***then***
$\quad\quad\quad \gamma(L_1, L_2) \leftarrow \gamma(L_1, L_2) + 1$
$\quad\quad$ ***end***
end
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0, 0)$ ***to*** (F, F) ***do***
$\quad$ ***if*** $\gamma(L_1, L_2) > max$ ***then***
$\quad\quad max \leftarrow \gamma(L_1, L_2)$
$\quad\quad maxkey \leftarrow (L_1, L_2)$
end
Ausgabe: $maxkey$

Im allgemeinen werden für eine erfolgreiche differentielle Attacke circa $t \approx c\varepsilon^{-1}$ Klartext-Kryptotext-Doppelpaare benötigt, wobei ε der Weitergabequotient der benutzten Spur und c eine „kleine“ Konstante ist (im Beispielfall reichen $t \approx 80$ Doppelpaare, wobei $\varepsilon^{-1} \approx 38$ ist).

2.5 Der Data Encryption Standard (DES)

Der DES wurde von IBM im Zuge einer im Mai 1973 veröffentlichten Ausschreibung des NBS (National Bureau of Standards; heute National Institute of Standards and Technology, NIST) als ein Nachfolger von Lucifer entwickelt, im März 1975 veröffentlicht, und im Januar 1977 als Verschlüsselungsstandard der US-Regierung für nicht geheime Nachrichten genormt. Obwohl der DES ursprünglich nur für einen Zeitraum von 10 bis 15 Jahren als Standard dienen sollte, wurde er circa alle 5 Jahre (zuletzt im Januar 1999) überprüft und als Standard fortgeschrieben.

Bereits im September 1997 veröffentlichte das NIST eine Ausschreibung für den AES (Advanced Encryption Standard) genannten Nachfolger des DES. Nach einer mehrjährigen Auswahlprozedur wurde im November 2001 der Rijndael-Algorithmus als AES genormt.

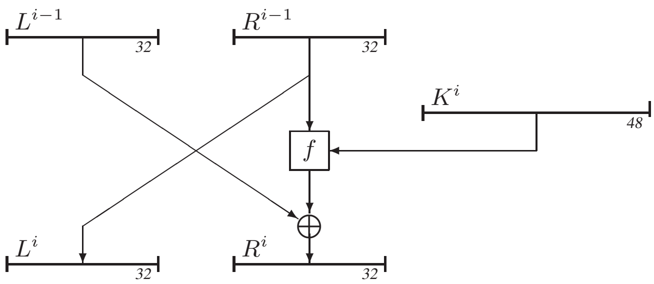
Der DES ist eine Feistel-Chiffre mit 16 Runden. Die Rundenfunktion g einer Feistel-Chiffre berechnet das Zwischenergebnis w^i aus den beiden Hälften L^{i-1} und R^{i-1} von

w^{i-1} gemäß der Vorschrift

$$g(K^i, L^{i-1}R^{i-1}) = L^iR^i,$$

wobei sich $w^i = L^iR^i$ zusammensetzt aus

$$\begin{aligned} L^i &= R^{i-1} \text{ und} \\ R^i &= L^{i-1} \oplus f(R^{i-1}, K^i). \end{aligned}$$



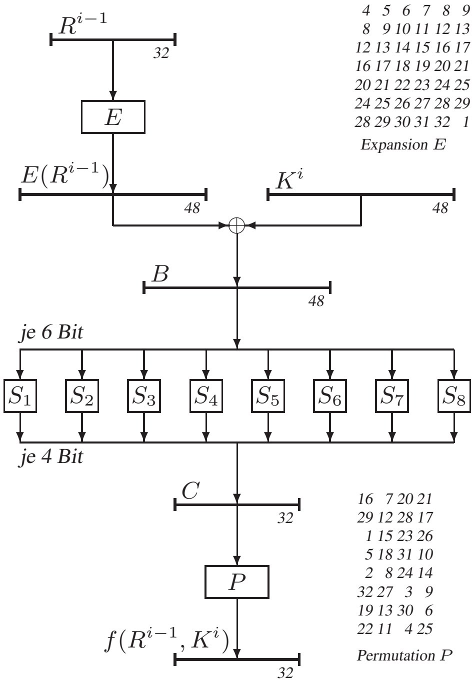
Der DES chiffriert Binärblöcke der Länge 64 und benutzt hierzu einen Schlüssel mit 56 Bit. Der Schlüssel ergibt zusammen mit 8 Paritätsbits (die Bits 8, 16,..., 64) einen ebenfalls 64 Bit langen Schlüsselblock K. Es gibt somit $2^{56} \approx 7{,}2 \cdot 10^{16}$ verschiedene Schlüssel.

Im Einzelnen werden folgende Chiffrierschritte ausgeführt:

1. Zuerst wird der Klartextblock x einer Initialpermutation IP unterzogen:

$$x_1x_2 \cdots x_{64} \mapsto IP(x) = x_{58}x_{50} \cdots x_7.$$

2. Danach wird 16 Mal die Rundenfunktion g mit den Rundenschlüsseln $K^1, \ldots, K^{16}$ angewendet, wobei die Funktion $f : \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$ wie folgt berechnet wird:



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

Expansion E

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

Permutation P

S_1 :	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2 :	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3 :	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4 :	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5 :	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6 :	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7 :	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8 :	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Substitutionsboxen

Bei Eingabe (R^{i-1}, K^i) wird R^{i-1} durch die Expansionsabbildung E auf einen 48-Bit Block $E(R^{i-1})$ erweitert. Dieser wird mit K^i bitweise addiert (x-or); als Ergebnis erhält man den Vektor $B = E(R^{i-1}) \oplus K^i$. Danach wird B in acht 6-Bit Blöcke $B_1, \ldots, B_8$ aufgeteilt, die mittels 8 S-Boxen $S_1, \ldots, S_8$ auf 4-Bit Blöcke $C_i = S_i(B_i)$ reduziert werden. Die S-Boxen sind in Form einer Tabelle dargestellt, die wie folgt ausgewertet wird:

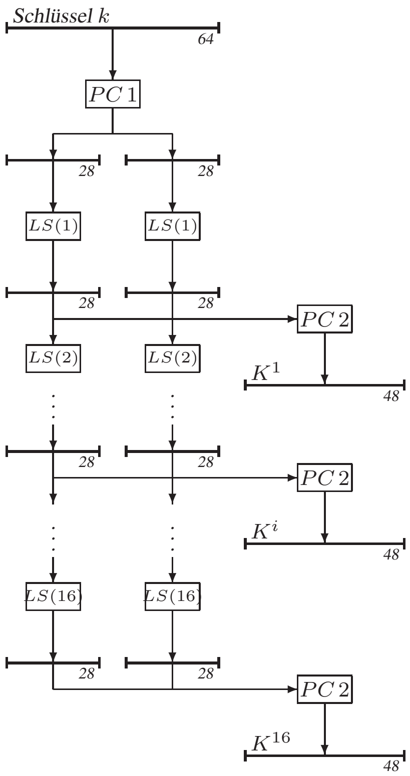
> Ist $B_i = b_1 \cdots b_6$, so findet man $S_i(B_i)$ in Zeile $b_1 b_6$ und Spalte $b_2 b_3 b_4 b_5$ (jeweils aufgefasst als Binärzahl) der Tabelle für S_i. Zum Beispiel ist $S_1(011010) = 1001$, da in Zeile $(00)_2 = 0$ und Spalte $(1101)_2 = 13$ der S_1-Tabelle die Zahl $9 = (1001)_2$ eingetragen ist.

Die Konkatenation der von den acht S-Boxen gelieferten Bitblöcke $C_1 \ldots C_8$ ergibt einen 32-Bit Vektor C, welcher noch der Permutation P unterworfen wird.

3. Aus dem nach der 16. Iteration erhaltenen Bitvektor $w^{16} = L^{16}R^{16}$ wird durch Vertauschen der beiden Hälften und Anwendung der inversen Initialpermutation der Kryptotext y gebildet:

$$L^{16}R^{16} \mapsto y = IP^{-1}(R^{16}L^{16}).$$

Die Schlüsselgenerierung. Zuerst wählt die Funktion $PC\,1$ (permuted choice 1) aus dem Schlüssel k die kryptographisch relevanten Bits aus und permutiert sie. Das erhaltene Ergebnis wird in zwei 28-Bit Blöcke unterteilt. Diese beiden Blöcke werden dann in 16 Runden jeweils zyklisch um ein oder zwei Bit verschoben (siehe dazu Tabelle $LS(i)$).



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

permuted choice 1

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

permuted choice 2

Iteration i	Anzahl der Links-Shifts $LS(i)$
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Aus den beiden Blöcken nach Runde i bestimmt die Funktion $PC\,2$ (permuted choice 2) jeweils den Rundenschlüssel K^i durch Entfernen der 8 Bits an den Stellen 9, 18, 22, 25, 35, 38, 43 und 56 sowie einer Permutation der verbleibenden 48 Bits.

Eigenschaften des DES. Der DES hat sich zwar weitgehend durchgesetzt, jedoch wurde er anfangs von manchen US-Behörden und -Banken nicht verwendet. Der Grund dafür liegt in folgenden Sicherheitsbedenken, die nach seiner Veröffentlichung im Jahre 1975 geäußert wurden:

– Die 56-Bit Schlüssellänge bietet eventuell eine zu geringe Sicherheit gegen Ausprobieren aller Schlüssel bei einem Angriff mit bekanntem oder gewähltem Klartext.
– Die Entwurfskriterien für die einzelnen Bestandteile, insbesondere für die S-Boxen, sind nicht veröffentlicht worden. Es wurde der Verdacht geäußert, dass der DES mit Hilfe von Falltürinformationen leicht zu brechen sei.
– Kryptoanalytische Untersuchungen, die von IBM und der US National Security Agency (NSA) durchgeführt wurden, sind nicht veröffentlicht worden. Als jedoch Biham und Shamir Anfang der 90er Jahre das Konzept der differentiellen

Kryptoanalyse veröffentlichten, gaben die Entwickler von DES bekannt, dass sie diese Angriffsmöglichkeit beim Entwurf von DES bereits kannten und speziell die S-Boxen entsprechend konzipiert hätten.

Im Fall von DES ist die lineare Kryptoanalyse effizienter als die differentielle Kryptoanalyse. Da hierzu jedoch circa 2^{43} Klartext-Kryptotext-Paare notwendig sind (deren Generierung bei einem von Matsui, dem Erfinder der linearen Kryptoanalyse, unternommenen Angriff bereits 40 Tage in Anspruch nahm), stellen diese Angriffe keine realistische Bedrohung dar.

Dagegen wurde im Juli 1998 mit einer von der Electronic Frontier Foundation (EFF) für 250 000 Dollar gebauten Maschine namens “DES Cracker” eine vollständige Schlüsselsuche in circa 56 Stunden durchgeführt (was den Gewinn der von RSA Laboratory ausgeschriebenen “DES Challenge II-2” bedeutete). Und im Januar 1999 gewann `Distributed.Net`, eine weltweite Vereinigung von Computerfans, den mit 10 000 Dollar dotierten “DES Challenge III”. Durch den kombinierten Einsatz eines Supercomputer namens “Deep Crack” von EFF und 100 000 PCs, die weltweit über das Internet kommunizierten, wurden nur 22 Stunden und 15 Minuten benötigt, um den Schlüssel für ein Klartext-Kryptotextpaar mit dem Klartext „See you in Rome (second AES Conference, March 22-23, 1999)“ zu finden.

Definition 101 (**schwache Schlüssel**)

Ein DES-Schlüssel K heißt **schwach**, falls alle durch ihn erzeugten Rundenschlüssel gleich sind (d.h. es gilt $\|\{K^1,\ldots, K^{16}\}\|$=1).

Es gibt vier schwache Schlüssel (siehe Übungen):

0101010101010101
FEFEFEFEFEFEFEFE
1F1F1F1F0E0E0E0E
E0E0E0E0F1F1F1F1

und für sie gilt

$$\mathrm{DES}(K, \mathrm{DES}(K, x)) = x.$$

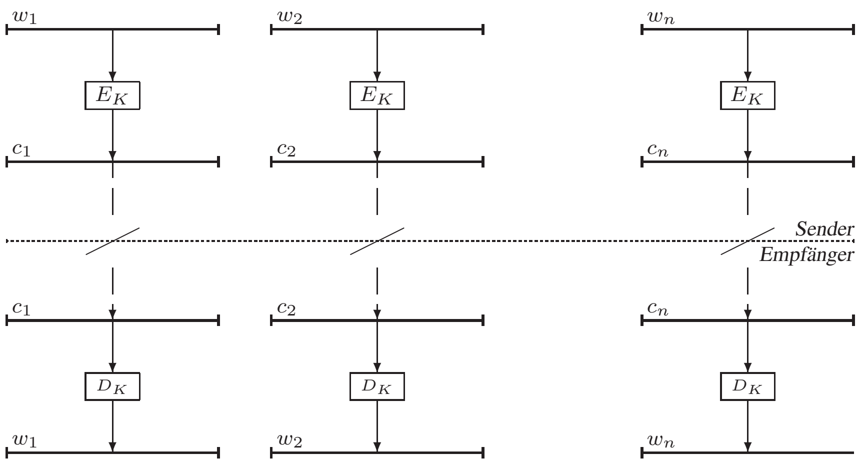
Neben diesen schwachen Schlüsseln existieren noch sechs weitere sogenannte „semischwache“ Schlüsselpaare (K, K'), für die gilt (siehe Übungen)

$$\mathrm{DES}(K', \mathrm{DES}(K, x)) = x.$$

2.6 Betriebsarten von Blockchiffren

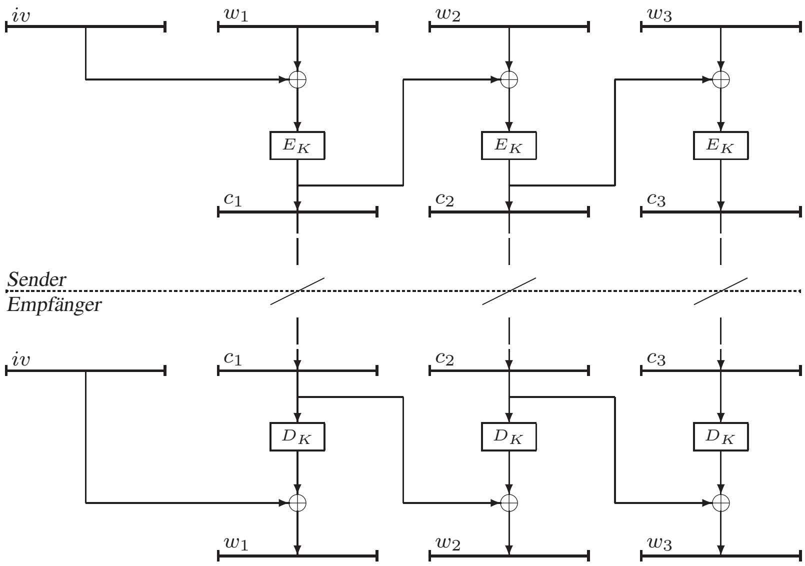
Für den DES wurden vier verschiedene Betriebsarten vorgeschlagen, in denen grundsätzlich jede Blockchiffre E mit beliebiger Blocklänge l betrieben werden kann. Bei den ersten beiden Betriebsarten (ECB und CBC) werden Kryptotextblöcke der Länge l übertragen. Mit einer Blockchiffre kann aber auch ein Stromsystem realisiert werden, mit dem sich Kryptotextblöcke einer beliebigen Länge t, $1 \leq t \leq l$, übertragen lassen (OFB und CFB).

