Vorlesungsskript

Einführung in die Kryptologie

Sommersemester 2020

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

11. August 2020

Inhaltsverzeichnis

1 Klassische Kryptoverfahren

1.1 Einführung

Kryptografische Verfahren schaffen Vertrauen in ungeschützten Umgebungen. Sie ermöglichen sichere Kommunikation über unsichere Kanäle und können verhindern, dass sich ein Kommunikationspartner unfair verhält. In unsicheren Umgebungen wie dem Internet können sie die aus direkter Interaktion gewohnte Sicherheit herstellen. Und auch die Interaktion in sicheren Umgebungen wird um Möglichkeiten erweitert, die ohne Kryptografie nicht denkbar wären.

In diesem Modul werden wir uns mit den mathematischen Grundlagen von kryptografischen Verfahren beschäftigen, wobei (symmetrische und asymmetrische) Verschlüsselungsverfahren im Vordergrund stehen. Im Mastermodul Kryptologie werden wir dann auch kryptografische Verfahren und Protokolle für andere Schutzziele betrachten wie z.B. Hashverfahren und digitale Signaturen sowie Pseudozufallsgeneratoren.

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.

Steganografische Maßnahmen: unsichtbare Tinte etc.

Andererseits können durch kryptografische 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 Kryptografie fallen auch die folgenden Begriffe.

Kryptografie: Lehre von der Geheimhaltung von Informationen durch Verschlüsselung. 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 kryptografischen Stärken bzw. Schwächen.

Kryptologie: Wissenschaft vom Entwurf, der Anwendung und der Analyse von kryptografischen Verfahren (umfasst Kryptografie 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. *Ein* ***Alphabet*** $A = \{a_0, \ldots, a_{m-1}\}$ *ist eine geordnete endliche Menge von* ***Zeichen*** a_i*. Eine Folge* $x = x_1 \ldots x_n \in A^n$ *heißt* ***Wort*** *(der* ***Länge*** n*). Die Menge aller Wörter über dem Alphabet* A *ist* $A^* = \bigcup_{n \geq 0} A^n$*.*

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

Definition 4. *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),*

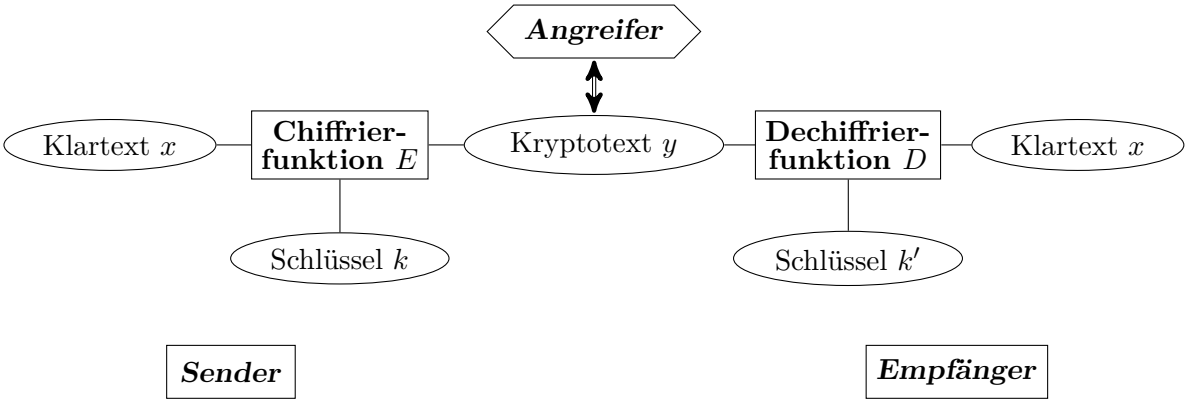


Abbildung 1.1: Schematische Darstellung der Funktionsweise eines Kryptosystems

- $D : K \times C \to M$, *die* ***Entschlüsselungsfunktion*** *(decryption function) und*
- $S \subseteq K \times K$, *eine Menge von Schlüsselpaaren* (k, k') *mit der Eigenschaft, dass für jeden Klartext* $x \in M$ *folgende Beziehung gilt:*

$$D(k', E(k, x)) = x \tag{1.1}$$

Bei symmetrischen Kryptosystemen ist $S = \{(k, k) \mid k \in K\}$, weshalb wir in diesem Fall 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 *injektiv.*

Beweis. Angenommen, für zwei Klartexte x_1 und x_2 gilt $E(k, x_1) = E(k, x_2)$. Dann folgt

$$x_1 \overset{(1.1)}{=} D(k', \underbrace{E(k, x_1)}_{E(k,x_2)}) = D(k', E(k, x_2)) \overset{(1.1)}{=} x_2$$

□

1.3 Die additive Chiffre

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

Definition 6 (**teilt-Relation, modulare Kongruenz**). *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*

$$a \equiv_m b \text{ oder } a \equiv b \pmod{m}$$

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

$$a \bmod m = \min\{a - dm \geq 0 \mid d \in \mathbb{Z}\}$$

Tabelle 1.1: 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*

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 $a = dm + r$. Sowohl r als auch d sind hierbei eindeutig bestimmt (siehe Übungen) und die Zahl d wird auch mit a div m bezeichnet.*

Die auf $\mathbb{Z}$ definierten Operationen

$$a \oplus_m b := (a + b) \bmod m \text{ und } a \odot_m b := ab \bmod m$$

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 $a \ominus_m b$. Wenn aus dem Kontext klar ist, dass $a, b \in \mathbb{Z}_m$ sind, schreiben wir anstelle von $a \oplus_m b$, $a \ominus_m b$ und $a \odot_m b$ auch einfach $a + b$, $a - b$ bzw. ab. Durch Identifikation der Zeichen a_i eines Alphabets $A = \{a_0, \ldots, a_{m-1}\}$ mit ihren Indizes können wir die auf $\mathbb{Z}_m$ definierten Rechenoperationen auf Buchstaben übertragen.

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*

$$\begin{array}{lll} a_i + a_j = a_{i+j}, & a_i - a_j = a_{i-j}, & a_i a_j = \quad a_{ij}, \\ a_i + z \ = a_{i+z}, & a_i - z \ = a_{i-z}, & z a_j \ = a_{z j \bmod m}. \end{array}$$

Mit Hilfe dieser Notation lässt sich die additive Chiffre , die auch als Verschiebechiffre oder Caesar-Chiffre bezeichnet wird, leicht beschreiben.

Definition 8. *Bei der* ***additiven Chiffre*** *ist $A = B = M = C$ ein beliebiges Alphabet mit $m := \|A\|$ und $K = \{0, \ldots, m-1\}$. Für $k \in K$, $x \in M$ und $y \in C$ gilt*

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

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 (siehe Tabelle 1.1). Natürlich kann mit dieser Substitution nicht ernsthaft die Vertraulichkeit von Nachrichten gewahrt werden. Vielmehr soll durch sie ein unbeabsichtigtes Mitlesen – etwa von Rätsellösungen – verhindert werden.

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

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

Da ROT13 zudem keinen Buchstaben auf sich selbst abbildet, ist sie sogar ***echt involutorisch***.

1.4 Die multiplikative Chiffre

Die Buchstabenrechnung legt folgende Modifikation der Caesar-Chiffre nahe. Anstatt auf jedes Klartextzeichen den Schlüsselwert k zu addieren, können wir die Klartextzeichen 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 das Zeichen $g(\mathtt{A}) = g(\mathtt{N}) = \mathtt{a}$ ab. Um eine hinreichende und notwendige Bedingung für die Zulässigkeit eines Schlüsselwerts k formulieren 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*

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

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

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

*das **kleinste gemeinsame Vielfache** von a und b. Ist $\mathrm{ggT}(a, b) = 1$, so nennt man a und b **teilerfremd** oder man sagt, a ist **relativ prim** zu b.*

Lemma 10. *Seien $a, b, c \in \mathbb{Z}$ mit $(a, b) \neq (0, 0)$. Dann gilt $\mathrm{ggT}(a, b) = \mathrm{ggT}(b, a + bc)$ und somit $\mathrm{ggT}(a, b) = \mathrm{ggT}(b, a \bmod b)$, falls $b \geq 1$ ist.*

Beweis. Jeder Teiler d von a und b ist auch ein Teiler von b und $a + bc$ und umgekehrt. □

Euklidischer 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*):

$$r_0 = a > r_1 = b > r_2 > \cdots > r_s > r_{s+1} = 0 \text{ und } d_2, d_3, \ldots d_{s+1}$$

mit

$$r_{i-1} = d_{i+1} r_i + r_{i+1} \text{für } i = 1, \ldots, s.$$

Hierzu sind s Divisionsschritte erforderlich. Wegen

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

folgt $\mathrm{ggT}(a, b) = \mathrm{ggT}(r_s, r_{s+1}) = r_s$.

Beispiel 11. *Für $a = 693$ und $b = 147$ erhalten wir*

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

und damit $\mathrm{ggT}(693, 147) = r_4 = 21$. ◁

*Also: $d_{i+1} = r_{i-1} \operatorname{div} r_i$ und $r_{i+1} = r_{i-1} \bmod r_i$.

Der euklidische Algorithmus lässt sich sowohl iterativ als auch rekursiv implementieren.

Prozedur $\texttt{Euklid}_{\texttt{it}}(a, b)$

```
1 repeat
2     r := a mod b
3     a := b
4     b := r
5 until r = 0
6 return(a)
```

Prozedur $\texttt{Euklid}_{\texttt{rek}}(a, b)$

```
1 if b = 0 then
2     return(a)
3 else
4     return(Euklid_rek(b, a mod b))
```

Zur Abschätzung von s 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 $i = s+1, s, \ldots, 0$ folgt $r_i \geq F_{s+1-i}$ und somit $a = r_0 \geq F_{s+1}$. Weiterhin lässt sich durch Induktion über $n \geq 0$ zeigen, dass $F_{n+1} \geq \phi^{n-1}$ ist, wobei $\phi = (1+\sqrt{5})/2$ der *goldene Schnitt* ist. Der Induktionsanfang ($n = 0$ oder 1) ist klar, da $F_2 = F_1 = 1 = \phi^0 \geq \phi^{-1}$ ist. Unter der Induktionsannahme $F_{i+1} \geq \phi^{i-1}$ für $i \leq n-1$ folgt wegen $\phi^2 = \phi + 1$

$$F_{n+1} = F_n + F_{n-1} \geq \phi^{n-2} + \phi^{n-3} = \phi^{n-3}(\phi + 1) = \phi^{n-1}.$$

Somit ist $a \geq \phi^{s-1}$, d. h. $s \leq 1 + \lfloor \log_\phi a \rfloor$.

Satz 12. *Seien $a > b > 0$ ganze Zahlen und sei n die Länge von a in Binärdarstellung. Dann führt der euklidische Algorithmus $O(n)$ Divisionsschritte zur Berechnung von* $\mathrm{ggT}(a, b)$ *durch. Dies führt auf eine Zeitkomplexität von $O(n^3)$, da jede Ganzzahldivision in Zeit $O(n^2)$ durchführbar ist.*

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

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

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

$$p_i = p_{i-2} - d_i p_{i-1} \text{ (mit } p_0 = 1 \text{ und } p_1 = 0)$$

und

$$q_i = q_{i-2} - d_i q_{i-1} \text{ (mit } q_0 = 0 \text{ und } q_1 = 1$$

für $i = 0, \ldots, s$ bestimmt. Dann gilt für $i = 0$ und $i = 1$,

$$ap_i + bq_i = r_i,$$

und wegen

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

folgt induktiv über $i = 2, \ldots, s$, dass diese Gleichung auch für $i = s$ gilt:

$$ap_s + bq_s = r_s = \mathrm{ggT}(a, b).$$

Korollar 13 (Lemma von Bezout)**.** *Der größte gemeinsame Teiler von a und b ist in der Form*

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

darstellbar.

Beispiel 14. *Für $a = 693$ und $b = 147$ erhalten wir wegen*

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

$p_i \cdot 693 +$	$q_i \cdot 147 =$	r_i
$1 \cdot 693 +$	$0 \cdot 147 =$	693
$0 \cdot 693 +$	$1 \cdot 147 =$	147
$1 \cdot 693 -$	$4 \cdot 147 =$	105
$-1 \cdot 693 +$	$5 \cdot 147 =$	42
$3 \cdot 693 -$	$14 \cdot 147 =$	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 15. *Der größte gemeinsame Teiler von a und b wird von allen gemeinsamen Teilern von a und b geteilt,*

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

Beweis. Seien $\mu, \lambda \in \mathbb{Z}$ mit $\mu a + \lambda b = \mathrm{ggT}(a, b)$. Falls x sowohl a als auch b teilt, dann teilt x auch die Produkte μa und λb und somit auch deren Summe. □

Korollar 16. $\mathrm{ggT}(a, b) = \min\{\lambda a + \mu b \geq 1 \mid \lambda, \mu \in \mathbb{Z}\}$.

Beweis. Sei $M = \{\lambda a + \mu b \geq 1 \mid \lambda, \mu \in \mathbb{Z}\}$, $m = \min M$ und $g = \mathrm{ggT}(a, b)$. Dann folgt $g \geq m$, da g in der Menge M enthalten ist, und $g \leq m$, da g jede Zahl in M teilt. □

Korollar 17. *Zwei Zahlen a und b sind genau dann zu einer Zahl $m \in \mathbb{Z}$ teilerfremd, wenn ihr Produkt ab teilerfremd zu m ist,*

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

Beweis. Da a und b teilerfremd zu m sind, existieren Zahlen $\mu, \lambda, \mu', \lambda' \in \mathbb{Z}$ mit $\mu a + \lambda m = \mu' b + \lambda' m = 1$. Somit ergibt sich aus der Darstellung

$$1 = (\mu a + \lambda m)(\mu' b + \lambda' m) = \underbrace{\mu\mu'}_{\mu''} ab + \underbrace{(\mu a \lambda' + \mu' b \lambda + \lambda \lambda' m)}_{\lambda''} m$$

und Korollar 16, dass auch ab teilerfremd zu m ist.

Gilt umgekehrt $\mathrm{ggT}(ab, m) = 1$, so existieren Zahlen $\mu, \lambda \in \mathbb{Z}$ mit $\mu ab + \lambda m = 1$. Mit Korollar 16 folgt sofort $\mathrm{ggT}(a, m) = \mathrm{ggT}(b, m) = 1$. □

Korollar 18 (Lemma von Euklid)**.** *Sind a und b teilerfremd und teilt a das Produkt bc, so teilt a auch c,*

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

Beweis. Wegen $\mathrm{ggT}(a,b) = 1$ existieren Zahlen $\mu, \lambda \in \mathbb{Z}$ mit $\mu a + \lambda b = 1$. Falls a das Produkt bc teilt, muss a auch die Zahl $\mu ac + \lambda bc = c$ teilen. □

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

Satz 19. *Seien b, y, m ganze Zahlen mit $m \geq 1$. Die lineare Kongruenzgleichung $bx \equiv_m y$ besitzt genau dann eine eindeutige Lösung $x \in \{0, \ldots, m-1\}$, 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 $x \not\equiv_m x'$. Folglich ist die Kongruenz $bx \equiv_m y$ nicht eindeutig lösbar.

Gilt umgekehrt $\mathrm{ggT}(b,m) = 1$, so folgt aus den Kongruenzen

$$bx_1 \equiv_m y$$

und

$$bx_2 \equiv_m y$$

sofort $b(x_1 - x_2) \equiv_m 0$, 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 $x_1 \equiv_m x_2$. Folglich hat die Kongruenz $bx \equiv_m y$ für jedes $y \in \mathbb{Z}_m$ höchstens eine Lösung $x \in \{0, \ldots, m-1\}$. Zudem folgt, dass die Abbildung $f : \mathbb{Z}_m \to \mathbb{Z}_m$ mit $f(x) = bx \bmod m$ injektiv ist. Da aber der Definitions- und der Wertebereich von f die gleiche Mächtigkeit haben, muss f dann auch surjektiv sein. Somit hat die Kongruenz $bx \equiv_m y$ für jedes $y \in \mathbb{Z}_m$ sogar genau eine Lösung $x \in \{0, \ldots, m-1\}$. □

Korollar 20. *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.*

Korollar 17 zeigt, dass die Menge

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

aller invertierbaren Elemente von $\mathbb{Z}_m$ unter der Operation $\odot_m$ abgeschlossen ist. Mit Korollar 20 folgt daher, dass $(\mathbb{Z}_m^*, \odot_m, 1)$ eine multiplikative Gruppe bildet. Allgemeiner zeigt man, dass die Multiplikation eines beliebigen Rings $(R, +, \cdot, 0, 1)$ mit Eins auf der Menge $R^* = \{a \in R \mid \exists b \in R : ab = 1 = ba\}$ aller **Einheiten** von R eine Gruppe bildet (siehe Übungen). Diese Gruppe $(R^*, \cdot, 1)$ wird als **Einheitengruppe** von R bezeichnet.

Das multiplikative Inverse von b modulo m ergibt sich aus der linearen Darstellung $\lambda b + \mu m = \mathrm{ggT}(b,m) = 1$ zu $b^{-1} = \lambda \bmod m$. Die folgende Tabelle gibt für jedes $b \in \mathbb{Z}_{26}^*$ das multiplikative Inverse b^{-1} an.

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

Bei Kenntnis von b^{-1} kann die Kongruenz $bx \equiv_m y$ leicht zu $x = yb^{-1} \bmod m$ gelöst werden.

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

1.5 Die affine Chiffre

Definition 21. *Bei der **affinen Chiffre** ist $A = B = M = C$ ein beliebiges Alphabet mit $m := \|A\|$ und $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$. Für $k = (b, c) \in K$, $x \in M$ und $y \in C$ gilt*

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

In diesem Fall liefert die Schlüsselkomponente $b = -1$ für jeden Wert von $c \in \mathbb{Z}_m$ eine involutorische Chiffrierfunktion $x \mapsto E_{(-1,c)}(x) = c - x$ (**verschobenes komplementäres Alphabet**). Wählen wir für c ebenfalls den Wert -1, so ergibt sich die Chiffrierfunktion $x \mapsto -x - 1$, die auch als **revertiertes Alphabet** bekannt ist. Offenbar ist diese Funktion genau dann echt involutorisch, wenn m gerade 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
$-x$	*a 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*
$-x-1$	*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*

Als nächstes illustrieren wir die Ver- und Entschlüsselung mit der affinen Chiffre an einem kleinen Beispiel.

Beispiel 22 (affine Chiffre)**.** *Sei $A = \{\mathsf{A}, \ldots, \mathsf{Z}\} = B$, also $m = 26$. Weiter sei $k = (9, 2)$, also $b = 9$ und $c = 2$. Um das Klartextzeichen $x = \mathsf{F}$ zu verschlüsseln, berechnen wir*

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

da der Index von F gleich 5, der von v gleich 21 und $9 \cdot 5 + 2 = 47 \equiv_{26} 21$ ist. Um ein Kryptotextzeichen 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
0						$1 \cdot 26 +$	$0 \cdot 9$	$=$	26
1	26	$=$	$2 \cdot 9$	$+$	8	$0 \cdot 26 +$	$1 \cdot 9$	$=$	9
2	9	$=$	$1 \cdot 8$	$+$	1	$1 \cdot 26 +$	$(-2) \cdot 9$	$=$	8
3	8	$=$	$8 \cdot \mathbf{1}$	$+$	0	$(-1) \cdot 26 +$	$\mathbf{3} \cdot 9$	$=$	1

zu $b^{-1} = q_3 = 3$ ergibt. Damit erhalten wir für das Kryptotextzeichen $y = \mathsf{v}$ das ursprüngliche Klartextzeichen

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

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

1.6 Die Eulersche Phi-Funktion

Zur Berechnung der Schlüsselzahl bei der multiplikativen und affinen Chiffre benötigen wir die Funktion

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

die sogenannte *Eulersche φ-Funktion*. Die folgende Tabelle zeigt die Werte $\varphi(m)$ für $m = 1, \ldots, 10$ (für die Menge $\{1, \ldots, n\}$, $n \in \mathbb{N}$, schreiben wir auch kurz $[n]$).

m	1	2	3	4	5	6	7	8	9	10
$\mathbb{Z}_m^*$	$\{0\}$	$\{1\}$	$[2]$	$\{1,3\}$	$[4]$	$\{1,5\}$	$[6]$	$\{1,3,5,7\}$	$\{1,2,4,5,7,8\}$	$\{1,3,7,9\}$
$\varphi(m)$	1	1	2	2	4	2	6	4	6	4

Für primes p gilt offensichtlich $\varphi(p) = p - 1$, da $\mathbb{Z}_p^* = [p-1]$ ist. Wegen

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

folgt zudem

$$\varphi(p^k) = p^k - p^{k-1} = p^{k-1}(p-1) \text{ für } k \geq 1.$$

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

$$f(x) = (x \bmod m, x \bmod l).$$

Beispiel 23. *Sei $m = 5$ und $l = 6$. Dann erhalten wir die Funktion $f : \mathbb{Z}_{30} \to \mathbb{Z}_5 \times \mathbb{Z}_6$ mit*

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

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

x	20	21	22	**23**	24	25	26	27	28	**29**
$f(x)$	$(0,2)$	$(\mathbf{1},3)$	$(\mathbf{2},4)$	$(\mathbf{3},\mathbf{5})$	$(\mathbf{4},0)$	$(0,\mathbf{1})$	$(\mathbf{1},2)$	$(\mathbf{2},3)$	$(\mathbf{3},4)$	$(\mathbf{4},\mathbf{5})$

*Man beachte, dass f eine Bijektion zwischen $\mathbb{Z}_{30}$ und $\mathbb{Z}_5 \times \mathbb{Z}_6$ ist. Zudem fällt auf, dass ein x-Wert genau dann in $\mathbb{Z}_{30}^*$ liegt, wenn der Funktionswert $f(x) = (y, z)$ zu $\mathbb{Z}_5^* \times \mathbb{Z}_6^*$ gehört (die Werte $x \in \mathbb{Z}_{30}^*$, $y \in \mathbb{Z}_5^*$ und $z \in \mathbb{Z}_6^*$ sind **fett** gedruckt). Folglich bildet f die Argumente in $\mathbb{Z}_{30}^*$ bijektiv auf die Werte in $\mathbb{Z}_5^* \times \mathbb{Z}_6^*$ ab. Für f^{-1} erhalten wir somit folgende Tabelle:*

f^{-1}	0	**1**	2	3	4	**5**
0	0	25	20	15	10	5
1	6	**1**	26	21	16	**11**
2	12	**7**	2	27	22	**17**
3	18	**13**	8	3	28	**23**
4	24	**19**	14	9	4	**29**

Die fett gedruckten Einträge bilden dann die Tabelle der Einschränkung $\hat{f}^{-1}$ von f^{-1} auf die Menge $\mathbb{Z}_5^* \times \mathbb{Z}_6^*$. Das Bild dieser Einschränkung ist genau die Menge $\mathbb{Z}_{30}^*$. ◁

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

$$\begin{aligned}\mathrm{ggT}(x,m\ell)=1 &\Leftrightarrow \mathrm{ggT}(x,m)=\mathrm{ggT}(x,\ell)=1\\ &\Leftrightarrow \mathrm{ggT}(x \bmod m,m)=\mathrm{ggT}(x \bmod \ell,\ell)=1\end{aligned}$$

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

$$\varphi(m\ell)=\|\mathbb{Z}_{m\ell}^*\|=\|\mathbb{Z}_m^*\times\mathbb{Z}_\ell^*\|=\|\mathbb{Z}_m^*\|\cdot\|\mathbb{Z}_\ell^*\|=\varphi(m)\varphi(\ell).$$

Satz 24. *Die Eulersche φ-Funktion ist multiplikativ, d. h. für teilerfremde Zahlen m und ℓ gilt $\varphi(m\ell)=\varphi(m)\varphi(\ell)$.*

Korollar 25. *Sei $m=\prod_{i=1}^{\ell}p_i^{k_i}$ die Primfaktorzerlegung von m. Dann gilt*

$$\varphi(m)=\prod_{i=1}^{\ell}p_i^{k_i-1}(p_i-1)=m\prod_{i=1}^{\ell}(p_i-1)/p_i.$$

Beweis. Es gilt $\varphi(\prod_{i=1}^{\ell}p_i^{k_i})=\prod_{i=1}^{\ell}\varphi(p_i^{k_i})=\prod_{i=1}^{\ell}(p_i^{k_i}-p_i^{k_i-1})=\prod_{i=1}^{\ell}p_i^{k_i-1}(p_i-1)$. □

1.7 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 26 (Chinesischer Restsatz (CRS))**.** *Falls $m_1,\ldots,m_k$ paarweise teilerfremd sind, dann hat das System*

$$\begin{aligned}x &\equiv_{m_1} b_1\\ &\vdots\\ x &\equiv_{m_k} b_k\end{aligned}\tag{1.2}$$

für beliebige Zahlen $b_1,\ldots,b_k\in\mathbb{Z}$ genau eine Lösung modulo $m=\prod_{i=1}^{k}m_i$.

Beweis. Zu jeder Zahl $n_i={}^{m}\!/_{m_i}$ existieren wegen $\mathrm{ggT}(n_i,m_i)=1$ Zahlen μ_i und λ_i mit

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

Für $i=1,\ldots,k$ löst daher die Zahl $s_i=\mu_i n_i$ das System

$$x\equiv_{m_j}\begin{cases}0, & j\neq i \quad (a)\\ 1, & j=i \quad (b)\end{cases}\tag{1.3}$$

Folglich gelten für $s = \sum_{i=1}^{k} b_i s_i$ die Kongruenzen $s \overset{(1.3a)}{\equiv}_{m_j} b_j s_j \overset{(1.3b)}{\equiv}_{m_j} b_j$, d.h. s löst das System (1.2). Dies zeigt, dass die Funktion

$$f : \mathbb{Z}_m \to \mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_k} \text{ mit } f(x) = (x \bmod m_1, \ldots, x \bmod m_k)$$

surjektiv ist. Da der Definitions- und der Wertebereich von f gleich groß sind, muss f auch injektiv sein und (1.2) ist eindeutig lösbar. □

Man beachte, dass der Beweis des chinesischen Restsatzes konstruktiv ist und die Lösung x unter Verwendung des erweiterten euklidischen Algorithmus' effizient berechnet werden kann.

Man verifiziert auch leicht, dass f ein Isomorphismus zwischen dem Ring $(\mathbb{Z}_m, \oplus_m, \odot_m)$ und dem direkten Produkt der Ringe $(\mathbb{Z}_{m_i}, \oplus_{m_i}, \odot_{m_i})$, $1 \leq i \leq k$, ist. Dies ist nicht nur für theoretische Überlegungen nützlich, sondern hat auch praktische Konsequenzen. Beispielsweise lässt sich dadurch die Laufzeit von bestimmten Berechnungen im Ring $\mathbb{Z}_m$ deutlich reduzieren, sofern die Primzahlzerlegung von m bekannt ist.

1.8 Die Hill-Chiffre

Die von Hill im Jahr 1929 publizierte Chiffre ist eine Erweiterung der multiplikativen Chiffre auf Buchstabenblöcke. Der Klartext wird also nicht zeichen- sondern blockweise verarbeitet. Die Blöcke haben eine feste Länge l und sowohl Klar- als auch Kryptotextraum bestehen aus allen Wörtern $x \in A^l$. Als Schlüssel dient eine $(l \times l)$-Matrix $k = (k_{ij})$ mit Koeffizienten in $\mathbb{Z}_m$. Diese transformiert einen Klartext $x = x_1 \ldots x_l \in A^l$ in den Kryptotext $y = y_1 \ldots y_l$ mit $y_i = x_1 k_{1i} + \cdots + x_l k_{li}$ für $i = 1, \ldots, l$:

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

Wir bezeichnen die Menge aller $(l \times l)$-Matrizen (k_{ij}) mit Koeffizienten $k_{ij} \in \mathbb{Z}_m$ mit $\mathbb{Z}_m^{l \times l}$. Als Schlüssel können nur invertierbare Matrizen k benutzt werden, da sonst der Chiffriervorgang nicht injektiv ist. Ob eine Matrix $k \in \mathbb{Z}_m^{l \times l}$ invertierbar ist, lässt sich an ihrer Determinante erkennen.

Definition 27 (Determinante). *Sei R ein kommutativer Ring mit Eins und sei $A = (a_{ij}) \in R^{n \times n}$. Eine Funktion $f : R^{n \times n} \to R$ heißt **Determinantenfunktion**, falls sie folgende drei Eigenschaften erfüllt*

- *f ist **multilinear**, d.h. für jede Matrix $A = (a_1, \ldots, a_n) \in R^{n \times n}$ mit Spalten $a_1, \ldots, a_n \in (R^n)^T$, jeden Spaltenvektor $b \in (R^n)^T$ und jedes $r \in R$ gilt*

$$f(a_1, \ldots, r a_i + b, \ldots, a_n) = r f(a_1, \ldots, a_i, \ldots, a_n) + f(a_1, \ldots, b, \ldots, a_n).$$

- *f ist **alternierend**, d.h. im Fall $a_i = a_j$ für $i \neq j$ gilt $f(a_1, \ldots, a_n) = 0$.*
- *f ist **normiert**, d.h. $f(E) = 1$, wobei E die Einheitsmatrix ist.*

Tatsächlich ist f durch diese drei Eigenschaften eindeutig festgelegt und wir bezeichnen $f(A)$ wie üblich mit $\det(A)$.

Eine explizite Darstellung für die Determinantenfunktion liefert der laplacesche Entwicklungssatz. Für $1 \leq i, j \leq n$ sei A_{ij} die durch Streichen der i-ten Zeile und j-ten Spalte aus A hervorgehende Matrix. Dann ist $det(A) = a_{11}$, falls $n = 1$, und für $n > 1$ ist

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

wobei $i \in \{1, \cdots, n\}$ beliebig wählbar ist (Entwicklung nach der i-ten Zeile). Das Produkt $(-1)^{i+j} det(A_{ij})$ wird **Kofaktor** genannt und mit $\tilde{a}_{ij}$ bezeichnet. Aus dieser Formel lässt sich zwar ein Algorithmus zur Berechnung der Determinante ableiten, allerdings hat dieser eine exponentielle Laufzeit. Das Gauß-Verfahren führt dagegen auf eine effiziente Berechnungsmethode für die Determinante (siehe Übungen).

Für die Dechiffrierung eines mit dem Schlüssel k berechneten Kryptotextes wird die inverse Matrix k^{-1} benötigt. Invertierbare Matrizen werden auch als **regulär** bezeichnet. Eine Matrix $k \in \mathbb{Z}_m^{l \times l}$ ist genau dann regulär, wenn $\mathrm{ggT}(det(k), m) = 1$ ist. In diesem Fall lässt sich k^{-1} mit dem Gauß-Jordan-Algorithmus effizient berechnen (siehe Übungen).

Definition 28. *Sei $A = \{a_0, \ldots, a_{m-1}\}$ ein beliebiges Alphabet und für eine natürliche Zahl $\ell \geq 2$ sei $M = C = A^\ell$. Bei der **Hill-Chiffre** ist $K = \{k \in \mathbb{Z}_m^{\ell \times \ell} \mid \mathrm{ggT}(det(k), m) = 1\}$ und es gilt*

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

Beispiel 29 (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*

$$(\mathtt{HILL}) \begin{pmatrix} 11 & 13 & 8 & 21 \\ 24 & 17 & 3 & 25 \\ 18 & 12 & 23 & 17 \\ 6 & 15 & 2 & 15 \end{pmatrix} = (\mathtt{nerx}) \quad \textit{bzw.} \quad \begin{aligned} 11\,\mathtt{H} + 24\,\mathtt{I} + 18\,\mathtt{L} + 6\,\mathtt{L} &= \mathtt{n} \\ 13\,\mathtt{H} + 17\,\mathtt{I} + 12\,\mathtt{L} + 15\,\mathtt{L} &= \mathtt{e} \\ 8\,\mathtt{H} + 3\,\mathtt{I} + 23\,\mathtt{L} + 2\,\mathtt{L} &= \mathtt{r} \\ 21\,\mathtt{H} + 25\,\mathtt{I} + 17\,\mathtt{L} + 15\,\mathtt{L} &= \mathtt{x} \end{aligned}$$

den Kryptotext $E(k, \mathtt{HILL}) = \mathtt{nerx}$. Für die Entschlüsselung wird die inverse Matrix k^{-1} benötigt. Diese wird in den Übungen berechnet. ◁

1.9 Die Vigenère-Chiffre und andere Stromsysteme

Die nach dem Franzosen Blaise de Vigenère (1523–1596) benannte Chiffre ersetzt den Klartext zeichenweise, allerdings je nach Position im Klartext unterschiedlich.

Definition 30. *Sei $A = B$ ein beliebiges Alphabet. Die **Vigenère-Chiffre** chiffriert unter einem Schlüssel $k = k_0 \ldots k_{d-1} \in K = A^*$ einen Klartext $x = x_0 \ldots x_{n-1}$ beliebiger Länge zu*

$$E(k, x) = y_0 \ldots y_{n-1} \quad \textit{mit} \quad y_i = x_i + k_{(i \bmod d)} \textit{ für } i = 1, \ldots, n-1$$

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

$$D(k, y) = x_0 \ldots x_{n-1} \quad \textit{mit} \quad x_i = y_i - k_{(i \bmod d)} \textit{ für } i = 1, \ldots, n-1.$$

Beispiel 31 (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*

$$E(\texttt{WIE}, \texttt{VIGENERE}) = \underbrace{\texttt{V}+\texttt{W}}_{r}\ \underbrace{\texttt{I}+\texttt{I}}_{q}\ \underbrace{\texttt{G}+\texttt{E}}_{k}\ \underbrace{\texttt{E}+\texttt{W}}_{a}\ \underbrace{\texttt{N}+\texttt{I}}_{v}\ \underbrace{\texttt{E}+\texttt{E}}_{i}\ \underbrace{\texttt{R}+\texttt{W}}_{n}\ \underbrace{\texttt{E}+\texttt{I}}_{m} = \texttt{rqkavinm}$$

◁

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 32. *Vigenère-Chiffre*

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

◁

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 33 (**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:			*Dechiffrierung:*		
	WIEWIEWI	(*Schlüsselstrom*)		WIEWIEWI	(*Schlüsselstrom*)
−	BEAUFORT	(*Klartext*)	−	veecdqfp	(*Kryptotext*)
	veecdqfp	(*Kryptotext*)		BEAUFORT	(*Klartext*)

◁

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***).[†]

Beispiel 34 (*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:			*Kryptotext-Schlüsselstrom:*		
	VIGENERE	(*Klartext*)		VIGENERE	(*Klartext*)
+	WIEVIGEN	(*Schlüsselstrom*)	+	WIERQKVD	(*Schlüsselstrom*)
	rqkzvkvr	(*Kryptotext*)		rqkvdomh	(*Kryptotext*)

◁

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.

1.10 Der One-Time-Pad

Eine weitere Möglichkeit ist, eine Textstelle in einem Buch als Schlüssel zu vereinbaren und den dort beginnenden Text als aperiodischen Schlüsselstrom zu benutzen (Lauftextverschlüsselung). Besser ist es jedoch, mithilfe von Pseudozufallsgeneratoren aus einem relativ kurzen Schlüssel einen deutlich längeren Schlüsselstrom zu erzeugen. Noch besser ist es, den Schlüsselstrom wirklich zufällig zu erzeugen. Dies führt auf eine absolut sichere Verschlüsselung, sofern der Schlüsselstrom nicht mehrmals benutzt wird.[‡] Ein solcher „Wegwerfschlüssel“ (engl. *One-Time-Pad* oder kurz *OTP*; im Deutschen auch

[†]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.

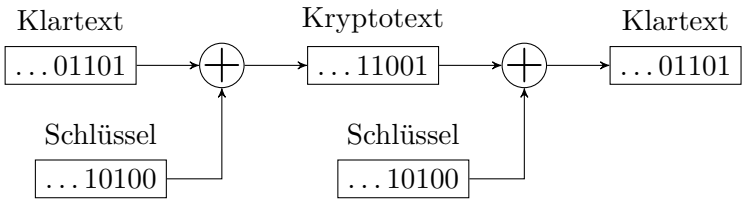
[‡]Diese Methode schlug der amerikanische Major Joseph O. Mauborgne im Jahr 1918 vor, nachdem ihm ein von Gilbert S. Vernam für den Fernschreibverkehr entwickeltes Chiffriersystem vorgestellt wurde.

als **individueller Schlüssel** bezeichnet) lässt sich für längere Klartexte allerdings nur mit großem Aufwand generieren und auf einem sicheren Kanal zwischen Sender und Empfänger verteilen, weshalb diese Chiffre nur wenig praktikabel ist.[§]

Beispiel 35 (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 jedes Klartextzeichen x_i ein neuer, zufällig generierter Schlüsselbuchstabe k_i addiert,*

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

Der Klartext wird also wie bei einer additiven Chiffre verschlüsselt, nur dass der Schlüssel nach einmaligem Gebrauch gewechselt wird. Wie diese ist der One-Time-Pad im Binärfall involutorisch.



1.11 Die Skytale-Chiffre

Bei den bisher betrachteten Chiffrierfunktionen handelt es sich um **Substitutionen**, d.h. sie bilden 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 36 (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.*

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

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

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

[§]Diese Methode wurde beispielsweise beim „heißen Draht“, der 1963 eingerichteten, direkten Fernschreibverbindung zwischen dem Weißen Haus in Washington und dem Kreml in Moskau, angewandt.

