Vorlesungsskript

Einführung in die Kryptologie

Wintersemester 2013/14

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

14. Februar 2014

Inhaltsverzeichnis

1 Klassische Verfahren

1.1 Einführung

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

Physikalische Maßnahmen: Tresor etc.

Organisatorische Maßnahmen: einsamer Waldspaziergang etc.

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

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

Kryptoanalyse: Analyse eines Kryptoverfahrens zum Zweck der Bewertung seiner 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 (**Alphabet**)**.** *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* $A^* = \bigcup_{n \geq 0} A^n$*.*

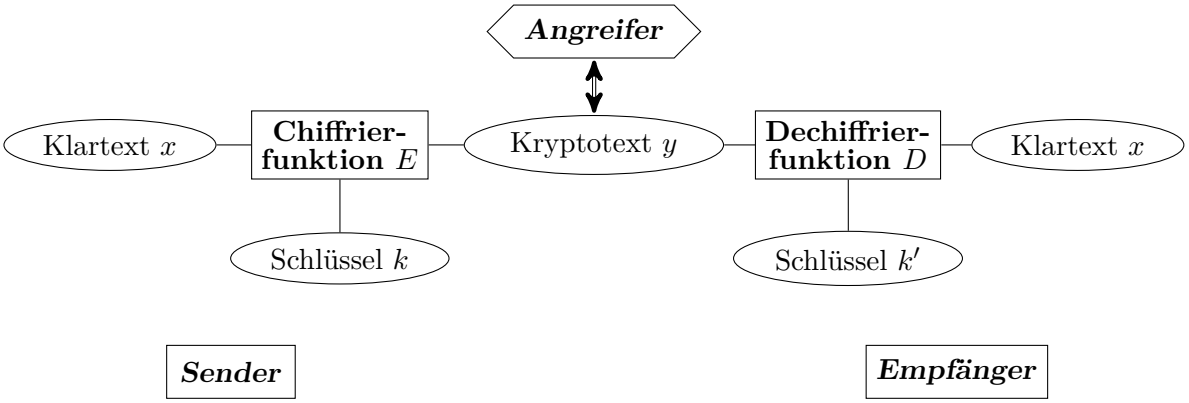
Beispiel 3. *Das* ***lateinische Alphabet*** A_{lat} *enthält die 26 Buchstaben* `A,...,Z`*. Bei der Abfassung von Klartexten wurde meist auf den Gebrauch von Interpunktions- und Leerzeichen sowie auf Groß- und Kleinschreibung verzichtet (*$\rightsquigarrow$ *Verringerung der Redundanz im Klartext).*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

Definition 4 (**Kryptosystem**)**.** *Ein* ***Kryptosystem*** *wird durch folgende Komponenten beschrieben:*

- A*, das* ***Klartextalphabet****,*
- B*, das* ***Kryptotextalphabet****,*
- K*, der* ***Schlüsselraum*** *(key space),*
- $M \subseteq A^*$*, der* ***Klartextraum*** *(message space),*
- $C \subseteq B^*$*, der* ***Kryptotextraum*** *(ciphertext space),*
- $E : K \times M \to C$*, die* ***Verschlüsselungsfunktion*** *(encryption function),*
- $D : K \times C \to M$*, die* ***Entschlüsselungsfunktion*** *(decryption function) und*
- $S \subseteq K \times K$*, eine Menge von Schlüsselpaaren* (k, k') *mit der Eigenschaft, dass für jeden Klartext* $x \in M$ *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 unterschiedliche Klartexte $x_1 \neq x_2$ ist $E(k, x_1) = E(k, x_2)$. Dann folgt

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

im Widerspruch zu (1.1). □

1.3 Die affine Chiffre

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

Definition 6 (**teilt-Relation, modulare Kongruenz**). *Seien a, b, m ganze Zahlen mit $m \geq 1$. Die Zahl a **teilt** b (kurz: $a|b$), falls ein $d \in \mathbb{Z}$ existiert mit $b = ad$. Teilt m die Differenz $a - b$, so schreiben wir hierfür*

$$a \equiv_m b$$

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

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

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

Die auf $\mathbb{Z}$ definierten Operationen

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

und

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

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*

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 $a \ominus_m b$.

Durch Identifikation der Buchstaben a_i 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*

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

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

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

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

Im Fall des lateinischen Alphabets führt der Schlüssel $k = 13$ auf eine interessante Chiffrierfunktion, die in UNIX-Umgebungen auch unter der Bezeichnung ROT13 bekannt ist (siehe Tabelle 1.3). Natürlich kann mit dieser Substitution nicht ernsthaft die Vertraulichkeit von Nachrichten geschützt werden. Vielmehr soll durch sie ein unbeabsichtigtes Mitlesen – etwa von Rätsellösungen – verhindert werden.

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

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

Da ROT13 zudem keinen Buchstaben auf sich selbst abbildet, ist sie sogar eine echt involutorische Abbildung.

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

Definition 9 (**ggT, kgV, teilerfremd**). *Seien $a, b \in \mathbb{Z}$. Für $(a, b) \neq (0, 0)$ ist*

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

*der **größte gemeinsame Teiler** von a und b. 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 $b\neq 0$. Dann gilt* $\mathrm{ggT}(a,b)=\mathrm{ggT}(b,a+bc)$.

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

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

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

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

mit

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

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	$\cdot$ **21**	+ 0

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

Der Euklidsche 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:

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

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

Durch Induktion über $i = s, s-1, \ldots, 0$ folgt $r_i \geq F_{s+1-i}$; also $a = r_0 \geq F_{s+1}$. 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. *Der Euklidsche Algorithmus führt $O(n)$ Divisionsschritte zur Berechnung von* $\mathrm{ggT}(a, b)$ *durch, wobei n die Länge der Eingabe $a > b > 0$ in Binärdarstellung bezeichnet. Dies führt auf eine Zeitkomplexität von $O(n^3)$, da jede Ganzzahldivision in Zeit $O(n^2)$ durchführbar ist.*

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

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

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

$$p_i = p_{i-2} - d_i p_{i-1}, \quad \text{wobei} \quad p_0 = 1 \quad \text{und} \quad p_1 = 0,$$

und

$$q_i = q_{i-2} - d_i q_{i-1}, \quad \text{wobei} \quad q_0 = 0 \quad \text{und} \quad q_1 = 1,$$

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

$$ap_i + bq_i = r_i,$$

und durch Induktion über i,

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

zeigt man, dass dies auch für $i = 2, \ldots, s$ gilt. Insbesondere gilt also

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

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

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

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	$p_i\cdot 693 +$	$q_i\cdot 147 =$	r_i
0						1	0	$1\cdot 693 +$	$0\cdot 147 =$	693
1	693	$=$	4	$\cdot 147 +$	105	0	1	$0\cdot 693 +$	$1\cdot 147 =$	147
2	147	$=$	1	$\cdot 105 +$	42	1	-4	$1\cdot 693 -$	$4\cdot 147 =$	105
3	105	$=$	2	$\cdot\ 42 +$	21	-1	5	$-1\cdot 693 +$	$5\cdot 147 =$	42
4	42	$=$	2	$\cdot\ \mathbf{21} +$	0	**3**	$\mathbf{-14}$	$3\cdot 693 -$	$14\cdot 147 =$	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. $\operatorname{ggT}(a,b) = \min\{\lambda a + \mu b \geq 1 \mid \lambda, \mu \in \mathbb{Z}\}$.

Beweis. Sei $M = \{\lambda a + \mu b \geq 1 \mid \lambda, \mu \in \mathbb{Z}\}$, $m = \min M$ und $g = \operatorname{ggT}(a,b)$. 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 16. *Der größte gemeinsame Teiler von* a *und* b *wird von allen gemeinsamen Teilern von* a *und* b *geteilt,*

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

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

Korollar 17 (Lemma von Euklid)**.** *Teilt* a *das Produkt* bc *und sind* a, b *teilerfremd, so teilt* a *auch* c,

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

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

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

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

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 15, dass auch ab teilerfremd zu m ist. □

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

Satz 19. *Sei $m \geq 1$. 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 $x \not\equiv_m x'$. 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 $x_1 \equiv_m x_2$.
Dies zeigt, dass die Abbildung $f : \mathbb{Z}_m \to \mathbb{Z}_m$ mit $f(x) = bx \bmod m$ injektiv ist. Da jedoch Definitions- und Wertebereich von f identisch sind, muss f dann auch surjektiv sein. Dies impliziert, dass die Kongruenz $bx \equiv_m y$ für jedes $y \in \mathbb{Z}_m$ lösbar ist. □

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. Die invertierbaren Elemente von $\mathbb{Z}_m$ werden in der Menge*

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

zusammengefasst.

Korollar 18 zeigt, dass $\mathbb{Z}_m^*$ unter der Operation $\odot_m$ abgeschlossen ist, und mit Korollar 20 folgt, dass $(\mathbb{Z}_m^*, \odot_m)$ eine multiplikative Gruppe bildet. Allgemeiner zeigt man, dass für einen beliebigen Ring $(R, +, \cdot, 0, 1)$ mit Eins die Multiplikation auf der Menge $R^* = \{a \in R \mid \exists b \in R : ab = 1 = ba\}$ aller **Einheiten** von R eine Gruppe $(R^*, \cdot, 1)$ (die so genannte **Einheitengruppe** von R) bildet.

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

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

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

Definition 21 (affine Chiffre). *Bei der **affinen Chiffre** ist $A = B = M = C$ ein beliebiges Alphabet mit $m := \|A\| > 1$ und $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$. Für $k = (b, c) \in K$, $x \in M$ und $y \in C$ gilt*

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

In diesem Fall liefert die Schlüsselkomponente $b = -1$ für jeden Wert von c eine involutorische Chiffrierfunktion $x \mapsto E(b, c; x) = c - x$ (**verschobenes komplementäres Alphabet**). Wählen wir für c ebenfalls den Wert -1, so ergibt sich die Chiffrierfunktion $x \mapsto -x - 1$, 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 den Klartextbuchstaben* $x = \mathsf{F}$ *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 einen Kryptotextbuchstaben wieder entschlüsseln zu können, benötigen wir das multiplikative Inverse von* $b = 9$*, das sich wegen*

i	r_{i-1}	$=$	$d_{i+1} \cdot r_i$	$+$	r_{i+1}	$p_i \cdot 26$	$+$	$q_i \cdot 9$	$=$	r_i
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 den Kryptotextbuchstaben* $y = \mathsf{V}$ *den ursprünglichen Klartextbuchstaben*

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

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

Eine wichtige Rolle spielt die Funktion

$$\varphi : \mathcal{N} \to \mathcal{N} \quad \text{mit} \quad \varphi(n) = \|\mathbb{Z}_n^*\| = \|\{a \mid 0 \leq a \leq n-1,\ \mathrm{ggT}(a, n) = 1\}\|,$$

die sogenannte *Eulersche* φ*-Funktion.*

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

Wegen

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

folgt sofort

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

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

◁

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

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

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

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

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

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

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

Beweis. Es gilt

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

□

Der Chinesische Restsatz

Die beiden linearen Kongruenzen

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

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

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

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

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

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

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

Dann gilt

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

und

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

für $j \neq i$. Folglich erfüllt $x = \sum_{j=1}^{k} \mu_j n_j b_j$ die Kongruenzen

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

für $i = 1, \ldots, k$. Dies zeigt, dass (1.2) lösbar, also die Funktion

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

mit $f(x) = (x \bmod m_1, \ldots, x \bmod m_k)$ surjektiv ist. Da der Definitions- und der Wertebereich von f die gleiche Mächtigkeit haben, muss f jedoch auch injektiv sein, d.h. (1.2) ist sogar eindeutig lösbar. □

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

1.4 Die Hill-Chiffre

Die von Hill im Jahr 1929 publizierte Chiffre ist eine Erweiterung der multiplikativen Chiffre auf Buchstabenblöcke, d.h. der Klartext wird nicht zeichenweise, sondern blockweise verarbeitet. Sowohl der Klartext- als auch der Kryptotextraum enthält alle Wörter x

über A einer festen Länge l. Zur Chiffrierung wird eine $(l \times l)$-Matrix $k = (k_{ij})$ mit Koeffizienten in $\mathbb{Z}_m$ benutzt, die einen Klartextblock $x = x_1 \dots x_l \in A^l$ in den Kryptotextblock $y_1 \dots y_l \in A^l$ transformiert, wobei

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

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

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

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

Definition 27 (Determinante). *Sei R ein kommutativer Ring mit Eins und sei $A = (a_{ij}) \in R^{l \times l}$. Für $1 \le i, j \le l$ sei A_{ij} die durch Streichen der i-ten Zeile und j-ten Spalte aus A hervorgehende Matrix. Die **Determinante** von A ist dann $det(A) = a_{11}$, falls $l = 1$, und*

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

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

Die Determinantenfunktion ist durch die drei Eigenschaften multilinear, alternierend und normiert eindeutig festgelegt. Sei $f : R^{n \times n} \to R$ eine Funktion.

- f heißt ***multilinear***, falls für jede Matrix $A = (a_1, \dots, a_n) \in R^{n \times n}$ mit Spalten $a_1, \dots, a_n \in (R^n)^T$, jeden Spaltenvektor $b \in (R^n)^T$ und jedes $r \in R$

  $$f(a_1, \dots, ra_i + b, \dots, a_n) = rf(a_1, \dots, a_i, \dots, a_n) + f(a_1, \dots, b, \dots, a_n)$$

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

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

Satz 28. *Sei A ein Alphabet und sei $k \in \mathbb{Z}_m^{l \times l}$ ($l \ge 1$, $m = \|A\|$). Die Abbildung $f : A^l \to A^l$ mit*

$$f(x) = xk,$$

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

Beweis. Siehe Übungen. □

Definition 29 (Hill-Chiffre). *Sei $A = \{a_0, \dots, a_{m-1}\}$ ein beliebiges Alphabet und für eine natürliche Zahl $l \ge 2$ sei $M = C = A^l$. Bei der **Hill-Chiffre** ist $K = \{k \in \mathbb{Z}_m^{l \times l} \mid \mathrm{ggT}(det(k), m) = 1\}$ und es gilt*

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

Beispiel 30 (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*

$$(\mathtt{HILL}) \begin{pmatrix} 11 & 13 & 8 & 21 \\ 24 & 17 & 3 & 25 \\ 18 & 12 & 23 & 17 \\ 6 & 15 & 2 & 15 \end{pmatrix} = (\mathit{NERX}) \quad \textit{bzw.} \quad \begin{aligned} 11\,\mathtt{H} + 24\,\mathtt{I} + 18\,\mathtt{L} + 6\,\mathtt{L} &= \mathit{N} \\ 13\,\mathtt{H} + 17\,\mathtt{I} + 12\,\mathtt{L} + 15\,\mathtt{L} &= \mathit{E} \\ 8\,\mathtt{H} + 3\,\mathtt{I} + 23\,\mathtt{L} + 2\,\mathtt{L} &= \mathit{R} \\ 21\,\mathtt{H} + 25\,\mathtt{I} + 17\,\mathtt{L} + 15\,\mathtt{L} &= \mathit{X} \end{aligned}$$

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

1.5 Die Vigenère-Chiffre und andere Stromsysteme

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

Definition 31 (**Vigenère-Chiffre**)**.** *Sei $A = B$ ein beliebiges Alphabet. Die **Vigenère-Chiffre** chiffriert unter einem Schlüssel $k = k_0 \dots k_{d-1} \in K = A^*$ einen Klartext $x = x_0 \dots x_{n-1}$ beliebiger Länge zu*

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

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

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

Beispiel 32 (Vigenère-Chiffre)**.** *Verwenden wir das lateinische Alphabet A_{lat} als Klartextalpabet und wählen wir als Schlüssel das Wort $k = \mathit{WIE}$, so ergibt sich für den Klartext* VIGENERE *beispielsweise der Kryptotext*

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

◁

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

Beispiel 33. *Vigenère-Chiffre*

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

Chiffrierung:

	`WIEWIEWI`	(*Schlüsselstrom*)
−	`BEAUFORT`	(*Klartext*)
	`XMEQNSNB`	(*Kryptotext*)

Dechiffrierung:

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

Klartext-Schlüsselstrom:

	`VIGENERE`	(*Klartext*)
+	`WIEVIGEN`	(*Schlüsselstrom*)
	`RQKZVKVR`	(*Kryptotext*)

Kryptotext-Schlüsselstrom:

	`VIGENERE`	(*Klartext*)
+	`WIERQKVD`	(*Schlüsselstrom*)
	`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.

1.6 Der One-Time-Pad

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

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

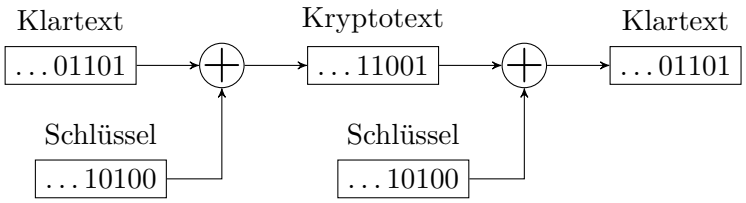
$$y = y_0 \ldots y_{n-1}, \text{ wobei } y_i = x_i + k_i.$$

◁

Der Klartext wird also wie bei einer additiven Chiffre verschlüsselt, nur dass der Schlüssel nach einmaligem Gebrauch gewechselt wird. Dies entspricht dem Gebrauch einer Vigenère-Chiffre, falls als Schlüssel ein zufällig gewähltes Wort von der Länge des Klartextes benutzt wird. Wie diese ist der One-time-pad im Binärfall also involutorisch.

[†] 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.

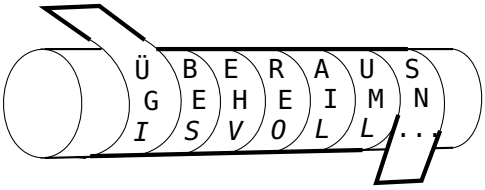
[‡] Diese Art der Schlüsselerzeugung schlug der amerikanische Major Joseph O. Mauborgne im Jahr 1918 vor, nachdem ihm ein von Gilbert S. Vernam für den Fernschreibverkehr entwickeltes Chiffriersystem vorgestellt wurde.



1.7 Klassifikation von Kryptosystemen

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

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



ÜBERAUS GEHEIMNISVOLL ...
⇝ *ÜGI ... BES ... EHV ... REO ... AIL ... UML ... SN ...*

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

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

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

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

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

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

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

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

Beispiel 38. *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*

◁

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

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

und

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

gilt.

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

Beispiel 40. *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 Zyklen $(i_1\ i_2\ i_3\ \ldots\ i_n)$ dargestellt, wobei i_1 auf i_2, i_2 auf i_3 usw. und schließlich i_3 auf i_1 abgebildet wird, so ist π^{-1} sehr leicht zu bestimmen.

Beispiel 41.

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

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

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

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

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

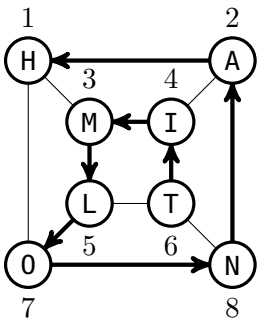
Beispiel 42. *Bei der* ***Matrix-Transposition*** *wird der Klartext zeilenweise in eine $k \times m$-Matrix eingelesen und der Kryptotext spaltenweise gemäß einer Spaltenpermutation π, die als Schlüssel dient, wieder ausgelesen. Für $\pi = (1\ 4\ 3)\ (2\ 6)$ wird also zuerst Spalte $\pi(1) = 4$, dann Spalte $\pi(2) = 6$ und danach Spalte $\pi(3) = 1$ usw. und zuletzt Spalte $\pi(6) = 2$ ausgelesen.*

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

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

◁

Beispiel 43. *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 ⇝ *TIMLONAH*

◁

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

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

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

DIE BLOCKLAENGE IST SECHS ⇝ EDBOIL LCANKE IGSSET EXCSYH

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

Wir wenden uns nun der Klassifikation von Substitutionschiffren 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 polygrafische Substitution, die auf Buchstabenpaaren operiert, wird **digrafisch** genannt. Das älteste bekannte polygrafische Chiffrierverfahren wurde von Giovanni Porta im Jahr 1563 veröffentlicht. Dabei werden je zwei aufeinanderfolgende Klartextbuchstaben durch ein einzelnes Kryptotextzeichen ersetzt.

Beispiel 44. *Bei der* ***Porta-Chiffre*** *werden 400 (!) unterschiedliche von Porta für diesen Zweck entworfene Kryptotextzeichen verwendet. Diese sind in einer* 20×20*-Matrix* $M = (y_{ij})$ *angeordnet, deren Zeilen und Spalten mit den* 20 *Klartextbuchstaben* $\mathsf{A}, \ldots, \mathsf{I}, \mathsf{L}, \ldots, \mathsf{T}, \mathsf{V}, \mathsf{Z}$ *indiziert sind. Zur Ersetzung des Buchstabenpaars* $a_i a_j$ *wird das in Zeile* i *und Spalte* j *befindliche Kryptotextzeichen*

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

benutzt. ◁

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

Ein frühes (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 ⇝ 30 24 21 43 01 13 24 33

Bei der **Polybios-Chiffre** dient eine 5×5-Matrix, die aus sämtlichen Klartextbuchstaben gebildet wird, als Schlüssel.[§] Die Verschlüsselung des Klartextes erfolgt buchstabenweise,

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

indem man einen in Zeile i und Spalte j eingetragenen Klartextbuchstaben 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 die einzelnen Klartextsegment unabhängig von ihrer Position im Klartext.

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

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

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

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

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

Definition 45 (**Blockchiffre**). *Sei A ein beliebiges Alphabet und es gelte $M = C = A^l$, $l \geq 1$. Eine **Blockchiffre** realisiert für jeden Schlüssel $k \in K$ eine bijektive Abbildung g auf A^l und es gilt*

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

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

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

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

Definition 46 (**Stromchiffre**). *Sei A ein beliebiges Alphabet und sei $M = C = A^l$ für eine natürliche Zahl $l \geq 1$. Weiterhin seien K und $\hat{K}$ Schlüsselräume. Eine **Stromchiffre** wird durch eine Verschlüsselungsfunktion $E : \hat{K} \times M \to C$ und einen Schlüsselstromgenerator $g : K \times A^* \to \hat{K}$ beschrieben. Der Generator g erzeugt aus einem externen*

Schlüssel $k \in K$ für einen Klartext $x = x_0 \dots x_{n-1}$, $x_i \in M$, 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).

Gespreizte Substitutionen

Bei den bisher betrachteten Substitutionen haben die einzelnen Blöcke, aus denen der Kryptotext zusammengesetzt wird, eine einheitliche Länge. Es liegt nahe, einem Gegner die unbefugte Rekonstruktion des Klartextes dadurch zu erschweren, dass man Blöcke unterschiedlicher Länge verwendet. Man spricht hierbei auch von einer **Spreizung** (*straddling*) des Kryptotextalphabets. Ein bekanntes Beispiel für diese Technik ist die sogenannte Spionage-Chiffre, die vorzugsweise von der ehemaligen sowjetischen Geheimpolizei NKWD (*Naródny Komissariàt Wnutrennich Del*; zu deutsch: Volkskommissariat des Innern) benutzt wurde.

Beispiel 47. *Bei der **Spionage-Chiffre** wird in die erste Zeile einer 3×10-Matrix ein Schlüsselwort w geschrieben, welches keinen Buchstaben mehrfach enthält und eine Länge von 6 bis 8 Zeichen hat (also zum Beispiel* `SPIONAGE`*). Danach werden die anderen beiden Zeilen der Matrix mit den restlichen Klartextbuchstaben (etwa in alphabetischer Reihenfolge) gefüllt.*

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

GESPREIZT
 ⇝ *2 7 4 1 5 4 7 9 5 7 5 1*

◁

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

Verwendung von Blendern und Homophonen

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

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

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

Definition 48 (**homophonen Substitutionschiffre**)**.** *Sei A ein Klartextalphabet und sei $M = A$. Weiter sei C ein Kryptotextraum der Größe $\|C\| > \|A\| = m$. In einer (einfachen)* ***homophonen Substitutionschiffre*** *beschreibt jeder Schlüssel $k \in K$ eine Zerlegung von C in m disjunkte Mengen $H(a)$, $a \in A$.*

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

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

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

Beispiel 49 (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 Klartextbuchstaben zerstört wird. Dieser Effekt kann dadurch noch verstärkt werden, dass man für häufig vorkommende Klartextzeichen a eine entsprechend größere Menge $H(a)$ an Homophonen vorsieht. Damit lässt sich erreichen, dass die Verteilung der im Geheimtext auftretenden Zeichen weitgehend nivelliert wird.

Beispiel 50 (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* $p(\mathsf{A}) = 0.0647$ *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

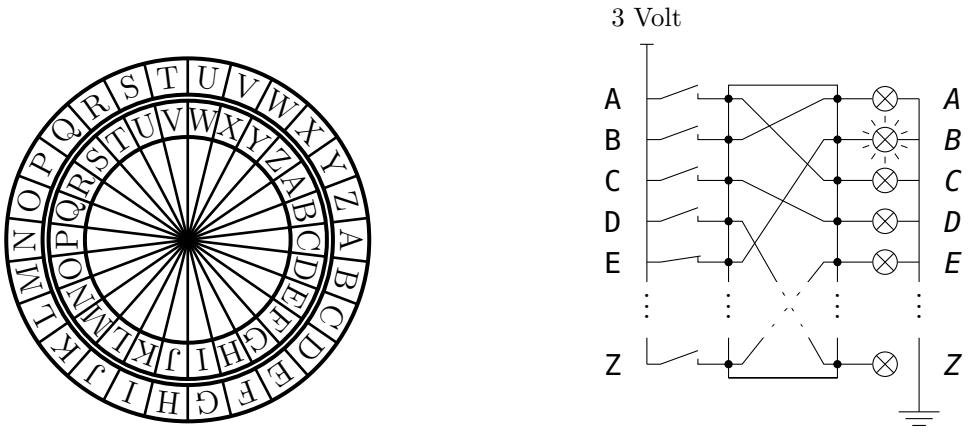
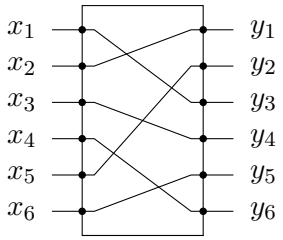


Abbildung 1.1: Realisierung von einfachen Substitutionen mit einer Drehscheibe und mit Hilfe von Steckverbindungen.