ECB-Mode (electronic code book; elektronisches Codebuch): Die Binär-Nachricht x wird in 64-Bit Blöcke w_i zerlegt. Der letzte Block w_n wird, falls nötig, mit einer vorher vereinbarten Bitfolge aufgefüllt. Die Blöcke werden nacheinander mit demselben Schlüssel K einzeln verschlüsselt, übertragen und auf Empfängerseite mittels der zu E gehörigen Dechiffrierfunktion D wieder entschlüsselt:

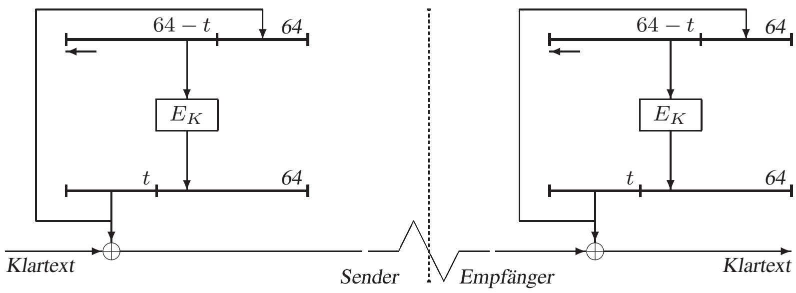


Um zu verhindern, dass ein Eindringling den Kryptotext verändert, ohne dass der Empfänger dies bemerkt, wird jeder Kryptotextblock nicht nur von dem zugehörigen Klartextblock, sondern auch von allen vorausgehenden Blöcken abhängig gemacht.

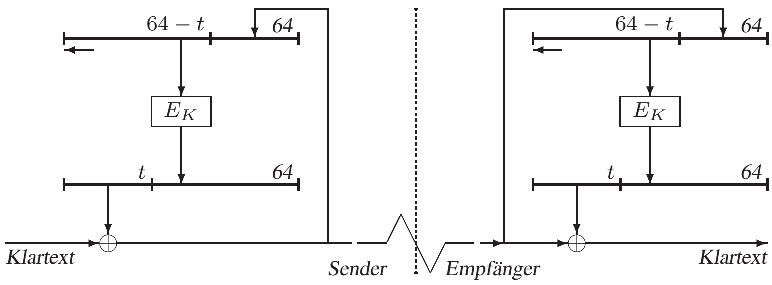
CBC-Mode (cipher block chaining; Blockverkettung des Schlüsseltextes): Jeder Klartextblock w_i wird mit dem Kryptotextblock $E_K(w_{i-1})$ bitweise (modulo 2) addiert, bevor er verschlüsselt wird (zur Verschlüsselung von w_1 wird ein Initialisierungsvektor iv verwendet:



OFB-Mode (output feedback; Rückführung der Ausgabe): Die Binär-Nachricht x wird in t-Bit Blöcke (für festes t: $1 \leq t \leq l$) zerlegt. Die Chiffrierfunktion E_K dient zur Erzeugung einer pseudozufälligen Folge von t-Bit Blöcken, die bitweise (modulo 2) zu den entsprechenden Klartextblöcken addiert werden. Als Eingabe für die Chiffrierfunktion E_K dient ein Schieberegister, das anfangs mit einem Initialisierungsvektor IV geladen ist. Bei jeder Übertragung eines t-Bit Klartextblockes w_i erzeugt die Chiffrierfunktion E_K zunächst einen Ausgabevektor, von dem nur die ersten t Bits verwendet werden. Diese dienen sowohl zur Verschlüsselung von w_i, als auch zur Modifikation des Eingaberegisters, in das sie von rechts geschoben werden:



CFB-Mode (cipher feedback; Rückführung des Kryptotextes): Zur Erneuerung des Eingaberegisters wird nicht die E_K-Ausgabe, sondern der t-Bit Kryptotextblock verwendet:



2.7 Endliche Körper

Wie wir bereits wissen, bildet $\mathbb{Z}_p$ für primes p einen endlichen Körper der Größe p. Dieser Körper läßt sich für jede Zahl $n \geq 1$ zu einem Körper der Größe p^n erweitern. Da bis auf Isomorphie nur ein Körper dieser Größe existiert, wird er einfach mit $\mathbb{F}(p^n)$ oder $\mathbb{F}_{p^n}$ bezeichnet. Um diesen Körper zu konstruieren, betrachten wir zunächst den Polynomring $\mathbb{Z}_p[x]$ über $\mathbb{Z}_p$.

Definition 102 **(Polynomring)**

Sei p prim. Dann enthält $\mathbb{Z}_p[x]$ alle Polynome

$$p(x) = a_n x^n + \cdots a_1 x + a_0$$

in der Variablen x mit Koeffizienten $a_i \in \mathbb{Z}_p$, $a_n \neq 0$. n heißt **Grad** von p (kurz: $deg(p) = n$). $\mathbb{Z}_p[x]$ bildet mit der üblichen Polynomaddition und Polynommultiplikation einen Ring.

Ein Polynom $m(x)$ ***teilt*** ein Polynom $g(x)$ (kurz: $m(x)|g(x)$), falls ein Polynom $d(x) \in \mathbb{Z}_p[x]$ existiert mit $g(x) = d(x)m(x)$. Teilt $m(x)$ die Differenz $f(x) - g(x)$ zweier Polynome, so schreiben wir hierfür

$$f(x) \equiv_{m(x)} g(x)$$

und sagen, $f(x)$ ist ***kongruent*** zu $g(x)$ modulo $m(x)$. Weiterhin bezeichne

$$f(x) \bmod m(x)$$

den bei der Polynomdivision von $f(x)$ durch $m(x)$ auftretenden ***Rest***, also dasjenige Polynom $r(x)$ vom Grad $deg(r) < deg(m)$, für das ein Polynom $d(x) \in \mathbb{Z}_p[x]$ existiert mit $f(x) = d(x)m(x) + r(x)$.

Ähnlich wie beim Übergang von $\mathbb{Z}$ zu $\mathbb{Z}_m$ können wir für ein fest gewähltes Polynom $m(x)$ vom Grad $deg(m) = n$ jedem Polynom $p(x) \in \mathbb{Z}_p[x]$ mittels

$$p(x) \mapsto p(x) \bmod m(x)$$

eindeutig ein Polynom vom Grad höchstens $n-1$ zuordnen. Auf diese Weise erhalten wir den Restklassenpolynomring $\mathbb{Z}_p[x]/m(x)$ aller Polynome vom Grad höchstens $n-1$, wobei die Addition und Multiplikation wie in $\mathbb{Z}_p[x]$, gefolgt von einer Reduktion modulo $m(x)$, definiert ist. Und wie $\mathbb{Z}_m$ ist $\mathbb{Z}_p[x]/m(x)$ genau dann ein Körper, wenn $m(x)$ keinen nichttrivialen Teiler besitzt.

Definition 103 (**irreduzibel**)

> Ein Polynom $m(x) \in \mathbb{Z}_p[x]$ heißt **irreduzibel**, falls keine Polynome $p(x), q(x) \in \mathbb{Z}_p[x]$ vom Grad $deg(p), deg(q) \geq 1$ existieren mit
>
> $$m(x) = p(x)q(x).$$

Da für jede Zahl $n \geq 1$ ein irreduzibles Polynom $m(x) = \sum_{i=0}^{n} m_i x^i \in \mathbb{Z}_p[x]$ vom Grad n existiert, lässt sich auf diese Weise für jede Primzahlpotenz p^n ein Körper der Größe p^n konstruieren. Hierbei können wir jedes Körperelement

$$a(x) = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{F}_{p^n}$$

durch die Folge $a_{n-1} \cdots a_0 \in (\mathbb{F}_p)^n$ seiner Koeffizienten darstellen. Im Fall $p = 2$ entspricht dann die Addition zweier Polynome $a(x) = \sum_{i=0}^{n-1} a_i x^i$ und $b(x) = \sum_{i=0}^{n-1} b_i x^i$ in $\mathbb{F}_{2^n}$ der bitweisen Addition modulo 2 (x-or),

$$a_{n-1} \cdots a_0 + b_{n-1} \cdots b_0 = a_{n-1} \cdots a_0 \oplus b_{n-1} \cdots b_0.$$

Die Multiplikation in $\mathbb{F}_{2^n}$ lässt sich wegen

$$a(x)b(x) = \sum_{a_i=1} x^i b(x)$$

auf die Addition und (wiederholte) Multiplikation mit dem Polynom $p(x) = x$ zurückführen. Wegen

$$x^n \equiv_{m(x)} \sum_{i=0}^{n-1} m_i x^i$$

ist

$$xb(x) = \begin{cases} \sum_{i=1}^{n-1} b_{i-1}x^i & b_{n-1} = 0, \\ \sum_{i=1}^{n-1} (b_{i-1} \oplus m_i)x^i \oplus m_0 & b_{n-1} = 1, \end{cases}$$

d.h. die Multiplikation von $b(x)$ mit x entspricht einem Linksshift um eine Stelle, dem sich im Fall $b_{n-1} = 1$ noch die Addition von $m_{n-1} \cdots m_0$ anschließt:

$$0 \cdots 010 \cdot b_{n-1} \cdots b_0 = \begin{cases} b_{n-2} \cdots b_0 0 & b_{n-1} = 0, \\ b_{n-2} \cdots b_0 0 \oplus m_{n-1} \cdots m_0 & b_{n-1} = 1. \end{cases}$$

Beispiel 104 *Sei $p = 2$ und $n = 3$. Zunächst benötigen wir ein irreduzibles Polynom $m(x) \in \mathbb{Z}_2[x]$ vom Grad 3,*

$$m(x) = a_3x^3 + a^2x^2 + a_1x + a_0.$$

Da $m(x)$ im Fall $a_0 = 0$ den nichttrivialen Teiler $p(x) = x$ hat und im Fall $a_3 = 0$ nicht den Grad 3 hat, genügt es, die 4 Kandidaten

$$\begin{aligned} m_1(x) &= x^3 + 1 \\ m_2(x) &= x^3 + x + 1 \\ m_3(x) &= x^3 + x^2 + 1 \\ m_4(x) &= x^3 + x^2 + x + 1 \end{aligned}$$

zu betrachten. Da nun aber

$$x^3 + 1 = (x + 1)(x^2 + x + 1)$$

und

$$x^3 + x^2 + x + 1 = (x + 1)(x^2 + 1)$$

ist, gibt es in $\mathbb{Z}_2[x]$ nur zwei irreduzible Polynome vom Grad 3: $x^3 + x + 1$ und $x^3 + x^2 + 1$. Nehmen wir bspw. $m(x) = x^3 + x + 1$, so ist

$$(x^2 + 1) + (x + 1) = x^2 + x$$

und

$$(x^2 + 1)(x + 1) = x^2$$

in $\mathbb{Z}_2[x]/(x^3 + x + 1)$, da

$$(x^2 + 1)(x + 1) = x^3 + x^2 + x + 1 = x^2 + (x^3 + x + 1) \equiv_{x^3+x+1} x^2$$

ist.

Wie das folgende Beispiel zeigt, lässt sich das **multiplikative Inverse** eines Polynoms $p(x) \neq 0$ in $\mathbb{F}_{p^n}$ mit dem erweiterten Euklidschen Algorithmus berechnen.

Beispiel 105 *Sei $p = 2$ und seien $m(x) = x^8 + x^4 + x^3 + x + 1$ und $a(x) = x^6 + x^4 + x + 1$ zwei Polynome. Dann können wir mit dem Euklidschen Algorithmus den (in Bezug auf den Grad) größten gemeinsamen Teiler $g(x)$ von $m(x)$ und $a(x)$ berechnen:*

i	$r_{i-1}(x)$	$=$	$d_{i+1}(x) \cdot r_i(x)$	$+\ r_{i+1}(x)$
0	$x^8 + x^4 + x^3 + x + 1$	$=$	$(x^2 + 1) \cdot (x^6 + x^4 + x + 1)$	$+\ x^2$
1	$x^6 + x^4 + x + 1$	$=$	$(x^4 + x^2) \cdot x^2$	$+\ x + 1$
2	x^2	$=$	$(x + 1) \cdot (x + 1)$	$+\ 1$
3	$x + 1$	$=$	$(x + 1) \cdot \mathbf{1}$	$+\ 0$

Es ist also $g(x) = r_3(x) = 1$. Der erweiterte Euklidsche Algorithmus berechnet nun Polynome $p_i(x)$ und $q_i(x)$ gemäß der Vorschrift

$$p_i(x) = p_{i-2}(x) - d_i(x) \cdot p_{i-1}(x), \quad \textit{wobei} \quad p_{-1}(x) = 1 \quad \textit{und} \quad p_0(x) = 0,$$

und

$$q_i(x) = q_{i-2}(x) - d_i(x) \cdot q_{i-1}(x), \quad \textit{wobei} \quad q_{-1}(x) = 0 \quad \textit{und} \quad q_0(x) = 1,$$

welche für $i = -1, 0, 1, 2, 3$ die Gleichung $p_i(x)m(x) + q_i(x)a(x) = r_i(x)$ erfüllen:

i	$p_i(x) \cdot m(x)\ +$	$q_i(x) \cdot a(x)$	$=$	$r_i(x)$
-1	$1 \cdot m(x)\ +$	$0 \cdot a(x)$	$=$	$m(x)$
0	$0 \cdot m(x)\ +$	$1 \cdot a(x)$	$=$	$a(x)$
1	$1 \cdot m(x)\ +$	$(x^2 + 1) \cdot a(x)$	$=$	x^2
2	$(x^4 + x^2) \cdot m(x)\ +$	$(x^6 + x^2 + 1) \cdot a(x)$	$=$	$x + 1$
3	$(x^5 + x^4 + x^3 + x^2 + 1) \cdot m(x)\ +$	$(x^7 + x^6 + x^3 + x) \cdot a(x)$	$=$	1

Aus der letzten Zeile können wir das multiplikative Inverse $q_3(x) = x^7 + x^6 + x^3 + x$ von $a(x)$ modulo $m(x)$ ablesen.

2.8 Der Advanced Encryption Standard (AES)

- Im September 1997 veröffentlichte das NIST eine Ausschreibung für den AES, in der eine Blocklänge von 128 Bit und variable Schlüssellängen von 128, 192 und 256 Bit gefordert wurden. Einreichungsschluss war der 15. Juni 1998.
- Von den 21 Einreichungen erfüllten 15 die geforderten Kriterien. Diese stammten aus den Ländern Australien, Belgien, Costa Rica, Deutschland, Frankreich, Großbritannien, Israel, Japan, Korea, Norwegen sowie den USA und wurden auf der 1. AES-Konferenz am 20. August 1998 als AES-Kandidaten akzeptiert.
- Im August 1999 wählte NIST auf der 2. AES-Konferenz in Rom die Finalisten MARS, RC6, Rijndael, Serpent und Twofish aus.