über. Dasselbe Resultat erhält man, wenn x zeilenweise in eine $k \times m$-Matrix geschrieben und spaltenweise wieder ausgelesen wird (sogenannte **Spaltentransposition**):

x_1	x_2	$\cdots$	x_m
x_{m+1}	x_{m+2}	$\cdots$	x_{2m}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$x_{(k-1)m+1}$	$x_{(k-1)m+2}$	$\cdots$	x_{km}

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.

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

Beispiel 37. *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 $\rightsquigarrow$ *zzleikakiiccn*

◁

1.12 Die Blocktransposition

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 38. *Sei $A = B$ ein beliebiges Alphabet und für eine natürliche Zahl $l \geq 2$ sei $M = C = A^\ell$. Bei einer* ***Blocktranspositionschiffre*** *wird durch jeden Schlüssel $k \in K$ eine Permutation π auf $[\ell]$ beschrieben, so dass für alle Zeichenfolgen $x_1 \cdots x_\ell \in M$ und $y_1 \cdots y_\ell \in C$*

$$E(k, x_1 \cdots x_\ell) = x_{\pi(1)} \cdots x_{\pi(\ell)}$$

und

$$D(k, y_1 \cdots y_\ell) = y_{\pi^{-1}(1)} \cdots y_{\pi^{-1}(\ell)}$$

gilt.

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

Beispiel 39. *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. Wird π durch eine Folge von 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_n auf i_1 abgebildet wird, so ist π^{-1} sehr leicht zu bestimmen.

Beispiel 40.

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

Obiges π hat beispielsweise die Zyklendarstellung

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

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

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

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

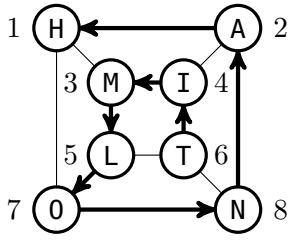
Beispiel 41. *Bei der* ***Matrix-Transposition*** *wird der Klartext zeilenweise in eine $k \times l$-Matrix eingelesen und der Kryptotext spaltenweise gemäß einer Spaltenpermutation $\pi \in S_l$, die als Schlüssel dient, wieder ausgelesen. Für $\pi = (1\ 4\ 3)\ (2\ 6)$ wird also zuerst Spalte $\pi(1) = 4$, 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
$\rightsquigarrow$ *srsta reneg dkxih eaihl ettsn iltcr*

◁

Beispiel 42. *Bei der* ***Weg-Transposition*** *wird als Schlüssel eine Hamiltonlinie in einem Graphen mit den Knoten $1, \ldots, l$ benutzt. (Eine Hamiltonlinie ist eine Anordnung aller Knoten, in der je zwei aufeinanderfolgende Knoten durch eine Kante verbunden sind.) Der Klartextblock $x_1 \cdots x_l$ wird gemäß der Knotennumerierung in den Graphen eingelesen und der zugehörige Kryptotext entlang der Hamiltonlinie wieder ausgelesen.*



HAMILTON $\rightsquigarrow$ *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, sich eine Permutationen in Form eines **Schlüsselworts** (oder einer aus mehreren Wörtern bestehenden **Schlüsselphrase**) ins Gedächtnis einzuprägen. 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 jedes einzelne Zeichen 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 Zeichen gemäß ihrer Position im Schlüsselwort an die Reihe genommen. Alternativ kann man auch alle im Schlüsselwort wiederholt vorkommenden Zeichen streichen, was im Fall des Schlüsselworts *CAESAR* auf eine Blocklänge von 5 führen würde.

Wir wenden uns nun der Klassifikation von Substitutionen zu. Ein wichtiges Unterscheidungsmerkmal ist z.B. die Länge der Klartexteinheiten, auf denen die Chiffre operiert.

Monografische Substitutionen ersetzen Einzelbuchstaben.

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

Eine Substitution heißt **monopartit**, falls sie die Klartextsegmente durch Einzelzeichen ersetzt, sonst **multipartit**. Eine polygrafische Substitution, die auf Zeichenpaaren operiert, wird **digrafisch** genannt. Wird der Kryptotext aus Zeichenpaaren zusammengesetzt, so spricht man von einer **bipartiten** Substitution.

1.13 Die Porta-Chiffre

Das älteste bekannte polygrafische Chiffrierverfahren wurde von Giovanni Porta im Jahr 1563 veröffentlicht. Dabei werden je zwei aufeinanderfolgende Klartextzeichen durch ein einzelnes Kryptotextzeichen ersetzt.

Beispiel 43. *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 *Klartextzeichen* $\mathsf{A}, \ldots, \mathsf{I}, \mathsf{L}, \ldots, \mathsf{T}, \mathsf{V}, \mathsf{Z}$ *indiziert sind. Zur Ersetzung des Zeichenpaars* $a_i a_j$ *wird das in Zeile* i *und Spalte* j *befindliche Kryptotextzeichen*

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

benutzt. ◁

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

	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

Die **Polybios-Chiffre** benutzt als Schlüssel eine 5×5-Matrix, die aus sämtlichen Klartextzeichen gebildet wird.[¶] Die Verschlüsselung des Klartextes erfolgt zeichenweise, indem man einen in Zeile i und Spalte j eingetragenen Klartextzeichen durch das Koordinatenpaar ij ersetzt. Der Kryptotextraum besteht also aus den Ziffernpaaren $\{00, 01, \ldots, 44\}$.

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 jedes einzelne Klartextsegment unabhängig von seiner Position im Klartext auf dieselbe Weise.

Polyalphabetische Substitutionen verwenden eine Ersetzungsregel, die in Abhängigkeit von den bereits verarbeiteten Klartextsegmenten varieren kann.

Die Bezeichnung „monoalphabetisch“ bringt zum Ausdruck, dass der Ersetzungsmechanismus im monografischen Fall für jeden Schlüssel auf einem festen Alphabet beruht. Die von Caesar benutzte Chiffriermethode mit dem Schlüssel $k = 3$ kann beispielsweise vollständig durch Angabe des Ersetzungsalphabets $\{d, e, f, g, w, \ldots, y, z, a, b, c\}$ beschrieben werden.

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.

1.14 Block- und Stromchiffren

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

Definition 44. *Sei A ein beliebiges Alphabet und es gelte $M = C = A^\ell$, $\ell \geq 1$. Eine **Blockchiffre** realisiert für jeden Schlüssel $k \in K$ eine bijektive Abbildung g auf A^ℓ und es gilt für alle $x \in M$ und $y \in C$,*

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

*Im Fall $\ell = 1$ spricht man auch von einer **einfachen Substitutionschiffre**.*

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

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

nennt die einzelnen Klartextblöcke in diesem Zusammenhang auch nicht ‚Blöcke' sondern ‚Zeichen' und spricht von **sequentiellen Chiffren** oder von **Stromchiffren**.

Definition 45. *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 $x = x_0 \dots x_{n-1}$, $x_i \in M$, eine Folge $\hat{k}_0, \dots, \hat{k}_{n-1}$ von internen Schlüsseln $\hat{k}_i = g(k, x_0 \dots x_{i-1}) \in \hat{K}$, unter denen x in den Kryptotext*

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

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 $x_0 \dots x_{i-1}$ den aktuellen internen Schlüssel $\hat{k}_i$ zu berechnen. Die bisher betrachteten Stromchiffren benutzen z.B. die folgenden Schlüsselstromgeneratoren.

Stromchiffre	Chiffrierfunktionen	Schlüsselstromgenerator
Vigenère	$E(\hat{k}, x) = x + \hat{k}$	$g(k_0 \dots k_{d-1}, x_0 \dots x_{i-1}) = k_{(i \bmod m)}$
Beaufort	$E(\hat{k}, x) = \hat{k} - x$	$g(k_0 \dots k_{d-1}, x_0 \dots x_{i-1}) = k_{(i \bmod m)}$
Autokey mit Klartext-Schlüsselstrom	$E(\hat{k}, x) = x + \hat{k}$	$g(k_0 \dots k_{d-1}, x_0 \dots x_{i-1}) = \begin{cases} k_i, & i < d \\ x_{i-d}, & i \geq d \end{cases}$
Autokey mit Kryptotext-Schlüsselstrom	$E(\hat{k}, x) = x + \hat{k}$	$g(k_0 \dots k_{d-1}, x_0 \dots x_{i-1}) = \begin{cases} k_i, & i < d \\ y_{i-d}, & i \geq d \end{cases}$ $= k_{(i \bmod d)} + \sum_{j=1}^{\lfloor i/d \rfloor} x_{i-jd}$

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).

1.15 Gespreizte und homophone 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 46. *Bei der **Spionage-Chiffre** wird in die erste Zeile einer 3×10-Matrix ein Schlüsselwort w geschrieben, welches kein Zeichen 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 Klartextzeichen (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
↝ *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 Klartextzeichen 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 Schlüsselwortzeichen 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 Klartextzeichen 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 47. *Sei A ein Klartextalphabet und sei $M = A$. Weiter sei C ein Kryptotextraum der Größe $\|C\| > \|A\| = m$. In einer* ***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 Klartextzeichen 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 48 (homophone Substitution)**.** *Sei* $A = \{\mathtt{A}, \ldots, \mathtt{Z}\}$, $B = \{\mathtt{0}, \ldots, \mathtt{9}\}$ *und* $C = \{\mathtt{00}, \ldots, \mathtt{99}\}$.

	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 Klartextzeichen 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 49 (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.

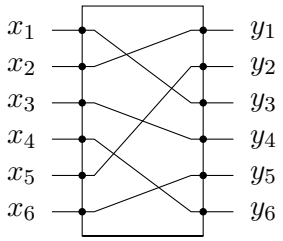
	1	2	3	4	5	6	7	8	9	0
1	N	E	C	S	A	O	D	X	I	N
2	R	G	S	N	N	A	U	C	H	Y
3	T	L	I	O	U	D	Z	M	N	E
4	H	R	E	A	N	E	E	S	I	T
5	N	I	E	T	P	H	S	L	A	R
6	E	U	M	F	R	J	E	N	E	D
7	N	E	K	S	C	T	I	T	A	A
8	H	N	I	B	R	E	U	G	V	E
9	T	E	L	S	D	R	E	O	S	E
0	B	D	W	E	Q	I	F	E	I	R

HOMOPHON $\rightsquigarrow$ 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.

1.16 Realisierung von Transpositionen und 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

$$y1 := x2;\quad y2 := x5;\quad \ldots\quad y6 := x4;$$

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 1.2 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 $a_0 = \mathsf{A}$ auf der äußeren Scheibe zur Deckung kommt. Auf der Drehscheibe in Abbildung 1.2 ist beispielsweise der Schlüssel $k = 2$ eingestellt, das heißt, $a_k = \mathsf{C}$. 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. Hierfür können

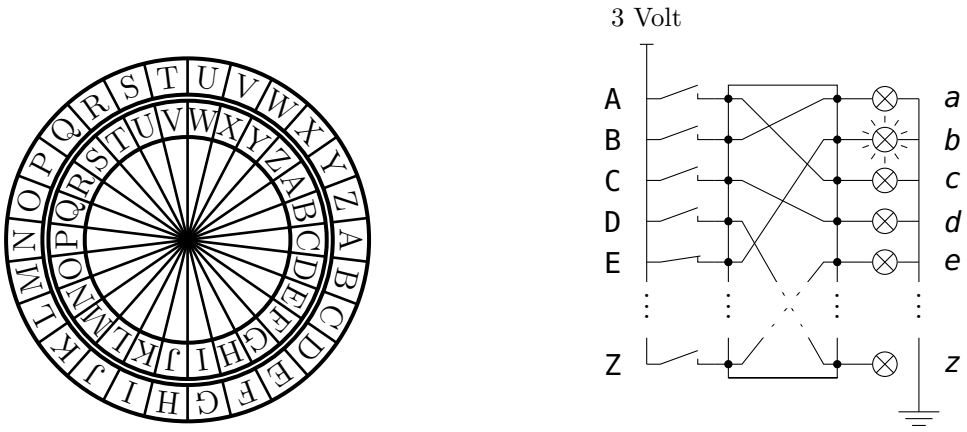


Abbildung 1.2: Realisierung von einfachen Substitutionen mit einer Drehscheibe und mit Hilfe von Steckverbindungen

beispielsweise zwei Steckkontaktleisten verwendet werden. Der aktuelle Schlüssel wird in diesem Fall durch Verbinden der entsprechenden Kontakte mit elektrischen Kabeln eingestellt (siehe Abbildung 1.2). Um etwa das Klartextzeichen 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 Feldelement 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 muss 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*

2 Analyse der klassischen Verfahren

2.1 Klassifikation von Angriffen gegen Kryptosysteme

Die Erfolgsaussichten eines Angriffs gegen ein Kryptosystem hängen sehr stark von der Ausgangslage des Angreifers ab. 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 nicht von irgendwelchen obskuren Annahmen über den Wissensstand des Gegners abhängig gemacht werden darf.

Goldene Regel für Kryptosystem-Designer (Kerckhoffs' Prinzip)
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 von den unterschiedlichen Bedrohungen, denen 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 (zumindest vorübergehend), sich zu Klartexten seiner Wahl die zugehörigen Kryptotexte zu besorgen. Kann hierbei die Wahl der Klartexte 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.

[*] Tatsächlich sind die Prinzipien fast aller heute im Einsatz befindlichen Kryptosysteme bekannt. Nur so kann einer Vielzahl von Kryptoanalytikern die Suche nach Schwachstellen ermöglicht werden.

Angriff bei frei (oder adaptiv) wählbarem Text (*chosen-text attack*)

Sowohl Klartexte als auch Kryptotexte sind frei (oder sogar adaptiv) wählbar.

Ohne Frage ist ein Kryptosystem, das bereits bei einem Angriff mit bekanntem Kryptotext Schwächen erkennen lässt, für die meisten Anwendungen 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 können. So ermöglicht beispielsweise das Auftreten stereotyper Klartext-Formulierungen einen Angriff bei bekanntem Klartext, sofern der Gegner diese Formulierungen kennt bzw. errät. Begünstigt durch derartige Unvorsichtigkeiten, die im praktischen Einsatz meist nicht vermeidbar sind, können sich selbst winzige Konstruktionsschwächen eines Kryptosystems sehr schnell zu einer ernsthaften Bedrohung auswachsen. Die Geschichte der Kryptografie belegt sehr eindrucksvoll, dass es häufig die Anwender eines Kryptosystems selbst sind, die – im unerschütterlichen Glauben an seine kryptografische Stärke – einen erfolgreichen Angriff ermöglichen.

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 bieten, wenn es dem Gegner etwa gelingt, in den Besitz des geheimen Schlüssels zu kommen – sei es aus Unachtsamkeit der Anwender oder infolge von Manipulationsversuchen von Seiten des Gegners (**Social Engineering bzw. Social Hacking**). Auch **Implementierungsangriffe** nutzen nicht Schwachstellen des Kryptoverfahrens aus. Vielmehr zielen sie darauf ab, durch physikalische Messungen wie bspw. des Stromverbrauchs oder der Laufzeit von Berechnungen (sog. **Seitenkanalangriffe**) Informationen über den unbekannten Schlüssel zu gewinnen.

2.2 Kryptoanalyse von einfachen Substitutionschiffren

Durch eine Häufigkeitsanalyse können insbesondere einfache Substitutionen g leicht gebrochen werden. Der Grund dafür ist, dass die einzelnen Zeichen a in der Klartextsprache meist mit unterschiedlichen Wahrscheinlichkeiten $p(a)$ auftreten (vergleiche Tabelle 2.1). Berechnet man die relativen Häufigkeiten h der Zeichen im Kryptotext, so gilt $p(a) \approx h(g(a))$ (vorausgesetzt der Klartext ist genügend lang). Für die Schilderung einer nach dieser Methode durchgeführten Kryptoanalyse sei auf die Erzählung „Der Goldkäfer“ von Edgar Allan Poe verwiesen.

Tabelle 2.1: Einteilung von Buchstaben in Cliquen mit vergleichbaren Häufigkeitswerten

	Deutsch	Englisch	Französisch
sehr häufig	E	E	E
häufig	N | IRS | AT	T | AOIN | SRH	AS | TI | RN | U
durchschnittlich	DHU | LGO | CM	LD | CUMF	LO | DMCP
selten	BFWKZ | PV	PGWYB | VK	V | HGFBQ | J
sehr selten	JYXQ	XJQZ	X | ZYK | W

Manche der bisher betrachteten Chiffrierverfahren verwenden einen so kleinen Schlüsselraum, dass ohne großen Aufwand eine **vollständige Schlüsselsuche** (auch **Brute-Force Angriff** genannt) ausgeführt werden kann.

Beispiel 50 (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	A	B	C	D	E	F	G	H	I	J	K	L	M
$D(k,y)$	SAXP	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. Weiß er zum Beispiel, dass er in deutscher Sprache verfasst ist, kann er einen Großteil der Texte x_i auszuschließen. Mit jedem Text x_i, der nicht als Klartext infrage 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.

Auch wenn der Gegner die Klartextsprache nicht kennt, kann eine Häufigkeitsanalyse erfolgreich sein. 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. Selbst zwischen unterschiedlichen Sprachen gibt es oft Gemeinsamkeiten. So kommt in fast allen europäischen Sprachen der Buchstabe `E` sehr häufig vor, während `X, Y` und `Z` nur selten auftreten. Diese für natürliche Sprachen typische Ungleichmäßigkeit der Buchstabenhäufigkeiten ist darauf zurückzuführen, dass sie relativ viel Redundanz enthalten.

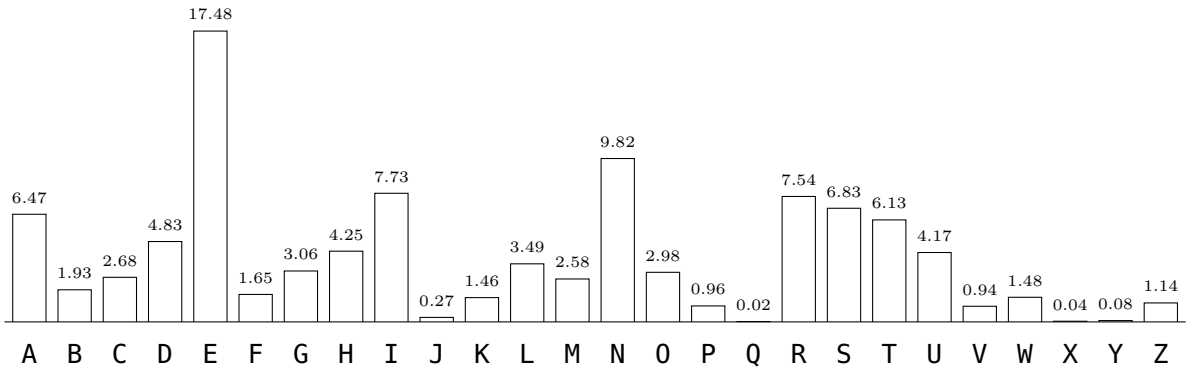


Abbildung 2.1: Häufigkeitsverteilung der Einzelbuchstaben im Deutschen (in %)

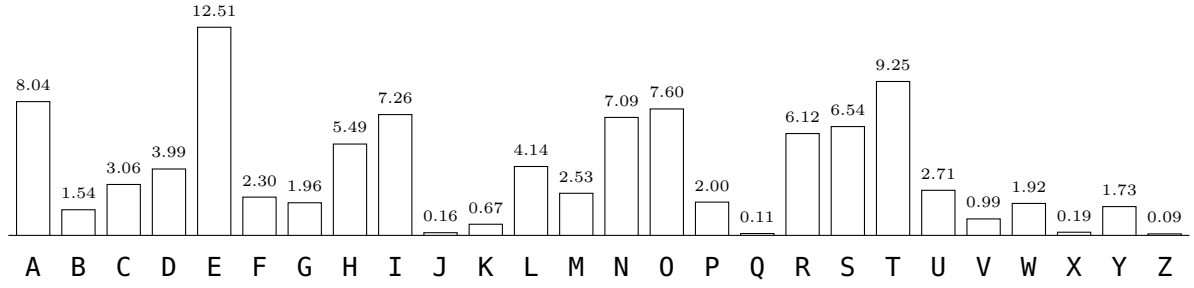


Abbildung 2.2: Häufigkeitsverteilung der Buchstaben im Englischen (in %)

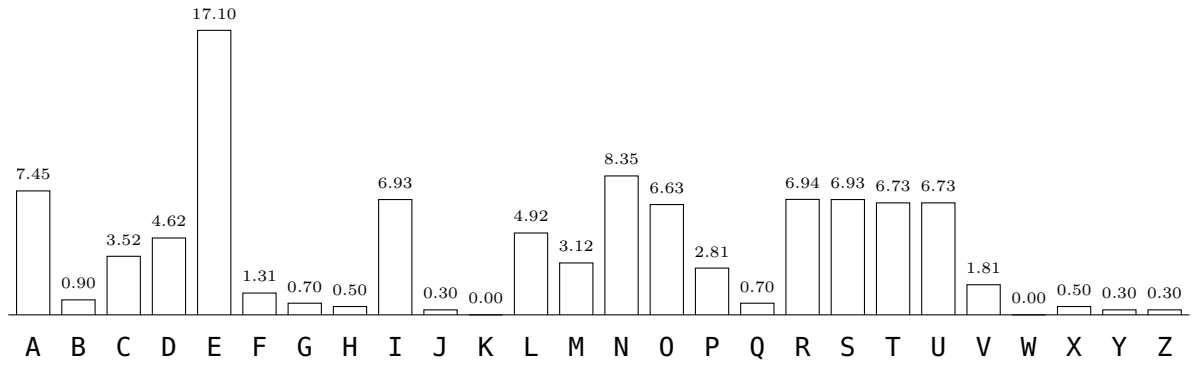


Abbildung 2.3: Häufigkeitsverteilung der Buchstaben im Französischen (in %)

Die Abbildungen 2.1, 2.2 und 2.3, 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. Diese führen auf zwei Gleichungen mit zwei Unbekannten für den gesuchten Schlüssel $k = (b, c)$.

Beispiel 51 (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 affinen Chiffre verschlüsselt wurde. Berechnen wir für jedes Chiffrezeichen y_i die (absolute) Häufigkeit $H_y(y_i)$ seines Auftretens in obigem Kryptotext y,

y_i	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
$H_y(y_i)$	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 `o` *für das Klartextzeichen* `E` *und das am zweithäufigsten vorkommende* `p` *für* `N` *steht. Unter*

dieser Annahme kann der gesuchte Schlüssel $k = (b, c)$ als Lösung der beiden Gleichungen

$$\begin{aligned} b \cdot \mathsf{E} + c &= \mathit{o} \\ b \cdot \mathsf{N} + c &= \mathit{p} \end{aligned}$$

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 (E, o) und (N, p), sondern auch für alle übrigen Paare $(D(k, y_i), y_i)$ eine gute Übereinstimmung zwischen der Häufigkeit $H_y(y_i)$ im Kryptotext y und der erwarteten Häufigkeit $H_{100}(D(k, y_i))$ auf, mit der das Zeichen $D(k, y_i)$ in einem typischen deutschen Text der Länge 100 vorkommt (die Tabelle zeigt die Werte von H_{100} gerundet):

y_i	*o*	*p*	*e*	*h*	*a*	*b*	*c*	*x*	*i*	*l*	*k*	*j*	*u*	*m*	*g*	*v*	*q*	*s*	*t*	*w*	*r*	*f*	*n*	*z*	*y*	*d*
$H_y(y_i)$	19	11	10	8	7	6	5	5	5	4	4	3	2	2	2	2	2	1	1	1	0	0	0	0	0	0
$H_{100}(D(k, y_i))$	17	10	7	6	8	8	6	4	3	5	4	3	3	3	1	1	1	3	0	0	2	2	1	1	0	0
$D(k, y_i)$	E	N	S	T	I	R	A	H	C	D	U	L	G	M	K	P	W	O	X	Y	F	B	V	Z	Q	J

◁

2.3 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 $S = (s_{ij}) = (S_1 \ldots S_\ell)$ mit ℓ Spalten $S_1, \ldots, S_\ell$ 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. Die Nachfolgespalte S_k von S_j (bzw. die Vorgängerspalte 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.

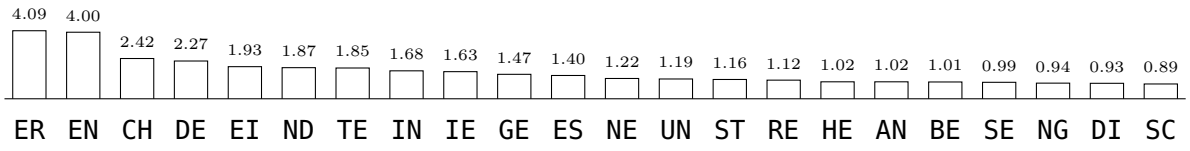


Abbildung 2.4: Die häufigsten Bigramme im Deutschen (Angaben in %)

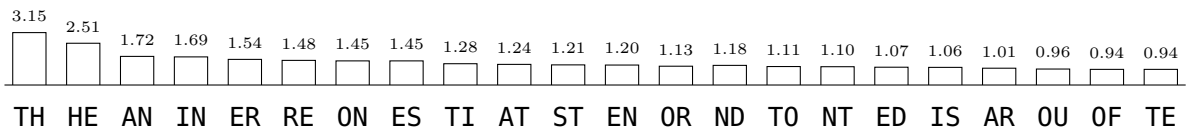


Abbildung 2.5: Die häufigsten Bigramme im Englischen (in %; nach O.P. Meaker, 1939)

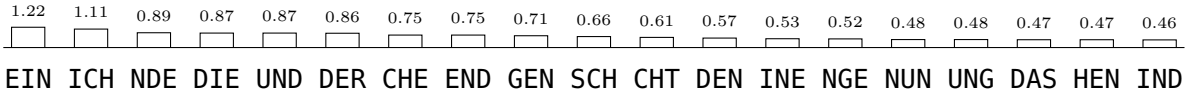


Abbildung 2.6: Die häufigsten Trigramme im Deutschen (in %)

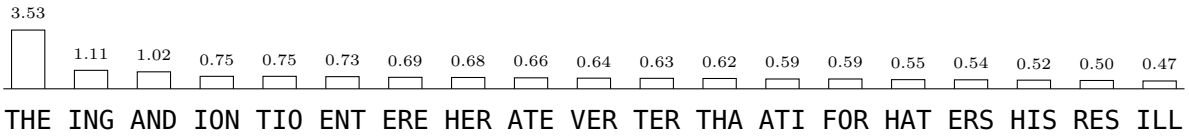


Abbildung 2.7: Die häufigsten Trigramme im Englischen (in %)

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 2.4) 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
1	4	2	2		1	4	2	1

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**
1	2	2			1	1	5

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**	
1	2				2	1	

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

◁

2.4 Kryptoanalyse von polygrafischen Chiffren

Blocksysteme mit kleiner Blocklänge ℓ (beispielsweise bigrafische Systeme) lassen sich ähnlich wie einfache Substitutionen durch Häufigkeitsanalysen brechen. Wird bei Hill-Chiffren l 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 $100\ldots0$, $010\ldots0$, $\ldots$, $0\ldots001 \in A^l$:

$$\begin{aligned} g(100\ldots0) &= k_{11}\,k_{12}\ldots k_{1l} \\ g(010\ldots0) &= k_{21}\,k_{22}\ldots k_{2l} \\ &\vdots \\ g(0\ldots001) &= k_{l1}\,k_{l2}\ldots k_{ll} \end{aligned}$$

und erhält damit die Schlüsselmatrix k.

BK-Angriff (bekannter Klartext). Ist bei einem BK-Angriff eine ausreichende Menge von Klartext-Kryptotextpaaren 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).

2.5 Kryptoanalyse von polyalphabetischen Chiffren

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

Ist die Periode d bekannt, gibt es mehrere Methoden, eine Vigenère-Chiffre zu brechen. So kann man beispielsweise den Kryptotext zeilenweise in eine d-spaltige Matrix schreiben. Verfahrensbedingt wurden dann die einzelnen Spalten $y_1, \ldots, y_d$ durch eine monoalphabetische Substitution (genauer: durch eine additive Chiffre) verschlüsselt. Sie können daher einzeln durch eine Häufigkeitsanalyse gebrochen werden (siehe Abschnitt 2.2). Hierbei liefert jede Spalte y_i den Buchstaben k_i des Schlüsselwortes. Wir werden auf die Bestimmung des Schlüsselwortes bei bekannter Periode im übernächsten Unterabschnitt noch näher eingehen.

Bestimmung der Schlüsselwortlänge

Zur Bestimmung der Schlüsselwortlänge 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 53 (Kasiski-Test)**.**

```
  DERERSTEUNDLETZTEVERS...   (Klartext x)
+ KASKASKASKASKASKASKAS...   (Schlüsselstrom k̂)
  nejorkdemxddotrdenork...   (Kryptotext y)
```

Da die Textstücke `ork`*, bzw.* `de` *im Kryptotext in den Entfernungen* $\delta_1 = 15$ *und* $\delta_2 = 9$ *vorkommen, liegt die Vermutung nahe, 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 54. *Der* ***Koinzidenzindex*** *(engl.* ***i****ndex of* ***c****oincidence) eines Textes* y *der Länge* n *über dem Alphabet* $\mathcal{B}$ *ist definiert als*

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

Hierbei ist $H_y(a)$ *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 $H_y(a)$ 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 $IC(y_i)$. Für genügend lange Kryptotexte ist dasjenige d, welches das maximale arithmetische Mittel der Spaltenindizes $IC(y_i)$ liefert mit hoher Wahrscheinlichkeit die gesuchte Periode. Enthält eine Spalte nämlich nur Kryptozeichen, die alle mit demselben Schlüsselbuchstaben k erzeugt wurden, so stimmt der Koinzidenzindex dieser Spalte mit dem Koinzidenzindex des zugehörigen Klartextes überein, nimmt also einen relativ großen Wert an. Wurden dagegen die Kryptozeichen einer Spalte mit unterschiedlichen Schlüsselbuchstaben generiert, so wird hierdurch eine Glättung der Häufigkeitsverteilung bewirkt, weshalb der Spaltenindex kleiner ausfällt.

Ist die Einzelbuchstabenverteilung $p : A \to [0, 1]$ der Klartextsprache bekannt, so kann der Suchraum für den Wert der Periode d erheblich eingeschränkt werden. Hierzu berechnet man den erwarteten Koinzidenzindex

$$\mathrm{E}_{d,n}(IC) = E(IC(Y)),$$

wobei Y ein mittels einer Vigenère-Chiffre mit einem zufälligen Schlüsselwort der Länge d aus einem zufälligen Klartext der Länge n generierter Kryptotext ist. Im Fall $d = 1$ gilt $IC(y) = IC(x)$. Zudem können wir bei längeren Texten von den gegenseitigen Abhängigkeiten der Zeichen im Text absehen und erhalten

$$\mathrm{E}_{1,\infty}(IC) = \sum_{a \in A} p(a)^2.$$

Dieser Wert wird auch als Koinzidenzindex der zugrunde liegenden Sprache bezeichnet.

Definition 55. *Der* ***Koinzidenzindex*** $\boldsymbol{IC_L}$ ***einer Sprache*** *mit Buchstabenverteilung* $p : A \to [0, 1]$ *ist definiert als*

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

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

Definition 56 (**Rauheitsgrad; Measure of Roughness**). *Der* ***Rauheitsgrad*** MR_L *einer Sprache* L *mit Einzelbuchstabenverteilung* p *ist*

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

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

Beispiel 57. *Für die englische Sprache* ($m = 26$) *gilt beispielsweise* $IC_{\text{Englisch}} \approx 0.0687$ *und* $MR_{\text{Englisch}} \approx 0.0302$. ◁

Übersteigt dagegen die Periode d die Klartextlänge n, so ist der Kryptotext bei zufälliger Wahl des Schlüsselswortes ebenfalls rein zufällig, was auf einen erwarteten Koinzidenzindex von

$$\mathrm{E}_{d,n}(IC) = \sum_{a \in A} \|A\|^{-2} = \|A\|^{-1} = 1/m, \quad d \geq n \geq 2$$

führt. Allgemein gilt für hinreichend großes n,

$$\mathrm{E}_{d,n}(IC) = \frac{n-d}{d \cdot (n-1)} \cdot IC_L + \frac{n \cdot (d-1)}{d \cdot (n-1)} \cdot m^{-1}, \quad 1 \leq d \leq n,$$

da von den $\binom{n}{2}$ möglichen Positionspaaren ungefähr $d \cdot \binom{n/d}{2} = n(n-d)/2d$ Paare nur eine Spalte (was einem Anteil von $(n-d)/d(n-1)$ entspricht) und $\binom{d}{2}(n/d)^2 = n^2(d-1)/2d$ Paare zwei unterschiedliche Spalten betreffen (was einem Anteil von $n(d-1)/d(n-1)$ entspricht).

Untenstehende Tabelle gibt den Erwartungswert $\mathrm{E}_{d,n}(IC)$ des Koinzidenzindexes für Kryptotexte der Länge $n = 100$ in Abhängigkeit von der Periodenlänge d einer Vigenère-Chiffre wieder (in Promille; Klartext ist ein zufällig gewählter Text der englischen Sprache mit 100 Buchstaben).

d	1	2	3	4	5	6	8	10	100
$\mathrm{E}_{d,100}(IC)$	69	54	48	46	44	43	42	41	39

Beispiel 58. *Berechnet sich der Koinzidenzindex eines Vigenère-Kryptotextes der Länge* 100 *zu* 0.045*, so liegt die Vermutung nahe, dass das verwendete Schlüsselwort die Länge vier oder fünf hat, falls* y *aus einem Klartext der englischen Sprache erzeugt wurde.* ◁

Der Koinzidenzindex kann auch Hinweise dafür liefern, mit welchem Kryptoverfahren ein vorliegender Kryptotext erzeugt wurde. Bei Transpositionschiffren sowie bei einfachen Substitutionen bleibt nämlich der Koinzidenzindex im Gegensatz zu polyalphabetischen und polygrafischen Verfahren erhalten. Erstere lassen sich von letzteren zudem dadurch unterscheiden, dass bei ihnen sogar die Buchstabenhäufigkeiten unverändert bleiben.