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

1.8 Realisierung von Blocktranspositionen und einfachen Substitutionen

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



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

Blocktranspositionen können auch leicht durch Software als eine Folge von Zuweisungen

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

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 1.1 ist gezeigt, wie sich mit einer solchen Drehscheibe beispielsweise die additive Chiffre realisieren lässt. Zur Einstellung des Schlüssels k müssen die Scheiben so gegeneinander verdreht werden,

dass der Schlüsselbuchstabe a_k auf der inneren Scheibe mit dem Klartextzeichen $a_0 = \mathsf{A}$ auf der äußeren Scheibe zur Deckung kommt. Auf der Drehscheibe in Abbildung 1.1 ist beispielsweise der Schlüssel $k = 3$ eingestellt, das heißt, $a_k = \mathsf{D}$. Die Verschlüsselung geschieht nun durch bloßes Ablesen der zugehörigen Kryptotextzeichen auf der inneren Scheibe, so dass von der Drehfunktion der Scheiben nur bei einem Schlüsselwechsel Gebrauch gemacht wird.

Aufgrund ihrer engen Verwandtschaft mit der Klasse der Blocktranspositionen lassen sich einfache Substitutionen auch mit Hilfe einer P-Box realisieren (vergleiche Abbildung). Hierfür können beispielsweise zwei Steckkontaktleisten verwendet werden. Der aktuelle Schlüssel wird in diesem Fall durch Verbinden der entsprechenden Kontakte mit elektrischen Kabeln eingestellt (siehe Abbildung 1.1). Um etwa den Klartextbuchstaben E zu verschlüsseln, drückt man auf die entsprechende Taste, und das zugehörige Kryptotextzeichen *B* wird im selben Moment durch ein aufleuchtendes Lämpchen signalisiert.

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

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

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

2 Kryptoanalyse der klassischen Verfahren

2.1 Klassifikation von Angriffen gegen Kryptosysteme

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

Goldene Regel für Kryptosystem-Designer (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 davon, welchen unterschiedlichen Bedrohungen ein Kryptosystem im praktischen Einsatz ausgesetzt sein kann.

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

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

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

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

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

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

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

2.2 Kryptoanalyse von einfachen Substitutionschiffren

Durch eine Häufigkeitsanalyse können insbesondere einfache Substitutionen g leicht gebrochen werden, sofern die einzelnen Buchstaben a in der benutzten Klartextsprache mit voneinander differierenden Häufigkeiten $p(a)$ auftreten (vergleiche Tabelle 2.1). Selbst wenn, was insbesondere bei kurzen Texten zu erwarten ist, die tatsächliche Häufigkeitsverteilung nur in etwa der vom Gegner angenommenen Verteilung entspricht, reduziert sich dadurch die Zahl der in Frage kommenden einfachen Substitutionen ganz erheblich. Berechnet man die relativen Häufigkeiten h der Kryptotextbuchstaben im Kryptotext, so gilt $p(a) \approx h(g(a))$ (vorausgesetzt der Kryptotext ist genügend lang). Für die Schilderung einer nach dieser Methode durchgeführten Kryptoanalse sei auf die Erzählung „Der Goldkäfer“ von Edgar Allan Poe verwiesen.

Tabelle 2.1: Einteilung von Buchstaben in Cliquen mit vergleichbaren Häufigkeitswerten.

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

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

Beispiel 51 (vollständige Schlüsselsuche)**.** *Es sei bekannt, dass das Kryptotextstück* $y =$ `SAXP` *mit einer additiven Chiffre erzeugt wurde* ($K = A = B = A_{lat}$)*. Entschlüsseln wir* y *probeweise mit allen möglichen Schlüsselwerten, so erhalten wir folgende Zeichenketten.*

k	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*
$D(k,y)$	RZWO	QYVN	PXUM	OWTL	NVSK	MURJ	LTQI	KSPH	JROG	IQNF	HPME	GOLD

N	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
FNKC	EMJB	DLIA	CKHZ	BJGY	AIFX	ZHEW	YGDV	XFCU	WEBT	VDAS	UCZR	TBYQ

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

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

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

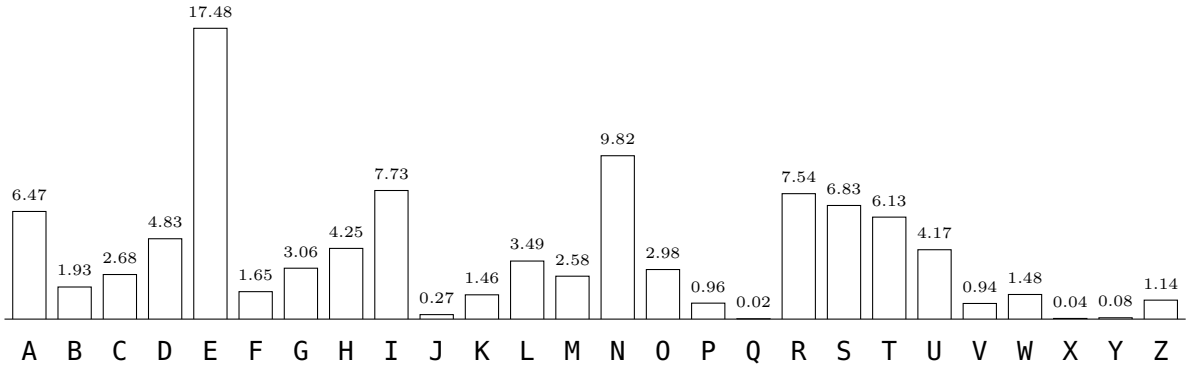


Abbildung 2.1: Häufigkeitsverteilung der Einzelbuchstaben im Deutschen (in %).

Die Abbildungen 2.1, 2.2 und 2.3, zeigen typische Verteilungen von Einzelbuchstaben in der deutschen, englischen und französischen Sprache (ohne Berücksichtigung von

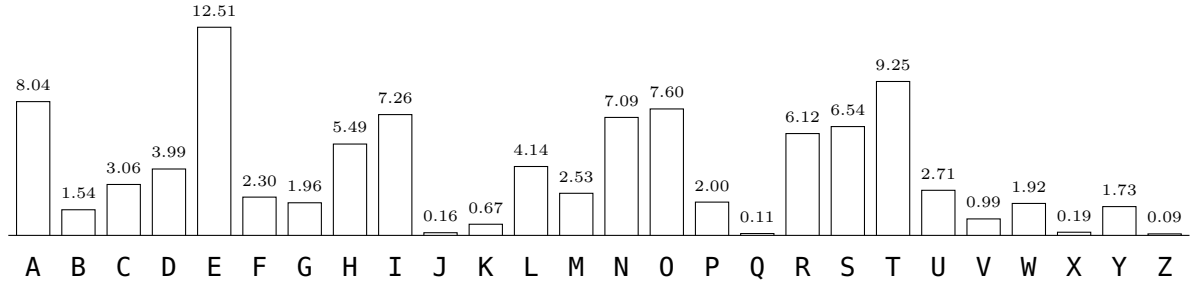


Abbildung 2.2: Häufigkeitsverteilung der Buchstaben im Englischen (in %).

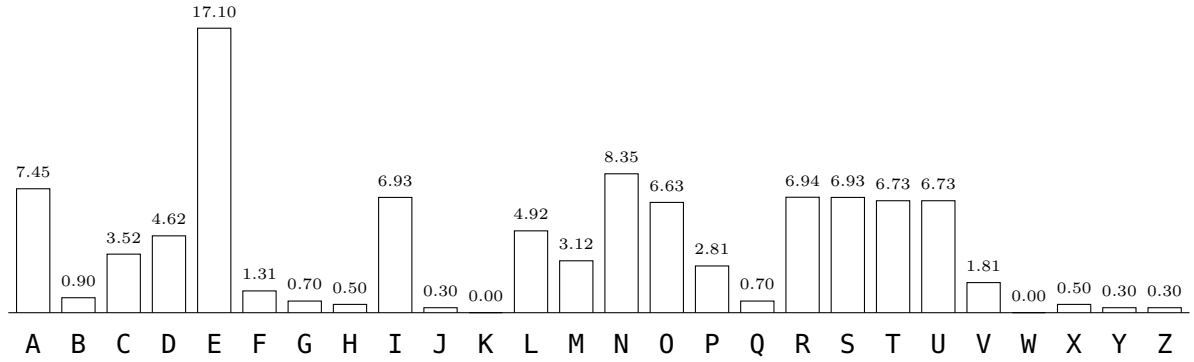


Abbildung 2.3: Häufigkeitsverteilung der Buchstaben im Französischen (in %).

Interpunktions- und Leerzeichen). Ein typischer deutscher Text besteht demnach zu 62% aus den sieben häufigsten Zeichen E, N, I, R, S, A, T (das sind nicht einmal 27% der Klartextzeichen).

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

Beispiel 52 (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 b die (absolute) Häufigkeit $H_y(b)$ seines Auftretens in obigem Kryptotext y,

b	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
$H(b)$	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 (a, b) eine gute Übereinstimmung zwischen der Häufigkeit $H_y(b)$, mit der $b = E(k, a)$ im Kryptotext vorkommt, und der erwarteten Häufigkeit $H_{100}(a)$ auf, mit der a in einem typischen deutschen Text der Länge 100 vorkommt (die Tabelle zeigt die Werte von $H_{100}(a)$ gerundet):

b	*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*
$H_y(b)$	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
$H_{100}(a)$	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
a	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 $S = (s_{ij})$ mit l Spalten $S_1, \ldots, S_l$ ein. Da jede Zeile dieser Matrix aus dem zugehörigen Klartextblock mit derselben Permutation π erzeugt wurde, müssen die Spalten S_j jetzt nur noch in die „richtige“ Reihenfolge gebracht werden, um den gesuchten Klartext zu erhalten. Der Nachfolger S_k von S_j (bzw. der Vorgänger S_j von S_k) kann sehr gut anhand der Werte von $\hat{p}(S_j, S_k) = \sum_i p(s_{ij}, s_{ik})$ bestimmt werden.

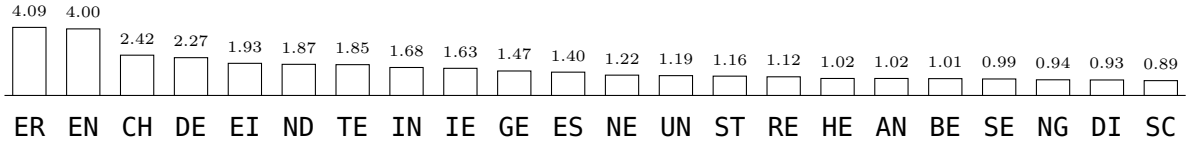


Abbildung 2.4: Die häufigsten Bigramme im Deutschen (Angaben in %).

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

Beispiel 53 (Häufigkeitsanalyse von Bigrammen)**.** *Für den mit einer Blocktransposition (mit vermuteter Blocklänge 5) erzeugten Kryptotext*

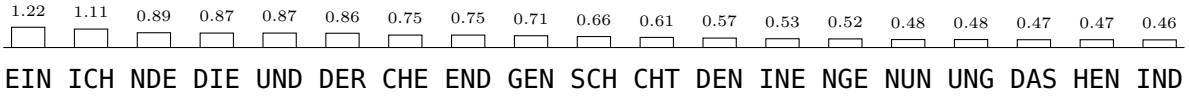


Abbildung 2.6: Die häufigsten Trigramme im Deutschen (in %).

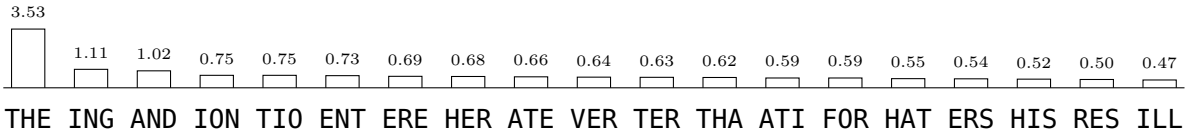


Abbildung 2.7: Die häufigsten Trigramme im Englischen (in %).

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

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

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

und erhält damit die Schlüsselmatrix k.

BK-Angriff (bekannter Klartext). Sind bei einem BK-Angriff ausreichend geeignete Klartext-Kryptotextpaare bekannt, so kann das Hill-System folgendermaßen gebrochen werden: Sind x_i, y_i $(i = 1, \ldots, \mu)$ Paare mit $x_i k = y_i$ und gilt $\mathrm{ggT}(detX, m) = 1$ 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, lassen sie sich mit statistischen Methoden ebenfalls leicht brechen, insbesondere wenn der Kryptotext im Verhältnis zur Periode d (Länge des Schlüsselwortes) genügend lang ist.

Bestimmung der Schlüsselwortlänge

Es gibt mehrere Methoden, eine Vigenère-Chiffre zu brechen, sobald die Länge des Schlüsselwortes bekannt ist. 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 Verschiebechiffre) verschlüsselt. Sie können daher einzeln wie eine additive Chiffre durch eine Häufigkeitsanalyse gebrochen werden. Hierbei liefert jede Spalte y_i einen Buchstaben k_i des Schlüsselwortes der Vigenère-Chiffre.

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 54 (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 55 (**Koinzidenzindex**). *Der* ***Koinzidenzindex*** *(engl.* ***i****ndex of* ***c****oincidence) eines Textes* y *der Länge* n *über dem Alphabet* $\mathcal{B}$ *ist definiert als*

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

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

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

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

Definition 57 (**Rauheitsgrad; Measure of Roughness**). *Der* ***Rauheitsgrad*** MR_L *einer Sprache* L *mit Einzelbuchstabenverteilung* p *ist*

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

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

Beispiel 58. *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

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

führt. Allgemein gilt

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

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

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 59. *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.

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

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

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

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

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

Beispiel 60. *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 61 (**Gegenseitiger Koinzidenzindex**). *Der* ***gegenseitge Koinzidenzindex*** *von zwei Texten y und y' mit den Längen n und n' über dem Alphabet $\mathcal{B}$ ist definiert als*

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

$IC(y, y')$ 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 Häufigkeitsverteilung 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 und $y_j + \delta$ (für $1 \leq i < j \leq d$) die Wahrscheinlichkeit an, dass man bei zufälliger Wahl einer Position in y_i und in $y_j + \delta$ denselben Buchstaben vorfindet, wobei

δ eine Verschiebung von Spalte y_j relativ zur Spalte y_i ist (mit $0 \leq \delta \leq 25$). Mit großer Wahrscheinlichkeit nimmt also $IC(y_i + \delta, y_j)$ 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 62. *Betrachten wir den Kryptotext aus vorigem Beispiel, so ergeben sich für $IC(y_i, y_j + \delta)$ 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)

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

Wir können nun alle Spalten relativ zur ersten Spalte so verschieben, dass der ganze Text eine einheitliche Verschiebung δ hat, also die zweite Spalte um -7, die dritte um -8 und die vierte um -23. Für die Bestimmung von δ, muss man nur den häufigsten Buchstaben in dem auf diese Weise erzeugten Text bestimmen (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 so vorgehen: Man geht zunächst davon aus, dass jeder Kryptotextbuchstabe durch Summation eines Klartext- und Schlüsselstrombuchstabens mit jeweils mittlerer bis hoher Wahrscheinlichkeit entstanden ist. Dies sind beispielsweise im Englischen die Buchstaben E, T, A, O, I, N, S, R, H. Zu einem Teilwort w des Kryptotextes bestimmt man dann alle Paare von Wörtern (w_1, w_2) mit $w_1 + w_2 = w$ und $w_1, w_2 \in \{\mathsf{E}, \mathsf{T}, \mathsf{A}, \mathsf{O}, \mathsf{I}, \mathsf{N}, \mathsf{S}, \mathsf{R}, \mathsf{H}\}$. 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 63. *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 Einzelbuchstaben von w:

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

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 eigentlich keinen großen kryptografischen 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 $l = 2$),

	$x_0\, x_1\, x_2\, x_3\, \underbrace{x_4\, x_5}\, x_6\, x_7\, x_8\ \ x_9\, \underbrace{x_{10}\, x_{11}}\, x_{12}\, x_{13}\, x_{14}\ \ldots$	Klartext x
$+$	$k_0\, k_1\, k_2\, k_3\, x_1\, x_2\ \ x_3\, \underbrace{x_4\, x_5}\, x_6\, x_7\, x_8\ \ x_9\ \ \underbrace{x_{10}\, x_{11}}\ldots$	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 zugrunde, die einzelne Nachrichten unter einer bestimmten Wahrscheinlichkeitsverteilung aussendet.

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

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

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

die Wahrscheinlichkeit für den Schlüssel k und

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

die Wahrscheinlichkeit für den Klartext x ist. Für einen Kryptotext y berechnet sich die Wahrscheinlichkeit zu

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

und für einen fest vorgegebenen 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 y aus dem Klartext x erzeugt wurde.

Definition 64 (**informationstheoretisch sicher**). *Ein Kryptosystem heißt unter einer Schlüsselverteilung $p(k)$* ***absolut sicher (informationstheoretisch sicher****), falls für jede Klartextverteilung $p(x)$ gilt:*

$$p(x) = p(x|y) \text{ für alle } x \in M \text{ und alle } y \in C \text{ mit } p(y) > 0.$$

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

Ist $p(x,y) > 0$, so folgt wegen $p(x|y)p(y) = p(x,y) = p(y|x)p(x)$

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

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

Beispiel 65. *Sei (M, C, E, D, K) ein Kryptosystem mit $M = \{x_1, \ldots, x_4\}$, $K = \{k_1, \ldots, k_4\}$, $C = \{y_1, \ldots, y_4\}$ und*

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

Weiter sei $p(k_1) = 1/2$, $p(k_2) = 1/4$ und $p(x_3) = p(x_4) = 1/8$. Unter der Klartextverteilung $p(x_1) = 1/2$, $p(x_2) = p(x_3) = p(x_4) = 1/6$ ergibt sich dann folgende Verteilung der Kryptotexte:

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

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, zumindest nicht unter der gegebenen Schlüsselverteilung.

Die Bedingung $p(x) = p(x|y)$ ist nach dem Satz von Bayes genau dann erfüllt, wenn $p(y) = p(y|x)$ ist. Da jedoch für jedes Paar (x, y) genau ein Schlüssel $k = k_{x,y} \in K$ mit $E(k, x) = y$ existiert, also $p(y|x) = p(k_{x,y})$ ist, ist dies äquivalent zu $p(y) = p(k_{x,y})$. Für $y = y_1$ bedeutet dies, dass alle Schlüssel $k_i = k_{x_i,y_1}$ die gleiche Wahrscheinlichkeit $p(k_i) = 1/4$ haben müssen. Eine leichte Rechnung zeigt, dass dann auch $p(y_i) = 1/4$ für $i = 1, \ldots, 4$ ist. Somit ist das betrachtete Kryptosystem genau dann absolut sicher, wenn der Schlüssel unter Gleichverteilung gewählt wird. ◁

Wie in diesem Beispiel lässt sich allgemein folgende hinreichende Bedingung für die absolute Sicherheit von Kryptosystemen zeigen.

Satz 66. *Ein Kryptosystem mit $\|M\| = \|C\| = \|K\|$, in dem es für jeden Klartext x und jeden Kryptotext y genau einen Schlüssel k mit $E(k,x) = y$ gibt, ist absolut sicher, wenn die Schlüssel unter Gleichverteilung gewählt werden.*

Beweis. Bezeichne $k_{x,y}$ den eindeutig bestimmten Schlüssel, der den Klartext x auf den Kryptotext y abbildet. Wegen $p(k_{x,y}) = \|K\|^{-1}$ für alle x, y folgt zunächst

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

und

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

also $p(x|y) = p(x)p(y|x)/p(y) = p(x)$. □

In den Übungen wird gezeigt, dass auch die Umkehrung dieses Satzes gilt.

Verwendet man beim One-time-pad nur Klartexte einer festen Länge n, d. h. $M \subseteq A^n$, so ist dieser nach obigem Satz absolut sicher (vorausgesetzt, der Schlüssel wird rein zufällig, also unter Gleichverteilung gewählt). 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 M haben muss (siehe Übungen), ist der Aufwand extrem hoch. Vor der Kommunikation muss ein Schlüssel, dessen Länge der des zu übertragenden Klartextes entspricht, zufällig generiert und zwischen den Partnern auf einem sicheren Kanal ausgetauscht werden. Wird hingegen keine absolute Sicherheit angestrebt, so kann der Schlüsselstrom auch von einem Pseudo-Zufallsgenerator erzeugt werden. Dieser erhält als Eingabe eine Zufallsfolge s_0 (den sogenannten *Keim*) und erzeugt daraus eine lange Folge $v_0\, v_1 \ldots$ von Pseudo-Zufallszahlen. Als Schlüssel muss jetzt nur noch das Wort s_0 ausgetauscht werden.

In der Informationstheorie wird die Unsicherheit, mit der eine durch X beschriebene Quelle ihre Nachrichten aussendet, nach ihrer Entropie bemessen. Das heißt, die Unsicherheit über X entspricht genau dem Informationsgewinn, der sich aus der Beobachtung der Quelle X ziehen lässt. Dabei wird die in einer einzelnen Nachricht x steckende Information um so höher bemessen, je seltener x auftritt. 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 $Inf_X(x) = 0$. Dieser Wert des Informationsgehalts ergibt sich zwangsläufig aus den beiden folgenden Forderungen:

- Der gemeinsame Informationsgehalt $Inf_{X,Y}(x, y)$ von zwei Nachrichten x und y, die aus stochastisch unabhängigen Quellen X und Y stammen, sollte gleich $Inf_X(x) + Inf_Y(y)$ sein;
- der Informationsgehalt einer Nachricht, die mit Wahrscheinlichkeit $1/2$ auftritt, soll genau 1 (bit) betragen.

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

Definition 67 (**Entropie**). *Sei X eine Zufallsvariable mit Wertebereich $W(X) = \{x_1, \ldots, x_n\}$ und sei $p_i = \Pr[X = x_i]$. Dann ist die **Entropie** von X definiert als*

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

Beispiel 68. *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

$$\mathcal{H}(X) = 1/4 \cdot (2 + 2) + 1/8 \cdot (3 + 3 + 3) + 1/16 \cdot (4 + 4) = 2.625.$$

◁

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

Definition 69 (**konkav**). *Eine reellwertige Funktion f ist **konkav** auf einem Intervall 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 70. *Die Funktion $f(x) = \log_2(x)$ ist streng konkav auf $(0, \infty)$.* ◁

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

Satz 71 (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 72. *Sei X eine Zufallsvariable mit endlichem Wertebereich $W(X) = \{x_1, \ldots, x_n\}$ und Verteilung $\Pr[X = x_i] = p_i$, $i = 1, \ldots, n$. Dann ist $H(X) \leq \log_2(n)$, wobei Gleichheit genau im Fall $p_i = 1/n$ für $i = 1, \ldots, n$ eintritt.*

Beweis. Es gilt

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

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

Eine wichtige Eigenschaft der Entropie ist, dass sie eine untere Schranke für die mittlere Codewortlänge von Binärcodes bildet. Ein **Binärcode** für X ist eine (geordnete) Menge $C = \{y_1, \ldots, y_n\}$ von binären Codewörtern y_i für die Nachrichten x_i mit der Eigenschaft, dass die Abbildung $c : M^* \to \{0,1\}^*$ 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 73. *Sei X die Zufallsvariable aus dem letzten Beispiel. Betrachten wir die beiden Codes $C_1 = \{\texttt{001}, \texttt{010}, \texttt{011}, \texttt{100}, \texttt{101}, \texttt{110}, \texttt{111}\}$ und $C_2 = \{\texttt{00}, \texttt{01}, \texttt{100}, \texttt{101}, \texttt{110}, \texttt{1110}, \texttt{1111}\}$, so erhalten wir für die mittlere Codewortlänge von C_1 den Wert $L(C_1) = 3$, während C_2 wegen $|y_i| = \log_2(1/p_i)$ den Wert $L(C_2) = \mathcal{H}(X)$ erreicht und somit optimal ist.* ◁

Die Redundanz eines Codes für eine Zufallsvariable X ist um so höher, je größer seine mittlere Codewortlänge im Vergleich zur Entropie von X ist. Um auch Codes über unterschiedlichen Alphabeten miteinander vergleichen zu können, ist es notwendig, die Codewortlänge in einer festen Einheit anzugeben. Hierzu berechnet man die **Bitlänge** eines Wortes x über einem Alphabet A mit $m > 2$ Buchstaben zu $|x|_2 = |x| \log_2(m)$. Beispielsweise ist die Bitlänge von `GOLD` (über dem lateinischen Alphabet) $|\texttt{GOLD}|_2 = 4 \log_2(26) = 18,8$. Entsprechend berechnet sich für einen Code $C = \{y_1, \ldots, y_n\}$ 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 74 (**Redundanz**). *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 75. *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. ◁

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

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

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

Definition 76 (**Entropie einer Sprache**)**.** *Für eine Sprache L über einem Alphabet A mit $\|A\| = m$ und n-Gramm-Verteilung L_n ist $\mathcal{H}(L_n)/n$ die **Entropie von** L_n (pro Buchstabe). Falls dieser Wert für n gegen ∞ 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 $\mathcal{R}(L_n)$ 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 $\mathcal{R}_{abs}(L) = \log_2 m - \mathcal{H}(L)$ in diesem Ausdruck wird auch als die **absolute Redundanz** der Klartextsprache (gemessen in bit/Buchstabe) bezeichnet.*

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

Beispiel 77. *Im Deutschen hat die Einzelbuchstabenverteilung L_1 eine Entropie von $\mathcal{H}(L_1) = 4,1$ bit, während eine auf A_{lat} gleichverteilte Zufallsvariable U einen Entropiewert von $\mathcal{H}(U) = \log(26) = 4,76$ hat. Für die Bi- und Trigramme ergeben sich Entropiewerte von $\mathcal{H}(L_2)/2 = 3,86$ und $\mathcal{H}(L_3)/3 = 3,61$ bit pro Buchstabe. Mit wachsender Länge sinkt die Entropie von deutschsprachigen Texten weiter ab und strebt gegen*

einen Grenzwert $\mathcal{H}(L)$ *von* $1,56$ *bit pro Buchstabe.*

n	$\mathcal{H}(L_n)$	$\mathcal{H}(L_n)/n$	$\mathcal{R}_{abs}(L_n)/n$	$\mathcal{R}(L_n)$
1	$4,10$	$4,10$	$0,66$	14%
2	$7,72$	$3,86$	$0,90$	19%
3	$10,82$	$3,61$	$1,15$	24%
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
∞	∞	$\mathcal{H}(L) = 1,56$	$\mathcal{R}_{abs}(L) = 3,20$	$\mathcal{R}(L) = 67\%$

Ein durchschnittlicher deutscher Text hinreichender Länge enthät also einen Redundanzanteil von ca. 67%*, so dass er sich bei optimaler Kodierung auf circa* $1/3$ *seiner ursprünglichen Länge komprimieren lässt.* ◁

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

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

wobei wir die Klartextlänge n variabel halten und der Einfachheit halber annehmen, dass die Menge C_n der zugehörigen 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 in Frage kommenden Schlüssel für y. $K(y)$ 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 desto unsicherer, je kleiner 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. Ist $\bar{s}_n$ gleich 0, so liefert der abgefangene Kryptotext y dem Gegner genügend Information, um den benutzten Schlüssel und somit den zu y gehörigen Klartext eindeutig bestimmen zu können (sofern er über unbegrenzte Ressourcen an Rechenkraft und Zeit verfügt).

