Anhang III zur Vorlesung Kryptologie: Blockchiffren

von
Peter Hellekalek

Fakultät für Mathematik, Universität Wien, und
Fachbereich Mathematik, Universität Salzburg

Tel: +43-(0)662-8044-5310
Fax: +43-(0)662-8044-137
e-mail: peter.hellekalek@sbg.ac.at
web: http://random.mat.sbg.ac.at/

Wien, 24. April 2009

Inhaltsverzeichnis

Kapitel 1

Blockchiffren: DES und AES

▷ **Inhalt**
In diesem Kapitel diskutieren wir wichtige Beispiel von Blockchiffren, die für die praktische und die theoretische Kryptographie von großer Bedeutung sind.

▷ **Ziel**
Wir stellen die Konzepte vor, auf denen DES und AES aufbauen.

▷ **Stichwörter**
Die Stichwörter zu diesem Kapitel lauten

- DES
- AES

▷ **Literatur**

Schneier, B.[Sch96] *Applied Cryptography.* 2nd edition, Wiley, New York 1996.

Stinson, D. R.[Sti06] *Cryptography. 3rd edition.* Chapman and Hall/CRC Press, Boca Raton 2006.

W. Trappe and L. Washington[TW06] *Introduction to Cryptography with Coding Theory.* Pearson Prentice Hall, 2nd edition, 2006.

sowie die folgende Monographie zu AES:
Daemen, J. and Rijmen, V.[DR02] *The Design of Rijndael.* Springer Verlag, New York 2002.

1.1 Grundlegendes zu Blockchiffren

Blockchiffren arbeiten den Klartext blockweise ab. Dazu wird der Klartext in Zeichenblöcke fester Länge zerlegt. Jeder Klartextblock wird dann getrennt verschlüsselt.

Blockchiffren sind zwar etwas langsamer als Stromchiffren, aber dafür wesentlich vielseitiger. Für verschiedene Aufgabenbereiche wurden verschiedene Betriebsmodi entwickelt, zum Beispiel kann eine Blockchiffre wie ein Stromchiffrierverfahren betrieben werden.

Blockchiffren sind Teil vieler moderner Sicherheitswerkzeuge wie PGP oder SSH. In vielen Fällen kann man in solchen Softwarepaketen sogar zwischen mehreren Blockchiffren wählen.

Wie bei jedem Kryptosystem muß die Verschlüsselungsfunktion injektiv sein. Wenn m die Blocklänge bezeichnet und $\mathcal{A}$ das Zeichenalphabet, über dem die Klartextblöcke gebildet werden, dann ist eine Blockchiffre also nichts anderes als eine injektive Abbildung der endlichen Menge $\mathcal{P} = \mathcal{A}^m$ in die endliche Menge $\mathcal{C} = \mathcal{A}^n$. Notwendigerweise muss $m \leq n$ gelten. Gründe wie die Verwendbarkeit der gleichen Implementation sowohl für die Ver- als auch die Entschlüsselung und die Notwendigkeit, unnötige Datenexpansion bei der Verschlüsselung zu vermeiden, haben dazu geführt, dass man bei Blockchiffren bei der Ver- und der Entschlüsselung mit gleicher Blocklänge arbeitet, dass also $m = n$ gilt.

Eine Blockchiffre ist also nichts anderes als eine Substitutionschiffre, die Elemente $p \in \mathcal{A}^m$ bijektiv auf Elemente $c \in \mathcal{A}^m$ abbildet. Aus der klassischen Kryptographie wissen wir, dass der Schlüsselraum riesig ist, dass aber die Häufigkeitsverteilung der Elemente des Klartextraums $\mathcal{P}$ bei der Verschlüsselung erhalten bleibt. Wie kann eine derartige Chiffre sicher sein?

In der modernen Datenverarbeitung wird $\mathcal{A} = \{0, 1\}$ gewählt. Typische Werte für die Blocklänge m sind zur Zeit $m = 64, 128$ oder 256.. Auf diese Weise werden einerseits kryptanalytische Attacken mittels Häufigkeitsanalyse erschwert, denn der Klartextraum ist nun riesig. Andererseits ist mit derartigen Blocklängen die moderne Prozessorarchitektur berücksichtigt.

1.1 Bemerkung (Designkriterien)
Die praktische Aufgabe für die Konstruktion einer Blockchiffre $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ lautet wie folgt: Finde eine Familie $\mathcal{E}$ von bijektiven Abbildungen von $\mathcal{P} = \mathcal{A}^m = \mathcal{C}$ auf sich mit den Eigenschaften:

1. (Hoher Datendurchsatz) $\forall\, e_k \in \mathcal{E}$ und $\forall\, d_k \in \mathcal{D}$, $k \in \mathcal{K}$, gilt:
   $e_k : \mathcal{P} \to \mathcal{C}$ und $d_k : \mathcal{C} \to \mathcal{P}$ können effizient berechnet werden.

2. (Sicherheit) Die Blockchiffre ist kryptographisch sicher.

In der Praxis wird mit parametrisierten Familien $\mathcal{E} = \{e_k : k \in \mathcal{K}\}$ gearbeitet, wobei $\mathcal{K}$ den Schlüsselraum der Chiffre bezeichnet. Auf Grund der derzeit bekannten kryptoanalytischen Attacken haben sich die folgenden Designkriterien für moderne Blockchiffren ergeben, die bei der Auswahl der Verschlüsselungsfunktionen e_k zu berücksichtigen sind:

1. *Vollständigkeit*
   Jedes Output-Bit soll von jedem Input-Bit abhängen und weiters von jedem Bit des Schlüssels.

2. *Lawineneffekt*
   Die Änderung eines einzigen Input-Bits soll etwa die Hälfte der Output-Bits ändern.

3. *Nichtlinearität*
   Kein Output-Bit soll affin-linear von Input-Bits abhängen.

Diese Kriterien entsprechen den Forderungen von Shannon nach Diffusion und Konfusion (siehe Shannon[Sha49]). Die Forderung der Nichtlinearität steht in Zusammenhang mit der differentiellen und linearen Kryptanalyse (siehe dazu Stinson [Sti06] und Menezes et al. [MvOV97]).

1.2 Algebraische Grundlagen

1.2.1 Vorbemerkungen

1.2 Lemma Sei I eine endliche oder abzählbar unendliche Indexmenge, $I = \{1, 2, \dots\}$, für alle $i \in I$ sei G_i eine Gruppe mit neutralem Element e_i und sei

$$G = G_1 \times G_2 \times \dots \;=\; \big\{(\dots, g_i, \dots) :\; g_i \in G_i,\, i \in I\big\}$$

das kartesische Produkt der Mengen G_i. Sei weiters

$$\bigoplus_{i \in I} G_i \;=$$
$$\big\{(\dots, g_i, \dots) :\; g_i = e_i, \text{ mit Ausnahme höchstens endlich vieler } i \in I\big\}.$$

Für $g = (\dots, g_i, \dots)$ und $h = (\dots, h_i, \dots)$ aus G definieren wir

$$g \circ h = (\dots, g_i\, h_i, \dots).$$

Dann gilt:

1. $(G, \circ)$ ist eine Gruppe.
2. $\bigoplus_{i \in I} G_i$ ist ein Normalteiler von G
3. Es gilt folgende Beziehung für die Kommutativität:

$$\begin{aligned} (G, \circ) \text{ ist abelsch.} \quad &\Leftrightarrow \quad \left(\bigoplus_{i \in I} G_i, \circ\right) \text{ ist abelsch.} \\ &\Leftrightarrow \quad \forall\, i \in I: \quad G_i \text{ ist abelsch.} \end{aligned}$$

4. Wenn die Indexmenge I endlich ist, gilt

$$\bigoplus_{i \in I} G_i \;=\; G.$$

Beweis. Der Beweis ist einfach, langweilig und wird daher ausgelassen. □

Wir merken an, dass diese Definitionen für eine beliebige Indexmenge I einen Sinn machen. Da wir diese Allgemeinheit im Folgenden nicht benötigen, verzichten wir auf dieses abstrakte Konzept.

1.3 Definition (Direktes Produkt, direkte Summe)
Die Gruppe $(G, \circ)$ heißt das direkte Produkt der Gruppen G_i und wird mit

$$\prod_{i \in I} G_i$$

bezeichnet.

Die Gruppe $\left(\bigoplus_{i \in I} G_i, \circ\right)$ heißt die (äußere) direkte Summe der Gruppen G_i.

1.4 Bemerkung Wenn $I = \{1, \ldots, n\}$ eine endlich Indexmenge ist, so schreiben wir

- an Stelle von $\prod_{i=1}^{n} G_i$ häufig $G_1 \times \cdots \times G_n$,
- an Stelle von $\bigoplus_{i=1}^{n} G_i$ häufig $G_1 \oplus \cdots \oplus G_n$,
- falls zusätzlich $G_1 = G_2 = \cdots = G_n = G$ gilt, an Stelle von

$$\prod_{i=1}^{n} G_i \quad \text{stets} \quad G^n.$$

1.2.2 Polynomringe

Wir gehen von einem Ring $(R, +, \cdot)$ aus. Wie könnte man den Begriff "Polynom mit Koeffizienten aus R" definieren?

Ein erster (naiver) Ansatz ist es, einen Ausdruck der Form

$$a_0 + a_1 x + \cdots + a_n x^n$$

mit Koeffizienten $a_i \in R$ als Polynom zu bezeichnen. Beispielsweise wäre dann für $(R, +, \cdot) = (\mathbb{Z}, +, \cdot)$ der Ausdruck

$$7x^2 - 3x + 2$$

ein Polynom mit Koeffizienten aus $\mathbb{Z}$.

Bei der Sache gibt es aber ein Problem. Betrachten wir dazu als den zugrundeliegenden Ring den Restklassenring $(\mathbb{Z}_2, +, \cdot)$. Sei weiters $f(x) = x^2 - x = x(x-1)$ und sei $g(x) = 0$, $x \in \mathbb{Z}_2$. Dann kommen wir in Schwierigkeiten, denn das Polynom f ist verschieden vom Polynom g, die Funktion $f : \mathbb{Z}_2 \to \mathbb{Z}_2$ ist aber gleich der Funktion $g : \mathbb{Z}_2 \to \mathbb{Z}_2$:

$$f(x) = g(x) = 0 \qquad \forall x \in \mathbb{Z}_2.$$

Wir haben also zwei Objekte f und g, für die gilt: Betrachtet man f und g als zwei Polynome, dann sind sie verschieden. Betrachtet man f und g aber als zwei Funktionen, dann sind sie gleich.