Bestimmung des Schlüsselwortes

Zur Bestimmung des Schlüsselwortes bei bekannter Periode d kann auch wie folgt vorgegangen werden. Man schreibt den Kryptotext y in eine Matrix mit d Spalten und berechnet für jedes Zeichen $a \in A$ die relative Häufigkeit $h_i(a)$ in jeder Spalte 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 59. *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 $\alpha_i(k)$ (in Promille):*

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\alpha_1(k)$	36	31	31	45	38	26	42	**73**	44	26	36	47	30	32	36	29	28	39	48	42	42	39	42	42	35	31
$\alpha_2(k)$	44	41	40	51	41	31	37	43	34	28	36	26	28	43	**68**	45	35	27	42	43	40	35	30	24	31	45
$\alpha_3(k)$	47	41	48	37	49	40	35	30	48	32	25	42	31	26	43	**76**	37	31	39	45	35	34	37	26	30	25
$\alpha_4(k)$	38	40	27	41	**65**	47	28	34	39	33	35	36	30	30	48	44	35	42	47	38	39	34	27	38	36	37

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 (aus der Erzählung „Der Goldkäfer“ von Edgar Allan Poe).*

```
A GOOD GLASS IN THE BISHOPS HOSTEL IN THE DEVILS SEAT
FORTYONE DEGREES AND THIRTEEN MINUTES NORTH EAST 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 60. *Der **gegenseitge Koinzidenzindex** von zwei Texten y und y' mit den Längen n und n' über dem Alphabet $\mathcal{B}$ ist definiert als*

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

$IC(y, y')$ 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 Häufigkeitsverteilungen von y und y' (d. h. H_y und $H_{y'}$) 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 $y_i + \delta$ und y_j (für $1 \leq i < j \leq d$ und $0 \leq \delta \leq 25$) die Wahrscheinlichkeit an, dass man bei zufälliger Wahl einer Position in $y_i + \delta$ und in y_j denselben Buchstaben vorfindet. Da die Einzelzeichenverteilungen von $y_i - k_i$ und von $y_j - k_j$ der der Klartextsprache entsprechen, haben $y_i + \delta$ und y_j für $\delta = k_j - k_i$ eine ähnliche Verteilung. Mit großer Wahrscheinlichkeit nimmt also $IC(y_i + \delta, y_j)$ 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 61. *Betrachten wir den Kryptotext aus vorigem Beispiel, so ergeben sich für $IC(y_i + \delta, y_j)$ die folgenden Werte (in Promille):*

δ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$IC(y_1 + \delta, y_2)$	40	31	25	38	25	21	46	**74**	50	33	31	44	43	34	31	28	24	31	44	45	37	48	64	44	25	31
$IC(y_1 + \delta, y_3)$	26	47	25	21	47	32	18	49	**91**	42	27	51	45	31	29	32	23	29	27	39	45	46	39	58	44	24
$IC(y_1 + \delta, y_4)$	38	40	29	31	35	24	32	58	42	32	44	50	43	39	31	20	34	36	30	40	45	24	42	**78**	47	22
$IC(y_2 + \delta, y_3)$	50	**85**	49	21	28	35	24	34	46	25	24	27	59	50	50	53	51	24	22	26	43	36	35	32	24	34
$IC(y_2 + \delta, y_4)$	46	53	40	37	51	42	29	23	24	32	40	55	38	31	32	45	**67**	49	25	27	29	29	34	37	38	35
$IC(y_3 + \delta, y_4)$	49	36	38	60	36	25	34	19	29	42	41	33	54	27	36	**78**	47	25	29	33	27	28	47	32	27	54

Also ist (mit großer Wahrscheinlichkeit)

$$\delta_{12} = 7,\ \delta_{13} = 8,\ \delta_{14} = 23,\ \delta_{23} = 1,\ \delta_{24} = 16,\ \delta_{34} = 15.$$

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 (oder eine vollständige Suche durchführen). Dieser ist L *$(16,3\%)$. Also ist $\delta =$* L$-$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 wie folgt vorgehen. Man geht zunächst davon aus, dass jedes Kryptotextzeichen durch Summation eines Klartext- und Schlüsselstromzeichens mit jeweils mittlerer bis hoher Wahrscheinlichkeit entstanden ist. Dies sind etwa im Englischen die Zeichen 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 $w_1, w_2 \in \{\mathsf{E}, \mathsf{T}, \mathsf{A}, \mathsf{O}, \mathsf{I}, \mathsf{N}, \mathsf{S}, \mathsf{R}, \mathsf{H}\}^*$. 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 62. *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:*

	abcde...i...t...z	*(Klartextzeichen)*
+	*MLKJI...E...T...N*	*(Schlüsselzeichen)*
=	MMMMM...M...M...M	*(Kryptotextzeichen)*

Es ergeben sich folgende wahrscheinliche Paare für die Zeichen 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*)

Diese führen auf folgende $3 \cdot 3 \cdot 1 \cdot 3 = 27$ Wortpaare (w_1, w_2):

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

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

Autokey Chiffren

Kryptotextschlüsselstrom. Diese Systeme bieten so gut wie keinen 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 $\delta = 1, 2, \ldots$ von dem um δ Positionen verschobenen Kryptotext – also $y_{0+\delta}\, y_{1+\delta}\, y_{2+\delta}\, y_{3+\delta}\, \ldots$ minus $y_0\, y_1\, y_2\, y_3 \ldots$ –, bis sinnvoller (Klar-) Text erscheint:

	dumsqmozkfn...	(Kryptotext y)
−	*DUMSQMO...*	(„Kryptotextschlüsselstrom“)
=	....NSCHUTZ...	(Klartext x)

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 $k_0 \ldots k_{d-1}$. Falls im Klartext die gleiche Buchstabenfolge $x_i \ldots x_{i+l-1}$ im Abstand $2d$ auftritt (beispielsweise $d = 3$ und $l = 2$),

	$\downarrow\downarrow \quad \downarrow\downarrow$	
	$x_0\, x_1\, x_2\, x_3\, \underset{\sim}{x_4\, x_5}\, x_6\, x_7\, x_8\ \ x_9\, \underset{\sim}{x_{10}\, x_{11}}\, x_{12}\, x_{13}\, x_{14}\ \ldots$	Klartext x
+	$k_0\, k_1\, k_2\, x_0\, x_1\, x_2\ \ x_3\, \underset{\sim}{x_4\, x_5}\, x_6\, x_7\, x_8\ \ x_9\ \ \underset{\sim}{x_{10}\, x_{11}}\ldots$	Klartextschlüsselstrom kx
=	$y_0\, y_1\, y_2\, y_3\, y_4\, y_5\ \ y_6\ \underline{y_7\, y_8}\ \ y_9\ \underline{y_{10}\, y_{11}}\ \ y_{12}\, y_{13}\, y_{14}\ \ldots$	Kryptotext y

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 $k_1 \ldots k_d$ des Schlüsselwortes der Reihe nach bestimmt werden: Da durch k_i die Klartextzeichen an den Positionen $i, d+i, 2d+i, \ldots$ eindeutig festgelegt sind, kann jedes einzelne k_i unabhängig von den anderen Schlüsselwortbuchstaben durch eine statistische Analyse bestimmt werden.

3 Sicherheit von Kryptosystemen

3.1 Informationstheoretische Sicherheit

Claude E. Shannon untersuchte die Sicherheit kryptografischer Systeme auf informationstheoretischer Basis (1945, freigegeben 1949). Seinen Untersuchungen liegt das Modell einer Nachrichtenquelle X zugrunde, die einzelne Klartextnachrichten x aus dem Klartextraum M unter einer bestimmten Wahrscheinlichkeitsverteilung $p(x) = \Pr[X = x]$ generiert.

Zudem nehmen wir an, dass der zur Verschlüsselung benutzte Schlüssel $k \in K$ von einem Schlüsselgenerator S unter einer bekannten Wahrscheinlichkeitsverteilung $p(k) = \Pr[S = k]$ erzeugt wird. Da der Schlüssel unabhängig vom Klartext gewählt wird, ist $p(k, x) = p(k)p(x)$ die Wahrscheinlichkeit dafür, dass X den Klartext x generiert und dieser mit dem Schlüssel k verschlüsselt wird. Dabei gehen wir davon aus, dass für jede Nachricht $x \in M$ ein neuer Schlüssel gewählt wird. Dies bedeutet, dass wir beispielsweise bei der additiven Chiffre den Klartextraum auf $M = A^n$ vergrößern müssen, falls der Schlüssel nach n Zeichen gewechselt wird.

Die Zufallsvariablen X und S induzieren eine Verteilung auf dem Kryptotextraum, die wir durch die Zufallsvariable Y beschreiben. Die Wahrscheinlichkeit eines Kryptotextes y berechnet sich zu

$$p(y) = \Pr[Y = y] = \sum_{k,x:E(k,x)=y} p(k, x)$$

und für einen beobachteten Kryptotext y (mit $p(y) > 0$) ist

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

die (bedingte) Wahrscheinlichkeit dafür, dass sich hinter dem Kryptotext y der Klartext x verbirgt. Da der Schlüsselgenerator für die Sicherheit eines Kryptosystems eine wichtige Rolle spielt, nehmen wir bei Sicherheitsbetrachtungen die Schlüsselverteilung S als sechste Komponente eines Kryptosystems hinzu.

Definition 63. *Ein Kryptosystem $KS = (M, C, E, D, K, S)$ mit Schlüsselverteilung S heißt **informationstheoretisch** (oder **absolut**) **sicher**, falls jede Klartextverteilung X auf M unabhängig von der zugehörigen Kryptotextverteilung Y auf C ist.*

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 ändert sich nicht, ob nun der Kryptotext y bekannt ist oder nicht. Die Kenntnis von y erlaubt somit keinerlei Rückschlüsse auf die gesendete Nachricht. Dies bedeutet, dass es dem Gegner nicht möglich ist, das System zu brechen; auch nicht mit unbegrenzten Rechenressourcen. Wie wir sehen werden, lässt sich dieses Maß an Sicherheit nur mit einem sehr hohen Aufwand erreichen.

Sind $p(x), p(y) > 0$, so gilt wegen $p(x|y)p(y) = p(x, y) = p(y|x)p(x)$ die Gleichheit

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \quad \text{(Satz von Bayes)}$$

und daher ist die Bedingung $p(x) = p(x|y)$ gleichbedeutend mit $p(y) = p(y|x)$, was wiederum mit der Unabhängigkeit der Ereignisse $X = x$ und $Y = y$ gleichbedeutend ist.

Beispiel 64. *Sei $KS = (M, C, E, D, K)$ ein Kryptosystem mit $M = \{x_1, \ldots, x_4\}$, $K = \{k_1, \ldots, k_4\}$, $C = \{y_1, \ldots, y_4\}$ und Verschlüsselungsfunktion*

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

Unter der Schlüssel- und Klartextverteilung

k_i	k_1	k_2	k_3	k_4
$p(k_i)$	1/2	1/4	1/8	1/8

bzw.

x_i	x_1	x_2	x_3	x_4
$p(x_i)$	1/2	1/6	1/6	1/6

ergibt sich wegen $p(y) = \sum_{k,x:E(k,x)=y} p(k, x)$ folgende Kryptotextverteilung:

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

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 Kryptosystem nicht absolut sicher. ◁

Lässt sich das Kryptosystem KS aus obigem Beispiel unter der vorgegebenen Klartextverteilung durch Verwendung eines anderen Schlüsselgenerators absolut sicher machen?

- KS ist genau dann absolut sicher, wenn $p(y_j) = p(y_j|x_i)$ für alle $(x_i, y_j) \in M \times C$ gilt.
- Da es jedoch in KS für jedes Paar (x_i, y_j) genau einen Schlüssel $k = k_{i,j} \in K$ mit $E(k, x_i) = y_j$ gibt, also $p(y_j|x_i) = p(k_{i,j})$ ist, ist dies äquivalent zur Bedingung, dass $p(y_j) = p(k_{i,j})$ für alle $(x_i, y_j) \in M \times C$ gilt.
- Für $j = 1$ bedeutet die Gleichheit $p(y_j) = p(k_{i,j})$ für alle i zum Beispiel, dass alle vier Schlüssel $k_{i,1} = k_i$ $(i = 1, \ldots, 4)$ die gleiche Wahrscheinlichkeit haben müssen.
- Wegen $p(y_j) = \sum_{i=1}^{4} p(x_i)p(y_j|x_i) = 1/4 \sum_{i=1}^{4} p(x_i) = 1/4 = p(k_{i,j}) = p(y_j|x_i)$ ist das System in diesem Fall tatsächlich absolut sicher.

Demnach ist das Kryptosystem KS aus Beispiel 64 genau dann absolut sicher, wenn der Schlüssel gleichverteilt ist. In Verallgemeinerung dieses Beispiels lässt sich für eine wichtige Klasse von Kryptosystemen die absolute Sicherheit wie folgt charakterisieren.

Satz 65. *Sei $KS = (M, C, E, D, K, S)$ ein Kryptosystem mit $\|M\| = \|C\| = \|K\|$, dessen Schlüsselraum K für jedes Klartext-Kryptotext-Paar $(x, y) \in M \times C$ genau einen Schlüssel k mit $E(k, x) = y$ enthält. Dann ist KS genau dann absolut sicher, wenn S auf K gleichverteilt ist.*

Beweis. Bezeichne $k_{x,y}$ den eindeutigen Schlüssel, der den Klartext x auf den Kryptotext y abbildet. Falls S auf K gleichverteilt ist, folgt wegen $p(k_{x,y}) = \|K\|^{-1}$ für alle x, y mit $p(x) > 0$ zunächst

$$p(y|x) = \sum_{k:E(k,x)=y} p(k) = p(k_{x,y}) = \|K\|^{-1}$$

und

$$p(y) = \sum_{x,p(x)>0} p(x)p(y|x) = \|K\|^{-1} \sum_{x} p(x) = \|K\|^{-1},$$

also $p(x, y) = p(x)p(y|x) = p(x)p(y)$, d.h. KS ist absolut sicher. Die Umkehrung wird in den Übungen gezeigt. □

Verwendet man beim One-Time-Pad nur Klartexte einer festen Länge n, so ist dieser nach obigem Satz genau dann absolut sicher, wenn der Schlüssel unter Gleichverteilung gewählt wird. 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:

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

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 2.5) rekonstruiert werden.

Da in einem absolut sicheren Kryptosystem der Schlüsselraum K mindestens die Größe des Klartextraumes X haben muss (siehe Übungen), erfordert die absolute Sicherheit einen extrem hohen Aufwand. Vor der Kommunikation muss ein Schlüssel, der mindestens so lang wie der zu übertragende Klartext ist, zufällig generiert und zwischen den Partnern auf einem sicheren Kanal ausgetauscht werden.

Für die meisten Anwendungen ist jedoch keine absolute Sicherheit erforderlich. Wie wir bei der Betrachtung von Stromsystemen gesehen haben, kann der Schlüsselstrom auch von einem Pseudo-Zufallsgenerator erzeugt werden. Dieser erhält als Eingabe eine Zufallszahl 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 der Keim ausgetauscht werden.

3.2 Der Entropiebegriff

In der Informationstheorie wird die Unsicherheit, mit der eine durch eine Zufallsvariable X beschriebene Quelle ihre Nachrichten aussendet, nach ihrer Entropie bemessen. Dabei entspricht die Unsicherheit über X genau dem Informationsgewinn, der sich aus der Beobachtung der Quelle X ziehen lässt. Intuitiv ist die in einer einzelnen Nachricht x steckende Information umso größer, desto unwahrscheinlicher sie ist. Tritt eine Nachricht x mit einer positiven Wahrscheinlichkeit $p(x) = \Pr[X = x] > 0$ auf, dann ist

$$Inf_X(x) = \log_2(1/p(x))$$

der **Informationsgehalt** von x. Ist dagegen $p(x) = 0$, so sei $Inf_X(x) = 0$. Diese Definition des Informationsgehalts ergibt sich zwangsläufig aus den beiden folgenden Axiomen:

- Der (gemeinsame) Informationsgehalt $Inf_{X,Y}(x, y)$ von zwei Nachrichten x und y, die aus unabhängigen Quellen X und Y stammen, ist $Inf_X(x) + Inf_Y(y)$.
- Eine Nachricht x, die mit Wahrscheinlichkeit $\Pr[X = x] = 1/2$ auftritt, hat den Informationsgehalt $Inf_X(x) = 1$.

Der Informationsgehalt wird in der Einheit bit (basic indissoluble information unit) gemessen. Die Entropie von X ist nun der erwartete Informationsgehalt einer von X generierten Nachricht.

Definition 66. *Sei X eine Zufallsvariable mit Wertebereich $W(X) = \{x_1, \ldots, x_n\}$ und sei $p_i = \Pr[X = x_i]$. Dann ist die **Entropie** von X definiert als*

$$\mathcal{H}(X) = \sum_{i=1}^{n} p_i \, Inf_X(x_i) = \sum_{i=1}^{n} p_i \log_2(1/p_i) = -\sum_{i=1}^{n} p_i \log_2(p_i).$$

Beispiel 67. *Sei X eine Zufallsvariable mit der Verteilung*

x_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

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

Die Entropie nimmt für $p_1 = \cdots = p_n = 1/n$ den Wert $\log_2(n)$ an. Für jede andere Verteilung $p_1, \ldots, p_n$ gilt dagegen $\mathcal{H}(X) < \log_2(n)$ (Beweis siehe unten). Bei vorgegebener Größe des Wertebereichs von X ist die Unsicherheit über X um so größer, je gleichmäßiger X verteilt ist. Bringt X dagegen nur einen einzigen Wert mit positiver Wahrscheinlichkeit hervor, dann (und nur dann) nimmt $\mathcal{H}(X)$ den Wert 0 an. Für den Nachweis von oberen Schranken für die Entropie benutzen wir folgende Hilfsmittel aus der Analysis.

Definition 68. *Sei $I \subseteq \mathbb{R}$ ein Intervall. Eine Funktion $f : I \to \mathbb{R}$ heißt **konkav** auf I, falls für alle $x \neq y \in I$ und $0 \leq t \leq 1$ gilt:*

$$f(tx + (1-t)y) \quad \geq \quad tf(x) + (1-t)f(y).$$

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

Beispiel 69. *Die Funktion $f(x) = \log_2(x)$ ist streng konkav auf $(0, \infty)$.* $\triangleleft$

Für den Beweis des nächsten Satzes benötigen wir die Jensensche Ungleichung, die wir ohne Beweis angeben.

Satz 70 (Jensensche Ungleichung)**.** *Sei f eine streng konkave Funktion auf I und seien $0 < a_1, \ldots, a_n < 1$ reelle Zahlen mit $\sum_{i=1}^{n} a_i = 1$. Dann gilt für alle $x_1, \ldots, x_n \in I$,*

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

Hierbei tritt Gleichheit genau dann ein, wenn alle x_i den gleichen Wert haben.

Satz 71. *Sei X eine Zufallsvariable auf einer n-elementigen Menge $\{x_1, \ldots, x_n\}$ mit der Verteilung $p_i = \Pr[X = x_i]$ für $i = 1, \ldots, n$. Dann ist $H(X) \leq \log_2(n)$, wobei Gleichheit genau im Fall $p_i = 1/n$ für $i = 1, \ldots, n$ eintritt.*

Beweis. Aufgrund der Jensenschen Ungleichung gilt

$$\mathcal{H}(X) = \sum_{i=1}^{n} p_i \log_2(1/p_i) \leq \log_2 \sum_{i=1}^{n} p_i/p_i = \log_2 n,$$

wobei Gleichheit genau im Fall $1/p_1 = \cdots = 1/p_n$ eintritt. Letzteres ist mit der Bedingung $p_i = 1/n$ für $i = 1, \ldots, n$ gleichbedeutend. □

Die Entropie liefert eine sehr gute untere Schranke für die mittlere Codewortlänge von Binärcodes. Ein **Binärcode** für X ist eine (geordnete) Menge $C = \{y_1, \ldots, y_n\}$ von binären Codewörtern y_i für die Nachrichten x_i mit der Eigenschaft, dass die Abbildung $c : W(X)^* \to \{0,1\}^*$ mit $c(x_{i_1} \cdots x_{i_k}) = y_{i_1} \cdots y_{i_k}$ injektiv ist. Die Injektivität von c stellt sicher, dass jede Folge $y_{i_1} \cdots y_{i_k}$ von Codewörtern eindeutig decodierbar ist.
Die **mittlere Codewortlänge** von C unter X ist

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

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

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

Beispiel 72. *Sei X die Zufallsvariable aus dem letzten Beispiel mit der Verteilung $p_1 = p_2 = 1/4$, $p_3 = p_4 = p_5 = 1/8$ und $p_6 = p_7 = 1/16$. 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 hat C_1 die mittlere Codewortlänge $L(C_1) = 3$, während $C_2 = \{y_1, \ldots, y_7\}$ wegen $|y_i| = \log_2(1/p_i)$ den optimalen Wert $L(C_2) = \mathcal{H}(X) = 2{,}625$ erreicht.* ◁

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 definiert 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) $|\mathtt{GOLD}|_2 = 4 \log_2(26) = 18{,}8$. Entsprechend berechnet sich für einen Code $C = \{y_1, \ldots, y_n\}$ unter einer Verteilung $p_1, \ldots, p_n$ die mittlere Codewortlänge (in bit) zu

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

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

Definition 73. *Die **(relative) Redundanz** eines Codes C für X ist definiert als*

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

Beispiel 74. *Während eine von X generierte Nachricht im Durchschnitt $\mathcal{H}(X) = 2.625$ 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. ◁

3.3 Redundanz von Sprachen

Auch Schriftsprachen wie Deutsch oder Englisch und Programmiersprachen wie C oder PASCAL können als eine Art Code aufgefasst werden. 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. Um die statistischen Eigenschaften einer 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 festen Codewortlänge n, so ist

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

die Redundanz dieses Codes.

Definition 75 (**Entropie einer Sprache**)**.** *Für eine Sprache L über einem Alphabet A mit $\|A\| = m$ ist $\mathcal{H}(L_n)/n$ die **n-Gramm-Entropie von** L (pro Buchstabe). Falls dieser Wert für n gegen ∞ von oben gegen einen Grenzwert*

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

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

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

*der als die **(relative) Redundanz** von L bezeichnet wird. Der Zähler*

$$\mathcal{R}_{abs}(L) = \log_2 m - \mathcal{H}(L) = \mathcal{R}(L) \log_2 m$$

*wird auch als **absolute Redundanz** von L bezeichnet (gemessen in bit/Zeichen).*

Die Redundanz von natürlichen Sprachen lässt sich näherungsweise bestimmen, indem man die Entropien $\mathcal{H}(L_n)$ ihrer n-Gramme empirisch ermittelt.

Beispiel 76. *Im Deutschen hat die Einzelzeichenverteilung eine Entropie von $\mathcal{H}(L_1) = 4,1$ bit, während eine auf A_{lat} gleichverteilte Zufallsvariable U einen Entropiewert von $\mathcal{H}(U) = \log(26) = 4,7$ bit hat. Für die Bigramme ergibt sich ein Entropiewert von*

$\mathcal{H}(L_2)/2 = 3,5$ *bit pro Buchstabe. Mit wachsender Länge sinkt die Entropie von deutschsprachigen Texten weiter ab und strebt gegen einen Grenzwert* $\mathcal{H}(L)$ *von* $1,5$ *bit pro Buchstabe.*

n	$\mathcal{H}(L_n)$	$\mathcal{H}(L_n)/n$	$\mathcal{R}_{abs}(L_n)/n$	$\mathcal{R}(L_n)$
1	$4,1$	$4,1$	$0,6$	13%
2	$7,0$	$3,5$	$1,2$	26%
3	$9,6$	$3,2$	$1,5$	32%
6	$12,2$	$2,0$	$2,7$	57%
15	$27,6$	$1,8$	$2,9$	62%
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
∞	∞	$\mathcal{H}(L) = 1,5$	$\mathcal{R}_{abs}(L) = 3,2$	$\mathcal{R}(L) = 67\%$

Deutsche Texte hinreichender Länge besitzen also eine durchschnittliche Redundanz von ca. 67%*, so dass ihre Länge bei optimaler Kodierung auf ca.* $1/3$ *komprimierbar ist.* ◁

3.4 Die Eindeutigkeitsdistanz

Wir betrachten nun den Fall, dass mit einem Kryptosystem Klartexte einer variablen Länge n verschlüsselt werden, ohne dabei den Schlüssel zu wechseln. Die Chiffrierfunktion hat also die Form

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

wobei die Klartextlänge n variabel ist und wir der Einfachheit halber annehmen, dass die Menge C_n der entsprechenden Kryptotexte die gleiche Kardinalität $\|C_n\| = \|A^n\| = m^n$ wie der Klartextraum hat. Ist y ein abgefangener Kryptotext, so ist

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

die Menge aller infrage kommenden Schlüssel. $K(y)$ besteht aus einem „echten“ (d. h. dem zur Generierung von y tatsächlich benutzten) und $\|K(y)\| - 1$ so genannten „unechten“ Schlüsseln. Aus informationstheoretischer Sicht ist das Kryptosystem unter der Klartextverteilung X umso sicherer, desto größer die erwartete Anzahl

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

der unechten Schlüssel ist. Im besten Fall kommen für jeden Kryptotext alle Schlüssel infrage, d.h. $\bar{s}_n = \|K\| - 1$. Ist dagegen $\bar{s}_n$ gleich 0, so liefert der abgefangene Kryptotext dem Gegner genügend Information, um den benutzten Schlüssel und somit den zugehörigen Klartext eindeutig bestimmen zu können (sofern er über genügend Ressourcen verfügt).

Definition 77. *Die* ***Eindeutigkeitsdistanz*** n_0 *eines Kryptosystems unter Klartextverteilung* X *ist der kleinste Wert von* n*, für den* $\bar{s}_n = 0$ *wird.*

Als nächstes wollen wir eine untere Schranke für $\bar{s}_n$ (und damit für n_0) herleiten. Hierzu benötigen wir den Begriff der bedingten Entropie $\mathcal{H}(X|Y)$ von X, wenn der Wert von Y bereits bekannt ist.

Definition 78. *Seien* X, Y *Zufallsvariablen. Die* ***bedingte Entropie*** *von* X *unter* Y *ist definiert als*

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

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

Satz 79. *Es gilt*

1. $\mathcal{H}(X,Y) = \mathcal{H}(Y) + \mathcal{H}(X|Y)$ *und*
2. $\mathcal{H}(X,Y) \leq \mathcal{H}(X) + \mathcal{H}(Y)$*, wobei Gleichheit genau dann eintritt, wenn X und Y unabhängig sind.*

Beweis. s. Übungen. □

Korollar 80. *Es gilt $\mathcal{H}(X|Y) \leq \mathcal{H}(X)$, wobei Gleichheit genau dann eintritt, wenn X und Y unabhängig sind.*

Satz 81. *In jedem Kryptosystem gilt für die Klartextentropie $\mathcal{H}(X)$, die Schlüsselentropie $\mathcal{H}(S)$ und die Kryptotextentropie $\mathcal{H}(Y)$ die Gleichung*

$$\mathcal{H}(S|Y) = \mathcal{H}(S) + \mathcal{H}(X) - \mathcal{H}(Y).$$

Beweis. Zunächst ist $\mathcal{H}(S|Y) = \mathcal{H}(S,Y) - \mathcal{H}(Y)$. Es reicht also zu zeigen, dass

$$\mathcal{H}(S,Y) = \mathcal{H}(S) + \mathcal{H}(X)$$

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 S unabhängig sind,

$$\mathcal{H}(S,Y) = \mathcal{H}(S,X,Y) - \underbrace{\mathcal{H}(X|S,Y)}_{=0} = \mathcal{H}(S,X) + \underbrace{\mathcal{H}(Y|S,X)}_{=0} = \mathcal{H}(S) + \mathcal{H}(X).$$

□

Jetzt verfügen wir über alle Hilfsmittel, um die erwartete Anzahl

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

der unechten Schlüssel nach unten abschätzen zu können.

Lemma 82. *Seien X_n und Y_n die Zufallsvariablen, die die Verteilungen der n-Gramme der Klartextsprache und der zugehörigen Kryptotexte beschreiben. Dann gilt*

1. $\mathcal{H}(S|Y_n) \leq \log_2(\bar{s}_n + 1)$,
2. $\mathcal{H}(S|Y_n) \geq \mathcal{H}(S) - n\mathcal{R}(L_n) \log_2 m$.

Beweis.

1. Unter Verwendung der Jensenschen Ungleichung folgt

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

2. Mit Satz 81 folgt
$$\mathcal{H}(S|Y_n) = \mathcal{H}(S) + \mathcal{H}(X_n) - \mathcal{H}(Y_n).$$
Für die Klartextentropie $\mathcal{H}(X_n)$ gilt
$$\mathcal{H}(X_n) = \mathcal{H}(L_n) = (1 - \mathcal{R}(L_n))n \log_2 m,$$
wobei $m = \|A\|$ ist. Zudem lässt sich die Kryptotextentropie $\mathcal{H}(Y_n)$ wegen $W(Y_n) = C_n$ und $\|C_n\| = m^n$ durch
$$\mathcal{H}(Y_n) \leq n \log_2 m$$
abschätzen. Somit ist
$$\mathcal{H}(S|Y_n) = \mathcal{H}(S) + \mathcal{H}(X_n) - \mathcal{H}(Y_n) \geq \mathcal{H}(S) - n\mathcal{R}(L_n) \log_2 m$$ □

Zusammen ergibt sich also
$$\log_2(\bar{s}_n + 1) \geq \mathcal{H}(S) - n\mathcal{R}(L_n) \log_2 m \geq \mathcal{H}(S) - n\mathcal{R}(L) \log_2 m.$$
Im Fall eines gleichverteilten Schlüssels erreicht $\mathcal{H}(S)$ den maximalen Wert $\log_2 \|K\|$, was auf die gesuchte Abschätzung für $\bar{s}_n$ führt.

Satz 83. *Werden mit einem Kryptosystem (M, C, E, D, K) mit $M = A^n$ und $\|C\| = m^n$ Klartexte einer Sprache L der festen Länge n mit gleichverteiltem Schlüssel $k \in K$ verschlüsselt, so gilt für die erwartete Anzahl $\bar{s}_n$ der unechten Schlüssel,*
$$\bar{s}_n \geq \frac{\|K\|}{m^{n\mathcal{R}(L_n)}} - 1.$$

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

Korollar 84. *Unter den Bedingungen des obigen Satzes gilt*
$$n_0 \geq \frac{\log_2 \|K\|}{\mathcal{R}(L_n) \log_2 m} \geq \frac{\log_2 \|K\|}{\mathcal{R}(L) \log_2 m} = \frac{\log_2 \|K\|}{\mathcal{R}_{abs}(L)}.$$

Man beachte, dass wir die Mindestmenge an Kryptotext, der zur eindeutigen Bestimmung des Schlüssels benötigt wird, nur nach unten abgeschätzt haben und die tatsächlich benötigte Menge deutlich größer sein kann. Natürlich erlaubt die eindeutige Bestimmung des Schlüssels auch die eindeutige Bestimmung des Klartexts. Unter Umständen kann jedoch der Klartext auch schon mit einer wesentlich geringeren Menge an Kryptotext eindeutig rekonstruierbar sein.

Beispiel 85. *Für Substitutionen bei deutschsprachigem Klartext ergeben sich folgende Werte $\log_2 \|K\|/\mathcal{R}_{abs}(L)$ als untere Schranke für die Eindeutigkeitsdistanz n_0 (wobei wir von einer absoluten Redundanz von $\mathcal{R}_{abs}(L) = 3.2$ bit/Zeichen ausgehen, was einer relativen Redundanz von $\mathcal{R}(L) = 3,2/4,7 \approx 67\%$ entspricht):*

Kryptosystem	*Schlüsselanzahl* $\|K\|$	$\log_2 \|K\|$	$\log_2 \|K\|/\mathcal{R}_{abs}(L)$
additive Chiffre	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	$4.7 \cdot d$	$1.5 \cdot d$

Dagegen erhalten wir für Blocktranspositionen folgende unteren Schranken für die Mindestmenge an Kryptotext, die zur eindeutigen Schlüsselbestimmung benötigt wird. Hierbei unterscheiden wir zusätzlich nach der Länge der bei der Häufigkeitsanalyse benutzten n-Gramme. Dies entspricht der Situation, dass die Wahrscheinlichkeit jedes Zeichens im Klartext höchstens von den $n-1$ vorausgehenden bzw. nachfolgenden Zeichen abhängt.

Untere Schranken für n_0 bei einer Analyse von Blocktranspositionen auf der Basis von		*Blocklänge ℓ*				
		10	*20*	*50*	*100*	*1 000*
Einzelzeichen-Häufigkeiten	$(\mathcal{R}(L_1) = 0,6)$	*59*	*165*	*578*	*1415*	*22 986*
Bigramm-Häufigkeiten	$(\mathcal{R}(L_2) = 1,2)$	*40*	*111*	*390*	*954*	*15 502*
Trigramm-Häufigkeiten	$(\mathcal{R}(L_3) = 1,5)$	*24*	*65*	*226*	*553*	*9 473*
n-Gramm-Häufigkeiten, $n \to \infty$	$(\mathcal{R}(L) = 3,2)$	*7*	*19*	*67*	*164*	*2 665*

◁

3.5 Weitere Sicherheitsbegriffe

Da die Benutzung eines informationstheoretisch sicheren Kryptosystems einen immensen Aufwand erfordert, begnügt man sich in der Praxis 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, der Zeitaufwand und die Kosten für einen erfolgreichen Angriff (sofern er überhaupt möglich ist) übersteigen den potentiellen Nutzen bei weitem.
- Ein Kryptosystem gilt als **nachweisbar sicher** (***provably secure***), wenn seine Sicherheit mit bekannten komplexitätstheoretischen Hypothesen verknüpft werden kann, deren Gültigkeit gemeinhin akzeptiert wird.
- 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 Verbreitung gefunden haben und allein schon deshalb ein attraktives Ziel für einen Angriff darstellen.

Die komplexitätstheoretische Analyse eines Kryptosystems ist äußerst schwierig, da der Aufwand für einen erfolgreichen Angriff unabhängig von der dabei benutzten Technik abgeschätzt werden muss. Es reicht also nicht, alle bekannten kryptoanalytischen Ansätze in Betracht zu ziehen, sondern alle *möglichen*. Dabei darf sich die Aufwandsanalyse nicht ausschließlich an einer vollständigen Rekonstruktion des Klartextes orientieren, da bereits ein kleiner Unterschied zwischen der A-posteriori- und A-priori-Wahrscheinlichkeit für den Gegner einen Vorteil bedeuten könnte.

Aus den genannten Gründen ist noch für kein praktikables Kryptosystem der Nachweis gelungen, dass es komplexitätstheoretisch sicher ist. Damit ist auch nicht so schnell zu rechnen, zumindest nicht solange der Status fundamentaler komplexitätstheoretischer Fragen wie etwa des berühmten $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$-Problems offen ist. Dagegen gibt es eine ganze Reihe praktikabler Kryptosysteme, die als nachweisbar sicher oder praktisch sicher gelten.

Wir schließen diesen Abschnitt mit einer Präzisierung des komplexitätstheoretischen Sicherheitsbegriffs, die unter dem Namen IND-CPA (indistinguishability under a chosen-

plaintext attack) bekannt ist. Hierzu ist es erforderlich, die Verletzung der Vertraulichkeit als ein algorithmisches Problem für den Gegner zu formulieren. Konkret läuft ein IND-CPA-Angriff wie folgt ab.

1. Zuerst wählt der Gegner zwei Klartexte $x_0 \neq x_1 \in M$.
2. Dann wird x_0 oder x_1 zufällig ausgewählt und der zugehörige Kryptotext y gebildet.
3. Dem Gegner wird der Kryptotext y vorgelegt und er muss raten, welcher der beiden Klartexte sich hinter y verbirgt.
4. Der Angriff ist erfolgreich, falls der Gegner richtig rät.

Die Erfolgsaussichten des Gegners bei diesem Angriff lassen sich wie folgt formalisieren. Dabei gehen wir davon aus, dass das gewünschte Maß an Sicherheit durch einen Parameter $s \in \mathbb{N}$ reguliert wird. Typischerweise werden Kryptosysteme nach ihrer Schlüssellänge $s = |k|$ parameterisiert. Aus Praktikabilitätsgründen sollten dann alle legalen Operationen (wie die Chiffrierung oder die Schlüsselgenerierung) effizient (d.h. in Zeit $s^{O(1)}$) durchführbar sein. Natürlich darf dann auch der Aufwand des Gegners in Abhängigkeit von s steigen, weshalb er zusätzlich den Parameterwert s erhält.

Definition 86 (IND-CPA Angriff). *Sei (M, C, E, D, K, S) ein Kryptosystem mit Sicherheitsparameter $s \in \mathbb{N}$. Ein (**IND-CPA-)Gegner** ist ein Tripel $G = (X_0, X_1, V)$ von probabilistischen Algorithmen, wobei X_0, X_1 bei Eingabe s zwei Klartexte aus M generieren und V bei Eingabe von s, $x_0, x_1 \in M$ und $y \in C$ ein Bit $V(s, x_0, x_1, y) \in \{0, 1\}$ ausgibt. Der **Vorteil** von G bei Parameterwert s ist*

$$\alpha_G(s) = 2(\Pr[V(X_0(s), X_1(s), E(S, X_B(s))) = B] - 1/2),$$

wobei B auf $\{0, 1\}$ gleichverteilt und von S, X_0, X_1, V unabhängig ist.

Ist der Wert des Sicherheitsparameters s irrelevant, fest vorgegeben oder aus dem Kontext ersichtlich, so verzichten wir meist auf seine explizite Angabe.

Wird beispielsweise eine Folge von Klartextblöcken $a_1, a_2, \ldots$ mit einer Blockchiffre verschlüsselt, indem die einzelnen Blöcke unabhängig voneinander mit demselben Schlüssel k zu einer Folge $b_1, b_2, \ldots$ von Kryptotextblöcken $b_i = E(k, a_i)$ verschlüsselt werden (so genannter *ECB-Modus*; electronic code book mode), so kann ein Gegner ohne großen Aufwand einen Vorteil von 1 erzielen (d.h. mit Wahrscheinlichkeit 1 den richtigen Klartext raten). Hierzu wählt er (deterministisch) zwei beliebige Klartexte $x_0 = a_1 a_2 \ldots$ und $x_1 = a_1' a_2' \ldots$ mit der Eigenschaft $a_1 = a_2$ und $a_1' \neq a_2'$. Dann kann er bei Vorlage eines Kryptotextes $y = b_1 b_2 \ldots$ leicht erkennen, aus welchem Klartext y generiert wurde:

$$V(x_0, x_1, y) = \begin{cases} 0, & b_1 = b_2 \\ 1, & \text{sonst.} \end{cases}$$

Erwartungsgemäß sind absolut sichere Kryptosysteme gegen IND-CPA-Angriffe resistent.

Satz 87. *Der maximale Vorteil gegenüber einem absolut sicheren Kryptosystem ist gleich Null (d.h. ein IND-CPA-Gegner kann höchstens mit Wahrscheinlichkeit $1/2$ den richtigen Klartext raten, auch wenn er über unbeschränkte Rechenressourcen verfügt).*

Beweis. Bei einem absolut sicheren Kryptosystem sind der Kryptotext $Y = E(S, X)$ und der Klartext X unabhängig. Daher sind auch die Zufallsvariablen $V(X_0, X_1, E(S, X_B))$

und B unabhängig und es folgt

$$\begin{aligned}
&\Pr[V(X_0, X_1, E(S, X_B)) = B] \\
&= \Pr[V(X_0, X_1, E(S, X_B)) = B = 0] + \Pr[V(X_0, X_1, E(S, X_B)) = B = 1] \\
&= \Pr[V(X_0, X_1, E(S, X_B)) = 0] \cdot \underbrace{\Pr[B = 0 \mid V(X_0, X_1, E(S, X_B)) = 0]}_{= \Pr[B=0] = 1/2} \\
&\quad + \Pr[V(X_0, X_1, E(S, X_B)) = 1] \cdot \underbrace{\Pr[B = 1 \mid V(X_0, X_1, E(S, X_B)) = 1]}_{= \Pr[B=1] = 1/2} \\
&= 1/2.
\end{aligned}$$

□

In den Übungen wird auch die umgekehrte Implikation bewiesen. Ein Kryptosystem ist somit genau dann absolut sicher, wenn kein Gegner einen Vorteil größer als 0 erzielen kann. Für die Präzisierung der komplexitätstheoretischen Sicherheit sind nun die folgenden beiden Fragen von entscheidender Bedeutung:

1. Über welche Rechenressourcen verfügt ein Gegner realistischerweise?
2. Wie groß darf der vom Gegner erzielbare Vorteil höchstens sein, ohne die Vertraulichkeit der verschlüsselten Nachricht zu verletzen?

Bezüglich Frage 1 geht man typischerweise davon aus, dass der Gegner über probabilistische Schaltkreise polynomieller Größe verfügt.

Definition 88.

a) Ein ***boolescher Schaltkreis*** *der Größe m mit Eingängen $x_1, \ldots, x_n$ und Ausgängen $i_1, \ldots, i_l \in [m]$ ist eine Folge $c = (g_1, \ldots, g_m)$ von* ***Gattern***

$$g_l \in \{0, 1, x_1, \ldots, x_n, (\neg, j), (\wedge, j, k), (\vee, j, k)\} \text{ mit } 1 \leq j, k < l.$$

b) Der von c bei Eingabe $a \in \{0,1\}^n$ am Gatter g_l berechnete Wert $g_l(a) \in \{0,1\}$ ist induktiv wie folgt definiert:

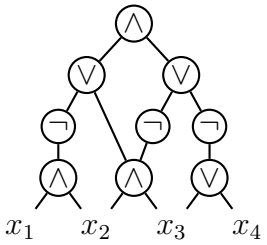
g_l	0	1	x_i	$(\neg, j)$	$(\wedge, j, k)$	$(\vee, j, k)$
$g_l(a)$	0	1	a_i	$1 - g_j(a)$	$g_j(a)g_k(a)$	$g_j(a) + g_k(a) - g_j(a)g_k(a)$

c) Die ***Ausgabe*** *von c bei Eingabe $a \in \{0,1\}^n$ ist die Bitfolge $c(a) = g_{i_1}(a) \ldots g_{i_l}(a)$.*

Beispiel 89. *Der Schaltkreis*

$$\begin{aligned}
c = \quad & (x_1, x_2, x_3, x_4, (\wedge, 1, 2), (\wedge, 2, 3), \\
& (\vee, 3, 4), (\neg, 5), (\neg, 6), (\neg, 7), \\
& (\vee, 6, 8), (\vee, 9, 10), (\wedge, 11, 12))
\end{aligned}$$

mit den Eingängen x_1, x_2, x_3, x_4 und Ausgängen $(11, 12, 13)$ gibt bei Eingabe $a = 0110$ die Bitfolge $c(0110) = 100$ aus.



◁

Ein **probabilistischer Schaltkreis** c hat neben den regulären Eingabegattern $x_1, \ldots, x_n$ noch eine beliebige Anzahl von Zufallsgattern $z_1, \ldots, z_m$. Hierbei werden die Eingabegatter x_i wie bisher mit den Bits a_i eines Eingabevektors $a = a_1 \ldots a_n \in \{0,1\}^n$ belegt, während die m Zufallsgatter unabhängig gleichverteilte Bits $Z_1, \ldots, Z_m$ erzeugen (d.h. es gilt $\Pr[Z_1 \ldots Z_m = b] = 2^{-m}$ für alle $b \in \{0,1\}^m$). Dadurch wird die Ausgabe $c(a, Z_1, \ldots, Z_m)$ zu einer Zufallsvariablen, die wir auch kurz mit $C(a)$ bezeichnen.

Bezüglich der zweiten Frage verlangt man, dass der Gegner für jedes Polynom p höchstens für endlich viele Parameterwerte s einen Vorteil größer gleich $1/p(s)$ erzielen darf. Andernfalls wäre die Sicherheit gefährdet, da er für jedes solche s nach polynomiell vielen Wiederholungen der probabilistischen Berechnung von $V(s, x_0, x_1, y)$ fast sicher den richtigen Klartext ausfindig machen könnte, indem er das mehrheitlich berechnete Bit ausgibt.

Definition 90. *Sei KS ein Kryptosystem mit variablem Sicherheitsparameter $s \in \mathbb{N}$.*

- *Eine Funktion $\varepsilon : \mathbb{N} \to \mathbb{R}$ heißt **vernachlässigbar**, wenn für jedes Polynom p eine Zahl $n_0 \in \mathbb{N}$ existiert, so dass $\varepsilon(n) < 1/p(n)$ für alle $n \geq n_0$ gilt.*
- *Ein Gegner $G = (X_0, X_1, V)$ heißt **effizient**, wenn probabilistische Schaltkreise c und c' der Größe $s^{O(1)}$ mit $C(s) = (X_0(s), X_1(s))$ und $C'(s, x_0, x_1, y) = V(s, x_0, x_1, y)$ existieren, wobei die Ein- und Ausgaben von c und c' binär kodiert sind.*
- *KS heißt **komplexitätstheoretisch sicher**, wenn jeder effiziente Gegner G nur einen vernachlässigbaren Vorteil erzielen kann (d.h. die Funktion $\alpha_G(s)$ ist vernachlässigbar).*

4 Moderne symmetrische Kryptosysteme & ihre Analyse

4.1 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 91. *Seien $KS_1 = (M_1, C_1, E_1, D_1, K_1, S_1)$ und $KS_2 = (M_2, C_2, E_2, D_2, K_2, S_2)$ Kryptosysteme mit $C_1 = M_2$. Dann ist das **Produktkryptosystem** $KS_1 \times KS_2$ von KS_1 und KS_2 definiert als $(M_1, C_2, E, D, K_1 \times K_2, S)$ mit $S = (S_1, S_2)$ und*

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

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

Der Schlüsselraum von $KS_1 \times KS_2$ 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 92. *Sei $A = \{a_0, \ldots, a_{m-1}\}$. Man sieht leicht, dass die affine Chiffre $KS = (M, C, K, E, D)$ mit $M = C = \mathcal{A}$ und $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$ das Produkt $KS = KS_1 \times KS_2$ der multiplikativen Chiffre $KS_1 = (M, C, K_1, E_1, D_1)$ und der additiven Chiffre $KS_2 = (M, C, K_2, E_2, D_2)$ ist, da für jeden Schlüssel $k = (k_1, k_2) \in K = K_1 \times K_2 = \mathbb{Z}_m^* \times \mathbb{Z}_m$ gilt:*

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

Das ist exakt die affine Chiffre. Welche Chiffre erhalten wir, wenn wir die Reihenfolge von KS_1 und KS_2 vertauschen? Für $KS' = KS_2 \times KS_1$ ergibt sich das Kryptosystem $KS' = (M, C, K', E', D')$ mit $K' = K_2 \times K_1 = \mathbb{Z}_m \times \mathbb{Z}_m^*$ und

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

für jeden Schlüssel $(k_2, k_1) \in K'$. Wir sehen also, dass die Abbildung

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

eine Bijektion zwischen den Schlüsselräumen K' und K ist und der Schlüssel (k_2, k_1) im System KS' die gleiche Chiffrierfunktion realisiert wie der Schlüssel $(k_1, k_1 k_2)$ in KS. Zudem können wir jeden Schlüsselgenerator S' für KS' in einen Schlüsselgenerator S für KS transformieren (und auch S wieder zurück in S'), so dass S in KS jede Chiffrierfunktion mit der gleichen Wahrscheinlichkeit erzeugt wie S' in KS'. Daher können wir die Kryptosysteme $KS = KS_1 \times KS_2$ und $KS' = KS_2 \times KS_1$ als gleich (genauer: äquivalent, siehe Übungen) ansehen, d.h. KS_1 und KS_2 kommutieren. ◁

Definition 93. *Ein Kryptosystem $KS = (M, C, K, D, E)$ mit $M = C$ heißt **endomorph**. Ein endomorphes Kryptosystem KS heißt **idempotent**, falls $KS \times KS$ äquivalent zu KS ist (in Zeichen: $KS \times KS = KS$).*

Beispiel 94. *Eine leichte Rechnung zeigt, dass die additive Chiffre, die multiplikative Chiffre 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 KS durch die Kombination $KS = KS_1 \times KS_2$ zweier idempotenter Verfahren KS_1 und KS_2 zu erhalten. Da KS im Fall $KS_1 \times KS_2 = KS_2 \times KS_1$ wegen

$$\begin{aligned}(KS_1 \times KS_2) \times (KS_1 \times KS_2) &= KS_1 \times (KS_2 \times KS_1) \times KS_2 \\ &= KS_1 \times (KS_1 \times KS_2) \times KS_2 \\ &= (KS_1 \times KS_1) \times (KS_2 \times KS_2) \\ &= KS_1 \times KS_2\end{aligned}$$

idempotent ist, dürfen hierbei KS_1 und KS_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. Einzelne Schlüssel eines Kryptosystems werden wir in diesem Kapitel mit K bezeichnen.

Eine iterierte Blockchiffre wird typischerweise durch eine **Rundenfunktion** (*round function*) g und einen **Key-Schedule Algorithmus** 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:

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

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

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

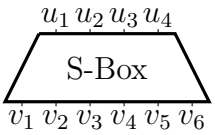
Beispiele für iterierte Chiffren sind der aus 16 Runden bestehende DES-Algorithmus und der AES mit einer variablen Rundenzahl $N \in \{10, 12, 14\}$, die wir in späteren Abschnitten behandeln werden.

4.2 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 95. *Für ein Wort $u = u_1 \cdots u_n \in \{0,1\}^n$ und Indizes $1 \le i \le j \le n$ bezeichne $u[i,j]$ das* ***Teilwort*** *$u_i \cdots u_j$ von u. Im Fall $n = ml$ 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 $|u_{(i)}| = l$ ist.*

Sei $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ eine Substitution, die Binärblöcke u der Länge l in Binärblöcke $v = \sigma_S(u)$ der Länge l' überführt (auch kurz als **S-Box** S bezeichnet, oft schreiben wir für $\sigma_S(u)$ auch einfach $S(u)$.



Durch parallele Anwendung von m Kopien der S-Box S erhalten wir die Substitution $\sigma_{mS} : \{0,1\}^{ml} \to \{0,1\}^{ml'}$ mit

$$\sigma_{mS}(u_1 \cdots u_{ml}) = \sigma_S(u_{(1)}) \cdots \sigma_S(u_{(m)}).$$

Auch hier schreiben wir für $\sigma_{mS}(u_1 \cdots u_{ml})$ einfach $S(u_1 \cdots u_{ml})$. Für die Speicherung einer S-Box $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ auf einem Speicherchip 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 Smartcard-Anwendungen bereits ausschließen würde.

Für eine Transposition P auf $\{0,1\}^\ell$ bezeichnen wir die zugehörige Permutation auf $[\ell]$ mit π_P oder einfach mit π, falls P aus dem Kontext bekannt ist, d.h.

$$P(u_1 \cdots u_\ell) = u_{\pi(1)} \cdots u_{\pi(\ell)}.$$

Definition 96. *Für natürliche Zahlen $m, l \ge 1$ sei $M = C = \{0,1\}^\ell$ mit $\ell = ml$. Ein* ***Substitutions-Permutations-Netzwerk (SPN)*** *wird durch eine S-Box S, eine Blocktransposition P und durch eine Funktion $f : \{0,1\}^k \to \{0,1\}^{ml(N+1)}$ beschrieben, wobei S eine Permutation σ_S auf $\{0,1\}^l$ realisiert, P die Blocklänge ℓ hat und $N \ge 1$ die* ***Rundenzahl*** *des SPN ist. Die Funktion f transformiert einen (externen) Schlüssel $K \in \{0,1\}^k$ in ein* ***Key-Schedule*** *$f(K) = (K^1, \ldots, K^{N+1})$ von $N+1$ Rundenschlüsseln K^r, $r = 1, \ldots, N+1$, unter denen ein Klartext $x \in \{0,1\}^\ell$ in N Runden durch folgenden Chiffrieralgorithmus in einen Kryptotext $y = E_{f,S,P}(K,x) \in \{0,1\}^\ell$ überführt wird:*

Chiffrierfunktion $E_{f,S,P}(K,x)$