Definition 78 (**Eindeutigkeitsdistanz**)**.** *Die* ***Eindeutigkeitsdistanz*** n_0 *eines Kryptosystems ist der kleinste Wert von* n*, für den* $\bar{s}_n = 0$ *wird.*

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

Definition 79 (**bedingte Entropie**)**.** *Seien* X, Y *Zufallsvariablen. Dann ist die* ***bedingte Entropie*** *von* X *unter* Y *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* $\Pr[X|y = x] = p(x|y) = \Pr[X = x \mid Y = y]$ *ist (d.h.* $\mathcal{H}(X|y) = \sum_{x \in W(X)} p(x|y) \cdot \log_2(1/p(x|y))$*.*

Satz 80.

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

Beweis. s. Übungen. □

Korollar 81. $\mathcal{H}(X|Y) \leq \mathcal{H}(X)$, *wobei Gleichheit genau dann eintritt, wenn* X *und* Y *stochastisch unabhängig sind.*

Satz 82. *In jedem Kryptosystem gilt für die Klartextentropie* $\mathcal{H}(X)$*, die Schlüsselentropie* $\mathcal{H}(K)$ *und die Kryptotextentropie* $\mathcal{H}(Y)$

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

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

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

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 K unabhängig sind,

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

□

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. Seien X_n und Y_n die Zufallsvariablen, die die Verteilungen der n-Gramme der Klartextsprache und der zugehörigen Kryptotexte beschreiben.

Lemma 83.

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

Beweis.

1. Unter Verwendung der Jensenschen Ungleichung folgt

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

2. Mit Satz 82 folgt
$$\mathcal{H}(K|Y_n) = \mathcal{H}(K) + \mathcal{H}(X_n) - \mathcal{H}(Y_n).$$
Die Klartextentropie $\mathcal{H}(X_n)$ lässt sich durch
$$\mathcal{H}(X_n) = \mathcal{H}(L_n) \geq n\mathcal{H}(L) = n(1 - \mathcal{R}(L)) \log_2 m$$
abschätzen, 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
$$\mathcal{H}(K|Y_n) = \mathcal{H}(K) + \underbrace{\mathcal{H}(X_n) - \mathcal{H}(Y_n)}_{\geq -n\mathcal{R}(L)\log_2 m}.$$

□

Zusammen ergibt sich also
$$\log_2(\bar{s}_n + 1) \geq \mathcal{H}(K) - n\mathcal{R}(L) \log_2 m.$$
Im Fall, dass der Schlüssel unter Gleichverteilung gezogen wird, erreicht $\mathcal{H}(K)$ den maximalen Wert $\log_2 \|K\|$, was auf die gesuchte Abschätzung für $\bar{s}_n$ führt. Wir fassen zusammen.

Satz 84. *Werden mit einem Kryptosystem Klartexte $x \in A^n$ der Länge n mit einem unter Gleichverteilung gezogenen Schlüssel $k \in K$ verschlüsselt, und ist $\|C_n\| = \|A^n\| = m^n$ für den zugehörigen Kryptotextraum $C_n = \{E(k, x) \mid k \in K, x \in A^n\}$, so gilt für die erwartete Anzahl $\bar{s}_n$ der unechten Schlüssel,*
$$\bar{s}_n \geq \frac{\|K\|}{m^{n\mathcal{R}(L)}} - 1.$$

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

Korollar 85. *Unter den Bedingungen des obigen Satzes gilt*
$$n_0 \geq \frac{\log_2 \|K\|}{\mathcal{R}(L) \log_2 m} = \frac{\log_2 \|K\|}{\log_2 m - \mathcal{H}(L)} = \frac{\log_2 \|K\|}{\mathcal{R}_{abs}(L)}.$$

Man beachte, dass wir nur die Mindestmenge an Kryptotext zur eindeutigen Bestimmung des *Schlüssels* abgeschätzt haben. Natürlich erlaubt die eindeutige Bestimmung des Schlüssels auch die eindeutige Bestimmung des Klartexts. Unter Umständen kann jedoch der *Klartext* auch schon bei Kenntnis von wesentlich weniger Kryptotext eindeutig bestimmbar sein.

Beispiel 86. *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 $\mathcal{R}(L) = 3,2/4,76 \approx 67\%$ 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 Bestimmung des Schlüssels benötigt wird:

Analyse auf der Basis von	$\mathcal{R}_{abs}(L)$	*Blocklänge* l				
		10	*20*	*50*	*100*	*1 000*
Einzelzeichen	0,66	*59*	*165*	*578*	*1415*	*22 986*
Bigrammen	0,90	*40*	*111*	*390*	*954*	*15 502*
Trigrammen	1,15	*24*	*65*	*226*	*553*	*9 473*
n-Grammen, $n \to \infty$	3,20	*7*	*19*	*67*	*164*	*2 665*

Auch wenn die unteren Schranken für n_0 *bei der Analyse auf der Basis von Einzelzeichen endlich sind, ist in diesem Fall* $n_0 = \infty$*, da eine solche Analyse nicht zum Ziel führen kann, unabhängig davon, über wie viel Kryptotext der Gegner verfügt.* ◁

3.2 Weitere Sicherheitsbegriffe

Wie wir gesehen haben, muss für die Benutzung eines informationstheoretisch sicheren Kryptosystems ein immenser Aufwand betrieben werden. Daher 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 Verbereitung gefunden haben und allein schon deshalb ein lohnenswertes Ziel für einen Angriff darstellen.

Die komplexitätstheoretische Analyse eines Kryptosystems ist äußerst schwierig. Dies hängt damit zusammen, daß der Aufwand eines erfolgreichen Angriffs unabhängig von der vom Gegner angewandten Strategie abgeschätzt werden muss. Das heißt, es müssen nicht nur alle derzeit bekannten kryptoanalytischen Ansätze, sondern alle *möglichen* in Betracht gezogen werden. Dabei darf sich die Aufwandsanalyse nicht ausschließlich an einer vollständigen Rekonstruktion des Klartextes orientieren, da bereits ein geringfügiger Unterschied zwischen dem a posteriori und dem a priori Wissen für den Gegner einen Vorteil bedeuten kann.

Aus den genannten Gründen ist es bis heute noch für kein praktikables Kryptosystem gelungen, seine komplexitätstheoretische Sicherheit mathematisch zu beweisen. Damit ist auch nicht so schnell zu rechnen, zumindest nicht solange der Status fundamentaler komplexitätstheoretischer Fragen wie etwa des berühmten $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$-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. Hierzu ist es erforderlich, die Verletzung der Vertraulichkeit als ein algorithmisches Problem für den Gegner zu formulieren.

Definition 87 (Vorteil eines Gegners). *Sei $S = (M, C, E, D, K)$ ein Kryptosystem mit Schlüsselverteilung K. Ein Gegner ist ein Paar (G, V) von probabilistischen Algorithmen, wobei $G = (X_0, X_1)$ zwei Klartexte $x_0 \neq x_1 \in M$ generiert und V bei Eingabe zweier Klartexte $x_0, x_1 \in M$ und eines Kryptotextes $y \in C$ ein Bit ausgibt. Der Vorteil von (G, V) ist*

$$\alpha(G, V) = \Pr[V(X_0, X_1, E(K, X_B)) = B] - 1/2,$$

wobei B eine unabhängig von G und V auf $\{0, 1\}$ gleichverteilte Zufallsvariable ist, d.h. $\Pr[B = 0] = \Pr[B = 1] = 1/2$.

Satz 88. *Bei einem absolut sicheren Kryptosystem S kann kein Gegner einen Vorteil größer als 0 erzielen.*

Beweis. Bei einem absolut sicheren Kryptosystem hängt die Kryptotextverteilung $Y = E(K, X)$ nicht von der zugrunde liegenden Klartextverteilung X ab. Daher sind auch die ZVen $V(X_0, X_1, E(K, X_B))$ und B stochastisch unabhängig und es folgt

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

□

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

- Über welche Rechenressourcen verfügt ein Gegner realistischerweise?
- Wie groß darf der vom Gegner erzielte Vorteil höchstens sein, damit die Vertraulichkeit der Nachricht noch gewahrt bleibt?

Eine Antwort auf diese Fragen liefert Definition 89. Dabei gehen wir davon aus, dass das gewünschte Maß an Sicherheit durch einen Parameter $s \in \mathbb{N}$ regulierbar ist. 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 Gegner (G_s, V_s) vom Parameterwert s abhängen. Typischerweise werden Kryptosysteme nach der Schlüssellänge $s = |k|$ parameterisiert.

Definition 89 (**komplexitätstheoretisch sicher**). *Sei S ein Kryptosystem mit variablem Sicherheitsparameter s.*

- *Eine Funktion $\varepsilon : \mathbb{N} \to \mathbb{R}$ heißt **vernachlässigbar**, wenn für jedes Polynom p eine Zahl n_0 existiert, so dass $\varepsilon(n) < 1/p(n)$ für alle $n \geq n_0$ ist.*
- *Ein Gegner (G_s, V_s) für S heißt **effizient**, wenn sowohl G_s als auch V_s durch probabilistische Schaltkreise der Größe $s^{O(1)}$ berechenbar sind.*
- *S heißt **komplexitätstheoretisch sicher**, wenn jeder effiziente Gegner (G_s, V_s) nur einen vernachlässigbaren Vorteil erzielen kann (d.h. die Funktion $s \mapsto \alpha(G_s, V_s)$ 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 90 (**Produktkryptosystem**). *Seien $S_1 = (M_1, C_1, E_1, D_1, K_1)$ und $S_2 = (M_2, C_2, E_2, D_2, K_2)$ Kryptosysteme mit $C_1 = M_2$. Dann ist das **Produktkryptosystem** von S_1 und S_2 definiert als $S_1 \times S_2 = (M_1, C_2, E, D, K_1 \times K_2)$ mit*

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

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

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

Beispiel 91. *Sei $A = \{a_0, \ldots, a_{m-1}\}$. Man sieht leicht, dass die affine Chiffre $S = (M, C, K, E, D)$ mit $M = C = \mathcal{A}$ und $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$ das Produkt $S = S_1 \times S_2$ der multiplikativen Chiffre $S_1 = (M, C, K_1, E_1, D_1)$ mit der additiven Chiffre $S_2 = (M, C, K_2, E_2, D_2)$ ist, da für jeden Schlüssel $k = (k_1, k_2) \in K = \mathbb{Z}_m^* \times \mathbb{Z}_m$ gilt:*

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

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

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

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

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

eine Bijektion zwischen den Schlüsselräumen K' und K ist und der Schlüssel (k_1, k_2) im System S' die gleiche Chiffrierfunktion realisiert wie der Schlüssel $(k_2, k_2 k_1)$ in S, sind die Kryptosysteme $S = S_1 \times S_2$ und $S' = S_2 \times S_1$ als gleich (genauer: äquivalent, siehe Übungen) anzusehen, d.h. S_1 und S_2 kommutieren. ◁

Definition 92 (**endomorph, idempotent**). *Ein Kryptosystem $S = (M, C, K, D, E)$ mit $M = C$ heißt **endomorph**. Ein endomorphes Kryptosystem S heißt **idempotent**, falls $S \times S = S$ ist.*

Beispiel 93. *Eine leichte Rechnung zeigt, dass die additive, die multiplikative und die affine Chiffre idempotent sind. Ebenso die Blocktransposition sowie die Vigenère- und Hill-Chiffre.* ◁

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

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

dürfen hierbei S_1 und S_2 jedoch nicht kommutieren.

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

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

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

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

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

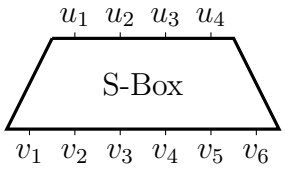
Beispiele für iterierte Chiffren sind der aus 16 Runden bestehende DES-Algorithmus und der AES mit einer variablen Rundenzahl $N \in \{10, 12, 14\}$, die wir in späteren Abschnitten behandeln werden.

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

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



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

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

Für die Speicherung einer S-Box $\pi_S : \{0,1\}^l \to \{0,1\}^{l'}$ 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 A^{lm} bezeichnen wir die zugehörige Permutation auf $\{1, \ldots, lm\}$ mit π_P, d.h.

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

Definition 95 (**Substitutions-Permutations-Netzwerk**). *Sei $A = \{0,1\}$ und sei $M = C = A^{lm}$ für natürliche Zahlen $l, m \geq 1$. Ein **Substitutions-Permutations-Netzwerk** (SPN) wird durch Permutationen $\pi_S : \{0,1\}^l \to \{0,1\}^l$ und $\pi_P : \{1, \ldots, lm\} \to \{1, \ldots, lm\}$ sowie durch einen Schlüsselgenerator $f : \{0,1\}^k \to \{0,1\}^{lm(N+1)}$ beschrieben. Der Generator f erzeugt aus einem (externen) Schlüssel $K \in \{0,1\}^k$ eine Folge $f(K) = (K^1, \ldots, K^{N+1})$ von $N+1$ Rundenschlüsseln K^r, unter denen ein Klartext $x \in \{0,1\}^{lm}$ gemäß folgendem Algorithmus in einen Kryptotext $y = E_{f,\pi_S,\pi_P}(K, x) \in \{0,1\}^{lm}$ überführt wird.*

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

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

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

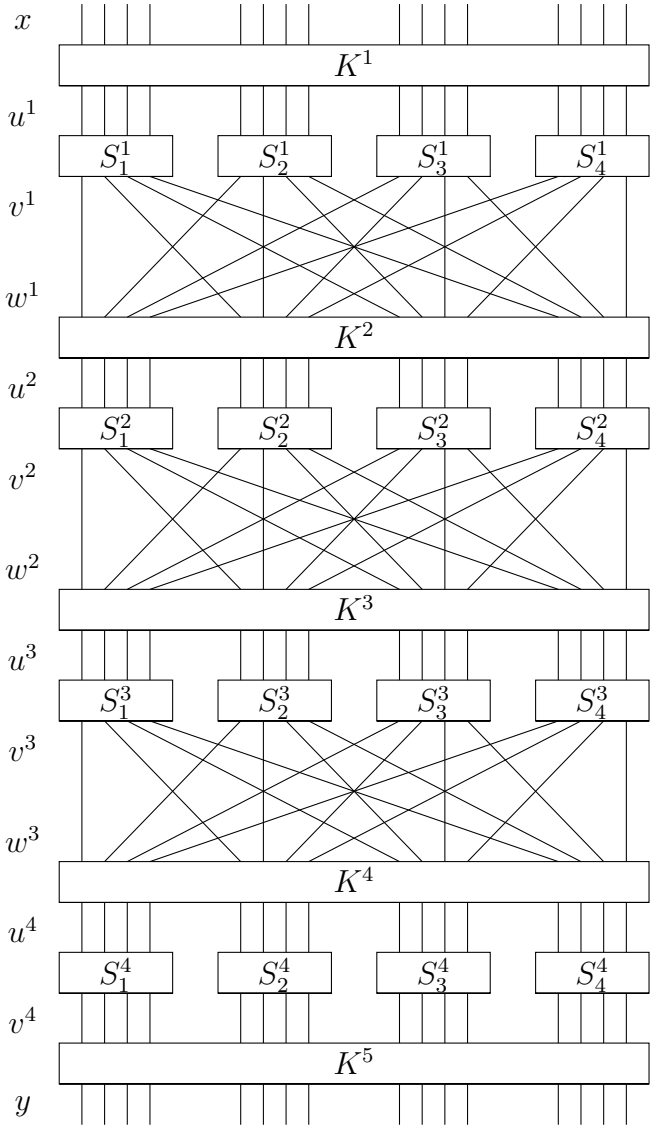


Abbildung 4.1: Ein Substitutions-Permutations-Netzwerk.

Beispiel 96. *Sei $l = m = N = 4$ und sei $k = 32$. Für f wählen wir die Funktion $f(K) = (K^1, \ldots, K^5)$ mit $K^r = K[4(r-1)+1, 4(r-1)+16]$. Weiter seien $\pi_S : \{0,1\}^4 \to \{0,1\}^4$ und $\pi_P : \{1,\ldots,16\} \to \{1,\ldots,16\}$ die folgenden Permutationen (wobei die Argumente und Werte von π_S hexadezimal dargestellt sind; siehe auch Abbildung 4.1):*

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

und

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

Für den Schlüssel $K = 0011\,1010\,1001\,0100\,1101\,0110\,0011\,1111$ liefert f beispielsweise

die Rundenschlüssel $f(K) = (K^1, \ldots, K^5)$ *mit*

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

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

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

◁

4.3 Lineare Approximationen

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

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

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

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

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

$$V_j = U_{i_1} \oplus \cdots \oplus U_{i_k}$$

für geeignete Indizes $1 \le i_1 < \cdots < i_k \le l$ darstellbar. Die Idee hinter der linearen Kryptoanalyse ist nun, Gleichungen der Form

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

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

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

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: $U_a \oplus V_b \oplus c = 0$) eine möglichst große Wahrscheinlichkeit besitzt.
Für eine Zufallsvariable X mit Wertebereich $W(X) = \{0,1\}$ und $p = \Pr[X = 0]$ bezeichne $\varepsilon(X)$ den Wert $\varepsilon(X) = p - 1/2$ (auch *Bias* von X genannt).
Unter Benutzung dieser Notation sollte also der Bias $\varepsilon(U_a \oplus V_b \oplus c)$ der Zufallsvariablen $U_a \oplus V_b \oplus c$ einen möglichst groß sein. Wegen

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

ist die durch a und b beschriebene **lineare Approximation** $U_a \oplus V_b$ also um so besser, je größer der Absolutbetrag $|\varepsilon(U_a \oplus V_b)|$ des Bias dieser Approximation ist.

Beispiel 97. *Wir betrachten die S-Box $\pi_S : \{0,1\}^4 \to \{0,1\}^4$ aus Beispiel 96. Dann nimmt die Zufallsvariable $(U_1, \ldots, U_4, V_1, \ldots, V_4)$ die folgenden 16 Werte 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 $L(a,b)$.

a	b															
	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
									⋮							
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 aus Beispiel 96 und führen eine lineare Kryptoanalyse durch. Dabei handelt es sich um einen Angriff bei bekanntem Klartext, d.h. es steht eine Menge M von t Klartext-Kryptotext-Paaren (x, y) zur Verfügung, die alle mit dem gleichen unbekannten Schlüssel K erzeugt wurden.

Seien $K^1, \ldots, K^5$ die zu K gehörigen Rundenschlüssel. Das Ziel besteht zunächst einmal darin, eine lineare Approximation für die Abbildung $x \mapsto u^4$ zu finden, bei der die Rundenschlüssel $K^1, \ldots, K^4$ benutzt werden (siehe Abbildung 4.2). Hierzu benutzen wir die beiden folgenden linearen Approximationen an die S-Box S:

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

mit einem Bias von $\varepsilon(T) = (L(B,4) - 8)/16 = (12 - 8)/16 = 1/4$ und

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

mit einem Bias von $\varepsilon(T') = (L(4,5) - 8)/16 = (4 - 8)/16 = -1/4$.

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

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

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

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

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

$$X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 \quad = \quad T_1 \oplus T_2 \oplus T_3 \oplus T_4 \oplus c \tag{4.1}$$

von $x \mapsto u^4$. An dieser Stelle ergeben sich folgende drei Fragen.

1. Warum gilt (4.1)?
2. Wie gut ist die lineare Approximation $X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4$?
3. Wie können wir mit ihrer Hilfe den Schlüssel bestimmen?

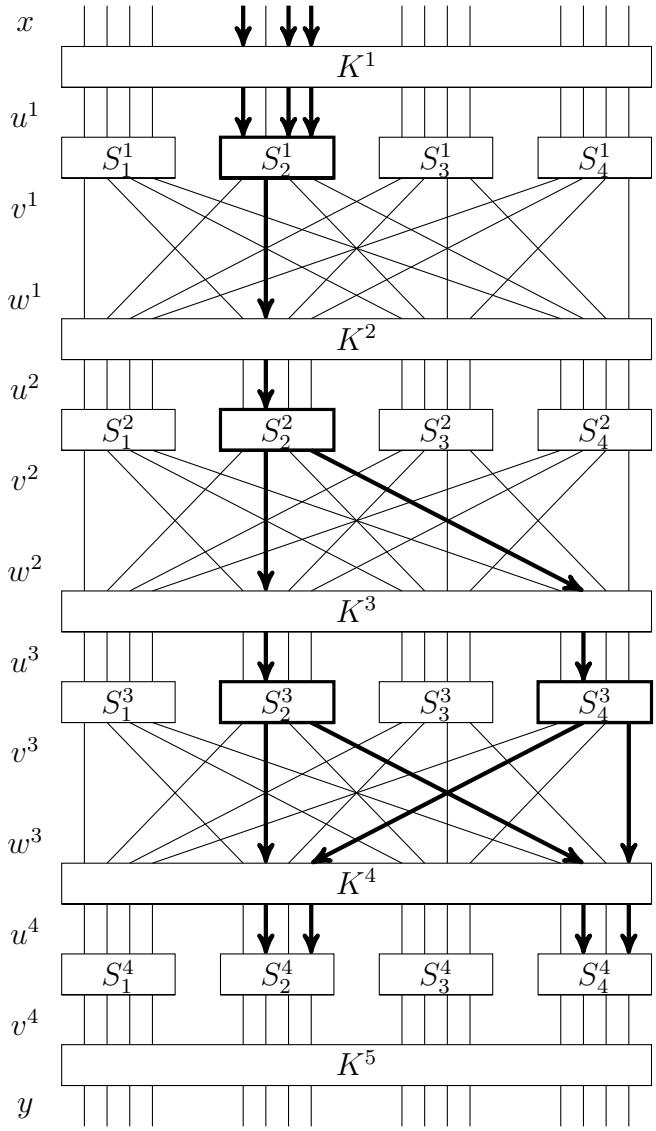


Abbildung 4.2: Eine lineare Approximation an ein Substitutions-Permutations-Netzwerk.

Wir gehen zunächst auf Frage 1 ein. Seien $c_1, \ldots, c_4$ die Schlüsselbitsummen

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

Dann gilt

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

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 $T_1 \oplus \cdots \oplus T_4$ 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 98 (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 99. *Seien X_1, X_2, X_3 Zufallsvariablen mit $\varepsilon(X_i) = 1/4$ für $i = 1,2,3$. Dann gilt nach obigem Lemma $\varepsilon_{i,j} = \varepsilon(X_i \oplus X_j) = 1/8$ für $1 \le i < j \le 3$. Man beachte, dass die Zufallsvariablen $X_1 \oplus X_2$ und $X_2 \oplus X_3$ nicht unabhängig sind, und daher das Piling-up-Lemma nicht anwendbar ist. Dieses würde nämlich für die Zufallsvariable*

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

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

Obwohl die Zufallsvariablen $T_1, \ldots, T_4$ nicht unabhängig sind, stellt sich in der Praxis heraus, dass sich der tatsächliche Wert $\varepsilon = \varepsilon(T_1 \oplus \cdots \oplus T_4)$ nicht zu sehr von diesem "hypothetischen" Wert unterscheidet, d.h.