1.5 Bemerkung
Der Begriff des Polynomes ist also etwas Anderes als einfach ein Name für spezielle Funktionen.

1.6 Definition (Polynom, formale Potenzreihe)
Sei $(R, +, \cdot)$ ein Ring. Wir bezeichnen die direkte Summe

$$(R, +) \oplus (R, +) \oplus (R, +) \oplus \dots$$

mit $R[X]$ und das direkte Produkt

$$(R, +) \times (R, +) \times (R, +) \times \dots$$

mit $R[[X]]$. Wir nennen die Elemente von $R[X]$ Polynome über R und die Elemente von $R[[X]]$ formale Potenzreihen über R.

Für $f = (a_0, a_1, a_2, \dots) \in R[X]$ heißen $a_0, a_1, \dots$ die Koeffizienten des Polynoms f.

Das spezielle Polynom $\mathbf{0} = (0, 0, 0, \dots)$ heißt das Nullpolynom.

Sei $f \in R[X]$ mit $f \neq \mathbf{0}$. Unter dem Grad von f verstehen wir die nichtnegative ganze Zahl

$$\deg(f) = \max\{n \in \mathbb{N} \cup \{0\} : \ a_n \neq 0\}.$$

Der Koeffizient a_k mit $k = \deg(f)$ heißt dann der Leitkoeffizient von f. Ein Polynom vom Grad 0 heißt ein konstantes Polynom. Ein nichtkonstantes Polynom $f \neq \mathbf{0}$ heißt normiert, wenn sein Leitkoeffizient gleich dem Einselement des Ringes $(R, +, \cdot)$ ist.

Wir nennen zwei Polynome $f = (a_0, a_1, a_2, \dots)$ und $g = (b_0, b_1, b_2, \dots)$ gleich, wenn

$$a_k \ = \ b_k \qquad \forall\, k = 0, 1, \dots$$

1.7 Bemerkung
Wir erinnern an die Definition der direkten Summe von Gruppen. Jedes Element f der direkten Summe

$$R[X] = (R, +) \oplus (R, +) \oplus (R, +) \oplus \dots$$

hat die Gestalt $f = (a_0, a_1, a_2, \dots)$ mit $a_i \in R$ und es sind nur endlich viele der a_i verschieden vom Nullelement von R. Ab einer Stelle sind also alle weiteren Koeffizienten a_i gleich 0,

$$f \ = \ (a_0, a_1, a_2, \dots, a_n, 0, 0, \dots).$$

In Ergänzung zu Definition 1.6 halten wir fest:

- Konstante Polynome haben die Gestalt $f = (a_0, 0, 0, \dots)$
- Der Begriff des normierten Polynomes setzt voraus, dass es im Ring $(R, +, \cdot)$ ein Einselement gibt. Normierte Polynome sind also nur definiert für Ringe mit Einselement.
- Ein normiertes Polynom hat die Gestalt $f = (a_0, a_1, \dots, a_{n-1}, 1)$, wobei $n = \deg(f)$, $n \geq 1$.

- Wenn wir ein Polynom $f \in R[X]$ in der Form $f = (a_0, \ldots, a_n)$ schreiben, dann meinen wir damit das Polynom $f = (a_0, a_1, \ldots, a_n, 0, 0, \ldots)$.

1.8 Bemerkung
Von den Eigenschaften der direkten Summe und des direkten Produkts von Gruppen her wissen wir, dass $(R[X], +)$ mit der Verknüpfung $(f = (a_0, a_1, a_2, \ldots), g = (b_0, b_1, b_2, \ldots))$

$$f + g \;=\; (a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots)$$

eine kommutative Gruppe ist.

Wir definieren nun auf $R[X]$ eine zweite innere Verknüpfung:

$$f \cdot g \;=\; (c_0, c_1, c_2, \ldots),$$

wobei

$$c_n \;:=\; \sum_{k=0}^{n} a_k\, b_{n-k} \qquad \forall\, n \geq 0.$$

Somit erhalten wir ein Tripel $(R[X], +, \cdot)$ mit dem Nullelement

$$\mathbf{0} = (0, 0, 0, \ldots)$$

und dem Einselement

$$\mathbf{1} = (1, 0, 0, \ldots),$$

falls der Ring $(R, +, \cdot)$ ein Einselement 1 besitzt.

1.9 Lemma
Sei $(R, +, \cdot)$ ein Ring mit Einselement und sei

$$X = (0, 1, 0, \ldots).$$

Dann kann jedes Element $f = (a_0, a_1, \ldots, a_n, 0, 0, \ldots)$ von $R[X]$ in der Form

$$f \;=\; a_0 + a_1 X + \cdots + a_n X^n$$

geschrieben werden.

Beweis. Aus der Definition der inneren Verknüpfung "$\cdot$" erhalten wir

$$X^2 \;=\; X \cdot X \;=\; (0, 0, 1, 0, 0, \ldots)$$

und

$$X^3 \;=\; X \cdot X \cdot X \;=\; (0, 0, 0, 1, 0, 0, \ldots).$$

Es liegt daher die Vermutung nahe, dass

$$X^n \;=\; (\underbrace{0, 0, \cdots, 0}_{n \text{ mal}}, 1, 0, 0, \ldots),$$

was leicht mittels Induktion zu beweisen ist.

Von nun an identifizieren wir das konstante Polynom $(a, 0, 0, \dots) \in R[X]$ mit dem Ringelement a. Weiters schreiben wir das Produkt

$$(a, 0, 0, \dots) \cdot f$$

für $f \in R[X]$ in der Form

$$af.$$

Diese Festlegung ist sinnvoll, denn

$$\begin{aligned} aX &= (a, 0, 0, \dots) \cdot (0, 1, 0, 0, \dots) \\ &= (0, a, 0, 0, \dots) \\ aX^2 &= (0, 0, a, 0, 0, \dots) \\ aX^3 &= (0, 0, 0, a, 0, 0, \dots) \\ &\dots \end{aligned}$$

für alle $a \in R$. Daher erhalten wir für $f = (a_0, a_1, \dots, a_n, 0, 0, \dots) \in R[X]$ die Darstellung

$$\begin{aligned} f &= (a_0, 0, 0, \dots) + (0, a_1, 0, 0, \dots) + \dots + (0, 0, \dots, 0, a_n, 0, 0, \dots) \\ &= a_0 + a_1X + a_2X^2 + \dots + a_nX^n. \end{aligned}$$

□

1.10 Bemerkung
Sei $(R, +, \cdot)$ ein Ring mit Einselement 1. Das Symbol X bezeichnet keine Unbestimmte, sondern etwas ganz Bestimmtes:

- X ist die Bezeichnung für das Polynom $(0, 1, 0, 0, \cdots) \in R[X]$.
- X^2 ist das Produkt des Polynoms $X = (0, 1, 0, 0, \cdots)$ mit sich selbst. Es gilt daher nach der Definition des Produktes zweier Polynome:
$$X^2 = (0, 0, 1, 0, 0, \cdots).$$
- X^3, X^4, X^5, ... ergeben sich analog.
- Nach Lemma 1.9 gilt für $f = (a_0, a_1, \dots, a_n) \in R[X]$ daher
$$f = a_0 + a_1X + a_2X^2 + \dots + a_nX^n.$$

1.11 Satz

1. Sei $(R, +, \cdot)$ ein Ring. Dann ist $(R[X], +, \cdot)$ ebenfalls ein Ring.
2. Wenn R kommutativ ist, dann ist auch $R[X]$ kommutativ.
3. Wenn R ein Einselement besitzt, dann besitzt auch $R[X]$ ein Einselement.
4. Wenn R ein Integritätsbereich ist, dann ist auch $R[X]$ ein Integritätsbereich.
5. Wenn R ein Körper ist, dann folgt daraus im Allgemeinen nicht, dass auch $R[X]$ ein Körper ist.

Beweis. Der Beweis zu Punkt 1 besteht aus langweiligem Rechnen und wird daher ausgelassen.

Zu 2.
Wir müssen zeigen, dass $f \cdot g = g \cdot f$ für alle Elemente f, g von $R[X]$ gilt. Dazu betrachten wir den Koeffizienten

$$c_n \;=\; \sum_{k=0}^{n} a_k \, b_{n-k}$$

des Polynomes $f \cdot g$. Wegen der Kommutativität von R kann man schreiben

$$c_n \;=\; \sum_{k=0}^{n} b_{n-k} \, a_k.$$

Da die Addition in einem Ring kommutativ ist, können wir die Reihenfolge der Summation verändern und erhalten

$$c_n \;=\; \sum_{j=0}^{n} b_j \, a_{n-j}.$$

Die Multiplikation ist somit auch in $R[X]$ kommutativ.

Zu 3.
Diese Behauptung ist trivial. Das Einselement von $R[X]$ ist $\mathbf{1} = (1, 0, 0, \dots)$.

Zu 4.
Sei $f = (a_0, a_1, a_2, \dots)$, $g = (b_0, b_1, b_2, \dots)$ und sei weiters

$$f \cdot g \;=\; (c_0, c_1, c_2, \dots) \;=\; \mathbf{0}.$$

Zusätzlich nehmen wir an, dass $g \neq \mathbf{0}$. Es ist zu zeigen, dass $f = \mathbf{0}$ sein muss.

Wegen $g \neq \mathbf{0}$ ist mindestens eines der b_k von Null verschieden. Deshalb ist die Definition

$$m \;=\; \min\{k \geq 0 : \; b_k \neq 0\}$$

sinnvoll. Dann gilt aber die Beziehung

$$c_m \;=\; a_0 b_m \,+\, \underbrace{a_1 b_{m-1} \,+\, \ldots \,+\, a_m b_0}_{=\,0} \;=\; 0.$$

Da $b_m \neq 0$ ist, muss $a_0 = 0$ sein. Es folgt

$$c_{m+1} \;=\; \underbrace{a_0 b_{m+1}}_{=\,0} + a_1 b_m \,+\, \underbrace{a_2 b_{m-1} \,+\, \ldots \,+\, a_{m+1} b_0}_{=\,0} \;=\; 0,$$

wir erhalten aus dieser Gleichung $a_1 b_m = 0$. Es folgt $a_1 = 0$. Wir können dieses Verfahren in induktiver Weise fortführen und erhalten so

$$a_k = 0 \qquad \forall \, k = 0, 1, 2, \dots$$

Also muss $f = \mathbf{0}$ sein.