```
1  w^0 := x
2  for r := 1 to N − 1 do
3     u^r := w^{r−1} ⊕ K^r
4     v^r := σ_mS(u^r)
5     w^r := P(v^r)
6  u^N := w^{N−1} ⊕ K^N
7  v^N := σ_mS(u^N)
8  y := v^N ⊕ 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). Das Resultat u^r wird den S-Boxen zugeführt und auf die Ausgabe v^r wird in jeder Runde $r \le N-1$ die Transposition P angewendet, was die Eingabe w^r für die nächste Runde $r+1$ liefert.

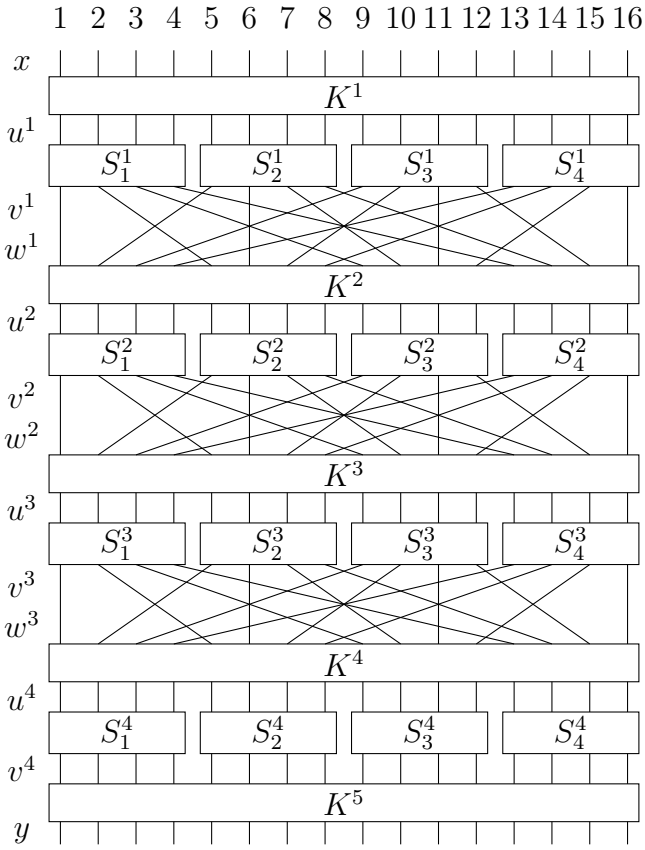


Abbildung 4.1: Ein Substitutions-Permutations-Netzwerk

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), was speziell für Hardware-Implementierungen nützlich ist.

Beispiel 97. *Wir betrachten ein SPN SP mit Parametern $l = m = N = 4$ und $k = 32$. 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 $\sigma_S : \{0,1\}^4 \to \{0,1\}^4$ und $\pi_P : \{1, \ldots, 16\} \to \{1, \ldots, 16\}$ die folgenden Permutationen (wobei die Argumente und Werte von σ_S hexadezimal dargestellt sind; siehe auch Abbildung 4.1):*

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

und

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

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

◁

4.3 Lineare Approximationen

Sei $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ die funktionale Beschreibung einer S-Box S. Wählen wir die Eingabe $U = U_1 \cdots U_l$ zufällig unter Gleichverteilung, so gilt für die zugehörige Ausgabe $V = \sigma_S(U) = V_1 \cdots V_{l'}$,

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

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

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

Ist die S-Box S **linear**, d.h σ ist eine **lineare** Funktion $\sigma_S(u) = uA$ für eine binäre $(l \times l')$-Matrix A (vgl. Definition 28), so lässt sich jedes Ausgabebit v_j von S über eine Funktion der Form $v_j = u_{i_1} \oplus \cdots \oplus u_{i_k}$ für geeignete Indizes $1 \le i_1 < \cdots < i_k \le l$ berechnen. In diesem Fall würde also

$$\Pr[V_j = U_{i_1} \oplus \cdots \oplus U_{i_k}] = 1$$

gelten. Die Idee hinter der linearen Kryptoanalyse ist nun, etwas allgemeinere Gleichungen der Form

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

mit $1 \leq i_1 < \cdots < i_k \leq l$, $1 \leq j_1 < \cdots < j_{k'} \leq l'$ und $c \in \{0,1\}$ zu finden, die mit möglichst großer Wahrscheinlichkeit gelten. Definieren wir für $a \in \{0,1\}^l$ und $b \in \{0,1\}^{l'}$ die Zufallsvariablen

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

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: $U_a \oplus V_b = c$) mit großer Wahrscheinlichkeit eintritt. In diesem Fall lässt sich nämlich der Wert von V_b bei Kenntnis von U_a entsprechend gut vorhersagen. Wegen $\Pr[U_a \oplus V_b = c] = 1 - \Pr[U_a \oplus V_b = c \oplus 1]$ kommt es nur darauf an, wie stark die Wahrscheinlichkeit $\Pr[U_a \oplus V_b = 0]$ von $1/2$ abweicht. Die durch das Paar (a, b) beschriebene **lineare Approximation** $L = U_a \oplus V_b$ an die S-Box S ist also um so besser, je größer der Absolutbetrag $|\Pr[L = 0] - 1/2|$ ist.

Definition 98. *Für eine Zufallsvariable X mit Wertebereich $W(X) = \{0,1\}$ bezeichne $\varepsilon(X)$ den Wert $\varepsilon(X) = \Pr[X = 0] - 1/2$ (auch **Bias** von X genannt).*

Unter Benutzung dieser Notation lässt sich also die Güte einer linearen Approximation $U_a \oplus V_b$ an eine S-Box S durch den Absolutbetrag $|\varepsilon(U_a \oplus V_b)|$ ihres Bias-Wertes bemessen.

Beispiel 99. *Wir betrachten wieder die S-Box S aus Beispiel 97. Dann nimmt die Zufallsvariable $(U_1, \ldots, U_4, V_1, \ldots, V_4)$ die 16 Werte in folgender Tabelle jeweils 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, genügt es, die Anzahl $L(a, b)$ der Zeilen zu bestimmen, für die $U_a = V_b$ ist. Dann gilt $\Pr[U_a \oplus V_b = 0] = \Pr[U_a = V_b] = L(a, b)/16$ 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
4	8	10	8	6	6	4	6	8	8	6	8	10	10	4	10	8
								⋮								
B	8	12	8	4	12	8	12	8	8	8	8	8	8	8	8	8
								⋮								
F	8	6	4	6	6	8	10	8	8	6	12	6	6	8	10	8

◁

4.4 Lineare Kryptoanalyse eines SPN

Wir betrachten nun das SPN *SP* aus Beispiel 97 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 (diese sind wie K unbekannt, aber konstant). Das Ziel besteht zunächst einmal darin, eine lineare Approximation für die Abbildung $x \mapsto u^4$ zu finden, bei der nur die ersten vier Rundenschlüssel $K^1, \ldots, K^4$ benutzt werden (siehe Abbildung 4.2). Hierzu verwenden wir die beiden linearen Approximationen

$$T = U_1 \oplus U_3 \oplus U_4 \oplus V_2 \quad \text{und} \quad T' = U_2 \oplus V_2 \oplus V_4$$

an die S-Box S mit den Bias-Werten $\varepsilon(T) = (L(B,4) - 8)/16 = (12-8)/16 = 1/4$ und $\varepsilon(T') = (L(4,5) - 8)/16 = (4-8)/16 = -1/4$ (also $\Pr[T = 0] = \Pr[T' = 1] = 3/4$).

Konkret verwenden wir 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 T' für die drei S-Boxen S_2^2, S_2^3, S_4^3,

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

Nun schalten wir diese vier linearen Approximationen an die S-Boxen S_2^1, S_2^2, S_2^3 und S_4^3 zu einer linearen Approximation

$$L = \underbrace{X_5 \oplus X_7 \oplus X_8}_{X_a \text{ für } a=\texttt{0B00}} \oplus \underbrace{U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4}_{U_b^4 \text{ für } b=\texttt{0505}} = X_a \oplus U_b^4$$

an die Abbildung $x \mapsto u^4$ zusammen und erhalten für ein Bit $c \in \{0,1\}$ die Gleichung

$$X_a \oplus U_b^4 = T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus c \tag{4.1}$$

An dieser Stelle ergeben sich nun folgende drei Fragen.

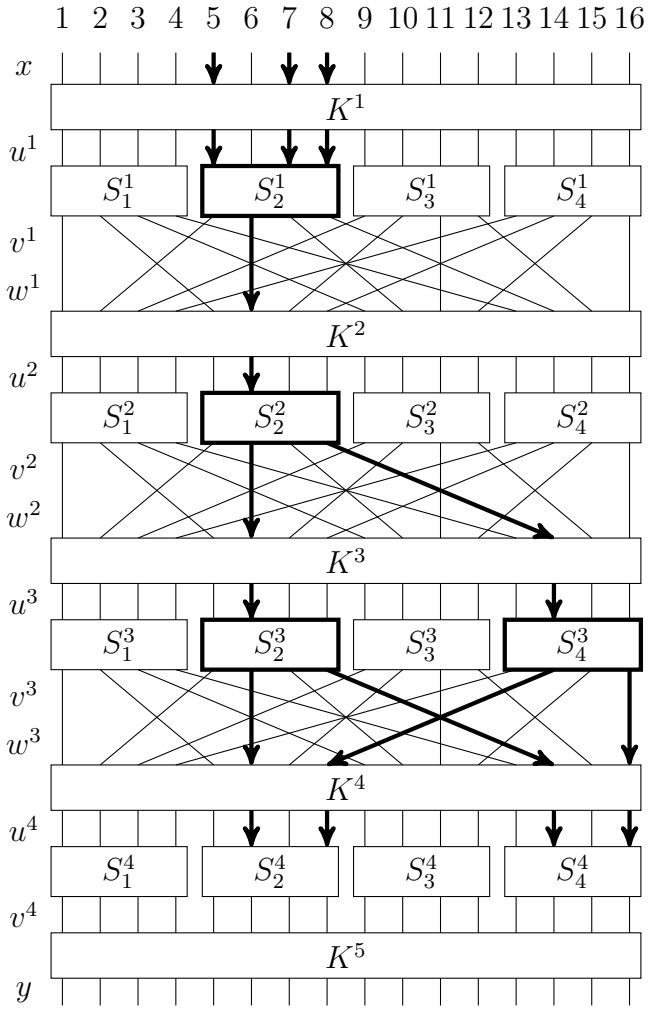


Abbildung 4.2: Eine lineare Approximation an ein Substitutions-Permutations-Netzwerk

1. Warum gilt (4.1)?
2. Wie gut ist die lineare Approximation L an die Abbildung $x \mapsto u^4$?
3. Wie können wir mit ihrer Hilfe einzelne Schlüsselbits bestimmen?

Die Antwort auf Frage 1 ist einfach: Seien $c_1, \ldots, c_4$ die Schlüsselbitsummen

$$c_1 = K_5^1 \oplus K_7^1 \oplus K_8^1, \quad c_2 = K_6^2, \quad c_3 = K_6^3 \oplus K_{14}^3, \quad c_4 = K_6^4 \oplus K_8^4 \oplus K_{14}^4 \oplus K_{16}^4$$

(für diese verwenden wir Kleinbuchstaben, da die einzelnen Schlüsselbits K_i^r als konstant

vorausgesetzt werden) und sei $c = c_1 \oplus c_2 \oplus c_3 \oplus c_4$. Dann gilt

$$\begin{aligned}
X_5 \oplus X_7 \oplus X_8 &= U_5^1 \oplus U_7^1 \oplus U_8^1 \oplus c_1 \\
&= T_1 \oplus V_6^1 \oplus c_1 \\
&= T_1 \oplus W_6^1 \oplus c_1 \\
&= T_1 \oplus U_6^2 \oplus c_1 \oplus c_2 \\
&= T_1 \oplus T_2 \oplus V_6^2 \oplus V_8^2 \oplus c_1 \oplus c_2 \\
&= T_1 \oplus T_2 \oplus W_6^2 \oplus W_{14}^2 \oplus c_1 \oplus c_2 \\
&= T_1 \oplus T_2 \oplus U_6^3 \oplus U_{14}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&= T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus V_6^3 \oplus V_8^3 \oplus V_{14}^3 \oplus V_{16}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&= T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus W_6^3 \oplus W_8^3 \oplus W_{14}^3 \oplus W_{16}^3 \oplus c_1 \oplus c_2 \oplus c_3 \\
&= T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \oplus \underbrace{c_1 \oplus c_2 \oplus c_3 \oplus c_4}_{c}
\end{aligned}$$

Nun zu Frage 2: Wären die Zufallsvariablen $T_1, \ldots, T_4$ unabhängig, so würde uns das folgende Piling-up-Lemma den Bias-Wert $2^3(1/4)(-1/4)^3 = -1/32$ für $\varepsilon(T_1 \oplus \cdots \oplus T_4)$ bzw. $(-1)^{c+1}/32$ für $\varepsilon(L)$ liefern. Sind nämlich X_1, X_2 unabhängige Zufallsvariablen mit Wertebereich $W(X_i) = \{0, 1\}$ und Bias $\varepsilon_i = \varepsilon(X_i)$, dann ist

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

und $\Pr[X_1 \oplus X_2 = 1] = 1/2 - 2\varepsilon_1\varepsilon_2$, d.h. es gilt $\varepsilon(X_1 \oplus X_2) = 2\varepsilon_1\varepsilon_2$. Diese Beobachtung lässt sich leicht verallgemeinern.

Lemma 100 (Piling-up Lemma).
Seien $X_1, \ldots, X_n$ unabhängige $\{0,1\}$-wertige Zufallsvariablen mit Bias $\varepsilon_i = \varepsilon(X_i)$. Dann gilt

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

Beweis. Wir führen den Beweis durch Induktion über n.

Induktionsanfang ($n = 1$): Klar.

Induktionsschritt ($n \rightsquigarrow n+1$): Nach Induktionsvoraussetzung hat die Zufallsvariable $Z = X_1 \oplus \cdots \oplus X_n$ den Bias $\varepsilon(Z) = 2^{n-1}\varepsilon(X_1) \cdots \varepsilon(X_n)$ und daher folgt

$$\varepsilon(X_1 \oplus \cdots \oplus X_{n+1}) = \varepsilon(Z \oplus X_{n+1}) = 2\varepsilon(Z)\varepsilon_{n+1} = 2^n\varepsilon_1 \cdots \varepsilon_{n+1}.$$ □

Beispiel 101. *Seien X_1, X_2, X_3 unabhängige Zufallsvariablen mit $\varepsilon(X_i) = 1/4$ für $i = 1, 2, 3$. Dann liefert das Piling-up Lemma die Bias-Werte $\varepsilon(X_i \oplus X_j) = 1/8$ für $1 \le i < j \le 3$. Man beachte, dass die Zufallsvariablen $Y = X_1 \oplus X_2$ und $Z = X_2 \oplus X_3$ nicht unabhängig sind und somit das Piling-up-Lemma in diesem Fall nicht anwendbar ist. Dieses würde nämlich für $Y \oplus Z$ einen Bias-Wert von $2(1/8)^2 = 1/32$ ergeben, wogegen*

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

und daher $\varepsilon(Y \oplus Z) = \varepsilon(X_1 \oplus X_3) = 1/8$ ist. ◁

Zwar sind die Zufallsvariablen T_i, aus denen eine lineare Approximation $X_a \oplus U_b^N = T_1 \oplus \cdots \oplus T_k \oplus c$ an die Abbildung $x \mapsto u^N$ gebildet wird, in der Regel nicht unabhängig. Dennoch zeigt sich in praktischen Anwendungen, dass der Bias-Wert $\varepsilon(T_1 \oplus \cdots \oplus T_k)$ von $T_1 \oplus \cdots \oplus T_k$ meist nicht zu sehr von dem "hypothetischen" Wert $2^{k-1} \prod_{i=1}^{k} \varepsilon(T_i)$ abweicht, welcher sich aus dem Piling-up Lemma ergeben würde. Daher können wir in unserem Beispiel

$$\varepsilon(T_1 \oplus \cdots \oplus T_4) \approx -1/32 \quad \text{bzw.} \quad \Pr[U^4_{0505} = X_{\texttt{0B00}}] \approx 1/2 + (-1)^{c+1}/32$$

annehmen.

Und nun zu Frage 3: Wir betrachten zuerst den (für den Gegner günstigen) Fall, dass anstelle von S eine S-Box benutzt wird, so dass die lineare Approximation L an die Abbildung $x \mapsto u^4$ den Bias-Wert ½ hat (d.h. wir nehmen an, dass alle Klartexte x auf ein Zwischenergebnis u^4 mit $x_a \oplus u^4_b = 0$ führen).

Sei $(x, y) \in M$ ein Klartext-Kryptotext-Paar, das mit dem gesuchten Schlüssel K erzeugt wurde. Dann können wir die Teilsumme $x_a = x_5 \oplus x_7 \oplus x_8$ berechnen. Da wir y und σ_S^{-1} kennen, können wir zudem für jeden Subschlüssel-Kandidaten (engl. *candidate subkey*) (I, J) für den Teilschlüssel $(K^5_{(2)}, K^5_{(4)})$ von K^5 aus dem Kryptotext y die zugehörigen u^4-Blöcke

$$u^4_{(2)}(I, J) = \sigma_S^{-1}(y_{(2)} \oplus I) \text{ und } u^4_{(4)}(I, J) = \sigma_S^{-1}(y_{(4)} \oplus J)$$

zurückrechnen (die beiden anderen Blöcke $u^4_{(1)}(I, J)$ und $u^4_{(3)}(I, J)$ werden für diesen Angriff nicht benötigt). Für den richtigen Kandidaten $(I, J) = (K^5_{(2)}, K^5_{(4)})$ fällt dann der Gleichheitstest

$$x_a = u^4_b(I, J) \tag{4.2}$$

für alle Paare $(x, y) \in M$ positiv aus. Dagegen besteht von den falschen Kandidaten $(I, J) \neq (K^5_{(2)}, K^5_{(4)})$ nur etwa die Hälfte diesen Test. Falls wir also alle Subkey-Kandidaten (I, J) dem Gleichheitstest (4.2) für eine hinreichend große Anzahl von Klartext-Kryptotext-Paaren (x, y) unterziehen, können wir den richtigen Kandidaten daran erkennen, dass er als einziger alle Tests besteht.

Im Fall, dass der Bias-Wert ε der linearen Approximation an die Abbildung $x \mapsto u^4$ zwar nicht gleich ½ ist, aber genügend weit von Null abweicht, besteht der richtige Kandidat $(I, J) = (K^5_{(2)}, K^5_{(4)})$ bei einer repräsentativen Auswahl M von Klartext-Kryptotext-Paaren ungefähr einen Anteil von $(½ + \varepsilon)$ der durchgeführten Tests, während die falschen Kandidaten etwa die Hälfte der Tests bestehen. Falls wir also eine hinreichende Anzahl von Klartext-Kryptotext-Paaren haben, können wir den richtigen Kandidaten nun daran erkennen, dass die Anzahl der von ihm bestandenen Tests am stärksten von $\|M\|/2$ abweicht.

Das Programmstück LinearAttack ermittelt für jeden Subkey-Kandidaten (I, J) die Anzahl $\alpha(I, J)$ der Klartext-Kryptotext-Paare $(x, y) \in M$ mit $x_{\texttt{0B00}} = u^4_b(I, J)$ und gibt den Kandidaten (I, J) aus, für den $\alpha(I, J)$ die stärkste Abweichung von $\|M\|/2$ aufweist.

Algorithmus LinearAttack

```
1  for (I,J) := (0,0) to (F,F) do
2    α(I,J) := 0
3  for each (x,y) ∈ M do
4    for (I,J) := (0,0) to (F,F) do
```

```
5        v^4_(2) := I ⊕ y_(2)
6        v^4_(4) := J ⊕ y_(4)
7        u^4_(2) := σ_S^{-1}(v^4_(2))
8        u^4_(4) := σ_S^{-1}(v^4_(4))
9        if x_5 ⊕ x_7 ⊕ x_8 ⊕ u^4_6 ⊕ u^4_8 ⊕ u^4_14 ⊕ u^4_16 = 0 then
10          α(I,J) := α(I,J) + 1
11   max := −1
12   for (I,J) := (0,0) to (F,F) do
13      β(I,J) := |α(I,J) − t/2|
14      if β(I,J) > max then
15         max := β(I,J)
16         maxkey := (I,J)
17   output(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 Beispiel reichen $t \approx 8000$ Paare, d.h. $c \approx 8$, da $\varepsilon^{-2} = 1024$ ist).

4.5 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 $x' = x \oplus x^*$ bilden.

Definition 102. *Seien $u, u^* \in \{0,1\}^l$ zwei Eingaben für eine S-Box $\sigma_S : \{0,1\}^l \to \{0,1\}^{l'}$ und seien $v = \sigma_S(u)$ und $v^* = \sigma_S(u^*)$ die zugehörigen Ausgaben. Dann wird $u' = u \oplus u^*$ die **Eingabedifferenz** (engl. input-xor) und $v' = \sigma_S(u) \oplus \sigma_S(u^*)$ die **Ausgabedifferenz** (engl. output-xor) des Paares (u, u^*) genannt. Für eine vorgegebene Eingabedifferenz $a' \in \{0,1\}^l$ sei weiter*

$$\Delta(a') = \{(u, u^*) \mid u, u^* \in \{0,1\}^l, u \oplus u^* = a'\} = \{(u, u \oplus a') \mid u \in \{0,1\}^l\}$$

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

Berechnen wir für alle Eingabepaare $(u, u^*) \in \Delta(a')$ die zugehörigen Ausgabedifferenzen, so verteilen sich diese auf die $2^{l'}$ möglichen Werte in $\{0,1\}^{l'}$. Man beachte, dass im Fall einer **affinen** S-Box $\sigma_S(u) = uA \oplus w$, wobei A eine binäre $(l \times l')$-Matrix und $w \in \{0,1\}^{l'}$ ist, alle Paare $(u, u^*) \in \Delta(a')$ auf dieselbe Ausgabedifferenz

$$\sigma_S(u) \oplus \sigma_S(u^*) = (u \oplus u^*)A = u'A = a'A$$

führen. Andernfalls kann die Eingabedifferenz a' zu unterschiedlichen Ausgabedifferenzen $\sigma_S(u) \oplus \sigma_S(u^*)$ führen, je nachdem, durch welches Eingabepaar $(u, u^*) \in \Delta(a')$ die Differenz a' realisiert wird. Um einer differentiellen Kryptoanalyse widerstehen zu können, sollten die Ausgabedifferenzen möglichst gleichmäßig verteilt sein.

Definition 103. *Sei $a' \in \{0,1\}^l$ eine Eingabe- und sei $b' \in \{0,1\}^{l'}$ eine Ausgabedifferenz für eine S-Box σ_S. Dann heißt (a', b') **Differential**. Die Anzahl der Eingabepaare (u, u^*),*

die die Eingabedifferenz a' in die Ausgabedifferenz b' überführen, bezeichnen wir mit $D(a', b')$, d.h.

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

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

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

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

$$\Pr[\underbrace{\sigma_S(U) \oplus \sigma_S(U^*)}_{V'} = b' \mid \underbrace{U \oplus U^*}_{U'} = a'],$$

dass zwei zufällig gewählte Eingaben U und U^* die Ausgabedifferenz $V' = b'$ erzeugen, wenn sie die Eingabedifferenz $U' = a'$ haben.

Beispiel 104. *Betrachten wir die S-Box $\sigma_S : \{0,1\}^4 \to \{0,1\}^4$ aus Beispiel 97, so erhalten wir für die Eingabedifferenz $a' = 1011$ die Menge*

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

von möglichen Eingabepaaren, die auf folgende Ausgabedifferenzen $v' = v \oplus v^* = \sigma_S(u) \oplus \sigma_S(u^*)$ führen:

u	u^*	v	v^*	v'
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

u	u^*	v	v^*	v'
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
⋮								⋮								
`B`	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
⋮								⋮								
`F`	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

◁

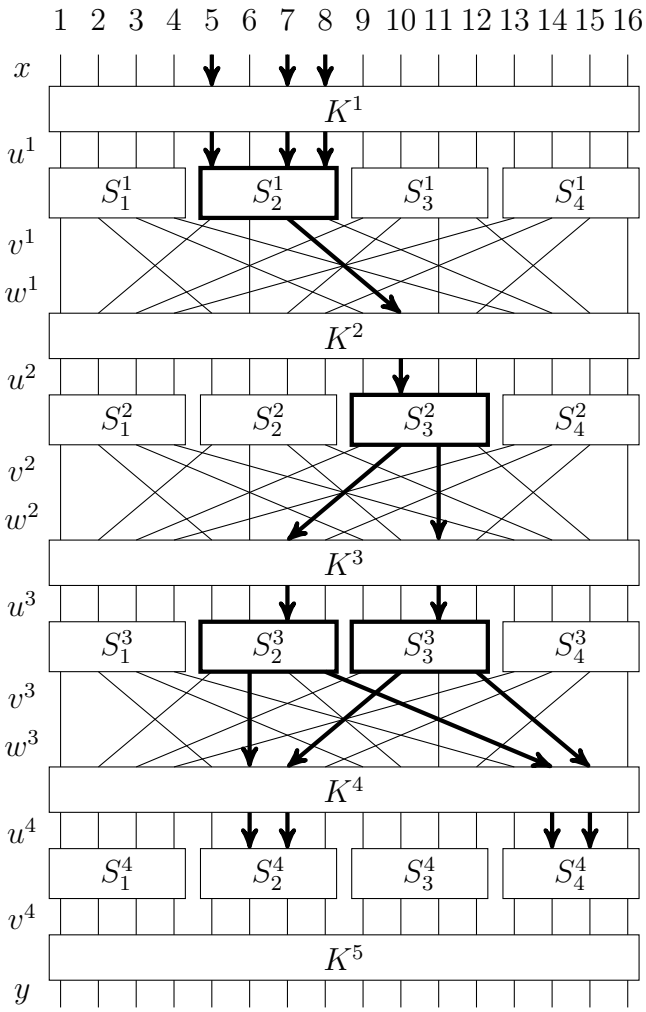


Abbildung 4.3: Eine Differentialspur für ein Substitutions-Permutations-Netzwerk

Können wir nun in einem SPN für bestimmte S-Boxen S_i^r Differentiale (a', b') finden, so dass die Eingabedifferenzen dieser Differentiale mit den (permutierten) Ausgabedifferenzen in der vorhergehenden Runde übereinstimmen (siehe Abbildung 4.3), so lassen sich diese Differentiale zu einer so genannten **Differentialspur** (engl. *differential trail*) für die Abbildung $x \mapsto u^4$ zusammensetzen. Unter der Annahme, dass die ausgewählten S-Boxen S_i^r (diese werden auch als **aktiv** bezeichnet) den zugeordneten Differentialen unabhängig voneinander folgen (oder nicht), lässt sich der Weitergabequotient der Spur als das Produkt der Weitergabequotienten der beteiligten Differentiale berechnen. Obwohl diese Annahme i.a. nicht zutrifft, weicht der tatsächliche Wert in praktischen Anwendungen kaum von diesem hypothetischen Wert ab.

Beispiel 105. *Betrachten wir das SPN SP aus Beispiel 97, so lassen sich folgende Differentiale zu einer Spur für die Abbildung $x \mapsto u^4$ kombinieren (siehe auch Abbildung 4.3):*

Für S_2^1: das Differential $(1011, 0010) = (\texttt{B,2})$ mit $Q(\texttt{B,2}) = 1/2$,

für S_3^2: das Differential $(0100, 0110) = (\texttt{4,6})$ mit $Q(\texttt{4,6}) = 3/8$ und

für S_2^3 und S_3^3: das Differential $(0010, 0101) = (\texttt{2,5})$ mit $Q(\texttt{2,5}) = 3/8$.

Gemäß dieser Spur führt also die Klartextdifferenz

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

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

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

welche wiederum mit Wahrscheinlichkeit 1 auf die Differenz

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

führt. Das Differential

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

für die Abbildung $x \mapsto u^4$ *hat also einen hypothetischen Weitergabequotienten von* $\varepsilon = Q(a', b') = 27/1024$. ◁

Sei nun (a', b') ein Differential für die Abbildung $x \mapsto u^4$ mit einem hypothetischen Weitergabequotienten $\varepsilon = Q(a', b')$. 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 vorgegebenen Differentialspur folgen und daher bei Verschlüsselung mit K Zwischenergebnisse u^4 und $(u^4)^*$ liefern, die die Differenz

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

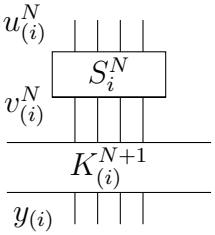
aufweisen. Doppelpaare mit dieser Eigenschaft werden **richtige Doppelpaare** (für das Differential (a', b')) genannt. Ein Großteil der falschen Doppelpaare lässt sich daran erkennen, dass die Kryptotext-Differenzen nicht die erwarteten 0^l-Blöcke aufweisen (im aktuellen Beispiel sind dies die Blöcke $y'_{(1)}$ und $y'_{(3)}$). Es empfiehlt sich, diese Doppelpaare auszufiltern, da sie (wie alle falschen Doppelpaare) nur „Hintergrundrauschen“ erzeugen und somit die Bestimmung des Schlüssels eher behindern.

Beobachtung 106. *Für die Ausgabe* $v^N_{(i)}$ *der S-Box* S^N_i *in Runde* 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*

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



Falls die S-Box S^N_i *nicht affin ist, hängt die aus den Kryptotextblöcken* $y_{(i)}$ *und* $(y_{(i)})^*$ *zurückgerechnete Eingabedifferenz*

$$(u^N_{(i)})' = u^N_{(i)} \oplus (u^N_{(i)})^* = \sigma_S^{-1}(y_{(i)} \oplus K^{N+1}_{(i)}) \oplus \sigma_S^{-1}((y_{(i)})^* \oplus K^{N+1}_{(i)})$$

von dem Schlüsselblock $K_{(i)}^{N+1}$ *ab. Ist also* (x, x^*, y, y^*) *ein richtiges Doppelpaar, so sind neben den Kryptotextblöcken* $y_{(i)}$ *und* $y^*_{(i)}$ *auch die Eingabedifferenzen* $b'_{(i)} = (u^N_{(i)})'$ *von* S_i^N *bekannt. Folglich kommen nur solche Subkey-Werte* I *für* $K_{(i)}^{N+1}$ *infrage, für die*

$$\sigma_S^{-1}(y_{(i)} \oplus I) \oplus \sigma_S^{-1}(y^*_{(i)} \oplus I) = b'_{(i)} \tag{4.3}$$

ist. Erfüllt I *Gleichung (4.3), so sagen wir auch,* I *ist mit dem Doppelpaar* (x, x^*, y, y^*) ***konsistent***.

Gemäß Beobachtung 106 kann jedes richtige Doppelpaar dazu benutzt werden, einige Kandidaten für den Rundenschlüsselblock $K_{(i)}^{N+1}$ auszuschließen. Ist M hinreichend groß, so wird sich schließlich der richtige Schlüsselblock als derjenige herausstellen, der mit den meisten Doppelpaaren konsistent ist. Wir benutzen nun die Spur aus Beispiel 105 für einen Angriff mittels differentieller Analyse.

Beispiel 107. *Der Algorithmus* DifferentialAttack *bestimmt für jeden Subschlüssel- Kandidaten* (I, J) *für* $(K^5_{(2)}, K^5_{(4)})$ *die Anzahl* $\gamma(I, J)$ *aller Doppelpaare* (x, x^*, y, y^*) *in* M*, die mit* (I, J) *konsistent sind und (in Zeile 3) nicht als falsch erkannt werden. Ausgegeben wird der Kandidat* (I, J) *mit dem größten* γ*-Wert.* ◁

Algorithmus DifferentialAttack