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

Und schließlich zu Frage 3: Wir wissen bereits, dass ein zufälliger Klartext X entweder mit hoher oder mit niedriger Wahrscheinlichkeit auf ein Zwischenresultat U^4 mit

$$X_5 \oplus X_7 \oplus X_8 \oplus U_6^4 \oplus U_8^4 \oplus U_{14}^4 \oplus U_{16}^4 = 0 \tag{4.2}$$

führt. Gehen wir also davon aus, dass M eine repräsentative Auswahl von Klartext-Kryptotext-Paaren (x, y) darstellt, so wird die Anzahl der Paare (x, y) in M, die (4.2) erfüllen, ebenfalls eine Mehrheit oder eine Minderheit in M bilden. Man beachte, dass

sich für jeden Subschlüssel-Kandidaten (engl. *candidate subkey*) (L_1, L_2) für $(K^5_{(2)}, K^5_{(4)})$ die zu einem Kryptotext y gehörigen Werte u^4_6, u^4_8, u^4_{14} und u^4_{16} leicht berechnen lassen, da π_S^{-1} bekannt ist.

Die Idee besteht nun darin, für jeden Kandidaten (L_1, L_2) die Anzahl $\alpha(L_1, L_2)$ aller Paare (x, y) in M zu bestimmen, die bei Benützung von (L_1, L_2) Gleichung (4.2) erfüllen. Für den richtigen Kandidaten wird diese Anzahl ungefähr bei $t/2 \pm t/32$ liegen, wogegen bei Benutzung eines falschen Subschlüssels mit einer Anzahl von circa $t/2$ zu rechnen ist. Für genügend große Werte von t lassen sich auf diese Weise 8 Bit von K^5 (und damit von K) bestimmen.

Algorithmus LinearAttack

```
1  for (L1, L2) := (0, 0) to (F, F) do
2     α(L1, L2) := 0
3  for each (x, y) ∈ M do
4     for (L1, L2) := (0, 0) to (F, F) do
5        v^4_(2) := L1 ⊕ y_(2)
6        v^4_(4) := L2 ⊕ y_(4)
7        u^4_(2) := π_S^{-1}(v^4_(2))
8        u^4_(4) := π_S^{-1}(v^4_(4))
9        if x5 ⊕ x7 ⊕ x8 ⊕ u^4_6 ⊕ u^4_8 ⊕ u^4_14 ⊕ u^4_16 = 0 then
10          α(L1, L2) := α(L1, L2) + 1
11 max := −1
12 for (L1, L2) := (0, 0) to (F, F) do
13    β(L1, L2) := |α(L1, L2) − t/2|
14    if β(L1, L2) > max then
15       max := β(L1, L2)
16       maxkey := (L1, L2)
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 Beispielfall 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 100 (Eingabe- und Ausgabedifferenz). *Seien $x, x^* \in \{0,1\}^l$ zwei Eingaben für eine S-Box $\pi_S : \{0,1\}^l \to \{0,1\}^{l'}$ und seien $y = \pi_S(x)$ und $y^* = \pi_S(x^*)$ die zugehörigen Ausgaben. Dann wird $x' = x \oplus x^*$ die **Eingabedifferenz** (engl. input-xor) und $y' = \pi_S(x) \oplus \pi_S(x^*)$ die **Ausgabedifferenz** (engl. output-xor) des Paares (x, x^*) genannt. Für eine vorgegebene Eingabedifferenz $a' \in \{0,1\}^l\}$ sei weiter*

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

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

Berechnen wir für alle Eingabepaare $(x, x^*) \in \Delta(a')$ die zugehörigen Ausgabedifferenzen, so verteilen sich diese mehr oder weniger gleichmäßig auf die $2^{l'}$ möglichen Werte in $\{0,1\}^{l'}$. Man beachte, dass im Fall einer linearen S-Box nur die Ausgabedifferenz $\pi_S(a')$ auftritt, da dann $\pi_S(x) \oplus \pi_S(x^*) = \pi_S(x \oplus x^*)$ ist. Ist dagegen π_S nicht linear, so kann die Eingabedifferenz a' auf unterschiedliche Ausgabedifferenzen führen, je nachdem, durch welches Eingabepaar $(x, x^*) \in \Delta(a')$ die Differenz a' realisiert wird. Im Allgemeinen lässt sich eine differentielle Kryptoanalyse um so leichter durchführen, je ungleichmäßiger die auftretenden Ausgabedifferenzen verteilt sind.

Definition 101 (Differential, Weitergabequotient). *Sei $a' \in \{0,1\}^l$ eine Eingabe- und sei $b' \in \{0,1\}^{l'}$ eine Ausgabedifferenz für eine S-Box π_S. Dann heißt (a', b') **Differential**. Die Anzahl der Eingabepaare (x, x^*), die die Eingabedifferenz a' in die Ausgabedifferenz b' überführen, bezeichnen wir mit $D(a', b')$, d.h.*

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

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

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

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

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

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

Beispiel 102. *Betrachten wir die S-Box $\pi_S : \{0,1\}^4 \to \{0,1\}^4$ aus Beispiel 96, so erhalten wir für die Eingabedifferenz $a' = 1011$ die Menge*

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

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

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

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

a'	b'															
	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

◁

Können wir nun in einem SPN für bestimmte S-Boxen S_i Differentiale (a'_i, b'_i) finden, so dass die Eingabedifferenz dieser Differentiale mit der (permutierten) Ausgabedifferenz der Differentiale in der jeweils vorhergehenden Runde übereinstimmt (siehe Abbildung 4.3), so können wir diese Differentiale zu einer so genannten **Differentialspur** (engl. differential trail) zusammen setzen. Unter der Annahme, dass die beteiligten S-Boxen S_i (diese werden auch als **aktiv** bezeichnet) unabhängig voneinander den zugeordneten Differentialen (a'_i, b'_i) folgen (bzw. nicht folgen), berechnet sich der Weitergabequotient der Spur als das Produkt der Weitergabequotienten der beteiligten Differentiale. Obwohl diese Annahme i.a. nicht zutrifft, treten in praktischen Anwendungen kaum große Abweichungen von diesem hypothetischen Wert auf.

Beispiel 103. *Betrachten wir das SPN aus Beispiel 96, 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 $(1011, 0010) = (B, 2)$ mit $Q(B, 2) = 1/2$,

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

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

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

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

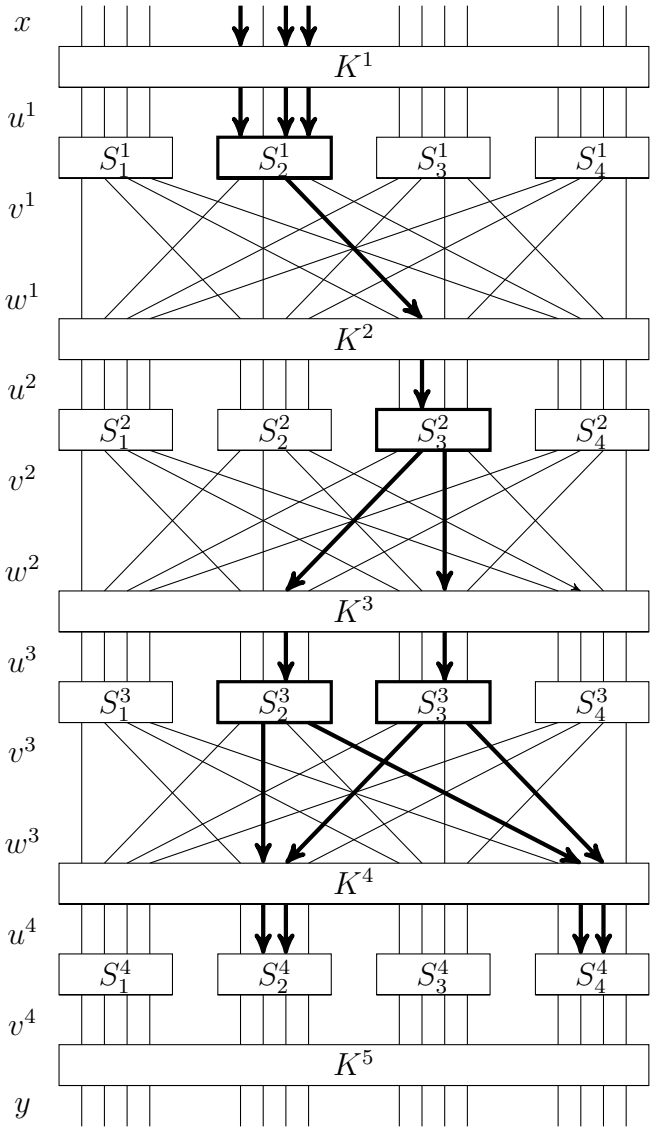


Abbildung 4.3: Eine Differentialspur für ein Substitutions-Permutations-Netzwerk.

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

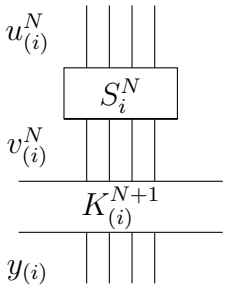
auszufiltern, da sie (wie alle falschen Doppelpaare) nur „Hintergrundrauschen“ erzeugen und somit die Bestimmung des Schlüssels eher behindern.

Beobachtung 104. *Für die Ausgabe $v_{(i)}^N$ der S-Box S_i^N in Runde N gilt*

$$v_{(i)}^N = y_{(i)} \oplus K_{(i)}^{N+1}$$

und die Eingabe $u_{(i)}^N$ der S-Box S_i^N in Runde N ist

$$u_{(i)}^N = \pi_S^{-1}(v_{(i)}^N) = \pi_S^{-1}(y_{(i)} \oplus K_{(i)}^{N+1})$$



Da die S-Box S_i^N nicht linear ist, hängt die aus den Kryptotextblöcken $y_{(i)}$ und $(y_{(i)})^*$ zurückgerechnete Eingabedifferenz

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

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

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

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

Gemäß Beobachtung 104 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 103 für einen Angriff mittels differentieller Analyse.

Beispiel 105. *Der Algorithmus* DifferentialAttack *bestimmt für jeden Subschlüssel-Kandidaten (L_1, L_2) für $(K_{(2)}^5, K_{(4)}^5)$ die Anzahl $\gamma(L_1, L_2)$ aller Doppelpaare (x, x^*, y, y^*) in M, die mit (L_1, L_2) konsistent sind und (in Zeile 4) nicht als falsch erkannt werden. Ausgegeben wird der Kandidat (L_1, L_2) mit dem größten γ-Wert.* ◁

Algorithmus DifferentialAttack

```
1  for (L1, L2) := (0, 0) to (F, F) do
2     γ(L1, L2) := 0
3  for each (x, x*, y, y*) ∈ M do
4     if y(1) = y*(1) und y(3) = y*(3) then
```

```
5        for (L1, L2) := (0, 0) to (F, F) do
6           v^4_(2) := L1 ⊕ y_(2)
7           v^4_(4) := L2 ⊕ y_(4)
8           u^4_(2) := π_S^{-1}(v^4_(2))
9           u^4_(4) := π_S^{-1}(v^4_(4))
10          (v^4_(2))* := L1 ⊕ y*_(2)
11          (v^4_(4))* := L2 ⊕ y*_(4)
12          (u^4_(2))* := π_S^{-1}((v^4_(2))*)
13          (u^4_(4))* := π_S^{-1}((v^4_(4))*)
14          (u^4_(2))' := u^4_(2) ⊕ (u^4_(2))*
15          (u^4_(4))' := u^4_(4) ⊕ (u^4_(4))*
16          if (u^4_(2))' = 0110 und (u^4_(4))' = 0110 then
17             γ(L1, L2) := γ(L1, L2) + 1
18  max−1
19  for (L1, L2) := (0, 0) to (F, F) do
20     if γ(L1, L2) > max then
21        max := γ(L1, L2)
22        maxkey := (L1, L2)
23  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 Beispielfall reichen $t \approx 80$ Doppelpaare, wobei $\varepsilon^{-1} \approx 38$ ist).

5 DES und AES

5.1 Der Data Encryption Standard (DES)

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

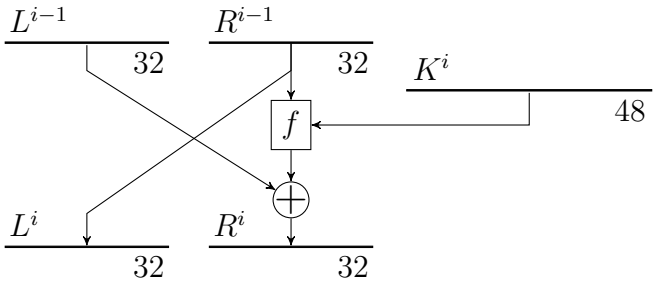
Bereits im September 1997 veröffentlichte das NIST eine Ausschreibung für den AES (Advanced Encryption Standard) genannten Nachfolger des DES. Nach einer mehrjährigen Auswahlprozedur wurde im November 2001 der Rijndael-Algorithmus als AES genormt.

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

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

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

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



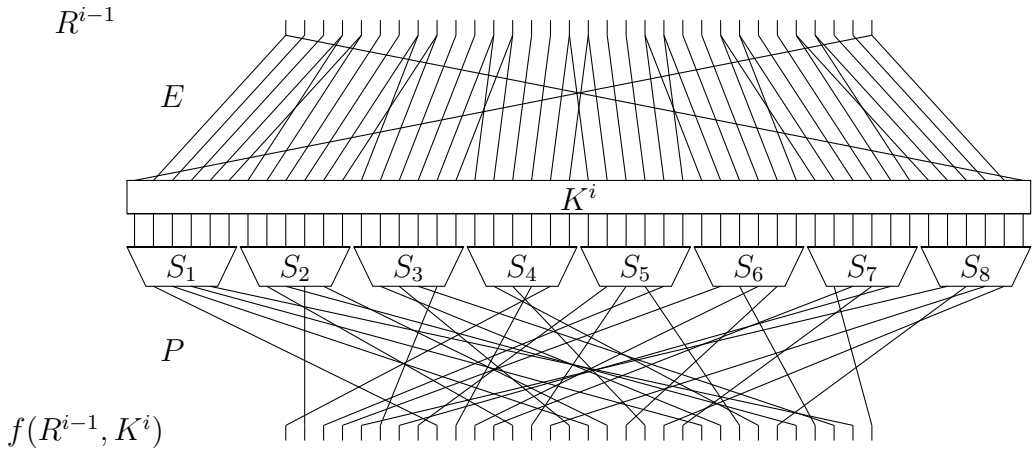
Der DES chiffriert Binärblöcke der Länge 64 und benutzt hierzu einen Schlüssel mit 56 Bit. Der Schlüssel ergibt zusammen mit 8 Paritätsbits (die Bits 8, 16,..., 64) einen ebenfalls 64 Bit langen Schlüsselblock K. Es gibt somit $2^{56} \approx 7.2 \cdot 10^{16}$ verschiedene Schlüssel. Im Einzelnen werden folgende Chiffrierschritte ausgeführt:

– Zuerst wird der Klartextblock x einer Initialpermutation IP unterzogen:

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

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

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



Berechnung der Funktion f

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

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

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

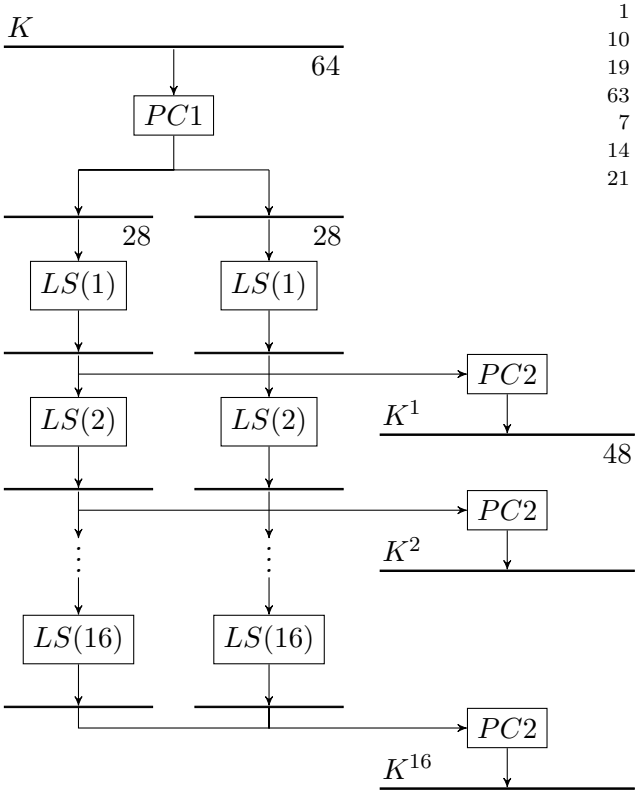
– Aus dem nach der 16. Iteration erhaltenen Bitvektor $w^{16} = L^{16}R^{16}$ wird durch Vertauschen der beiden Hälften und Anwendung der inversen Initialpermutation der Kryptotext y gebildet:

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

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

Die acht Substitutionsboxen

Die Schlüsselgenerierung. 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 jeweils zyklisch um ein oder zwei Bit verschoben (siehe dazu Tabelle $LS(i)$).



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

permuted choice 1 *permuted choice 2*

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

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

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

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

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

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

Definition 106 (**schwache Schlüssel**). *Ein DES-Schlüssel K heißt **schwach**, falls alle durch ihn erzeugten Rundenschlüssel gleich sind (d.h. es gilt $\|\{K^1,\ldots,K^{16}\}\|=1$).*

Es gibt vier schwache Schlüssel (siehe Übungen):

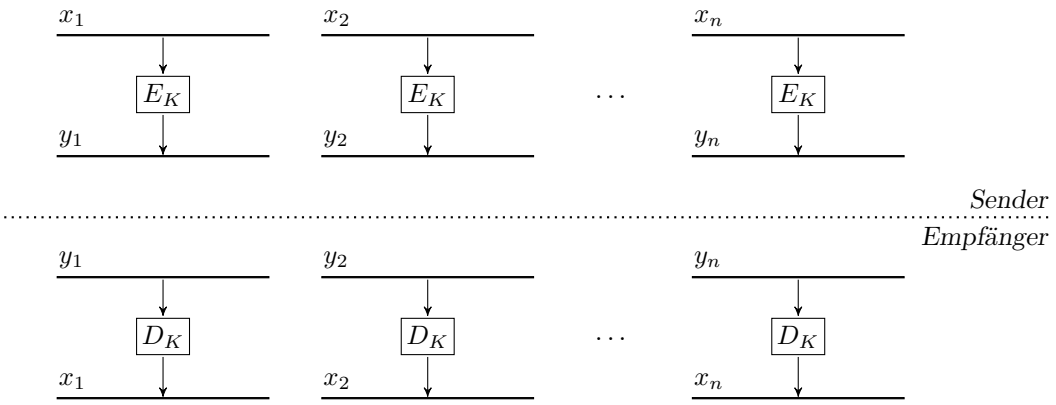
```
0101010101010101
FEFEFEFEFEFEFEFE
1F1F1F1F0E0E0E0E
E0E0E0E0F1F1F1F1
```

und für sie gilt $\mathrm{DES}(K, \mathrm{DES}(K, x)) = x$.

Neben diesen schwachen Schlüsseln existieren noch sechs weitere sogenannte „semischwache" Schlüsselpaare (K, K'), für die $\mathrm{DES}(K', \mathrm{DES}(K, x)) = x$ gilt (siehe Übungen).

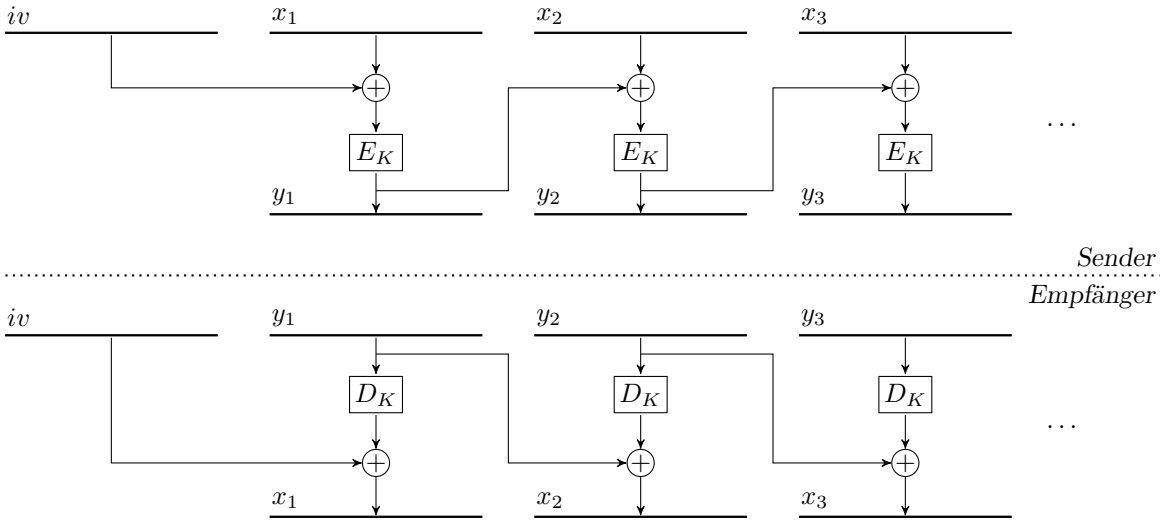
5.2 Betriebsarten von Blockchiffren

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, $1 \le t \le l$, übertragen lassen (OFB und CFB).



ECB-Mode (electronic code book; elektronisches Codebuch): Die Binär-Nachricht x wird in 64-Bit Blöcke x_i 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.

Um zu verhindern, dass ein Eindringling den Kryptotext verändert, ohne dass der Empfänger dies bemerkt, wird beim CBC-Mode 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.



CBC-Mode (cipher block chaining; Blockverkettung des Schlüsseltextes): 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.

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

CFB-Mode (cipher feedback; Rückführung des Kryptotextes): Ähnlich zum OFB-Mode, nur dass zur Erneuerung des Eingaberegisters nicht die ersten t Bits der E_K-Ausgabe, sondern der daraus gewonnene t-Bit Kryptotextblock verwendet wird.

Eine weitere Variante des OFB-Modes ist der **Counter-Mode**, bei dem die Pseudozufallsfolge mit Hilfe von E_K aus einer fortlaufenden Binärblockfolge $T_0, T_1, \ldots$ mit $T_{i+1} = T_i + 1 \bmod 2^l$ erzeugt wird. Dies hat den Vorteil, dass spätere Blöcke der Pseudozufallsfolge nicht von den vorhergehenden abhängen, und daher die Blöcke $E_K(T_i)$ parallel berechnet werden können.

5.3 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 107 (**Polynomring**). *Sei p prim. Dann enthält $\mathbb{Z}_p[x]$ alle Polynome*

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

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

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

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

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

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

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

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

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

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

Definition 108 (**irreduzibel**). *Ein Polynom $m(x) \in \mathbb{Z}_p[x]$ heißt **irreduzibel**, falls keine Polynome $p(x), q(x) \in \mathbb{Z}_p[x]$ vom Grad $deg(p), deg(q) \geq 1$ existieren mit*

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

Satz 109. *Der Restklassenpolynomring $\mathbb{Z}_p[x]/m(x)$ ist genau dann ein Körper, wenn $m(x)$ in $\mathbb{Z}_p[x]$ irreduzibel ist.*

Beweis. siehe Übungen. □

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

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

durch den Koeffizientenvektor $(a_{n-1},\ldots,a_0) \in (\mathbb{F}_p)^n$ 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 $\mathbb{F}_{2^n}$ entspricht dann der üblichen Vektoraddition (komponentenweisen Addition modulo p):

$$(a_{n-1},\ldots,a_0) + (b_{n-1},\ldots,b_0) = (c_{n-1},\ldots,c_0) \text{ mit } c_i = a_i + b_i \text{ für } i = 0,\ldots,n-1.$$

Im Fall $p = 2$ ist dies also die bitweise Addition modulo 2 (x-or). Die Multiplikation in $\mathbb{F}_{p^n}$ lässt sich wegen

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

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

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

ist

$$\begin{aligned} xb(x) &\equiv_{m(x)} b_{n-1}x^n + \sum_{i=0}^{n-2} b_i x^{i+1} \\ &\equiv_{m(x)} \sum_{i=1}^{n-1} b_{i-1}x^i - b_{n-1}\sum_{i=0}^{n-1} m_i x^i \\ &\equiv_{m(x)} \sum_{i=0}^{n-1} (b_{i-1} - b_{n-1}m_i)x^i, \end{aligned}$$

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

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

bzw. in Vektorschreibweise:

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

Es ist leicht zu sehen, dass die Multiplikation mit einem festen Körperelement $(a_{n-1},\ldots,a_0) \in \mathbb{F}_{p^n}$, also die Abbildung $(b_{n-1},\ldots,b_0) \mapsto (a_{n-1},\ldots,a_0)\cdot(b_{n-1},\ldots,b_0)$ linear über $\mathbb{F}_p$ ist. Folglich ist jede lineare Abbildung $f : (\mathbb{F}_{p^n})^k \to (\mathbb{F}_{p^n})^l$ über dem Körper $\mathbb{F}_{p^n}$ auch linear über $\mathbb{F}_p$, falls wir f als Abbildung von $(\mathbb{F}_p)^{nk}$ nach $(\mathbb{F}_p)^{nl}$ auffassen (siehe Übungen).

Beispiel 110. *Sei $p = 2$ und $n = 3$. Zunächst benötigen wir ein irreduzibles Polynom $m(x) \in \mathbb{Z}_2[x]$ vom Grad 3,*

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

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

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

zu betrachten. Da nun aber

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

und

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

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

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

und

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

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

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

ist. ◁

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

Beispiel 111. *Sei* $p = 2$ *und seien* $m(x) = x^8 + x^4 + x^3 + x + 1$ *und* $a(x) = x^6 + x^4 + x + 1$ *zwei Polynome. Dann können wir mit dem Euklidschen Algorithmus den (in Bezug auf den Grad) größten gemeinsamen Teiler* $g(x)$ *von* $m(x)$ *und* $a(x)$ *berechnen:*

i	$r_{i-1}(x)$	$=$	$d_{i+1}(x) \cdot r_i(x)$	$+ r_{i+1}(x)$
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 Euklidsche Algorithmus berechnet nun Polynome* $p_i(x)$ *und* $q_i(x)$ *gemäß der Vorschrift*

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

und

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

welche für $i = 0, 1, 2, 3, 4$ *die Gleichung* $p_i(x)m(x) + q_i(x)a(x) = r_i(x)$ *erfüllen:*

i	$p_i(x) \cdot m(x) +$	$q_i(x) \cdot a(x)$	$=$	$r_i(x)$
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 $q_4(x) = x^7 + x^6 + x^3 + x$ *von* $a(x)$ *modulo* $m(x)$ *ablesen.* ◁

5.4 Der Advanced Encryption Standard (AES)

5.4.1 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:

l	k				
	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 $l = 128$ Bit Blocklänge und $k = 128$ Bit Schlüssellänge.