– Im April 2000 wurde der Rijndael-Algorithmus auf der 3. AES-Konferenz zum Sieger erklärt und im November 2001 als AES genormt.

Die wichtigsten Entscheidungskriterien waren

– Sicherheit,
– Kosten (Effizienz bei Software-, Hardware- und Smartcard-Implementationen) sowie
– Algorithmen- und Implementations-Charakeristika (unter anderem Flexibilität und Einfachheit des Designs).

Die Blocklänge und die Schlüssellänge können beim Rijndael unabhängig voneinander im Bereich 128, 192 oder 256 Bit gewählt werden. Die Rundenzahl N des Rijndael hängt wie folgt von der Blocklänge und der gewählten Schlüssellänge ab:

N	128	192	256
128	10	12	14
192	12	12	14
256	14	14	14

Wir beschränken uns hier auf die Beschreibung des AES mit 128 Bit Blocklänge und 128 Bit Schlüssellänge.

Im Einzelnen werden folgende Chiffrierschritte ausgeführt:

```
ADDROUNDKEY(K^0)
for i:= 1 downto 9 do
    SUBBYTES
    SHIFTROWS
    MIXCOLUMNS
    ADDROUNDKEY(K^i)
end
SUBBYTES
SHIFTROWS
ADDROUNDKEY(K^10)
```

1. Zuerst wird der Klartextblock x einer Addition mit dem 128-Bit Rundenschlüssel K^0 unterworfen. Diese Operation wird mit AddRoundKey bezeichnet.
2. Danach werden 9 Runden ausgeführt, wobei in jeder Runde i eine Substitution namens SubBytes, eine Permutation namens ShiftRows, eine lineare Substitution namens MixColumns und eine AddRoundKey Operation mit dem Rundenschlüssel K^i durchgeführt werden.
3. Es folgt Runde 10 mit den Operationen SubBytes, ShiftRows und AddRoundKey(K^{10}).

Der Klartext (und alle daraus berechneten Zwischenergebnisse) werden in Form eines Arrays

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

dargestellt, das wie folgt initialisiert wird:

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

Die Substitution SUBBYTES basiert auf einer 8-Bit S-Box π_{SUBBYTES}, die durch folgenden Algorithmus berechnet wird (die Indexrechnung in Zeile 8 erfolgt modulo 8).

Algorithmus 106 $\pi_{\text{SUBBYTES}}(a_7 \cdots a_0)$

1 **Eingabe:** $(a_7 \cdots a_0)$
2 $\quad z \leftarrow$ BINARYTOFIELD$(a_7 \cdots a_0)$
3 $\quad$ **if** $z \neq 0$ **then**
4 $\qquad z \leftarrow$ FIELDINV(z)
5 $\quad a_7 \cdots a_0 \leftarrow$ FIELDTOBINARY(z)
6 $\quad c_7 \cdots c_0 \leftarrow (01100011)$
7 $\quad$ **for** $i \leftarrow 0$ **to** 7 **do**
8 $\qquad b_i \leftarrow a_i \oplus a_{i+4} \oplus a_{i+5} \oplus a_{i+6} \oplus a_{i+7} \oplus c_i$ **end**
9 **Ausgabe:** $(b_7 \cdots b_0)$

Die hierbei benutzten Funktionen sind wie folgt definiert:

- FIELDINV: $\mathbb{F}(2^8) \to \mathbb{F}(2^8)$ berechnet das multiplikative Inverse im Körper $\mathbb{F}(2^8) = \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1)$,
- BINARYTOFIELD: $\{0,1\}^8 \to \mathbb{F}(2^8)$ berechnet aus der 8-Bit Koeffizienten-Darstellung das zugehörige Körperelement.
- FIELDTOBINARY: $\mathbb{F}(2^8) \to \{0,1\}^8$ berechnet die Inverse der Funktion BINARYTOFIELD.

Beispiel 107 *Wir berechnen* $\pi_{\text{SUBBYTES}}(01010011)$. *Die Funktion* BINARYTOFIELD *liefert das zugehörige Polynom*

$$z = \text{BINARYTOFIELD}(01010011) = x^6 + x^4 + x + 1.$$

Aus Beispiel 105 kennen wir bereits das multiplikative Inverse

$$x^7 + x^6 + x^3 + x$$

von z in $\mathbb{F}(2^8)$. Die Funktion FIELDTOBINARY *liefert die zugehörige Koeffizienten-Darstellung*

$$\text{FIELDTOBINARY}(x^7 + x^6 + x^3 + x) = (11001010).$$

Es folgt die Berechnung der Ausgabe $(b_7 \cdots b_0) = (11101101)$ *mittels*

$$\begin{aligned} b_0 &= a_0 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 \oplus c_0 \\ &= 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \\ &= 1 \\ b_1 &= a_1 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_0 \oplus c_1 \\ &= 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \\ &= 0 \end{aligned}$$

usw. Somit ist $\pi_{\text{SUBBYTES}}(01010011) = (11001010)$ *oder hexadezimal dargestellt:* $\pi_{\text{SUBBYTES}}(\mathit{53}) = \mathit{ED}$.

Wir können die AES S-Box in Form einer 16×16-Matrix angeben, wobei in Zeile X und Spalte Y der Wert $\pi_{\text{SUBBYTES}}(XY)$ eingetragen ist:

	Y															
X	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
⋮								⋮								
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

SHIFTROWS ist eine 128-Bit Permutation, die wie folgt definiert ist:

$$\begin{array}{|c|c|c|c|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ \hline s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ \hline s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\ \hline \end{array} \quad \mapsto \quad \begin{array}{|c|c|c|c|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ \hline s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ \hline s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ \hline s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \\ \hline \end{array}$$

MIXCOLUMNS ist eine lineare 32-Bit Substitution, die auf den Spalten der Zwischenergebnisse operiert. Zu ihrer Berechnung wird folgende Funktion benutzt:

- FIELDMULT: $\mathbb{F}(2^8) \times \mathbb{F}(2^8) \to \mathbb{F}(2^8)$ führt die Multiplikation im Körper $\mathbb{F}(2^8)$ aus.

Algorithmus 108 MIXCOLUMNS(c_0, c_1, c_2, c_3)

Eingabe: (c_0, c_1, c_2, c_3)
  for $i \leftarrow 0$ **to** 3 **do**
    $t_i \leftarrow \textsc{BinaryToField}(c_i)$ **end**
  $u_0 \leftarrow \textsc{FieldMult}(x, t_0) + \textsc{FieldMult}(x+1, t_1) + t_2 + t_3$
  $u_1 \leftarrow \textsc{FieldMult}(x, t_1) + \textsc{FieldMult}(x+1, t_2) + t_3 + t_0$
  $u_2 \leftarrow \textsc{FieldMult}(x, t_2) + \textsc{FieldMult}(x+1, t_3) + t_0 + t_1$
  $u_3 \leftarrow \textsc{FieldMult}(x, t_3) + \textsc{FieldMult}(x+1, t_0) + t_1 + t_2$
  for $i \leftarrow 0$ **to** 3 **do**
    $c_i \leftarrow \textsc{FieldToBinary}(u_i)$ **end**
Ausgabe: (c_0, c_1, c_2, c_3)

MixColumns führt eine lineare Transformation in $(\mathbb{F}_{2^8})^4$ aus, die sich auch wie folgt beschreiben lässt (hierbei benutzen wir die Koeffizientendarstellung der Polynome in $\mathbb{F}_{2^8}$, etwa 03 für $x+1$):

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \mapsto \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Besonders elegant lässt sich die Operation MixColumns im Polynom-Restklassenring $\mathbb{F}_{2^8}[z]/(z^4+1)$ beschreiben. Sei $c(z) = \sum_{i=0}^{3} c_i z^i$ das durch die Spalte (c_0, c_1, c_2, c_3) repräsentierte Polynom in diesem Ring und sei $a(z)$ das Polynom $a(z) = 03z^3 + 01z^2 + 01z + 02$. Dann ist

$$\textsc{MixColumns}(c(z)) = a(z)c(z).$$

D.h., MixColumns ist eine multiplikative Chiffre mit festem Schlüssel $a(z)$ im Ring $\mathbb{F}_{2^8}[z]/(z^4+1)$. Da das Polynom z^4+1 nicht irreduzibel in $\mathbb{F}_{2^8}[z]$ ist, ist $\mathbb{F}_{2^8}[z]/(z^4+1)$ zwar kein Körper. Da jedoch $a(z)$ invertierbar im Ring $\mathbb{F}_{2^8}[z]/(z^4+1)$ ist, kann die Inverse zu MixColumns mittels

$$\textsc{MixColumns}^{-1}(c(z)) = a^{-1}(z)c(z)$$

berechnet werden, wobei $a^{-1}(z) = 0Bz^3 + 0Dz^2 + 09z + 0E$ ist.

Die Schlüsselgenerierung. Beim 10-Runden AES mit Block- und Schlüssellänge $l = k = 128$ werden 11 Rundenschlüssel $K^0, \ldots, K^{10}$ der Länge 128 benutzt. Jedes K^i besteht also aus 16 Bytes bzw. 4 Worten mit jeweils 4 Bytes. Reihen wir die 11 Rundenschlüssel aneinander, so entsteht ein Array $w[0], \ldots, w[43]$ von 44 Worten, die gemäß folgendem Algorithmus aus dem 128-Bit Schlüssel K berechnet werden.

Algorithmus 109 KeyExpansion(K)

Eingabe: $K = K[0] \cdots K[15]$
  $RCon[1] \leftarrow 01000000$

$RCon[2] \leftarrow 02000000$
$RCon[3] \leftarrow 04000000$
$RCon[4] \leftarrow 08000000$
$RCon[5] \leftarrow 10000000$
$RCon[6] \leftarrow 20000000$
$RCon[7] \leftarrow 40000000$
$RCon[8] \leftarrow 80000000$
$RCon[9] \leftarrow 1B000000$
$RCon[10] \leftarrow 36000000$
for $i \leftarrow 0$ **to** 3 **do**
  $w[i] \leftarrow (K[4i], K[4i+1], K[4i+2], K[4i+3])$ **end**
for $i \leftarrow 4$ **to** 43 **do**
  $temp \leftarrow w[i-1]$
  if $i \equiv_4 0$ **then**
    $temp \leftarrow \textsc{SubWord}(\textsc{RotWord}(temp)) \oplus RCon[i/4]$
  $w[i] \leftarrow w[i-4] \oplus temp$ **end**
Ausgabe: $w[0] \ldots w[43]$

Die hierbei benutzten Funktionen sind wie folgt definiert:

– $\textsc{RotWord} : \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \to \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8$ führt eine zyklische Verschiebung der 4 Eingabebytes um ein Byte nach links durch:
$$\textsc{RotWord}(B_0, B_1, B_2, B_3) = (B_1, B_2, B_3, B_0),$$

– $\textsc{SubWord} : \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \to \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8$ ersetzt jedes Eingabebyte B_i durch $\pi_{\textsc{SubBytes}}(B_i)$:
$$\begin{aligned}&\textsc{SubWord}(B_0, B_1, B_2, B_3)\\ &\quad = \textsc{SubBytes}(B_0, B_1, B_2, B_3)\\ &\quad = (\pi_{\textsc{SubBytes}}(B_0), \pi_{\textsc{SubBytes}}(B_1), \pi_{\textsc{SubBytes}}(B_2), \pi_{\textsc{SubBytes}}(B_3))\end{aligned}$$

3 Zahlentheoretische Grundlagen

In diesem Abschnitt stellen wir die Hilfsmittel aus der Zahlentheorie bereit, die wir zum Verständnis der Public-Key Verfahren, die im nächsten Abschnitt vorgestellt werden, benötigen.

Satz 110 (Euler-Fermat) *Für alle $a \in \mathbb{Z}_m^*$ gilt*

$$a^{\varphi(m)} \equiv_m 1$$

Beweis: Sei $\mathbb{Z}_m^* = \{u_1, \ldots, u_{\varphi(m)}\}$ und $a \in \mathbb{Z}_m^*$ beliebig. Wegen $a \cdot u_i \not\equiv_m a \cdot u_j$ für $i \not\equiv_m j$ folgt $\mathbb{Z}_m^* = \{a \cdot u_1, \ldots, a \cdot u_{\varphi(m)}\}$. Dies impliziert $\prod_{i=1}^{\varphi(m)} u_i \equiv_m \prod_{i=1}^{\varphi(m)} a \cdot u_i \equiv_m a^{\varphi(m)} \cdot \prod_{i=1}^{\varphi(m)} u_i$. Also muss $a^{\varphi(m)} \equiv_m 1$ sein. ∎

Korollar 111 (Satz von Fermat) *Ist p eine Primzahl und a eine natürliche Zahl, die nicht durch p teilbar ist, also $a \in \mathbb{Z}_p^*$, so ist $a^{p-1} - 1$ durch p teilbar:*

$$a^{p-1} \equiv_p 1.$$

3.1 Diskrete Logarithmen

Nehmen wir ein beliebiges Element a aus $\mathbb{Z}_m^*$ und betrachten die Folge $a^0 = 1$, $a^1 = a$, a^2, a^3, …, so wissen wir nach dem Satz von Euler-Fermat, dass spätestens für $e = \varphi(m)$ wieder $a^e = 1$ gilt.

Definition 112 (Ordnung)

Es sei $m \geq 1$ und $\mathrm{ggT}(a, m) = 1$. Die **Ordnung** von a modulo m ist

$$\mathrm{ord}_m(a) = \min\{e > 1 \mid a^e \equiv_m 1\}.$$

Für das folgende besonders interessant sind Elemente a aus $\mathbb{Z}_m^*$, die ganz $\mathbb{Z}_m^*$ aufspannen.

Definition 113 (Primitivwurzel/Erzeuger)

Eine Zahl g heißt **Primitivwurzel** modulo m (oder **Erzeuger** von $\mathbb{Z}_m^*$), falls $\mathbb{Z}_m^* = \{g^0, g^1, g^2, \ldots, g^{\varphi(m)-1}\}$ ist.

Ein Element $a \in \mathbb{Z}_m^*$ ist also genau dann ein Erzeuger, wenn $\mathrm{ord}_m(a) = \varphi(m)$ ist. Falls $\mathbb{Z}_m^*$ einen Erzeuger besitzt, wird $\mathbb{Z}_m^*$ auch **zyklisch** genannt. Da $\{a^0, a^1, a^2, \ldots, a^{\mathrm{ord}_m(a)-1}\}$ eine Untergruppe von $\mathbb{Z}_m^*$ ist, ist $\mathrm{ord}_m(a)$ für alle $a \in \mathbb{Z}_m^*$ ein Teiler von $\varphi(m)$, d. h. $\varphi(m) \equiv_{\mathrm{ord}_m(a)} 0$. Allgemeiner gilt (siehe Übungen)

$$a^i \equiv_m a^j \;\Leftrightarrow\; i \equiv_{\mathrm{ord}_m(a)} j.$$

Satz 114 (Gauß) *Genau für* $m \in \{1, 2, 4, p^k, 2p^k \mid 2 < p \text{ prim}\}$ *ist* $\mathbb{Z}_m^*$ *zyklisch.*
(Ohne Beweis)

Wählen wir als Basis einen Erzeuger g von $\mathbb{Z}_m^*$, so ist die Exponentiation $e \mapsto g^e$ eine bijektive Abbildung von der Menge $\{0, 1, \ldots, \varphi(m) - 1\}$ auf $\mathbb{Z}_m^*$. Die zugehörige Umkehrabbildung spielt eine sehr wichtige Rolle in der Kryptographie.