```
1  for (I,J) := (0,0) to (F,F) do γ(I,J) := 0
2  for each (x,x*,y,y*) ∈ M do
3    if y_(1) = y*_(1) und y_(3) = y*_(3) then
4      for (I,J) := (0,0) to (F,F) do
5        v^4_(2) := I ⊕ y_(2)
6        v^4_(4) := J ⊕ y_(4)
7        u^4_(2) := σ_S^{-1}(v^4_(2))
8        u^4_(4) := σ_S^{-1}(v^4_(4))
9        (v^4_(2))* := I ⊕ y*_(2)
10       (v^4_(4))* := J ⊕ y*_(4)
11       (u^4_(2))* := σ_S^{-1}((v^4_(2))*)
12       (u^4_(4))* := σ_S^{-1}((v^4_(4))*)
13       (u^4_(2))' := u^4_(2) ⊕ (u^4_(2))*
14       (u^4_(4))' := u^4_(4) ⊕ (u^4_(4))*
15       if (u^4_(2))' = 0110 und (u^4_(4))' = 0110 then γ(I,J) := γ(I,J) + 1
16 max := −1
17 for (I,J) := (0,0) to (F,F) do
18   if γ(I,J) > max then
19     max := γ(I,J)
20     maxkey := (I,J)
21 output(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 Beispiel reichen $t \approx 80$ Doppelpaare, wobei $\varepsilon^{-1} \approx 38$ ist, d.h. $c \approx 2$).

5 DES und AES

5.1 Der Data Encryption Standard (DES)

5.1.1 Geschichte des 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 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 und im Mai 2002 wurde DES von AES als Standard abgelöst. Allerdings wurde Triple DES (auch TDES oder 3DES genannt) vom NIST als Standard bis 2030 fortgeschrieben.

Der DES ist eine Feistel-Chiffre mit $N = 16$ Runden. Die Rundenfunktion g einer **Feistel-Chiffre** berechnet das Zwischenergebnis $w^r = g(K^r, w^{r-1}) \in \{0,1\}^\ell$ in Runde r aus den beiden Hälften L^{r-1} und R^{r-1} von $w^{r-1} \in \{0,1\}^\ell$ gemäß der Vorschrift

$$L^r = R^{r-1} \text{ und } R^r = L^{r-1} \oplus f(R^{r-1}, K^r),$$

wobei $f : \{0,1\}^{\ell/2} \times \{0,1\}^{k'} \to \{0,1\}^{\ell/2}$ eine beliebige Funktion und k' die Länge der Rundenschlüssel $K^1, \ldots, K^N$ ist (siehe auch Abb. 5.1). Aus der Ausgabe $w^N = L^N R^N$ in Runde N wird durch Vertauschung von L^N und R^N der Kryptotext $y = R^N L^N$ gebildet.

5.1.2 Aufbau der DES-Chiffrierfunktion.

Die Blocklänge des DES beträgt $\ell = 64$ Bit und die (effektive) Schlüssellänge ist 56 Bit. Der 56 Bit Schlüssel K^- ergibt zusammen mit 8 Paritätsbits (Bits 8, 16,. . . , 64) einen ebenfalls $k = 64$ Bit langen Schlüsselblock K. Es gibt somit $2^{56} \approx 7.2 \cdot 10^{16}$ verschiedene Schlüssel. Bei Eingabe von K und x führt der DES-Algorithmus nacheinander die folgenden Chiffrierschritte aus:

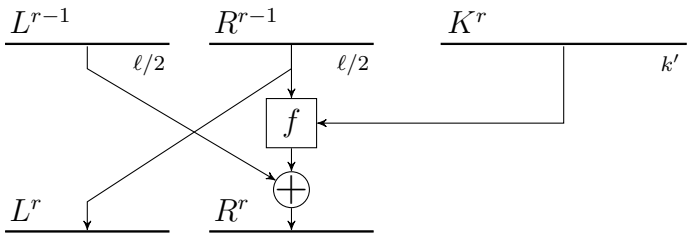


Abbildung 5.1: Graphische Darstellung der Rundenfunktion g einer Feistel-Chiffre

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

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

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

Abbildung 5.2: Initialpermutation IP, Expansion E und Permutation P

1. Zuerst wird der Klartext x einer Initialpermutation $IP\colon x_1x_2\cdots x_{64} \mapsto x_{58}x_{50}\cdots x_7$ (siehe Abb. 5.2; der Werteverlauf von IP ist also zeilenweise dargestellt) unterzogen.
2. Danach erfolgen 16 Runden mit einer Feistel-Rundenfunktion g und sechzehn Rundenschlüsseln $K^1, \ldots, K^{16}$ (die Berechnung der Schlüssel K^r aus K wird weiter unten beschrieben). Die Rundenfunktion g basiert auf der in Abb. 5.3 dargestellten Funktion $f : \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$, die wie folgt berechnet wird.

   Die Eingabe von f ist (R^{r-1}, K^r) (vgl. Abb. 5.1). Zuerst wird der 32-Bit Block R^{r-1} mittels der Expansionsabbildung E (siehe Abb. 5.2) auf einen 48-Bit Block $E(R^{r-1})$ erweitert. Auf diesen wird bitweise der Rundenschlüssel K^r addiert. Als Ergebnis erhalten wir den 48-Bit Block $B = E(R^{r-1}) \oplus K^r$. Dieser wird in acht 6-Bit Blöcke $B = B_{(1)}, \ldots, B_{(8)}$ aufgeteilt, die mit den acht S-Boxen $S_1, \ldots, S_8$ auf acht 4-Bit Blöcke $C_{(i)} = S_i(B_{(i)})$ verkleinert werden. Die Tabellen der S-Boxen S_i sind in Abb. 5.4 dargestellt. Aus diesen erhält man die Werte $S_i(B_{(i)})$ wie folgt:

   > Ist $B_{(i)} = b_1 \cdots b_6$, so findet man $S_i(B_{(i)})$ in Zeile b_1b_6 und Spalte $b_2b_3b_4b_5$ (in Hexadezimaldarstellung) der Tabelle für S_i. Zum Beispiel ist $S_1(011010) = 1001$, da in Zeile $(00)_2 = 0$ und Spalte $(1101)_2 = D$ die die Hexadezimalziffer $9 = (1001)_2$ steht.

   Die Konkatenation der von den acht S-Boxen berechneten 4-Bit Blöcke ergibt einen 32-Bit Block $C = C_{(1)} \ldots C_{(8)}$, welcher noch der Permutation P (siehe Abb. 5.2) unterworfen wird.
3. Aus dem nach der 16. Runde ausgegebenen 64-Bit Block $w^{16} = L^{16}R^{16}$ wird durch Vertauschen der beiden Hälften und Anwendung der inversen Initialpermutation IP^{-1} der Kryptotext $\mathsf{DES}(K, x) = IP^{-1}(R^{16}L^{16})$ gebildet.

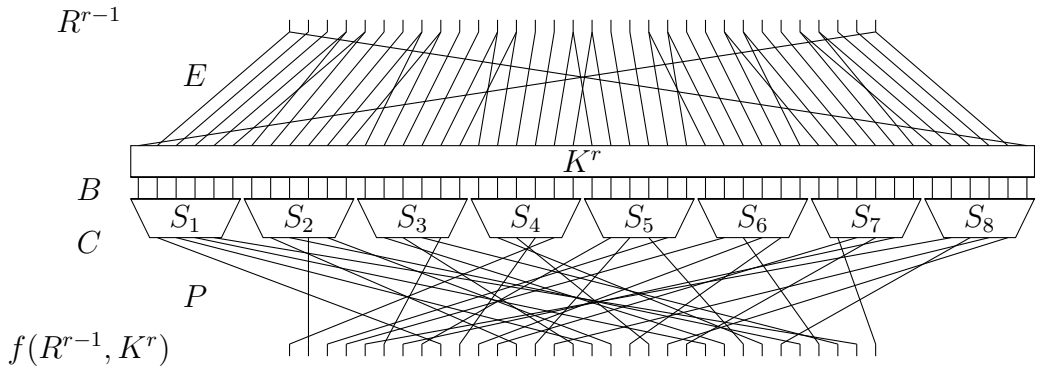


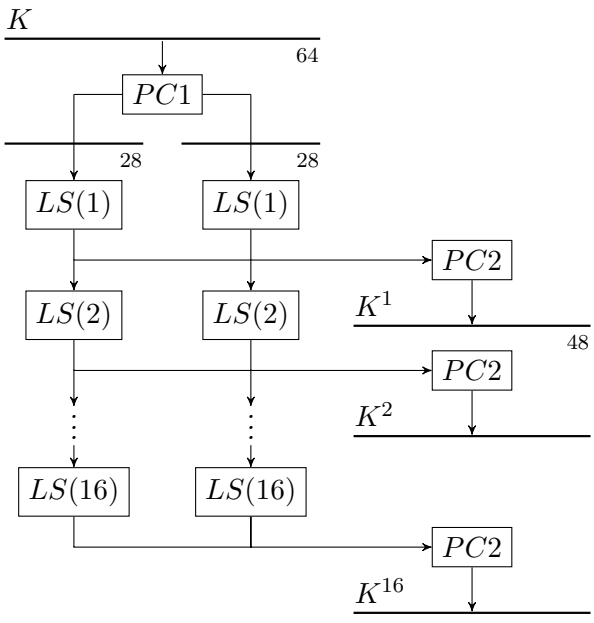
Abbildung 5.3: Graphische Darstellung der DES-Funktion f

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S_1:	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D
S_2:	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9
S_3:	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C
S_4:	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E
S_5:	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3
S_6:	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D
S_7:	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C
S_8:	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

Abbildung 5.4: Tabellarische Darstellung der acht im DES benutzten S-Boxen

5.1.3 Der DES Key-Schedule Algorithmus

Der Key-Schedule Algorithmus des DES berechnet aus dem externen 64-Bit Schlüssel K wie folgt die zugehörigen 16 Rundenschlüssel $K^1, \ldots, K^{16}$ (siehe Abb. 5.5). Zuerst wählt die Funktion $PC\,1$ (permuted choice 1) aus dem Schlüssel K die kryptografisch relevanten Bits aus und permutiert sie. Das erhaltene Ergebnis wird in zwei 28-Bit Blöcke unterteilt. Diese beiden Blöcke werden dann in 16 Runden $r = 1, \ldots, 16$ jeweils zyklisch um $LS(r) \in \{1, 2\}$ Bit (siehe Abb. 5.5) verschoben.
Aus den beiden Blöcken nach Runde r bestimmt die Funktion $PC\,2$ (permuted choice 2) jeweils den Rundenschlüssel K^r durch Entfernen der 8 Bits an den Stellen 9, 18, 22, 25, 35, 38, 43 und 56 sowie einer Permutation der verbleibenden 48 Bits.



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

r	1	2	3	4	5	6	7	8
$LS(r)$	1	1	2	2	2	2	2	2
r	9	10	11	12	13	14	15	16
$LS(r)$	1	2	2	2	2	2	2	1

Abbildung 5.5: Der Key-Schedule Algorithmus des DES

Definition 108. *Ein DES-Schlüssel K heißt **schwach**, falls alle durch ihn erzeugten Rundenschlüssel gleich sind (d.h. es gilt $\{K^1,\ldots,K^{16}\} = \{K^1\}$).*

Der DES hat die vier schwachen Schlüssel (hexadezimal)

K	erzeugter Rundenschlüssel
0101010101010101	000000000000
1F1F1F1F0E0E0E0E	000000111111
E0E0E0E0F1F1F1F1	111111000000
FEFEFEFEFEFEFEFE	111111111111

Für jeden von ihnen gilt $\mathsf{DES}(K, \mathsf{DES}(K, x)) = x$ (siehe Übungen). Neben diesen vier schwachen Schlüsseln existieren noch sechs weitere sogenannte „semischwache“ Schlüsselpaare (K, K') mit der Eigenschaft $\mathsf{DES}(K', \mathsf{DES}(K, x)) = x$ (siehe Übungen).

5.1.4 Eigenschaften von DES.

Der DES konnte sich nicht sofort nach seiner Veröffentlichung im Jahre 1975 durchsetzen. Er wurde anfangs von manchen Behörden und Banken in den USA nicht verwendet, da folgende Sicherheitsbedenken gegen ihn geäußert wurden:

- Die 56-Bit Schlüssellänge bietet eventuell eine zu geringe Sicherheit gegen einen Brute-Force Angriff bei bekanntem oder wählbarem 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. Es gibt mittlerweile sogar kommerzielle Angebote im Internet (z.B. `crack.sh`), innerhalb von 26 Stunden eine vollständige Schlüsselsuche bei bekanntem Klartext auf spezieller Hardware auszuführen, um alle passenden DES-Schlüssel zu finden.

Als Vorbereitung zum AES-Algorithmus gehen wir im nächsten Abschnitt kurz auf die Arithmetik in endlichen Körpern ein. Diese spielt beim AES eine sehr wichtige Rolle.

5.2 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 109. *Sei R ein Ring.*

- *Der **Polynomring** $R[x]$ enthält für alle $n \geq 0$ alle Polynome $p(x)$ in der Variablen x mit Koeffizienten in R, d.h. $p(x)$ hat die Form*

$$p(x) = a_n x^n + \cdots a_1 x + a_0 \quad mit \quad a_0, \ldots, a_n \in R$$

  *Man sagt, $R[x]$ entsteht aus R durch **Adjunktion der Variablen** x.*
- *Der **Grad** von p (bezeichnet mit $\mathsf{deg}(p)$) ist im Fall $a_n \neq 0$ gleich n und im Fall $n = a_n = 0$ gleich -1.*
- *Ein Polynom $q(x)$ **teilt** ein Polynom $p(x)$ (kurz: $q(x)|p(x)$), falls ein Polynom $d(x) \in R[x]$ existiert mit $p(x) = d(x)q(x)$. Teilt $q(x)$ die Differenz $f(x) - g(x)$ zweier Polynome, so schreiben wir hierfür*

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

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

$$p(x) \bmod q(x)$$

  *das bei der Polynomdivision von $p(x)$ durch $q(x)$ auftretende **Restpolynom**, also dasjenige Polynom $r(x)$ vom Grad $\mathsf{deg}(r) < \mathsf{deg}(q)$, für das ein Polynom $d(x) \in R[x]$ existiert mit $p(x) = d(x)q(x) + r(x)$.*

Man überprüft leicht, dass $R[x]$ mit der üblichen Polynomaddition und Polynommultiplikation tatsächlich einen Ring bildet. Ähnlich wie beim Übergang von $\mathbb{Z}$ zum Restklassenring $\mathbb{Z}_m$ können wir für ein fest gewähltes Polynom $m(x)$ vom Grad $\mathsf{deg}(m) = n$ jedem Polynom $p(x) \in \mathbb{Z}_m[x]$ mittels

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

eindeutig ein Polynom vom Grad höchstens $n-1$ zuordnen. Auf diese Weise erhalten wir den endlichen Polynomring (genauer **Faktorring**) $\mathbb{Z}_m[x]/m(x)$ aller Polynome vom Grad höchstens $n-1$, wobei die Addition und Multiplikation wie in $\mathbb{Z}_m[x]$, gefolgt von einer Reduktion modulo $m(x)$, definiert ist. In den Übungen wird gezeigt, dass $\mathbb{Z}_m[x]/m(x)$ genau dann ein Körper ist, wenn m prim ist und $m(x)$ nur triviale Teiler besitzt.

Definition 110. *Ein Polynom $m(x) \in \mathbb{Z}_p[x]$, p prim, vom Grad $n \geq 1$ heißt **irreduzibel** (über $\mathbb{Z}_p$), falls keine Polynome $p(x), q(x) \in \mathbb{Z}_p[x]$ vom Grad $\mathsf{deg}(p), \mathsf{deg}(q) \geq 1$ existieren mit*

$$m(x) = p(x)q(x).$$

Satz 111. *Der Faktorring $\mathbb{Z}_m[x]/m(x)$ ist genau dann ein Körper, wenn m prim und $m(x)$ in $\mathbb{Z}_m[x]$ irreduzibel ist.*

Beweis. Siehe Übungen. □

Zudem kann man zeigen, dass für primes p und jede Zahl $n \geq 1$ ein irreduzibles Polynom $m(x) = x^n + \sum_{i=0}^{n-1} m_i x^i$ vom Grad n in $\mathbb{Z}_p[x]$ existiert. Daher lässt sich für jede Primzahlpotenz p^n ein Körper $\mathbb{Z}_p[x]/m(x)$ der Größe p^n konstruieren. Tatsächlich gibt es bis auf Isomorphie nur einen Körper mit p^n Elementen, den wir mit $\mathbb{F}_{p^n}$ bezeichnen.

Wir können Polynome $a(x) = \sum_{i=0}^{n-1} a_i x^i$ auch als Koeffizientenvektoren $\vec{a} = (a_{n-1}, \ldots, a_0)$ darstellen. 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 $\mathbb{F}_{p^n}$ entspricht dann der üblichen Vektoraddition (also komponentenweisen Addition modulo p), d.h. die Vektordarstellung $\vec{c}$ von $c(x) = a(x) + b(x)$ ist

$$(c_{n-1}, \ldots, c_0) = (a_{n-1} + b_{n-1}, \ldots, a_0 + b_0).$$

Im Fall $p = 2$ ist dies also die bitweise Addition modulo 2 (xor). Die Multiplikation in $\mathbb{Z}_p[x]/m(x)$ lässt sich wegen

$$a(x)b(x) = \sum_{i=0}^{n-1} a_i x^i b(x)$$

auf die Addition und (iterierte) Multiplikation mit dem Polynom $p(x) = x$ zurückführen. Da $m(x)$ die Form $m(x) = \sum_{i=0}^{n} m_i x^i$ mit $m_n = 1$ hat, gilt für diese

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

wobei wir $b_{-1} = 0$ setzen. Die Multiplikation von $b(x)$ mit x entspricht somit einem Linksshift von $\vec{b}$ um eine Stelle, dem sich im Fall $b_{n-1} \neq 0$ noch die Subtraktion des Vektors $(b_{n-1}m_{n-1}, \ldots, b_{n-1}m_0)$ anschließt. Im Fall $p = 2$ erhalten wir also

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

und die Vektordarstellung $\vec{c}$ von $c(x) = xb(x)$ ist

$$(c_{n-1}, \ldots, c_0) = \begin{cases} (b_{n-2}, \ldots, b_0, 0), & b_{n-1} = 0, \\ (b_{n-2}, \ldots, b_0, 0) \oplus (m_{n-1}, \ldots, m_0), & b_{n-1} = 1. \end{cases}$$

Beispiel 112. *Sei $p = 2$ und $n = 3$. Zunächst benötigen wir ein irreduzibles Polynom $m(x) \in \mathbb{Z}_2[x]$ vom Grad 3. Setzen wir*

$$m(x) = x^3 + a_2x^2 + a_1x + a_0$$

so sehen wir, dass $m(x)$ im Fall $a_0 = 0$ den nichttrivialen Teiler $p(x) = x$ hat. Daher 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

$$x^3+1=(x+1)(x^2+x+1) \quad \textit{und} \quad x^3+x^2+x+1=(x+1)(x^2+1)$$

sowie

$$x^3+x+1=(x+1)(x^2+x)+1 \quad \textit{und} \quad x^3+x^2+1=(x+1)x^2+1$$

ist, gibt es in $\mathbb{Z}_2[x]$ *nur zwei irreduzible Polynome vom Grad* 3*, nämlich* x^3+x+1 *und* x^3+x^2+1 *(da sie weder* x *noch* $x+1$ *als Teiler haben).*
Nehmen wir $m(x)=x^3+x+1$*, so gilt in* $\mathbb{Z}_2[x]/m(x)$ *bspw. wegen* $1+1=0$ *die Gleichung*

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

und wegen

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

die Gleichung $(x^2+1)(x+1)=x^2$*.* ◁

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

Beispiel 113. *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 in* $\mathbb{Z}_2[x]$*. Dann können wir mit dem euklidischen Algorithmus den (in Bezug auf den Grad) größten gemeinsamen Teiler* $g(x)$ *von* $m(x)$ *und* $a(x)$ *wie folgt berechnen:*

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

Es ist also $g(x)=r_4(x)=1$*. Der erweiterte euklidische Algorithmus berechnet nun Polynome* $p_i(x)$ *und* $q_i(x)$ *gemäß der Vorschrift*

$$p_i(x)=p_{i-2}(x)-d_i(x)\cdot p_{i-1}(x), \ \textit{wobei}\ p_0(x)=1 \ \textit{und}\ p_1(x)=0,$$

und

$$q_i(x)=q_{i-2}(x)-d_i(x)\cdot q_{i-1}(x), \ \textit{wobei}\ q_0(x)=0 \ \textit{und}\ q_1(x)=1,$$

welche die Gleichung $p_i(x)m(x)+q_i(x)a(x)=r_i(x)$ *erfüllen. Im Fall* $r_i(x)=1$ *ist also* $q_i(x)$ *das multiplikative Inverse von* $a(x)$ *modulo* $m(x)$*.*

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

Aus der letzten Zeile können wir das multiplikative Inverse $a^{-1}(x)=q_4(x)=x^7+x^6+x^3+x$ *von* $a(x)$ *modulo* $m(x)$ *ablesen.* ◁

5.3 Der Advanced Encryption Standard (AES)

5.3.1 Geschichte des 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, 160, 192, 224 oder 256 Bit gewählt werden. Die Rundenzahl N des Rijndael hängt wie folgt von der Blocklänge ℓ und der gewählten Schlüssellänge k ab:

	k				
ℓ	128	160	192	224	256
128	**10**	11	**12**	13	**14**
160	11	11	12	13	14
192	12	12	12	13	14
224	13	13	13	13	14
256	14	14	14	14	14

Beim AES-Standard wurde die Blocklänge auf 128 Bit fixiert und die Schlüssellänge auf die Werte 128, 192 oder 256 Bit beschränkt. Wir beschränken uns hier auf die Beschreibung des 10-Runden AES mit $\ell = 128$ Bit Blocklänge und $k = 128$ Bit Schlüssellänge.

5.3.2 Die AES S-Box SubByte

Die Elemente $a(x) = \sum_{i=0}^{7} a_i x^i$ des Körpers $\mathbb{F}_{2^8} = \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1)$ können jeweils durch ein Byte $a_7 \dots a_0$ dargestellt werden. Zur expliziten Konvertierung dieser beiden Darstellungen verwenden wir die Funktionen FieldToBinary und BinaryToField, die wie folgt definiert sind:

- BinaryToField: $\{0,1\}^8 \to \mathbb{F}_{2^8}$ überführt die Byte-Darstellung $a_7 \dots a_0$ in das zugehörige Polynom $a(x) = \sum_{i=0}^{7} a_i x^i$.
- FieldToBinary: $\mathbb{F}_{2^8} \to \{0,1\}^8$ ist die Umkehrfunktion von BinaryToField.

Sowohl der Key-Schedule Algoritmus als auch die AES-Chiffrierfunktion verwenden eine S-Box SubByte. Diese benutzt als nicht-linearen Bestandteil die Funktion

$$\mathsf{FieldInv} : \mathbb{F}_{2^8}^* \to \mathbb{F}_{2^8}^*,$$

die das multiplikative Inverse aller Einheiten des Körpers $\mathbb{F}_{2^8}$ berechnet. Konkret wird SubByte durch folgenden Algorithmus berechnet.

SubByte$(a_7 \cdots a_0)$

```
1  input  a_7 ··· a_0
2  z := BinaryToField(a_7 ··· a_0)
3  if  z ≠ 0  then  z := FieldInv(z)
4  a_7 ··· a_0 := FieldToBinary(z)
5  c_7 ··· c_0 := 01100011
6  for  i := 0  to  7  do
7     b_i := a_i ⊕ a_{i+4} ⊕ a_{i+5} ⊕ a_{i+6} ⊕ a_{i+7} ⊕ c_i
8  output  b_7 ··· b_0
```

Die Indexrechnung in Zeile 7 erfolgt modulo 8. Die Zeilen 5-8 realisieren eine affine Hill-Chiffrierfunktion.

Beispiel 114.

- *Bei Eingabe* 01010011 *liefert die Funktion* BinaryToField *in Zeile 2 das zugehörige Polynom* $z = \mathsf{BinaryToField}(01010011) = x^6 + x^4 + x + 1$.
- *Die Funktion* FieldInv *berechnet das multiplikative Inverse* $\mathsf{FieldInv}(z) = z^{-1} = x^7 + x^6 + x^3 + x$ *von* z *in* $\mathbb{F}_{2^8}$ *(siehe Beispiel 113).*
- *Die Funktion* FieldToBinary *liefert die zugehörige Koeffizienten-Darstellung* $a_7 \ldots a_0 = \mathsf{FieldToBinary}(x^7 + x^6 + x^3 + x) = 11001010$.
- *Es folgt die Berechnung der Ausgabe* $b_7 \ldots b_0$ *mittels*

$$b_7 \ldots b_0 = 11001010 \begin{pmatrix} 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{pmatrix} \oplus 01100011 = 11101101$$

- *Somit ist* $\mathsf{SubByte}(01010011) = 11101101$ *oder hexadezimal:* $\mathsf{SubByte}(\mathsf{53}) = \mathsf{ED}$. ◁

Wir können die AES S-Box SubByte in Form einer (16×16)-Matrix angeben, wobei der Eintrag in Zeile X und Spalte Y den Wert $\mathsf{SubByte}(XY)$ enthält:

	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

5.3.3 Der AES Key-Schedule Algorithmus

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. Bei der Berechnung der Rundenschlüssel werden Wort-Konstanten $RCon[1], \ldots, RCon[10]$ mit $RCon[i] = \mathsf{FieldToBinary}(x^{i-1})0^{24} \in \{0,1\}^{32}$ benutzt. In Hexadezimal-Darstellung ergeben sich folgende Werte:

i	1	2	3	4	5
$RCon[i]$	01000000	02000000	04000000	08000000	10000000
i	6	7	8	9	10
$RCon[i]$	20000000	40000000	80000000	$1B000000$	36000000

Reihen wir die 11 Rundenschlüssel $K^0, \ldots, K^{10}$ aneinander, so entsteht ein Array $K^0 \ldots K^{10} = w[0] \ldots w[43]$ von 44 Worten $w[i]$, die gemäß folgendem Algorithmus aus dem 128-Bit Schlüssel K berechnet werden.

$\mathsf{KeyExpansion}(K)$

```
1  input  K = K[0] ··· K[15]
2  for  i := 0  to  3  do
3     w[i] := K[4i]K[4i + 1]K[4i + 2]K[4i + 3]
4  for  i := 4  to  43  do
5     temp := w[i − 1]
6     if  i ≡4 0  then  temp := SubWord(RotWord(temp)) ⊕ RCon[i/4]
7     w[i] := w[i − 4] ⊕ temp
8  output  w[0] ... w[43]
```

Die hierbei benutzten Funktionen sind wie folgt definiert:

- RotWord: eine 32-Bit Transposition, die die 4 Eingabebytes zyklisch um ein Byte nach links verschiebt: $\mathsf{RotWord}(B_0B_1B_2B_3) = B_1B_2B_3B_0$.
- SubWord: eine 32-Bit Substitution, die durch 4 parallel geschaltete SubByte S-Boxen realisiert wird.

5.3.4 Der AES Chiffrieralgorithmus

Unter Benutzung der 11 Rundenschlüssel $K^0, \ldots, K^{10}$ wird der 128 Bit Klartextblock wie folgt chiffriert:

Chiffrierfunktion $\mathsf{AES}(K, x)$

```
1  AddRoundKey(K^0)
2  for r := 1 to 9 do
3     SubBytes
4     ShiftRows
5     MixColumns
6     AddRoundKey(K^r)
7  SubBytes
8  ShiftRows
9  AddRoundKey(K^10)
```

Im einzelnen werden also die folgenden Chiffrierschritte ausgeführt:

- Zuerst wird der Klartextblock x einer Addition mit dem 128-Bit Rundenschlüssel K^0 unterworfen. Diese Operation wird mit $\mathsf{AddRoundKey}(K^0)$ bezeichnet.
- Danach werden 9 Runden ausgeführt, wobei in jeder Runde r der Reihe nach folgende Operationen ausgeführt werden:
  - SubBytes: eine nichtlineare 128-Bit Substitution, die durch 16 parallel geschaltete S-Boxen SubByte realisiert wird,
  - ShiftRows: eine 128-Bit Transposition (siehe unten)
  - MixColumns: eine lineare 128-Bit Substitution (siehe unten) und
  - $\mathsf{AddRoundKey}(K^r)$: eine Schlüsseladdition mit dem Rundenschlüssel K^r.
- Es folgt Runde 10 mit den Operationen SubBytes, ShiftRows und $\mathsf{AddRoundKey}(K^{10})$.

Abgesehen von der zusätzlichen linearen Substitution MixColumns entspricht der Aufbau des AES also exakt dem in Abschnitt 4.2 beschriebenen Aufbau eines SPNs.

5.3.5 Die AES Transposition ShiftRows

Um die beiden Operationen ShiftRows und MixColumns zu beschreiben, stellen wir den Klartext $x = x_0 \cdots x_{15}$, $x_i \in \{0,1\}^8$, und alle daraus berechneten Zwischenergebnisse in Form einer Matrix $M \in \mathbb{F}_{2^8}^{(4\times 4)}$ dar. Diese wird wie folgt initialisiert:

$$\begin{matrix} x_0 & x_4 & x_8 & x_{12} \\ x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \end{matrix}$$

Die Operation ShiftRows ist eine 128-Bit Transposition, die wie folgt definiert ist:

$$\mathsf{ShiftRows}: \begin{matrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{matrix} \mapsto \begin{matrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{matrix}$$

5.3.6 Die AES S-Box MixColumn

Die Operation MixColumns basiert auf einer linearen 32-Bit S-Box MixColumn, die parallel auf den vier Spalten des aktuellen Zwischenergebnisses ausgeführt wird. Bei ihrer Berechnung wird die Multiplikation FieldMult: $\mathbb{F}_{2^8} \times \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ im Körper $\mathbb{F}_{2^8}$ benutzt.

MixColumn$(s_{3,j}, s_{2,j}, s_{1,j}, s_{0,j})$

1 **for** $i := 0$ **to** 3 **do** $t_i :=$ BinaryToField(s_{ij})
2 $u_0 :=$ FieldMult$(x, t_0) +$ FieldMult$(x+1, t_1) + t_2 + t_3$
3 $u_1 :=$ FieldMult$(x, t_1) +$ FieldMult$(x+1, t_2) + t_3 + t_0$
4 $u_2 :=$ FieldMult$(x, t_2) +$ FieldMult$(x+1, t_3) + t_0 + t_1$
5 $u_3 :=$ FieldMult$(x, t_3) +$ FieldMult$(x+1, t_0) + t_1 + t_2$
6 **for** $i := 0$ **to** 3 **do** $s'_{ij} :=$ FieldToBinary(u_i)
7 **output** $(s'_{3,j}, s'_{2,j}, s'_{1,j}, s'_{0,j})$

Die Operation MixColumn führt also eine lineare Transformation in dem Vektorraum $(\mathbb{F}_{2^8})^4$ aus, die sich auch wie folgt beschreiben lässt (hierbei repräsentieren wir die Elemente des Körpers $\mathbb{F}_{2^8}$ als Koeffizientenvektoren in Hexadezimaldarstellung, d.h. `03` steht für $x+1$ usw.):

$$\mathsf{MixColumn} : (c_3, \ldots, c_0) \mapsto (c_3, \ldots, c_0) \underbrace{\begin{pmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{01} \\ \texttt{01} & \texttt{02} & \texttt{03} & \texttt{01} \\ \texttt{01} & \texttt{01} & \texttt{02} & \texttt{03} \\ \texttt{03} & \texttt{01} & \texttt{01} & \texttt{02} \end{pmatrix}}_{Z} = (c'_3, \ldots, c'_0).$$

Die S-Box MixColumn realisiert somit eine lineare 32-Bit Substitution $c \mapsto cZ$ auf dem Vektorraum $(\mathbb{F}_{2^8})^4$. Zudem ist jede Zeile von Z relativ zur darüber liegenden Zeile um eine Position zyklisch nach rechts verschoben. Aufgrund dieser speziellen Form von Z lässt sich MixColumn im Faktorring $R = \mathbb{F}_{2^8}[y]/(y^4+1)$ als multiplikative Chifrierfunktion $c(y) \mapsto z(y)c(y)$ beschreiben. Stellen wir nämlich einen Vektor $(c_3, c_2, c_1, c_0) \in (\mathbb{F}_{2^8})^4$ als ein Polynom $c(y) = \sum_{i=0}^{3} c_i y^i$ in R dar und wählen wir für $z(y)$ das Polynom $z(y) = \texttt{03}y^3 + \texttt{01}y^2 + \texttt{01}y + \texttt{02}$ in R, so gilt (siehe Übungen)

$$\mathsf{MixColumn}(c(y)) = z(y)c(y).$$

Der Ring $\mathbb{F}_{2^8}[y]/(y^4+1)$ ist zwar kein Körper, da das Polynom y^4+1 in $\mathbb{F}_{2^8}[y]$ nicht irreduzibel ist (siehe Übungen). Da aber die Matrix Z invertierbar ist, kann die inverse Abbildung $\mathsf{MixColumn}^{-1}$ von MixColumn mittels $c \mapsto cZ^{-1}$ berechnet werden. Somit hat auch das Polynom $z(y)$ im Ring $\mathbb{F}_{2^8}[y]/(y^4+1)$ ein multiplikatives Inverses $z^{-1}(y)$.

