Vorlesungsskript

Kryptologie 1

Wintersemester 07/08

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

14. Dezember 2007

Inhaltsverzeichnis

1 Klassische Verfahren **3**

1.1 Einführung 3

1.2 Kryptosysteme 4

1.3 Die affine Chiffre 6

1.4 Die Hill-Chiffre 16

1.5 Die Vigenère-Chiffre und andere Stromsysteme 18

1.6 Der One-Time-Pad 20

1.7 Klassifikation von Kryptosystemen 21

1.8 Realisierung von Blocktranspositionen und einfachen Substitutionen 31

2 Kryptoanalyse der klassischen Verfahren **34**

2.1 Klassifikation von Angriffen gegen Kryptosysteme 34

2.2 Kryptoanalyse von einfachen Substitutionschiffren 36

2.3 Kryptoanalyse von Blocktranspositionen 39

2.4 Kryptoanalyse von polygraphischen Chiffren 42

2.5 Kryptoanalyse von polyalphabetischen Chiffren 43

3 Sicherheit von Kryptosystemen **52**

3.1 Informationstheoretische Sicherheit 52

3.2 Komplexitätstheoretische Sicherheit 63

4 Moderne symmetrische Kryptosysteme & ihre Analyse **67**

4.1 Produktchiffren 67

4.2 Substitutions-Permutations-Netzwerke 69
4.3 Lineare Approximationen 73
4.4 Lineare Kryptoanalyse eines SPN 75
4.5 Differentielle Kryptoanalyse von SPNs 79

1 Klassische Verfahren

1.1 Einführung

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

Physikalische Maßnahmen: Tresor etc.

Organisatorische Maßnahmen: einsamer Waldspaziergang etc.

Steganographische Maßnahmen: unsichtbare Tinte etc.

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

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

- von Informationsdienstleistungen

In das Umfeld der Kryptographie fallen auch die folgenden Begriffe.

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

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

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

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

1.2 Kryptosysteme

Es ist wichtig, Kryptosysteme von Codesystemen zu unterscheiden.

Codesysteme

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

Beispiel 1.1 (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 1.2 (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 $A^* = \bigcup_{n \geq 0} A^n$.*

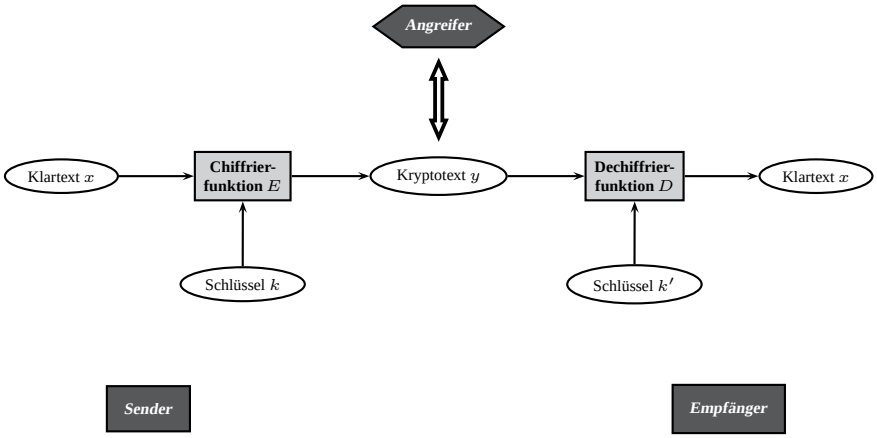
Beispiel 1.3 Das ***lateinische Alphabet*** A_{lat} enthält die 26 Buchstaben `A, ..., Z`. Bei der Abfassung von Klartexten wurde meist auf den Gebrauch von Interpunktions- und Leerzeichen sowie auf Groß- und Kleinschreibung verzichtet ($\leadsto$ Verringerung der Redundanz im Klartext).

Definition 1.4 (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 1.5 *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 1.6 (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*

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

Tabelle 1.1: Werte der additiven Chiffrierfunktion ROT13 (Schlüssel $k = 13$).

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

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

Die auf $\mathbb{Z}$ definierten Operationen

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

und

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

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

Durch Identifikation der Buchstaben a_i mit ihren Indizes können wir die auf $\mathbb{Z}_m$ definierten Rechenoperationen auf Buchstaben übertragen.

Definition 1.7 (Buchstabenrechnung) *Sei $A = \{a_0, \ldots, a_{m-1}\}$ ein Alphabet. Für Indizes $i, j \in \{0, \ldots, m-1\}$ und eine ganze Zahl $z \in \mathbb{Z}$ ist*

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

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

Definition 1.8 (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*

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

Im Fall des lateinischen Alphabets führt der Schlüssel $k = 13$ auf eine interessante Chiffrierfunktion, die in UNIX-Umgebungen auch unter der Bezeichnung ROT13 bekannt ist (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

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

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 $g(\mathsf{A}) = g(\mathsf{N}) = \mathsf{A}$ ab. Um die vom Schlüsselwert k zu erfüllende Bedingung angeben zu können, führen wir folgende Begriffe ein.

Definition 1.9 (ggT, kgV, teilerfremd) *Seien $a, b \in \mathbb{Z}$. Für $(a, b) \neq (0, 0)$ ist*

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

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

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

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

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

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

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

mit

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

Hierzu sind n Divisionsschritte erforderlich. Wegen

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

folgt $\text{ggT}(a, b) = \text{ggT}(r_n, r_{n+1}) = r_n$.

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

Beispiel 1.10

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.

Algorithmus 1.11 EUKLID$_{it}(a, b)$

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

Algorithmus 1.12 EUKLID$_{rek}(a, b)$

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

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

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

Durch Induktion über $i = n, n-1, \ldots, 0$ folgt $r_i \geq f_{n+1-i}$; also $a \geq f_{n+1}$. Wegen $f_n \geq \Re^n$ (wobei $\Re = \frac{1+\sqrt{5}}{2}$; Beweis durch Induktion) ist dann $a \geq \Re^n$, d. h. $n \leq \log_{\Re} a$.

Theorem 1.13 *Der Euklidsche Algorithmus führt zur Berechnung von* $\mathrm{ggT}(a, b)$ *(unter der Annahme* $a > b > 0$*) höchstens* $\lfloor \log_R a \rfloor + 1$ *Divisionsschritte durch. Dies führt auf eine Zeitkomplexität von* $O(n^3)$*, wobei* n *die Länge der Eingabe in Binärdarstellung bezeichnet und wir* $O(n^2)$ *Rechenschritte für eine einzelne Ganzzahldivision ansetzen.*

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 $i = 2, \ldots, n$ gilt. Insbesondere gilt also

$$ap_n + bq_n = r_n = \mathrm{ggT}(a, b).$$

Korollar 1.14 (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 1.15 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	$\cdot$ **21**	$+$	0	**3**	**-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 1.16 $\mathrm{ggT}(a,b) = \min\{\lambda a + \mu b \geq 1 \mid \lambda, \mu \in \mathbb{Z}\}$.

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

Korollar 1.17 *Der größte gemeinsame Teiler von a und b wird von allen gemeinsamen Teilern von a und b geteilt,*

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

Beweis Sei $g = \mathrm{ggT}(a,b)$. Dann existieren Zahlen $\mu, \lambda \in \mathbb{Z}$ mit $\mu a + \lambda b = g$. Da x nach Voraussetzung sowohl a als auch b teilt, teilt x auch die Zahlen μa und λb und somit auch deren Summe $\mu a + \lambda b = g$. □

Korollar 1.18 (Lemma von Euklid) *Teilt a das Produkt bc und sind a, b teilerfremd, so ist a auch Teiler von c,*

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

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

Korollar 1.19 *Wenn sowohl a als auch b zu einer Zahl $m \in \mathbb{Z}$ teilerfremd sind, so ist auch das Produkt ab teilerfremd zu m,*

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

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

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

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 1.20 *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 1.21 *Im Fall* $\mathrm{ggT}(b, m) = 1$ *hat die Kongruenz* $bx \equiv_m 1$ *genau eine Lösung, die das* ***multiplikative Inverse*** *von* b *modulo* m *genannt und mit* $b^{-1} \bmod m$ *(oder einfach mit* b^{-1}*) bezeichnet wird. Die invertierbaren Elemente von* $\mathbb{Z}_m$ *werden in der Menge*

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

zusammengefasst.

Korollar 1.19 zeigt, dass $\mathbb{Z}_m^*$ unter der Operation $\odot_m$ abgeschlossen ist, und mit Korollar 1.21 folgt, dass $(\mathbb{Z}_m^*, \odot_m)$ eine multiplikative Gruppe bildet.