Zu 5.
Wir zeigen, dass das Polynom $X = (0,\, 1,\, 0,\, 0,\, \dots)$ in $R[X]$ kein Inverses besitzt. Sei dazu $X \cdot g = \mathbf{1} = (1, 0, 0, \dots)$, also g das Inverse zu X,

$$\begin{aligned} X &= (a_0,\, a_1,\, a_2,\, \dots) = (0,\, 1,\, 0,\, 0,\, \cdots), \\ g &= (b_0,\, b_1,\, b_2,\, \dots). \end{aligned}$$

Dann folgt durch Koeffizientenvergleich in $X \cdot g = \mathbf{1}$ die Beziehung

$$a_0 b_0 = 1 \implies 0 \cdot b_0 = 0 = 1,$$

was ein Widerspruch ist. □

1.12 Definition (Teiler, Vielfaches, irred. Polynom)
Sei $(R,\, +,\, \cdot)$ (und damit auch $(R[X],\, +,\, \cdot)$) ein Ring mit Einselement. Seien $f,\, g \in R[X]$. Wir sagen, das Polynom g teilt das Polynom f, falls ein $h \in R[X]$ existiert mit

$$f = g \cdot h.$$

Das Polynom g heißt dann ein Teiler von f. Weiters heißt f ein Vielfaches von g, Schreibweise: $g \mid f$ Das Polynom g heißt ein echter Teiler von f, wenn

$$0 < \deg g < \deg f.$$

Ein Polynom f heißt irreduzibel, wenn es keine echten Teiler besitzt. Andernfalls heißt f reduzibel.

1.13 Lemma
Sei $(R,\, +,\, \cdot)$ ein Ring und $\mathbf{0}$ das Nullpolynom in $R[X]$. Wir definieren

$$\deg(\mathbf{0}) = -\infty$$

und rechnen mit dem Symbol $-\infty$ wie üblich, also

$$\begin{aligned} -\infty + n &= -\infty \qquad \forall\, n \in \mathbb{N} \cup \{0\}, \\ -\infty + -\infty &= -\infty. \end{aligned}$$

Dann gilt

1. $\deg(f \cdot g) \leq \deg f + \deg g$ für alle $f,\, g \in R[X]$.
2. Wenn das Produkt der Leitkoeffizienten von f und g von Null verschieden ist, dann gilt sogar
$$\deg(f \cdot g) = \deg f + \deg g.$$

Beweis.
Sei $f = (a_0,\, a_1,\, \dots,\, a_n)$ mit $a_n \neq 0$, also $\deg f = n$. Sei $g = (b_0,\, b_1,\, \dots,\, b_m)$ mit $b_m \neq 0$, also $\deg g = m$. Dann gilt

$$f \cdot g = (c_0,\, c_1,\, c_2,\, \dots) \quad \text{mit } c_k = \sum_{i=0}^{k} a_i\, b_{k-i}.$$

Wie man leicht nachrechnet, gilt $c_{m+n} = a_n \cdot b_m$, für alle weiteren Koeffizienten folgt $c_k = 0$ für alle $k > m + n$. Daraus folgen die beiden Behauptungen. □

1.14 Korollar (Gradregel)
Falls $(R, +, \cdot)$ ein Integritätsbereich ist und f und g zwei Polynome aus $R[X]$ sind, so folgt die Beziehung

$$\deg f \cdot g \;=\; \deg f \;+\; \deg g.$$

1.15 Satz (Division mit Rest)
Sei $(R, +, \cdot)$ ein Ring mit Einselement. Seien $f, g \in R[X]$,

$$f \;=\; \sum_{i=0}^{n} a_i X^i \quad \text{mit } a_n \neq 0$$
$$g \;=\; \sum_{i=0}^{m} b_i X^i \quad \text{mit } b_m = 1.$$

Dann gibt es zwei Polynome $q, r \in R[X]$ mit den Eigenschaften

- $f \;=\; q \cdot g \;+\; r$,
- $r \;=\; \mathbf{0}$ oder $\deg r \;<\; \deg g$,
- r und q sind eindeutig bestimmt.

Beweis. Wir beweisen zuerst die Existenz von q und r. Falls $\deg f < \deg g$ gilt, dann existiert trivialerweise eine derartige Darstellung: $f \;=\; \mathbf{0} \cdot g + f$. Sei deshalb $\deg f \geq \deg g$ und sei

$$f_1 \;=\; f \;-\; a_n X^{n-m} g.$$

Wegen $b_m = 1$ folgt daraus

$$k_1 \;=\; \deg f_1 \;<\; \deg f.$$

Für den Fall, dass $k_1 < m = \deg g$ ist, sind wir schon fertig, denn dann gilt die Beziehung

$$f \;=\; \underbrace{f_1}_{r} \;+\; \underbrace{a_n X^{n-m}}_{q} g.$$

Falls aber $k_1 \geq m$ sein sollte, dann betrachten wir das Polynom

$$f_1 = c_0 + c_1 X + \cdots + c_{k_1} X^{k_1}$$

und verwenden den gleichen Trick ein zweites Mal. Wir definieren

$$f_2 \;=\; f_1 \;-\; c_{k_1} X^{k_1 - m} g,$$

dessen Grad wieder kleiner ist als der von f_1, dann

$$f_3 \;=\; f_2 \;-\; c_{k_2} X^{k_2 - m} g, \quad \text{usw. } \ldots$$

Auf diese Weise erhalten wir eine Kette

$$n \;>\; k_1 \;>\; k_2 \;>\; \cdots .$$

Nach endlich vielen Schritten erhalten wir ein Polynom f_j mit $\deg f_j < m$. Das ursprüngliche Polynom f lässt sich schreiben als

$$f = \underbrace{\big(a_n X^{n-m} + c_{k_1} X^{k_1-m} + \cdots + c_{k_{j-1}} X^{k_{j-1}-m}\big)}_{q} \cdot g \;+\; \underbrace{f_j}_{r}.$$

Zur Eindeutigkeit.
Angenommen es gäbe Konstanten q, r und q', r' mit

$$f = q \cdot g \;+ r \quad \text{und} \quad \deg r \;< \deg g \tag{1.1}$$
$$f = q' \cdot g \;+ r' \quad \text{und} \quad \deg r' < \deg g. \tag{1.2}$$

Aus den beiden Bedingungen $\deg r$, $\deg r' < \deg g$ folgt

$$\deg(r - r') \;<\; \deg g.$$

Durch Subtraktion der Zeile (1.2) von der Zeile (1.1) erhalten wir

$$(q - q') \cdot g \;=\; r - r'.$$

Angenommen $q - q' \neq \mathbf{0}$, Dann gilt

$$\deg\big((q - q') \cdot g\big) \;=\; \deg\big(r - r'\big).$$

Angenommen $q - q' \neq \mathbf{0}$. Das heißt, dass der Leitkoeffizient von $q - q'$ ungleich 0 ist. Da der Leitkoeffizient von g gleich 1 ist, wäre daher der Leitkoeffizient von $(q - q') \cdot g$ ungleich 0.

Nach Lemma 1.13(2) gilt dann

$$\deg\big(r - r'\big) \;=\; \deg\big((q - q') \cdot g\big) \;=\; \deg(q - q') + \deg g \geq \deg g,$$

was aber ein Widerspruch zu

$$\deg(r - r') \;<\; \deg g.$$

ist. Somit gilt $q = q'$. Es folgt $r = r'$. □

1.16 Korollar
Wenn $(R, +, \cdot)$ ein Körper ist, dann reicht in Satz 1.15 die Voraussetzung $b_m \neq 0$ aus. Die Bedingung $b_m = 1$ ist in diesem Fall unnötig.

1.17 Definition (Polynomfunktion)
Sei $(R, +, \cdot)$ ein Ring und sei $(R[X], +, \cdot)$ der Polynomring über R. Unter der Polynomfunktion $\overline{f}$ zum Polynom $f = (a_0, a_1, \ldots, a_n) \in R[X]$ verstehen wir die Funktion

$$\begin{aligned} \overline{f}: \quad R \quad &\to \quad R, \\ \overline{f}(x) \;&=\; a_0 \;+\; a_1 x \;+\; \ldots \;+\; a_n x^n, \qquad x \in R. \end{aligned}$$

1.18 Beispiel Sei $(R, +, \cdot) = (\mathbb{Z}_2, +, \cdot)$, sei $f = (0, 1, 1) = X + X^2$ und sei $g = \mathbf{0} = (0, \ldots, 0)$. Dann gilt offensichtlich $f \neq g$. Betrachten wir aber die zugeordneten Polynomfunktionen, so sehen wir, dass $\overline{f} = \overline{g} = \overline{\mathbf{0}}$:

$$\overline{f}(x) \;=\; x \,+\, x^2 \;=\; 0 \qquad \forall x \in R = \mathbb{Z}_2.$$

Zwei verschiedenen Polynomen kann also die gleiche Polynomfunktion zugeordnet sein.

1.19 Frage Für welche Ringe gilt

$$f \;=\; g \quad \Leftrightarrow \quad \overline{f} \;=\; \overline{g} \quad ?$$

1.20 Lemma Sei $(R, +, \cdot)$ ein kommutativer Ring, seien $f, g \in R[X]$, und sei a ein Element aus R. Dann folgt

1. $\overline{f+g} \;=\; \overline{f} + \overline{g}$
2. $\overline{f \cdot g} \;=\; \overline{f} \cdot \overline{g}$
3. $\overline{af} \;=\; a\overline{f}$

Beweis. Seien $f \;=\; (a_0, a_1, \ldots, a_n, \ldots)$ und $g \;=\; (b_0, b_1, \ldots, b_m, \ldots)$ Elemente von $R[X]$.

Zu 1.
Es gilt $f + g \;=\; (a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots)$ und daher

$$\overline{f+g}(x) \;=\; (a_0 + b_0) \,+\, (a_1 + b_1)x + \ldots \;=\; \sum_{j=0}^{\infty} (a_j + b_j)x^j \;=$$

$$= \sum_{j=0}^{\infty} a_j x^j \,+\, \sum_{j=0}^{\infty} b_j x^j \;=\; \overline{f}(x) \,+\, \overline{g}(x).$$

Zu 2.
Nach Definition des Produktes zweier Polynome gilt

$$f \cdot g = (c_0, c_1, c_2, \ldots) \quad \text{mit } c_n = \sum_{k=0}^{n} a_k b_{n-k},\; n = 0, 1, 2, \ldots$$

Daraus folgt

$$\overline{f \cdot g}(x) \;=\; \sum_{k=0}^{0} a_k b_{0-k} \,+\, x \sum_{k=0}^{1} a_k b_{1-k} \,+\, x^2 \sum_{k=0}^{2} a_k b_{2-k} \,+\, \ldots \;=$$

$$= \sum_{j=0}^{\infty} x^j \sum_{k=0}^{j} a_k b_{j-k} \;=\; \sum_{k=0}^{\infty} a_k x^k \,\cdot\, \sum_{k=0}^{\infty} b_k x^k \;=\; \overline{f}(x) \cdot \overline{g}(x).$$

Zu 3.
af ist eine Kurzschreibweise für

$$(a, 0, 0, \ldots) \cdot (a_0, a_1, \ldots, a_n, 0, \ldots).$$

Damit folgt Punkt 3. aus Punkt 2. als Spezialfall. □

1.21 Definition (Nullstelle)
Sei $(R, +, \cdot)$ ein Ring und $f \in R[X]$. Ein Element $a \in R$ heißt eine Nullstelle von f, wenn $\overline{f}(a) = 0$.