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 durch eine 8-Bit Koeffizientenfolge $(a_7, \ldots, a_0)$ (also durch 2 Bytes) dargestellt werden. Hierzu verwenden wir die Funktionen FieldToBinary und BinaryToField, die wie folgt definiert sind:
- BinaryToField: $\{0,1\}^8 \to \mathbb{F}(2^8)$ berechnet aus der 8-Bit Koeffizienten-Darstellung das zugehörige Körperelement.
- FieldToBinary: $\mathbb{F}(2^8) \to \{0,1\}^8$ berechnet die Inverse der Funktion BinaryToField.

5.4.2 Die AES S-Box.

Sowohl bei der Schlüsselgenerierung als auch bei der Chiffrierung wird eine Substitution SubBytes verwendet, die auf einer 8-Bit S-Box π_{SubBytes} basiert. Diese S-Box benutzt als nicht-linearen Bestandteil die Funktion FieldInv: $\mathbb{F}(2^8) \to \mathbb{F}(2^8)$, die das multiplikative Inverse im Körper $\mathbb{F}(2^8)$ berechnet. Konkret wird die S-Box π_{SubBytes} durch folgenden Algorithmus berechnet (die Indexrechnung in Zeile 8 erfolgt modulo 8).

$\pi_{\text{SubBytes}}(a_7 \cdots a_0)$

```
1  input  a_7 ··· a_0
2  z := BinaryToField(a_7 ··· a_0)  if  z ≠ 0  then
3     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
```

Beispiel 112. *Wir berechnen* $\pi_{\text{SubBytes}}(01010011)$. *Die Funktion* BinaryToField *liefert das zugehörige Polynom*

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

Das multiplikative Inverse von z *in* $\mathbb{F}(2^8)$ *ist*

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

(siehe Beispiel 111). Die Funktion FieldToBinary *liefert die zugehörige Koeffizienten-Darstellung*

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

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

$$\begin{aligned}
b_7 &= a_7 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus c_7 = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 1 \\
b_6 &= a_6 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus c_6 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 1 \\
&\vdots \\
b_1 &= a_1 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_0 \oplus c_1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0 \\
b_0 &= a_0 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7 \oplus c_0 = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1
\end{aligned}$$

Somit ist $\pi_{\text{SubBytes}}(01010011) = 11101101$ *oder hexadezimal:* $\pi_{\text{SubBytes}}(\texttt{53}) = \texttt{ED}$. ◁

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

X	Y														
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	**ED**	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
⋮								⋮								
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

5.4.3 Die Schlüsselgenerierung.

Beim 10-Runden AES mit Block- und Schlüssellänge $l = k = 128$ werden 11 Rundenschlüssel $K^0, \ldots, K^{10}$ der Länge 128 benutzt. Jedes K^i besteht also aus 16 Bytes bzw. 4 Worten mit jeweils 4 Bytes. Bei der Berechnung der Rundenschlüssel werden (Wort-)Konstanten $RCon[1], \ldots, RCon[10]$ mit $RCon[i] = \textsc{FieldToBinary}(x^{i-1})0^{24} \in \{0,1\}^{32}$ 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	$1B$000000	36000000

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

$\textsc{KeyExpansion}(K)$

```
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:

- $\textsc{RotWord} : \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \to \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8$ führt eine zyklische Verschiebung der 4 Eingabebytes um ein Byte nach links durch:

$$\textsc{RotWord}(B_0, B_1, B_2, B_3) = (B_1, B_2, B_3, B_0),$$

- $\textsc{SubWord} : \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \to \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8 \times \mathbb{F}(2)^8$ ersetzt

jedes Eingabebyte B_i durch $\pi_{\text{SubBytes}}(B_i)$:

$$\begin{aligned}&\text{SubWord}(B_0, B_1, B_2, B_3)\\ &= \text{SubBytes}(B_0, B_1, B_2, B_3)\\ &= (\pi_{\text{SubBytes}}(B_0), \pi_{\text{SubBytes}}(B_1), \pi_{\text{SubBytes}}(B_2), \pi_{\text{SubBytes}}(B_3))\end{aligned}$$

5.4.4 Der AES-Chiffrieralgorithmus

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

AES-Verschlüsselung

```
1  AddRoundKey(K^0)
2  for i := 1 to 9 do
3    SubBytes
4    ShiftRows
5    MixColumns
6    AddRoundKey(K^i)
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 AddRoundKey bezeichnet.
- Danach werden 9 Runden ausgeführt, wobei in jeder Runde i eine Substitution namens SubBytes, eine Permutation namens ShiftRows, eine lineare Substitution namens MixColumns und eine AddRoundKey Operation mit dem Rundenschlüssel K^i durchgeführt werden.
- Es folgt Runde 10 mit den Operationen SubBytes, ShiftRows und AddRoundKey(K^{10}).

Der Klartext $x = x_0 \cdots x_{15}$, $x_i \in \{0,1\}^4$, (und alle daraus berechneten Zwischenergebnisse) werden in Form eines Arrays

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

dargestellt, das wie folgt initialisiert wird:

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

ShiftRows ist eine 128-Bit Permutation, die wie folgt definiert ist:

$$\begin{array}{|cccc|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \\ \hline \end{array} \quad \mapsto \quad \begin{array}{|cccc|} \hline s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \\ \hline \end{array}$$

MixColumns ist eine lineare 32-Bit Substitution, die auf den Spalten der Zwischenergebnisse operiert. Zu ihrer Berechnung wird folgende Funktion benutzt:

- FieldMult: $\mathbb{F}(2^8) \times \mathbb{F}(2^8) \to \mathbb{F}(2^8)$ führt die Multiplikation im Körper $\mathbb{F}(2^8)$ aus.

MixColumns(c_0, c_1, c_2, c_3)

1 **input** (c_0, c_1, c_2, c_3)
2 **for** $i := 0$ **to** 3 **do** $t_i :=$ `BinaryToField`(c_i)
3 $u_0 :=$ `FieldMult`$(x, t_0) +$ `FieldMult`$(x+1, t_1) + t_2 + t_3$
4 $u_1 :=$ `FieldMult`$(x, t_1) +$ `FieldMult`$(x+1, t_2) + t_3 + t_0$
5 $u_2 :=$ `FieldMult`$(x, t_2) +$ `FieldMult`$(x+1, t_3) + t_0 + t_1$
6 $u_3 :=$ `FieldMult`$(x, t_3) +$ `FieldMult`$(x+1, t_0) + t_1 + t_2$
7 **for** $i := 0$ **to** 3 **do** $c_i :=$ `FieldToBinary`(u_i)
8 **output** (c_0, c_1, c_2, c_3)

MixColumns führt eine lineare Transformation in dem Vektorraum $(\mathbb{F}_{2^8})^4$ aus, die sich auch wie folgt beschreiben lässt (hierbei stellen wir die 8 Bit Koeffizientenvektoren der Polynome in $\mathbb{F}_{2^8}$ hexadezimal dar, also 03 für $x + 1$):

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} \quad \mapsto \quad \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Besonders elegant lässt sich die Operation MixColumns im Polynom-Restklassenring $\mathbb{F}_{2^8}[y]/(y^4 + 1)$ beschreiben. Sei $c(y) = \sum_{i=0}^{3} c_i y^i$ das durch die Spalte (c_0, c_1, c_2, c_3) repräsentierte Polynom in diesem Ring und sei $a(y)$ das Polynom $a(y) = 03y^3 + 01y^2 + 01y + 02$. Dann ist leicht zu sehen, dass

$$\text{MixColumns}(c(y)) = a(y)c(y)$$

ist, d.h. bei MixColumns handelt es sich um eine multiplikative Chiffre mit festem Schlüssel $a(y)$ im Ring $\mathbb{F}_{2^8}[y]/(y^4 + 1)$. Da das Polynom $y^4 + 1$ nicht irreduzibel in $\mathbb{F}_{2^8}[y]$ ist, ist $\mathbb{F}_{2^8}[y]/(y^4 + 1)$ zwar kein Körper. Da jedoch $a(y)$ invertierbar im Ring $\mathbb{F}_{2^8}[y]/(y^4 + 1)$ ist, kann die Inverse zu MixColumns mittels

$$\text{MixColumns}^{-1}(c(y)) = a^{-1}(y)c(y)$$

berechnet werden, wobei $a^{-1}(y) = 0By^3 + 0Dy^2 + 09y + 0E$ ist.

5.4.5 Kryptoanalytische Betrachtungen

Bis heute konnten keine Schwachstellen gefunden werden, d.h. alle bekannten Angriffe sind mindestens so aufwändig wie eine vollständige Schlüsselsuche. Die Tatsache, dass

für die S-Box die Inversen-Operation in einem endlichen Körper gewählt wurde, hat zur Folge, 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 Transformation MixColumns lineare und differentielle Angriffe mit nur wenigen aktiven S-Boxen (diese Technik wird von den AES-Entwicklern als *wide trail strategy* bezeichnet).

6 Zahlentheoretische Grundlagen

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

Satz 113. *Sei G eine endliche Gruppe der Ordnung $\|G\| = m$. Dann 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 114 (Kleiner Satz von Fermat)**.** *Ist p eine Primzahl und a eine nicht durch p teilbare Zahl (d.h. $a \in \mathbb{Z}_p^*$), dann ist $a^{p-1} - 1$ durch p teilbar:*

$$\forall a \in \mathbb{Z}_p^* : a^{p-1} \equiv_p 1.$$

6.1 Diskrete Logarithmen

Nehmen wir ein beliebiges Element a aus G und betrachten die Folge $a^0 = 1$, $a^1 = a$, a^2, a^3, …, so wissen wir nach obigem Satz, dass spätestens für $e = \|G\|$ wieder $a^e = 1$ gilt.

Definition 115 (**Ordnung**)**.** *Die **Ordnung** von a in G ist*

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

Die von a in G erzeugte Untergruppe $\{a^e \mid e \geq 0\} = \{a^0, \ldots, a^{\mathrm{ord}_G(a)-1}\}$ bezeichnen wir mit $[a]_G$ oder mit $[a]$, wenn G aus dem Kontext ersichtlich ist.

Im Fall $G = \mathbb{Z}_m^*$ schreiben wir auch einfach $\mathrm{ord}_m(a)$. Für das folgende besonders interessant sind Elemente a aus G, die die gesamte Gruppe erzeugen.

Definition 116 (**Primitivwurzel/Erzeuger**)**.** *Sei G eine endliche Gruppe der Ordnung $\|G\| = m$. Ein Element $g \in G$ der Ordnung $\mathrm{ord}_G(g) = \|G\| = m$ heißt **Erzeuger** von G.*

Ein Element $a \in G$ ist also genau dann ein Erzeuger, wenn die von a erzeugte Untergruppe $[a]$ gleich G ist. Falls G einen Erzeuger besitzt, wird G auch **zyklisch** genannt. Da $\mathrm{ord}_G(a) = \|[a]\|$ ist und $[a]$ eine Untergruppe von G ist, ist $\mathrm{ord}_G(a)$ für alle $a \in G$ ein Teiler von $\|G\| = m$. Zudem gilt für beliebige ganze Zahlen i, j (siehe Übungen)

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

Satz 117 (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).*

Für ein beliebiges Gruppenelement $b \in G$ ist die Exponentiation $e \mapsto b^e$ eine bijektive Abbildung von der Menge $\{0, 1, \ldots, \ ord_g(b) - 1\}$ auf $[b]_G$. Die zugehörige Umkehrabbildung spielt in der Kryptografie eine wichtige Rolle.

Definition 118 (**Index/diskreter Logarithmus**)**.** *Seien* $b \in G$ *und* $a \in [b]$*. Dann heißt der eindeutig bestimmte Exponent* $e \in \{0, 1, \ldots, \mathrm{ord}_G(b) - 1\}$ *mit*

$$b^e = a$$

***Index** oder **diskreter Logarithmus** von a zur Basis b in G (kurz:* $e = \log_{G,b}(a)$*). Im Fall* $G = \mathbb{Z}_m^*$ *schreiben wir auch einfach* $e = \log_{m,b}(a)$*.*

Während die diskrete Exponentialfunktion $e \mapsto b^e$ durch **wiederholtes Quadrieren und Multiplizieren** (siehe nächsten Abschnitt) effizient berechenbar ist, sind bis heute keine effizienten Verfahren zur Berechnung des diskreten Logarithmus bekannt.

Beispiel 119. *Betrachte die Gruppe* $G = \mathbb{Z}_{11}^*$*. Dann ist* $g = 2$ *ein Erzeuger von G, d.h.* $\mathrm{ord}_{11}(2) = 10$*.*

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

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

◁

Im Beweis des nächsten Satzes bestimmen wir die Anzahl aller Erzeuger einer zyklischen Gruppe G.

Satz 120. *Eine endliche Gruppe G der Ordnung* $\|G\| = m$ *ist genau dann zyklisch, falls jede Gleichung der Form* $x^e = 1$*,* $e \geq 1$*, höchstens e verschiedene Lösungen* $a \in G$ *hat. In diesem Fall hat G genau* $\varphi(m)$ *Erzeuger.*

Beweis. Falls G zyklisch und g ein Erzeuger von G ist, so ist g^i, $i \geq 0$, genau dann eine Lösung von $x^e = 1$, wenn $g^{ie} = 1$ also $ie \equiv_m 0$ ist. Daher hat $x^e = 1$ genau $\mathrm{ggT}(e, m) \leq e$ verschiedene Lösungen.

Wir zeigen nun, dass G für jeden Teiler d von m genau $\varphi(d)$ Elemente der Ordnung d enthält, falls die Polynomgleichung $x^e = 1$ für jedes $e \geq 1$ höchstens e verschiedene Lösungen hat. Sei

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

die Menge aller Elemente der Ordnung d und sei $a \in S_d$ beliebig. Dann reicht es zu zeigen, dass $S_d = \{a^i \mid i \in \mathbb{Z}_d^*\}$ ist.

Jedes Element in S_d erfüllt die Gleichung $x^d = 1$, die nach Voraussetzung höchstens d verschiedene Lösungen hat. Da mit a auch $a^2, \ldots, a^d$ paarweise verschiedene Lösungen dieser Gleichung sind, folgt $S_d \subseteq \{a, a^2, \ldots, a^d\}$. Zudem hat a^i genau dann die Ordnung d, wenn $\mathrm{ggT}(i, d) = 1$ ist (siehe Übungen). □

Da die Gleichung $x^d = 1$ in einem Körper höchstens d verschiedene Lösungen hat (siehe Übungen), hat die multiplikative Gruppe $\mathbb{F}_{p^n}^*$ genau $\varphi(p^n - 1)$ Erzeuger. Insbesondere hat die Gruppe $\mathbb{F}_p^* = \mathbb{Z}_p^*$ genau $\varphi(p - 1)$ Erzeuger.

Falls die Primfaktorzerlegung von der Gruppenordnung m bekannt ist, lässt sich effizient überprüfen, ob ein gegebenes Element $a \in G$ ein Erzeuger ist oder nicht.

Satz 121. *Sei G eine endliche Gruppe der Ordnung $\|G\| = m$. Ein Element $a \in G$ ist genau dann ein Erzeuger, wenn für jeden Primteiler q von m gilt:*

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

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/q, q prim.

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

$$a^{m/q} = a^{d\,\mathrm{ord}(a)/q} = (a^{\mathrm{ord}(a)})^{d/q} = 1.$$

□

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

ComputeGenerator$(G, q_1, \ldots, q_k)$

```
1 input zyklische Gruppe G und alle Primteiler q_1, ..., q_k von m = ||G||
2 repeat
3   guess randomly a ∈ G
4 until a^{m/q_i} ≠ 1 für 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 $O(\ln \ln m)$.

6.2 Effiziente Berechnung von Potenzen

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

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.

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
```

Beispiel 122. *Sei* $a = 1920$, $e = 19$ *und* $G = \mathbb{Z}_m^*$ *für* $m = 2773$. *Dann berechnen die Algorithmen* Pot *und* HornerPot *die modulare Potenz* $1920^{19} \bmod 2773 = 1868$ *wie folgt.*

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	$1016^2 \cdot 1920^1 = \mathbf{1868}$

◁

6.3 Primzahlen

Sei $\pi : \mathcal{N} \to \mathcal{N}_0$ mit

$$\pi(n) = \|\{2 \leq p \leq n \mid p \in \mathcal{P}\}\|$$

die Anzahl der Primzahlen kleiner gleich n. Mit $\pi_{a,\,m}(n)$ bezeichnen wir die Anzahl der Primzahlen kleiner gleich n, die von der Form $p = m \cdot k + a$ für ein $k \in \mathcal{N}$ sind.

Satz 123. (Primzahlsatz, Hadamard, de la Vallée Poussin 1896)
Ist $\mathrm{ggT}(a, m) = 1$, *so gilt*[*]

$$\pi_{a,\,m}(n) \sim \frac{n}{\varphi(m) \cdot \ln n}$$

Insbesondere gilt also

$$\pi(n) \sim \frac{n}{\ln n}.$$

Eine bessere Abschätzung liefert die Funktion $Li(n) = \int_2^n (\ln x)^{-1} dx$, wie folgende Tabelle zeigt.

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

[*] $f(n) \sim g(n)$ bedeutet $\lim_{n\to\infty} f(n)/g(n) = 1$.

Beispiel 124. *Die Anzahl der Primzahlen in einem Intervall* $[n, m]$ *ist ungefähr* $\frac{m}{\ln m} - \frac{n}{\ln n}$. *Für das Intervall* $I = [10\,000, 10\,100]$ *ergibt sich z. B. ein Näherungswert von*

$$\|I \cap \mathcal{P}\| \approx \frac{10\,100}{\ln 10\,100} - \frac{10^4}{\ln 10^4} \approx \frac{10\,100}{9.22} - \frac{10^4}{9.21} \approx 1095.4 - 1085.7 = 9.7 \approx 10,$$

während der tatsächliche Wert gleich 11 ist.

Für die Anzahl aller 100-stelligen Primzahlen (in Dezimaldarstellung), also aller Primzahlen im Intervall $I' = [10^{99}, 10^{100} - 1]$ *erhalten wir z. B. den Näherungswert*

$$\|I' \cap \mathcal{P}\| \approx \frac{10^{100}}{100 \ln 10} - \frac{10^{99}}{99 \ln 10} \approx \frac{10^{100}}{230.3} - \frac{10^{99}}{228.0} \approx (4.34 - 0.44)10^{97} \approx 3.9 \cdot 10^{97}.$$

Vergleicht man diese Zahl mit der Anzahl $10^{100} - 10^{99} = 9 \cdot 10^{99} = 900 \cdot 10^{97}$ *aller 100-stelligen Dezimalzahlen, so sehen wir, dass ungefähr jede* $900/3.9 \approx 230$*-te 100-stellige Dezimalzahl prim ist.* ◁

Der Beweis des Primzahlsatzes ist sehr aufwändig. Mit elementaren Mitteln lässt sich jedoch folgender Satz beweisen, der für die meisten Anwendungen vollkommen ausreicht.

Satz 125 (Tschebyscheff)**.** *Für alle* $n > 200$ *gilt* $\pi(n) > \frac{2 \cdot n}{3 \cdot \ln n}$.
(Ohne Beweis)

6.4 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 $\mathcal{P}_n \subseteq \mathbb{Z}_n^*$ wählt, die die folgenden drei Eigenschaften für alle $n \geq n_0$ erfüllen:

1. Für ein gegebenes $a \in \mathbb{Z}_n^*$ kann effizient, d. h. in Polynomialzeit getestet werden, ob $a \in \mathcal{P}_n$ ist.
2. Für primes n ist $\mathcal{P}_n = \mathbb{Z}_n^*$.
3. Für zusammengesetztes n ist ein konstanter Anteil aller Elemente von $\mathbb{Z}_n^*$ nicht in $\mathcal{P}_n$ enthalten, d. h. $\|\mathcal{P}_n\| \leq \varepsilon\varphi(n)$ für eine Konstante $\varepsilon < 1$.

Typischerweise wählt man für $\mathcal{P}_n$ 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 ∉ P_n then return (zusammengesetzt)
4 return (prim)
```

Hierbei steuert der Parameter k die maximale Fehlerwahrscheinlichkeit von $GT(n, k)$. Gilt nämlich $\|\mathcal{P}_n\| \leq \varepsilon\varphi(n)$ für zusammengesetztes n und eine Konstante $\varepsilon < 1$, so gibt $GT(n, k)$ für primes n immer „prim“ aus und für zusammengesetztes n höchstens mit Wahrscheinlichkeit ε^k.

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 ein „?“ ausgeben.

Es liegt nahe, den Satz von Fermat zur Konstruktion einer „Testmengensequenz“ $\mathcal{P}_n^{FT} = \{a \in \mathbb{Z}_n^* \mid a^{n-1} \equiv_n 1\}$ zu verwenden. Dies führt auf folgenden Fermat-Test (FT).

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