Das multiplikative Inverse von b modulo m ergibt sich aus der linearen Darstellung $\lambda b + \mu m = \mathrm{ggT}(b, m) = 1$ 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 1.22 (affine Chiffre) *Bei der* ***affinen Chiffre*** *ist* $A = B = M = C$ *ein beliebiges Alphabet mit* $m := \|A\| > 1$ *und* $K = \mathbb{Z}_m^* \times \mathbb{Z}_m$. *Für* $k = (b, c) \in K$, $x \in M$ *und* $y \in C$ *gilt*

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

In diesem Fall liefert die Schlüsselkomponente $b = -1$ für jeden Wert von c eine involutorische Chiffrierfunktion $x \mapsto E(b, c; x) = c - x$ (**verschobenes komplementäres Alphabet**). Wählen wir für c ebenfalls den Wert -1, so ergibt sich die Chiffrierfunktion $x \mapsto -x - 1$, 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
$-x-1$	*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 1.23 (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

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

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 $m \in \mathbb{N}$ 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 1.24

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

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

Theorem 1.25 *Die Eulersche φ-Funktion ist multiplikativ, d. h. für teilerfremde Zahlen m und l gilt $\varphi(ml) = \varphi(m)\varphi(l)$.*

Korollar 1.26 *Sei $m = \prod_{i=1}^{k} p_i^{e_i}$ die Primfaktorzerlegung von m. Dann gilt*

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

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.

Theorem 1.27 (Chinesischer Restsatz) *Falls $m_1, \ldots, m_k$ paarweise teilerfremd sind, dann hat das System*

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

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

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

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

Dann gilt

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

und

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

für $j \neq i$. Folglich erfüllt $x = \sum_{j=1}^{k} \mu_j n_j b_j$ 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 $x = x_1 \ldots x_l \in A^l$ in den Kryptotextblock $y_1 \ldots y_l \in A^l$ transformiert, wobei

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

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

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

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

Definition 1.28 (Determinante) *Sei $A = (a_{ij})$ eine $l\times l$-Matrix. Für $1 \le i, j \le l$ sei A_{ij} die durch Streichen der i-ten Zeile und j-ten Spalte aus K hervorgehende Matrix. Die **Determinante** von A ist dann $\det(A) = a_{11}$, falls $l = 1$, und*

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

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

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

Satz 1.29 *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 1.30 (Hill-Chiffre) *Sei $A = \{a_0, \ldots, a_{m-1}\}$ 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*

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

Beispiel 1.31 (Hill-Chiffre) Benutzen wir zur Chiffrierung von Klartextblöcken der Länge $l = 4$ über dem lateinischen Alphabet A_{lat} die Schlüsselmatrix

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

so erhalten wir beispielsweise für den Klartext HILL wegen

$$(\mathsf{H\,I\,L\,L}) \begin{pmatrix} 11 & 13 & 8 & 21 \\ 24 & 17 & 3 & 25 \\ 18 & 12 & 23 & 17 \\ 6 & 15 & 2 & 15 \end{pmatrix} = (\mathsf{N\,E\,R\,X}) \;\text{ bzw. }\; \begin{array}{r} 11\,\mathsf{H} + 24\,\mathsf{I} + 18\,\mathsf{L} + 6\,\mathsf{L} = \mathsf{N} \\ 13\,\mathsf{H} + 17\,\mathsf{I} + 12\,\mathsf{L} + 15\,\mathsf{L} = \mathsf{E} \\ 8\,\mathsf{H} + 3\,\mathsf{I} + 23\,\mathsf{L} + 2\,\mathsf{L} = \mathsf{R} \\ 21\,\mathsf{H} + 25\,\mathsf{I} + 17\,\mathsf{L} + 15\,\mathsf{L} = \mathsf{X} \end{array}$$

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

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

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

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

Beispiel 1.33 (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 1.33 ((Vigenère-Chiffre, Fortsetzung))

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 1.34 (Beaufort-Chiffre) Verschlüsseln wir den Klartext BEAUFORT beispielsweise unter dem Schlüsselwort $k = \mathit{WIE}$, 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 1.35 (*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:		Kryptotext-Schlüsselstrom:	
`VIGENERE`	(Klartext)	`VIGENERE`	(Klartext)
+ `WIEVIGEN`	(Schlüsselstrom)	+ `WIERQKVD`	(Schlüsselstrom)
`RQKZVKVR`	(Kryptotext)	`RQKVDOMH`	(Kryptotext)

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

1.6 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**

†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.

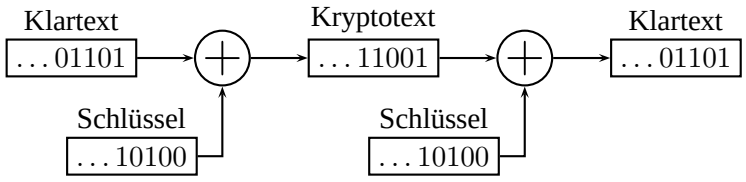
‡Diese Art der Schlüsselerzeugung schlug der amerikanische Major Joseph O. Mauborgne im Jahr 1918 vor, nachdem ihm ein von Gilbert S. Vernam für den Fernschreibverkehr entwickeltes Chiffriersystem vorgestellt wurde.

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 1.36 (*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,

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

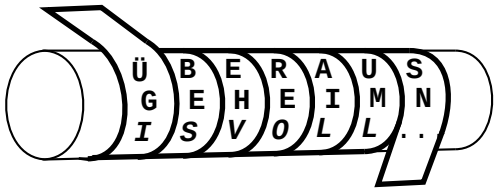
Der Klartext wird also wie bei einer additiven Chiffre verschlüsselt, nur dass der Schlüssel nach einmaligem Gebrauch gewechselt wird. Dies entspricht dem Gebrauch einer Vigenère-Chiffre, falls als Schlüssel ein zufällig gewähltes Wort von der Länge des Klartextes benutzt wird. Wie diese ist der *One-time-pad*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



1.7 Klassifikation von Kryptosystemen

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

Beispiel 1.37 (Skytale-Chiffre) Die älteste bekannte Verschlüsselungstechnik stammt aus der Antike und wurde im 5. Jahrhundert v. Chr. von den Spartanern entwickelt: Der Sender wickelt einen Papierstreifen spiralförmig um einen Holzstab (die sogenannte **Skytale**) und beschreibt ihn in Längsrichtung mit der Geheimbotschaft.



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

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

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

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

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

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

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

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

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

Monographische Substitutionen ersetzen Einzelbuchstaben.

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

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

Beispiel 1.38 Bei der **Porta-Chiffre** werden 400 (!) unterschiedliche von Porta für diesen Zweck entworfene Kryptotextzeichen verwendet. Diese sind in einer 20×20-Matrix $M = (y_{ij})$ angeordnet, deren Zeilen und Spalten mit den 20 Klartextbuchstaben $\mathtt{A}, \ldots, \mathtt{I}, \mathtt{L}, \ldots, \mathtt{T}, \mathtt{V}, \mathtt{Z}$ indiziert sind. Zur Ersetzung des Buchstabenpaars $a_i a_j$ wird das in Zeile i und Spalte j befindliche Kryptotextzeichen

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

benutzt.

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

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

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

POLYBIOS $\rightsquigarrow$ *3024214301132433*

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

[§]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.

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

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

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

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

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

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

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

Definition 1.40 (Stromchiffre) *Sei A ein beliebiges Alphabet und sei $M = C = A^l$ für eine natürliche Zahl $l \geq 1$. Weiterhin seien K und $\hat{K}$ Schlüsselräume. Eine **Stromchiffre** wird durch eine Verschlüsselungsfunktion $E : \hat{K} \times M \to C$ und einen Schlüsselstromgenerator $g : K \times A^* \to \hat{K}$ beschrieben. Der Generator g erzeugt aus einem externen Schlüssel $k \in K$ für einen Klartext $x = x_0 \dots x_{n-1}$, $x_i \in M$, 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 1.41 Bei der **Spionage-Chiffre** wird in die erste Zeile einer 3×10-Matrix ein Schlüsselwort w geschrieben, welches keinen Buchstaben mehrfach enthält und eine Länge von 6 bis 8 Zeichen hat (also zum Beispiel *SPIONAGE*). Danach werden die anderen beiden Zeilen der Matrix mit den restlichen Klartextbuchstaben (etwa in alphabetischer Reihenfolge) gefüllt.

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

GESPREIZT
$\rightsquigarrow$ *2741547957 51*

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

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

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

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

Beispiel 1.43 (homophone Substitution) Sei $A = \{\mathtt{A}, \ldots, \mathtt{Z}\}$, $B = \{\mathtt{0}, \ldots, \mathtt{9}\}$ und $C = \{\mathtt{00}, \ldots, \mathtt{99}\}$.

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

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

Genau wie bei Polybios wird eine 5×5-Matrix M als Schlüssel benutzt. Die Zeilen und Spalten von M sind jedoch nicht nur mit jeweils einer, sondern mit

zwei Ziffern versehen, so dass jeder Klartextbuchstabe x über vier verschiedene Koordinatenpaare ansprechbar ist. Der Kryptotextraum wird durch M also in 25 Mengen $H(a)$, $a \in A$, mit je 4 Homophonen partitioniert.