1.22 Satz Sei $(R, +, \cdot)$ ein kommutativer Ring mit Einselement, $f \in R[X]$ und a eine Nullstelle von f. Dann gibt es ein g in $R[X]$ mit

$$f = (X - a) \cdot g.$$

Beweis. Falls $f = \mathbf{0}$, dann ist $g = \mathbf{0}$ eine passende Wahl. Dies ist der triviale Fall.

Sei also $f \neq \mathbf{0}$. Wäre $\deg f = 0$, dann wäre $f = (a_0, 0, 0, \dots)$ mit $a_0 \neq 0$. Damit wäre $\overline{f}(a) = a_0 \neq 0$. Dies ist aber ein Widerspruch zur Voraussetzung, dass a eine Nullstelle von f ist. Es muss also $\deg f \geq 1$ gelten. Nach Satz 1.15 (Division mit Rest) gibt es eindeutig bestimmte Polynome q und r mit

$$f = q \cdot (X - a) + r,$$

wobei entweder $r = \mathbf{0}$ oder $\deg r < \deg(X - a) = 1$. Es folgt $\deg r = 0$ oder $r = \mathbf{0}$, also gilt in jedem Fall $r = (b, 0, 0, \dots)$ mit einem $b \in R$. Wir erhalten

$$\overline{f} = \overline{q} \cdot \overline{X - a} + b,$$

also

$$\overline{f}(x) = \overline{q}(x) \cdot (x - a) + b.$$

Wegen $\overline{f}(a) = 0$ folgt daraus, dass $b = 0$ sein muss. Somit gilt $r = \mathbf{0}$. □

1.23 Satz
Sei $(R, +, \cdot)$ ein Integritätsbereich und sei $f \in R[X]$, $f \neq \mathbf{0}$. Dann besitzt f in R höchstens $\deg f$ Nullstellen.

Beweis. Falls $\deg f = 0$ gilt, dann besitzt f keine Nullstelle in R. Für diesen Fall stimmt die Behauptung also trivialerweise. Falls $\deg f \in \mathbb{N}$ gilt, so führen wir einen induktiven Beweis nach $n = \deg f$.

Induktionsanfang:
Sei $n = 1$. Dann hat f die Form $f = aX + b$ mit $a, b \in R$, $a \neq 0$. Also ist $\overline{f}(x) = ax + b$. Die Abbildung $\overline{f}$ ist injektiv, wie man leicht nachprüft. Um die Nullstellen von f zu finden, suchen wir alle $x \in R$, für die gilt

$$ax + b = 0.$$

Aus der Injektivität von $\overline{f}$ folgt, dass jede Nullstelle von f eindeutig bestimmt ist, wenn sie existiert. Daher f besitzt höchstens eine Nullstelle.

Induktionsvoraussetzung:
Jedes Polynom $g \in R[X]$ mit $g \leq \mathbf{0}$, mit Grad $k \neq n$ besitzt höchstens k Nullstellen in R.

Induktionsschritt:
Sei nun $f \in R[X]$ mit $\deg f = n+1$. Wenn f keine Nullstelle in R besitzt, dann sind wir bereits fertig. Wenn f eine Nullstelle a in R besitzt, dann folgt nach Satz 1.22 $f = (X-a) \cdot g$, mit $g \in R[X]$, $g \neq \mathbf{0}$. Weiters gilt wegen Korollar 1.14 die Beziehung $\deg g = n$, denn wir rechnen in einem Integritätsbereich. Es folgt

$$\overline{f}(x) \;=\; \overline{X-a} \cdot \overline{g}(x) \;=\; (x-a) \cdot \overline{g}(x).$$

Daher ist jede Nullstelle von f, die von a verschieden ist, auch eine Nullstelle von g. Nach Induktionsvoraussetzung besitzt g höchstens n Nullstellen. Somit besitzt f höchstens $n+1$ Nullstellen. □

1.24 Korollar
Sei $(R, +, \cdot)$ ein Integritätsbereich mit unendlich vielen Elementen. Dann gilt für alle Polynome $f, g \in R[X]$ die Beziehung

$$f \;=\; g \quad \Leftrightarrow \quad \overline{f} \;=\; \overline{g}.$$

Beweis.
Zu ($\Rightarrow$) Trivial.

Zu ($\Leftarrow$)
Aus $\overline{f} = \overline{g}$ folgt

$$\overline{f}(x) = \overline{g}(x) \qquad \forall\, x \in R.$$

Daraus folgt weiters

$$\overline{f}(x) - \overline{g}(x) = 0 \quad \forall x \in R \quad \Rightarrow \quad \overline{f-g}(x) = 0 \quad \forall x \in R.$$

Das Polynom $f-g$ besitzt also unendlich viele Nullstellen. Nach Satz 1.23 muss das Polynom $f-g$ daher das Nullpolynom $\mathbf{0}$ sein. □

Auf Grund dieses Korollars werden wir im Folgenden für die Fälle $R = \mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ nicht mehr so streng zwischen Polynomen und Polynomfunktionen unterscheiden.

Wir haben nun geklärt, wann die Gleichheit von Polynomen gleichbedeutend ist mit der Gleichheit der zugehörigen Polynomfunktionen. Als Nächstes untersuchen wir, wann ein Polynom irreduzibel ist.

1.25 Bemerkung
Sei $(R, +, \cdot)$ ein Integritätsbereich. Dann sind alle Polynome über R vom Grad Eins irreduzibel.

1.26 Satz (Fundamentalsatz der Algebra)
Jedes nichtkonstante Polynom $f \in \mathbb{C}[X]$ besitzt mindestens eine Nullstelle in $\mathbb{C}$.

Für den Beweis des Fundamentalsatzes der Algebra verweisen wir auf die Literatur.

1.2.3 Zur Arithmetik in Endlichen Körpern

Sei $m \in \mathbb{N}$ gegeben, bezeichne $\mathcal{A} = \{0, 1\}$ und sei $x \in \mathcal{A}^m$. Welche injektiven (und damit notwendigerweise bijektiven) Funktionen von $\mathcal{A}^m$ in sich sind für die Kryptographie geeignet?

Damit wir solche Funktionen überhaupt konstruieren können, benötigen wir auf $\mathcal{A}^m$ arithmetische Operationen, zum Beispiel eine Addition und eine Multiplikation von Elementen von $\mathcal{A}^m$.

Wie definieren wir die Addition und die Multiplikation von Binärstrings der Länge m und welche Eigenschaften fordern wir von diesen Operationen?

Wenn wir mit binären Wörtern rechnen könnten wie mit rationalen Zahlen oder wie mit Restklassen modulo einer Primzahl, dann stünde uns die (mächtige!) Körpertheorie zur Verfügung. Wir müssen also in $\mathcal{A}^m$ zwei binäre Operationen zur Verfügung haben, eine Addition und eine Multiplikation.

Die Elemente von $\mathcal{A}$ heißen Bits und werden mit 0 und 1 bezeichnet. Auf $\mathcal{A}$ kann man eine Addition definieren, bezüglich der $(\mathcal{A}, +)$ eine abelsche Gruppe ist: $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, $1 + 1 = 0$. In der Informatik spricht man von der XOR-Verknüpfung der Bits. Aus der Algebra weiß man, dass es auf $\mathcal{A}$ auch eine Multiplikation gibt, bezüglich der $(\mathcal{A}, +, \cdot)$ einen Körper bildet. Es handelt sich um den Körper $\mathbb{Z}_2$ der Restklassen modulo 2. In der Algebra ist es üblich, den bis auf Isomorphie eindeutig bestimmten Körper mit zwei Elementen mit dem Symbol $\mathbb{F}_2$ oder GF(2) zu bezeichnen.

Wie addiert und multipliziert man nun zwei Binärstrings $a = (a_0, a_1, \ldots, a_{m-1})$ und $b = (b_0, b_1, \ldots, b_{m-1})$, $a, b \in \mathcal{A}^m$? Das Konzept für die Addition ist einfach. Wir gehen von der Gruppe $(\mathcal{A}, +)$ aus und betrachten das m-fache direkte (gruppentheoretische) Produkt,

$$\mathcal{A}^m = \underbrace{\mathcal{A} \times \mathcal{A} \cdots \times \mathcal{A}}_{m \text{ Faktoren}}.$$

Dann ist $(\mathcal{A}^m, +)$ eine abelsche Gruppe. Zwei Elemente $a = (a_0, a_1, \ldots, a_{m-1})$ und $b = (b_0, b_1, \ldots, b_{m-1})$ von $\mathcal{A}^m$ werden wie folgt addiert:

$$\begin{aligned} & \underbrace{(a_0, a_1, \ldots, a_{m-1})}_{a} \\ + \quad & \underbrace{(b_0, b_1, \ldots, b_{m-1})}_{b} \\ = \quad & \underbrace{(a_0 + b_0, a_1 + b_1, \ldots, a_{m-1} + b_{m-1})}_{a+b}. \end{aligned}$$

Es ist klar, dass die oben definierte Addition nichts Anderes als das in der Informatik wohlbekannte XOR der beiden Binärstrings a und b ist.

Wie sollen wir nun das Produkt $a \cdot b$ von zwei Binärstrings a und b erklären? Wir interpretieren den String $a = (a_0, a_1, \ldots, a_{m-1})$ als das Polynom

$$a = a_0 + a_1 X + \cdots + a_{m-1} X^{m-1}$$

über dem Körper $(\mathbb{F}_2, +, \cdot)$. Ebenso interpretieren wir $b = (b_0, b_1, \ldots, b_{m-1})$,

$$b = b_0 + b_1 X + \cdots + b_{m-1} X^{m-1}.$$

Dann ist das Produkt der beiden Polynome a und b definiert und es gilt nach den Rechenregeln für Polynome in $\mathbb{F}_2[X]$ die Beziehung

$$a \cdot b \;=\; a_0 b_0 \;+\; (a_1 b_0 + a_0 b_1)X \;+\; \ldots \;+\; a_{m-1} b_{m-1} X^{2m-2}.$$

Diesem Produktpolynom entspricht der Binärstring

$$(a_0 b_0,\; a_1 b_0 + a_0 b_1, \ldots,\; a_{m-1} b_0 + \cdots + a_0 b_{m-1}, \ldots,\; a_{m-1} b_{m-1}).$$

Wir haben mit diesem Trick, die beiden Binärstrings a und b als Polynome zu interpretieren und diese Polynome miteinander zu multiplizieren, einen Binärstring der Länge $2m-1$ erhalten. Unser Ziel ist es aber, das Produkt von a und b so zu definieren, dass sich wieder ein Binärstring der Länge m ergibt. Wir möchten ja eine Multiplikation auf $\mathcal{A}^m$ erhalten.