5.3.7 Kryptoanalytische Betrachtungen

Bis heute konnten keine Schwachstellen gefunden werden, d.h. alle bekannten Angriffe gegen den AES bringen keinen signifikanten Vorteil gegenüber einer vollständigen Schlüsselsuche. Die Tatsache, dass für die S-Box SubByte die Inversenbildung in einem endlichen Körper benutzt wird, führt dazu, dass die Tabellen für die Güte der linearen Approximationen und für die Weitergabequotienten der Differenzenpaare einen hohen Grad an Uniformität aufweisen. Dadurch wird die S-Box resistent gegen lineare und differentielle Analysen. Zudem verhindert die lineare Substitution MixColumns lineare

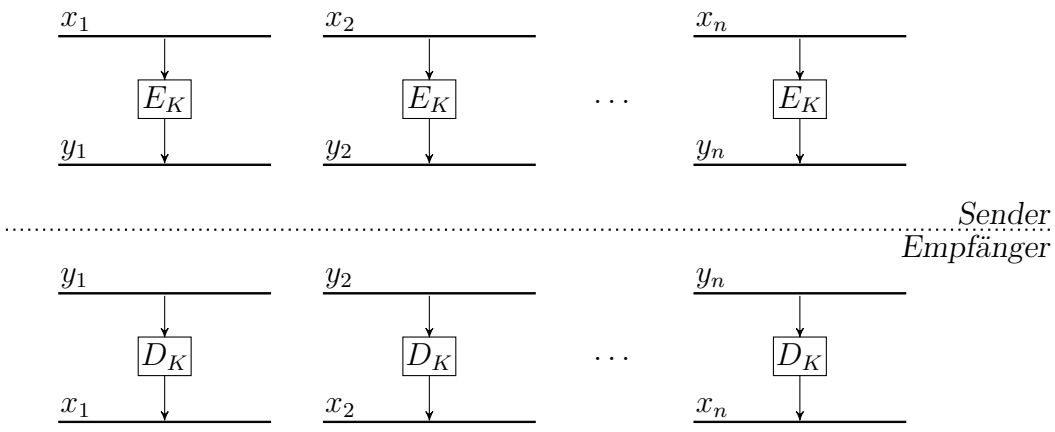
und differentielle Angriffe mit nur wenigen aktiven S-Boxen (diese Technik wird von den AES-Entwicklern als *wide trail strategy* bezeichnet).

5.4 Betriebsarten von Blockchiffren

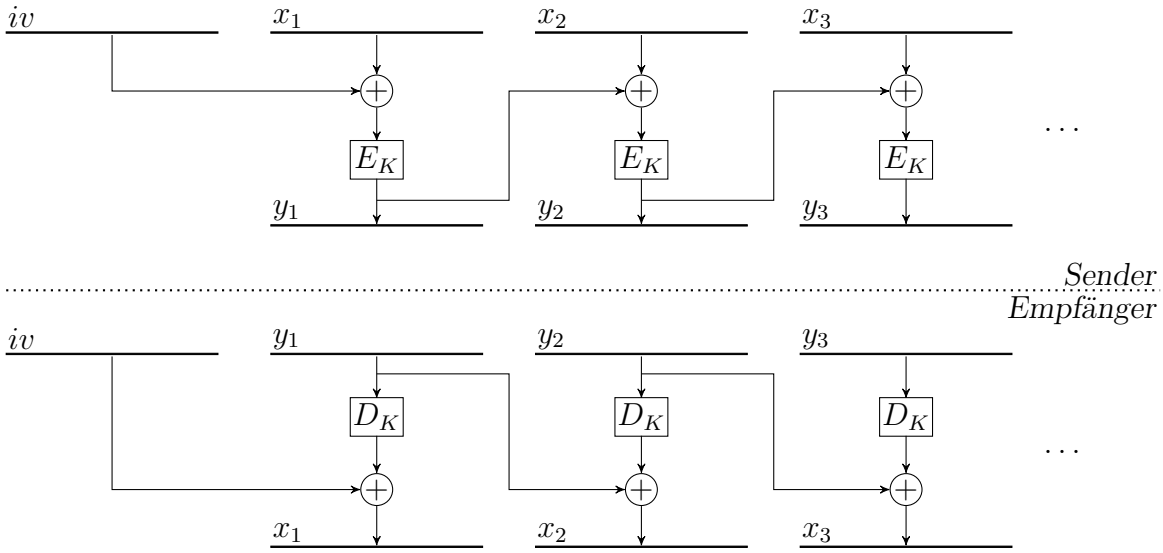
Für den DES wurden vier verschiedene Betriebsarten vorgeschlagen, in denen grundsätzlich jede Blockchiffre E mit beliebiger Blocklänge ℓ betrieben werden kann. Bei den ersten beiden Betriebsarten (ECB und CBC) werden Kryptotextblöcke der Länge ℓ übertragen. Mit einer Blockchiffre kann aber auch ein Stromsystem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

Der ECB-Modus ist die naheliegendste Betriebsart, wird aber in der Praxis so gut wie nie benutzt, da sie eine Reihe von Angriffsmöglichkeiten bietet (vgl. z.B. Abschnitt 3.5).

ECB-Modus (electronic codebook; elektronisches Codebuch). Die Binärnachricht x wird in Klartextblöcke x_i der Länge ℓ zerlegt. Der letzte Block x_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.



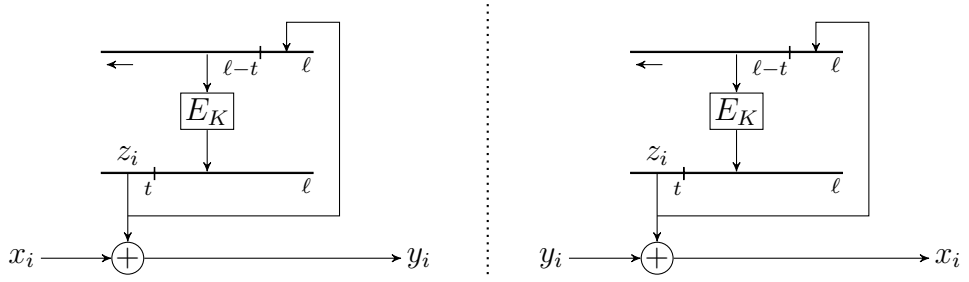
CBC-Modus (cipher block chaining; Blockverkettung des Schlüsseltextes). Um zu verhindern, dass ein Eindringling den Kryptotext verändert, ohne dass der Empfänger dies bemerkt, wird beim CBC-Modus jeder Kryptotextblock nicht nur von dem zugehörigen Klartextblock, sondern auch von allen vorausgehenden Blöcken abhängig gemacht. Dies hat auch zur Folge, dass gleiche Klartextblöcke auf unterschiedliche Kryptotextblöcke abgebildet werden.



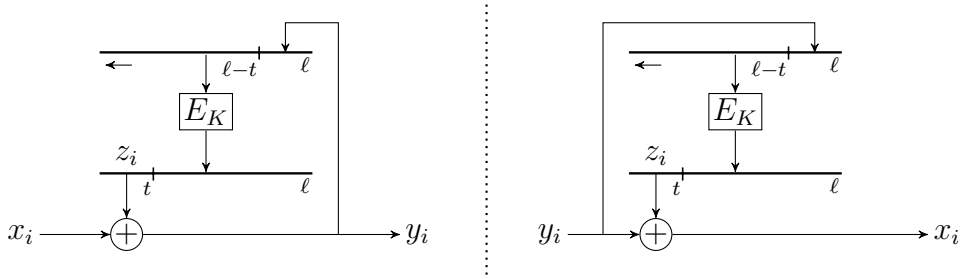
Jeder Klartextblock x_i wird mit dem Kryptotextblock $E_K(x_{i-1})$ bitweise (modulo 2) addiert, bevor er verschlüsselt wird (zur Verschlüsselung von x_1 wird ein Initialisierungsvektor iv verwendet. Der Initialisierungsvektor wird üblicherweise unverschlüsselt übertragen, sollte aber nicht mehrmals verwendet werden. Er wird meist durch einen Pseudozufallsgenerator erzeugt.

Der CBC-Modus verhindert auch, dass sich bestimmte Klartextmuster direkt auf den Kryptotext übertragen (was beim ECB-Modus der Fall ist). Ein extremes Beispiel ist die Verschlüsselung einer Graphikdatei x, deren Pixel durch ℓ-Bit Blöcke kodiert sind. Zwar werden dann beim ECB-Modus die Pixel substituiert, aber ansonsten stellt der Kryptotext y exakt dieselbe Graphik wie der Klartext x dar. Dadurch wird eine „visuelle Entschlüsselung“ durch Betrachten der verschlüsselten Graphikdatei y ermöglicht.

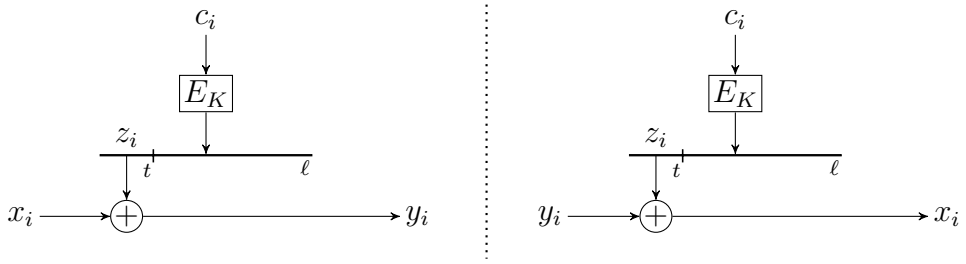
OFB-Modus (output feedback; Rückführung der Ausgabe). Die Binärnachricht x wird in t-Bit Blöcke (für festes t: $1 \leq t \leq \ell$) zerlegt. Die Chiffrierfunktion E_K dient zur Erzeugung einer pseudozufälligen Folge von t-Bit Blöcken z_i, die bitweise (modulo 2) auf die entsprechenden Klartextblöcke x_i addiert werden. Als Eingabe für die Chiffrierfunktion E_K dient ein Schieberegister, das anfangs mit einem Initialisierungsvektor iv geladen wird. Zur Verschlüsselung jedes t-Bit Klartextblockes x_i erzeugt die Chiffrierfunktion E_K zunächst einen Ausgabevektor, von dem nur die ersten t Bits verwendet werden. Dieser t-Bit Block z_i dient sowohl zur Berechnung des Kryptoxtblocks $y_i = x_i \oplus z_i$, als auch zur Modifikation des Eingaberegisters von E_K, in das sie von rechts geschoben werden.



CFB-Modus (cipher feedback; Rückführung des Kryptotextes). Ähnlich zum OFB-Modus, nur dass zur Erneuerung des Eingaberegisters nicht die ersten t Bits z_i der E_K-Ausgabe, sondern der zugehörige t-Bit Kryptotextblock $y_i = x_i \oplus z_i$ verwendet wird.



CTR-Modus (counter mode). Eine weitere Variante des OFB-Modus' ist der **CTR-Modus**, bei dem die Pseudozufallsfolge mit Hilfe von E_K aus einer fortlaufenden Binärblockfolge $c_0, c_1, \ldots$ mit $c_{i+1} = c_i + 1 \bmod 2^\ell$ erzeugt wird. Dies hat den Vorteil, dass spätere Blöcke der Pseudozufallsfolge nicht von den vorhergehenden abhängen, und daher mehrere Blöcke $E_K(c_i)$ parallel berechnet werden können.



6 Zahlentheoretische Grundlagen

In diesem Abschnitt stellen wir die Hilfsmittel aus der Zahlentheorie bereit, die wir zum Verständnis der Public-Key Verfahren benötigen, die im nächsten Abschnitt vorgestellt werden.

Nehmen wir ein beliebiges Element a einer endlichen Gruppe G und betrachten die Folge der Potenzen $a^0 = 1$, $a^1 = a$, a^2, a^3, ..., so stellt sich die Frage, ob es einen Exponenten $n \geq 1$ mit $a^n = 1$ gibt. In den Übungen werden wir sehen, dass das so ist und bis dahin alle Potenzen paarweise verschieden sind.

Definition 115. *Sei G eine endliche Gruppe. Die **Ordnung von** G ist die Anzahl $\|G\|$ ihrer Elemente. Die **Ordnung eines Elements** $a \in G$ ist*

$$\mathrm{ord}_G(a) = \min\{n \geq 1 \mid a^n = 1\}.$$

Im Fall $G = \mathbb{Z}_m^*$ schreiben wir auch einfach $\mathrm{ord}_m(a)$ anstelle von $\mathrm{ord}_{\mathbb{Z}_m^*}(a)$. Die von a in G erzeugte Untergruppe $\{a^n \mid n \geq 0\} = \{a^0, \ldots, a^{\mathrm{ord}_G(a)-1}\}$ bezeichnen wir mit $\langle a \rangle_G$ oder mit $\langle a \rangle$, wenn G aus dem Kontext ersichtlich ist.

Für beliebige ganze Zahlen $i, j \in \mathbb{Z}$ gilt

$$a^i = a^j \Leftrightarrow i \equiv_{\mathrm{ord}(a)} j$$

(siehe Übungen). Da $\mathrm{ord}(a) = \|\langle a \rangle\|$ die Ordnung einer Untergruppe von G ist, muss $\mathrm{ord}(a)$ ein Teiler der Gruppenordnung $\|G\|$ sein (Satz von Lagrange).

Beispiel 116. *Wir betrachten die Gruppe $G = \mathbb{Z}_7^*$. Die folgende Tabelle zeigt für jedes Element $a \in G$ die von a erzeugte Untergruppe $\langle a \rangle$ sowie dessen Ordnung $\mathrm{ord}_G(a) = \|\langle a \rangle\|$:*

a	1	2	3	4	5	6
$\langle a \rangle$	$\{1\}$	$\{1,2,4\}$	$\{1,3,2,6,4,5\}$	$\{1,4,2\}$	$\{1,5,4,6,2,3\}$	$\{1,6\}$
$\mathrm{ord}_G(a)$	1	3	6	3	6	2

◁

Satz 117 (Euler-Fermat)**.** *In jeder endlichen Gruppe $(G, \cdot, 1)$ der Ordnung $\|G\| = m$ gilt $a^m = 1$ für alle $a \in G$.*

Beweis. Wir betrachten hier nur den Fall, dass G kommutativ ist. Der allgemeine Fall wird in den Übungen bewiesen.

Sei also $G = \{b_1, \ldots, b_m\}$ abelsch und sei $a \in G$ beliebig. Wegen $ab_i \neq ab_j$ für $i \neq j$ folgt $G = \{ab_1, \ldots, ab_m\}$. Dies impliziert $\prod_{i=1}^m b_i = \prod_{i=1}^m ab_i = a^m \prod_{i=1}^m b_i$. Also muss $a^m = 1$ sein. □

Korollar 118 (Kleiner Satz von Fermat)**.** *Für jede Primzahl p und jede Zahl a mit $a \not\equiv_p 0$ gilt $a^{p-1} \equiv_p 1$.*

6.1 Diskrete Logarithmen

Für ein beliebiges Gruppenelement $a \in G$ ist die Exponentiation $\exp_{G,a} : x \mapsto a^x$ in G zur Basis a eine Bijektion zwischen der Menge $\mathbb{Z}_{ord(a)} = \{0, 1, \ldots, ord(a) - 1\}$ und und der Untergruppe $\langle a \rangle$. Die zugehörige Umkehrabbildung spielt in der Kryptografie eine wichtige Rolle.

Definition 119. *Seien $a, b \in G$ mit $b \in \langle a \rangle$. Dann heißt der eindeutig bestimmte Exponent $x \in \mathbb{Z}_{\mathrm{ord}(a)}$ mit $a^x = b$* ***Index*** *oder* ***diskreter Logarithmus von b zur Basis a in G****, kurz*

$$x = \log_{G,a}(b).$$

Im Fall $G = \mathbb{Z}_m^*$ schreiben wir auch einfach $x = \log_{m,a}(b)$ anstelle von $\log_{\mathbb{Z}_m^*,a}(b)$.

Während die diskrete Exponentialfunktion $\exp_{m,a} : x \mapsto a^x$ für alle $m \geq 2$ durch **wiederholtes Quadrieren und Multiplizieren** (siehe nächsten Abschnitt) effizient berechenbar ist, sind bis heute keine effizienten Verfahren zur Berechnung von $\log_{m,a}(b)$ bekannt (falls die Parameter a und m geeignet gewählt werden).

Beispiel 120. *Das Element $a = 2$ hat in der Gruppe $G = \mathbb{Z}_{11}^*$ die maximal mögliche Ordnung $\mathrm{ord}_{11}(2) = \|G\| = 10$. Die folgenden Tabellen zeigen den Werteverlauf der Funktionen $\exp_{11,2}$ und $\log_{11,2}$.*

x	0	1	2	3	4	5	6	7	8	9
2^x	1	2	4	8	5	10	9	7	3	6

b	1	2	3	4	5	6	7	8	9	10
$\log_{11,2}(b)$	0	1	8	2	4	9	7	3	6	5

◁

6.2 Zyklische Gruppen

Für manche Anwendungen sind Elemente $a \in G$ nützlich, mit denen sich die gesamte Gruppe erzeugen lässt.

Definition 121 (**Primitivwurzel/Erzeuger**)**.** *Sei G eine endliche Gruppe der Ordnung $\|G\| = m$. Ein Element $g \in G$ der Ordnung $\mathrm{ord}_G(g) = m$ heißt* ***Erzeuger*** *von G. G heißt* ***zyklisch****, falls G mindestens einen Erzeuger besitzt.*

Ein Element $a \in G$ ist also genau dann ein Erzeuger, wenn die von a erzeugte Untergruppe $\langle a \rangle$ die gesamte Gruppe G umfasst.

Satz 122 (Gauß)**.** *Genau für $m \in \{1, 2, 4, p^k, 2p^k \mid 2 < p \text{ prim}\}$ ist die Gruppe $\mathbb{Z}_m^*$ zyklisch (ohne Beweis).*

Das folgende Lemma benötigen wir für den Beweis des nächsten Satzes, der eine Charakterisierung von zyklischen Gruppen erlaubt.

Lemma 123 (**Euler**)**.** *Für alle $m \geq 1$ gilt*

$$\sum_{d|m} \varphi(d) = m,$$

wobei die Summe über alle Teiler $d \geq 1$ von m läuft.

Beweis. Für jeden Teiler $d \geq 1$ von m sei $T_d := \{a \in \mathbb{Z}_m \mid \mathrm{ggT}(a, m) = d\}$. Dann folgen wegen

$$\varphi(m/d) = \|\{b \in \mathbb{Z}_{m/d} \mid \underbrace{\mathrm{ggT}(b, m/d) = 1}_{\Leftrightarrow\, \mathrm{ggT}(bd, m) = d}\}\| = \|T_d\|$$

die Gleichungen

$$\sum_{d|m} \varphi(d) = \sum_{d|m} \varphi(m/d) = \sum_{d|m} \|T_d\| = \|\mathbb{Z}_m\| = m$$

□

Wir zeigen nun, dass G genau dann zyklisch ist, wenn jede Gleichung der Form $x^n = 1$ höchstens n verschiedene Lösungen in G hat.

Satz 124. *Eine endliche Gruppe G der Ordnung $\|G\| = m$ ist genau dann zyklisch, wenn jede Gleichung der Form $x^n = 1$ $(1 \leq n \leq m)$ höchstens n verschiedene Lösungen $a \in G$ hat. In diesem Fall hat G genau $\varphi(m)$ Erzeuger.*

Beweis. Falls G zyklisch und a ein Erzeuger von G ist, so ist $G = \{a^k \mid k \in \mathbb{Z}_m\}$. Eine Potenz $a^k \in G$ ist genau dann eine Lösung von $x^n = 1$, wenn $a^{kn} = 1$ ist. Sei $g = \mathrm{ggT}(n, m)$ und seien $n' = n/g$ sowie $m' = m/g$. Da $\mathrm{ggT}(n', m') = 1$ ist, existiert ein Inverses $(n')^{-1}$ von n' modulo m'. Wegen

$$a^{kn} = 1 \Leftrightarrow kn \equiv_m 0 \Leftrightarrow kn' \equiv_{m'} 0 \Leftrightarrow kn'(n')^{-1} \equiv_{m'} 0 \Leftrightarrow k \equiv_{m'} 0.$$

hat also $x^n = 1$ nur die $g \leq n$ Lösungen $a^k = a^{jm'}$, $j = 0, 1, \ldots, g-1$.
Für die Rückrichtung betrachten wir für jeden Teiler d von m die Menge

$$S_d = \{a \in G \mid \mathrm{ord}(a) = d\}$$

aller Elemente der Ordnung d in G. Es ist klar, dass jedes Element $a \in S_d$ eine Lösung von $x^d = 1$ ist, d.h. es gilt $S_d \subseteq \{x \in G \mid x^d = 1\}$. Daher kann S_d nach Voraussetzung nicht mehr als d Elemente enthalten. Wir zeigen, dass die Größe von S_d sogar durch $\varphi(d)$ beschränkt ist. Da jedes Element $a \in S_d$ eine Untergruppe $\langle a \rangle$ der Größe d erzeugt, folgt nach dem Satz von Euler-Fermat für jedes $a \in S_d$ die Inklusion $\langle a \rangle \subseteq \{x \in G \mid x^d = 1\}$, die sogar eine Gleichheit ist, da $x^d = 1$ nicht mehr als d Lösungen hat:

$$\forall a \in S_d : S_d \subseteq \langle a \rangle.$$

Zudem hat a^i genau dann die Ordnung $\mathrm{ord}(a^i) = \mathrm{ord}(a) = d$, wenn $\mathrm{ggT}(i, d) = 1$ ist (siehe Übungen), was für jedes $a \in S_d$ die Gleichheit $S_d = \{a^i \mid i \in \mathbb{Z}_d^*\}$ und somit $\|S_d\| \leq \varphi(d)$ impliziert.
Da die Mengen S_d, $d|m$, eine Partition von G bilden, folgt $\sum_{d|m} \|S_d\| = m$. Andererseits gilt $\sum_{d|m} \varphi(d) = m$ nach Lemma 123, woraus

$$\sum_{d|m} \|S_d\| = \sum_{d|m} \varphi(d)$$

folgt. Da aber, wie gerade gezeigt, S_d entweder 0 oder $\varphi(d)$ Elemente enthält, muss S_d für jedes d genau $\varphi(d)$ und insbesondere S_m genau $\varphi(m)$ Elemente enthalten. □

In den Übungen wird gezeigt, dass die Gleichung $x^n = 1$ in einem Körper höchstens n verschiedene Lösungen hat. Daher ist die Einheitengruppen $\mathbb{F}_q^*$ jedes endlichen Körpers $\mathbb{F}_q$ zyklisch und hat genau $\varphi(q-1)$ Erzeuger. Insbesondere sind auch die Gruppen $\mathbb{Z}_p^*$, p prim, zyklisch (vgl. Satz 122).

Sofern die Primfaktorzerlegung der Gruppenordnung m bekannt ist, lässt sich effizient überprüfen, ob ein Gruppenelement $a \in G$ ein Erzeuger ist oder nicht.

Satz 125. *Sei G eine endliche Gruppe der Ordnung $\|G\| = m$. Ein Element $a \in G$ ist genau dann ein Erzeuger, wenn für jeden Primteiler p von m gilt:*

$$a^{m/p} \neq 1.$$

Beweis. Falls a ein Erzeuger von G ist, so gilt $a^e \neq 1$ für alle Exponenten $e \in \{1, \ldots, m-1\}$ und somit auch für alle Exponenten e der Form m/p, p prim.

Ist dagegen $a \in G$ kein Erzeuger, so ist $\operatorname{ord}(a) < m$, und da $\operatorname{ord}(a)$ ein Teiler von m ist, existiert eine Zahl $d \geq 2$ mit $d \cdot \operatorname{ord}(a) = m$. Sei p ein beliebiger Primteiler von d. Dann gilt

$$a^{m/p} = a^{d \operatorname{ord}(a)/p} = (a^{\operatorname{ord}(a)})^{d/p} = 1. \qquad \square$$

Der folgende probabilistische Algorithmus ComputeGenerator berechnet einen Erzeuger a für eine zyklische Gruppe G, falls alle Primteiler p von $m = \|G\|$ bekannt sind und sich die Elemente von G zufällig generieren lassen.

ComputeGenerator$(G, p_1, \ldots, p_k)$

```
1 input zyklische Gruppe G und alle Primteiler p_1, ..., p_k von m = ||G||
2 repeat
3   guess randomly a ∈ G
4 until a^{m/p_i} ≠ 1 für alle i = 1, ..., k
5 output a
```

Da $\varphi(m) \geq m/(2 \ln \ln m)$ für hinreichend große m gilt, findet der Algorithmus in jedem Schleifendurchlauf mit Wahrscheinlichkeit $\varphi(m)/m \geq 1/(2 \ln \ln m)$ einen Erzeuger. Die erwartete Anzahl der Schleifendurchläufe ist also $O(\ln \ln m)$.

6.3 Effiziente Berechnung von Potenzen

Falls sich in einer Halbgruppe oder einem Ring das Produkt zweier Elemente effizient berechnen lässt, sind auch die Potenzen a^e durch **wiederholtes Quadrieren und Multiplizieren** effizient berechenbar. Hierzu sind maximal $2\lceil \log e \rceil$ Multiplikationen erforderlich.

Pot(a, e)

```
1 x := a; y := a^{e_0}
2 for i := 1 to r do
3   x := x^2; y := y · x^{e_i}
4 return(y)
```

HornerPot(a, e)

```
1 z := a
2 for i := r − 1 downto 0 do
3   z := z^2 · a^{e_i}
4 return(z)
```

Sei $e = \sum_{i=0}^{r} e_i \cdot 2^i$ mit $r = \lfloor \log_2 e \rfloor$ die Binärdarstellung von e. Dann können wir den Exponenten e sukzessive mittels $b_0 = e_0$ und $b_i = b_{i-1} + e_i 2^i = \sum_{j=0}^{i} e_j \cdot 2^j$ für $i = 1, \ldots, r$ zu $b_r = e$ berechnen. Der Algorithmus Pot berechnet nach diesem Schema in der Variablen y die Potenzen a^{b_i} für $i = 0, \ldots, r$.

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 $c_i = \sum_{j=i}^{r} e_j \cdot 2^{j-i}$, also $c_0 = \sum_{j=0}^{r} e_j \cdot 2^j = e$. Dies führt auf den Algorithmus HornerPot, der in der Variablen z die Potenzen a^{c_i} für $i = r, \ldots, 0$ berechnet.

Beispiel 126. *Wir berechnen die Potenz a^e für $a = 1920$ und $e = 19$ im Ring $\mathbb{Z}_{2773}$:*

Pot$(1920, 19)$					*HornerPot*$(1920, 19)$				
i	e_i	b_i	$x_i = a^{2^i}$	$y_i = a^{b_i}$	i	e_i	c_i	$z_i = a^{c_i}$
0	1	1	$1920^1 = 1920$	$1920^1 = 1920$	4	1	1	$1920^1 = 1920$
1	1	3	$1920^2 = 1083$	$1920 \cdot 1083^1 = 2383$	3	0	2	$1920^2 \cdot 1920^0 = 1083$
2	0	3	$1083^2 = 2683$	$2383 \cdot 2683^0 = 2383$	2	0	4	$1083^2 \cdot 1920^0 = 2683$
3	0	3	$2683^2 = 2554$	$2383 \cdot 2554^0 = 2383$	1	1	9	$2683^2 \cdot 1920^1 = 1016$
4	1	19	$2554^2 = 820$	$2383 \cdot 820^1 = \mathbf{1868}$	0	1	19	$1016^2 \cdot 1920^1 = \mathbf{1868}$

◁

6.4 Der Primzahlsatz

Bezeichne $\mathcal{P}$ die Menge der Primzahlen und sei π die Funktion, die jeder Teilmenge $A \subseteq \mathbb{N}$ die Anzahl

$$\pi(A) = \|A \cap \mathcal{P}\|$$

der Primzahlen in der Menge A zuweist. Zudem bezeichnen wir die Zahl $\pi([1, n])$ auch einfach mit $\pi(n)$ und für $c \in \mathbb{Z}_m$ sei

$$\pi_{c,m}(n) = \pi(\{p \in \mathcal{P} \mid p \equiv_m c\}).$$

Satz 127. (Hadamard und de la Vallée Poussin, 1896)
Es gilt $\pi(n) \sim n/\ln n$ und für $c \in \mathbb{Z}_m^*$ gilt $\pi_{c,m}(n) \sim n/(\varphi(m) \ln n)$.

Hierbei bedeutet $f(n) \sim g(n)$, dass die beiden Funktionen f und g asymptotisch äquivalent sind (d.h. es gilt $\lim_{n\to\infty} f(n)/g(n) = 1$). Wie folgende Tabelle zeigt, liefert die Funktion $Li(n) = \int_2^n (\ln x)^{-1} dx$ im Vergleich zu $n/\ln n$ eine deutlich bessere Abschätzung von $\pi(n)$.

n	$\pi(n)$	$\pi(n) - n/\ln n$	$Li(n) - \pi(n)$
10	4	-0.3	2.2
100	25	3.3	5.1
1 000	168	23	10
10 000	1 229	143	17
10 100	1 240	144	18
10^6	78 498	6 116	130
10^9	50 847 534	2 592 592	1 701
10^{12}	37 607 912 018	1 416 705 193	38 263
10^{15}	29 844 570 422 669	891 604 962 452	1 052 619
10^{18}	24 739 954 287 740 860	612 483 070 893 536	21 949 555
10^{21}	21 127 269 486 018 731 928	446 579 871 578 168 707	597 394 254

Beispiel 128. *Verwenden wir die Abschätzung $\pi(n) \approx \int_2^n (\ln x)^{-1} dx$, so erhalten wir für die Anzahl $\pi([a,\, b])$ der Primzahlen in einem Intervall $[a,\, b]$ den Näherungswert*

$$\pi([a,b]) \approx \int_a^b (\ln x)^{-1} dx \geq (b-a)/\ln b.$$

Für das Intervall $I = [a,b] = [10\,000,\, 10\,100]$ ergibt sich z. B. der Näherungswert

$$\int_a^b (\ln x)^{-1} dx \geq 100/\ln 10\,100 \approx 10{,}85$$

während der exakte Wert $\pi(10\,100) - \pi(10\,000) = 11$ ist.
Für die Anzahl aller 100-stelligen Primzahlen (in Dezimaldarstellung), also aller Primzahlen im Intervall $I' = [a,b] = [10^{99}, 10^{100}]$ erhalten wir den Näherungswert

$$\int_a^b (\ln x)^{-1} dx \geq 9 * 10^{99}/\ln 10^{100} = 9 * 10^{97}/\ln 10 \approx 3{,}91 \cdot 10^{97}$$

Vergleicht man diese Zahl mit der Anzahl $10^{100} - 10^{99} = 9 \cdot 10^{99}$ aller 100-stelligen Dezimalzahlen, so sehen wir, dass ungefähr jede $900/3{,}91 \approx 230$-te 100-stellige Dezimalzahl prim ist.
Für die Anzahl aller 1000-stelligen Primzahlen (in Dezimaldarstellung), also aller Primzahlen im Intervall $I' = [a,b] = [10^{999}, 10^{1000}]$ erhalten wir dagegen den Näherungswert

$$\int_a^b (\ln x)^{-1} dx \geq 9 * 10^{999}/\ln 10^{1000} = 9 * 10^{996}/\ln 10 \approx 3{,}91 \cdot 10^{996}$$

Hier sehen wir, dass ungefähr jede $9000/3{,}91 \approx 2303$-te der $10^{1000} - 10^{999} = 9 \cdot 10^{999}$ 1000-stelligen Dezimalzahlen prim ist. ◁

6.5 Pseudo-Primzahlen und der Fermat-Test

Bei der Konstruktion eines probabilistischen Monte-Carlo Primzahltests geht man üblicherweise so vor, dass man eine Folge von Teilmengen $\mathcal{A}_n \subseteq \mathbb{Z}_n^*$ definiert, die für hinreichend großes n (d.h. für alle $n \geq n_0$, wobei n_0 eine Konstante ist) folgende drei Bedingungen erfüllen:

T1: Für gegebene Zahlen $a, n \in \mathbb{N}$ kann effizient, d. h. in Polynomialzeit getestet werden, ob $a \in \mathcal{A}_n$ ist.

T2: Für primes n ist $\mathcal{A}_n = \mathbb{Z}_n^*$.

T3: Für (ungerades) zusammengesetztes n ist ein konstanter Anteil aller Elemente von $\mathbb{Z}_n^*$ nicht in $\mathcal{A}_n$ enthalten, d. h. $\|\mathcal{A}_n\| \leq (1-\varepsilon)\varphi(n)$ für eine Konstante $\varepsilon > 0$.

Typischerweise wählt man für $\mathcal{A}_n$ daher eine Eigenschaft, die für alle Elemente $a \in \mathbb{Z}_n^*$ gilt, falls n prim ist. Der zugehörige generische Primzahltest GT arbeitet dann wie folgt.

$GT(n,k)$, $k \geq 1$

```
1 for j := 1 to k do
2   guess randomly a ∈ {1,...,n−1}
3   if a ∉ A_n then return(zusammengesetzt)
4 return(prim)
```

Hierbei steuert der Parameter k die maximale Fehlerwahrscheinlichkeit von $GT(n,k)$. Gilt nämlich $\|\mathcal{A}_n\| \leq (1-\varepsilon)\varphi(n)$ für zusammengesetztes n und eine Konstante $\varepsilon > 0$, so gibt $GT(n,k)$ für zusammengesetztes n mit Wahrscheinlichkeit

$$p = (a_n/(n-1))^k < (a_n/\varphi(n))^k \leq (1-\varepsilon)^k$$

„prim“ aus, wobei $a_n = \|\mathcal{A}_n\|$ ist. Für primes n gibt $GT(n,k)$ dagegen immer „prim“ aus.

Da der 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 schuldig bleiben, also „?“ ausgeben.

Es liegt nahe, den Satz von Fermat zur Konstruktion einer „Testmengensequenz“

$$\mathcal{A}_n^{FT} = \{a \in \mathbb{Z}_n^* \mid a^{n-1} \equiv_n 1\}$$

zu verwenden. Dies führt auf folgenden Fermat-Test (FT).

$FT(n,k)$, $n \geq 3$ ungerade und $k \geq 1$

```
1 berechne die Binärdarstellung n − 1 = Σ_{i=0}^{r} e_i · 2^i mit e_r = 1
2 for j := 1 to k do
3   guess randomly a ∈ {1,...,n−1}
4   z := a
5   for i := r − 1 downto 0 do
6     z := z^2 mod n
7     if e_i = 1 then z := z · a mod n
8   if z ≢_n 1 then return(zusammengesetzt)
9 return(prim)
```

Der Fermat-Test berechnet also die Potenz $z_0 = a^{n-1}$ genau wie der Algorithmus HornerPot ausgehend von $z_r = a$ iterativ mittels $z_{i-1} = z_i^2 a^{e_{i-1}} \bmod n$ für $i = r-1, \ldots, 0$. Er erkennt n als zusammengesetzt, falls $z_0 \neq 1$ ist.

Man nennt eine zusammengesetzte Zahl n, die den Fermat-Test bei Wahl von $a \in \mathbb{Z}_n^*$ besteht (d. h. es gilt $a^{n-1} \equiv_n 1$) eine *Fermat-Pseudo-Primzahl* oder einfach *Pseudo-Primzahl zur Basis a*. Man sagt auch, a ist ein *(falscher) Primzahlzeuge* für n.