Definition 115 (**Index/diskreter Logarithmus**)

Sei g ein Erzeuger von $\mathbb{Z}_m^*$ und $a \in \mathbb{Z}_m^*$. Dann heißt der eindeutig bestimmte Exponent $e \in \{0, 1, \ldots, \varphi(m) - 1\}$ mit

$$g^e \equiv_m a$$

Index oder **diskreter Logarithmus** modulo m von a zur Basis g (kurz: $e = \log_{m,g}(a)$).

Während die diskrete Exponentialfunktion $e \mapsto g^e$ durch **Wiederholtes Quadrieren und Multiplizieren** (siehe nächsten Abschnitt) effizient berechnet werden kann, sind bis heute keine effizienten Verfahren zur Berechnung des diskreten Logarithmus bekannt.

Beispiel 116 $m = 11$, $g = 2$.

e	0	1	2	3	4	5	6	7	8	9
2^e	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\log_{11,2}(a)$	0	1	8	2	4	9	7	3	6	5

◁

Die nächsten beiden Sätze benötigen wir zur Bestimmung der Anzahl aller Erzeuger von $\mathbb{Z}_p^*$, falls p prim ist.

Satz 117 (Euler) *Sei* $m \geq 1$, *dann gilt*

$$\sum_{d|m} \varphi(d) = m,$$

wobei die Summe über alle Teiler $d \geq 1$ *von* m *läuft.*

Beweis: Mit $\varphi_d(m) = \|\{a \in \mathbb{Z}_m \mid \mathrm{ggT}(a, m) = d\}\|$ ist $\sum_{d|m} \varphi_d(m) = m$. Wegen

$$\mathrm{ggT}(a, m) = d \quad \Longleftrightarrow \quad \mathrm{ggT}(a/d, m/d) = 1$$

ist $\varphi_d(m) = \varphi(m/d)$ und damit

$$\sum_{d|m} \varphi(d) = \sum_{d|m} \varphi(m/d) = \sum_{d|m} \varphi_d(m) = m. \qquad \blacksquare$$

Satz 118 (Lagrange) *Sei p eine Primzahl und sei $\mathrm{ggT}(a_n, p) = 1$. Dann hat die Polynomkongruenz*

$$f(x) = a_n \cdot x^n + \cdots + a_1 \cdot x^1 + a_0 \cdot x^0 \quad \equiv_p \quad 0 \tag{6}$$

höchstens n verschiedene Lösungen modulo p.

Beweis: Durch Induktion über n (es wird nur die Körpereigenschaft von $\mathbb{Z}_p$ benutzt).

$n = 0$: klar.

$n - 1 \rightsquigarrow n$: Angenommen, die Polynomkongruenz (6) habe $n + 1$ verschiedene Lösungen $x_1, \ldots, x_{n+1}$ modulo p. Mit

$$\begin{aligned} & x^i - x_{n+1}^i \\ &= \underbrace{x^i + x_{n+1}x^{i-1} + \cdots + x_{n+1}^{i-1}x}_{xh_i(x)} \underbrace{-x_{n+1}x^{i-1} - \cdots - x_{n+1}^{i-1}x - x_{n+1}^i}_{-x_{n+1}h_i(x)} \\ &= (x - x_{n+1}) \underbrace{(x^{i-1} + x_{n+1}x^{i-2} + \cdots + x_{n+1}^{i-2}x + x_{n+1}^{i-1})}_{h_i(x)} \end{aligned}$$

folgt

$$\begin{aligned} g(x) &= f(x) - f(x_{n+1}) \\ &= a_n \cdot (x^n - x_{n+1}^n) + \cdots + a_1 \cdot (x - x_{n+1}) \\ &= (x - x_{n+1}) \underbrace{\sum_{i=1}^{n} a_i h_i(x)}_{h(x)} . \end{aligned}$$

Aus $g(x_j) = f(x_j) - f(x_{n+1}) \equiv_p 0$ und $x_j - x_{n+1} \not\equiv_p 0$ folgt $h(x_j) \equiv_p 0$ (für $j = 1, \ldots, n$), was im Widerspruch zur Induktionsvoraussetzung steht, da das Polynom $h(x)$ vom Grad $n - 1$ ist. ■

Satz 119 (Gauß) *Ist p prim, so gibt es genau $\varphi(p - 1)$ Erzeuger von $\mathbb{Z}_p^*$.*

Beweis: Für jeden Teiler d von $\varphi(p) = p - 1$ sei

$$S_d = \{a \in \mathbb{Z}_p^* \mid \mathrm{ord}_p(a) = d\}.$$

Da die Menge S_{p-1} aus allen Erzeugern von $\mathbb{Z}_p^*$ gebildet wird, müssen wir $\|S_{p-1}\| = \varphi(p-1)$ zeigen. Zuerst überlegen wir uns, dass die Menge S_d für ein beliebiges $a \in S_d$ in der Menge $\{a^e \mid e \in \mathbb{Z}_d^*\}$ enthalten ist, woraus $\|S_d\| \leq \varphi(d)$ folgt.

Jedes $a \in S_d$ erfüllt die Kongruenz $x^d \equiv_p 1$, die nach dem Satz von Lagrange (Satz 118) höchstens d verschiedene Lösungen modulo p besitzt. Da neben a auch $a^2, \ldots, a^d$ Lösungen dieser Kongruenz sind, die alle modulo p verschieden sind, folgt $S_d \subseteq \{a, a^2, \ldots, a^d\}$.

Angenommen, es gibt ein a^e in S_d mit $\mathrm{ggT}(e,d) = k > 1$. Dann folgt $(a^e)^{d/k} \equiv_p (a^d)^{e/k} \equiv_p 1^{e/k} \equiv_p 1$, also $\mathrm{ord}_p(a^e) \le d/k < d$ (Widerspruch).

Da die Mengen S_d eine Partition von $\mathbb{Z}_p^*$ bilden, folgt mit Satz 117, dass

$$\sum_{d|(p-1)} \|S_d\| = p - 1 = \sum_{d|(p-1)} \varphi(d)$$

ist. Da aber, wie oben gezeigt, $\|S_d\| \le \varphi(d)$ ist, muss $\|S_d\| = \varphi(d)$ und insbesondere $\|S_{p-1}\| = \varphi(p-1)$ gelten. ■

Satz 120 *Sei p prim. Dann ist $a \in \mathbb{Z}_p^*$ genau dann ein Erzeuger, wenn für jeden Primteiler q von $p-1$ gilt:*

$$a^{(p-1)/q} \not\equiv_n 1.$$

Beweis: Falls $a \in \mathbb{Z}_p^*$ ein Erzeuger ist, so gilt $a^e \not\equiv_p 1$ für alle Exponenten $e \in \{1, \dots, p-2\}$ und somit auch für alle Exponenten e der Form $(p-1)/q$, q prim.

Ist dagegen $a \in \mathbb{Z}_p^*$ kein Erzeuger, so ist $\mathrm{ord}_p(a) < p-1$, und da $\mathrm{ord}_p(a)$ ein Teiler von $p-1$ ist, existiert eine Zahl $d \ge 2$ mit $d \cdot \mathrm{ord}_p(a) = p-1$. Sei q ein beliebiger Primteiler von d. Dann gilt

$$a^{(p-1)/q} \equiv_p a^{d\,\mathrm{ord}_p(a)/q} \equiv_p (a^{\mathrm{ord}_p(a)})^{d/q} \equiv_p 1.$$

■

Folgender probabilistische Algorithmus berechnet einen Erzeuger $a \in \mathbb{Z}_p^*$, falls alle Primteiler q von $p-1$ bekannt sind.

Algorithmus 121 COMPUTEGENERATOR$(p, q_1, \dots, q_k)$

Eingabe: Primzahl p, Primteiler $q_1, \dots, q_k$ von $p-1$
  repeat
    rate zufällig $a \in \{2, \dots, p-1\}$
  until $a^{(p-1)/q_i} \not\equiv_p 1$ für $i = 1, \dots, k$
Ausgabe: a

Da $\varphi(n) \ge n/(2\ln\ln n)$ für hinreichend große n ist, findet der Algorithmus in jedem Schleifendurchlauf mit Wahrscheinlichkeit $\varphi(p-1)/(p-2) \ge 1/(2\ln\ln(p-1))$ einen Erzeuger. Die erwartete Anzahl der Schleifendurchläufe ist also $O(\ln\ln p)$.

3.2 Modulares Potenzieren

Modulare Potenzen $a^e \bmod m$ lassen sich durch **Wiederholtes Quadrieren und Multiplizieren** in $O(\log^2 a + \log e \log^2 m)$ Schritten berechnen (Zeitkomplexität $O(n^3)$, wobei n die Länge der Eingabe in Binärdarstellung ist).

Dazu sei $e = \sum_{i=0}^{r} e_i \cdot 2^i$ mit $r = \lfloor \mathrm{ld}\, e \rfloor$, d.h. die e_i bilden die Dualdarstellung von e. Dann können wir den Exponenten e sukzessive mittels $b_0 = e_0$ und $b_{i+1} = b_i + e_i 2^i$ für $i = 0, \ldots, r-1$ zu $e = b_r$ berechnen. Der folgende Algorithmus berechnet nach diesem Schema in der Variablen z die Potenzen $a^{b_i} \bmod m$ für $i = 0, \ldots, r$.

Algorithmus 122 $\textsc{ModPot}(a, e, m)$

$y \leftarrow a$
$z \leftarrow a^{e_0} \bmod m$
for $i \leftarrow 1$ **to** r **do**
  $y \leftarrow y^2 \bmod m$
  $z \leftarrow z \cdot y^{e_i} \bmod m$
end
return z

Beispiel 123 *Sei* $a = 1920$, $e = 19$ *und* $m = 2773$. *Dann erhalten wir mit obigem Algorithmus* $1920^{19} \bmod 2773 = 1868$*:*

i	e_i	$y = a^{2^i}$	$z = a^{b_i}$
0	1	1920	1920
1	1	1083	2383
2	0	2683	2383
3	0	2554	2383
4	1	820	**1868**

wobei $b_i = \sum_{j=0}^{i} e_j \cdot 2^j$. ◁

Alternativ können wir auch das **Horner-Schema** zur Berechnung von e benutzen: Sei $c_r = e_r = 1$ und sei $c_{i-1} = 2c_i + e_{i-1}$ für $i = r, \ldots, 1$, dann ist $e = c_0$. Dies führt auf folgenden Algorithmus, der in der Variablen z die Potenzen $a^{c_i} \bmod m$ für $i = r, \ldots, 0$ berechnet.

Algorithmus 124 $\textsc{ModPot}^*(a, e, m)$

$z \leftarrow 1$
for $i \leftarrow r$ **downto** 0 **do**
  $z \leftarrow z^2 \cdot a^{e_i} \bmod m$
end
return z

Beispiel 125 *Mit diesem Algorithmus erhalten wir natürlich ebenfalls* $1920^{19} \bmod 2773 = 1868$*:*

i	e_i	$z = a^{c_i}$
4	1	1920
3	0	1083
2	0	2683
1	1	1016
0	1	**1868**

wobei $c_i = \sum_{j=i}^{4} e_j \cdot 2^{j-i}$. ◁

3.3 Quadratische Reste

In diesem Abschnitt beschäftigen wir uns mit dem Problem, Lösungen für eine quadratische Kongruenzgleichung

$$x^2 \equiv_m a \tag{7}$$

zu bestimmen. Zunächst gehen wir der Frage nach, wie sich feststellen lässt, ob überhaupt Lösungen existieren.

Definition 126 (**Quadratischer (Nicht-)Rest, Legendre-Symbol**)

Ein Element $a \in \mathbb{Z}_m^*$ heißt **quadratischer Rest** modulo m (kurz: $a \in \mathsf{QR}_m$), falls ein $x \in \mathbb{Z}_m^*$ existiert mit $x^2 \equiv_m a$. Andernfalls heißt a **quadratischer Nichtrest** (kurz: $a \in \mathsf{QNR}_m$).

Sei $p > 2$ eine Primzahl und $\mathrm{ggT}(a, p) = 1$. Dann heißt

$$\mathcal{L}(a,p) = \left(\frac{a}{p}\right) = \begin{cases} 1, & a \in \mathsf{QR}_p \\ -1, & a \in \mathsf{QNR}_p \end{cases}$$

das **Legendre-Symbol** von a modulo p.

Die Kongruenzgleichung (7) besitzt also für ein $a \in \mathbb{Z}_m^*$ genau dann eine Lösung, wenn $a \in \mathsf{QR}_m$ ist. Wie das folgende Lemma zeigt, kann die Lösbarkeit von (7) für primes m effizient entschieden werden. Am Ende dieses Abschnitts werden wir noch eine andere Methode zur effizienten Berechnung des Legendre-Symbols kennenlernen.

Lemma 127 *Sei $a \in \mathbb{Z}_p^*$, $p > 2$ prim, und sei $k = \log_{p,\,g}(a)$ für einen beliebigen Erzeuger g von $\mathbb{Z}_p^*$. Dann sind die folgenden drei Bedingungen äquivalent:*

a) $a^{(p-1)/2} \equiv_p 1$,

b) k *ist gerade,*

c) $a \in \mathsf{QR}_p$.

Beweis:

$a \Rightarrow b$: Angenommen, $a \equiv_p g^k$ für ein ungerades $k = 2 \cdot j + 1$. Dann ist

$$a^{(p-1)/2} \equiv_p \underbrace{g^{j\cdot(p-1)}}_{=1} \underbrace{g^{(p-1)/2}}_{\neq 1} \equiv_p g^{(p-1)/2} \not\equiv_p 1.$$

$b \Rightarrow c$: Ist $a \equiv_p g^k$ für $k = 2j$ gerade, so folgt $a \equiv_p (g^j)^2$, also $a \in \mathsf{QR}_p$.

$c \Rightarrow a$: Sei $a \in \mathsf{QR}_p$, d. h. $b^2 \equiv_p a$ für ein $b \in \mathbb{Z}_p^*$. Dann folgt mit dem Satz von Fermat,

$$a^{(p-1)/2} \equiv_p b^{p-1} \equiv_p 1.$$

■

Satz 128 (Eulers Kriterium) *Für alle $a \in \mathbb{Z}_p^*$, $p > 2$ prim, gilt*

$$a^{(p-1)/2} \equiv_p \left(\frac{a}{p}\right).$$

Beweis: Nach obigem Lemma reicht es zu zeigen, dass $a^{(p-1)/2} \equiv_p \pm 1$ sein muss. Da jedoch die Kongruenz $x^2 \equiv_p 1$ nach dem Satz von Lagrange (Satz 118) nur die beiden Lösungen 1 und -1 hat, folgt dies aus der Tatsache, dass $a^{(p-1)/2}$ Lösung dieser Kongruenz ist. ■

Korollar 129 *Für alle $a, b \in \mathbb{Z}_p^*$, $p > 2$ prim, gilt*

a) $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = \begin{cases} 1, & p \equiv_4 1, \\ -1, & p \equiv_4 3, \end{cases}$

b) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right)$.