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

Beispiel 1.44 (homophone Substitution, verbesserte Version) Ist $p(a)$ die Wahrscheinlichkeit, mit der ein Zeichen $a \in A$ in der Klartextsprache auftritt, so sollte $\|H(a)\| \approx 100 \cdot p(a)$ sein.

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

Da der Buchstabe A im Deutschen beispielsweise mit einer Wahrscheinlichkeit von $p(\mathsf{A}) = 0{,}0647$ 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.

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

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

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

Transpositionen

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

Beispiel 1.45 Bei Ausführung einer ***Zick-Zack-Transposition*** wird der Klartext in eine Zick-Zack-Linie geschrieben und horizontal wieder ausgelesen. Die Höhe der Zick-Zack-Linie kann als Schlüssel vereinbart werden.

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

ZICKZACKLINIE $\rightsquigarrow$ *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 1.46 (Blocktranspositionschiffre) *Sei $A = B$ ein beliebiges Alphabet und für eine natürliche Zahl $l \geq 2$ sei $M = C = A^l$. Bei einer **Blocktranspositionschiffre** wird durch jeden Schlüssel $k \in K$ eine Permutation π beschrieben, so dass für alle Zeichenfolgen $x_1 \cdots x_l \in M$ und $y_1 \cdots y_l \in C$*

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

und

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

gilt.

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

Beispiel 1.47 Eine Skytale, die mit 4 Zeilen der Länge 6 beschrieben wird, realisiert beispielsweise folgende Blocktransposition:

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

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

Beispiel 1.48

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

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

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

Beispiel 1.49 Obiges π hat beispielsweise die Zyklendarstellung

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

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

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

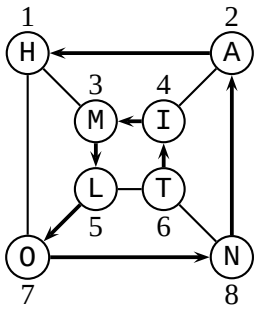
wenn wir jeden Zyklus mit seinem kleinsten Element beginnen lassen und die Zyklen nach der Größe dieser Elemente anordnen.

Beispiel 1.50 Bei der ***Matrix-Transposition*** wird der Klartext zeilenweise in eine $k \times m$-Matrix eingelesen und der Kryptotext spaltenweise gemäß einer Spaltenpermutation π, die als Schlüssel dient, wieder ausgelesen. Für $\pi = (1\ 4\ 3)\ (2\ 6)$ wird also zuerst Spalte $\pi(1) = 4$, dann Spalte $\pi(2) = 6$ und danach Spalte $\pi(3) = 1$ usw. und zuletzt Spalte $\pi(6) = 2$ ausgelesen.

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

DIESER KLARTEXT IST NICHT SEHR LANG
$\rightsquigarrow$ *SRSTA RENEG DKXIH EAIHL ETTSN ILTCR*

Beispiel 1.51 Bei der ***Weg-Transposition*** wird als Schlüssel eine Hamiltonlinie in einem Graphen mit einer vorgegebenen Knotennummerierung benutzt. (Eine Hamiltonlinie ist eine Anordnung aller Knoten des Graphen, in der je zwei aufeinanderfolgende Knoten durch eine Kante verbunden sein müssen.) Der Klartext wird gemäß der Knotennumerierung in den Graphen eingelesen und der Kryptotext entlang der Hamiltonlinie wieder ausgelesen.



HAMILTON $\rightsquigarrow$ *TIMLONAH*

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

Sehr beliebt ist auch die Methode, 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 $\rightsquigarrow$
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.

1.8 Realisierung von Blocktranspositionen und einfachen Substitutionen

Abschließend möchten wir eine einfache elektronische Realisierungsmöglichkeit von Blocktranspositionen erwähnen, die auf binär kodierten Klartexten operieren

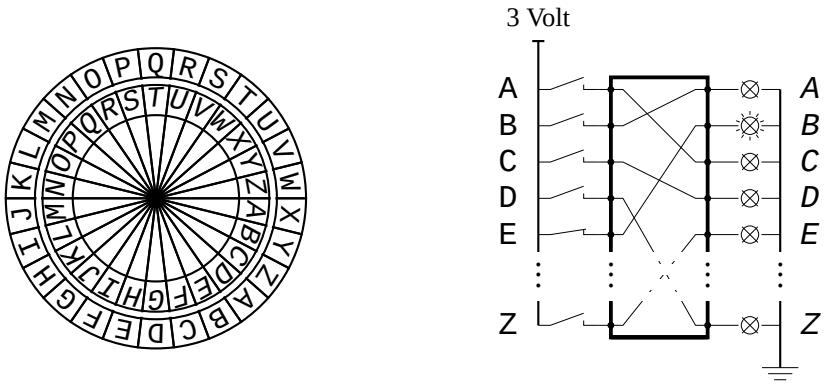
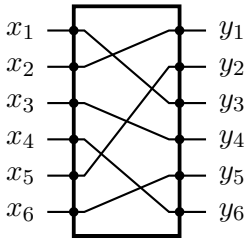


Abbildung 1.1: Realisierung von einfachen Substitutionen mit einer Drehscheibe und mit Hilfe von Steckverbindungen.

(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.

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

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

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

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

weiterer Kryptotexte oder die Bestimmung der benutzten Schlüssel wesentlich erleichtert.

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

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

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

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

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

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 2.1 (vollständige Schlüsselsuche) Es sei bekannt, dass das Kryptotextstück $y = \mathsf{SAXP}$ 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 $x = \texttt{GOLD}$ (Schlüsselwert $k = \textit{M}$) und $x = \texttt{WEBT}$ ($k = \textit{W}$) 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.

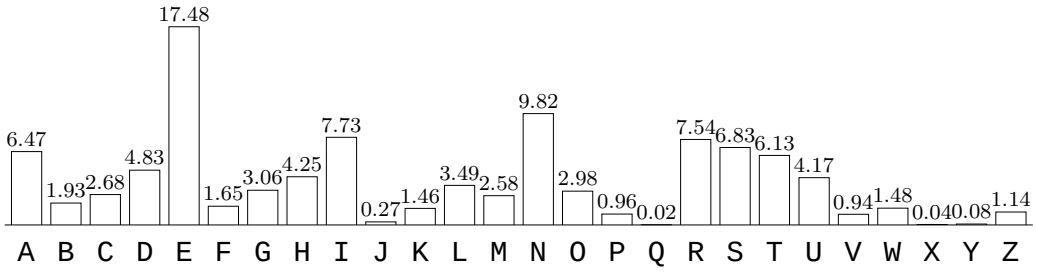


Abbildung 2.1: Häufigkeitsverteilung der Einzelbuchstaben im Deutschen (in %).

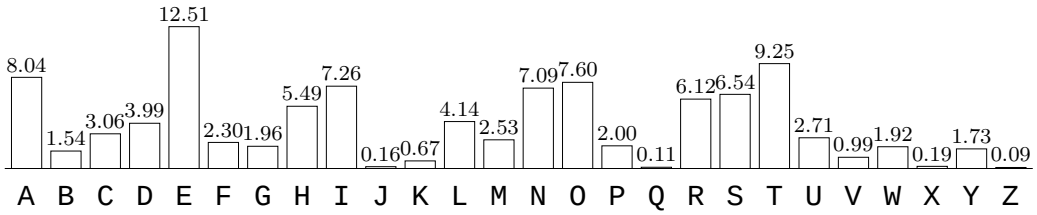


Abbildung 2.2: Häufigkeitsverteilung der Buchstaben im Englischen (in %).

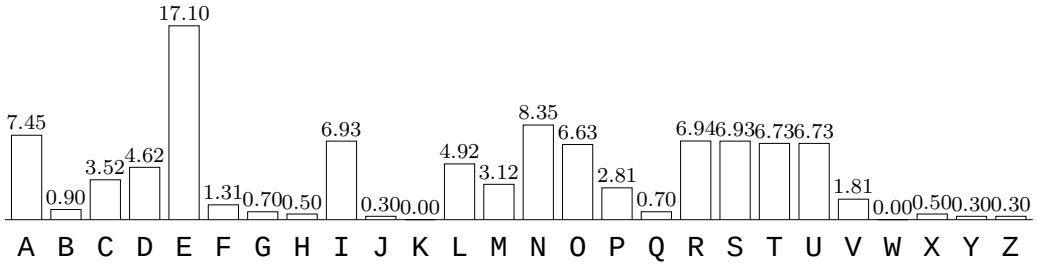


Abbildung 2.3: Häufigkeitsverteilung der Buchstaben im Französischen (in %).