Zum Beispiel ist die Zahl 91 pseudo-prim zur Basis 3. Es gibt sogar Zahlen (z. B. $n = 561$) die pseudo-prim zu jeder Basis $a \in \mathbb{Z}_n^*$ sind (sogenannte *Carmichael-Zahlen*). Für diese Zahlen ist Bedingung T3 in obiger Aufzählung nicht erfüllt, weshalb der Fermat-Test als Pseudo-Primzahltest bezeichnet wird. Es ist leicht zu sehen, dass Bedingung T3 für jede zusammengesetzte Zahl, die keine Carmichael-Zahl ist, mit $\varepsilon = 1/2$ erfüllt ist. Carmichael-Zahlen kommen nur sehr selten vor (erst 1992 konnte die Existenz unendlich vieler Carmichael-Zahlen nachgewiesen werden).

6.6 Der Miller-Rabin Test

Der Fermat-Pseudoprimzahltest kann zu einem Monte-Carlo Primzahltest (dem sogenannten Miller-Rabin Test, kurz MRT) erweitert werden. Wie wir gesehen haben, berechnet der Fermat-Test die Potenz $a^{n-1} = z_0$ über eine Folge $z_r, \ldots, z_0$ von Potenzen mit $z_r = a$ und $z_{i-1} = z_i^2 a^{e_{i-1}} \bmod n$ für $i = r-1, \ldots, 0$. Er erkennt n als zusammengesetzt, falls $z_0 \neq 1$ ist. Der Miller-Rabin Test überprüft nun zusätzlich bei jeder Quadrierung, ob $z_i^2 \equiv_n 1$ und $z_i \not\equiv_n \pm 1$ ist. Ist dies der Fall, so muss n zusammengesetzt sein, da z_i eine nichttriviale Lösung der Kongruenz $x^2 \equiv_n 1$ in $\mathbb{Z}_n^*$ ist. Die MRT-Testmenge ist also

$$\mathcal{A}_n^{MRT} = \{a \in \mathcal{A}_n^{FT} \mid \text{ für alle } i = r, \ldots, 1 \text{ mit } z_i^2 \equiv_n 1 \text{ gilt } z_i \equiv_n \pm 1\}.$$

Es ist klar, dass diese Testmengen die Bedingungen T1 und T2 erfüllen. Mit etwas zahlentheoretischem Aufwand lässt sich zeigen, dass auch Bedingung T3 für $\varepsilon = 3/4$ erfüllt ist. Weiter unten werden wir dies für $\varepsilon = 1/2$ zeigen.
Der Miller-Rabin Test lässt sich in Pseudocode wie folgt implementieren.

$MRT(n, k)$, $n \geq 3$ ungerade und $k \geq 1$

```
1  berechne die Binärdarstellung n − 1 = Σ_{i=0}^{r} e_i · 2^i mit e_r = 1
2  for j := 1 to k do
3     guess randomly a ∈ {1, ..., n − 1}
4     z := a
5     for i := r − 1 downto 0 do
6        y := z
7        z := z^2 mod n
8        if z ≡_n 1 ∧ y ≢_n ±1 then return(zusammengesetzt)
9        if e_i = 1 then z := z · a mod n
10    if z ≢_n 1 then return(zusammengesetzt)
11 return(prim)
```

Beispiel 129. *Sei $n = 221 = 13 \cdot 17$. Dann berechnet der Miller-Rabin Test für $a = 137$, $a' = 18$ und $a'' = 174$ die folgenden Werte z_i, z_i' bzw. z_i'' (die dünn gedruckten Werte*

werden nur vom Fermat-Test berechnet, da der Miller-Rabin Test vorher abbricht).

i	e_i	$z_i \equiv_n (z_{i+1})^2 a^{e_i}$	z_i^2	$z'_i \equiv_n (z'_{i+1})^2 (a')^{e_i}$	$(z'_i)^2$	$z''_i \equiv_n (z''_{i+1})^2 (a'')^{e_i}$	$(z''_i)^2$
7	1	**137**	**205**	**18**	**103**	**174**	**220**
6	1	$\mathbf{205 \cdot 137 = 18}$	**103**	$\mathbf{103 \cdot 18 = 86}$	**103**	$\mathbf{220 \cdot 174 = 47}$	**220**
5	0	**103**	**1**	**103**	**1**	**220**	**1**
4	1	$1 \cdot 137 = 137$	205	$1 \cdot 18 = 18$	103	$\mathbf{1 \cdot 174 = 174}$	**220**
3	1	$205 \cdot 137 = 18$	103	$103 \cdot 18 = 86$	103	$\mathbf{220 \cdot 174 = 47}$	**220**
2	1	$103 \cdot 137 = 188$	205	$103 \cdot 18 = 86$	103	$\mathbf{220 \cdot 174 = 47}$	**220**
1	0	205	35	103	1	**220**	**1**
0	0	35		1		**1**	

Bei Wahl von $a = 137$ erkennen also beide Tests die Zahl $n = 221$ als zusammengesetzt. Bei Wahl von $a' = 18$ tut dies nur der Miller-Rabin Test und bei Wahl von $a'' = 174$ keiner von beiden. ◁

Die Zahlen $a \in \mathcal{A}_n^{\text{MRT}}$ werden *starke Primzahlzeugen* für n genannt. Falls n zusammengesetzt ist, sagt man auch, n ist eine *starke Pseudo-Primzahl zur Basis a*. Es gibt nur eine Zahl $n < 2,5 \cdot 10^{10}$, die stark pseudo-prim zu den Basen 2, 3, 5 und 7 ist: $n = 3\,215\,031\,751 = 151 \cdot 751 \cdot 28\,351$.

Wir zeigen nun, dass jede ungerade zusammengesetzte Zahl $n > 2$ höchstens $\varphi(n)/2$ starke Primzahlzeugen hat. Sei $n - 1 = 2^m u$ mit u ungerade, d.h. m ist der kleinste Index i mit $e_i = 1$ bzw. $m - 1$ der größte Index i, so dass $e_0 = \cdots = e_i = 0$ ist. Zudem wählen wir ℓ als den kleinsten Index $i \geq 0$, so dass ein $a \in \mathbb{Z}_n^*$ existiert mit $z_i \equiv_n a^{c_i} \equiv_n -1$, d.h. für alle $i < \ell$ gilt $z_i \not\equiv_n -1$. Da z_m für $a = n - 1$ den Wert $z_m \equiv_n (-1)^u \equiv_n -1$ hat, ist $\ell \leq m$. Sei nun $U_n = \{a \in \mathbb{Z}_n^* \mid a^{2^j u} \equiv_n \pm 1\}$, wobei $j = m - \ell$ ist.

i	e_i	c_i	$a^{c_i} \equiv_n z_i$	
r	1	1	a	
$\vdots$				
m	1	u	a^u	$\equiv_n -1$ ist möglich
$m-1$	0	$2u$	a^{2u}	
$\vdots$				
$\ell = m - j$	0	$2^j u$	$a^{2^j u}$	
$\ell - 1$	0	$2^{j+1} u$	$a^{2^{j+1} u}$	$\not\equiv_n -1$
$\vdots$				
0	0	$2^m u$	$a^{2^m u}$	

Behauptung 130. *U_n ist eine Untergruppe von $\mathbb{Z}_n^*$.*

Es genügt zu zeigen, dass U_n unter Multiplikation abgeschlossen ist: Für $a, b \in U_n$ gilt

$$(ab)^{2^j u} = a^{2^j u} b^{2^j u} \equiv_n (\pm 1)(\pm 1) = \pm 1.$$ □

Behauptung 131. $\mathcal{A}_n^{\text{MRT}} \subseteq U_n$.

Sei $a \in \mathcal{A}_n^{\text{MRT}}$ und sei $z_r, \ldots, z_0$ die zugehörige Folge. Dann gilt $z_0 \equiv_n a^{n-1} \equiv_n a^{2^m u} \equiv_n 1$ und für alle $i = r, \ldots, 1$ mit $z_i^2 \equiv_n 1$ gilt $z_i \equiv_n \pm 1$. Wegen $z_{m-i} \equiv_n a^{2^i u}$ für $i = 0, \ldots, m$ folgt also

$$\forall i \in [m] : a^{2^i u} \equiv_n 1 \Rightarrow a^{2^{i-1} u} \equiv_n \pm 1 \qquad (*)$$

Zudem folgt aus der Definition von ℓ $(= m - j)$,

$$\forall i \in \{j+1, \ldots, m\} : a^{2^i u} \not\equiv_n -1 \qquad (**)$$

Insgesamt erhalten wir also aus (*) und (**) die Implikationen

$$a^{2^m u} \equiv_n 1 \overset{(*,**)}{\Rightarrow} a^{2^{m-1} u} \equiv_n 1 \overset{(*,**)}{\Rightarrow} \cdots \overset{(*,**)}{\Rightarrow} a^{2^{j+1} u} \equiv_n 1 \overset{(*)}{\Rightarrow} a^{2^j u} \equiv_n \pm 1$$

und somit folgt $a \in U_n$. □

Behauptung 132. *Für zusammengesetztes $n \equiv_2 1$ ist U_n eine echte Untergruppe von $\mathbb{Z}_n^*$.*

Falls $n = p^k$ eine Primzahlpotenz mit $p > 2$ und $k \geq 2$ ist, gilt $(p^{k-1}+1)^{p^k - 1} \not\equiv_{p^k} \pm 1$ (siehe Übungen) und somit $a = p^{k-1} + 1 \notin U_n$. Andernfalls können wir n in teilerfremde Faktoren $n = n_1 n_2$ mit $n_1, n_2 > 2$ zerlegen. Zudem existiert nach Definition von j eine Zahl $b \in \mathbb{Z}_n^*$ mit $b^{2^j u} \equiv_n -1$. Dann ist aber die eindeutige Lösung $a \in \mathbb{Z}_n^*$ von

$$\begin{aligned} x &\equiv_{n_1} b, \\ x &\equiv_{n_2} 1 \end{aligned}$$

wegen

$$a^{2^j u} \equiv_{n_1} b^{2^j u} \equiv_{n_1} -1 \Rightarrow a^{2^j u} \not\equiv_n 1 \quad \text{und} \quad a^{2^j u} \equiv_{n_2} 1^{2^j u} = 1 \Rightarrow a^{2^j u} \not\equiv_n -1$$

nicht in U_n enthalten. □

Da U_n als echte Untergruppe von $\mathbb{Z}_n^*$ höchstens halb so groß wie $\mathbb{Z}_n^*$ sein kann, folgt also für ungerades zusammengesetztes n,

$$\left\| \mathcal{A}_n^{\mathrm{MRT}} \right\| \leq \|U_n\| \leq \varphi(n)/2.$$

Damit ist gezeigt, dass der Miller-Rabin Test die Bedingung T3 für $\varepsilon = 1/2$ erfüllt.

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. Unter dieser Hypothese kann der Miller-Rabin Test daher zu einem deterministischen Polynomialzeit-Algorithmus derandomisiert werden (mit der Folge, dass das Primzahlproblem in P lösbar ist). Erst 2002 fanden Agrawal, Kayal und Saxena einen Algorithmus, der das Primzahlproblem auch ohne diese Voraussetzung in P löst.

7 Asymmetrische Kryptosysteme

Diffie und Hellman hatten 1976 die Idee, dass ein Kryptosystem auch dann noch sicher sein könnte, wenn der Chiffrierschlüssel öffentlich bekannt ist. Natürlich setzt dies voraus, dass Sender und Empfänger verschiedene Schlüssel $k \neq k'$ verwenden und dass insbesondere der Dechiffrierschlüssel k' nicht mit vertretbarem Aufwand aus dem Chiffrierschlüssel k berechnet werden kann. Ist dies der Fall, so kann sich jeder Kommunikationsteilnehmer X ein Schlüsselpaar k_X, k'_X erzeugen lassen. X kann dann seinen Chiffrierschlüssel k_X öffentlich bekannt geben und muss nur seinen Dechiffrierschlüssel k'_X geheim halten. Dies hat den großen Vorteil, dass k_X über einen authentisierten Kanal (anstatt über einen sicheren Kanal) zum Sender gelangen kann (d.h. der Empfänger muss nur die Herkunft und Originalität von k_X verifizieren können).

- 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.

Ein symmetrisches Kryptosystem wird auch als **konventionell** oder als **Kryptosystem mit geheimen Schlüsseln** bzw. **Secret-Key-Kryptosystem** bezeichnet. Dagegen spricht man bei asymmetrischen Kryptosystemen auch von **Kryptosystemen mit öffentlichen Schlüsseln** oder von **Public-Key-Kryptosystemen**.

Wie der Name schon sagt, sind bei einem symmetrischen Kryptosystem die Rollen von Sender und Empfänger untereinander austauschbar: Die Vertraulichkeit der Nachrichten beruht auf einem *gemeinsamen Geheimnis* in Form des symmetrischen Schlüssels.

Der Unterschied zwischen symmetrischer und asymmetrischer Verschlüsselung lässt sich sehr schön durch folgende Analogie verdeutlichen, bei der ein Bankschließfach zur Übergabe von Geheiminformationen benutzt wird.

Symmetrische Verschlüsselung: Alice und Bob sind im Besitz eines Schlüssels k für das Schließfach und dieses lässt sich mit k sowohl auf- als auch zuschließen. Alice schließt die Nachricht in den Tresor ein und Bob öffnet danach das Schließfach wieder und liest die Nachricht.

Asymmetrische Verschlüsselung: Das Schließfach ist mit einem Zahlenschloß ausgestattet und nur Bob ist im Besitz der zugehörigen Zahlenkombination k'_B. Alice, die nur die Schließfachnummer k_B kennt, legt ihre Nachricht in das unverschlossene Schließfach und verdreht anschließend das Schloß. Danach öffnet Bob das Schließfach mit seinem „privaten“ Schlüssel k'_B wieder und liest die Nachricht.

An dieser Analogie wird auch deutlich, warum für die Übermittlung des öffentlichen Schlüssels k_B von Bob an Alice ein authentisierter Kanal benutzt werden muss. Andernfalls könnte nämlich ein Angreifer Bobs Schlüssel k_B gegen seinen eigenen austauschen und er könnte dann die für Bob bestimmte Nachricht lesen (und ggf. mit k_B verschlüsselt an Bob weiterleiten ohne dass Alice und Bob dies bemerken).

Da Alice bei der asymmetrischen Verschlüsselung nicht im Besitz von Bobs privatem

Schlüssel k'_B ist, kann sie im Gegensatz zu Bob keine der mit k_B verschlüsselten Nachrichten entschlüsseln, also auch keine Kryptotexte, die Bob von anderen Teilnehmern erhält. Dies hat den Vorteil, dass für jeden Teilnehmer nur ein asymmetrisches Schlüsselpaar generiert werden muss, während für die Kommunikation zwischen n Teilnehmern bis zu $\binom{n}{2}$ symmetrische Schlüssel nötig wären. Zu beachten ist auch, dass mit Bobs Schlüsselpaar (k_B, k'_B) nur eine Nachrichtenübermittlung (von Alice oder anderen Teilnehmern) an Bob möglich ist, und für die Übermittlung an Alice das Schlüsselpaar (k_A, k'_A) von Alice benutzt werden muss.

Die Tatsache, dass bei der Verschlüsselung kein geheimer Schlüssel benutzt wird, hat andererseits aber den Nachteil, dass ein asymmetrisches Kryptosystem nicht absolut sicher sein kann (siehe Übungen). Da die Chiffrierfunktion E_{k_B} von Bob öffentlich bekannt und zudem effizient berechenbar ist, kann ein Gegner bei bekanntem Kryptotext alle Klartexte ausprobieren. Damit das System dennoch sicher ist, muss E_{k_B} eine **Einwegfunktion** (engl. *one-way function*) sein, d.h. die inverse Funktion $D_{k'_B}$ darf ohne Kenntnis von Bobs privatem Schlüssel k'_B nicht effizient berechenbar sein. Da es aber ein Geheimnis (nämlich k'_B) gibt, dessen Kenntnis eine effiziente Invertierung ermöglicht, spricht man von einer **Falltürfunktion** (engl. *trapdoor one-way function*). Da E_{k_B} zudem bijektiv ist, handelt es sich genauer um eine **Falltürpermutation** (engl. *trapdoor one-way permutation*). Es ist leicht zu sehen (siehe Übungen), dass mit deterministischen Public-Key Verfahren keine komplexitätstheoretische Sicherheit erreicht werden kann. Dies ist nur möglich, wenn der Wegfall eines geheimen Chiffrierschlüssels durch Verwendung von Zufall kompensiert wird (siehe Abschnitt über probabilistische Kryptosysteme).

7.1 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. Genauer gesagt beruht das RSA-Verfahren darauf, dass die Primzahleigenschaft effizient getestet werden kann, aber keine effizienten Faktorisierungsalgorithmen bekannt sind.

Für jeden Teilnehmer des RSA-Kryptosystems werden zwei große Primzahlen p, q sowie zwei Exponenten e, d mit $ed \equiv_{\varphi(n)} 1$ generiert, wobei $n = pq$ und $\varphi(n) = (p-1)(q-1)$ ist.

Öffentlicher Schlüssel: $k = (e, n)$,
Privater Schlüssel: $k' = (d, n)$.

Jede Nachricht x wird durch eine Folge $x_1, x_2, \ldots$ von Zahlen $x_i \in \mathbb{Z}_n$ dargestellt, die einzeln wie folgt ver- und entschlüsselt werden:

$$\begin{aligned} \mathsf{RSA}((e,n),x) &= x^e \bmod n, \\ \mathsf{RSA}^{-1}((d,n),y) &= y^d \bmod n. \end{aligned}$$

Die Chiffrierfunktionen $\mathsf{RSA}_{(e,n)}$ und $\mathsf{RSA}^{-1}_{(d,n)}$ können durch „Wiederholtes Quadrieren und Multiplizieren“ effizient berechnet werden.

Der Schlüsselraum ist also

$$K = \{(c,n) \mid \text{es gibt Primzahlen } p \text{ und } q \text{ mit } n = pq \text{ und } c \in \mathbb{Z}^*_{\varphi(n)}\}$$

und

$$S = \{((e,n),(d,n)) \in K \times K \mid ed \equiv_{\varphi(n)} 1\}$$

ist die Menge aller zueinander passenden Schlüsselpaare. Der folgende Satz garantiert die Korrektheit des RSA-Systems.

Satz 133. *Für jedes Schlüsselpaar* $((e,n),(d,n)) \in S$ *und alle* $x \in \mathbb{Z}_n$ *gilt*

$$x^{ed} \equiv_n x.$$

Beweis. Sei $n = pq$ und sei z eine natürliche Zahl mit $ed = z\varphi(n)+1$. Wir zeigen $x^{ed} \equiv_p x$. Die Kongruenz $x^{ed} \equiv_q x$ folgt analog und beide Kongruenzen zusammen implizieren $x^{ed} \equiv_n x$.
Wegen $\varphi(n) = (p-1)(q-1)$ und wegen

$$x^{p-1} \equiv_p \begin{cases} 0, & x \equiv_p 0, \\ 1, & x \not\equiv_p 0 \end{cases}$$

folgt

$$x^{ed} = x^{z\varphi(n)+1} = x^{z(p-1)(q-1)}x = (x^{p-1})^{z(q-1)}x \equiv_p x$$

□

Praktische Durchführung

Bestimmung von p und q: Man wählt zufällig eine Zahl x der Form $30z$ und der verlangten Größenordnung (z. B. $x \in I = [10^{500}, 10^{501})$) und führt einen Primzahltest für die Zahlen $x+1$, $x+7$, $x+11$, $x+13$, $x+17$, $x+19$, $x+23$, $x+29$, $x+30+1$, $x+30+7$, ... durch, bis eine Primzahl p gefunden ist. Wegen $\pi(I)/\|I\| \approx 1/(\ln p)$ und da nur 8 von 30 Zahlen getestet werden, sind hierzu ungefähr $^8/_{30} \ln p$ Primzahltests durchzuführen (bei 500-stelligen Dezimalzahlen sind das ca. 300 Tests).

Bestimmung von d: d soll teilerfremd zu $\varphi(n) = (p-1)(q-1)$ sein. Diese Bedingung wird z. B. von jeder Primzahl größer als $\max\{p,q\}$ erfüllt.

Bestimmung von e: Da $\mathrm{ggT}(d, \varphi(n)) = 1$ ist, liefert der erweiterte euklidische Algorithmus das multiplikative Inverse e von d modulo $\varphi(n)$.

Ver- und Entschlüsselung: Modulares Exponentieren durch wiederholtes Quadrieren und Multiplizieren. Im Vergleich zu symmetrischen Verfahren wie z.B. 3DES oder AES ist RSA aber mindestens um den Faktor 100 langsamer. Daher wird RSA meist nur zum Ver- und Entschlüsseln eines symmetrischen Schlüssels (auch Sitzungsschlüssel genannt) benutzt, mit dem dann größere Datenmengen chiffriert und dechiffriert werden (sogenannte **hybride Verschlüsselung**).

Kryptoanalytische Betrachtungen

1. Es ist klar, dass das RSA-Verfahren gebrochen ist, falls es dem Gegner gelingt, den Modul n zu faktorisieren. In diesem Fall kann er $\varphi(n)$ und damit auch den privaten Dechiffrierexponenten aus dem öffentlichen Exponenten e berechnen. Es ist auch möglich, die Primfaktoren p, q bei Kenntnis von $\varphi(n)$ zu berechnen. Sei $n = pq$ (mit $p, q \in \mathcal{P}$; $p > q$). Wegen

$$\varphi(n) = (p-1)(q-1) = (p-1)(^n/_p - 1) = -p + n + 1 - {}^n/_p$$

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 und q zu $\frac{c \pm \sqrt{c^2-4n}}{2}$ bestimmen lassen.

2. Die Primfaktoren p und q sollten nicht zu nahe beieinander liegen, da n sonst leicht faktorisiert werden kann. Sei $p > q$. Dann gilt $q < \sqrt{n} < a < p$, wobei $a = \frac{(p+q)}{2}$ das arithmetische Mittel von p und q ist. Sei $b = \frac{(p-q)}{2}$ die Entfernung zwischen a und q. Ist nun $p - q$ klein, so ist auch $\lfloor\sqrt{n}\rfloor - q < a - q = b$ klein und daher kann q ausgehend von $\lfloor\sqrt{n}\rfloor$ nach höchstens b Schritten gefunden werden. Um dies zu verhindern, genügt es, $p > 2q$ zu wählen, da dann $\sqrt{n} - q = \sqrt{pq} - q > \sqrt{2}q - q > q/3$ ist.

   Mit dem Verfahren der Differenz der Quadrate lässt sich q sogar in $a - \lceil\sqrt{n}\rceil$ Schritten finden. Wegen

$$n = pq = (a+b)(a-b) = a^2 - b^2$$

   genügt es nämlich, eine Zahl $a > \sqrt{n}$ zu finden, so dass $a^2 - n = b^2$ eine Quadratzahl ist. Für $n = 124\,711$ ist zum Beispiel $\lceil\sqrt{n}\rceil = 353$. Bereits für $a = 356$ ist $a^2 - n = 126\,736 - 124\,711 = 2025 = 45^2$ eine Quadratzahl, woraus wir die beiden Faktoren $p = a + 45 = 401$ und $q = a - 45 = 311$ erhalten.

   Der Aufwand für die Suche ist proportional zur Differenz $a - \sqrt{n}$, die sich wegen $\sqrt{x+y} \leq \sqrt{x} + \frac{y}{2\sqrt{x}}$ wie folgt nach unten abschätzen lässt:

$$a - \sqrt{n} = a - \sqrt{a^2 - b^2} \geq \frac{b^2}{2a}.$$

   Im Fall $p \geq 2q$ gilt jedoch wegen $b = (p-q)/2 = (p+q)/6 + (p-2q)/3 \geq (p+q)/6 = a/3$ (also $3b/a \geq 1$),

$$a - \sqrt{n} \geq \frac{b^2}{2a} = \frac{3b}{a} \cdot \frac{b}{6} \geq \frac{b}{6} \geq \frac{q}{12}.$$

   Daher bringt dieser Angriff in diesem Fall keinen nennenswerten Vorteil gegenüber obiger Faktorisierungsmethode.

3. Für unterschiedliche Teilnehmer sollten verschiedene Module $n = pq$ gewählt werden. Wie wir später sehen werden, erlaubt nämlich die Kenntnis eines Schlüsselpaares $(e, n), (d, n)$ mit $ed \equiv_{\varphi(n)} 1$ die effiziente Faktorisierung von n (siehe Satz 136).
4. Aus Effizienzgründen wird der Verschlüsselungsexponent e meist klein gewählt. Kleinere Werte als z.B. die vierte Fermat-Zahl $2^{16}+1 = 65537$ sollte man jedoch nicht verwenden, da dies zu Angriffsmöglichkeiten führt. Wird etwa dieselbe Nachricht an mehrere Empfänger gesendet, kann eine Dechiffrierung mithilfe des chinesischen Restsatzes möglich sein (Angriff von Hastad, siehe Übungen).
5. Auch die Wahl des Entschlüsselungsexponenten d sollte nicht zu klein ausfallen. Beträgt die Bitlänge von d weniger als ein Viertel der Bitlänge von n, kann d unter Umständen mit einem auf Kettenbrüchen basierenden Verfahren effizient berechnet werden (Angriff von Wiener).

7.1.1 Sicherheit des privaten RSA-Schlüssels

Wie wir gesehen haben, ist das RSA-System gebrochen, falls die Faktorisierung des Moduls n bekannt ist. Das Brechen von RSA ist daher höchstens so schwer wie das Faktorisieren von n.

Dagegen ist nicht bekannt, ob auch umgekehrt aus einem effizienten Algorithmus, der bei Eingabe von (e, n) und y ein x mit $x^e \equiv_n y$ berechnet, ein effizienter Faktorisierungsalgorithmus für n gewonnen werden kann. Es ist also nach heutigem Kenntnisstand nicht ausgeschlossen, dass RSA leichter zu brechen ist als n zu faktorisieren.

Wie der folgende Satz zeigt, erfordert die Bestimmung des geheimen Schlüssels dagegen den gleichen Aufwand wie das Faktorisieren von n. Bei Kenntnis von d kann nämlich leicht ein Vielfaches $v = ed - 1$ von $k = \mathrm{kgV}(p-1, q-1)$ bestimmt und somit n faktorisiert werden. Zunächst beweisen wir jedoch folgendes Lemma, auf welchem die Faktorisierung von n bei Kenntnis von v beruht.

Lemma 134. *Sei $m \geq 1$ und seien y, z zwei Lösungen der Kongruenz $x^2 \equiv_m a$ mit $y \not\equiv_m \pm z$. Dann ist $\mathrm{ggT}(y+z, m)$ ein nicht-trivialer Faktor von m.*

Beweis. Wegen $y^2 \equiv_m z^2$ existiert ein $t \in \mathbb{Z}$ mit

$$(y+z)(y-z) = y^2 - z^2 = tm.$$

Da m also das Produkt $(y+z)(y-z)$ teilt, aber wegen $y \not\equiv_m \pm z$ keiner der beiden Faktoren $y+z$ und $y-z$ durch m teilbar ist, müssen sich die Faktoren von m auf $y+z$ und $y-z$ verteilen, was $1 < \mathrm{ggT}(y+z, m) < m$ impliziert. □

Um nun n bei Kenntnis des privaten Dechiffrierexponenten d zu faktorisieren, betrachten wir folgenden Las-Vegas Algorithmus RSA-Factorize, der durch eine leichte Modifikation aus dem Miller-Rabin Primzahltest hervorgeht.

MRT(n), n ungerade

```
1  sei ∑_{i=0}^{r} e_i · 2^i, e_r = 1, die
       Binärdarstellung von n − 1
2  guess randomly a ∈ {1, ..., n − 1}
3  z := a
4  for i := r − 1 downto 0 do
5     y := z
6     z := z^2 mod n
7     if z ≡_n 1 ∧ y ≢_n ±1 then
8        return(zusammengesetzt)
9     if e_i = 1 then z := z · a mod n
10 if z ≢_n 1 then return(zus.gesetzt)
11 else return(prim)
```

RSA-Factorize(n, v)

```
1  sei ∑_{i=0}^{r} e_i · 2^i, e_r = 1, die
       Binärdarstellung von v
2  guess randomly a ∈ {1, ..., n − 1}
3  z := a
4  for i := r − 1 downto 0 do
5     y := z
6     z := z^2 mod n
7     if z ≡_n 1 ∧ y ≢_n ±1 then
8        return(ggT(y + 1, n))
9     if e_i = 1 then z := z · a mod n
10 if ggT(z, n) > 1 then
       return(ggT(z, n))
11 else return(?)
```

Beispiel 135. *Sei $n = 221 = 13 \cdot 17$. Dann ist $\varphi(221) = 12 \cdot 16 = 192$ und $k = \mathrm{kgV}(12, 16) = \mathrm{kgV}(2^2 3, 2^4) = 3 \cdot 2^4 = 48$. Angenommen, der Gegner könnte zu dem öffentlichen Schlüssel $(e, n) = (25, 221)$ den zugehörigen privaten Schlüssel $(d, n) =$*

$(169, 221)$ *bestimmen. Dann ergibt sich* $v = ed - 1$ *zu* $v = 4224$ *und* RSA-Factorize *berechnet für* $a = 174$, $a' = 111$ *und* $a'' = 137$ *die folgenden Werte* z_i, z_i' *bzw.* z_i''*:*

i	e_i	c_i	$z_i = 174^{c_i}$	$(z_i)^2$	$z_i' = 111^{c_i}$	$(z_i')^2$	$z_i'' = 137^{c_i}$	$(z_i'')^2$
12	1	1	174	220	111	166	137	205
11	0	2	220	1	166	152	205	35
10	0	4	1	1	152	120	35	120
9	0	8	1	1	120	35	120	35
8	0	16	1	1	35	120	35	120
7	1	33	174	220	$120 \cdot 111 = 60$	64	$120 \cdot 137 = 86$	103
6	0	66	220	1	64	118	103	1
5	0	132	1	1	118	1		
4	0	264	1	1				
3	0	528	1	1				
2	0	1056	1	1				
1	0	2112	1	1				
0	0	4224	1					

RSA-Factorize *gelingt also die Faktorisierung von* $n = 221$ *bei Wahl von* $a = 174$ *nicht, wohl aber bei Wahl von* $a' = 111$ *und* $a'' = 137$*. Im ersten Fall findet* RSA-Factorize *den Faktor* $\mathrm{ggT}(118 + 1, 221) = 17$ *und im zweiten den Faktor* $\mathrm{ggT}(103 + 1, 221) = 13$. ◁

Satz 136. *Sei* $n = pq$ *(*$p, q \geq 3$ *prim) und* $v > 0$ *ein Vielfaches von* $k = \mathrm{kgV}(p-1, q-1)$*. Dann gibt* RSA-Factorize(n, v) *mit Wahrscheinlichkeit größer* $1/2$ *einen Primfaktor von* n *aus.*

Beweis. Es ist klar, dass jede Ausgabe von RSA-Factorize in Zeile 10 ein nichttrivialer Faktor von n sein muss. Mit Lemma 134 folgt

$$y \not\equiv_n \pm 1, y^2 \equiv_n 1 \quad \Rightarrow \quad \mathrm{ggT}(y + 1, n) \in \{p, q\},$$

womit auch die Korrektheit jeder Ausgabe in Zeile 8 gezeigt ist.

Wir schätzen nun die Wahrscheinlichkeit ab, dass die Faktorisierung von n nicht gelingt und RSA-Factorize ein Fragezeichen ausgibt.

Sei $v = 2^m u$, $p - 1 = 2^i u_1$ und $q - 1 = 2^j u_2$ mit u, u_1, u_2 ungerade und sei o. B. d. A. $i \leq j$. Zudem sei $F(n)$ die Menge aller Basen $a \in \mathbb{Z}_n^*$, bei deren Wahl RSA-Factorize ein Fragezeichen ausgibt und sei $S(n)$ die Menge

$$S(n) = \{a \in \mathbb{Z}_n^* \mid a^u \equiv_n 1 \ \vee \ \exists t \geq 0 : a^{2^t u} \equiv_n -1\}.$$

RSA-Factorize findet bei Wahl einer Basis $a \in \mathbb{Z}_n^* \setminus S(n)$ einen Primfaktor, da dann zwar $a^u \not\equiv_n 1$ und auch $a^{2^t u} \not\equiv_n -1$ für alle $t \geq 0$ gilt, aber $a^{2^m u} \equiv_n a^v \equiv_n 1$ ist. Daher gilt $F(n) \subseteq S(n)$ und es folgt

$$\Pr[\text{RSA-Factorize}(n, v) = ?] \leq \sigma(n)/(n-1),$$

wobei $\sigma(n) = \|S(n)\|$ ist. Um $\sigma(n)$ zu berechnen, betrachten wir für $t \geq 0$ die Funktionen $\alpha(n) = \|\{a \in \mathbb{Z}_n^* \mid a^u \equiv_n 1\}\|$ und $\alpha_t(n) = \|\{a \in \mathbb{Z}_n^* \mid a^{2^t u} \equiv_n -1\}\|$.

Behauptung 137. *Es gilt* $\mathrm{ggT}(2^t u, p-1) = 2^{\min(t,i)} u_1$ *und* $\mathrm{ggT}(2^t u, q-1) = 2^{\min(t,j)} u_2$.

Wegen

$$k = \mathrm{kgV}(p-1, q-1) = \mathrm{kgV}(2^i u_1, 2^j u_2) = 2^{\max(i,j)}\,\mathrm{kgV}(u_1, u_2)$$

und $k \mid v = 2^m u$ folgt $u_1 | u$ und $u_2 | u$. Da u ungerade ist, folgt somit

$$\mathrm{ggT}(2^t u, p-1) = \mathrm{ggT}(2^t u, 2^i u_1) = 2^{\min(t,i)} u_1$$

und

$$\mathrm{ggT}(2^t u, q-1) = \mathrm{ggT}(2^t u, 2^j u_2) = 2^{\min(t,j)} u_2. \qquad \square$$

Behauptung 138. $\alpha(n) = u_1 u_2$.