```
1  berechne die Binärdarstellung ∑_{i=0}^{r} e_i · 2^i, e_r = 1, von n − 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 m
7        if e_i = 1 then
8           z := z · a mod m
9     if z ≢_n 1 then return (zusammengesetzt)
10 return (prim)
```

Der Fermat-Test berechnet also die Potenz $a^{n-1} = z_0$ genau wie der Algorithmus HornerPot über eine Folge $z_r, \ldots, z_0$ mit $z_r = a$ und $z_{i-1} = z_i^2 a^{e_{i-1}} \bmod m$ 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 pseudoprim zur Basis 3. Es gibt sogar Zahlen n (z. B. $n = 561$) die pseudo-prim zu jeder Basis $a \in \mathbb{Z}_n^*$ sind (sogenannte *Carmichael-Zahlen*). Für diese Zahlen ist Bedingung c) in obiger Aufzählung nicht erfüllt, weshalb der Fermat-Test als Pseudo-Primzahltest bezeichnet wird. In den Übungen wird gezeigt, dass Bedingung c) 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).

6.5 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 $z_{i-1} = z_i^2 a^{e_{i-1}} \bmod m = a^{c_i} \bmod m$ für $i = r-1, \ldots, 0$, wobei $c_i = \sum_{j=i}^{r} e_j \cdot 2^{j-i}$ ist. 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 ebenfalls 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

$$\mathcal{P}_n^{MRT} = \{a \in \mathbb{Z}_n^* \mid z_0 \equiv_n 1 \text{ und } \forall i = r, \ldots, 1 : z_i^2 \equiv_n 1 \Rightarrow z_i \equiv_n \pm 1\}.$$

Es ist klar, dass diese Testmengen die Bedingungen a) und b) erfüllen. Mit etwas zahlentheoretischem Aufwand lässt sich zeigen, dass sie auch Bedingung c) für $\varepsilon = 1/4$ erfüllen. 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 ∑_{i=0}^{r} e_i · 2^i von n − 1, wobei e_r = 1 ist
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 m
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 126. *Sei $n = 221$. Dann berechnet der Miller-Rabin Test für $a = 174$, $a' = 137$ und $a'' = 18$ 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	c_i	$z_i = (a)^{c_i}$	$(z_i)^2$	$z_i' = (a')^{c_i}$	$(z_i')^2$	$z_i'' = (a'')^{c_i}$	$(z_i'')^2$
7	1	1	**174**	**220**	**137**	**205**	**18**	**103**
6	1	3	$\mathbf{220 \cdot 174 = 47}$	**220**	$\mathbf{205 \cdot 137 = 18}$	**103**	$\mathbf{103 \cdot 18 = 86}$	**103**
5	0	6	**220**	**1**	**103**	**1**	**103**	**1**
4	1	13	$\mathbf{1 \cdot 174 = 174}$	**220**	$1 \cdot 137 = 137$	205	$1 \cdot 18 = 18$	103
3	1	27	$\mathbf{220 \cdot 174 = 47}$	**220**	$205 \cdot 137 = 18$	103	$103 \cdot 18 = 86$	103
2	1	55	$\mathbf{220 \cdot 174 = 47}$	**220**	$103 \cdot 137 = 188$	205	$103 \cdot 18 = 86$	103
1	0	110	**220**	**1**	205	35	103	1
0	0	220	**1**	1	35	120	1	1

Der Miller-Rabin Test erkennt also die Zahl $n = 221$ bei Wahl von $a = 174$ nicht als zusammengesetzt, wohl aber bei Wahl von $a = 137$ und $a = 18$. Dagegen würde dies der Fermat-Test bei Wahl von $a = 18$ ebenfalls nicht erkennen. ◁

Die Zahlen $a \in \mathcal{P}_n^{\mathrm{MRT}}$ 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 und sei

$$J_n = \{a \in \mathbb{Z}_n^* \mid a^{2^j u} \equiv_n \pm 1\}, \text{ wobei } j = \max\{0 \leq i \leq m \mid \exists a \in \mathbb{Z}_n^* : a^{2^i u} \equiv_n -1\}.$$

Behauptung 127. *J_n ist eine Untergruppe von $\mathbb{Z}_n^*$.*

Es genügt zu zeigen, dass J_n unter Multiplikation abgeschlossen ist. Seien hierzu $a, b \in J_n$. Dann gilt

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

Behauptung 128. $\mathcal{P}_n^{\mathrm{MRT}} \subseteq J_n$.

Sei $a \in \mathcal{P}_n^{\mathrm{MRT}}$. Dann gilt

$$a^{n-1} \equiv_n 1 \qquad (*)$$

$$\forall i \in \{1, \ldots, m\} : a^{2^i u} \equiv_n 1 \rightarrow a^{2^{i-1} u} \equiv_n \pm 1 \qquad (**)$$

Aus (*) folgt unmittelbar $a^{2^m u} \equiv_n 1$. Daraus folgt mit (**) und der Definition von j:

$$\forall i \in \{j+1, \ldots, m\} : a^{2^i u} \equiv_n 1.$$

Mit (**) folgt schließlich $a^{2^j u} \equiv_n \pm 1$, und damit $a \in J_n$.

Behauptung 129. *Falls n zusammengesetzt ist, ist J_n eine echte Untergruppe von $\mathbb{Z}_n^*$ und daher*

$$\left\| \mathcal{P}_n^{\mathrm{MRT}} \right\| \leq \|J_n\| \leq \frac{1}{2} \left\| \mathbb{Z}_n^* \right\|.$$

Sei $n = n_1 n_2$ mit teilerfremden Faktoren $n_1, n_2 > 2$. Nach Definition von j existiert dann eine Zahl $v \in \mathbb{Z}_n^*$ mit $v^{2^j u} \equiv_n -1$. Dann ist aber die Zahl $w \in \mathbb{Z}_n^*$ mit

$$w \equiv_{n_1} v,$$
$$w \equiv_{n_2} 1$$

nicht in J_n enthalten:

$$w^{2^j u} \equiv_{n_1} v^{2^j u} \equiv_{n_1} -1 \Rightarrow w^{2^j u} \not\equiv_n 1,$$
$$w^{2^j u} \equiv_{n_2} 1^{2^j u} = 1 \Rightarrow w^{2^j u} \not\equiv_n -1.$$

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 kamen 1976 auf die Idee, dass die Geheimhaltung des Chiffrierschlüssels keine notwendige Voraussetzung für die Sicherheit eines Kryptosystems sein muss. Natürlich setzt dies voraus, dass die vom Sender und Empfänger benutzten Schlüssel k und k' voneinander verschieden sind und dass insbesondere der Dechiffrierschlüssel k' nur sehr schwer aus dem Chiffrierschlüssel k berechenbar ist. Ist dies gewährleistet, so kann jedem Kommunikationsteilnehmer X ein Paar von zusammengehörigen Schlüsseln k_X, k'_X zugeteilt werden. X kann nun den Chiffrierschlüssel k_X öffentlich bekannt geben, muss aber den Dechiffrierschlüssel k'_X unter Verschluss halten. Die Tatsache, dass sich mit k_X die Nachrichten nicht wieder entschlüsseln lassen, hat den entscheidenden Vorteil, dass k_X über einen authentisierten Kanal zum Sender gelangen kann (d.h. es ist zwar nicht notwendig, k_X geheim zu halten, aber die Herkunft von k_X muss verifizierbar sein).

- Von einem **symmetrischen Kryptosystem** spricht man, wenn die Kenntnis des Chiffrierschlüssels gleichbedeutend mit der Kenntnis des Dechiffrierschlüssels ist, der eine also leicht aus dem anderen berechnet werden kann.
- Dagegen sind bei einem **asymmetrischen Kryptosystem** nur die Dechiffrierschlüssel geheimzuhalten, während die Chiffrierschlüssel öffentlich bekanntgegeben werden können.

Für symmetrische Kryptosysteme sind auch die Bezeichnungen **konventionelles Kryptosystem**, **Kryptosystem mit geheimen Schlüsseln** oder ***Secret-Key*-Kryptosystem** üblich, wogegen asymmetrische Kryptosysteme auch häufig auch als **Kryptosysteme mit öffentlichen Schlüsseln** oder ***Public-Key*-Kryptosysteme** bezeichnet werden.

Bei einem symmetrischen Kryptosystem sind die Rollen von Sender und Empfänger untereinander austauschbar (beziehungsweise *symmetrisch*), da die Vertraulichkeit der Nachrichten auf einem *gemeinsamen Geheimnis* beruht, welches sich die beiden Kommunikationspartner in Form des zwischen ihnen vereinbarten Schlüssels verschaffen.

Der prinzipielle Unterschied zwischen symmetrischer und asymmetrischer Verschlüsselung kann sehr schön am Beispiel eines Tresors veranschaulicht werden, den Alice dazu verwendet, Bob geheime Dokumente zukommen zu lassen.

Symmetrische Verschlüsselung: Alice und Bob besitzen beide den gleichen Schlüssel k. Alice schließt mit ihrem Schlüssel die Nachricht in den Tresor ein und Bob öffnet ihn später wieder mit seinem Schlüssel. Das Tresorschloss lässt sich also mit k sowohl auf- als auch zuschließen.

Asymmetrische Verschlüsselung: Alice schließt die Nachricht mit dem Schlüssel k_B in den Tresor ein. Danach lässt sich das Tresorschloss mit diesem Schlüssel nicht mehr öffnen. Dies ist nur mit dem in Bobs Besitz befindlichen Schlüssel k'_B möglich. Obwohl also beide Schlüssel in das Schloss passen, können k_B und k'_B jeweils nur in eine von beiden Richtungen gedreht werden.

Da Alice nicht im Besitz von Bobs privatem Schlüssel k'_B ist, kann sie im Gegensatz zu Bob nicht alle mit k_B verschlüsselten Nachrichten entschlüsseln, insbesondere keine Kryptotexte, die Bob von anderen Teilnehmern zugeschickt werden. Dies hat zur Folge,

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 benötigt werden. Zu beachten ist auch, dass mit den beiden Schlüsseln k_B und k'_B von Bob nur eine Nachrichtenübermittlung von Alice an Bob möglich ist. Für die umgekehrte Richtung müssen dagegen die beiden Schlüssel k_A und k'_A von Alice benutzt werden.

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.

Für jeden Teilnehmer des RSA-Kryptosystems werden zwei große Primzahlen p, q und Exponenten e, d mit $ed \equiv_{\varphi(n)} 1$ bestimmt, wobei $n = pq$ und $\varphi(n) = (p-1)(q-1)$ ist.

Öffentlicher Schlüssel: $k = (e, n)$,
Geheimer Schlüssel: $k' = (d, n)$.

Jede Nachricht x wird durch eine Folge $x_1, x_2, \ldots$ von natürlichen Zahlen $x_i < n$ dargestellt, die einzeln wie folgt ver- und entschlüsselt werden.

$$\begin{aligned} E((e,n),x) &= x^e \bmod n, \\ D((d,n),y) &= y^d \bmod n. \end{aligned}$$

Die Chiffrierfunktionen E und D können durch „Wiederholtes Quadrieren und Multiplizieren“ effizient berechnet werden.

Der Schlüsselraum ist also

$$K = \{(c,n) \mid \text{ es ex. Primzahlen } p \text{ und } q \text{ mit } n = pq \text{ und } c \in \mathbb{Z}^*_{\varphi(n)}\}$$

und S enthält alle Schlüsselpaare $((e,n),(d,n)) \in K \times K$ mit $ed \equiv_{\varphi(n)} 1$.

Satz 130. *Für jedes Schlüsselpaar $((e,n),(d,n)) \in S$ und alle $x \in \mathbb{Z}_n$ gilt*

$$x^{ed} \equiv_n x.$$

Beweis. Sei $ed = z\varphi(n)+1$ für eine natürliche Zahl z. 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

$$x^{ed} = x^{z(p-1)(q-1)}x = (x^{p-1})^{z(q-1)}x \equiv_p x.$$

□

Zur praktischen Durchführung

Bestimmung von p und q: Man beginnt mit einer Zahl x der Form $30z$ (mit $z \in \mathbb{Z}$) und der verlangten Größenordnung (z. B. 10^{100}) und führt einen Primzahltest für $x+1$ durch. Ist die Antwort negativ, testet man der Reihe nach die Zahlen $x+7$, $x+11$, $x+13$, $x+17$, $x+19$, $x+23$, $x+29$, $x+30+1$, $x+30+7$, ... bis eine Primzahl gefunden ist. Wegen $\frac{\pi(n)}{n} > \frac{2}{(3\ln n)}$ und da nur 8 von 30 Zahlen getestet werden, sind hierzu ungefähr $\frac{8\cdot 3\cdot \ln x}{30\cdot 2} = {}^2\!/_5 \ln x$ Primzahltests durchzuführen (bei 100-stelligen Dezimalzahlen sind das um die 92 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 Euklidsche Algorithmus das multiplikative Inverse e von d modulo $\varphi(n)$.

Ver- und Entschlüsselung: Modulares Exponentieren durch wiederholtes Quadrieren und Multiplizieren. Es gibt auch Hardware-Implementierungen, die (unter Verwendung des Chinesischen Restsatzes) mit Geschwindigkeiten von bis zu 225 Kbits/sec arbeiten und somit circa 1500 mal langsamer als der DES sind.

Kryptoanalytische Betrachtungen

1. 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 $\frac{c \pm \sqrt{c^2-n}}{2}$ 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.

   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 $\sqrt{x+y} \le \sqrt{x} + \frac{y}{2\sqrt{x}}$ wie folgt nach oben und unten abschätzen lässt:

   $$\frac{b^2}{2a} \le a - \sqrt{a^2 - b^2} = a - \sqrt{n} = \sqrt{n + b^2} - \sqrt{n} \le \frac{b^2}{2\sqrt{n}}.$$

Im Fall $p \geq 2q$ gilt jedoch wegen $b = (p-q)/2 = (p+q)/6+(p-2q)/3 \geq (p+q)/6 = a/3$

$$\frac{b^2}{2a} = \frac{3b}{a} \cdot \frac{b}{6} \geq \frac{b}{6} \geq \frac{q}{12}.$$

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 unten).

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, 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 $k = \text{kgV}(p-1,\ q-1)$ bestimmt und somit n faktorisiert werden. Die Faktorisierung von n beruht auf folgendem Lemma.

Lemma 131. *Seien y, z zwei Lösungen der Kongruenz $x^2 \equiv_n a$ mit $y \not\equiv_n \pm z$. Dann ist $\text{ggT}(y+z,\ n)$ ein nicht-trivialer Faktor von n.*

Beweis. Wegen $y^2 \equiv_n z^2$ existiert ein $t \in \mathbb{Z}$ mit

$$(y+z)(y-z) = y^2 - z^2 = tn.$$

Da jedoch wegen $y \not\equiv_n \pm z$ weder $y+z$ noch $y-z$ durch n teilbar ist, folgt $1 < \text{ggT}(y+z,\ n) < n$. □

Betrachte 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  if ggT(a, n) > 1 then
4    return (zusammengesetzt)
5  b := a
6  for i := r − 1 downto 0 do
7    c := b
8    b := b^2 mod m
9    if b ≡_n 1 ∧ c ≢_n ±1 then
10     return (zusammengesetzt)
11   if e_i = 1 then b := b · a mod n
12 if b ≢_n 1 then return (zus.gesetzt)
13 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  if ggT(a, n) > 1 then
4    return (ggT(a, n))
5  b := a
6  for i := r − 1 downto 0 do
7    c := b
8    b := b^2 mod m
9    if b ≡_n 1 ∧ c ≢_n ±1 then
10     return (ggT(c + 1, n))
11   if e_i = 1 then b := b · a mod n
12 return (?)
```

Satz 132. *Sei $n = pq$ (p, q prim) und $v > 0$ ein Vielfaches von $k = \mathrm{kgV}(p-1, q-1)$. Dann gibt* RSA-Factorize(n, v) *mit Wahrscheinlichkeit $> 1/2$ einen Primfaktor von n aus.*

Beweis. Mit Lemma 131 folgt

$$b \not\equiv_n \pm 1, b^2 \equiv_n 1 \quad \Rightarrow \quad \mathrm{ggT}(b+1,\ n) \in \{p,\ q\},$$

womit die Korrektheit der Ausgabe von RSA-Factorize in Zeile 10 gezeigt ist. In den folgenden Behauptungen schätzen wir die Wahrscheinlichkeit ab, mit der dem Algorithmus eine Faktorisierung von n gelingt.

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. $i \leq j$.

Behauptung 133. $\mathrm{ggT}(2^t u,\ p-1) = 2^{\min(t,i)} u_1$ *und* $\mathrm{ggT}(2^t u,\ q-1) = 2^{\min(t,j)} u_2$.

Wegen $k = \mathrm{kgV}(p-1, q-1) = 2^{\max(i,j)}\,\mathrm{kgV}(u_1, u_2) | v = 2^m u$ folgt $u_1 | u$ und $u_2 | u$. Da nun u ungerade ist, folgt $\mathrm{ggT}(2^t u, p-1) = \mathrm{ggT}(2^t u, 2^i u_1) = 2^{\min(t,i)} u_1$ und $\mathrm{ggT}(2^t u, q-1) = \mathrm{ggT}(2^t u, 2^j u_2) = 2^{\min(t,j)} u_2$.

Behauptung 134. $\underbrace{\|\{a \in \mathbb{Z}_n^* \mid a^u \equiv_n 1\}\|}_{=:\alpha} = u_1 u_2$.

Mit dem Chinesischen Restsatz folgt nämlich

$$\alpha = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^u \equiv_p 1\}\|}_{=:\beta} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^u \equiv_q 1\}\|}_{=:\gamma}.$$

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 $\beta = \mathrm{ggT}(u,\ p-1) \overset{Beh.\ 1}{=} u_1$. Analog folgt $\gamma = u_2$. Für $t \geq 0$ sei $\alpha_t = \|\{a \in \mathbb{Z}_n^* \mid a^{2^t u} \equiv_n -1\}\|$.

Behauptung 135. *Für $t \geq i$ ist $\alpha_t = 0$.*

Es gilt nämlich für alle $a \in \mathbb{Z}_n^*$:

$$t \geq i \Rightarrow 2^t u \equiv_{p-1} 0 \Rightarrow a^{2^t u} \equiv_p 1 \Rightarrow a^{2^t u} \not\equiv_p -1 \Rightarrow a^{2^t u} \not\equiv_n -1.$$

Behauptung 136. *Für $t = 0, \ldots, i-1$ ist $\alpha_t = 2^{2t} u_1 u_2$.*

Mit dem Chinesischen Restsatz folgt zunächst

$$\alpha_t = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^{2^t u} \equiv_p -1\}\|}_{=:\beta_t} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^{2^t u} \equiv_q -1\}\|}_{=:\gamma_t}.$$

Sei nun g ein Erzeuger von $\mathbb{Z}_p^*$. Dann gilt

$$g^{k 2^t u} \equiv_p -1 \quad \Leftrightarrow \quad k 2^t u \equiv_{p-1} \frac{p-1}{2}.$$

Wegen $t < i$ ist $\mathrm{ggT}(2^t u,\ p-1) \overset{Beh.\ 1}{=} 2^t u_1$ ein Teiler von $\frac{p-1}{2} = 2^{i-1}u_1$ und daher ist $\beta_t = \mathrm{ggT}(2^t u,\ p-1) = 2^t u_1$ ($\gamma_t = 2^t u_2$ folgt analog).
Da RSA-Factorize nur dann keinen Primfaktor von n ausgibt, wenn $a \in \mathbb{Z}_n^*$ ist und entweder $a^u \equiv_n \pm 1$ oder $a^{2^t u} \equiv_n -1$ für ein $t \geq 1$ gilt, tritt dieses Ereignis mit Wahrscheinlichkeit $g(n)/(n-1)$ ein, wobei

$$\begin{aligned} g(n) &= \|\{a \in \mathbb{Z}_n^* \mid a^u \equiv_n 1 \ \vee\ \exists t \geq 0 : a^{2^t u} \equiv_n -1\}\| = \alpha + \textstyle\sum_{t\geq 0} \alpha_t \\ &= u_1u_2 + \sum_{t=0}^{i-1} 2^{2t}u_1u_2 = u_1u_2(1 + \sum_{t=0}^{i-1} 2^{2t}) = u_1u_2(1 + \frac{2^{2i}-1}{3}) = u_1u_2(2^{2i}+2)/3 \\ &\leq u_1u_2(2^{i+j} + 2^{i+j-1})/3 = \varphi(n)(1+2^{-1})/3 = \varphi(n)/2 \end{aligned}$$

ist. Da $\varphi(n) < n-1$ ist, folgt $g(n)/(n-1) \leq \varphi(n)/2(n-1) < 1/2$ und damit ist der Satz bewiesen. □

Beispiel 137. *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* $a = 174$, $a' = 137$ *und* $a'' = 111$ *die folgenden Werte* z_i, z_i' *bzw.* z_i''.

i	e_i	c_i	$z_i = 174^{c_i}$	$(z_i)^2$	$z_i' = 137^{c_i}$	$(z_i')^2$	$z_i'' = 111^{c_i}$	$(z_i'')^2$
12	1	1	174	220	137	205	111	166
11	0	2	220	1	205	35	166	152
10	0	4	1	1	35	120	152	120
9	0	8	1	1	120	35	120	35
8	0	16	1	1	35	120	35	120
7	1	33	174	220	$120 \cdot 137 = 86$	103	$120 \cdot 111 = 60$	64
6	0	66	220	1	103	1	64	118
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	1				

RSA-Factorize *gelingt also die Faktorisierung von* $n = 221$ *bei Wahl von* $a = 174$ *nicht, wohl aber bei Wahl von* $a = 137$ *und* $a = 111$. *Im ersten Fall findet* RSA-Factorize *den Faktor* $\mathrm{ggT}(103+1, 221) = 13$ *und im zweiten den Faktor* $\mathrm{ggT}(118+1, 221) = 17$. ◁

Als nächstes gehen wir der Frage nach, wie sicher einzelne Bits der Klartextnachricht sind. Falls es möglich wäre, aus n, e, $y = x^e \bmod n$ die Parität von x in Polynomialzeit zu bestimmen, so könnte auch der gesamte Klartext x in Polynomialzeit aus n, e und y berechnet werden. Das letzte Bit des Klartextes ist also genau so sicher wie der gesamte Klartext. Falls RSA nicht total gebrochen werden kann, kann auch nicht das letzte Bit des Klartextes ermittelt werden.

Sei nämlich

$$\text{klartext-parity}(y) = \text{parity}(x) = \begin{cases} 1 & \text{falls } x \text{ ungerade,} \\ 0 & \text{falls } x \text{ gerade.} \end{cases}$$

und

$$\text{klartext-half}(y) = \text{half}(x) = \begin{cases} 0 & \text{falls } 0 \le x \le (n-1)/2, \\ 1 & \text{falls } (n+1)/2 \le x \le n-1 \end{cases}$$

Wegen

$$2x \bmod n = \begin{cases} 2x & \text{half}(x) = 0, \\ 2x - n & \text{sonst} \end{cases}$$

gilt dann $\text{half}(x) = \text{parity}(2x \bmod n)$. 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 x/n als Binärzahl

$$x/n = \sum_{i=1}^{\infty} b_i 2^{-i}$$

dar, so berechnet sich die Bitfolge b_i, $i = 1, 2, \ldots$ zu

$$b_i = \text{half}(2^{i-1}x \bmod n) = \text{parity}(2^i x \bmod n) = \text{klartext-parity}(2^{ie} y \bmod n).$$

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 (da $\sum_{i=\lceil \log_2 n \rceil + 1}^{\infty} (n/2^i) = n/2^{\lceil \log_2 n \rceil} < 1$ ist):

```
1  x := 0
2  for i := 1 to ⌈log2 n⌉ do
3     y := 2^e y mod n
4     if klartext-parity(y) then
5        x := x + n/2^i
6  output ⌈x⌉
```

Beispiel 138. *Sei $n = 1\,457$, $e = 779$ und $y = 722$. Angenommen, das Orakel klartext-parity liefert die in folgender Tabelle angegebenen Werte $b_i = $ klartext-parity(y_i) für $y_i = 2^{ie} y \bmod n$. Dann berechnet obiger Algorithmus die zugehörigen Werte $x_i = n \sum_{j=1}^{i} b_i/2^i$ (siehe Tabelle).*

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
$n/2^i$	728,5	364,3	182,1	91,1	45,5	22,8	11,4	5,7	2,8	1,4	0,7
x_i	728,5	728,5	910,6	910,6	956,2	978,9	990,3	996	998,8	998,8	998,8
z_i	541	1082	707	1414	1371	1285	1113	769	81	162	324

Der gesuchte Klartext ist also $x = \lceil 998,8 \rceil = 999$. Dass dieser tatsächlich die vorgegebene Paritätsbitfolge generiert, kann anhand der Werte $z_i = 2^i x \bmod n$ in der letzten Tabellenzeile überprüft werden. ◁

7.2 Quadratische Reste

In diesem Abschnitt beschäftigen wir uns mit dem Problem, Lösungen für eine quadratische Kongruenzgleichung

$$x^2 \equiv_m a \tag{7.1}$$

zu bestimmen. Zunächst gehen wir der Frage nach, wie sich feststellen lässt, ob überhaupt Lösungen existieren.

Definition 139. *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 $x^2 \equiv_m a$. $\mathsf{QNR}_m := \mathbb{Z}_m^* \setminus \mathsf{QR}_m$ ist die Menge der **quadratischen Nichtreste** modulo m.*
Sei $p > 2$ eine Primzahl und $a \in \mathbb{Z}_p$. Dann heißt

$$\mathcal{L}(a,p) = \left(\frac{a}{p}\right) = \begin{cases} 1, & a \in \mathsf{QR}_p \\ -1, & a \in \mathsf{QNR}_p \\ 0, & \text{sonst} \end{cases}$$

*das **Legendre-Symbol** von a modulo p.*

Die Kongruenzgleichung (7.1) besitzt also für ein $a \in \mathbb{Z}_m^*$ genau dann eine Lösung, wenn $a \in \mathsf{QR}_m$ ist. Wie das folgende Lemma zeigt, kann die Lösbarkeit von (7.1) für primes m effizient entschieden werden. Am Ende dieses Abschnitts werden wir noch eine andere Methode zur effizienten Berechnung des Legendre-Symbols kennenlernen.

Lemma 140. *Sei $a \in \mathbb{Z}_p^*$, $p > 2$ prim, und sei $k = \log_{p,g}(a)$ für einen beliebigen Erzeuger g von $\mathbb{Z}_p^*$. Dann sind die folgenden drei Bedingungen äquivalent:*

1. $a^{(p-1)/2} \equiv_p 1$,
2. *k ist gerade,*
3. $a \in \mathsf{QR}_p$.

Beweis.
$1 \Rightarrow 2$: Angenommen, $a \equiv_p g^k$ für ein ungerades $k = 2 \cdot j + 1$. Dann ist

$$a^{(p-1)/2} \equiv_p \underbrace{g^{j\cdot(p-1)}}_{=1} \underbrace{g^{(p-1)/2}}_{\neq 1} \equiv_p g^{(p-1)/2} \not\equiv_p 1.$$

$2 \Rightarrow 3$: Ist $a \equiv_p g^k$ für $k = 2j$ gerade, so folgt $a \equiv_p (g^j)^2$, also $a \in \mathsf{QR}_p$.
$3 \Rightarrow 1$: Sei $a \in \mathsf{QR}_p$, d. h. $b^2 \equiv_p a$ für ein $b \in \mathbb{Z}_p^*$. Dann folgt mit dem Satz von Fermat,

$$a^{(p-1)/2} \equiv_p b^{p-1} \equiv_p 1.$$

□

Satz 141 (Eulers Kriterium)**.** *Für alle a und $p > 2$ prim gilt*

$$a^{(p-1)/2} \equiv_p \left(\frac{a}{p}\right).$$

Beweis. Nach obigem Lemma reicht es zu zeigen, dass für alle $a \in \mathbb{Z}_p^*$ die Kongruenz $a^{(p-1)/2} \equiv_p \pm 1$ gelten muss. Da jedoch die Kongruenz $x^2 \equiv_p 1$ nach dem Satz von Lagrange nur die beiden Lösungen 1 und -1 hat, folgt dies aus der Tatsache, dass $a^{(p-1)/2}$ Lösung dieser Kongruenz ist. □

Korollar 142. *Für alle $a, b \in \mathbb{Z}_p^*$, $p > 2$ prim, gilt*

1. $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2} = \begin{cases} 1, & p \equiv_4 1, \\ -1, & p \equiv_4 3, \end{cases}$
2. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right)$.

Als weiteres Korollar aus Eulers Kriterium erhalten wir eine Methode, quadratische Kongruenzgleichungen im Fall $p \equiv_4 3$ zu lösen. Für beliebige Primzahlen p ist kein effizienter, deterministischer Algorithmus bekannt. Es gibt jedoch einen probabilistischen Algorithmus von Adleman, Manders und Miller (1977).

Korollar 143. *Sei $p > 2$ prim, dann besitzt die quadratische Kongruenzgleichung $x^2 \equiv_p a$ für jedes $a \in \mathsf{QR}_p$ genau zwei Lösungen. Im Fall $p \equiv_4 3$ sind dies $\pm a^k \bmod p$ (für $k = (p+1)/4$), wovon nur $a^k \bmod p$ ein quadratischer Rest ist.*

Beweis. Sei $a \in \mathsf{QR}_p$, d. h. es existiert ein $b \in \mathbb{Z}_p^*$ mit $b^2 \equiv_p a$. Mit b ist auch $-b$ eine Lösung von $x^2 \equiv_p a$, die von b verschieden ist (p ist ungerade). Nach Lagrange existieren keine weitere Lösungen.
Sei nun $p \equiv_4 3$. Dann gilt

$$\left(\frac{-b}{p}\right) = \left(\frac{-1}{p}\right) \cdot \left(\frac{b}{p}\right) = -\left(\frac{b}{p}\right)$$

nach Korollar 142. Demnach ist genau eine der beiden Lösungen $\pm b$ ein quadratischer Rest. Schließlich liefert Eulers Kriterium die Kongruenz $a^{(p-1)/2} \equiv_p 1$. Daher folgt 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 $a^k \bmod p$ ein quadratischer Rest ist, 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 mindestens so schwer zu brechen ist wie das Faktorisierungsproblem, ist genau dies beim Rabin-System der Fall. Die Sicherheit des Rabin-Systems ist also äquivalent zur Schwierigkeit des Faktorisierungsproblems. Es verwendet als Falltürfunktion eine quadratische Polynomfunktion modulo dem Produkt $n = pq$ zweier großer Primzahlen. Jeder Teilnehmer wählt ein Paar p, q von Primzahlen mit der Eigenschaft $p \equiv_4 q \equiv_4 3$ und eine Zahl $e \in \{0, \ldots, n-1\}$. Die Zahlen n und e werden öffentlich bekannt gegeben, die Faktorisierung von n wird jedoch geheimgehalten.