Die Abbildungen 2.1, 2.2 und 2.3, zeigen typische Verteilungen von Einzelbuchstaben in der deutschen, englischen und französischen Sprache (ohne Berücksichtigung von Interpunktions- und Leerzeichen). Ein typischer deutscher Text besteht demnach zu 62% aus den sieben häufigsten Zeichen `E`, `N`, `I`, `R`, `S`, `A`, `T` (das sind nicht einmal 27% der Klartextzeichen).

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

Beispiel 2.2 (Analyse einer affinen Chiffre mittels Buchstabenhäufigkeiten) Es sei bekannt, dass sich hinter dem Kryptotext

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

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

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

so liegt die Vermutung nahe, dass das am häufigsten vorkommende Chiffrezeichen *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 (x, y) eine gute Übereinstimmung zwischen der Häufigkeit $H(y)$, mit der $y = E(k, x)$ im Kryptotext vorkommt, und der erwarteten Häufigkeit $E(x)$ auf, mit der x in einem typischen deutschen Text der Länge 100 vorkommt (die Tabelle zeigt die Werte von $E(x)$ gerundet):

y	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
$H(y)$	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
$E(x)$	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
x	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.

Beispiel 2.3 (Häufigkeitsanalyse von Bigrammen) Für den mit einer Blocktransposition (mit vermuteter Blocklänge 5) erzeugten Kryptotext

```
IHEHR BWEAN RNEII NRKEU ELNZK RXTAE VLOTR ENGIE
```

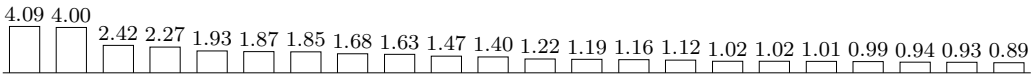


Abbildung 2.4: Die häufigsten Bigramme im Deutschen (Angaben in %).

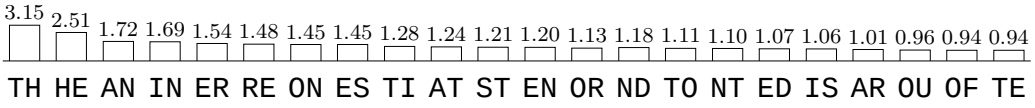


Abbildung 2.5: Die häufigsten Bigramme im Englischen (in %; nach O.P. Meaker, 1939).

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

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.

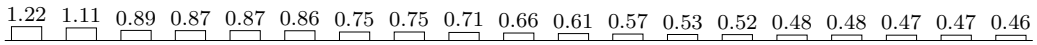


Abbildung 2.6: Die häufigsten Trigramme im Deutschen (in %).

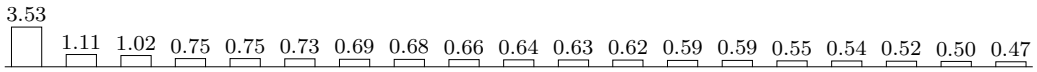


Abbildung 2.7: Die häufigsten Trigramme im Englischen (in %).

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

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

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

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

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

2.4 Kryptoanalyse von polygraphischen Chiffren

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

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

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

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 $\mathrm{ggT}(\det X, m) = 1$ für eine aus l Blöcken x_i, $i \in I$, als Zeilen gebildete Matrix X, so lässt sich die Schlüsselmatrix k zu $k = YX^{-1}$ bestimmen (Y ist die aus den Blöcken y_i, $i \in I$, gebildete Matrix).

2.5 Kryptoanalyse von polyalphabetischen Chiffren

Die Vigenère-Chiffre galt bis ins 19. Jahrhundert als sicher. Da der Schlüsselstrom bei der Vigenère-Chiffre periodisch ist, 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 2.4 (Kasiski-Test)

	DERERSTEUNDLETZTEVERS...	(Klartext x)
+	KASKASKASKASKASKASKAS...	(Schlüsselstrom $\hat{k}$)
	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 2.5 (Koinzidenzindex) *Der **Koinzidenzindex** (engl. **i**ndex of **c**oincidence) eines Textes y der Länge n über dem Alphabet $\mathcal{B}$ ist definiert als*

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

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

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

Um die Periode d einer Vigenère-Chiffre zu bestimmen, schreibt man den Kryptotext y für $d = 1, 2, 3, \ldots$ in eine Matrix mit d Spalten und berechnet für jede Spalte y_i den Koinzidenzindex $IC(y_i)$. 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 2.6 (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 2.7 (Rauheitsgrad; Measure of Roughness) *Der **Rauheitsgrad** MR_L einer Sprache L mit Einzelbuchstabenverteilung p ist*

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

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

Beispiel 2.8 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 2.9 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 polygraphischen 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 2.10 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
$\alpha_1(k)$	36	31	31	45	38	26	42	**73**	44	26	36	47	30
$\alpha_2(k)$	44	41	40	51	41	31	37	43	34	28	36	26	28
$\alpha_3(k)$	47	41	48	37	49	40	35	30	48	32	25	42	31
$\alpha_4(k)$	38	40	27	41	**65**	47	28	34	39	33	35	36	30

k	13	14	15	16	17	18	19	20	21	22	23	24	25
$\alpha_1(k)$	32	36	29	28	39	48	42	42	39	42	42	35	31
$\alpha_2(k)$	43	**68**	45	35	27	42	43	40	35	30	24	31	45
$\alpha_3(k)$	26	43	**76**	37	31	39	45	35	34	37	26	30	25
$\alpha_4(k)$	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

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

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

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

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

Ist nun y ein Kryptotext, der mit einem Schlüsselwort bekannter Länge d erzeugt wurde, und sind y_i, $i = 1, \ldots, d$ die zugehörigen Spalten, so gibt der gegenseitige Koinzidenzindex der Spalten y_i und $y_j + \delta$ (für $1 \leq i < j \leq d$) die Wahrscheinlichkeit an, dass man bei zufälliger Wahl einer Position in y_i und in $y_j + \delta$ denselben Buchstaben vorfindet, wobei δ eine Verschiebung von Spalte y_j relativ zur Spalte y_i ist (mit $0 \leq \delta \leq 25$). Mit großer Wahrscheinlichkeit nimmt also $IC(y_i + \delta, y_j)$ 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 2.12 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
$IC(y_1 + \delta, y_2)$	40	31	25	38	25	21	46	**74**	50	33	31	44	43
$IC(y_1 + \delta, y_3)$	26	47	25	21	47	32	18	49	**91**	42	27	51	45
$IC(y_1 + \delta, y_4)$	38	40	29	31	35	24	32	58	42	32	44	50	43
$IC(y_2 + \delta, y_3)$	50	**85**	49	21	28	35	24	34	46	25	24	27	59
$IC(y_2 + \delta, y_4)$	46	53	40	37	51	42	29	23	24	32	40	55	38
$IC(y_3 + \delta, y_4)$	49	36	38	60	36	25	34	19	29	42	41	33	54

δ	13	14	15	16	17	18	19	20	21	22	23	24	25
$IC(y_1 + \delta, y_2)$	34	31	28	24	31	44	45	37	48	64	44	25	31
$IC(y_1 + \delta, y_3)$	31	29	32	23	29	27	39	45	46	39	58	44	24
$IC(y_1 + \delta, y_4)$	39	31	20	34	36	30	40	45	24	42	**78**	47	22
$IC(y_2 + \delta, y_3)$	50	50	53	51	24	22	26	43	36	35	32	24	34
$IC(y_2 + \delta, y_4)$	31	32	45	**67**	49	25	27	29	29	34	37	38	35
$IC(y_3 + \delta, y_4)$	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 2.13 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 als wahrscheinliche Paare für die Einzelbuchstaben von w:

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

und damit (w_1, w_2) zu

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

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

Autokey Chiffren

Kryptotextschlüsselstrom. Diese Systeme bieten eigentlich keinen großen kryptographischen Schutz, da sie ohne Kenntnis des Schlüsselwortes sehr leicht entschlüsselt werden können (falls die Länge des Schlüsselwortes im Verhältnis zur

Länge des Kryptotextes relativ kurz ist). Man subtrahiert dazu den Kryptotext y für $\delta = 1, 2, \ldots$ von dem um δ Positionen verschobenen Kryptotext – also $y_{0+\delta}\, y_{1+\delta}\, y_{2+\delta}\, y_{3+\delta}\, \ldots$ minus $y_0\, y_1\, y_2\, y_3 \ldots$ –, bis sinnvoller (Klar-) Text erscheint:

	DUMSQMOZKFN...	(Kryptotext y)
$-$	*DUMSQMO...*	(„Kryptotextschlüsselstrom“)
$=$	....NSCHUTZ...	(Klartext x)

Klartextschlüsselstrom. Neben der oben beschriebenen Analyse der Lauftextverschlüsselung kann das Brechen der *Autokey*-Systeme mit Klartextschlüsselstrom auch analog zur Kasiski-Methode erfolgen: Sei d die Länge des Schlüsselwortes $k_0 \ldots k_{d-1}$. Falls im Klartext die gleiche Buchstabenfolge $x_i \ldots x_{i+l-1}$ im Abstand $2d$ auftritt (beispielsweise $d = 3$ und $l = 2$),

	$\downarrow\ \downarrow \quad \downarrow\ \downarrow$	
	$x_0\, x_1\, x_2\, x_3\, \underset{\sim}{x_4}\, \underset{\sim}{x_5}\, x_6\, x_7\, x_8\ x_9\, \underset{\sim}{x_{10}}\, \underset{\sim}{x_{11}}\, x_{12}\, x_{13}\, x_{14}\ \ldots$	Klartext x
$+$	$k_0\, k_1\, k_2\, k_3\, x_1\, x_2\ x_3\ \underset{\sim}{x_4}\, \underset{\sim}{x_5}\, x_6\, x_7\ x_8\ \ x_9\ \ \underset{\sim}{x_{10}}\, \underset{\sim}{x_{11}} \ldots$	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.