Mit dem chinesischen Restsatz folgt

$$\alpha(n) = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^u \equiv_p 1\}\|}_{=:\beta(n)} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^u \equiv_q 1\}\|}_{=:\gamma(n)}.$$

Sei nun g ein Erzeuger von $\mathbb{Z}_p^*$. Dann gilt

$$g^{ku} \equiv_p 1 \quad \Leftrightarrow \quad ku \equiv_{p-1} 0.$$

Dies zeigt $\beta(n) = \mathrm{ggT}(u,\ p-1) \overset{\text{Beh. 137}}{=} u_1$. Analog folgt $\gamma(n) = u_2$. $\square$

Behauptung 139. *Für $t = 0, \ldots, i-1$ ist $\alpha_t(n) = 2^{2t} u_1 u_2$ und für $t \geq i$ ist $\alpha_t(n) = 0$.*

Mit dem chinesischen Restsatz folgt zunächst

$$\alpha_t(n) = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^{2^t u} \equiv_p -1\}\|}_{=:\beta_t(n)} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^{2^t u} \equiv_q -1\}\|}_{=:\gamma_t(n)}.$$

Sei nun g ein Erzeuger von $\mathbb{Z}_p^*$. Dann gilt

$$g^{k 2^t u} \equiv_p -1 \quad \Leftrightarrow \quad k 2^t u \equiv_{p-1} (p-1)/2.$$

Da $\mathrm{ggT}(2^t u, p-1) \overset{\text{Beh. 137}}{=} 2^t u_1$ genau dann ein Teiler von $(p-1)/2 = 2^{i-1} u_1$ ist, wenn $t \leq i-1$ ist, folgt $\beta_t(n) = 2^t u_1$ für $t = 0, \ldots, i-1$ und $\beta_t(n) = 0$ für alle $t \geq i$.
Analog folgt $\gamma_t(n) = 2^t u_2$ für $t = 0, \ldots, j-1$ und $\gamma_t(n) = 0$ für alle $t \geq j$ und damit die Behauptung. $\square$

Behauptung 140. *Es gilt $\sigma(n) \leq \varphi(n)/2$.*

Wegen $\sigma(n) = \alpha(n) + \sum_{t \geq 0} \alpha_t(n)$ folgt mit obigen Behauptungen

$$\begin{aligned}
\sigma(n) &= u_1 u_2 + \sum_{t=0}^{i-1} 2^{2t} u_1 u_2 = u_1 u_2 (1 + \sum_{t=0}^{i-1} 2^{2t}) \\
&= u_1 u_2 (1 + (2^{2i} - 1)/3) = u_1 u_2 (2^{2i} + 2)/3 \\
&\leq u_1 u_2 (2^{i+j} + 2^{i+j-1})/3 = \varphi(n)(1 + 2^{-1})/3 = \varphi(n)/2.
\end{aligned}$$ $\square$

Wegen $\varphi(n) = n - p - q + 1 < n - 1$ folgt nun $\sigma(n)/(n-1) \leq \varphi(n)/2(n-1) < 1/2$, womit der Satz bewiesen ist. ■

7.1.2 Sicherheit partieller Klartextinformationen

Als nächstes gehen wir der Frage nach, wie sicher einzelne Bits der Klartextnachricht sind. Falls es möglich wäre, dem Kryptotext y und dem öffentlichen Schlüssel (e, n) die Parität des Klartextes x effizient zu bestimmen, so könnte auch der gesamte Klartext x effizient berechnet werden. Das letzte Bit des Klartextes ist also genau so sicher wie der gesamte Klartext. Einem Angreifer ist es daher nicht möglich, das letzte Bit des Klartextes zu ermitteln, außer wenn es ihm gelingt, RSA vollständig zu brechen. Wir werden später sehen, dass andere Eigenschaften des Klartextes sehr wohl durch den zugehörigen Kryptotext preisgegeben werden.
Für $x, y \in \mathbb{Z}_n$ mit $y \equiv_n x^e$ sei

$$\text{klartext-parity}(y) = \text{parity}(x) = \begin{cases} 1 & \text{falls } x \text{ ungerade,} \\ 0 & \text{falls } x \text{ gerade.} \end{cases}$$

und

$$\text{klartext-half}(y) = \text{half}(x) = \begin{cases} 0 & \text{falls } 0 \le x < n/2, \\ 1 & \text{falls } n/2 \le x < n \end{cases}$$

Wegen

$$2x \bmod n = \begin{cases} 2x & \text{half}(x) = 0, \\ 2x - n & \text{sonst} \end{cases}$$

gilt dann $(2x \bmod n) \equiv_2 \text{half}(x)$ und somit $\text{half}(x) = \text{parity}(2x \bmod n)$. Daher lässt sich die Berechnung von klartext-half(y) auf die Berechnung von klartext-parity(y) reduzieren:

$$\text{klartext-half}(y) = \text{half}(x) = \text{parity}(2x \bmod n) = \text{klartext-parity}(2^e y \bmod n).$$

Stellen wir die Zahl x/n in der Form

$$x/n = \sum_{i=1}^{\infty} b_i 2^{-i}$$

dar, so berechnet sich die Bitfolge b_i, $i = 1, 2, \ldots$ wegen

$$2^{i-1}x = n(2^{i-2}b_1 + \cdots + b_{i-1} + b_i/2 + b_{i+1}/4 + \cdots) \equiv_n n(b_i/2 + b_{i+1}/4 + \cdots)$$

zu

$$b_i = \text{half}(2^{i-1}x \bmod n) = \text{parity}(2^i x \bmod n) = \text{klartext-parity}(2^{ie} y \bmod n).$$

Setzen wir $z_i = n \sum_{j=1}^{i} b_j 2^{-j}$, so gilt für alle $i > \log_2 n$

$$0 \le x - z_i = n \sum_{j=i+1}^{\infty} b_j 2^{-j} \le n \sum_{j=i+1}^{\infty} 2^{-i} = n/2^i < 1,$$

d.h. $x = \lceil z_{\lceil \log_2 n \rceil} \rceil$. Daher lässt sich x mit Orakelfragen an klartext-parity durch folgenden Algorithmus unter Berechnung der Bits b_i für $i = 1, 2, \ldots, \lceil \log_2 n \rceil$ bestimmen:

```
1  z := 0
2  for i := 1 to ⌈log_2 n⌉ do
3     y := 2^e y mod n
4     b_i := klartext-parity(y)
5     if b_i then z := z + n2^{-i}
6  output ⌈z⌉
```

Beispiel 141. *Sei $n = 1\,457$, $e = 779$ und $y = 722$. Angenommen, das Orakel klartext-parity liefert die in folgender Tabelle angegebenen Werte $b_i = \text{klartext-parity}(y_i)$ für die Kryptotexte $y_i = 2^{ie}y \bmod n$. Dann berechnet obiger Algorithmus die folgenden Werte $z_i = n\sum_{j=1}^{i} b_i 2^{-i}$:*

i	1	2	3	4	5	6	7	8	9	10	11
y_i	1136	847	1369	1258	1156	826	444	408	1320	71	144
b_i	1	0	1	0	1	1	1	1	1	0	0
$n2^{-i}$	728,5	364,3	182,1	91,1	45,5	22,8	11,4	5,7	2,8	1,4	0,7
z_i	728,5	728,5	910,6	910,6	956,2	978,9	990,3	996	998,8	998,8	998,8
x_i	541	1082	707	1414	1371	1285	1113	769	81	162	324

Der gesuchte Klartext ist also $x = \lceil z_{11} \rceil = \lceil 998,8 \rceil = 999$. Dass x tatsächlich die vorgegebene Paritätsbitfolge (b_i) generiert, lässt sich durch Berechnung der zu den Kryptotexten y_i gehörigen Klartexte $x_i = 2^i x \bmod n$ verifizieren (siehe letzte Tabellenzeile). ◁

7.2 Quadratische Reste

In diesem Abschnitt beschäftigen wir uns mit dem Problem, Lösungen für eine quadratische Kongruenzgleichung $x^2 \equiv_m a$ zu bestimmen. Zunächst gehen wir der Frage nach, wie sich feststellen lässt, ob überhaupt Lösungen existieren.

Definition 142. *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 $x^2 \equiv_m a$. Die Menge **$\mathsf{QNR}_m := \mathbb{Z}_m^* \setminus \mathsf{QR}_m$** enthält alle **quadratischen Nichtreste modulo m**.*
Sei $p > 2$ eine Primzahl und $a \in \mathbb{Z}$. Dann heißt

$$\mathcal{L}(a,p) = \left(\frac{a}{p}\right) = \begin{cases} 1, & a \bmod p \in \mathsf{QR}_p \\ -1, & a \bmod p \in \mathsf{QNR}_p \\ 0, & \text{sonst} \end{cases}$$

*das **Legendre-Symbol von a modulo p**.*

Die quadratische Kongruenz $x^2 \equiv_m a$ besitzt also für ein $a \in \mathbb{Z}_m^*$ genau dann eine Lösung, wenn $a \in \mathsf{QR}_m$ ist. Da mit $a, b \in \mathsf{QR}_m$ auch $ab \in \mathsf{QR}_m$ ist, bildet QR_m eine Untergruppe von $\mathbb{Z}_m^*$. Wie das folgende Lemma zeigt, kann die Lösbarkeit von $x^2 \equiv_m a$ für primes m effizient entschieden werden.

Lemma 143. *Sei $a \in \mathbb{Z}_p^*$, $p > 2$ prim, und sei g ein beliebiger Erzeuger von $\mathbb{Z}_p^*$. Dann sind die folgenden drei Bedingungen äquivalent:*

1) $a \in \mathsf{QR}_p$
2) $a^{(p-1)/2} \equiv_p 1$
3) $\log_{p,g}(a)$ *ist gerade*

Beweis.
1) $\Rightarrow$ 2): 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,

$$a^{(p-1)/2} \equiv_p b^{p-1} \equiv_p 1$$

2) $\Rightarrow$ 3): Angenommen, $a \equiv_p g^k$ für ein ungerades $k = 2 \cdot j + 1$. Dann ist

$$a^{(p-1)/2} \equiv_p g^{k(p-1)/2} \equiv_p \underbrace{g^{j\cdot(p-1)}}_{\equiv_p (g^{(p-1)})^j \equiv_p 1} g^{(p-1)/2} \equiv_p g^{(p-1)/2} \equiv_p -1 \not\equiv_p 1$$

3) $\Rightarrow$ 1): Ist $a \equiv_p g^k$ für $k = 2j$ gerade, so folgt $a \equiv_p (g^j)^2$, also $a \in \mathsf{QR}_p$. □

Somit zerfällt $\mathbb{Z}_p$ in die drei Teilmengen QR_p, QNR_p und $\mathbb{Z}_p \setminus \mathbb{Z}_p^* = \{0\}$, wobei die ersten beiden jeweils $(p-1)/2$ Elemente enthalten. Zudem ist das Produkt ab von $a, b \in \mathbb{Z}_p^*$ genau dann in QR_p, wenn $a, b \in \mathsf{QR}_p$ oder $a, b \in \mathsf{QNR}_p$ sind. Als weitere Folgerung erhalten wir folgende Formel zur effizienten Berechnung des Legendre-Symbols.

Satz 144 (**Eulers Kriterium**). *Für alle $a \in \mathbb{Z}$ und $p > 2$ prim gilt*

$$a^{(p-1)/2} \equiv_p \left(\frac{a}{p}\right)$$

Beweis. Es ist klar, dass diese Kongruenz im Fall $a \equiv_p 0$ gilt. Nach obigem Lemma gilt sie auch im Fall $a \bmod p \in \mathsf{QR}_p$, da dann $a^{(p-1)/2} \equiv_p 1 = \left(\frac{a}{p}\right)$ ist.

Es bleibt also der Fall, dass $a \bmod p \in \mathsf{QNR}_p$ ist. Da das Polynom $x^2 - 1$ in $\mathbb{Z}_p$ höchstens zwei Nullstellen hat und neben $x = 1$ nach dem Satz von Fermat auch $a^{(p-1)/2} \bmod p$ eine Nullstelle ist, muss $a^{(p-1)/2} \equiv_p \pm 1$ sein. Daraus folgt nun $a^{(p-1)/2} \equiv_p -1$, da im Fall $a^{(p-1)/2} \equiv_p 1$ die Zahl $a \bmod p$ in QR_p und somit nicht in QNR_p wäre. □

Korollar 145. *Für alle $a, b \in \mathbb{Z}$ und $p > 2$ prim gilt*

- $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = \begin{cases} 1, & p \equiv_4 1 \\ -1, & p \equiv_4 3 \end{cases}$
- $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right)$

Als weiteres Korollar aus Eulers Kriterium erhalten wir eine Methode, quadratische Kongruenzgleichungen im Fall $p \equiv_4 3$ effizient zu lösen. Im Fall $p \equiv_4 1$ ist dagegen kein effizienter deterministischer Lösungsalgorithmus bekannt. Allerdings gibt es hierfür effiziente probabilistische Algorithmen (z.B. von Tonelli und Shanks).

Korollar 146. *Sei $p > 2$ prim, dann besitzt die quadratische Kongruenzgleichung $x^2 \equiv_p a$ für jedes $a \in \mathsf{QR}_p$ in $\mathbb{Z}_p$ genau zwei Lösungen. Im Fall $p \equiv_4 3$ sind dies $\pm a^k \bmod p$ (für $k = (p+1)/4$), wovon nur $a^k \bmod p$ ein quadratischer Rest ist.*

Beweis. Da $a \in \mathsf{QR}_p$ ist, 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). Da $\mathbb{Z}_p$ ein Körper ist, existieren keine weiteren Lösungen.

Sei nun $p \equiv_4 3$. Dann liefert Eulers Kriterium für $k = (p+1)/4$ die Kongruenz

$$(a^k)^2 = a^{(p+1)/2} = a^{(p-1)/2} \cdot a \equiv_p a.$$

Da mit a auch $a^k \bmod p \in \mathsf{QR}_p$ ist, folgt

$$\left(\frac{-a^k}{p}\right) = \left(\frac{-1}{p}\right) \cdot \left(\frac{a^k}{p}\right) = -\left(\frac{a^k}{p}\right) = -1$$

Also ist $-a^k \bmod p$ ein quadratischer Nichtrest. □

7.3 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 nur durch Faktorisierung des Moduls n gebrochen werden kann, erfüllt das Rabin-System diese Bedingung. Ähnlich wie bei RSA verwendet das Rabin-System als Falltürfunktion eine Polynomfunktion $E_k(x) = x(x+e) \bmod n$, wobei $n = pq$ das Produkt zweier großer Primzahlen ist. Um die Dechiffrierung zu erleichtern, wählt jeder Teilnehmer ein Primzahlpaar p, q mit $p \equiv_4 q \equiv_4 3$, während für e eine beliebige Zahl in $\mathbb{Z}_n$ gewählt werden kann (wir werden weiter unten sehen, dass e keine kryptografische Relevanz hat und daher auch weggelassen bzw. $e = 0$ gesetzt werden kann).

Öffentlicher Schlüssel: e, n
Geheimer Schlüssel: p, q

Der Klartextraum ist $M = \mathbb{Z}_n = \{0, \ldots, n-1\}$ und die Verschlüsselungsfunktion ist

$$E((e,n),x) \quad = \quad x(x+e) \bmod n = y$$

Zur Entschlüsselung eines Kryptotextes $y \in \{0, \ldots, n-1\}$ muss der legale Empfänger Bob die quadratische Kongruenzgleichung $x(x+e) \equiv_n y$ lösen, die äquivalent zu der Kongruenz

$$(\underbrace{x + 2^{-1}e}_{x'})^2 \equiv_n \underbrace{y + (2^{-1}e)^2}_{y'}$$

(*quadratische Ergänzung*) ist, wobei $2^{-1} = (n+1)/2$ das multiplikative Inverse zu 2 modulo n ist.

Setzen wir also $x' = x + 2^{-1}e$ und $y' = y + (2^{-1}e)^2$, so genügt es, alle Lösungen x'_i der Kongruenz $(x')^2 = y'$ zu bestimmen, und daraus die zugehörigen Klartext-Kandidaten $x_i = x'_i - 2^{-1}e \bmod n$ zu berechnen. Im Fall $y' \equiv_n 0$ gibt es nur eine Lösung $x' = 0$. Ist dagegen $\mathrm{ggT}(y', n) \in \{p, q\}$, so gibt es zwei Lösungen (dieser Fall ist sehr unwahrscheinlich und würde dem Gegner die Faktorisierung von n ermöglichen). Im verbliebenen Fall $\mathrm{ggT}(y', n) = 1$, also $y' \in \mathbb{Z}_n^*$, hat die Kongruenz $(x')^2 = y'$ vier Lösungen für x' und der folgende Satz zeigt, wie sich diese bei Kenntnis von p und q effizient bestimmen lassen.

Das Rabin-System erfüllt also nicht die Bedingung der eindeutigen Dechiffrierbarkeit. Wir werden jedoch weiter unten sehen, wie man den Klartextraum auf eine geeignete Teilmenge $M'' \subseteq \mathbb{Z}_n^*$ einschränken kann, so dass diese Bedingung erfüllt ist.

Satz 147. *Sei $n = pq$ für Primzahlen p, q mit $p \equiv_4 q \equiv_4 3$. Dann besitzt die quadratische Kongruenz $x^2 \equiv_n a$ für jedes $a \in \mathsf{QR}_n$ genau vier Lösungen, wovon genau eine ein quadratischer Rest ist.*

Beweis. Mit der Kongruenz $x^2 \equiv_n a$ besitzen wegen $n = pq$ auch die beiden Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ Lösungen, und zwar jeweils genau zwei (siehe Korollar 146):

$$\begin{aligned} u_1 &= a^{(p+1)/4} \bmod p \in \mathsf{QR}_p \\ v_1 &= a^{(q+1)/4} \bmod q \in \mathsf{QR}_q \\ u_2 &= -a^{(p+1)/4} \bmod p \in \mathsf{QNR}_p \\ v_2 &= -a^{(q+1)/4} \bmod q \in \mathsf{QNR}_q \end{aligned}$$

Mit dem chinesischen Restsatz lässt sich für jedes Paar $(i,j) \in [2] \times [2]$ eine Lösung x_{ij} des Systems

$$x \equiv_p u_i$$
$$x \equiv_q v_j$$

bestimmen. Die Kongruenz $x^2 \equiv_n a$ kann nicht mehr als diese vier Lösungen haben, da sonst für mindestens eine der beiden Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ mehr als zwei Lösungen existieren würden.

Wegen

$$x_{ij} \in \mathsf{QR}_n \;\Rightarrow\; \exists s : s^2 \equiv_n x_{ij} \;\Rightarrow\; s^2 \equiv_p u_i \wedge s^2 \equiv_q v_j \;\Rightarrow\; u_i \in \mathsf{QR}_p \wedge v_j \in \mathsf{QR}_q$$

können $x_{1,2}, x_{2,1}, x_{2,2}$ keine quadratischen Reste modulo n sein.

Da aber u_1 und v_1 quadratische Reste modulo p bzw. q sind, gibt es Zahlen $s \in \mathbb{Z}_p^*$ und $t \in \mathbb{Z}_q^*$ mit $s^2 \equiv_p u_1$ und $t^2 \equiv_q v_1$. Folglich erfüllt die Lösung $w \in \mathbb{Z}_n^*$ des Systems

$$x \equiv_p s$$
$$x \equiv_q t$$

die Kongruenzen

$$w^2 \equiv_p s^2 \equiv_p u_1 \equiv_p x_{1,1} \quad \text{und} \quad w^2 \equiv_q t^2 \equiv_q v_1 \equiv_q x_{1,1}$$

und somit $w^2 \equiv_n x_{1,1}$, d.h. $x_{1,1} \in \mathsf{QR}_n$. □

Als weitere für die Kryptografie interessante zahlentheoretische Funktionen erhalten wir somit für jedes $n = pq$, wobei p, q Primzahlen mit $p \equiv_4 q \equiv_4 3$ sind, die **diskrete Quadratfunktion** $x \mapsto x^2 \bmod n$, die nach vorigem Satz eine Permutation auf QR_n ist. Ihre Umkehrfunktion $x \mapsto \sqrt{x} \bmod n$ heißt **diskrete Quadratwurzelfunktion** auf QR_n. Wir werden später sehen, dass sich diese Funktion nur bei Kenntnis der Primfaktoren p und q von n effizient berechnen lässt. Ohne Kenntnis der Faktoren von n lässt sich nicht einmal effizient entscheiden, ob eine gegebene Zahl $a \in \mathbb{Z}_n^*$ in QR_n ist oder nicht. Aus diesem Grund können wir den Klartextraum des Rabin-Systems auch nicht einfach auf die Menge QR_n einschränken, um die Chiffrierfunktion injektiv zu machen.

Beispiel 148. *Wählen wir $p = 7$, $q = 11$ und $e = 2$, so erhalten wir die folgenden Schlüssel.*
Öffentlicher Schlüssel: $k = (e, n) = (2, 77)$
Privater Schlüssel: $k' = (p, q) = (7, 11)$

Um den Klartext $x = 12$ zu verschlüsseln, wird der Kryptotext

$$y = E(k, x) = 12(12 + 2) \bmod 77 = 14$$

berechnet. Da $2^{-1}e = 2^{-1} \cdot 2 = 1$ 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

$$u^2 \equiv_7 1 \equiv_7 1 \text{ und } v^2 \equiv_{11} 15 \equiv_{11} 4$$

zu $u_{1,2} = \pm 15^2 \bmod 7 = \pm 1$ (wegen $\frac{p+1}{4} = 2$) und $v_{1,2} = \pm 4^3 \bmod 11 = \pm 64 \bmod 11 = \mp 2 \bmod 11$ (wegen $\frac{q+1}{4} = 3$). Mit dem chinesischen Restsatz lassen sich $u_{1,2}$ und $v_{1,2}$ zu den vier Lösungen $x'_{ij} = 64, 57, 20$ und 13 zusammensetzen, die auf die vier Klartextkandidaten $12, 19, 56$ und 63 führen. ◁

Da auch ein Angreifer die Kongruenz $x(x+e) \equiv_n y$ ohne nennenswerten Aufwand in die Kongruenz $(x')^2 \equiv_n y'$ mit $x' = x + 2^{-1}e$ und $y' = y + (2^{-1}e)^2$ überführen kann, kann e auch gleich auf Null gesetzt werden. Zudem lässt sich die Anzahl der Klartextkandidaten von vier auf zwei reduzieren, wenn man den Klartextraum von $M = \mathbb{Z}_n$ auf die Menge $M' = \{1, \ldots, (n-1)/2\}$ einschränkt.

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. Um das Rabin-System zu brechen, wird ein effizienter Algorithmus A benötigt, der bei Eingabe (a, n) mit $a \in \mathsf{QR}_n$ eine Zahl $c = A(a, n)$ mit $c^2 \equiv_n a$ berechnet. Dabei können wir o.B.d.A. annehmen, dass $c \leq (n-1)/2$ ist. Unter Verwendung von A erhalten wir nun folgenden probabilistischen Algorithmus Rabin-Factorize zur Faktorisierung von n.

Rabin-Factorize(n)

```
1  repeat forever
2    guess randomly x ∈ {1, ..., (n−1)/2}
3    if ggT(x, n) > 1 then
4      return(ggT(x, n))
5    a := x^2 mod n
6    z := A(a, n)
7    if z ≢_n ±x then
8      return(ggT(x + z, n))
```

Satz 149. *Der Algorithmus Rabin-Factorize gibt bei Eingabe $n = pq$, p und q prim mit $p \equiv_4 q \equiv_4 3$, einen Primfaktor von n aus. Die Wahrscheinlichkeit, dass er die repeat-Schleife mehr als t-mal durchläuft, ist kleiner als 2^{-t}.*

Beweis. Es ist klar, dass $\mathrm{ggT}(x, n)$ im Fall $\mathrm{ggT}(x, n) > 1$ (Zeile 3) ein Primfaktor von n ist. Lemma 134 stellt sicher, dass dies auch für die Ausgabe in Zeile 7 zutrifft.

Um die Laufzeit von Rabin-Factorize abzuschätzen, betrachten wir die Zufallsvariable X, die die Wahl von x in der Menge $M' = \{1, \ldots, \frac{n-1}{2}\}$ beschreibt. Zudem sei p_1 die Wahrscheinlichkeit, dass dem Algorithmus in einem Schleifendurchlauf die Faktorisierung von n gelingt. Sei $x' \neq x$ die neben x zweite Lösung in $M' \cap \mathbb{Z}_n^*$ der Kongruenz $x^2 \equiv_n a$. Dann gilt

$$p_1 = \underbrace{\Pr[\mathrm{ggT}(X, n) > 1]}_{\Pr[X \notin \mathbb{Z}_n^*] =: \alpha} + \underbrace{\Pr[X \in \mathbb{Z}_n^* \text{ und } X \not\equiv_n \pm A(X^2, n)]}_{\beta}.$$

mit

$$\begin{aligned}
\beta &= \sum_{x \in M' \cap \mathbb{Z}_n^*} \Pr[X = x \wedge A(X^2, n) = x'] \\
&= \sum_{x \in M' \cap \mathbb{Z}_n^*} \Pr[A(X^2, n) = x'] \underbrace{\Pr[X = x \mid A(X^2, n) = x']}_{1/2} \\
&= \Pr[A(X^2, n) \in M' \cap \mathbb{Z}_n^*]/2 = \Pr[X \in \mathbb{Z}_n^*]/2 = (1 - \alpha)/2.
\end{aligned}$$

Somit ist $p_1 = \alpha + \beta = (\alpha + 1)/2 > 1/2$. Die Wahrscheinlichkeit, dass Rabin-Factorize mehr als t Schleifendurchläufe ausführt, ist also $(1 - p_1)^t < 2^{-t}$. □

Um eine eindeutige Dechiffrierung zu erhalten, erweitern wir das Legendre-Symbol zum *Jacobi-Symbol*.

Definition 150. *Das* Jacobi-Symbol *ist für alle a und alle ungeraden $m = p_1^{e_1} \cdots p_r^{e_r} \geq 3$ 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 $p_1 < \cdots < p_r$ die Primfaktoren von m sind. Ist zwar $\left(\frac{a}{m}\right) = 1$, aber $a \in \mathsf{QNR}_m$ kein quadratischer Rest modulo m, so heißt a **quadratischer Pseudorest** modulo m (kurz: $a \in \widetilde{\mathsf{QR}}_m$).*

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. Zum Beispiel gibt es in $\mathbb{Z}_n^*$ ($n = p \cdot q$ für Primzahlen p und q mit $p \equiv_4 q \equiv_4 3$) wie wir gesehen haben, genau $\varphi(n)/4$ quadratische Reste und $3\varphi(n)/4$ quadratische Nichtreste, wogegen nur für die Hälfte aller $a \in \mathbb{Z}_n^*$ die Gleichung $\left(\frac{a}{m}\right) = -1$ gilt. Folglich gibt es in diesem Fall genau so viele quadratische Reste wie quadratische Pseudoreste.

Offensichtlich überträgt sich die in Korollar 145 angegebene Multiplikativität des Legendre-Symbols auf das Jacobi-Symbol. 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 151 (Quadratisches Reziprozitätsgesetz, Gauß)**.** *Es seien $m, n > 2$, ungerade und teilerfremd. Dann gilt*

$$\left(\frac{n}{m}\right) \cdot \left(\frac{m}{n}\right) = (-1)^{(m-1)\cdot(n-1)/4}$$

Satz 152. *Für ungerades m gilt*

$$\left(\frac{2}{m}\right) = (-1)^{\frac{m^2-1}{8}}.$$

Man beachte, dass $\frac{m^2-1}{8}$ genau dann gerade ist, wenn $m \equiv_8 1$ oder $m \equiv_8 7$ gilt, und dass $(m-1)\cdot(n-1)/4$ genau dann gerade ist, wenn $m \equiv_4 1$ oder $n \equiv_4 1$ gilt.

Korollar 153. *Seien a und m gegeben mit $m \geq 3$ ungerade und $\mathrm{ggT}(a,m) = 1$, dann lässt sich $\left(\frac{a}{m}\right)$ durch einen Algorithmus der Zeitkomplexität $O(n^3)$ berechnen.*

Beweis. Dies folgt, ähnlich wie beim euklidischen Algorithmus, aus folgenden Gleichungen.

$$\left(\frac{a}{m}\right) = \begin{cases} 1, & \text{falls } a = 1 \\ \left(\frac{m \bmod a}{a}\right)(-1)^{(a-1)(m-1)/4}, & \text{falls } a \neq 1 \text{ ungerade} \\ \left(\frac{b}{m}\right), & \text{falls } a = 2^{2k}b, b \text{ ungerade} \\ \left(\frac{b}{m}\right)(-1)^{(m^2-1)/8}, & \text{falls } a = 2^{2k+1}b, b \text{ ungerade} \end{cases}$$

□

Beispiel 154. *Das Jacobi-Symbol von 73 modulo 83 ist*

$$\left(\tfrac{73}{83}\right) = \left(\tfrac{10}{73}\right) \underbrace{(-1)^{82\cdot 72/4}}_{=1} = \underbrace{\left(\tfrac{2}{73}\right)}_{=1} \left(\tfrac{5}{73}\right) = \left(\tfrac{3}{5}\right) \underbrace{(-1)^{72\cdot 4/4}}_{=1} = \left(\tfrac{2}{3}\right) = -1$$

◁

Schränken wir nun den Klartextraum auf die Teilmenge $M'' = \{x \in M' \mid \left(\frac{x}{n}\right) = 1\}$ der Menge $M' = \{1, \ldots, (n-1)/2\}$ ein, so erhalten wir als Kryptotextraum die Menge QR_n. Zudem existiert zu jedem $y \in \mathsf{QR}_n$ genau ein Klartext $x \in M''$ mit $x^2 \equiv_n y$. Wir wissen bereits, dass genau eine Quadratwurzel $\sqrt{y} \in \mathsf{QR}_n$ von y existiert. Neben $x' = \sqrt{y} \bmod n$ ist $x'' = -\sqrt{y} \bmod n$ wegen (vgl. Korollar 146)

$$\left(\frac{x''}{n}\right) = \underbrace{\left(\frac{x''}{p}\right)}_{-\left(\frac{x'}{p}\right)} \underbrace{\left(\frac{x''}{q}\right)}_{-\left(\frac{x'}{q}\right)} = \left(\frac{x'}{n}\right)$$

die einzige Lösung der Kongruenz $x^2 \equiv_n y$ mit $\left(\frac{x}{n}\right) = 1$. Da aber genau eine dieser beiden Lösungen in M' enthalten ist, existiert in M'' genau ein x mit $x^2 \equiv_n y$. Zwar lässt sich die Einschränkung des Klartextraums auf M' problemlos realisieren, aber eine weitere Einschränkung auf M'' erscheint problematisch. Einfacher ist es, das Bit $\left(\frac{x}{n}\right)$ an den Empfänger zu übermitteln (entweder unverschlüsselt, wie es bei „Schulbuch-RSA“ der Fall ist, oder in verschlüsselter Form).

7.4 Das ElGamal-Kryptosystem

Das System von ElGamal (1985) ist ein probabilistisches Public-key Verfahren auf der Basis des diskreten Logarithmus.

Sei p eine große Primzahl und α ein Erzeuger von $\mathbb{Z}_p^*$ (p und α sind öffentlich). Jeder Teilnehmer X wählt als privaten Schlüssel eine Zahl $a \in \mathbb{Z}_{p-1} = \{0, \ldots, p-2\}$ und gibt $\beta = \alpha^a \bmod p$ öffentlich bekannt:

Öffentlicher Schlüssel: $k = (p, \alpha, \beta)$
Privater Schlüssel: $k' = (p, \alpha, \beta, a)$

Will Alice nun eine Nachricht $x \in \mathbb{Z}_p^* = \{1, \ldots, p-1\}$ an Bob senden, so

- wählt Alice zufällig eine Zahl $z \in \mathbb{Z}_{p-1}$,
- berechnet den „Schlüssel“ $\gamma = \beta^z \bmod p$ und
- sendet den Kryptotext (y_1, y_2) mit $y_1 = \alpha^z \bmod p$ und $y_2 = \gamma x \bmod p$ an Bob.

Der Kryptotext wird also auf die doppelte Länge des Klartextes aufgebläht. Nach Erhalt des Kryptotextpaars (y_1, y_2)

- berechnet Bob zunächst $\gamma = y_1^a \bmod p$ (es gilt $y_1^a \equiv_p \alpha^{za} \equiv_p \beta^z \equiv_p \gamma$),
- und anschließend den Klartext $x = y_2 \gamma^{-1} \bmod p$.

Beispiel 155. *Sei $p = 2579$ und $\alpha = 2$. Bob wählt den privaten Schlüssel $a = 765$ aus der Menge $\{0, \ldots, 2577\}$ und gibt die Zahl $\beta = \alpha^a \bmod p = 2^{765} \bmod 2579 = 949$ bekannt. Um den Klartext $x = 1299$ an Bob zu übermitteln, berechnet Alice den zugehörigen Kryptotext (y_1, y_2) wie folgt:*

- *Sie wählt zufällig eine Zahl z aus der Menge $\{0, \ldots, 2577\}$ (z. B. $z = 853$),*

- *berechnet* $\gamma = 949^{853} \bmod 2579 = 2424$ *und sendet*
- *das Paar* $(y_1, y_2) = (2^{853} \bmod 2579, 1299 \cdot 2424 \bmod 2579) = (435, 2396)$ *an Bob.*

Nach Erhalt von $y = (435, 2396)$ *berechnet Bob zunächst* $\gamma = 435^{765} \bmod 2579 = 2424$ *und anschließend den Klartext* $x = 2396 \cdot 2424^{-1} \bmod 2579 = 1299$. ◁

Es ist klar, dass das ElGamal-System gebrochen ist, falls es dem Gegner gelingt, den privaten Schlüssel $a = \log_{p,\alpha}(\beta)$ zu berechnen. Eine notwendige Bedingung für die Sicherheit von ElGamal ist daher, dass der diskrete Logarithmus in $\mathbb{Z}_p^*$ nicht mit vertretbarem Aufwand zu berechnen ist. Hierfür sollte p mindestens eine 300-stellige Dezimalzahl sein.

Im Folgenden betrachten wir verschiedene Sicherheitsaspekte des ElGamal-Systems.

Zunächst untersuchen wir die komplexitätstheoretische Sicherheit des ElGamal-Systems. Da wir mit Eulers Kriterium leicht die Parität von a aus $\beta = \alpha^a \bmod p$ und die Parität von z aus $y_1 = \alpha^z \bmod p$ ableiten können, lässt sich leicht ermitteln, ob $\gamma = \beta^z$ in QR_p ist oder nicht. Da zudem $y_2 = \gamma x \bmod p$ genau dann in QR_p ist, wenn entweder $\gamma, x \in \mathsf{QR}_p$ oder $\gamma, x \in \mathsf{QNR}_p$ sind, lässt sich auch leicht ermitteln, ob x in QR_p ist oder nicht. Ersteres trifft nämlich genau dann zu, wenn y_2 und γ beide in QR_p oder beide in QNR_p sind.

Daraus ergibt sich unmittelbar, dass folgender Gegner $G = (X_0, X_1, V)$ einen Vorteil von $\alpha(G) = 1$ erzielen kann. (X_0, X_1) wählen zwei Klartexte x_0 und x_1 mit $x_0 \in \mathsf{QR}_p$ und $x_1 \notin \mathsf{QR}_p$. $V(x_0, x_1, (y_1, y_2))$ ermittelt anhand des Kryptotextes $(y_1, y_2) = E((p, \alpha, \beta), x_b)$ wie oben beschrieben das richtige Bit b. Um diese Schwachstelle zu beheben, genügt es, von der Gruppe $\mathbb{Z}_p^*$ zur zyklischen Untergruppe $\mathsf{QR}_p = \langle \alpha^2 \rangle$ überzugehen. Wählen wir zudem p von der Form $p = 2q + 1$ mit p, q prim, so hat QR_p die Ordnung q, d.h. jedes Element $\alpha \in \mathsf{QR}_p \setminus \{1\}$ ist ein Erzeuger von QR_p. In diesem Fall ist wegen $p \equiv_4 3$ für jedes $x \in \mathbb{Z}_p^*$ genau eine der beiden Zahlen x und $p - x$ in QR_p. Dies liefert eine einfach zu berechnende Bijektion zwischen QR_p und $\mathbb{Z}_q$, weshalb wir auch $\mathbb{Z}_q$ anstelle von QR_p als Klartextraum deklarieren können.

Als nächstes gehen wir der Frage nach, wie sicher die einzelnen Bits des privaten Schlüssels a sind. Da a genau dann gerade ist, wenn $\beta = \alpha^a \bmod p$ ein quadratischer Rest ist, lässt sich das niederwertigste Bit von a leicht nach Eulers Kriterium bestimmen. Bezeichnen wir das Bit an der Stelle $i \geq 0$ von $a = \log_{p,\alpha}(\beta)$ mit $L_i(\beta)$ (d.h. $a = (L_r(\beta) \cdots L_0(\beta))_2$ für $r = \lfloor \log_2(p-2) \rfloor$), so gilt

$$L_0(\beta) = 0 \Leftrightarrow \beta^{(p-1)/2} \equiv_p 1.$$

Allgemeiner kann man zeigen, dass sich im Fall $p - 1 = 2^m u$, u ungerade, die m niederwertigen Bits $L_{m-1}(\beta), \ldots, L_0(\beta)$ von a effizient berechnen lassen. Dagegen ist die Berechnung des nächsten Bits $L_m(\beta)$ nicht effizient möglich, außer wenn alle Bits von a (und damit der diskrete Logarithmus) effizient berechenbar sind.

Wir zeigen dies für den Spezialfall $m = 1$ (d.h. $p \equiv_4 3$; der Fall $m = 2$ bzw. $p \equiv_8 5$ wird in den Übungen betrachtet).

Unter der Annahme, dass für ein gegebenes $\gamma \in \langle \alpha \rangle$ nicht nur $L_0(\gamma)$, sondern auch $L_1(\gamma)$ effizient berechenbar ist, können wir $L_2(\beta)$ wie folgt berechnen:

- Zuerst setzen wir das niederwertigste Bit $L_0(\beta)$ im Fall $L_0(\beta) = 1$ auf 0, indem wir β durch $\beta\alpha^{-1} \bmod p$ ersetzen.
- Als nächstes berechnen wir die beiden Quadratwurzeln $\omega_{1,2}$ von β, d.h. $\omega_{1,2} = \pm\beta^{(p+1)/4} \bmod p$.
- Da $L_0(\beta) = 0$ ist, erhalten wir die Binärdarstellung des diskreten Logarithmus

$\log_{p,\alpha}(\omega_j)$ einer dieser beiden Wurzeln aus der Binärdarstellung von $\log_{p,\alpha}(\beta)$ durch einen Rechtsshift um eine Stelle, d.h. es gilt $L_{i+1}(\beta) = L_i(\omega_j)$ für $i = 0, \ldots, r-1$.

- Wegen $\omega_1 \in \mathsf{QR}_p$ und $\omega_2 \in \mathsf{QNR}_p$ ist $L_0(\omega_1) = 0 \neq 1 = L_0(\omega_2)$.
- Da wir nach Voraussetzung $L_1(\beta)$ effizient berechnen können, lässt sich die gesuchte Wurzel ω_j also daran erkennen, dass sie die Bedingung $L_0(\omega_j) = L_1(\beta)$ erfüllt.
- Da nach Voraussetzung auch $L_1(\omega_j)$ effizient berechenbar und $L_2(\beta) = L_1(\omega_j)$ ist, ist auch $L_2(\beta)$ effizient berechenbar.

Durch die effiziente Berechnung von $L_2(\beta)$ haben wir wegen $L_{i+1}(\beta) = L_i(\omega_j)$ für $i = 0, \ldots, r-1$ das Problem, die Bits $L_r(\beta), \ldots, L_2(\beta)$ zu bestimmen, auf das Problem reduziert, die Bits $L_{r-1}(\omega_j), \ldots, L_2(\omega_j)$ zu bestimmen. Indem wir dies wiederholen, haben wir spätestens nach $r-1$ Iterationen sämtliche Bits von a bestimmt. Der folgende Algorithmus fasst diese Vorgehensweise zusammen.

Reduktion der Berechnung von $a = \log_{p,\alpha}(\beta)$ auf die Berechnung von $L_1(\beta)$

```
1  a_0 := L_0(β)
2  β_1 := βα^{-a_0} mod p
3  i := 1
4  while β_i ≠ 1 do
5     a_i := L_1(β_i)
6     ω_i := β_i^{(p+1)/4}
7     if a_i = 0 then δ_i := ω_i else δ_i := p − ω_i
8     β_{i+1} := δ_i α^{-a_i} mod p
9     i := i + 1
10 return(a_{i-1} ··· a_0)_2
```

Beispiel 156. *Sei $p = 19$, $\alpha = 2$ und $\beta = 6$. Dann berechnet der Algorithmus die folgenden Werte.*

i	a_i	ω_i	δ_i	β_{i+1}
0	0	-	-	6
1	1	5	14	7
2	1	11	8	4
3	1	17	2	1

Die Ausgabe ist also $(a_3 \cdots a_0)_2 = (1110)_2 = 14$. ◁