Die Lösung dieser Aufgabe ist einfach und dem modularen Rechnen mit ganzen Zahlen abgeschaut. Sei $\mathbb{F}_p$ der Körper mit p Elementen, p eine Primzahl. $\mathbb{F}_p$ ist bis auf Isomorphie eindeutig bestimmt, insbesondere ist der Restklassenkörper $\mathbb{Z}_p$ isomorph zu $\mathbb{F}_p$. Wir können daher $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$ schreiben. Das Rechnen in $\mathbb{F}_p$ ist wegen der Isomorphie $\mathbb{F}_p \cong \mathbb{Z}_p$ sehr einfach, $a + b = a + b \pmod p$, $a \cdot b = a \cdot b \pmod p$. Wie berechnen wir dann zum Beispiel die ganze Zahl $a \cdot b \pmod p$? Sei zum Beispiel $p = 11$, $a = 5$ und $b = 7$. Wir müssen das Produkt $a \cdot b$ modulo der Zahl p reduzieren, im Beispiel also die Zahl $35 \pmod{11} = 2$ berechnen. Wie führen wir diese Reduktion konkret durch? Dazu benützen wir die Division mit Rest, wir dividieren die ganze Zahl ab durch den Modul p. Der auftretende Rest r ergibt dann die Lösung.

Die gleiche Idee können wir bei der Reduktion von Polynomen durchführen. Auch in $\mathbb{F}_2[X]$ steht uns die Division mit Rest zur Verfügung. Wir reduzieren also das Produktpolynom $a \cdot b$ modulo einem irreduziblen Polynom vom Grad m, mittels der Division mit Rest. Sei dazu

$$f = t_0 + t_1 X + \cdots + t_m X^m$$

ein fest gewähltes irreduzibles Polynom aus $\mathbb{F}_2[X]$ vom Grad m. Dann existiert ein eindeutig bestimmtes Polynom $c \in \mathbb{F}_2[X]$ vom Grad kleiner oder gleich $m-1$, sodass gilt:

$$a \cdot b \equiv c \pmod f.$$

Diesen Rest c des Polynoms $a \cdot b$ bei der Division durch f definieren wir als das Produkt der beiden Polynome a und b. Da das Polynom c einen Grad kleiner oder gleich $m-1$ besitzt, hat es die Gestalt

$$c \;=\; c_0 + c_1 X + \cdots + c_{m-1} X^{m-1}.$$

Es entspricht also dem Binärstring

$$c \;=\; (c_0,\; c_1,\; \ldots,\; c_{m-1}).$$

Dieser Binärstring ist das gesuchte Produkt der beiden Binärstrings a und b.

An dieser Stelle eine Erinnerung: unter einem Integritätsbereich versteht man einen nullteilerfreien, kommutativen Ring $(R, +, \cdot)$ mit Einselement. Die Menge $R[X]$ der Polynome über dem Integritätsbereich R bildet selbst wieder einen

Integritätsbereich. Daher ist speziell für jeden Körper K der Polynomring $K[X]$ ein Integritätsbereich, denn jeder Körper ist ein Integritätsbereich. Ein irreduzibles Element eines Polynomringes $R[X]$ nennt man ein irreduzibles Polynom. Irreduzible Polynome sind daher nur in triviale Faktoren zerlegbar, in Einheiten und in assoziierte Elemente. Was sind die Einheiten des Polynomringes $K[X]$ über dem Körper K? Da jedes Element von K^* ein multiplikatives Inverses besitzt, sind die Einheiten von $K[X]$ genau die Elemente von K^*, also die Elemente ungleich 0 des Grundkörpers K. Die zu einem gegebenen Polynom $f \in K[X]$ assoziierten Elementen sind daher die Polynome $a \cdot f$, $a \in K, a \neq 0$. Im Fall des Körpers $K = \mathbb{Z}_2$ ist die Situation besonders einfach, da es nur ein einziges Körperelement ungleich 0 gibt, das Einselement 1. Daher gibt es nur eine Einheit und damit auch nur ein einziges assoziiertes Polynom, nämlich das ursprüngliche Polynom.

1.27 Bemerkung (Zusammenfassung)
Das Produkt $a \cdot b$ zweier Binärstrings a und b der Länge m wird als jener Binärstring c der Länge m definiert, der sich ergibt, wenn man die beiden durch a und b definierten Polynome miteinander multipliziert und das Produktpolynom modulo einem gegebenen irreduziblen Polynom vom Grad m reduziert.

Wenn wir auf $\mathcal{A}^m$ eine Multiplikation definieren wollen, dann müssen wir als Erstes das irreduzible Polynom f vom Grad m wählen. Die Irreduzibilität des Polynomes f ist eine wesentliche Voraussetzung, auf die nicht verzichtet werden kann. Das Interessante an dieser Vorgangsweise ist die Tatsache, dass aus der Irreduzibilität von f die Körpereigenschaft für $(\mathcal{A}^m, +, \cdot)$ folgt.

Man kann zeigen:

- $(\mathcal{A}^m, +, \cdot)$ ist ein Körper mit 2^m Elementen.
- Jeder Körper mit 2^m Elementen ist zu diesem Körper isomorph.
- Der (bis auf Isomorphie eindeutige) Körper mit 2^m Elementen wird mit $\mathrm{GF}(2^m)$ oder mit $\mathbb{F}_{2^m}$ bezeichnet. Beachten Sie in diesem Zusammenhang, dass mit $\mathbb{F}_2^m$ üblicherweise die *additive Gruppe* $(\mathbb{F}_2^m, +)$ gemeint ist. In Gegensatz dazu bezeichnet $\mathbb{F}_{2^m}$ den *Körper* mit 2^m Elementen.
- Eine andere Wahl des irreduziblen Polynoms führt zu einem isomorphen Körper.
- Die Einschränkung auf $\mathcal{A}$ ist hier unwesentlich, wir können diese Art von Arithmetik über jedem Grundkörper $\mathbb{F}_p$, p prim, definieren.

Das Standardwerk zu dieser "Theorie der endlichen Körper" ist das Buch

R. Lidl and H. Niederreiter. *Finite Fields.* Addison-Wesley, 1983.

Dieses Buch enthält nicht nur die gesamte Theorie, sondern auch umfangreiche Tabellen irreduzibler Polynome.

1.3 DES

1.28 Bemerkung (Grundsätzliches)
Die bekannteste Blockchiffre ist der *Data Encryption Standard* (DES). DES ist aus mehreren Gründen bedeutsam. Zum ersten Mal in der Geschichte der Krytographie wurde ein Chiffrierverfahren im Rahmen einer öffentlichen Ausschreibung ausgewählt und dann in Form eines Industriestandards im Detail veröffentlicht. Weiters waren die Konstruktionsprinzipien von DES wegweisend und beim Versuch DES zu knacken wurden die bis heute wirkungsvollsten Attacken auf Blockchiffrierverfahren entwickelt.

DES wird in zahlreicher Sicherheitssoftware verwendet, oft in der sogenannten Triple-DES Variante, die bis heute aktuell ist. DES selbst ist auf Grund der schnellen modernen Computer für wichtige Anwendungen nicht mehr sicher genug.

1.29 Bemerkung (Geschichte)
Bis zum Beginn der 70er-Jahre war Kryptographie fast ausschließlich ein Werkzeug der Militärs. Die rasche Entwicklung der elektronischen Datenverarbeitung in kommerziellen Bereichen wie im Bankenwesen erforderte zu diesem Zeitpunkt erstmals auch für den zivilen Einsatz ein effizientes und sicheres Verfahren zur Verschlüsselung und zur Authentifikation. Das Verfahren sollte nichts mit dem militärischen Bereich zu tun haben, es war daher eine Neuentwicklung notwendig. Das National Bureau of Standards (NBS, jetzt NIST), eine Unterbehörde des US-Wirtschaftsministeriums, veröffentlichte im Mai 1975 eine Ausschreibung für ein kryptographisches Verfahren zur Verschlüsselung von Daten. Die Beteiligung an dieser Ausschreibung war gering, ausgewählt wurde der einzige ernsthafte Kandidat, ein Vorschlag von IBM namens LUCIFER. Nach einigen Modifikationen durch die NSA wurde das modifizierte Verfahren unter dem Namen Data Encryption Standard (DES) im Jänner 1977 zu einem nationalen Standard.

Die Modifikationen der NSA waren der Anlaß für eine intensive Debatte zur Sicherheit von DES. Zwar wurde der DES-Algorithmus im Detail offengelegt, ganz im Sinne des Kerkhoffschen Prinzips, aber die genauen Designkriterien hinter DES blieben geheim. Es wurde immer wieder vermutet, dass die NSA eine "Hintertür" eingebaut hat, die ihr ein rascheres Knacken von DES ermöglicht. Es gibt bis heute keine Bestätigung dieser Hypothese. Tatsache ist, dass die NSA auf einer Reduktion der Schlüssellänge von 128 Bit bei LUCIFER auf 56 Bit bei DES bestand. Eine brute-force Attacke auf DES ist daher wesentlich leichter durchzuführen als auf den ursprünglichen Algorithmus von IBM. Dies war der am stärksten kritisierte Aspekt bei der Modifikation von LUCIFER durch die NSA.

Im Laufe der Jahre stellte sich heraus, dass die Entwickler von DES beim Entwurf von DES einige Attacken bereits berücksichtigt hatten, die erst Jahre später von der wissenschaftlichen Öffentlichkeit entdeckt wurden. Die bekannteste Attacke ist hier die *differentielle Kryptanalyse* von Biham und Shamir (siehe dazu Stinson [Sti06]).

1.30 Bemerkung (Notation)

In der internationalen Literatur zur angewandten Kryptographie ist es üblich, die Addition auf $\mathbb{F}_2 = \{0, 1\}$ und auf $\mathbb{F}_2^n$ mit dem Symbol $\oplus$ zu bezeichnen. Wir werden uns dieser Konvention anschließen. Weiters werden wir im Folgenden die linke Hälfte eines binären Wortes w mit $L(w)$ und die rechte Hälfte mit $R(w)$ bezeichen und das Wort w dann, je nach Zusammenhang, in der Form $w = L(w)R(w)$ oder auch als $w = (L(w), R(w))$ schreiben.

1.31 Proposition Sei $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^{2n}$ und $\mathcal{K}$ beliebig ($\mathcal{K} \neq \emptyset$). Für jedes $k \in \mathcal{K}$ sei eine Abbildung $f_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$ definiert. Für jedes Klartextelement $p \in \mathcal{P}$ sei $L = L(p)$ die linke Hälfte der Bits von p und $R = R(p)$ die rechte Hälfte der Bits von p, also $L, R \in \mathbb{F}^n$ und $p = LR$. Dann ist für jedes $k \in \mathcal{K}$ die folgende Abbildung

$$\begin{aligned} &e_k : \mathcal{P} \to \mathcal{C}, \\ &e_k(p) = (R(p), L(p) \oplus f_k(R(p))) , \end{aligned}$$

eine bijektive Abbildung.