Als weiteres Korollar aus Eulers Kriterium erhalten wir eine Methode, quadratische Kongruenzgleichungen im Fall $p \equiv_4 3$ zu lösen. Für beliebige Primzahlen p ist kein effizienter, deterministischer Algorithmus bekannt. Es gibt jedoch einen probabilistischen Algorithmus von Adleman, Manders und Miller (1977).

Korollar 130 *Sei $p > 2$ prim, dann besitzt die quadratische Kongruenzgleichung $x^2 \equiv_p a$ für jedes $a \in \mathsf{QR}_p$ genau zwei Lösungen. Im Fall $p \equiv_4 3$ sind dies $\pm a^k$ (für $k = (p+1)/4$) von denen genau eine ein quadratischer Rest ist.*

Beweis: Sei $a \in \mathsf{QR}_p$, d. h. es existiert ein $b \in \mathbb{Z}_p^*$ mit $b^2 \equiv_p a$. Mit b ist auch $-b$ eine Lösung von $x^2 \equiv_p a$, die von b verschieden ist (p ist ungerade). Nach Lagrange (Satz 118) existieren keine weitere Lösungen.

Sei nun $p \equiv_4 3$. Dann gilt

$$\left(\frac{-b}{p}\right) = \left(\frac{-1}{p}\right) \cdot \left(\frac{b}{p}\right) = -\left(\frac{b}{p}\right)$$

nach Korollar 129. Demnach ist genau eine der beiden Lösungen $\pm b$ ein quadratischer Rest. Schließlich liefert Eulers Kriterium für $k = (p+1)/4$

$$a^{2k-1} = a^{(p-1)/2} \equiv_p 1$$

Also folgt $(a^k)^2 \equiv_p a$. ■

Zum Schluss dieses Abschnitts erweitern wir das Legendre-Symbol zum Jacobi-Symbol und zeigen, wie auch dieses effizient berechnet werden kann.

Definition 131 (**Jacobi-Symbol**)

Das **Jacobi-Symbol** ist für alle ungeraden $m > 3$ und $a \in \mathbb{Z}_m^*$ durch

$$\mathcal{J}(a,m) = \left(\frac{a}{m}\right) = \left(\frac{a}{p_1}\right)^{e_1} \cdots \left(\frac{a}{p_r}\right)^{e_r}$$

definiert, wobei $p_1^{e_1} \cdots p_r^{e_r}$ die Primfaktorzerlegung von m ist.

Man beachte, dass im Gegensatz zum Legendre-Symbol die Eigenschaft $\left(\frac{a}{m}\right) = 1$ für ein $a \in \mathbb{Z}_m^*$ nicht unbedingt mit $a \in \mathsf{QR}_m$ gleichbedeutend ist. Interessanterweise ist das Jacobi-Symbol auch ohne Kenntnis der Primfaktorzerlegung des Moduls effizient berechenbar. Der Algorithmus basiert auf den folgenden beiden Sätzen, die wir ohne Beweis angeben.

Satz 132 (Quadratisches Reziprozitätsgesetz, Gauß) *Es seien $m, n > 2$, ungerade und teilerfremd. Dann gilt*

$$\left(\frac{n}{m}\right) = (-1)^{(m-1)\cdot(n-1)/4}\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{m}{n}\right), & m \equiv_4 n \equiv_4 3, \\ \left(\frac{m}{n}\right), & \text{sonst} \end{cases}$$

Satz 133 *Für ungerades m gilt*

$$\left(\frac{2}{m}\right) = (-1)^{\frac{m^2-1}{8}} = \begin{cases} 1, & m \equiv_8 \pm 11, \\ -1, & m \equiv_8 \pm 3. \end{cases}$$

Korollar 134 *Das Jacobi-Symbol ist effizient berechenbar.*

Beweis: Dies folgt, ähnlich wie beim Euklidschen Algorithmus, aus folgenden Gleichungen.

$$\left(\frac{a}{m}\right) = \begin{cases} 1, & \text{falls } a = 1 \\ \left(\frac{m \bmod a}{a}\right) \cdot (-1)^{(a-1)(m-1)/4}, & \text{falls } a \text{ ungerade} \\ \left(\frac{b}{m}\right), & \text{falls } a = 2^{2k} \cdot b,\ b \text{ ungerade} \\ \left(\frac{b}{m}\right) \cdot (-1)^{(m^2-1)/8}, & \text{falls } a = 2^{2k+1} \cdot b,\ b \text{ ungerade.} \end{cases}$$

■

Beispiel 135 *Das Jacobi-Symbol zu 73 modulo 83:*

$$\mathcal{J}(73, 83) = \left(\frac{73}{83}\right) = \left(\frac{83}{73}\right) = \left(\frac{2}{73}\right) \cdot \left(\frac{5}{75}\right) = \left(\frac{73}{5}\right) = \left(\frac{5}{3}\right) = -1. \quad \triangleleft$$

3.4 Primzahlen

Sei $\pi : \mathbb{N} \to \mathbb{N}_0$ mit

$$\pi(n) = \|\{2 \leq p \leq n \mid p \in \mathbb{P}\}\|$$

die Anzahl der Primzahlen kleiner gleich n. Mit $\pi_{a,\,m}(n)$ bezeichnen wir die Anzahl der Primzahlen kleiner gleich n, die von der Form $p = m \cdot k + a$ für ein $k \in \mathbb{N}$ sind.

Satz 136 (Primzahlsatz, Dirichlet, Hadamard, de la Vallée Poussin)
Ist $\mathrm{ggT}(a, m) = 1$*, so gilt**

$$\pi_{a,\,m}(n) \sim \frac{n}{\varphi(m) \cdot \ln n}$$

Insbesondere gilt also

$$\pi(n) \sim \frac{n}{\ln n}.$$

n	10	100	1 000	10 000	10 100	100 000	1 000 000	10 000 000
$\pi(n)$	4	25	168	1 229	1 240	9 592	78 498	664 579
$n/\ln n$	4,34	21,7	144,8	1 085,8	1 095,5	8 685,9	72 382,2	620 420,7

Beispiel 137 *Die Anzahl der Primzahlen in einem Intervall* $[n,\, m]$ *ist demnach näherungsweise* $\frac{m}{\ln m} - \frac{n}{\ln n}$*. Für das Intervall* $[10\,000,\, 10\,100]$ *ergibt sich z. B. ein Näherungswert von* $9{,}674 \approx 10$*, während der tatsächliche Wert gleich 11 ist.* ◁

Beispiel 138 *Für die Anzahl* $\|\mathbb{P}_{100}\|$ *aller 100-stelligen Primzahlen (in Dezimaldarstellung) erhalten wir z. B. den Näherungswert*

$$\|\mathbb{P}_{100}\| \approx \frac{10^{100}}{100 \cdot \ln 10} - \frac{10^{99}}{99 \cdot \ln 10} \approx 3{,}91 \cdot 10^{97}.$$

Vergleicht man diese Zahl mit der Anzahl aller 100-stelligen Dezimalzahlen, so sehen wir, dass ungefähr jede 230-te 100-stellige Dezimalzahl prim ist. ◁

Der Beweis des Primzahlsatzes ist sehr aufwendig. Mit elementaren Mitteln lässt sich jedoch folgender Satz beweisen, der für die meisten Anwendungen vollkommen ausreicht.

Satz 139 (Tschebyscheff) *Für alle* $n > 200$ *gilt* $\pi(n) > \frac{2 \cdot n}{3 \cdot \ln n}$.
(Ohne Beweis)

* $f(n) \sim g(n)$ bedeutet $\lim_{n\to\infty} f(n)/g(n) = 1$.

Der Solovay-Strassen Test

Eulers Kriterium besagt, dass im Falle p prim für alle $a \in \mathbb{Z}_p^*$ die Kongruenz $a^{(p-1)/2} \equiv_p \left(\frac{a}{p}\right)$ gilt. Der Solovay-Strassen-Test basiert auf der Tatsache, dass für zusammengesetztes n mindestens die Hälfte aller $a \in \mathbb{Z}_n^*$ diese Kongruenz nicht erfüllen. Für $n \geq 3$ sei

$$\mathcal{P}_n = \{a \in \mathbb{Z}_n^* \mid \left(\tfrac{a}{n}\right) \not\equiv_n a^{(n-1)/2}\}.$$

Satz 140 *Eigenschaften von $\mathcal{P}_n$:*

a) $n \in \mathbb{P} \quad \Rightarrow \quad \mathcal{P}_n = \emptyset$,

b) n *zusammengesetzt und ungerade* $\quad \Rightarrow \quad \|\mathcal{P}_n\| \geq \frac{\varphi(n)}{2}$.

Beweis: Teil a folgt unmittelbar aus Eulers Kriterium. Für Teil b sei n zusammengesetzt und ungerade. Es ist leicht zu sehen, dass die Menge $U = \mathbb{Z}_n^* - \mathcal{P}_n$ eine Untergruppe von $\mathbb{Z}_n^*$ bildet. Da somit $\|U\|$ ein Teiler von $\varphi(n)$ ist, reicht es zu zeigen, dass mindestens ein Element $w \in \mathcal{P}_n$ existiert.

1. Fall : n ist quadratfrei, d. h. $n = p_1 \cdots p_k$ für k verschiedene Primzahlen. Dann garantiert der Chinesische Restsatz die Existenz einer Zahl $a \in \mathbb{Z}_n^*$ mit $a \equiv_{p_1} s$ und $a \equiv_{p_i} 1$ für $i = 2, \ldots, k$, wobei s beliebig aus QNR_{p_1} gewählt ist. Nun gilt

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right) \cdot \left(\frac{a}{p_2}\right) \cdot \ldots \cdot \left(\frac{a}{p_k}\right) = \left(\frac{a}{p_1}\right) = -1,$$

aber $a^{(n-1)/2} \not\equiv_n -1$, da andernfalls $a^{(n-1)/2} \equiv_{p_2 \cdots p_k} -1$ folgen würde, was im Widerspruch zu $a^{(n-1)/2} \equiv_{p_2 \cdots p_k} 1$ steht.

2. Fall : n ist nicht quadratfrei, d. h. es gibt eine Primzahl p mit $p^2 | n$. Betrachte $a = 1 + {}^{n}\!/_{p} \in \mathbb{Z}_n^*$. Dann gilt

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{q_i}\right)^{e_i}$$

wobei $\prod_{i=1}^{k} q_i^{e_i}$ die Primfaktorzerlegung von n ist. Wegen $a \equiv_{q_i} 1$ folgt $\left(\frac{a}{n}\right) = 1$. Andererseits muss aber $a^{(n-1)/2} \not\equiv_n 1$ gelten, da sonst

$$1 \equiv_n (1 + {}^{n}\!/_{p})^{(n-1)/2} = \sum_{i=0}^{(n-1)/2} \binom{{}^{(n-1)}\!/_{2}}{i} ({}^{n}\!/_{p})^i \equiv_n 1 + \tfrac{n-1}{2} \cdot \tfrac{n}{p},$$

da $({}^{n}\!/_{p})^i \equiv_n 0$ für $i \geq 2$. Also folgt $\frac{n-1}{2} \cdot \frac{n}{p} \equiv_n 0$, und damit wäre p ein Teiler von ${}^{(n-1)}\!/_{2}$, was ein Widerspruch ist. ■

Obiger Satz führt unmittelbar auf folgenden effizienten probabilistischen Primzahltest, der nach Solovay und Strassen benannt ist ($n > 2$ und ungerade, sowie $k \geq 1$):

Algorithmus 141 SOLOVAYSTRASSEN$(n,\ k)$

while $k \geq 1$ **do**
  $k \leftarrow k - 1$
  wähle zufällig ein $a \in \{2, \ldots, n-1\}$
  if ($\mathrm{ggT}(a, n) \neq 1$ **oder** $a \in \mathcal{P}_n$**) then**
    return “zusammengesetzt”
  end
end
return “prim”

Ist n eine Primzahl, so gibt der Algorithmus mit Wahrscheinlichkeit 1 „prim“ aus, andernfalls erzeugt er mit einer Wahrscheinlichkeit größer als $1 - 2^{-k}$ die Ausgabe „zusammengesetzt“. Da obiger Algorithmus (mit beliebig kleiner Wahrscheinlichkeit) eine falsche Ausgabe produzieren kann, handelt es sich um einen sogenannten **Monte-Carlo-Algorithmus** (mit einseitigem Fehler, da es nur im Fall n zusammengesetzt zu einer falschen Ausgabe kommen kann). Im Gegensatz hierzu gibt ein sogenannter **Las-Vegas-Algorithmus** nie eine falsche Antwort. Allerdings darf ein Las-Vegas-Algorithmus (mit kleiner Wahrscheinlichkeit) die Antwort verweigern, also ein „?“ ausgeben.

3.5 Pseudo-Primzahlen

Wie wir im vorigen Abschnitt gesehen haben, geht man bei der Konstruktion eines probabilistischen Monte-Carlo Primzahltests üblicherweise so vor, dass man eine Folge von Teilmengen $\mathcal{P}_n \subseteq \mathbb{Z}_n^*$ wählt, die die folgenden drei Eigenschaften für alle $n \geq n_0$ erfüllen:

a) Für ein gegebenes $a \in \mathbb{Z}_n^*$ kann effizient, d. h. in Polynomialzeit getestet werden, ob $a \in \mathcal{P}_n$ ist.

b) Für $n \in \mathbb{P}$ ist $\mathcal{P}_n$ leer.

c) Für zusammengesetztes n enthält $\mathcal{P}_n$ mindestens einen konstanten Anteil aller Elemente von $\mathbb{Z}_n^*$, d. h. $|\mathcal{P}_n| \geq \varepsilon \cdot \varphi(n)$ für eine Konstante $0 < \varepsilon \leq 1$.

Typischerweise wählt man für $\mathcal{P}_n$ daher die Negation einer Eigenschaft, die für alle Primzahlen n und alle Elemente $a \in \mathbb{Z}_n^*$ gilt.

Im Fall des Solovay-Strassen Tests ist $\mathcal{P}_n = \{a \in \mathbb{Z}_n^* \mid \left(\frac{a}{n}\right) \not\equiv_n a^{(n-1)/2}\}$. Weist eine zusammengesetzte Zahl n für ein zufällig gewähltes $a \in \mathbb{Z}_n^*$ das gleiche Verhalten wie eine Primzahl auf; gilt also

$$\mathrm{ggT}(a,n) = 1 \qquad \text{und} \qquad \left(\tfrac{a}{n}\right) \equiv_n a^{(n-1)/2},$$

so spricht man von einer *Euler-Pseudo-Primzahl zur Basis* a.