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

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

	HIERKOMMTEINMUSTER	(Klartext)
$+$	*DERHIERKOMMTEINMUS*	(Schlüsselstrom)
$=$	*KMVYSSDWHQUGQCFFYJ*	(Kryptotext)

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

	KMVYSSDWHQUGQCFFYJ	(Kryptotext)
−	*AAAKMVOGXPQKBEWPYJ*	(Schlüsselstrom)
=	KMVOGXPQKBEWPYJQAA	(modifizierter Klartext)

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

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

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

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

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

3 Sicherheit von Kryptosystemen

3.1 Informationstheoretische Sicherheit

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

Bei der Betrachtung der informationstheoretischen Eigenschaften von Kryptosystemen gehen wir von einer Wahrscheinlichkeitsverteilung auf den Paaren $(k, x) \in K \times M$ aus, d. h. $p(k, x)$ gibt die Wahrscheinlichkeit an, dass der Klartext x mit dem Schlüssel k verschlüsselt wird. Dabei setzen wir voraus, dass nach jeder Verschlüsselung einer Nachricht x ein neuer Schlüssel gewählt wird. Dies bedeutet, dass beispielsweise bei der additiven Chiffre für $M = A^n$ zu setzen ist (und nicht $M = A$ wie in 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. O.B.d.A. sei $p(x) > 0$ für alle Klartexte $x \in M$, da wir andernfalls x aus M entfernen können. Für einen Kryptotext y berechnet sich die Wahrscheinlichkeit zu

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

und für einen fest vorgegebenen Kryptotext y 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 entstanden ist. Wir nehmen o.B.d.A. an, dass für alle $y \in C$ $p(y) > 0$ ist (andernfalls kann y aus C entfernt werden).

Definition 3.1 (informationstheoretisch sicher) *Ein Kryptosystem heißt **absolut sicher (informationstheoretisch sicher)**, falls für alle $x \in M$ und alle $y \in C$ gilt:*

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

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

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

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

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

Beispiel 3.2 Sei (M, C, E, D, K) ein Kryptosystem mit $M = \{x_1, \ldots, x_4\}$, $K = \{k_1, \ldots, k_4\}$, $C = \{y_1, \ldots, y_4\}$ und

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

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

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

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

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

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

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

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

Theorem 3.3 *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

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

und damit

$$p(x|y) = \frac{p(x,y)}{p(y)} = \frac{p(x)\cdot p(y|x)}{p(y)} = \frac{p(x)\cdot p(k_{x,y})}{p(y)} = \frac{p(x)\cdot\|K\|^{-1}}{\|K\|^{-1}} = p(x).$$

□

In den Übungen wird gezeigt, dass im Fall $p(x) > 0$ für alle $x \in M$ 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 mit $p(x) > 0$ für alle $x \in M$ 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 (***m**essage*) m steckende Information um so höher bemessen, je seltener m auftritt. Tritt eine Nachricht m mit einer positiven Wahrscheinlichkeit $p(m) = \Pr[X = m] > 0$ auf, dann ist

$$\mathit{Inf}_X(m) = \log_2(1/p(m))$$

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

- Der gemeinsame Informationsgehalt $\mathit{Inf}_{X,Y}(m, m')$ von zwei Nachrichten m und m', die aus stochastisch unabhängigen Quellen X und Y stammen, sollte gleich $\mathit{Inf}_X(m) + \mathit{Inf}_Y(m')$ sein;

– der Informationsgehalt einer Nachricht, die mit Wahrscheinlichkeit $1/2$ auftritt, soll genau 1 (bit) betragen.

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

Definition 3.4 (Entropie) *Sei X eine Zufallsvariable mit Wertebereich $W(X) = \{m_1, \ldots, m_n\}$ und sei $p_i = \Pr[X = m_i]$. Dann ist die **Entropie** von X definiert als*

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

Die Entropie nimmt also im Fall $p_1 = \cdots p_n = 1/n$ den Wert $\log_2(n)$ an. Für jede andere Verteilung $p_1, \ldots, p_n$ gilt dagegen $H(X) < \log_2(n)$ (Beweis 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 $H(X)$ den Wert 0 an. Für den Nachweis von oberen Schranken für die Entropie benutzen wir folgende Hilfsmittel aus der Analysis.

Definition 3.5 (konkav) *Eine reellwertige Funktion f ist **konkav** auf einem Intervall I, falls für alle $x \neq y \in I$ gilt:*

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

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

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

Theorem 3.6 (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*

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

Hierbei tritt Gleichheit genau dann ein, wenn $x_1 = \cdots = x_n$ ist.

Beispiel 3.7 Die Funktion $f(x) = \log_2(x)$ ist streng konkav auf $(0, \infty)$.

Theorem 3.8 *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

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

Nach obigem Satz tritt Gleichheit genau im Fall $1/p_1 = \cdots = 1/p_n$ ein, was mit $p_i = 1/n$ für $i = 1, \ldots, n$ 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 $C = \{x_1, \ldots, x_n\}$ von binären Codewörtern x_i für die Nachrichten m_i mit der Eigenschaft, dass die Abbildung $c : M^* \to \{0,1\}^*$ mit $c(m_{i_1} \cdots m_{i_k}) = x_{i_1} \cdots x_{i_k}$ injektiv ist. Die mittlere Codewortlänge von C unter X ist

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

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

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

Beispiel 3.9 (Entropie) Sei X eine Zufallsvariable mit der Verteilung

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

Dann ergibt sich die Entropie von X zu

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

Betrachten wir die beiden Codes $C_1 = \{\texttt{001}, \texttt{010}, \texttt{011}, \texttt{100}, \texttt{101}, \texttt{110}, \texttt{111}\}$ 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 $|x_i| = \log_2(1/p_i)$ den Wert $L(C_2) = H(X)$ erreicht und somit optimal ist.

Die Redundanz eines Codes für eine Zufallsvariable X ist um so höher, je größer seine mittlere Codewortlänge im Vergleich zur Entropie von X ist. Um auch Codes über unterschiedlichen Alphabeten miteinander vergleichen zu können, ist es

notwendig, die Codewortlänge in einer festen Einheit anzugeben. Hierzu berechnet man die **Bitlänge** eines Wortes x über einem Alphabet A mit $m > 2$ Buchstaben zu $|x|_2 = |x| \log_2(m)$. Beispielsweise ist die Bitlänge von `GOLD` (über dem lateinischen Alphabet) $|\texttt{GOLD}|_2 = 4 \log_2(26) = 18,8$. Entsprechend berechnet sich für einen Code $C = \{x_1, \ldots, x_n\}$ unter einer Verteilung $p_1, \ldots, p_n$ die mittlere Codewortlänge (in bit) zu

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

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

Definition 3.10 (Redundanz) *Die **(relative) Redundanz** eines Codes C für X ist definiert als*

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

Beispiel 3.9 ((Entropie, Fortsetzung))

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

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

wogegen C_2 keine Redundanz besitzt.

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

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

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

Definition 3.11 (Entropie einer Sprache) *Für eine Sprache L über einem Alphabet A mit $\|A\| = m$ und n-Gramm-Verteilung L_n ist $H(L_n)/n$ die **Entropie von** L_n (pro Buchstabe). Falls dieser Wert für n gegen ∞ gegen einen Grenzwert*

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

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

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

*der als die **(relative) Redundanz** von L bezeichnet wird. Der Zähler $\mathcal{R}_{abs}(L) = \log_2 m - H(L)$ 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 3.12 Im Deutschen hat die Einzelbuchstabenverteilung L_1 eine Entropie von $H(L_1) = 4,1$ bit, während eine auf A_{lat} gleichverteilte Zufallsvariable U einen Entropiewert von $H(U) = \log(26) = 4,76$ hat. Für die Bi- und Trigramme ergeben sich Entropiewerte von $H(L_2)/2 = 3,86$ und $H(L_3)/3 = 3,61$ bit pro Buchstabe. Mit wachsender Länge sinkt die Entropie von deutschsprachigen Texten weiter ab und strebt gegen einen Grenzwert $H(L)$ von $1,56$ bit pro Buchstabe.

n	$H(L_n)$	$H(L_n)/n$	$\mathcal{R}_{abs}(L_n)/n$	$R(L_n)$
1	$4,10$	$4,10$	$0,66$	14%
2	$7,72$	$3,86$	$0,90$	19%
3	$10,82$	$3,61$	$1,15$	24%
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
∞	∞	$1,56$	$3,20$	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 3.13 (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 $H(X|Y)$ von X, wenn Y bereits bekannt ist.