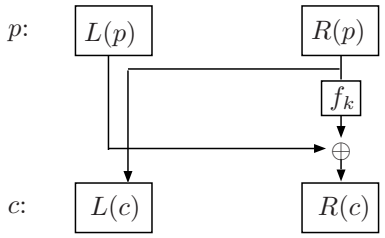


Abbildung 1.1: Das Schema der Abbildung

Beweis.
Die beiden Eigenschaften, die hier nachzuweisen sind, also die Injektivität und die Surjektivität, ergeben sich leicht aus dem Schema 1.1. Zum Nachweis der Injektivität nehmen wir an, dass $e_k(p) = e_k(p')$ gilt.

$$\begin{aligned} &\Rightarrow \quad R(p) = R(p') \quad \text{und} \quad L(p) \oplus f_k(R(p)) = L(p') \oplus f_k(R(p')) \\ &\Rightarrow \quad L(p) = L(p') \\ &\Rightarrow \quad p = p' \end{aligned}$$

Aus der Injektivitität von e_k folgt bereits die Surjektivität, da $\mathbb{F}_2^{2n}$ endlich ist. Zur Übung rechnen wir trotzdem die Surjektivität nach. Sei dazu $c = L(c)R(c) \in \mathcal{C}$ gegeben. Dann gilt

$$c = e_k(R(c) \oplus f_k(L(c)), L(c)),$$

denn offensichtlich gilt der Schluß

$$R(c) = x \oplus f_k(L(c)) \quad \Rightarrow \quad x = R(c) \oplus f_k(L(c)).$$

□

1.32 Bemerkung

1. Für die Bijektivität der Funktion $e_k : \mathbb{F}_2^{2n} \to \mathbb{F}_2^{2n}$ war für die Funktion f_k keine zusätzliche Voraussetzung notwendig, wie etwa die Bijektivität. Wir sind in der Wahl von f_k also völlig frei.

2. Die Funktion e_k ist noch keine geeignete Verschlüsselungsfunktion, denn der rechte Klartextblock $R(p)$ bleibt ja unverändert. Wenn wir aber die Funktion e_k iterieren, dann ist es vorstellbar, dass wir auf diese Weise die Bits gut durchmischen, sofern f_k passend gewählt ist.

3. Die Nichtlinearität von e_k hängt eng mit der Nichtlinearität von f_k zusammen (siehe Proposition 1.33).

1.33 Proposition Die Abbildung

$$\begin{aligned} &e_k : \mathbb{F}_2^{2n} \to \mathbb{F}_2^{2n} \\ &e_k(p) = (R(p), L(p) \oplus f_k(R(p))) \end{aligned}$$

ist genau dann linear, wenn $f_k : \mathbb{F}_2^n \to \mathbb{F}_2^n$ linear ist.

Beweis.
Eine Funktion $f : \mathbb{F}_2^t \to \mathbb{F}_2^t, x \mapsto f(x)$, heißt linear, wenn gilt:

$$\forall\, x, y \in \mathbb{F}_2^t : \quad f(x \oplus y) = f(x) \oplus f(y).$$

Es reicht hier aus, die Additivität von f zu fordern, die Bedingung $\lambda \cdot f(x) = f(\lambda \cdot x)$ ist dann für alle $\lambda \in \mathbb{F}_2$ trivialerweise erfüllt.

Zur Äquivalenz der Linearität von e_k und f_k überlegen wir Folgendes. Wir schreiben für $p, p' \in \mathcal{P} = \mathbb{F}_2^{2n}$ kurz $R = R(p), R' = R(p')$. Dann gilt

$$\begin{aligned} e_k \text{ linear} &\Leftrightarrow \forall\, p, p' : e_k(p \oplus p') = e_k(p) \oplus e_k(p') \\ &\Leftrightarrow \forall\, p, p' : f_k(R \oplus R') = f_k(R) \oplus f_k(R') \\ &\Leftrightarrow f_k \text{ linear}. \end{aligned}$$

□

1.34 Definition (Feistel-Chiffre)
Seien r, n und m natürliche Zahlen größer gleich 2. Unter einer Feistel-Chiffre verstehen wir eine Blockchiffre $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ der folgenden Gestalt:

1. $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^{2n}$, $\mathcal{K} = \mathbb{F}_2^m$.

2. Jedem Schlüssel $k \in \mathcal{K}$ werden r sogenannte Rundenschlüssel $k_1, \ldots, k_r$ zugeordnet.

3. Jedem Rundenschlüssel k_i wird eine Funktion $f_{k_i} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ zugeordnet.

4. Die Verschlüsselungsfunktionen $e_k \in \mathcal{E}$ haben die folgende Gestalt: Für jeden Klartextblock $p = (L_0, R_0) \in \mathcal{P}$ wird in r Schritten (Runden genannt) das zugehörige Geheimtextelement $c \in \mathcal{C}$ nach folgender Vorschrift berechnet:

$$\forall\, i, 1 \leq i \leq r: \quad (L_i, R_i) = (R_{i-1}, L_{i-1} \oplus f_{k_i}(R_{i-1}))$$

Das Geheimtextelement $c = e_k(p)$ ist definiert durch

$$c = e_k(p) = (R_r, L_r).$$

$$p = L_0R_0 \xrightarrow{k_1} L_1R_1 \xrightarrow{k_2} L_2R_2 \xrightarrow{k_3} \ldots \xrightarrow{k_r} L_rR_r \overset{\text{Vertau-schung}}{\longrightarrow} c = R_rL_r$$

1.35 Bemerkung (Entschlüsselung)
Bei einer Feistel-Chiffre erfolgt die Entschlüsselung in umgekehrter Reihenfolge zur Verschlüsselung. Die Vertauschung der beiden Hälften L_r und R_r im letzten Schritt dient dazu, den Algorithmus direkt auf das Chiffrat c anwenden zu können. Der einzige Unterschied zwischen Ver- und Entschlüsselung ist die Reihenfolge, in der die Rundenschlüssel k_i angewendet werden. Damit ist die Feistel-Chiffre für die Implementierung in einem Chip besonders geeignet. Dies war auch die Absicht der Konstrukteure von DES, dem wichtigsten Beispiel für eine Feistel-Chiffre.

Wir illustrieren nun die Entschlüsselung. Sei $c = (R_r, L_r)$, $p = (L_0, R_0)$.
Es gilt

$$\begin{aligned} L_r &= R_{r-1} \\ R_r &= L_{r-1} \oplus f_{k_r}(R_{r-1}) \\ \Rightarrow L_{r-1} &= R_r \oplus f_{k_r}(L_r) \end{aligned}$$

Die Schritte sehen also folgendermaßen aus:

$$c = R_rL_r \xrightarrow{k_r} R_{r-1}L_{r-1} \xrightarrow{k_{r-1}} R_{r-2}L_{r-2} \xrightarrow{k_{r-2}} \ldots \xrightarrow{k_1} R_0L_0 \overset{\text{Vertau-schung}}{\longrightarrow} p = L_0R_0$$

1.36 Bemerkung (Grobstruktur von DES)
DES ist wie folgt definiert:

1. DES ist eine Feistel-Chiffre, also eine symmetrische Blockchiffre.
2. $\mathcal{P} = \mathcal{C} = \mathbb{F}_2^{64}$
3. $\mathcal{K} = \mathbb{F}_2^{56}$. Die Schlüssellänge bei DES beträgt nur scheinbar 64 Bits. Die Bits $8, 16, 24, 32, 40, 48, 56, 64$ sind sogenannte Paritätsbits. Sie hängen von den jeweils vorangegangenen sieben Bit ab und werden so gewählt, dass die Ziffernsumme des jeweiligen Bytes ungerade ist. Es sind also im Schlüssel nur 56 Bits frei wählbar.

Erster Schritt von DES:
Der 64-Bit Klartextblock p wird einer festen Eingangspermutation π, $p \mapsto \pi(p)$ unterworfen. Dies dient nicht der kryptographischen Sicherheit, sondern der leichteren Hardwareimplementierung. Die Permutation π wird in der Literatur stets einer Form geschrieben, die nicht sehr aufschlußreich ist, siehe Abbildung 1.2. Wenn wir uns aber die 64 Bits Byte für Byte zeilenweise aufgeschrieben

58	50	42	34	26	18	10	2
62	54	46	38	30	22	14	6
...							

60	52	44	36	28	20	12	4
64	56	48	40	32	24	16	8
...							

Abbildung 1.2: Die Eingangspermutation von DES

denken, dann sehen wir, dass die Permutation zuerst die Bits der zweiten Spalte in umgekehrter Reihenfolge abarbeitet, dann auf gleiche Weise die vierte Spalte, und so weiter (siehe Tabelle 1.1).

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
⋮	26		28				⋮
⋮	34		36				⋮
⋮	42		44				⋮
⋮	50		52				⋮
⋮	58	...	60	61	62	63	64

Tabelle 1.1: Die 64 Bits des Klartextblockes

Zweiter Schritt von DES:
Hier findet die eigentliche Verschlüsselung statt. Das 64-Bit Wort $\pi(p)$ wird in die linke Hälfte L_0 und die rechte Hälfte R_0 aufgeteilt. In 16 gleichartigen Schritten, die jeweils von einem 48-Bit Teilschlüssel des ursprünglichen Schlüssels abhängen, werden zwei 32-Bit Blöcke L_{16} und R_{16} erzeugt und abschließend zu einem 64-Bit Block (R_{16}, L_{16}) zusammengefügt. Die dabei benötigten 16 Teilschlüssel werden durch ein Schlüsselauswahlverfahren (Englisch: *key schedule*) erzeugt.

Dritter Schritt von DES.
Der 64-Bit Block (R_{16}, L_{16}) wird der festen Permutation π^{-1} unterworfen, dies ergibt dann den Geheimtext c.

1.37 Bemerkung (Die Funktion f_{k_i})
In jeder Runde der Feistel-Chiffre DES wird eine Funktion $f_{k_i}(R)$ verwendet, wobei $k_i \in \mathbb{F}_2^{48}$ und $R \in \mathbb{F}_2^{32}$ gilt. Diese Funktion mit zwei Argumenten (k_i und

R) ist als die folgende Funktion F definiert:

$$\begin{aligned} F: \quad & \mathbb{F}_2^{32} \times \mathbb{F}_2^{48} \to \mathbb{F}_2^{32}, \\ & (R, K) \mapsto F(R, K). \end{aligned}$$

Diese Schreibweise besagt, dass $F(R, K) = f_K(R)$ gilt.