Öffentlicher Schlüssel: $e,\ n$
Geheimer Schlüssel: $p,\ q.$

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 B die quadratische Kongruenzgleichung $x(x+e) \equiv_n y$ lösen, die äquivalent zu der Kongruenz

$$(\underbrace{x + 2^{-1}e}_{x'})^2 \equiv_n y + (2^{-1}e)^2,$$

(*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. Falls $y' \equiv_n 0$ ist, gibt es nur eine Lösung $x' = 0$. Ist dagegen $\mathrm{ggT}(y', n) \in \{p, q\}$, so gibt es zwei Lösungen (dieser Fall ist glücklicherweise sehr unwahrscheinlich, da er dem Gegner die Faktorisierung von n ermöglicht). Im verbliebenen Fall $y' \in \mathbb{Z}_n^*$ gibt es vier Lösungen und der folgende Satz zeigt, wie sich diese bei Kenntnis von p und q effizient bestimmen lassen.

Satz 144. *Sei $n = pq$ für Primzahlen p, q mit $p \equiv_4 q \equiv 3$. Dann besitzt die quadratische Kongruenz $x^2 \equiv_n a$ für jedes $a \in \mathsf{QR}_n$ genau vier Lösungen, wovon genau eine ein quadratischer Rest ist.*

Beweis. Mit $x^2 \equiv_n a$ besitzen wegen $n = pq$ auch die beiden quadratischen Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ Lösungen, und zwar jeweils genau zwei (siehe Korollar 143): $y_1 = a^{(p+1)/4} \bmod p \in \mathsf{QR}_p$, $y_2 = -a^{(p+1)/4} \bmod p \in \mathsf{QNR}_p$, $z_1 = a^{(q+1)/4} \bmod q \in \mathsf{QR}_q$ und $z_2 = -a^{(q+1)/4} \bmod q \in \mathsf{QNR}_q$. Mit dem Chinesischen Restsatz können wir vier verschiedene Lösungen $x_{i,j}$, $1 \le i, j \le 2$ mit

$$\begin{aligned} x &\equiv_p y_i \\ x &\equiv_q z_j \end{aligned}$$

bestimmen. Es können aber auch nicht mehr als diese vier Lösungen existieren, da sich daraus für mindestens eine der beiden Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ mehr als zwei Lösungen ableiten ließen.

Wegen

$$\begin{aligned} x \in \mathsf{QR}_n \quad &\Rightarrow \quad \exists u : u^2 \equiv_n x \\ &\Rightarrow \quad \exists u : u^2 \equiv_p x \equiv_q u^2 \\ &\Rightarrow \quad x \bmod p \in \mathsf{QR}_p \wedge x \bmod q \in \mathsf{QR}_q \end{aligned}$$

können $x_{1,2}, x_{2,1}, x_{2,2}$ keine quadratischen Reste modulo n sein.

Weiterhin gibt es Zahlen $l \in \mathbb{Z}_p^*$, $k \in \mathbb{Z}_q^*$ mit $l^2 \equiv_p y_1$ und $k^2 \equiv_q z_1$. Sei $m \in \mathbb{Z}_n^*$ eine Lösung für das System

$$\begin{aligned} x &\equiv_p l \\ x &\equiv_q k \end{aligned}$$

Dann folgt

$$x_{1,1} \equiv_p y_1 \equiv_p l^2 \equiv_p m^2 \quad \text{und} \quad x_{1,1} \equiv_q z_1 \equiv_q k^2 \equiv_q m^2$$

und daher $x_{1,1} \equiv_n m^2$. Also ist $x_{1,1}$ ein quadratischer Rest modulo n. □

Als weitere zahlentheoretische Funktion mit für die Kryptografie wichtigen Eigenschaften erhalten wir somit die Quadratfunktion $x^2 : \mathsf{QR}_n \to \mathsf{QR}_n$, die nach vorigem Satz bijektiv ist (falls $n = pq$ für Primzahlen p, q mit $p \equiv_4 q \equiv 3$). Ihre Umkehrfunktion $x \mapsto \sqrt{x}$ heißt **diskrete Wurzelfunktion**, und kann (nur) bei Kenntnis der Primfaktoren p und q von n effizient berechnet werden. Wir werden später sehen, dass die Berechnung dieser Funktion äquivalent zur Faktorisierung von n ist. Es ist nicht einmal ein effizientes Verfahren bekannt, mit dem man ohne Kenntnis der Faktorisierung von n für ein gegebenes $a \in \mathbb{Z}_n^*$ entscheiden kann, ob $a \in \mathsf{QR}_n$ ist oder nicht. Aus diesem Grund können wir die Rabin-Verschlüsselung nicht einfach injektiv machen, indem wir den Klartextraum auf QR_n einschränken.

Beispiel 145. *Wählen wir $p = 7$, $q = 11$ und $e = 2$, so ergeben sich*

Öffentlicher Schlüssel: $e = 2,\ n = 77$
Geheimer Schlüssel: $p = 7,\ q = 11.$

Um den Klartext $x = 12$ zu verschlüsseln, wird der Kryptotext

$$y = E(12; 2, 77) = 12(12 + 2) \bmod 77 = 14$$

erzeugt. 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

$$y^2 \equiv_7 15 \equiv 1 \quad \textit{und} \quad z^2 \equiv_{11} 15 \equiv 4$$

zu $y_{1,2} = \pm 15^2 \bmod 7 = \pm 1$ (wegen $\frac{p+1}{4} = 2$) und $z_{1,2} = \pm 4^3 \bmod 11 = \pm 2$ (wegen $\frac{q+1}{4} = 3$). Mit dem Chinesischen Restsatz lassen sich $y_{1,2}$ und $z_{1,2}$ zu den vier Lösungen $x'_{i,j} = 57, 64, 13$ und 20 zusammensetzen, die auf die vier Klartextkandidaten $12, 19, 56$ und 63 führen.
Durch Reduktion von $M = \mathbb{Z}_{77}$ auf $M' = \{x \in \mathbb{Z}_{77} \mid x + 1 \bmod n \leq 38\} = \{0, 1, \ldots, 37, 76\}$ entfallen die beiden Kandidaten 56 und 63. Wurde dem Empfänger zudem das Bit $b = \left(\frac{x'}{n}\right) = \left(\frac{13}{77}\right)$ übermittelt, so kann er den Klartext $x = 12$ sogar eindeutig bestimmen. ◁

Es ist klar, dass das System gebrochen ist, sobald n in seine Primfaktoren p, q zerlegt werden kann. Wie wir gleich sehen werden, sind für Zahlen n von dieser Bauart das Faktorisierungsproblem und das Problem, eine Lösung der quadratischen Kongruenz $x^2 \equiv_n a$ für ein gegebenes $a \in \mathsf{QR}_n$ zu finden, äquivalent. Kann also das Rabin-System gebrochen werden, so muss ein effizienter Algorithmus A existieren, der bei Eingabe (c, n), $c \in \mathsf{QR}_n$, eine Zahl $a = A(c, n)$ mit $a^2 \equiv_n d$ berechnet. Unter Verwendung von A lässt sich folgender effizienter probabilistischer Algorithmus RABIN-FACTORIZE angeben, der n faktorisiert.

RABIN-FACTORIZE(n)

```
1  repeat forever
2    guess randomly a ∈ {1, ..., (n−1)/2}
3    if ggT(a, n) > 1 then
4      return (ggT(a, n))
```

```
5    c := a^2 mod n
6    b := A(c, n)
7    if b ≢_n ±a then
8      return (ggT(a + b, n))
```

Satz 146. *Falls* Rabin-Factorize(n) *hält, gibt er einen Primfaktor von* $n = pq$*,* p, q *prim, aus. Die Wahrscheinlichkeit, dass hierfür mehr als* t *Schleifendurchläufe benötigt werden, ist kleiner als* 2^{-t}*.*

Beweis. Es ist klar, dass $\mathrm{ggT}(a, n)$ im Fall $\mathrm{ggT}(a, n) > 1$ ein Primfaktor von n ist. Lemma 131 zeigt, dass dies im Fall $b \not\equiv_n \pm a$ auch für $\mathrm{ggT}(a + b, n)$ gilt.

Sei nun X die ZV, die die Wahl von a beschreibt, und sei α die Wahrscheinlichkeit, dass der Algorithmus in einem Schleifendurchlauf einen Primfaktor findet. Wir nehmen o.B.d.A. an, dass $A(c, n)$ im Fall $c \in \mathsf{QR}_n$ eine Zahl b in der Menge $U = \mathbb{Z}_n^* \cap \{1, \ldots, \frac{n-1}{2}\}$ zurückliefert. Für $a \in U$ sei a' die in $U - \{a\}$ eindeutig bestimmte Lösung x von $x^2 \equiv_n a^2$. Dann gilt

$$\alpha \;=\; \underbrace{\Pr[\mathrm{ggT}(X, n) > 1]}_{\Pr[X \notin U] =: \beta} + \underbrace{\Pr[X \in U \text{ und } X \not\equiv_n \pm A(X^2, n)]}_{\gamma}.$$

mit

$$\begin{aligned}\gamma \;&=\; \sum_{a \in U} \Pr[X \in U \wedge A(X^2, n) = a] \underbrace{\Pr[X = a' \mid X \in U \wedge A(X^2, n) = a]}_{1/2} \\ &=\; \frac{1}{2} \sum_{a \in U} \Pr[X \in U \wedge A(X^2, n) = a] = \frac{1}{2} \Pr[X \in U] = (1 - \beta)/2.\end{aligned}$$

Somit ist $\alpha = \beta + \gamma = (\beta + 1)/2 > 1/2$, d.h. die Wahrscheinlichkeit, dass Rabin-Factorize mehr als t Schleifendurchläufe ausführt, ist $(1 - \alpha)^t < 2^{-t}$. □

7.4 Das Diffie-Hellman-Protokoll

Wie Diffie und Hellman bereits 1976 darlegten, kann eine kommutative Kryptofunktion E mit der Eigenschaft

$$E(k_1, E(k_2, x)) = E(k_2, E(k_1, x))$$

als Grundbaustein für ein einfaches Protokoll zur Schlüsselvereinbarung dienen. Um in den Besitz eines gemeinsam erzeugten Schlüssels $k_{A,B}$ zu kommen, der allen anderen verborgen bleibt, gehen Alice und Bob wie folgt vor:

> Alice und Bob wählen je einen geheimen Schlüssel $\overline{k}_A$ und $\overline{k}_B$ und geben die zugehörigen Einwegfunktionswerte $k_A = E(\overline{k}_A, x)$ und $k_B = E(\overline{k}_B, x)$ öffentlich bekannt. Auf diese Art kann der symmetrische Schlüssel
>
> $$k_{A,B} = E(\overline{k}_A, k_B) = E(\overline{k}_B, k_A)$$
>
> sowohl von Alice (aus $\overline{k}_A$ und k_B) als auch von Bob (aus $\overline{k}_B$ und k_A) berechnet werden.

Konkret haben Diffie und Hellman vorgeschlagen, für E die modulare Exponentialfunktion $E(k, x) = x^k \bmod p$ für eine große Primzahl p und für x einen Erzeuger g von $\mathbb{Z}_p^*$ zu verwenden.

Beispiel 147. *Wählen wir $p = 29$ und $g = 2$, so haben die beiden Teilnehmer A und B mit geheimen Schlüsseln $\overline{k}_A = 5$ und $\overline{k}_B = 12$ die öffentlichen Schlüssel $k_A = 2^5 \bmod 29 = 3$ und $k_B = 2^{12} \bmod 29 = 7$. Der gemeinsame Schlüssel k_{AB} für ein geeignetes klassisches Kryptosystem berechnet sich dann zu $k_{AB} \equiv 2^{5 \cdot 12} \equiv 3^{12} \equiv 7^5 \equiv 16 (\text{mod } 29)$.*

Es ist kein effizientes Verfahren bekannt, mit dem k_{AB} nur unter Kenntnis von p, g, k_A und k_B berechnet werden kann (dieses Problem wird als Diffie-Hellman Problem bezeichnet). Notwendig für die Sicherheit des Verfahrens ist, daß die modulare Exponentialfunktion $(x, g, p) \mapsto (g^x \bmod p, g, p)$ eine Einwegfunktion ist, da sonst leicht aus dem öffentlichen k_A das zugehörige geheime $\overline{k}_A$ berechnet werden könnte.

Man-in-the-Middle-Angriff auf das Diffie-Hellman-Protokoll

Um nicht einem sogenannten Man-in-the-Middle Angriff zum Opfer zu fallen, müssen Alice und Bob für den Austausch der beiden öffentlichen Schlüssel k_A und k_B einen authentisierten Kanal benutzen. Falls sie sich nämlich nicht davon überzeugen, daß die Schlüssel k_A und k_B tatsächlich von Alice bzw. Bob stammen (Schlüsselauthentifizierung), könnte es einem Gegner gelingen, sich Alice gegenüber als Bob und Bob gegenüber als Alice auszugeben.

Gelingt ein solcher Angriff, so hat dies zur Folge, daß Alice und Bob nach Ablauf des Protokolls zwar in den Besitz von gewissen Schlüsseln gelangt sind, nicht jedoch in den Besitz eines gemeinsamen Schlüssels $k_{A,B}$. Vielmehr berechnet hierbei Alice einen Schlüssel $k_{A,G}$, von dem sie annimmt, er sei mit Bob vereinbart. Bob wiederum ist der festen Überzeugung, er würde in seinem Schlüssel $k_{G,B}$ ein gemeinsames Geheimnis mit Alice hüten. In Wirklichkeit kennt (außer Alice) nur der Gegner den Schlüssel $k_{A,G}$, und auch der Schlüssel $k_{G,B}$ ist (außer Bob) nur dem Gegner bekannt. Verschlüsselt nun Alice ihre Nachrichten nichtsahnend mit $k_{A,G}$ und schickt sie diese an Bob, so kann der Gegner diese Nachrichten abfangen und mühelos entschlüsseln.

Darüber hinaus kann er den Inhalt der Nachrichten manipulieren und die veränderten Nachrichten mit $k_{G,B}$ verschlüsselt an Bob weiterleiten.

7.5 Shamirs *No-Key*-Protokoll

Ein weiteres interessantes Protokoll zur Schlüsselweitergabe wurde von Shamir entworfen. Mit diesem kann der Sender einen symmetrischen Schlüssel (oder sonst eine geheime Nachricht x) an einen Kommunikationspartner übermitteln, ohne zuvor einen Schlüssel mit diesem vereinbart zu haben (daher der Name No-Key-Protokoll). Ähnlich wie beim Diffie-Hellman-Protokoll kann bei der Ausführung des *No-Key*-Protokolls auf die Mitwirkung einer dritten Partei verzichtet werden. Im Gegensatz zum Diffie-Hellman-Protokoll wird bei Shamirs *No-Key*-Protokoll jedoch kein Schlüssel *vereinbart*, sondern lediglich ein zuvor auf andere Art erzeugter Schlüssel *übermittelt*.

Das Funktionsprinzip von Shamirs *No-Key*-Protokoll läßt sich sehr gut durch eine physikalische Analogie beschreiben.

> Alice und Bob verwenden einen Tresor, dessen Tür mit zwei unterschiedlichen Schlössern verriegelbar ist. Mit dem in Alice' Besitz befindlichen Schlüssel k_A läßt sich nur das eine der beiden Schlösser öffnen und schließen, während in das andere Schloß nur Bobs Schlüssel k_B paßt. Alice schließt die zu überbringende

Nachricht in den Tresor ein, indem sie das erste Schloß mit ihrem Schlüssel k_A verschließt. Nachdem Bob den Tresor erhalten hat, schließt er die bereits von Alice verschlossene Tür ein zweites Mal ab, indem er auch das zweite Schloß mit seinem Schlüssel k_B verschließt. Hat nun Alice den zweifach abgeschlossenen Tresor von Bob zurückbekommen, so öffnet sie das erste Schloß wieder und schickt den Tresor auf die dritte Etappe seiner Reise. Bob schließlich ist nach Erhalt des Tresors in der Lage, auch das zweite Schloß wieder zu entriegeln und die Tresortür zu öffnen.

Erfüllt die Verschlüsselungsfunktion E eines symmetrischen Kryptosystems die oben angegebene Kommutativitätseigenschaft, so kann Alice Bob eine geheime Nachricht x wie folgt zukommen lassen.

Alice verschlüsselt x mit dem nur ihr selbst bekannten Schlüssel k_A und schickt das Resultat $y = E(k_A, x)$ an Bob. Dieser verschlüsselt den empfangenen Kryptotext y mit seinem geheimen Schlüssel k_B erneut zu $z = E(k_B, E(k_A, x))$ und sendet z zurück an Alice. Aufgrund der Kommutativitätseigenschaft gilt

$$z = E(k_B, E(k_A, x)) = E(k_A, E(k_B, x)),$$

das heißt, durch Entschlüsseln von z mit ihrem geheimen Schlüssel k_A erhält Alice den Kryptotext $E(k_B, x)$, den sie an Bob zurücksendet. Bob schließlich bereitet es keine Schwierigkeiten, daraus die ursprüngliche Nachricht x zu erhalten.

Es versteht sich von selbst, daß auch bei diesem Protokoll ein authentisierter Kanal benötigt wird, bzw. wirkungsvolle Authentifizierungsmaßnahmen zur Abwehr eines Man-in-the-Middle-Angriffs zu ergreifen sind.

7.6 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 B erhält als geheimen Schlüssel eine Zahl $a \in \mathbb{Z}_p^* = \{1, \ldots, p-1\}$ und gibt $\beta = \alpha^a \bmod p$ öffentlich bekannt:

Öffentlicher Schlüssel: (p, α, β),
Geheimer Schlüssel: a.

Will A nun eine Nachricht $x \in \mathbb{Z}_p^* = \{1, \ldots, p-1\}$ an B senden, so

- wählt A zufällig eine Zahl $z \in \{1, \ldots, p-1\}$,
- berechnet den „Schlüssel“ $k = \beta^z \bmod p$ und
- sendet den Kryptotext $E((p, \alpha, \beta), x) = (y_1, y_2)$ an B, wobei

$$y_1 = \alpha^z \bmod p \qquad \text{und} \qquad y_2 = kx \bmod p$$

ist.

Der Kryptotext wird also auf die doppelte Länge des Klartextes aufgebläht. Nachdem B das Kryptotextpaar (y_1, y_2) empfangen hat,

- berechnet B zunächst $k = y_1^a \bmod p$ (es gilt $y_1^a \equiv_p \alpha^{za} \equiv_p \beta^z \equiv_p k$),
- und damit den Klartext $x = y_2 k^{-1} \bmod p$.

Beispiel 148. *Wir wählen* $p = 2579$ *und* $\alpha = 2$. *B wählt einen geheimen Schlüssel* $a = 765$ *aus der Menge* $\{1, \ldots, 2578\}$ *und gibt die Zahl* $\beta = \alpha^a \bmod p = 2^{765} \bmod 2579 = 949$ *bekannt. Um den Klartext* $x = 1299$ *an B zu übermitteln, berechnet A den zugehörigen Kryptotext* (y_1, y_2) *wie folgt.*

- *A wählt zufällig eine Zahl z aus der Menge* $\{1, \ldots, 2578\}$ *(z. B.* $z = 853$*),*
- *berechnet* $k = 949^{853} \bmod 2579 = 2424$ *und*
- *sendet das Kryptotextpaar* $(2^{853} \bmod 2579,\ 1299 \cdot 2424 \bmod 2579) = (435, 2396)$.

Nachdem B den Kryptotext $y = (435, 2396)$ *empfangen hat, berechnet er* $k = 435^{765} \bmod 2579 = 2424$ *und daraus den Klartext* $x = 2396 \cdot 2424^{-1} \bmod 2579 = 1299$. ◁

Es ist klar, dass das ElGamal-System gebrochen ist, sobald 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 $\mathbb{Z}_p^*$ sehr schwer zu berechnen ist. Hierfür sollte p mindestens eine 300-stellige Dezimalzahl sein. Wie der Satz von Pohlig und Hellman im nächsten Abschnitt zeigt, ist es für die Sicherheit des Systems weiterhin notwendig, dass $p - 1$ mindestens einen großen Primfaktor enthält.

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 $k = \beta^z$ in QR_p ist oder nicht. Da zudem $y_2 = kx \bmod p$ genau dann in QR_p ist, wenn entweder $k, x \in \mathsf{QR}_p$ oder $k, x \notin \mathsf{QR}_p$ sind, lässt sich auch leicht ermitteln, ob x in QR_p ist oder nicht. Ersteres trifft nämlich genau dann zu, wenn y_2 und k denselben Status bzgl. Zugehörigkeit zu QR_p haben.

Daraus ergibt sich unmittelbar, dass ein Gegner (G, V) einen Vorteil von $\alpha(G, V) = 1/2$ erzielen kann, indem G Klartexte $x_0 \in \mathsf{QR}_p$ und $x_1 \notin \mathsf{QR}_p$ generiert, und V anhand des Kryptotextes $(y_1, y_2) = E((p, \alpha, \beta), x_b)$ das Bit b ermittelt. Um diese Schwachstelle zu beheben, genügt es, von der Gruppe $\mathbb{Z}_p^*$ zu einer zyklischen Untergruppe von QR_p überzugehen. Dies lässt sich beispielsweise erreichen, indem man p von der Form $p = 2q+1$ mit p, q prim wählt und α durch α^2 ersetzt. In diesem Fall ist nämlich $\mathsf{QR}_p = [\alpha^2]$ eine zyklische Untergruppe (der Ordnung q) von $\mathbb{Z}_p^*$.