Definition 3.14 (bedingte Entropie) *Seien X, Y Zufallsvariablen. Dann ist die **bedingte Entropie** von X unter Y definiert als*

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

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

Theorem 3.15

(i) $H(X,Y) = H(Y) + H(X|Y)$.

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

Beweis s. Übungen. □

Korollar 3.16 *$H(X|Y) \leq H(X)$, wobei Gleichheit genau dann eintritt, wenn X und Y stochastisch unabhängig sind.*

Theorem 3.17 *In jedem Kryptosystem gilt für die Klartextentropie $H(X)$, die Schlüsselentropie $H(K)$ und die Kryptotextentropie $H(Y)$*

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

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

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

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

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

□

Jetzt verfügen wir über alle Hilfsmittel, um die erwartete Anzahl

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

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.

Behauptung 3.18

(i) $H(K|Y_n) \leq \log_2(\bar{s}_n + 1)$,

(ii) $H(K|Y_n) \geq H(K) - nR(L)\log_2 m$.

Beweis

(i) Unter Verwendung der Jensenschen Ungleichung folgt

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

(ii) Mit Satz 3.17 folgt

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

Die Klartextentropie $H(X_n)$ lässt sich durch

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

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

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

abschätzen. Somit ist

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

□

Zusammen ergibt sich also

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

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

Theorem 3.19 *Werden mit einem Kryptosystem Klartexte $x \in A^n$ der Länge n mit einem unter Gleichverteilung gezogenen Schlüssel $k \in K$ verschlüsselt, und ist $\|C_n\| = \|A^n\| = m^n$ für den zugehörigen Kryptotextraum $C_n = \{E(k, x) \mid k \in K, x \in A^n\}$, so gilt für die erwartete Anzahl $\bar{s}_n$ der unechten Schlüssel,*

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

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

Korollar 3.20 *Unter den Bedingungen des obigen Satzes gilt*

$$n_0 \geq \frac{\log_2 \|K\|}{R(L) \log_2 m} = \frac{\log_2 \|K\|}{\log_2 m - H(L)} = \frac{\log_2 \|K\|}{\mathcal{R}_{abs}(L)}.$$

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 3.21 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 $R(L) = 3,2/4,76 \approx 67\%$ entspricht):

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

Dagegen erhalten wir für Blocktranspositionen folgende 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	22986
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 Schätzwerte 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 Komplexitätstheoretische Sicherheit

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 3.22 (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 zwei Klartexte $x_0 \neq x_1 \in M$ generiert und V bei Eingabe dieser Klartexte und eines Kryptotextes $y \in C$ ein Bit ausgibt. Der Vorteil von (G, V) ist*

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

wobei die Zufallsvariablen X_0, X_1 die von G generierte Ausgabe und B die zufällige Wahl eines Bits beschreiben, d.h. $\Pr[B = 0] = \Pr[B = 1] = 1/2$.

Theorem 3.23 *Ist ein Kryptosystem S unter einer Klartextverteilung $p(x) > 0$ für alle $x \in M$ absolut sicher, dann erzielt kein Gegner einen Vorteil größer 0.*

Beweis Falls S unter einer Klartextverteilung $p(x) > 0$ für alle $x \in M$ absolut sicher ist, so ist S unter jeder Klartextverteilung absolut sicher (siehe Übungen). Folglich ist die Kryptotextverteilung $E(K, X)$ von der Klartextverteilung X stochastisch unabhängig (auch wenn wir die Klartextverteilung X abändern). Somit sind die beiden ZVen $(X_0, X_1, E(k, X))$ und $(X_0, X_1, E(k, X'))$ für jeden Generator $G = (X_0, X_1)$ und beliebige Klartextverteilungen X, X' identisch verteilt, weshalb auch $V(X_0, X_1, E(k, X))$ und $V(X_0, X_1, E(k, X'))$ identisch verteilt sind. Daher folgt

$$\begin{aligned}\Pr[V(X_0, X_1, E(k, X_B)) = B] &= \Pr[V(X_0, X_1, E(k, X_0)) = B] \\ &= 1/2.\end{aligned}$$

□

In den Übungen wird auch die umgekehrte Implikation bewiesen: Jedes KS, bei dem kein Gegner einen Vorteil größer 0 erzielt, ist absolut sicher. 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?

Wir beantworten diese Fragen durch folgende Definitionen. Dabei gehen wir davon aus, dass die Sicherheit des Kryptosystems S von der Wahl eines prinzipiell beliebig groß wählbaren Sicherheitsparameters $s \in \mathbb{N}$ abhängt. Alle legalen Operationen (wie die Schlüsselgenerierung oder die Chiffrierung) sollten effizient (d.h. in Zeit $s^{O(1)}$) durchführbar sein. Typischerweise werden Kryptosysteme nach der Schlüssellänge $|k|$ parameterisiert, wobei dann nur Klartexte der Länge $|k|^{O(1)}$ mit demselben Schlüssel verschlüsselt werden.

Definition 3.24 (komplexitätstheoretisch sicher) *Sei S ein Kryptosystem, das mit variablem Parameterwert s betrieben werden kann.*

- *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, V) für S heißt **effizient**, wenn sowohl G als auch V durch probabilistische Schaltkreise der Größe $s^{O(1)}$ berechenbar sind.*
- *S heißt **komplexitätstheoretisch sicher**, wenn jeder effiziente Gegner nur einen vernachlässigbaren Vorteil $\alpha(s)$ hat.*

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 4.1 (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 4.2

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 4.3 (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 4.4 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:

$$w^1 \leftarrow g(K^1, w^0)$$
$$\vdots$$
$$w^N \leftarrow g(K^N, w^{N-1})$$

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

$$w^{N-1} \leftarrow g^{-1}(K^N, w^N)$$
$$\vdots$$
$$w^0 \leftarrow g^{-1}(K^1, w^1)$$

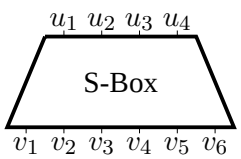
Beispiele für iterierte Chiffren sind der aus 16 Runden bestehende DES-Algorithmus und der AES mit einer variablen Rundenzahl $N \in \{10, 12, 14\}$, die wir in späteren Abschnitten behandeln werden.

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 4.5 (Teilwort) *Für ein Wort $u = u_1 \cdots u_n \in \{0,1\}^n$ und Indizes $1 \le i \le j \le n$ bezeichne $u[i,j]$ das **Teilwort** $u_i \cdots u_j$ von u. Im Fall $n = lm$ 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 4.6 (Substitutions-Permutations-Netzwerk) *Sei $A = \{0,1\}$ und sei $M = C = A^{lm}$ für natürliche Zahlen $l, m \geq 1$. Ein **Substitutions-Permutations-Netzwerk** (SPN) wird durch Permutationen $\pi_S : \{0,1\}^l \to \{0,1\}^l$ und $\pi_P : \{1, \ldots, lm\} \to \{1, \ldots, lm\}$ sowie durch einen Schlüsselgenerator $f : \{0,1\}^k \to \{0,1\}^{lm(N+1)}$ beschrieben. Der Generator f erzeugt aus einem (externen) Schlüssel $K \in \{0,1\}^k$ eine Folge $f(K) = (K^1, \ldots, K^{N+1})$ von $N+1$ Rundenschlüsseln K^r, unter denen ein Klartext $x \in \{0,1\}^{lm}$ gemäß folgendem Algorithmus in einen Kryptotext $y = E_{f,\pi_S,\pi_P}(K, x) \in \{0,1\}^{lm}$ überführt wird.*