Die Schritte bei der Berechnung von F lauten:

1. Der 32-Bit Block R wird mit Hilfe einer Expansionsabbildung E in einen 48-Bit Block $E(R)$ übergeführt, indem er mit einer (48,32)-Matrix über $\mathbb{F}_2$ multipliziert wird. Dies ergibt den 48-Bit Block $E(R)$.

   Die Matrix zu dieser Expansionsabbildung verdoppelt manche Bits. Sie hat die Gestalt

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ & & & & & & \vdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \dots & & & & & & & & \end{pmatrix}$$

   Dabei werden genau 16 Bits zweimal verwendet, wir haben hier die Mehrfachverwendung der Bits Nr. 4 und 5 illustriert. Wenn wir die Bits von R durchnummerieren, dann lauten die ersten Bits von $E(R)$ wie folgt: (32 1 2 3 4 5 4 5 6 7 8 9 8 9 ...). Für die Details verweisen wir auf [Ert01, Sch96, Sti06].

2. Die beiden 48-Bit Blöcke $E(R)$ und K werden addiert:

$$E(R) \oplus K.$$

3. Der 48-Bit Block $E(R) \oplus K$ wird in 8 Teilblöcke $B_1, B_2, \dots, B_8$ zerlegt. Jeder Block B_j besitzt also 6 Bits.

4. *Der kryptographisch wichtigste Schritt bei DES*
   Jeder Block B_j wird mittels einer S-Box S_j in einen 4-Bit Block abgebildet. S_j ist eine $(4, 16)$-Matrix, die Einträge in dieser Matrix sind nichtnegative ganze Zahlen im Bereich $\{0, 1, \dots, 15\}$.

   Für den 6-Bit Block $B = b_1b_2b_3b_4b_5b_6$ wird der Wert $S(B) \in \mathbb{F}_2^4$ wie folgt berechnet. Das Bitpaar b_1b_6 wird als Binärdarstellung einer ganzen Zahl $s \in \{0, 1, 2, 3\}$ interpretiert und legt die Zeile s der Matrix S fest. Das binäre Wort $b_2b_3b_4b_5$ wird ebenso als eine Binärentwicklung interpretiert und legt die Spalte $t \in \{0, 1, \dots, 15\}$ der Matrix S fest.

Die Zahl, die in der s-ten Zeile und t-ten Spalte der Matrix S aufscheint, liegt im Bereich $\{0, 1, \ldots, 15\}$ und wird daher durch vier Bits dargestellt. Dieses 4-Bit Wort ist dann der gesuchte Wert $S(B)$.

5. Auf das so entstandene 32-Bit Wort $S_1(B_1)S_2(B_2)\ldots S_8(B_8)$ wird dann noch eine Permutation angewandt. Dies ergibt nun den gesuchten Wert

$$F(R, K) \in \mathbb{F}_2^{32}$$

1.38 Bemerkung (Nichtlinearität von DES)
Jede Permutation eines Bitblocks $B \in \mathbb{F}_2^n$ ist eine lineare Abbildung von $\mathbb{F}_2^n$ in sich. Für $B = b_1 b_2 \ldots b_n$ lautet das Bild unter der Permutation π daher $\pi(B) = b_{\pi(1)} b_{\pi(2)} \ldots b_{\pi(n)}$. Dann gilt aber $\pi(B)^T = A \cdot B^T$ mit einer (n, n)-Permutationsmatrix $A = (a_{ij})_{i,j=1}^n$, wobei $a_{ij} = \delta_{\pi(i),j}$ ($a_{ij} = 1$ falls $j = \pi(i)$ und $a_{ij} = 0$ sonst). Daher gilt: Die Abbildung

$$\begin{aligned} &\mathrm{DES} : \mathcal{P} \times \mathcal{K} \to \mathcal{C} \\ &\mathrm{DES}(p, k) \text{ wie vorhin beschrieben} \end{aligned}$$

ist nichtlinear genau dann, wenn die Funktionen $f_{k_i} : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ nichtlinear sind. Letzteres ist aber der Fall:

- Die Expansionsabbildung $R \mapsto E(R)$ ist linear, $E(R) = (A \cdot R^T)^T$.
- Die Abbildungen $B \mapsto S(B)$ sind alle nichtlinear, denn es gilt

$$\begin{aligned} S(000000) &= \text{Binärdarstellung des Elementes links oben in S} \\ &\neq 0000. \end{aligned}$$

  Dazu beachtet man, dass bei jeder der acht S-Boxen $S_1, S_2, \ldots, S_8$ die Zahl im linken oberen Eck ungleich 0 ist. Damit kann S nicht linear sein, denn bei jeder linearen Abbildung geht der Nullvektor in den Nullvektor über.

Somit gilt: DES ist nichtlinear. Die einzigen nichtlinearen Funktionen in DES sind die acht S-Boxen $S_1, S_2, \ldots, S_8$.

1.39 Bemerkung (Schwache Schlüssel)
Für die Sicherheit von DES ist wesentlich, dass wir die einzelnen Rundenschlüssel nur mit größter Mühe knacken können. Daher spricht man von *schwachen* Schlüsseln von DES, wenn die Menge der Rundenschlüssel weniger als 16 Elemente enthält, wenn also manche Rundenschlüssel gleich sind.

Bei den so genannten *weak keys* sind alle 16 Rundenschlüssel identisch. Es gibt, wie man zeigen kann, genau vier derartig schwache Schlüssel von DES. Man spricht von *semi-weak keys*, wenn unter den 16 Rundenschlüsseln nur zwei verschiedene auftauchen. Auch die 12 derartigen Schlüssel sind bekannt.

In der Praxis wird DES zusammen mit einer Liste dieser schwachen Schlüssel implementiert, damit die zufällige Verwendung von derartigen Schlüsseln ausgeschlossen werden kann.

1.40 Bemerkung (Sicherheit von DES)
In Zusammenhang mit Attacken auf Blockchiffren wurden zwei wichtige theoretische Konzepte entwickelt, die lineare Kryptoanalyse (LC) und die differentielle Kryptoanalyse (DC). Beide Methoden sind für die praktische Kryptoanalyse von DES nicht wirklich geeignet, da sehr große Mengen an Klartext-Geheimtext-Paaren bekannt sein müssen (Größenordnung etwa 2^{40}). Immer noch ist eine brute-force-Attacke auf den Schlüssel am effizientesten. Wegen des relativ kleinen Schlüsselraums $|K| = 2^{56}$ ist dies heute technisch machbar. Historisch ist bemerkenswert, daß DES bereits so konstruiert wurde, dass DC praktisch unmöglich ist. Die NSA wußte also bereits Anfang der 70-er Jahre von dieser Methode. In der freien wissenschaftlichen Literatur ist erst Anfang der 90er Jahre darüber publiziert worden (Biham und Shamir [BS93]).

1.41 Bemerkung (Triple DES)
Die Dreifachverschlüsselung der Form E-D-E (encrypt-decrypt-encrypt) ist eine bekannte Technik, um die effektive Schlüssellänge von Blockchiffren zu erhöhen:

$$c \;=\; e_{k_3}(d_{k_2}(e_{k_1}(p)))$$

Dabei muß natürlich $k_1 \neq k_2$ und $k_2 \neq k_3$ gelten. In der Praxis ist $k_1 = k_3$ üblich, wie eben bei Triple DES:

$$c = \mathrm{DES}_{k_1}(\mathrm{DES}_{k_2}^{-1}(\mathrm{DES}_{k_1}(p)))$$

Dieses Verfahren ist abwärts kompatibel zum klassischen DES-Verfahren, indem wir $k_1 = k_2$ setzen. Die effektive Schlüssellänge bei Triple DES beträgt $112 = 56 + 56$ Bits.

1.42 Bemerkung (DES-X, DESX)
Die Sicherheit von DES kann auch auf folgende Weise erhöht werden. R. Rivest hat 1984 das nachstehende Konzept vorgeschlagen:

$$c \;=\; k_3 \oplus e_{k_2}(p \oplus k_1),$$

wobei $k_i \neq k_j$, $i \neq j$, vorausgesetzt wird. Wenn man $e_{k_2} = \mathrm{DES}_{k_2}$ wählt, dann wird die Chiffre DES-X (oder oft einfach "DESX") genannt.

1.43 Bemerkung (Meet-in-the-middle Attacke)
Alice möchte die Sicherheit der von ihr verwendeten Chiffre erhöhen, indem sie nicht nur einen, sondern zwei voneinander unabhängige Schlüssel k_1 und k_2 verwendet und damit den Klartext p zweimal verschlüsselt:

$$c \;=\; e_{k_2}((e_{k_1}(p)).$$

Sofern die Chiffre von Alice bezüglich der Hintereinanderausführung *keine Gruppe* bildet, das heißt, dass kein k_3 existiert mit $e_{k_3} = e_{k_2} \circ e_{k_1}$, scheint der Gegner nun nicht nur einen, sondern sogar zwei Schlüssel finden zu müssen. Da es $|\mathcal{K}|^2$ solcher Schlüsselpaare (k_1, k_2) gibt, scheint der Schlüsselraum dieser Produktchiffre gewaltig gewachsen zu sein, eben von $|\mathcal{K}|$ auf $|\mathcal{K}|^2$ Elemente.

Dem ist leider nicht so, wie die *"meet-in-the-middle"*-Attacke zeigt, eine Attacke mit gekanntem Klartext ("known-plaintext attack"), siehe dazu Merkle und Hellman[MH81].

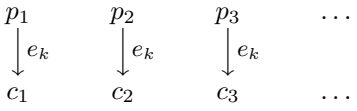
1.3.1 Betriebsarten einer Blockchiffre

Betriebsarten einer Blockchiffre kombinieren die Chiffre, fallweise eine Form von Rückkoppelung und einige einfache Operationen. Sie dürfen die Sicherheit der zugrunde liegenden Chiffre nicht beeinträchtigen.

Betriebsarten, wie wir sie am Beispiel von DES besprechen werden, dienen der Lösung bestimmter Aufgaben, wie zum Beispiel der Verhinderung von Manipulationen am Klartext durch Einschleusen von Fehlern in den Geheimtext, oder der Erhöhung der Fehlertoleranz, um zum Beispiel die Fortpflanzung von Übertragungsfehlern so weit wie möglich zu verhindern.

Die im Folgenden besprochenen Betriebsarten sind für jede Blockchiffre anwendbar, sie sind nicht auf DES beschränkt. Für detaillierte Informationen verweisen wir auf Schneier [Sch96] sowie Stinson [Sti06, Ch.3.4.1].

1.44 Bemerkung (ECB Modus)
Beim *electronic codebook* Modus (ECB) wird jeder Klartextblock unabhängig von allen anderen verschlüsselt:



Es gilt

1. *Effizienz*
   Der ECB Modus eignet sich sehr gut dazu, die Effizienz der Verschlüsselung zu steigern. Da jeder Klartextblock getrennt verschlüsselt wird, können wir den Vorgang parallelisieren.