Beispiel 142 *Sei* $n = 91$ *und* $a = 10$*:* $\qquad 10^{45} \bmod 91 = 90 \equiv_{91} -1$

$$\left(\frac{10}{91}\right) = \left(\frac{2}{91}\right) \cdot \left(\frac{5}{91}\right) = -\left(\frac{91}{5}\right) = -1 \qquad \triangleleft$$

Es liegt nahe, neben Eulers Kriterium auch den Satz von Fermat zur Konstruktion der „Testmengensequenz“ $\mathcal{P}_n$ (für $n \geq n_0$) zu verwenden. Analog zu den Euler-Pseudo-Primzahlen nennt man eine zusammengesetzte Zahl n, die den resultierenden Primzahltest bei Wahl von $a \in \mathbb{Z}_n^*$ besteht, d. h. es gilt

$$\mathrm{ggT}(a,n) = 1 \qquad \text{und} \qquad a^{n-1} \equiv_n 1,$$

eine *Fermat-Pseudo-Primzahl* oder einfach *Pseudo-Primzahl zur Basis* a. Es ist leicht zu sehen, dass jede Euler-Pseudo-Primzahl zur Basis a auch pseudo-prim zur Basis a ist.

Die Umkehrung gilt jedoch nicht. Zum Beispiel ist die Zahl 91 zwar pseudo-prim aber nicht euler-pseudo-prim zur Basis 3. Es gibt sogar Zahlen n (z. B. $n = 561$) die pseudo-prim zu jeder Basis $a \in \mathbb{Z}_n^*$ sind (sogenannte *Carmichael-Zahlen*). Für diese Zahlen ist Bedingung c in obiger Aufzählung nicht erfüllt, so dass der Satz von Fermat im Gegensatz zu Eulers Kriterium nicht unmittelbar auf einen probabilistischen Primzahltest führt.

Carmichael-Zahlen kommen allerdings nur sehr selten vor (erst 1992 konnte die Existenz unendlich vieler Carmichael-Zahlen nachgewiesen werden), so dass man auf der Basis des Satzes von Fermat einen für viele praktische Anwendungen brauchbaren Primzahltest (genauer Pseudo-Primzahltest) erhält. Wie im Beweis zu Satz 143 lässt sich nämlich zeigen, dass Bedingung c in obiger Aufzählung für $\varepsilon = {}^{1}\!/_{2}$ erfüllt ist, sofern man nur zusammengesetzte Zahlen betrachtet, die keine Carmichael-Zahl sind.

Der auf dem Satz von Fermat beruhende Pseudo-Primzahltest kann zu einem Monte-Carlo Primzahltest (dem sogenannten Miller-Rabin Test) wie folgt erweitert werden:

> Gilt für das zufällig gewählte $a \in \{2, \ldots, n-1\}$ sowohl $\mathrm{ggT}(a, n) = 1$ als auch $a^{n-1} \equiv_n 1$, so gib nicht sofort „prim“ aus, sondern berechne der Reihe nach die Zahlen $b_i = a^{(n-1)/2^i} \bmod n$ (für $0 \leq i \leq m$) bis entweder $(n-1)/2^m$ ungerade, oder $b_m \neq 1$ ist. Gib nur „prim“ aus, falls $b_m = -1$ ist.

Mit etwas zahlentheoretischem Aufwand kann gezeigt werden, dass der Miller-Rabin Test obige Bedingung c (sogar mit $\varepsilon = {}^{3}\!/_{4}$) erfüllt. Die durch den Miller-Rabin Test definierten Pseudo-Primzahlen werden *starke Pseudo-Primzahlen zur Basis* a genannt. Starke Pseudo-Primzahlen zur Basis a sind immer auch Euler-Pseudo-Primzahlen zur Basis a.
Ist $n \equiv_4 3$, so gilt hiervon auch die Umkehrung.

Es gibt nur eine Zahl $n < 2{,}5 \cdot 10^{10}$, die stark pseudo-prim zu den Basen 2, 3, 5 und 7 ist: $n = 3\,215\,031\,751 = 151 \cdot 751 \cdot 28\,315$. Unter Verwendung der verallgemeinerten Riemannschen Hypothese kann man sogar zeigen, dass es keine Zahl n gibt, die stark pseudo-prim zu allen Basen a mit $a < 2 \cdot (\ln n)^2$ ist. Falls die verallgemeinerte Riemannsche Hypothese zutrifft, folgt also, dass das Primzahlproblem in Polynomialzeit von einem deterministischen Algorithmus entschieden werden kann.

Der Miller-Rabin Test

Für $n \geq 3$ sei $n - 1 = 2^l m$, m ungerade, und

$$\mathcal{P}'_n = \{a \in \mathbb{Z}_n^* \mid a^m \not\equiv_n 1 \text{ und } a^{m2^i} \not\equiv_n -1 \text{ für } i = 0, \ldots, l\}.$$

Satz 143 *Eigenschaften von $\mathcal{P}'_n$:*

a) $n \in \mathbb{P} \quad \Rightarrow \quad \mathcal{P}'_n = \emptyset$,

b) n *zusammengesetzt und ungerade* $\quad \Rightarrow \quad \|\mathcal{P}'_n\| \geq \frac{3\varphi(n)}{4}$.

(ohne Beweis.)

Algorithmus 144 MILLER-RABIN$(n,\ k)$

while $k \geq 1$ **do**
  $k \leftarrow k - 1$
  wähle zufällig ein $a \in \{2, \ldots, n-1\}$
  if ($a \in \mathcal{P}'_n$) **then**
    return "zusammengesetzt"
  end
end
return "prim"

4 Asymmetrische Kryptosysteme

Diffie und Hellman kamen 1976 auf die Idee, dass die Geheimhaltung des Chiffrierschlüssels keine notwendige Voraussetzung für die Sicherheit eines Kryptosystems sein muss. Natürlich setzt dies voraus, dass die vom Sender und Empfänger benutzten Schlüssel k und k' voneinander verschieden sind und dass insbesondere der Dechiffrierschlüssel k' nur sehr schwer aus dem Chiffrierschlüssel k berechenbar ist. Ist dies gewährleistet, so kann jedem Kommunikationsteilnehmer X ein Paar von zusammengehörigen Schlüsseln k_X, k'_X zugeteilt werden. X kann nun den Chiffrierschlüssel k_X öffentlich bekannt geben, muss aber den Dechiffrierschlüssel k'_X unter Verschluss halten. Die Tatsache, dass sich mit k_X die Nachrichten nicht wieder entschlüsseln lassen, hat den entscheidenden Vorteil, dass k_X über einen authentisierten Kanal zum Sender gelangen kann (d.h. es ist zwar nicht notwendig, k_X geheim zu halten, aber die Herkunft von k_X muss verifizierbar sein).

- Von einem **symmetrischen Kryptosystem** spricht man, wenn die Kenntnis des Chiffrierschlüssels gleichbedeutend mit der Kenntnis des Dechiffrierschlüssels ist, der eine also leicht aus dem anderen berechnet werden kann.
- Dagegen sind bei einem **asymmetrischen Kryptosystem** nur die Dechiffrierschlüssel geheimzuhalten, während die Chiffrierschlüssel öffentlich bekanntgegeben werden können.

Für symmetrische Kryptosysteme sind auch die Bezeichnungen **konventionelles Kryptosystem**, **Kryptosystem mit geheimen Schlüsseln** oder ***Secret-Key*-Kryptosystem** üblich, wogegen asymmetrische Kryptosysteme auch häufig auch als **Kryptosysteme mit öffentlichen Schlüsseln** oder ***Public-Key*-Kryptosysteme** bezeichnet werden.

Bei einem symmetrischen Kryptosystem sind die Rollen von Sender und Empfänger untereinander austauschbar (beziehungsweise *symmetrisch*), da die Vertraulichkeit der Nachrichten auf einem *gemeinsamen Geheimnis* beruht, welches sich die beiden Kommunikationspartner in Form des zwischen ihnen vereinbarten Schlüssels verschaffen.

Der prinzipielle Unterschied zwischen symmetrischer und asymmetrischer Verschlüsselung kann sehr schön am Beispiel eines Tresors veranschaulicht werden, den Alice dazu verwendet, Bob geheime Dokumente zukommen zu lassen.

Symmetrische Verschlüsselung: Alice und Bob besitzen beide den gleichen Schlüssel k. Alice schließt mit ihrem Schlüssel die Nachricht in den Tresor ein und Bob öffnet ihn später wieder mit seinem Schlüssel. Das Tresorschloss lässt sich also mit k sowohl auf- als auch zuschließen.

Asymmetrische Verschlüsselung: Alice schließt die Nachricht mit dem Schlüssel k_B in den Tresor ein. Danach lässt sich das Tresorschloss mit diesem Schlüssel nicht mehr öffnen. Dies ist nur mit dem in Bobs Besitz befindlichen Schlüssel k'_B möglich. Obwohl also beide Schlüssel in das Schloss passen, können k_B und k'_B jeweils nur in eine von beiden Richtungen gedreht werden.

Da Alice nicht im Besitz von Bobs privatem Schlüssel k'_B ist, kann sie im Gegensatz zu Bob nicht alle mit k_B verschlüsselten Nachrichten entschlüsseln, insbesondere keine Kryptotexte, die Bob von anderen Teilnehmern zugeschickt werden. Zu beachten ist auch, dass mit den beiden Schlüsseln k_B und k'_B von Bob nur eine Nachrichtenübermittlung von Alice an Bob möglich ist. Für die umgekehrte Richtung müssen dagegen die beiden Schlüssel k_A und k'_A von Alice benutzt werden.

4.1 Das RSA-System

Das RSA-Kryptosystem basiert auf dem Faktorisierungsproblem und wurde 1978 von seinen Erfindern Rivest, Shamir und Adleman veröffentlicht. Während beim Primzahlproblem nur eine Ja-Nein-Antwort auf die Frage „Ist n prim?" gesucht wird, muss ein Algorithmus für das Faktorisierungsproblem im Falle einer zusammengesetzten Zahl mindestens einen nicht-trivialen Faktor berechnen.

Für jeden Teilnehmer des RSA-Kryptosystems werden zwei große Primzahlen p, q und Exponenten e, d mit $e \cdot d \equiv_{\varphi(n)} 1$ bestimmt, wobei $n = p \cdot q$ und $\varphi(n) = (p-1) \cdot (q-1)$ ist.

Öffentlicher Schlüssel: $k = (e, n)$,
Geheimer Schlüssel: $k' = (d, n)$.

Jede Nachricht x wird durch eine Folge $x_1, x_2, \ldots$ von natürlichen Zahlen $x_i < n$ dargestellt, die einzeln wie folgt ver- und entschlüsselt werden.

$$\begin{aligned} E((e,n),x) &= x^e \bmod n, \\ D((d,n),y) &= y^d \bmod n. \end{aligned}$$

Die Chiffrierfunktionen E und D können durch „Wiederholtes Quadrieren und Multiplizieren" effizient berechnet werden.

Der Schlüsselraum ist also

$$K = \{(c,n) \mid \text{ es ex. Primzahlen } p \text{ und } q \text{ mit } n = pq \text{ und } c \in \mathbb{Z}^*_{\varphi(n)}\}$$

und S enthält alle Schlüsselpaare $((e,n),(d,n)) \in K \times K$ mit $e \cdot d \equiv_{\varphi(n)} 1$.

Satz 145 *Für jedes Schlüsselpaar* $((e,n),(d,n)) \in S$ *und alle* $x \in \mathbb{Z}_n$ *gilt*

$$x^{e \cdot d} \equiv_n x.$$

Beweis: Sei $e \cdot d = z \cdot \varphi(n) + 1$ für eine natürliche Zahl z.

1. Fall: $\mathrm{ggT}(x,\, n) = 1$: Mit dem Satz von Euler-Fermat folgt unmittelbar

$$x^{e \cdot d} \equiv_n x^{z \cdot \varphi(n)+1} \equiv_n x.$$

2. Fall: $\mathrm{ggT}(x,\, n) = p$: Dann ist $x \in \mathbb{Z}_q^*$ und es gilt $x = l \cdot p \in \mathbb{Z}_q^*$ für eine natürliche Zahl l. Daher folgt

$$x^{e \cdot d} = x^{z \cdot (p-1) \cdot (q-1)} \cdot x = (\underbrace{x^{q-1}}_{\equiv_q 1})^{z \cdot (p-1)} x \equiv_q x$$

und

$$x^{e \cdot d} = (l \cdot p)^{e \cdot d} \equiv_p 0 \equiv_p x.$$

Somit erhalten wir $x^{ed} \equiv_{pq} x$.

3. Fall: $\mathrm{ggT}(x,\, n) = q$: analog zu Fall 2.

4. Fall: $\mathrm{ggT}(x,\, n) = n$: Dann gilt $x^{ed} \equiv_n 0 \equiv_n x$. ∎

Zur praktischen Durchführung

1. Bestimmung von p und q:

   Man beginnt mit einer Zahl x der Form $30 \cdot z$ (mit $z \in \mathbb{Z}$) und der verlangten Größenordnung (z. B. 10^{100}) und führt einen Primzahltest für $x+1$ durch. Ist die Antwort negativ, testet man der Reihe nach die Zahlen $x+7$, $x+11$, $x+13$, $x+17$, $x+19$, $x+23$, $x+29$, $x+30+1$, $x+30+7, \ldots$ bis eine Primzahl gefunden ist. Wegen $\frac{\pi(n)}{n} > \frac{2}{(3 \cdot \ln n)}$ und da nur 8 von 30 Zahlen getestet werden, sind hierzu ungefähr $\frac{8 \cdot 3 \cdot \ln x}{30 \cdot 2} = {}^2\!/_5 \cdot \ln x$ Primzahltests durchzuführen (bei 100-stelligen Dezimalzahlen sind das um die 92 Tests).

2. Bestimmung von d:

   d soll teilerfremd zu $\varphi(n) = (p-1) \cdot (q-1)$ sein. Diese Bedingung wird z. B. von jeder Primzahl größer als $\max\{p, q\}$ erfüllt.

3. Bestimmung von e:

   Da $\mathrm{ggT}(d, \varphi(n)) = 1$ ist, liefert der erweiterte Euklidsche Algorithmus das multiplikative Inverse e von d modulo $\varphi(n)$.

4. Ver- und Entschlüsselung:

   Modulares Exponentieren durch wiederholtes Quadrieren und Multiplizieren. Es gibt auch Hardware-Implementierungen, die (unter Verwendung des Chinesischen Restsatzes) mit Geschwindigkeiten von bis zu 225 Kbits/sec arbeiten und somit circa 1500 mal langsamer als der DES sind.

Kryptoanalytische Betrachtungen

1. Die Primfaktoren p und q sollten nicht zu nahe beieinander liegen, da sonst n durch das Verfahren der Differenz der Quadrate faktorisiert werden kann:

Sei $p > q$. Dann gilt mit $b = \frac{(p-q)}{2}$ und $a = p - b = q + b$

$$n = p \cdot q = (a + b) \cdot (a - b) = a^2 - b^2.$$

Um n zu faktorisieren, genügt es daher, eine Zahl $a > \sqrt{n}$ zu finden, so dass $a^2 - n = b^2$ eine Quadratzahl ist.

Für $n = 97\,343$ ist zum Beispiel $\lceil\sqrt{n}\rceil = 312$. Bereits für $a = 312$ ist $a^2 - n = 97\,344 - 97\,343 = 1 = 1^2$ eine Quadratzahl, woraus wir die beiden Faktoren $p = a + 1 = 313$ und $q = a - 1 = 311$ erhalten.