Algorithmus 4.7 $E_{f,\pi_S,\pi_P}(K, x)$

```
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  end
7  u^N ← w^{N−1} ⊕ K^N
8  v^N ← S(u^N)
9  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).

Beispiel 4.8 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:

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

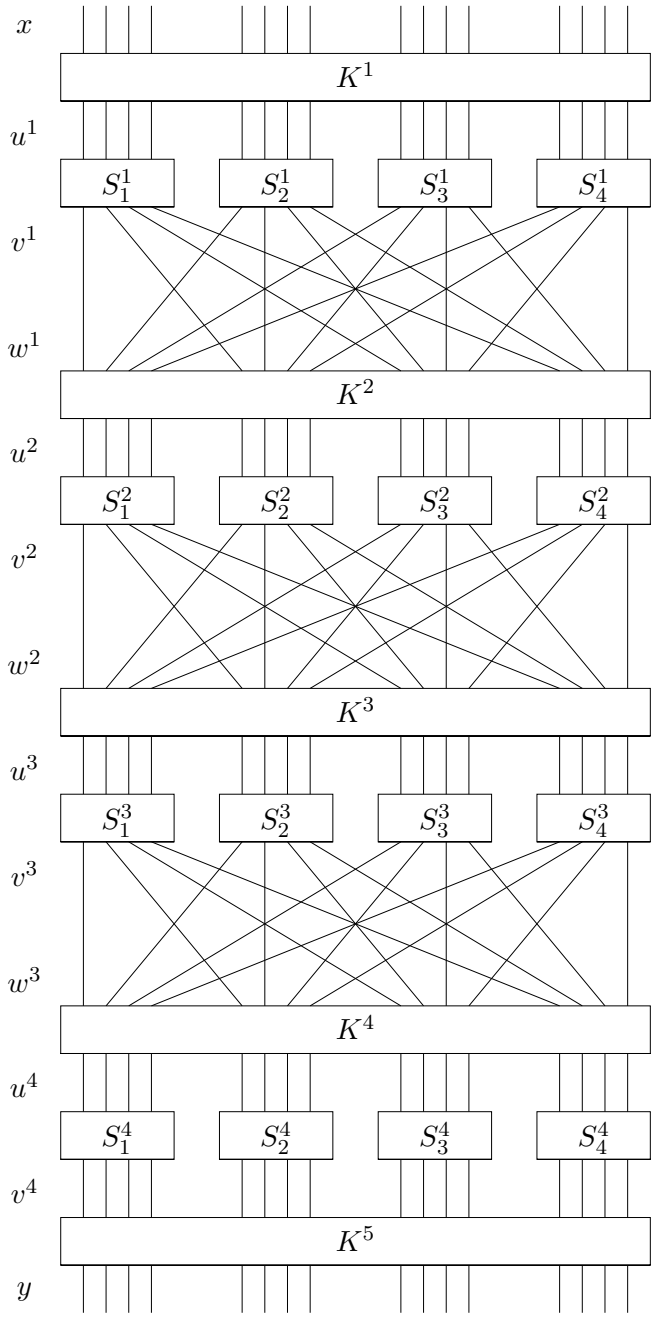


Abbildung 4.1: Ein Substitutions-Permutations-Netzwerk.

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 das Bias $\varepsilon(U_a \oplus V_b \oplus c)$ der Zufallsvariablen $U_a \oplus V_b \oplus c$ einen möglichst großen Wert haben. 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 4.9 Wir betrachten die S-Box $\pi_S : \{0,1\}^4 \to \{0,1\}^4$ aus Beispiel 4.8. 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 $L(a,b)$.

	b															
a	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
1	8	8	6	6	8	8	6	14	10	10	8	8	10	10	8	8
2	8	8	6	6	8	8	6	6	8	8	10	10	8	8	2	10
3	8	8	8	8	8	8	8	8	10	**2**	6	6	10	10	6	6
⋮									⋮							
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 4.8 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

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

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?

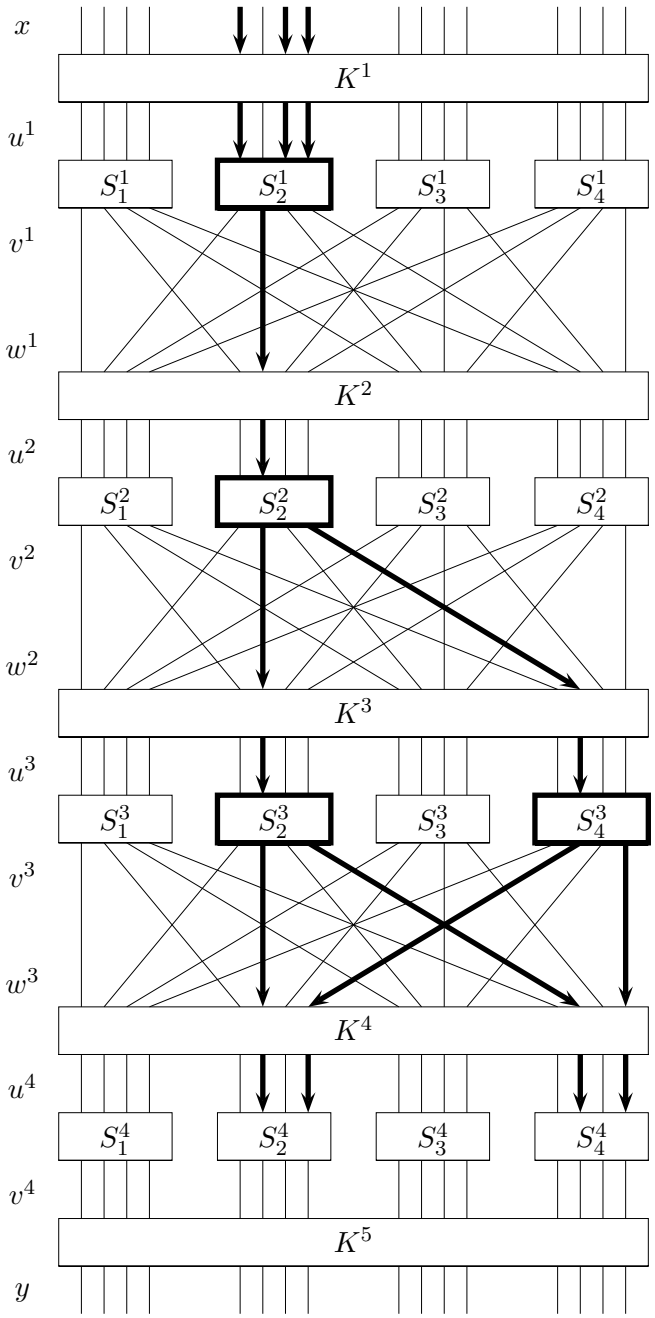


Abbildung 4.2: Eine lineare Approximation an ein Substitutions-Permutations-Netzwerk.

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

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

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

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

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

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

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

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

Nun zu Frage 2: Wären die Zufallsvariablen $T_1, \ldots, T_4$ unabhängig, so würde uns das 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 4.10 (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 IV hat die Zufallsvariable $Z = X_1 \oplus \cdots \oplus X_n$ ein Bias von $\varepsilon(Z) = 2^{n-1}\varepsilon(X_1)\cdots\varepsilon(X_n)$ 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 4.11 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 $1 \leq i < j \leq 3$. 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 $(K_{(2)}^5, K_{(4)}^5)$ die zu einem Kryptotext y gehörigen Werte u_6^4, u_8^4, u_{14}^4 und u_{16}^4 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 4.12 LINEARATTACK

for $(L_1, L_2) \leftarrow (0,0)$ **to** (F,F) **do**
  $\alpha(L_1, L_2) \leftarrow 0$ **end**
for each $(x, y) \in M$ **do**
  for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F) **do**
    $v^4_{(2)} \leftarrow L_1 \oplus y_{(2)}$; $v^4_{(4)} \leftarrow L_2 \oplus y_{(4)}$
    $u^4_{(2)} \leftarrow \pi_S^{-1}(v^4_{(2)})$; $u^4_{(4)} \leftarrow \pi_S^{-1}(v^4_{(4)})$
    if $x_5 \oplus x_7 \oplus x_8 \oplus u^4_6 \oplus u^4_8 \oplus u^4_{14} \oplus u^4_{16} = 0$ **then**
      $\alpha(L_1, L_2) \leftarrow \alpha(L_1, L_2) + 1$
  end
end
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0,0)$ **to** (F, F) **do**
  $\beta(L_1, L_2) \leftarrow |\alpha(L_1, L_2) - t/2|$
  if $\beta(L_1, L_2) > max$ **then**
    $max \leftarrow \beta(L_1, L_2)$
    $maxkey \leftarrow (L_1, L_2)$
end
Ausgabe: maxkey

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

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

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

die Menge aller Eingabepaare, die die Differenz 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 4.14 (Differential, Weitergabequotient) *Sei $a' \in \{0,1\}^l$ eine Eingabe- und sei $b' \in \{0,1\}^{l'}$ eine Ausgabedifferenz für eine S-Box π_S. Dann heißt (a', b') **Differential**. Die Anzahl der Eingabepaare (x, x^*), die die Eingabedifferenz a' in die Ausgabedifferenz b' überführen, bezeichnen wir mit $D(a', b')$, d.h.*

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

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

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

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

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

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

Beispiel 4.15 Betrachten wir die S-Box $\pi_S : \{0,1\}^4 \to \{0,1\}^4$ aus Beispiel 4.8, so erhalten wir für die Eingabedifferenz $a' = 1011$ die Menge

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

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

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

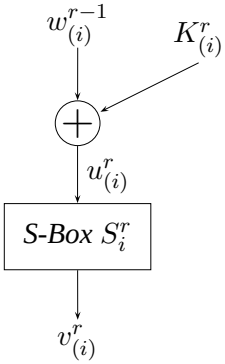
Die Ausgabedifferenz $b' = 0010$ kommt also $D(a', 0010) = 8$ Mal vor, während die Differenzen 0101, 0111, 1101 und 1111 je zwei Mal und die übrigen Werte überhaupt nicht vorkommen (siehe Zeile B in nachfolgender Tabelle). Führen wir diese Berechnungen für jede der $2^4 = 16$ Eingabedifferenzen $a' \in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