Als nächstes gehen wir der Frage nach, wie sicher die einzelnen Bits des privaten Schlüssels a sind. Da a genau dann gerade ist, wenn $\beta = \alpha^a$ 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. $a = L_k(\beta) \cdots L_0(\beta)$ für $k = \lfloor \log_2(p-2) \rfloor$), 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 $m = 1$ und $p \equiv_4 3$. Angenommen, wir könnten nicht nur $L_0(\beta)$, sondern auch $L_1(\beta)$ effizient berechnen. Dann berechnen wir $L_2(\beta)$ wie folgt. Zuerst setzen wir das niederwertigste Bit auf 0, indem wir β' im Fall $L_0(\beta) = 1$ zu $\beta' = \beta\alpha^{-1} \bmod p$ berechnen andernfalls $\beta' = \beta$ setzen. Als nächstes berechnen wir die beiden Quadratwurzeln $\gamma_{1,2}$ von β', d.h. $\gamma_{1,2} = \pm\beta'^{(p+1)/4} \bmod p$. Es ist klar, dass sich

die diskreten Logarithmen von β' und von einer dieser beiden Wurzeln nur durch einen Rechtsshift um eine Stelle unterscheiden, d.h. $L_{i+1}(\beta') = L_i(\gamma_j)$ für $i = 0, \ldots, k-1$. Welche Wurzel γ_j dies ist, können wir an der Stelle $i = 0$ erkennen, da $L_0(\gamma_1) \neq L_0(-\gamma_1) = L_0(\gamma_2)$ ist. Damit ist $L_2(\beta) = L_1(\gamma_j)$ und wir haben zudem das Problem, die Bits $L_k(\beta), \ldots, L_0(\beta)$ zu bestimmen, auf das Problem reduziert, die Bits $L_{k-1}(\gamma_j), \ldots, L_0(\gamma_j)$ zu bestimmen. Indem wir dies wiederholen, haben wir spätestens nach $k-1$ 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 := βα^{-a_0} mod p
3  i := 1
4  while β ≠ 1 do
5     a_i := L_1(β)
6     if L_0(γ) = a_i then β := γ else β := p − γ
7     β := βα^{-a_i} mod p
8     i := i + 1
9  return (a_{i−1} ··· a_0)_2
```

Beispiel 149. *Sei $p = 19$, $\alpha = 2$ und $\beta = 6$. Dann berechnet der Algorithmus die folgenden Werte.*

i	a_i	γ	β	$\beta\alpha^{-a_i}$
0	0	-	-	6
1	1	5	14	7
2	1	11	8	4
3	1	17	2	1

Die Ausgabe ist also $(a_3 \cdots a_0)_2 = (1110)_2 = 14$. ◁

7.7 Quadratische Pseudoreste

Wir erweitern nun das Legendre-Symbol zum *Jacobi-Symbol* und zeigen, wie sich damit das Problem der eindeutigen Entschlüsselung lösen lässt.

Definition 150 (Jacobi-Symbol)**.** *Das* Jacobi-Symbol *ist für alle a und alle ungeraden $m > 3$ 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 $p_1^{e_1} \cdots p_r^{e_r}$ die Primfaktorzerlegung von m ist. 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.

Allerdings überträgt sich die in Teil 1 von Korollar 142 festgehaltene Eigenschaft des Legendre-Symbol 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 Euklidschen Algorithmus, aus folgenden Gleichungen.

$$\left(\frac{a}{m}\right) = \begin{cases} 1, & \text{falls } a = 1 \\ \left(\frac{m \bmod a}{a}\right) \cdot (-1)^{(a-1)(m-1)/4}, & \text{falls } a \text{ ungerade} \\ \left(\frac{b}{m}\right), & \text{falls } a = 2^{2k} \cdot b,\ b \text{ ungerade} \\ \left(\frac{b}{m}\right) \cdot (-1)^{(m^2-1)/8}, & \text{falls } a = 2^{2k+1} \cdot b,\ b \text{ ungerade.} \end{cases}$$

□

Beispiel 154. *Das Jacobi-Symbol von 73 modulo 83 ist*

$$\mathcal{J}(73, 83) = \left(\frac{73}{83}\right) = \left(\frac{83}{73}\right) = \left(\frac{2}{73}\right) \cdot \left(\frac{5}{75}\right) = \left(\frac{73}{5}\right) = \left(\frac{5}{3}\right) = -1.$$

Sei $n = pq$ das Produkt zweier Primzahlen p, q mit der Eigenschaft $p \equiv_4 q \equiv_4 3$. Wir wissen bereits, dass die Berechnung der Umkehrfunktion $\sqrt{x} : \mathsf{QR}_n \to \mathsf{QR}_n$ der Quadratfunktion $x^2 : \mathsf{QR}_n \to \mathsf{QR}_n$ äquivalent zur Faktorisierung von n ist. Folglich ist die diskrete Wurzelfunktion $\sqrt{x}$ schwer zu berechnen, falls die Faktorisierung von n schwer ist. Man nimmt sogar an, dass bereits die Frage, ob eine gegebene Zahl $x \in \mathbb{Z}_n^*$ ein quadratischer Rest, ein schwieriges Problem ist. Da dieses Problem für Eingaben x mit Jacobisymbol $\left(\frac{x}{n}\right) = -1$ trivial ist, schließt man sie üblicherweise von der Betrachtung aus.

Quadratische-Reste-Problem (QR-Problem):

Gegeben: Zahlen n und $x \in \mathbb{Z}_n^*$ mit Jacobisymbol $\left(\frac{x}{n}\right) = 1$, wobei n das Produkt zweier unbekannter Primzahlen ist.

Gefragt: Ist $x \in \mathsf{QR}_n$?

Beim QR-Problem geht es also darum, quadratische Reste von quadratischen Pseudoresten zu unterscheiden.

7.8 Das Goldwasser-Micali-System

Im System von Goldwasser und Micali aus dem Jahr 1984 wird jedes einzelne Klartextbit x_i durch einen zufälligen quadratischen Rest $y_i \in \mathsf{QR}_n$ (im Fall $x_i = 0$) bzw. durch einen zufälligen quadratischen Pseudorest $y_i \in \widetilde{\mathsf{QR}}_n$ (im Fall $x_i = 1$) verschlüsselt. Da der legale Empfänger die Primfaktoren von $n = pq$ kennt, kann er effizient zwischen den beiden Fällen $y_i \in \mathsf{QR}_n$ und $y_i \in \widetilde{\mathsf{QR}}_n$ unterscheiden und somit das Klartextbit rekonstruieren.

Öffentlicher Schlüssel: $n = p \cdot q$ (mit p, q prim und $p \equiv q \equiv 3 \bmod 4$)
Geheimer Schlüssel: $p,\ q$

Zur Verschlüsselung eines Klartextbits $x \in X = \{0, 1\}$ wählt der Sender A zufällig ein $r \in R = \mathbb{Z}_n^*$ und berechnet

$$E(r, x, n) = (-1)^x r^2 \bmod n.$$

Die Entschlüsselung ergibt sich dann zu

$$D(y, p) = \begin{cases} 0, & y^{(p-1)/2} \equiv_p 1, \\ 1, & \text{sonst.} \end{cases}$$

7.9 Der BBS-Generator

Der BBS-Pseudozufallsgenerator von Blum, Blum und Shub 1986 verwendet die Quadratfunktion

$$x^2 : \mathsf{QR}_n \mapsto \mathsf{QR}_n,$$

mit $n = p \cdot q$ für p, q prim und $p \equiv_4 q \equiv_4 3$. Seine Sicherheit lässt sich unter der Voraussetzung beweisen, daß ohne Kenntnis der Faktoren $p,\ q$ für fast alle $y \in \mathsf{QR}_n$ das niederwertigste Bit von $\sqrt{y}$ nur mit Wahrscheinlichkeit $1/2$ richtig geraten werden kann. Als Keim wird eine zufällig aus $\mathbb{Z}_n^*$ gewählte Zahl s verwendet. Daraus werden der Reihe nach Zahlen $z_i \in \mathsf{QR}_n$ durch Quadrieren berechnet, deren Paritäten die Bits der Ausgabefolge bilden.

Algorithmus $\mathsf{BBS}_{n,l}(s)$

```
1  z_0 := s^2 mod n
2  for i := 1 to l do
3     z_i := z_{i-1}^2 mod n
4     b_i := z_i mod 2
5  ouput(b_1, ... b_l)
```

Beispiel 155. *Wählen wir z. B. die Primzahlen $p = 11$, $q = 19$, also $n = 209$, und als Keim $s = 20$, so erhalten wir die Pseudo-Zufallsbitfolge (mit dem Startwert $z_0 = 20^2 \bmod 209 = 191$)* $\mathrm{BBS}_{209}(20) = 110011001\ldots$

i	0	1	2	3	4	5	6	7	8	...
z_i	191	115	58	20	191	115	58	20	191	...
b_i	1	1	0	0	1	1	0	0	1	...

Eine wichtige Eigenschaft des BBS-Generators ist, daß jedes beliebige Bit der Pseudo-Zufallsfolge aus einem beliebigen z_i effizient bestimmt werden kann, falls p und q bekannt sind.

Satz 156. *b_j kann bei Eingabe von p, q, z_i, i, j effizient bestimmt werden.*

Beweis.
$j \geq i$: Es gilt

$$\begin{aligned} b_j &= z_j \bmod 2 \\ &= (z_i^{2^{j-i}} \bmod n) \bmod 2 \\ &= (z_i^{2^{j-i} \bmod \varphi(n)} \bmod n) \bmod 2. \end{aligned}$$

$j < i$: Wegen $z_i = \sqrt{z_{i+1}}$ ist

$$z_j = \underbrace{\sqrt{\ldots\sqrt{x_i}}}_{(i-j)\text{-mal}}.$$

Sei $t = i - j$. Durch Berechnen der Wurzeln

$$\begin{aligned} u &= z_i^{(\frac{p+1}{4})^t \bmod p-1} \bmod p \\ v &= z_i^{(\frac{q+1}{4})^t \bmod q-1} \bmod q \end{aligned}$$

kann also z_j aus der linearen Darstellung $a{\cdot}p{+}b{\cdot}q = 1$ zu $z_j = (b{\cdot}q{\cdot}u{+}a{\cdot}p{\cdot}v) \bmod n$ bestimmt werden.

□

7.10 Sicherheit von Pseudozufallszahlen-Generatoren

Sei $l = l(k) \geq k + 1$ eine Funktion. Ein (k, l)-*Generator* ist eine Funktion f auf $\{0,1\}^*$, die Strings der Länge k auf Strings der Länge $l(k)$ abbildet und in Polynomialzeit berechenbar ist. Die Einschränkung von f auf die Menge $\{0,1\}^k$ bezeichnen wir mit f_k, d.h. $f_k : \{0,1\}^k \to \{0,1\}^{l(k)}$.

Seien X und Y Zufallsvariablen mit Wertebereich $\{0,1\}^*$, und sei $\varepsilon > 0$. Ein ε-*Unterscheider* zwischen X und Y ist ein probabilistischer Entscheidungsalgorithmus D mit

$$|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| \geq \varepsilon.$$

Hierbei ist $\Pr[D(X) = 1]$ die Wahrscheinlichkeit, daß D bei einer zufällig gemäß X gewählten Eingabe akzeptiert.

Sei $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ eine Funktion. Ein (k, l)-Generator f heißt ε-*unterscheidbar*, falls es einen in Polynomialzeit berechenbaren probabilistischen Algorithmus D gibt, so dass D für unendlich viele Werte von k ein $\varepsilon(l(k))$-Unterscheider zwischen den Zufallsvariablen U und $f_k(S)$ ist, wobei U und S auf $\{0,1\}^{l(k)}$ bzw. $\{0,1\}^k$ gleichverteilt sind.

f heißt *(kryptografisch) sicher*, falls f höchstens für vernachlässigbare Funktionen $\varepsilon : \mathbb{N} \to \mathbb{R}^+$ ε-unterscheidbar ist, d.h. für kein Polynom p ist f $1/p$-unterscheidbar.

Es ist bekannt, dass der BBS-Generator sicher ist, falls das QR-Problem nicht effizient lösbar ist.

7.11 Das Blum-Goldwasser-System

Im System von Blum und Goldwasser (1985) wird bei jeder Verschlüsselung eine Zufallszahl $s \in \mathbb{Z}_n^*$ als Keim gewählt und mit einem Pseudo-Zufallsgenerator auf einen Binärstring der gewünschten Länge aufgebläht, der dann als eigentlicher Schlüssel verwendet wird. Dem legalen Empfänger wird genügend Information über den verwendeten Keim mitgeteilt, so daß er die Pseudo-Zufallsfolge rekonstruieren und den Kryptotext entschlüsseln kann.

Öffentlicher Schlüssel: $n = pq$ (mit p, q prim und $p \equiv q \equiv 3 \bmod 4$)
Geheimer Schlüssel: $p,\ q$

Zur Verschlüsselung wählt der Sender A zufällig ein $r \in R = \mathbb{Z}_n^*$ und berechnet

$$E(r, x, n) = (z_{l+1}, x \oplus \mathrm{BBS}_{n,l}(r))$$

wobei $x \in \{0,1\}^l$, und $z_{l+1} = r^{2^{l+1}} \bmod n$ ist. Die Entschlüsselung ergibt sich dann zu

$$D(z, y, p, q) = y \oplus \mathrm{BBS}_{n,l}(r)$$

wobei $l = |y|$ und r die 2^{l+1}-te Wurzel von z modulo n ist.

Beispiel 157. *Gegeben seien* $m = 01100\,01111$, $p = 11$, $q = 19$, $n = 209$, $s = 20$ *und* $l = 10$.

Verschlüsselung von m*:*

$$\begin{aligned}
\mathrm{BBS}_{209,10}(20) &= 11001\,10011 \\
z_{10} &= 20^{2^{10+1}} \bmod 209 = 58 \\
E(20,\, x,\, 209) &= (58,\ x \oplus \mathrm{BBS}_{209,10}(20)) \\
&= (58,\ 01100\,01111 \oplus 11001\,10011) \\
&= (58,\ 10101\,11100) = y
\end{aligned}$$

Entschlüsselung von c*:*

$$\begin{aligned}
z_{10} &\equiv 3 \pmod{11} \\
u &= 3^{\left(\frac{11+1}{4}\right)^{10+1} \bmod (10+1)} \bmod 11 = 3^3 \bmod 11 = 9 \\
z_{10} &\equiv 1 \pmod{19} \\
v &= 1^{\left(\frac{19+1}{4}\right)^{10+1} \bmod (10+1)} \bmod 19 = 1^5 \bmod 19 = 1 \\
\mathrm{ggT}(11,\, 19) &= \lambda \cdot 11 + \mu \cdot 19 = \underbrace{7 \cdot 11}_{s=77} + \underbrace{(-4) \cdot 19}_{r=-76} \\
\Rightarrow s &= ru + sv \bmod n = -76 \cdot 9 + 77 \cdot 1 \bmod 209 = 20 \\
D((58,\ 10101\,11100),\ (11,\ 19)) &= 10101\,11100 \oplus \mathrm{BBS}_{209,10}(20) \\
&= 10101\,11100 \oplus 11001\,10011 \\
&= 01100\,01111 = x
\end{aligned}$$

7.12 Algorithmen zur Berechnung des diskreten Logarithmus

Sei $(G, *, 1)$ eine Gruppe und sei $\alpha \in G$. Weiter bezeichne $\langle\alpha\rangle = \{\alpha^i | i = 0 \cdots n-1\}$ die von α in G erzeugte Untergruppe, wobei $n = \mathrm{ord}_G(a) = \min\{e \geq 1 \mid \alpha^e = 1\}$ die Ordnung von α ist. Dann heißt die eindeutig bestimmte Zahl $e \in \{0, \ldots, n-1\}$ mit $\beta = \alpha^e$ der **diskrete Logarithmus** von β zur Basis α in G (kurz: $e = \log_{G,\alpha}(\beta)$).

Das diskrete Logarithmusproblem (DLP):

Gegeben: Gruppe G, ein Element $\alpha \in G$ und die Ordnung $n = \mathrm{ord}_G(a)$ von α sowie ein Element $\beta \in \langle\alpha\rangle$.

Gesucht: Der diskrete Logarithmus $e = \log_{G,\alpha}(\beta)$ von β zur Basis α in G.

Für viele Gruppen G ist die Funktion $e \mapsto \alpha^e$ effizient mittels wiederholtem Quadrieren und Multiplizieren berechenbar. In einigen Fällen ist jedoch kein effizienter Algorithmus zur Bestimmung der Umkehrfunktion, also von $\log_\alpha(\beta)$ bekannt, d.h. $e \mapsto \alpha^e$ ist Kandidat für eine Einwegfunktion.

Beispiel 158. *Sei $G = (\mathbb{Z}_p^*, *)$, p prim, und sei α ein Erzeuger von $\mathbb{Z}_p^*$. Dann ist $\langle\alpha\rangle = \mathbb{Z}_p^*$ und α hat die Ordnung $n = p - 1$. Ist p hinreichend groß und enthält $p - 1$ mindestens einen großen Primfaktor, so sind keine effiziente Algorithmen zur Berechnung von $\log_{p,\alpha}(\beta)$ bekannt.*

Wir betrachten zunächst eine Reihe von naiven Algorithmen für das DLP.

Berechnung von $\log_{G,\alpha}(\beta)$

```
1  γ := 1
2  for i := 0 to n − 1 do
3    if γ = β then output(i)
4    γ := αγ
```

Dieser Algorithmus läuft in Zeit $\mathcal{O}(n)$ (wobei wir annehmen, dass elementare Gruppenoperationen in konstanter Zeit ausführbar sind) und benötigt nur logarithmischen Speicherplatz. Falls wir im Vorfeld eine Tabelle mit den Logarithmen aller möglichen Werte für β erstellen, können wir danach für jedes β den diskreten Logarithmus durch eine Binärsuche in Zeit $O(\log n)$ bestimmen. Für die Precomputation fallen jedoch Zeit $\mathcal{O}(n)$ und Platz $\mathcal{O}(n \log n)$ an.

DLP-Berechnung mittels Precomputation

```
1  Precomputation: Sortiere die Paare (α^i, i), i = 0, ..., n − 1, nach der
     ersten Komponente in eine Tabelle T
2  Computation: Ermittle in T mittels Binärsuche den Eintrag (β, i)
     und gib i aus
```

Der folgende Algorithmus von Shanks berechnet ebenfalls im Vorfeld eine Tabelle von DLP-Werten, allerdings nur für Potenzen der Form α^{jm}, $j = 0, \ldots, m-1$, wobei $m = \lceil\sqrt{n}\rceil$ ist. Dadurch erhöht sich zwar die Laufzeit zur Bestimmung des diskreten Logarithmus für β von $O(\log n)$ auf $O(\sqrt{n})$, im Gegenzug geht jedoch der Speicherplatzverbrauch von $\mathcal{O}(n \log n)$ auf $\mathcal{O}(\sqrt{n} \log n)$ zurück.

Algorithmus `Shanks`(G, n, α, β)

```
1  Precomputation:
2    m := ⌈√n⌉
3    sortiere die Paare (α^{jm}, j), j = 0, ..., m − 1, nach der ersten
       Komponente in eine Tabelle T1
4  Computation:
5    sortiere die Paare (βα^{−i}, i) nach der ersten Komponente in eine
       Tabelle T2
```

```
6   ermittle durch parallele sequentielle Suche Paare (γ, j) in T1
     und (γ, i) in T2 mit derselben ersten Komponente
7   output(mj + i)
```

Der Pohlig-Hellman-Algorithmus

Die Ordnung der Potenzen eines Elements $\alpha \in G$ der Ordnung n lässt sich wie folgt berechnen:

$$\mathrm{ord}_G(\alpha^i) = n/\,\mathrm{ggT}(n,i).$$

Ist insbesondere q ein Teiler von n, so hat $\alpha^{n/q}$ die Ordnung q.

Sei $a = \log_{G,\alpha}(\beta)$ der diskrete Logarithmus von β zur Basis α in der Gruppe $G = \mathbb{Z}_p^*$, wobei wir annehmen, dass α ein Erzeuger von $\mathbb{Z}_p^*$ ist, also die Ordnung $n = p-1$ hat. (Der Pohlig-Hellman-Algorithmus kann nicht nur zur Berechnung des diskreten Logarithmus in der Gruppe $\mathbb{Z}_p^*$ benutzt werden, sondern für eine beliebige Gruppe G.) Sei $n = \prod_{i=1}^{k} p_i^{e_i}$ die Primfaktorzerlegung von $n = p-1$. Falls wir für $i = 1, \ldots, k$ die Werte $x_i = a \bmod p_i^{e_i}$ kennen, so lässt sich daraus a leicht mit dem Chinesischen Restsatz berechnen. Schreiben wir x_i als Zahl zur Basis p_i, so erhalten wir Ziffern $a_0, \ldots, a_{e_i-1}$ mit $x_i = \sum_{j=0}^{e_i-1} a_j p_i^j$. Weiter ex. eine Zahl s_i mit $a = x_i + s_i p_i^{e_i}$.

Um nun die Ziffern $a_0, \ldots, a_{e_i-1}$ zu berechnen, betrachten wir für $j = 0, \ldots, e_i - 1$ und $\beta_j = \beta\alpha^{-a_0 - a_1 p_i - a_2 p_i^2 \cdots - a_{j-1} p_i^{j-1}}$ die Gleichung

$$\beta_j^{n/p_i^{j+1}} = \alpha^{a_j n/p_i},$$

die sich leicht verifizieren lässt:

$$\begin{aligned}
\beta_j^{n/p_i^{j+1}} &= (\alpha^{a - a_0 - a_1 p_i - a_2 p_i^2 \cdots - a_{j-1} p_i^{j-1}})^{n/p_i^{j+1}} \\
&= (\alpha^{a_j p_i^j + a_{j+1} p_i^{j+1} + \cdots + a_{e_i-1} p_i^{e_i-1} + s_i p_i^{e_i}})^{n/p_i^{j+1}} \\
&= (\alpha^{a_j p_i^j + t p_i^{j+1}})^{n/p_i^{j+1}} \text{ für eine Zahl } t \\
&= \alpha^{a_j n/p_i} \alpha^{tn} \\
&= \alpha^{a_j n/p_i}
\end{aligned}$$

Der folgende Algorithmus berechnet sukzessive die Zahlen β_j und dazu die Ziffern $a_j = \log_{G,\alpha^{n/p_i}}(\beta^{n/p_i^{j+1}})$, die sich wegen $\mathrm{ord}_G(\alpha^{n/p_i}) = p_i$ in Zeit $\mathcal{O}(\sqrt{p_i})$ (etwa mit dem Algorithmus von Shanks) ermitteln lassen. Insgesamt erhalten wir somit eine Laufzeit von $\mathcal{O}(e_i\sqrt{p_i})$ zur Bestimmung von x_i.

Pohlig-Hellman$(G, n = \mathrm{ord}(\alpha), \alpha, \beta, p, e)$

```
1  j := 0
2  β_j := β
3  while j ≤ e − 1 do
4      δ := β_j^{n/p^{j+1}}
5      a_j := log_{α^{n/p}} δ // z.B. mit Shanks
6      β_{j+1} := β_j α^{−a_j p^j}
7      j := j + 1
8  return Σ_{i=1}^{e−1} a_i p^i
```

Algorithmus `Pohlig-Hellman-DLP`$(G, n, \alpha, \beta, p_i, e_i)$

for $j := 0$ **to** $e_i - 1$ **do**
2 $a_j := \log_{G,\alpha^{n/p_i}}(\beta^{n/p_i^{j+1}})$
3 $\beta := \beta\alpha^{-a_j p_i^j}$
output$(a_0 \cdots a_{e_i-1})$

7.13 Die Rho-Algorithmen von Pollard

Von Pollard wurde eine heuristische Strategie entwickelt, die sich sowohl zur Lösung des DLP als auch des Faktorisierungsproblems eignet. Die Idee dabei ist, mit wenig Speicherplatz eine Kollision $a_i = a_j$ mit $i \neq j$ für eine Folge (a_n) der Form $a_{n+1} = f(a_n)$ zu finden. Zahlenfolgen dieser Bauart haben die Eigenschaft, dass $a_i = a_j$ die Gleichheit $a_{i+k} = a_{j+k}$ für alle $k \geq i$ impliziert.

Der Rho-Faktorisierungsalgorithmus

Sei n eine Zahl mit mindestens 2 verschiedenen Primteilern $p < q$ (falls n nur einen Primteiler hat, also eine Primzahlpotenz ist, lässt sich n leicht durch Berechnung der k-ten Wurzeln für $k = 2, \ldots, \log_2(n)$ faktorisieren).

Angenommen, wir wählen zufällig eine Menge $X \subseteq \mathbb{Z}_n$ der Größe $\sqrt{p}$, so enthält X mit großer Wahrscheinlichkeit 2 Elemente $x \neq x'$ mit $x \equiv_p x'$, die auf den nichttrivialen Faktor $d = \mathrm{ggT}(x - x', n)$ von n führen.

Wählen wir nun anstelle von X eine pseudozufällige Menge der Form $X = \{x_1, x_2 = f(x_1), \ldots, x_j = f(x_{j-1})\}$, wobei x_1 ein zufällig gewählter Startwert ist, so tritt bei geeigneter Wahl von $f : \mathbb{Z}_n \to \mathbb{Z}_n$ für $j \approx \sqrt{p}$ mit großer Wahrscheinlichkeit eine Kollision auf. Eine gute Wahl für f ist beispielsweise $f(x) = x^2 \pm 1 \bmod n$.

Werden zur Berechnung von f nur die Ringoperationen von $\mathbb{Z}_n$ verwendet, so impliziert $x_i \equiv_p x_j$ die Kongruenz $f(x_i) \equiv_p f(x_j)$, was wiederum für $l = j - i$ die Kongruenz $x_k \equiv_p x_{k+dl}$ für alle $k \geq i$ und $d \geq 1$ impliziert. Insbesondere folgt also $x_k \equiv_p x_{2k}$ für alle $k \geq i$ mit $k \equiv_l 0$. Daher können wir in X ein Kollisionspaar (x_i, x_j) mit $x_i \equiv_p x_j$ wie folgt bestimmen (ohne p zu kennen).

Algorithmus `Pollard-Rho-Factorize`(n)

1 `wähle zufällig` $x \in \mathbb{Z}_n$
2 $y := x^2 + 1 \bmod n$
3 **while** $\mathrm{ggT}(x - y, n) = 1$ **do**
4 $\quad x := f(x)$
5 $\quad y := f(f(y))$
6 **if** $d = \mathrm{ggT}(x - y, n) < n$ **then output**(d)
7 **else output**$(?)$

Der Rho-DLP-Algorithmus

Dieser Algorithmus berechnet eine pseudozufällige Folge von Paaren $(c_i, d_i) \in \mathbb{Z}_n \times \mathbb{Z}_n$. Ziel ist es, zwei verschiedene Paare (c_i, d_i) und (c_j, d_j) mit $\alpha^{c_i}\beta^{d_i} = \alpha^{c_j}\beta^{d_j}$ zu finden. Im

Fall $\mathrm{ggT}(d_j - d_i, n) = 1$ lässt sich hieraus wegen

$$\alpha^{c_i + a d_i} = \alpha^{c_i}\beta^{d_i} = \alpha^{c_j}\beta^{d_j} = \alpha^{c_j + a d_j}$$

der diskrete Logarithmus $\log_{G,\alpha}(\beta) = (c_i - c_j)(d_j - d_i)^{-1} \bmod n$ leicht bestimmen. Andernfalls erhalten wir $g = \mathrm{ggT}(d_j - d_i, n)$ Kandidaten $a_1, \ldots, a_g$, unter denen der richtige ebenfalls leicht zu ermitteln ist, sofern g nicht zu groß ist. Zur Bildung der Pseudozufallsfolge kann bspw. die Funktion f in folgendem Algorithmus benutzt werden. Aus Effizienzgründen berechnet sie auch gleich die Werte $x_i = \alpha^{c_i}\beta^{d_i}$. Die Mengen S_1, S_2, S_3 bilden eine Partition von G in drei etwa gleich große Mengen, wobei das neutrale Element 1 von G nicht in S_2 enthalten sein sollte.

Algorithmus `Pollard-Rho-DLP`(G, n, α, β)

```
1  function f(x, c, d)
2    case
3      x ∈ S_1: return(βx, c, d + 1 mod n)
4      x ∈ S_2: return(x^2, 2c mod n, 2d mod n)
5      x ∈ S_3: return(αx, c + 1 mod n, d)
6
7  wähle zufällig c, d ∈ Z_n
8  x := α^c β^d
9  (y, e, f) := f(x, c, d)
10 while x ≠ y do
11   (x, c, d) := f(x, c, d)
12   (y, e, f) := f(f(y, e, f))
13 g := ggT(f − d, n)
14 bestimme alle Lösungen a_1, ..., a_g von (f − d)a ≡_n (c − e)
15 output a_i mit α^{a_i} = β
```

Ähnlich wie beim Rho-Faktorisierungsalgorithmus lässt sich argumentieren, dass die while-Schleife nach ca. $\sqrt{n}$ Iterationen abbricht.