Der Aufwand für die Suche ist proportional zur Differenz $a - \sqrt{n}$, die sich wegen $\sqrt{a + x} \leq \sqrt{a} + \frac{x}{2\sqrt{a}}$ wie folgt nach oben und unten abschätzen lässt:

$$\frac{b^2}{2a} \leq a - \sqrt{a^2 - b^2} = a - \sqrt{n} = \sqrt{n + b^2} - \sqrt{n} \leq \frac{b^2}{2\sqrt{n}}.$$

Im Fall $c = p/q \geq 2$ gilt jedoch

$$\frac{b^2}{2a} = \frac{(c-1)^2 p}{4(c+1)} \geq p/12,$$

so dass dieser Angriff keinen nennenswerten Vorteil mehr bringt.

2. Das kleinste gemeinsame Vielfache $k = \mathrm{kgV}(p - 1,\ q - 1)$ sollte möglichst groß sein, damit nicht ein zu d äquivalentes $\widetilde{d}$ durch Raten oder durch eine systematische Suche gefunden werden kann. Wie folgende Äquivalenzumformungen zeigen, ist nämlich ein Exponent $\widetilde{d}$ äquivalent zu d, falls $\widetilde{d} \equiv_k d$ ist.

$$\begin{array}{ll}
 & \forall x : x^{e \cdot \widetilde{d}} \equiv_n x \\
\Leftrightarrow & \forall x : x^{e \cdot \widetilde{d}} \equiv_p x \quad \wedge \quad \forall x : x^{e \cdot \widetilde{d}} \equiv_q x \\
\Leftrightarrow & (p-1) | (e \cdot \widetilde{d} - 1) \quad \wedge \quad (q-1) | (e \cdot \widetilde{d} - 1) \qquad \text{(Satz von Gauß)} \\
\Leftrightarrow & e \cdot \widetilde{d} \equiv_{p-1} 1 \wedge \quad e \cdot \widetilde{d} \equiv_{q-1} 1 \\
\Leftrightarrow & \widetilde{d} \equiv_{p-1} d \wedge \quad \widetilde{d} \equiv_{q-1} d \\
\Leftrightarrow & \widetilde{d} \equiv_k d
\end{array}$$

Wegen

$$(p-1) \cdot (q-1) = \mathrm{kgV}(p-1,\ q-1) \cdot \mathrm{ggT}(p-1,\ q-1)$$

sollte $\mathrm{ggT}(p - 1,\ q - 1)$ also relativ klein sein, d. h. $(p - 1)$ und $(q - 1)$ müssen große, nicht gemeinsam vorkommende Primfaktoren enthalten. Um dies zu erreichen, wählt man zur Bestimmung von p und q jeweils eine große Primzahl $\widehat{p}$ bzw. $\widehat{q}$ und testet die Zahlen $l \cdot \widehat{p} + 1$, (für $l = 2, 4, 6, \ldots$) mit einem Primzahltest. Hierzu sind gewöhnlich $l/2 \leq \ln \widehat{p}$ Tests auszuführen (nach Elliot und Halberstam).

3. Ist damit zu rechnen, dass ein Benutzer die gleiche Nachricht x an mehrere Empfänger schickt, sollten die Module $n = p \cdot q$ aller Teilnehmer paarweise verschieden gewählt werden. Ist sonst nämlich $\mathrm{ggT}(e_1, e_2) = 1$ (die Wahrscheinlichkeit hierfür ist ungefähr 0,81), so kann x leicht aus e_1, e_2, $y_1 = x^{e_1} \bmod n$ und $y_2 = x^{e_2} \bmod n$ berechnet werden:

   1\. Fall: $\mathrm{ggT}(y_1, n) > 1$. Dann ist $\mathrm{ggT}(y_1, n) \in \{p, q\}$, d. h. das System ist gebrochen.

   2\. Fall: $\mathrm{ggT}(y_1, n) = 1$. Berechne mit dem erweiterten Euklidschen Algorithmus Zahlen a, b, y_1^{-1} mit $a \cdot e_1 + b \cdot e_2 = 1$ und $y_1 \cdot y_1^{-1} \equiv_n 1$, wobei o. B. d. A. $a < 0$ sei. Dann ist $y_1^{-1} \equiv_n x^{\varphi(n)-e_1}$, also

   $$(y_1^{-1})^{-a} \cdot y_2^b \equiv_n (x^{\varphi(n)-e_1})^{-a} \cdot x^{e_2 \cdot b} \equiv_n x^{e_1 \cdot a + e_2 \cdot b} \equiv_n x.$$

4. Sei $n = p \cdot q$ (mit $p, q \in \mathbb{P}$; $p > q$). Dann können p, q leicht aus n und $\varphi(n)$ berechnet werden. Wegen

   $$\varphi(n) = (p-1) \cdot (q-1) = (p-1) \cdot (n/p - 1) = -p + n + 1 - n/p$$

   erhalten wir die Gleichung

   $$p - \underbrace{(n + 1 - \varphi(n))}_{c} + n/p = 0,$$

   die auf die quadratische Gleichung $p^2 - cp + n = 0$ führt, aus der sich p zu $\frac{c+\sqrt{c^2-n}}{2}$ bestimmen lässt.

Wie wir gesehen haben, ist das RSA-System gebrochen, falls die Faktoriesierung des Moduls n gelingt. Das Brechen von RSA ist daher höchstens so schwer wie das Faktorisieren von n. Dagegen ist nicht bekannt, ob auch umgekehrt ein effizienter Algorithmus, der bei Eingabe von e, n, y ein x mit $x^e \equiv_n y$ berechnet, in einen effizienten Faktorisierungsalgorithmus für n umgewandelt werden kann. D.h., es ist offen, ob das Brechen von RSA mindestens so schwer ist wie das Faktorisieren von n. Wie der folgende satz zeigt, ist die Bestimmung des geheimen Sclüssels nicht leichter als das Faktorisieren von n, da bei Kenntnis von d leicht ein Vielfaches $v = ed - 1$ von $\varphi(n)$ bestimmt werden kann. Im Beweis benutzen wir folgendes Lemma.

Lemma 146 *Seien y, z zwei Lösungen der Kongruenz $x^2 \equiv_n a$ mit $y \not\equiv_n \pm z$. Dann ist $\mathrm{ggT}(y + z,\ n)$ ein nicht-trivialer Faktor von n.*

Beweis: Wegen $y^2 \equiv_n z^2$ existiert ein $t \in \mathbb{Z}$ mit

$$y^2 - z^2 = (y + z) \cdot (y - z) = t \cdot n.$$

Da jedoch weder $y + z$ noch $y - z$ durch n teilbar ist, folgt $\mathrm{ggT}(y + z,\ n) \notin \{1,\ n\}$. ■

Satz 147 *Es gibt einen Las-Vegas Algorithmus, der bei Eingabe von n, v einen Primfaktor von n mit Wahrscheinlichkeit $> 1/2$ ausgibt, falls $n = p \cdot q$ (p, q prim) und $v > 0$ ein Vielfaches von $\varphi(n)$ ist.*

Beweis: Betrachte folgenden probabilistischen Algorithmus RSA-FACTORIZE, der durch eine leichte Modifikation des Miller-Rabin Primzahltests entsteht.

Algorithmus 148 MILLERRABIN(n)
$(n-1 = 2^w \cdot m$ und m ungerade)

1 **choose** $a \in \{1,\ 2,\ \ldots,\ n-1\}$
2 **if** $\mathrm{ggT}(a,\ n) > 1$ **then**
3   **output** "zusammengesetzt"
4 **else**
5   $b \leftarrow a^m \bmod n$
6 **end**

7 **while** $b^2 \not\equiv 1 (\mathrm{mod}\ n)$ **do**
8   $b \leftarrow b^2 \bmod n$
9 **end**

10 **if** $b \equiv \pm 1 (\mathrm{mod}\ n)$ **then**
11   **output** "prim"
12 **else**
13   **output** "zusammengesetzt"
14 **end**

Algorithmus 149 FACTORIZE$_1(n,\ v)$
$(v = 2^w \cdot m$ und m ungerade)

1 **choose** $a \in \{1,\ 2,\ \ldots,\ n-1\}$
2 **if** $\mathrm{ggT}(a,\ n) > 1$ **then**
3   **output** $\mathrm{ggT}(a, n)$
4 **else**
5   $b \leftarrow a^m \bmod n$
6   $s \leftarrow 0$
7 **end**
8 **while** $b^2 \not\equiv 1 (\mathrm{mod}\ n)$ **and** $(s \leq w)$ **do**
9   $b \leftarrow b^2 \bmod n$
10   $s \leftarrow s + 1$
11 **end**
12 **if** $b \equiv \pm 1 (\mathrm{mod}\ n)$ **then**
13   **output** "?"
14 **else**
15   **output** $\mathrm{ggT}(b+1,\ n)$
16 **end**

Mit Lemma 146 folgt

$$b \not\equiv_n \pm 1, b^2 \equiv_n 1 \quad \Rightarrow \quad \mathrm{ggT}(b+1,\ n) \in \{p,\ q\},$$

womit die Korrektheit der Ausgabe von RSA-FACTORIZE in Zeile 13 gezeigt ist. In den folgenden Behauptungen schätzen die Wahrscheinlichkeit ab, mit der dem Algorithmus eine Faktorisierung von n gelingt.

Sei $p - 1 = 2^i \cdot m_1$ und $q - 1 = 2^j \cdot m_2$ mit m_1, m_2 ungerade und sei o. B. d. A. $i \leq j$.

Behauptung 150 $\mathrm{ggT}(m,\ p-1) = m_1$ *und* $\mathrm{ggT}(m,\ q-1) = m_2$.

Beweis: Wegen $\varphi(n) = (p-1)\cdot(q-1) = 2^{i+j} \cdot m_1 \cdot m_2 | v = 2^w m$ folgt $m_1 m_2 | m$ und $i + j \leq w$. Da nun m ungerade ist und $p - 1 = 2^i \cdot m_1$ ist, folgt $\mathrm{ggT}(m,\ p-1) = m_1$ ($\mathrm{ggT}(m,\ q-1) = m_2$ folgt analog). □

Behauptung 151 $\underbrace{\|\{a \in \mathbb{Z}_n^* \mid a^m \equiv_n 1\}\|}_{=:\alpha} = m_1 \cdot m_2$.

Beweis: Mit dem Chinesischen Restsatz folgt

$$\alpha = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^m \equiv_p 1\}\|}_{=:\beta} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^m \equiv_q 1\}\|}_{=:\gamma}.$$

Sei g ein Erzeuger von $\mathbb{Z}_p^*$. Dann gilt

$$g^{um} \equiv_p 1 \quad \Leftrightarrow \quad um \equiv_{p-1} 0.$$

Dies zeigt $\beta = \mathrm{ggT}(m,\ p-1) = m_1$. Analog folgt $\gamma = m_2$. □

Behauptung 152 $t \geq i \quad \Rightarrow \quad a^{2^t \cdot n} \not\equiv_n -1.$

Beweis:

$$\begin{array}{rlll} t \geq i & \Rightarrow & 2^t \cdot n & \equiv_{p-1} & 0 \\ & \Rightarrow & a^{2^t \cdot n} & \equiv_p & 1 \\ & \Rightarrow & a^{2^t \cdot n} & \not\equiv_p & -1 \\ & \Rightarrow & a^{2^t \cdot n} & \not\equiv_n & -1. \end{array}$$

□

Behauptung 153 $0 \leq t < i \quad \Rightarrow \quad \underbrace{\|\{a \in \mathbb{Z}_n^* \mid a^{2^t m} \equiv_n -1\}\|}_{=:\alpha_t} = 2^{2t} m_1 m_2.$

Beweis: Mit dem Chinesischen Restsatz folgt

$$\alpha_t = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^{2^t m} \equiv_p -1\}\|}_{=:\beta_t} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^{2^t m} \equiv_q -1\}\|}_{=:\gamma_t}.$$

Sei g ein Erzeuger von $\mathbb{Z}_p^*$. Dann gilt

$$g^{u 2^t m} \equiv_p -1 \quad \Leftrightarrow \quad u 2^t m \equiv_{p-1} \frac{p-1}{2}.$$

Wegen $t < i$ ist $\mathrm{ggT}(2^t m,\ p-1) = 2^t m_1$ ein Teiler von $\frac{p-1}{2} = 2^{i-1} m_1$ und daher ist $\beta_t = \mathrm{ggT}(2^t m,\ p-1) = 2^t m_1$ ($\gamma_t = 2^t m_2$ folgt analog). □

Behauptung 154 $|\{a \in \mathbb{Z}_n^* \mid a^n \equiv_n 1 \ \vee\ \exists t, 0 \leq t < i : a^{2^t \cdot n} \equiv_n -1\}|$

Beweis:

$$\begin{aligned} & |\{a \in \mathbb{Z}_n^* \mid a^n = 1 \ \vee\ \exists t, 0 \leq t < i : a^{2^t \cdot n} = -1\}| \\ \leq\ & m_1 \cdot m_2 + \sum_{t=0}^{i-1} 2^{2t} \cdot m_1 \cdot m_2 \\ =\ & m_1 \cdot m_2 \cdot (1 + \sum_{t=0}^{i-1} 2^{2t}) = m_1 m_2 \cdot (1 + \frac{2^{2i}-1}{3}) \\ =\ & m_1 m_2 ({}^2\!/\!_3 + 2^{2i}/3) \\ =\ & {}^2\!/\!_3 \underbrace{m_1 m_2}_{\leq \varphi(n)/4} + {}^1\!/\!_3 \underbrace{2^{2i} m_1 m_2}_{\leq 2^{i+j} m_1 m_2 = \varphi(n)} \\ \leq\ & \varphi(n)({}^1\!/\!_6 + {}^1\!/\!_3) \\ =\ & \varphi(n)/2. \end{aligned}$$

□ ■

■

Bemerkung 155 *Falls es möglich wäre, aus n, e, $y = x^e \bmod n$ die Parität von x in Polynomialzeit zu bestimmen, so könnte auch der gesamte Klartext x in Polynomialzeit aus n, e und y berechnet werden. Das letzte Bit des Klartextes ist also genau so sicher wie der gesamte Klartext: Falls RSA nicht total gebrochen werden kann, kann auch nicht das letzte Bit des Klartextes ermittelt werden.*

Sei nämlich

$$\textit{klartext-half}(y) = \textit{half}(x) = \begin{cases} 0 & \textit{falls } 0 \le x \le {}^{(n-1)}/_2 \\ 1 & \textit{falls } {}^{(n+1)}/_2 \le x \le n-1 \end{cases}$$

und

$$\textit{klartext-parity}(y) = \textit{parity}(x) = \begin{cases} 1 & \textit{falls } x \textit{ ungerade} \\ 0 & \textit{falls } x \textit{ gerade} \end{cases}$$

dann kann wegen half$(x) = $ *parity*$(2x)$ *die Berechnung von klartext-half*(y) *auf die Berechnung von klartext-parity*(y) *reduziert werden:*

$$\textit{klartext-half}(y) = \textit{klartext-parity}\big((2x)^e \bmod n\big) = \textit{klartext-parity}(y \cdot 2^e \bmod n).$$