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

Wie können wir uns die Abhängigkeit der Ausgabedifferenz y' von der konkreten Realisierung (x, x^*) der Eingabedifferenz x' zunutze machen, um einzelne Schlüsselbits zu bestimmen? Eine erste Antwort auf diese Frage gibt die folgende Beobachtung.

Beobachtung 4.16 *Für die Eingabe $u^r_{(i)}$ der S-Box S^r_i in Runde r gilt*

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



Daher hängt die Eingabedifferenz

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

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

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

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

Können wir nun in einem SPN für bestimmte S-Boxen S_i Differentiale (a'_i, b'_i) finden, so dass die Eingabedifferenz dieser Differentiale mit der (permutierten) Ausgabedifferenz der Differentiale in der jeweils vorhergehenden Runde übereinstimmt (siehe Abbildung 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 oder nicht, berechnet sich der Weitergabequotient der Spur als das Produkt der Weitergabequotienten der beteiligten Differentiale. Obwohl diese Annahme i.a. nicht zutrifft, treten in der praktischen Anwendung kaum große Abweichungen von diesem hypothetischen Wert auf.

Beispiel 4.17 Betrachten wir das SPN aus Beispiel 4.8, 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 eine Eingabedifferenz

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

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

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

welche wiederum mit Wk 1 auf die Differenz

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

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

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

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

Sei nun (a', b') eine Spur 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 Spur (a', b') folgen und daher bei Verschlüsselung mit K Zwischenergebnisse u^4 und $(u^4)^*$ liefern, die die Differenz

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

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

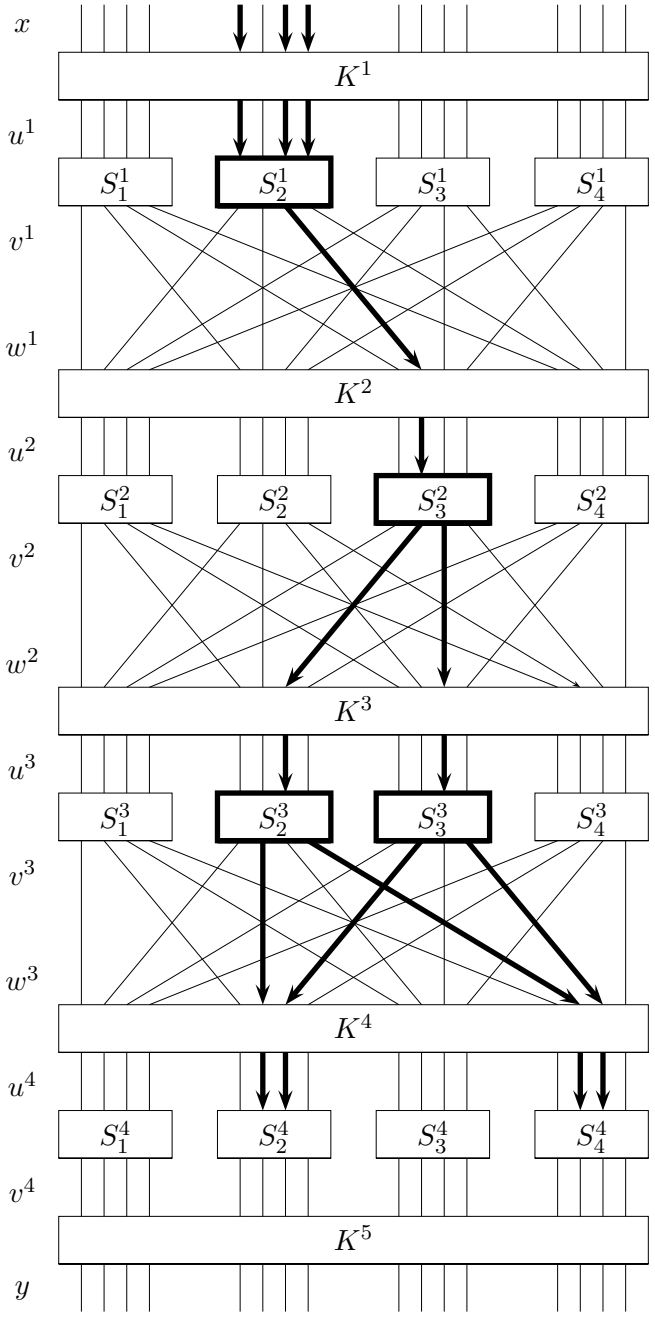


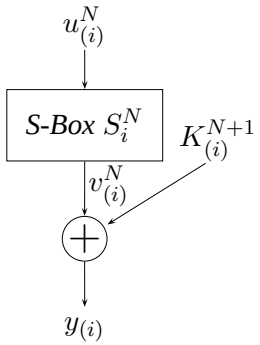
Abbildung 4.3: Eine Differentialspur für ein Substitutions-Permutations-Netzwerk.

Beobachtung 4.18 *Für die Ausgabe $v^r_{(i)}$ der S-Box S^r_i in Runde $r = N$ gilt*

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

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

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



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

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

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

$$\pi^{-1}_{S^N_i}(y_{(i)} \oplus k) \oplus \pi^{-1}_{S^N_i}(y^*_{(i)} \oplus k) = b'_{(i)} \tag{4.3}$$

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

Gemäß Beobachtung 4.18 kann jedes richtige Doppelpaar dazu benutzt werden, die in Frage kommenden Werte für bestimmte Blöcke des Rundenschlüssels K^5 einzuschränken. Ist M hinreichend groß, so wird sich schließlich der richtige Teilschlüssel als derjenige heraus kristallisieren, der mit den meisten Doppelpaaren konsistent ist. Wir benutzen nun die in Beispiel 4.17 gefundene Spur für einen Angriff mittels differentieller Analyse.

Beispiel 4.19 In Algorithmus 4.5 wird für jeden Subschlüssel-Kandidaten (L_1, L_2) für $(K^5_{(2)}, K^5_{(4)})$ die Anzahl $\gamma(L_1, L_2)$ aller Doppelpaare (x, x^*, y, y^*) in M bestimmt, die nicht als falsch erkannt werden und mit (L_1, L_2) konsistent sind. Ausgegeben wird der Kandidat (L_1, L_2) mit dem größten γ-Wert.

Algorithmus 4.20 DIFFERENTIALATTACK

for $(L_1, L_2) \leftarrow (0, 0)$ **to** (F,F) **do**
  $\gamma(L_1, L_2) \leftarrow 0$ **end**
for each $(x, x^*, y, y^*) \in M$ **do**
  if $y_{(1)} = y^*_{(1)}$ und $y_{(3)} = y^*_{(3)}$ **then**
    for $(L_1, L_2) \leftarrow (0, 0)$ **to** (F, F) **do**
      $v^4_{(2)} \leftarrow L_1 \oplus y_{(2)}$; $v^4_{(4)} \leftarrow L_2 \oplus y_{(4)}$
      $u^4_{(2)} \leftarrow \pi_S^{-1}(v^4_{(2)})$; $u^4_{(4)} \leftarrow \pi_S^{-1}(v^4_{(4)})$
      $(v^4_{(2)})^* \leftarrow L_1 \oplus y^*_{(2)}$; $(v^4_{(4)})^* \leftarrow L_2 \oplus y^*_{(4)}$
      $(u^4_{(2)})^* \leftarrow \pi_S^{-1}((v^4_{(2)})^*)$; $(u^4_{(4)})^* \leftarrow \pi_S^{-1}((v^4_{(4)})^*)$
      $(u^4_{(2)})' \leftarrow u^4_{(2)} \oplus (u^4_{(2)})^*$; $(u^4_{(4)})' \leftarrow u^4_{(4)} \oplus (u^4_{(4)})^*$
      if $(u^4_{(2)})' = 0110$ und $(u^4_{(4)})' = 0110$ **then**
        $\gamma(L_1, L_2) \leftarrow \gamma(L_1, L_2) + 1$
    end
end
$max \leftarrow -1$
for $(L_1, L_2) \leftarrow (0, 0)$ **to** (F, F) **do**
  if $\gamma(L_1, L_2) > max$ **then**
    $max \leftarrow \gamma(L_1, L_2)$
    $maxkey \leftarrow (L_1, L_2)$
end
Ausgabe: maxkey

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