Also lässt sich x *mit Hilfe von klartext-half (oder von klartext-parity) leicht durch eine Intervallschachtelung bestimmen:*

```
k ← ⌊log₂ n⌋
for i ← 0 to k do
  h[i] ← klartext-half(y)
  y ← y · 2^e mod n
end
low ← 0
high ← n
for i ← 0 to k do
  mid ← (high+low)/2
  if h[i] = 1 then
    low ← ⌈mid⌉
  else
    high ← ⌊mid⌋
  end
end
output high
```

Beispiel 156 *Sei* $n = 1\,457$, $e = 779$ *und* $y = 722$*. Angenommen, klartext-half liefert* *die Werte* $h[0\cdots k] = 10101111100$. *Dann führt die Intervallschachtelung*

i	$h[i]$	*mid*	*low*	*high*
-	-	-	0	1 457
0	1	728,5	729	1 457
1	0	1 092,5	729	1 092
2	1	910,5	911	1 092
3	0	1 001,5	911	1 001
4	1	956,0	957	1 001
5	1	978,5	979	1 001
6	1	990,0	990	1 001
7	1	995,5	996	1 001
8	1	998,5	999	1 001
9	0	1 000,0	999	1 000
10	0	999,5	999	999

auf den Klartext $x =$ *high* $= 999$. ◁

4.2 Das Rabin-System

Wie das RSA-Verfahren beruht das Rabin-System darauf, dass es zwar effiziente Algorithmen für das Testen der Primzahleigenschaft gibt, effiziente Faktorisierungsalgorithmen aber nicht bekannt sind. Im Gegensatz zum RSA-Verfahren, von dem nicht bekannt ist, dass es mindestens so schwer zu brechen ist wie das Faktorisierungsproblem, ist genau dies beim Rabin-System der Fall. Die Sicherheit des Rabin-Systems ist also äquivalent zur Schwierigkeit des Faktorisierungsproblems. Es verwendet als Falltürfunktion eine quadratische Polynomfunktion modulo dem Produkt $n = p \cdot q$ zweier großer Primzahlen. Jeder Teilnehmer wählt ein Paar p, q von Primzahlen mit der Eigenschaft $p \equiv_4 q \equiv_4 3$ und eine Zahl $e \in \{0, \dots, n-1\}$. Die Zahlen n und e werden öffentlich bekannt gegeben, die Faktorisierung von n wird jedoch geheimgehalten.

Öffentlicher Schlüssel: $e,\ n$
Geheimer Schlüssel: $p,\ q$.

Jede Nachricht m wird in eine Folge von natürlichen Zahlen $m_i \in \{0, \dots, n-1\}$ aufgeteilt, die der Reihe nach wie folgt verschlüsselt werden.

$$E(x,\ (e,n)) \quad = \quad x \cdot (x+e) \bmod n = y.$$

Zur Entschlüsselung eines Kryptotextes $y \in \{0, \dots, n-1\}$ muss der legale Empfänger B die quadratische Kongruenzgleichung $x \cdot (x+e) \equiv_n y$ lösen, die äquivalent zu der Kongruenz

$$(\underbrace{x + 2^{-1} \cdot e}_{x'})^2 \equiv_n (2^{-1} \cdot e)^2 + y,$$

(*quadratische Ergänzung*) ist, wobei $2^{-1} = {}^{(n+1)}/_{2}$ das multiplikative Inverse zu 2 modulo n ist. Unter Verwendung des folgenden Satzes können die vier Lösungen x'_i (für $1 \le i \le 4$) dieser Kongruenx effizient von B bestimmt werden.

Satz 157 *Sei $n = p \cdot q$ für Primzahlen p, q mit $p \equiv_4 q \equiv 3$. Dann besitzt die quadratische Kongruenz $x^2 \equiv_n a$ für jedes $a \in \mathsf{QR}_n$ genau vier Lösungen, wovon nur eine ein quadratischer Rest ist.*

Beweis: Mit $x^2 \equiv_n a$ besitzen wegen $n = p \cdot q$ auch die beiden quadratischen Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ Lösungen, und zwar jeweils genau zwei: y_1, y_2 und z_1, z_2, die sich nach Korollar 130 bestimmen lassen. Mit dem Chinesischen Restsatz können wir je eine Lösung der beiden Systeme

$$\begin{array}{c} r \equiv_p 1 \\ r \equiv_q 0 \end{array} \quad \text{und} \quad \begin{array}{c} s \equiv_p 0 \\ s \equiv_q 1 \end{array}$$

bestimmen, die sich aus der linearen Darstellung $k \cdot p + l \cdot q = 1$ von $\mathrm{ggT}(p, q) = 1$ zu $r = l \cdot q$ und $s = k \cdot p$ ergeben. Nun ist leicht zu sehen, dass

$$\begin{aligned} x_1 &= r \cdot y_1 + s \cdot z_1 \bmod n, \\ x_2 &= r \cdot y_1 + s \cdot z_2 \bmod n, \\ x_3 &= r \cdot y_2 + s \cdot z_1 \bmod n, \\ x_4 &= r \cdot y_2 + s \cdot z_2 \bmod n \end{aligned}$$

vier verschiedene Lösungen der quadratischen Kongruenz $x^2 \equiv_n a$ sind. Es können aber auch nicht mehr als vier Lösungen existieren, da sich daraus für mindestens eine der beiden Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ mehr als zwei Lösungen ableiten ließen.

Um schließlich zu zeigen, dass genau eine der vier Lösungen ein quadratischer Rest ist, nehmen wir o. B. d. A. an, dass $y_1 \in \mathsf{QR}_p$, $y_2 \in \mathsf{QNR}_p$ und $z_1 \in \mathsf{QR}_q$, $z_2 \in \mathsf{QNR}_q$ sind. Betrachten wir folgendes Lemma als Zwischenschritt.

Lemma 158 *Jeder quadratische Rest a modulo n ist auch quadratischer Rest modulo k, falls k Teiler von n ist.*

$$a \in \mathsf{QR}_k \wedge k | n \quad \Rightarrow \quad a \in \mathsf{QR}_k$$

Beweis:

$$\begin{aligned} a \in \mathsf{QR}_n &\Rightarrow \exists b \in \mathbb{Z}_n^* : b^2 \equiv_n a \\ &\Rightarrow \exists b \in \mathbb{Z}_k^* : b^2 \equiv_k a \\ &\Rightarrow a \in \mathsf{QR}_k. \end{aligned}$$

□

Mit diesem Lemma folgt, daß x_2, x_3, x_4 keine quadratischen Reste modulo n sein können. Weiterhin gibt es Zahlen $l \in \mathbb{Z}_p^*$, $k \in \mathbb{Z}_q^*$ mit $l^2 \equiv_p y_1$ und $k^2 \equiv_q z_1$.

$$\begin{array}{rllllll} \text{Wegen} & (r \cdot l + s \cdot k)^2 & \equiv_p & l^2 & \equiv_p & y_1 & \equiv_p x_1 \\ \text{und} & (r \cdot l + s \cdot k)^2 & \equiv_q & k^2 & \equiv_q & z_1 & \equiv_q x_1 \\ \text{folgt} & (r \cdot l + s \cdot k)^2 & \equiv_n & x_1, & & & \end{array}$$

also ist x_1 ein quadratischer Rest modulo n. ■

Als weitere zahlentheoretische Funktion mit für die Kryptographie wichtigen Eigenschaften erhalten wir also die Quadratfunktion $x^2 : \mathsf{QR}_n \to \mathsf{QR}_n$, die nach vorigem Satz bijektiv ist (falls $n = p \cdot q$ für Primzahlen p, q mit $p \equiv_4 q \equiv 3$). Ihre Umkehrfunktion $x \mapsto \sqrt{x}$ heißt **diskrete Wurzelfunktion**, und kann (nur) bei Kenntnis der Primfaktoren p und q von n effizient berechnet werden. Bis heute ist kein effizientes Verfahren bekannt, mit dem man ohne Kenntnis der Faktorisierung von n für ein $a \in \mathbb{Z}_n^*$ entscheiden kann, ob $a \in \mathsf{QR}_n$ ist oder nicht.

Das Problem, aus den vier Kandidaten den richtigen Klartext auszuwählen, kann entweder unter Ausnutzung der Redundanz der Klartextsprache (es muss sich ein „sinnvoller“ Text ergeben) oder durch Senden folgender Zusatzinformation gelöst werden. Sei $x' = x + 2^{-1} \cdot e$, dann lässt sich x' (und damit x) durch Angabe der beiden Bits

$$c_1 = \begin{cases} 0 & \text{falls } \left(\frac{x'}{n}\right) = -1, \\ 1 & \text{sonst} \end{cases} \quad \text{und} \quad c_2 = \begin{cases} 0 & \text{falls } x' \leq {}^{(n-1)}\!/_2 \\ 1 & \text{sonst} \end{cases}$$

eindeutig unter den vier Lösungen von $(x')^2 \equiv_n (2^{-1} \cdot e)^2 + y$ kennzeichnen, da

$$\left(\frac{-b}{n}\right) = \underbrace{\left(\frac{-b}{p}\right)}_{-\left(\frac{b}{p}\right)} \cdot \underbrace{\left(\frac{-b}{q}\right)}_{-\left(\frac{b}{q}\right)} = \left(\frac{b}{n}\right)$$

ist.

Beispiel 159 *Wählen wir $p = 7$, $q = 11$ und $e = 2$, so ergeben sich*

Öffentlicher Schlüssel: $e = 2,\ n = 77$
Geheimer Schlüssel: $p = 7,\ q = 11.$

Um den Klartext $x = 12$ zu verschlüsseln, wird der Kryptotext

$$y = E(12; 2, 77) = 12 \cdot (12 + 2) \bmod 77 = 14$$

erzeugt. Da $2^{-1} \cdot e = 2^{-1} \cdot 2 = 1$ ist, kann dieser durch Lösen der quadratischen Kongruenz

$$(x + 1)^2 \equiv_{77} y + 1 = 15,$$

entschlüsselt werden. Hierzu löst der legale Empfänger zunächst die beiden Kongruenzen

$$(y + 1)^2 \equiv_7 15 \equiv 1 \quad \textit{und} \quad (z + 1)^2 \equiv_{11} 15 \equiv 4$$

zu $y_{1,2} = \pm 1^2 - 1 \bmod 7$, also $y_1 = 0$, $y_2 = 5$ (da sich hier der Exponent k zu $k = \frac{p+1}{4} = 2$ ergibt; vgl. Korollar 130) und $z_{1,2} = \pm 4^3 - 1 \bmod 11$, also $z_1 = 1$, $z_2 = 8$ (da sich hier k zu $k = \frac{q+1}{4} = 3$ ergibt). Wegen $2 \cdot 11 - 3 \cdot 7 = 1$, d. h. $r = 22$, $s = -21 \bmod 77 = 56$, lassen sich $y_{1,2}$ und $z_{1,2}$ mit dem Chinesischen Restsatz zu den vier Lösungen

$$\begin{aligned} x_1 &= 22 \cdot 0 + 56 \cdot 1 \bmod 77 = 56 \\ x_2 &= 22 \cdot 0 + 56 \cdot 8 \bmod 77 = 63 \\ x_3 &= 22 \cdot 5 + 56 \cdot 1 \bmod 77 = 12 \\ x_4 &= 22 \cdot 5 + 56 \cdot 8 \bmod 77 = 19 \end{aligned}$$

zusammensetzen. ◁

Es ist klar, dass das System gebrochen ist, sobald n in seine Primfaktoren p, q zerlegt werden kann. Wie wir gleich sehen werden, sind für Zahlen n von dieser Bauart das Faktorisierungsproblem und das Problem, eine Lösung der quadratischen Kongruenz $x^2 \equiv_n a$ für ein gegebenes $a \in \mathsf{QR}_n$ zu finden, äquivalent. Kann also das Rabin-System gebrochen werden, so muss ein effizienter Algorithmus A existieren, der bei Eingabe (d, n), $d \in \mathsf{QR}_n$, eine Zahl $x = A(d, n)$ mit $x^2 \equiv_n d$ berechnet. Unter Verwendung von A lässt sich ein effizienter probabilistischer Algorithmus RABIN-FACTORIZE angeben, der n faktorisiert:

Algorithmus 160 RABIN-FACTORIZE(n)

```
repeat forever
  rate zufällig a ∈ {1, ..., (n−1)/2}
  if ggT(a, n) > 1 then output ggT(a, n)
  d ← a² mod n
  b ← A(d, n)
  if b ≢_n ±a then output ggT(a + b, n)
```

Satz 161 *Falls* RABIN-FACTORIZE(n) *hält, gibt er einen Primfaktor von $n = pq$, p, q prim, aus. Die Wahrscheinlichkeit, dass hierfür mehr als t Schleifendurchläufe benötigt werden, ist kleiner als 2^{-t}.*

Beweis: Es ist klar, dass $\mathrm{ggT}(a, n)$ im Fall $\mathrm{ggT}(a, n) > 1$ ein Primfaktor von n ist. Lemma 146 zeigt, dass dies im Fall $b \not\equiv_n \pm a$ auch für $\mathrm{ggT}(a + b, n)$ gilt.

Sei nun X die ZV, die die Wahl von a beschreibt, und sei α die Wk, dass der Algorithmus in einem Schleifendurchlauf einen Primfaktor findet. Wir nehmen o.B.d.A. an, dass $A(d, n)$ im Fall $d \in \mathsf{QR}_n$ eine Zahl b in $\mathbb{Z}_n^*/2 := \{1, \ldots, \frac{n-1}{2}\} \cap \mathbb{Z}_n^*$ und sonst $b = 0$ zurückliefert. Für $a \in \mathbb{Z}_n^*/2$ sei a' die in $\mathbb{Z}_n^*/2 - \{a\}$ eindeutig bestimmte Lösung x von $x^2 \equiv_n a^2$. Dann gilt

$$\alpha = \underbrace{Prob[\mathrm{ggT}(X, n) > 1]}_{\beta} + \underbrace{Prob[\mathrm{ggT}(X, n) = 1 \text{ und } X \not\equiv_n \pm A(X^2, n)]}_{\gamma},$$

mit

$$\gamma = \sum_{a \in \mathbb{Z}_n^*/2} Prob[A(X^2, n) = a] \cdot Prob[X = a' \mid A(X^2, n) = a].$$

Wegen

$$Prob[X = a' \mid A(X^2, n) = a] = Prob[X = a \mid A(X^2, n) = a] = 1/2$$

folgt

$$\begin{aligned} \gamma &= \frac{1}{2} \sum_{a \in \mathbb{Z}_n^*/2} Prob[A(X^2, n) = a] \\ &= \frac{1}{2} Prob[\mathrm{ggT}(X, n) = 1] = \frac{1 - \beta}{2} \end{aligned}$$

und somit

$$\alpha = \beta + \frac{1-\beta}{2} = \frac{1+\beta}{2} > 1/2.$$

Folglich ist die Wahrscheinlichkeit, dass mehr als t Schleifendurchläufe ausgeführt werden, gleich $(1-\alpha)^t < 2^{-t}$. ■