2. *Kodebuch*
   Jeder Klartextblock wird bei konstantem Schlüssel k stets in den gleichen Geheimtextblock verschlüsselt. Wir können also zumindest theoretisch ein Kodebuch (englisch: codebook) erstellen. Bei einer Blocklänge von 64 Bits müßte dieses Kodebuch aber 2^{64} Einträge besitzen. Dies ist viel mehr, als in der numerischen Praxis handhabbar wäre. Aber es gilt: wegen der stereotypen Anfangs- und Endsequenzen bei vielen Nachrichten (Stichwort: Vorspann und Nachspann bei Dateien) gelingt es trotzdem, ein kleines Kodebuch aufzubauen und für Attacken zu nutzen.

3. *Datenbanken*
   Der ECB Modus ist bei Datenbankanwendungen sehr praktisch. Wenn die Datenbank im ECB Modus verschlüsselt ist, dann kann ein record hinzugefügt, gelöscht oder verändert werden, ohne dass deswegen die ganze Datenbank ver- oder entschlüsselt werden muß. Damit ist der ECB Modus für diese Art von Anwendung sehr effizient.

4. *Fehlertoleranz*
   Wegen des Lawineneffektes verfälscht ein gekipptes Bit im Geheimtextblock bereits etwa 50% der Bits im Klartextblock. Damit wird dieser Block unentschlüsselbar. Die restlichen Geheimtextblöcke bleiben davon unberührt.

5. *Synchronisation*
   Fehlende oder versehentlich hinzugefügte Bits im Geheimtext machen den ganzen nachfolgenden Geheimtext unentschlüsselbar, da sich die Blockgrenzen verschieben.

1.45 Bemerkung (CBC Modus)
Beim *cipher block chaining* Modus (CBC) wird jeder Klartextblock mit dem vorangegangenen Geheimtextblock verschlüsselt, die Klartextblöcke werden also nicht mehr unabhängig voneinander verschlüsselt. Dazu wird zuerst ein *Initialisierungsblock* c_0 gewählt, mittels XOR zum ersten Klartextblock p_1 addiert und dann das Ergebnis verschlüsselt. Dies ergibt den ersten Geheimtextblock c_1. Dieser wird zum zweiten Klartextblock p_2 addiert, das Ergebnis verschlüsselt, und so weiter (siehe Abbildung 1.3).

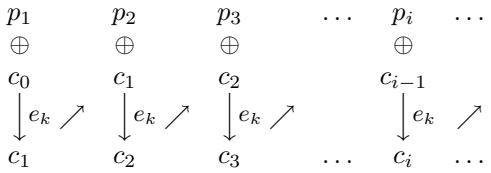


Abbildung 1.3: CBC Modus, Verschlüsselung

Zum Entschlüsseln geht man nach dem gleichen Schema vor (siehe Abbildung 1.4).

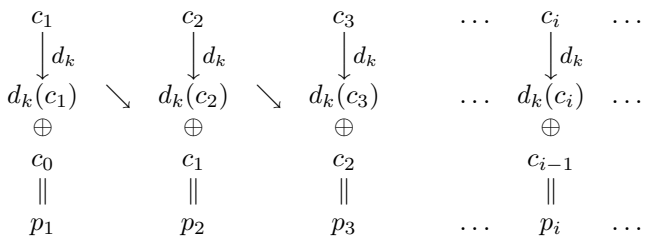


Abbildung 1.4: CBC Modus, Entschlüsselung

Es gilt

1. *Initialisierungsblock*
   Der Initialisierungsblock c_0 ermöglicht es, identische Klartexte in verschiedene Geheimtexte zu verschlüsseln. Sei zum Beispiel c_0 ein Zeitstempel. Dann ergibt die Verschlüsselung des gleichen Klartextes zu verschiedenen Zeiten verschiedene Geheimtexte. Der Initialisierungsblock c_0 muß nicht geheim gehalten werden. Er kann sogar im Klartext übertragen werden und muß auch nicht mit jeder Nachricht gewechselt werden. Wir müssen c_0 auch gar nicht übertragen: Alice wählt c_0 zufällig und erzeugt dann $c_1 = e_k(p_1 \oplus c_0)$, $c_2 = e_k(p_2 \oplus c_1)$, und so fort. Bob entschlüsselt $p_1' = d_k(c_1) \oplus c_0'$, wobei c_0' von ihm ebenfalls zufällig gewählt wurde.

Das wird einen Unsinn ergeben, es wird also mit größter Wahrscheinlichkeit $p'_1 \neq p_1$ gelten, aber schon beim nächsten Block stimmt alles, denn $p_2 = d_k(c_2) \oplus c_1$! Bob muß also nur den ersten unbrauchbaren Block p'_1 ignorieren, der Rest ist in Ordnung.

2. *Fehlerfortpflanzung*
   Ein falsches Bit in einem Klartextblock verändert den Geheimtextblock und damit alle folgenden Blöcke c_i. Beim Entschlüsseln erhalten wir wieder den bis auf ein Bit verfälschten Klartext, also ist die Fehlerfortpflanzung perfekt gering. Viel häufiger tritt aber das Phänomen auf, dass ein Bit im Geheimtext kippt, durch Fehler bei der Übertragung. Sei also der empfangene Geheimtextblock c'_i ungleich dem gesendeten Block c_i, mit Hamming-Distanz Eins. Dann gilt:

$$\begin{array}{rcll} p_i & = & d_k(c'_i) \oplus c_{i-1} & \text{ist stark verfälscht (Lawineneffekt)} \\ p_{i+1} & = & d_k(c_{i+1}) \oplus c'_i & \text{ist nur um das Bit von } c'_i \text{ verfälscht} \\ p_{i+2} & = & d_k(c_{i+2}) \oplus c_{i+1} & \text{ist unverfälscht} \end{array}$$

   Diese Eigenschaft des CBC Modus heißt im Englischen *self recovering*.

3. *Synchronisation*
   Genausowenig wie der ECB Modus so kann auch der CBC Modus einen Synchronisationsfehler, das heißt ein verlorenes oder ein hinzugefügtes Bit, nicht korrigieren.

4. *Wartezeit*
   Die Verschlüsselung kann erst beginnen, wenn ein kompletter Datenblock erstellt ist. Das kann in manchen zeitkritischen Anwendungen von Nachteil sein.

1.46 Bemerkung (CFB Modus)
Beim *Cipher Feedback* Modus (CFB) wird jeder Klartextblock zum vorangegangenen verschlüsselten(!) Geheimtextblock addiert. Dazu wird zuerst ein Initialisierungsblock c_0 gewählt, dann wird c_0 zu $e_k(c_0)$ verschlüsselt und zu p_1 addiert. Dies ergibt den ersten Geheimtextblock $c_1 = p_1 \oplus e_k(c_0)$. In der Abbildung 1.5 wird die Verschlüsselung mittels e_k mit dem Symbol $\rightsquigarrow$ bezeichnet.

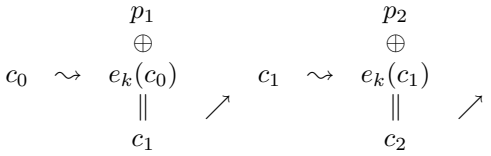


Abbildung 1.5: CFB Modus, Verschlüsselung

Es gilt:

1. *Initialisierungsblock*
   Der Initialisierungsblock muß mit jeder Nachricht gewechselt werden, damit identische Nachrichten verschieden verschlüsselt werden.

2. *Fehlerfortpflanzung*
   Ein falsches Bit im empfangenen Geheimtext c_i ergibt ein falsches Bit beim entschlüsselten Klartext p_i, aber wegen des Lawineneffektes ergibt dieses eine falsche Bit in c_i viele falsche Bits in $e_k(c_i)$ (ca. 50% der Bits kippen ja!) und damit viele falsche Bits in $p_{i+1} = c_{i+1} \oplus e_k(c_i)$. Wenn wir annehmen, dass c_{i+1} korrekt empfangen wurde, dann stimmt aber bereits der nächste Klartextblock wieder, denn $p_{i+2} = c_{i+2} \oplus e_k(c_{i+1})$.

1.47 Bemerkung (OFB Modus)
Beim *Output Feedback* Modus (OFB) wird zuerst ein Initialisierungsblock c_0 gewählt, dann wird durch Iteration der Funktion e_k eine Pseudozufallszahlenfolge $(z_i)_{i\geq 0}$ erzeugt, $z_i = e_k^{(i)}(z_0)$, $e_k^{(i+1)} = e_k \circ e_k^{(i)}$, $i = 1, 2, \ldots$, und schließlich dieser Schlüsselstrom wie bei einer synchronen Stromchiffre Block für Block zum Klartextstrom $(p_i)_{i\geq 0}$ addiert, $c_i = p_i \oplus z_i$. Dies ergibt den Geheimtextstrom $(c_i)_{i\geq 0}$.

In der Abbildung 1.6 wird die Verschlüsselung mittels e_k mit dem Symbol $\rightsquigarrow$ bezeichnet.

$$\begin{array}{ccccccccc}
 & & p_1 & & p_2 & \cdots & p_i & \cdots \\
 & & \oplus & & \oplus & & \oplus & \\
c_0 & \rightsquigarrow & e_k(c_0) & \rightsquigarrow & e_k^{(2)}(c_0) & & e_k^{(i)}(c_0) & \rightsquigarrow \\
 & & \| & & \| & & \| & \\
 & & c_1 & & c_2 & \cdots & c_i & \cdots
\end{array}$$

Abbildung 1.6: OFB Modus, Verschlüsselung

Die Entschlüsselung im OFB Modus ist ebenso einfach (siehe Abbildung 1.7):

$$\begin{array}{ccccccccc}
 & & c_1 & & c_2 & \cdots & c_i & \cdots \\
 & & \oplus & & \oplus & & \oplus & \\
c_0 & \rightsquigarrow & e_k(c_0) & \rightsquigarrow & e_k^{(2)}(c_0) & & e_k^{(i)}(c_0) & \rightsquigarrow \\
 & & \| & & \| & & \| & \\
 & & p_1 & & p_2 & \cdots & p_i & \cdots
\end{array}$$

Abbildung 1.7: OFB Modus, Entschlüsselung

Es gilt:

1. *Initialisierungsblock*
   Dieser darf sogar im Klartext übertragen werden, denn der Schlüsselstrom $(e_k^{(i)}(c_0))_{i\geq 0}$ bleibt trotzdem geheim. Der Gegner kennt ja den Schlüssel k nicht und kann daher den Schlüsselstrom nicht erzeugen.

2. *Fehlerfortpflanzung*
   Ein falsches Bit hat keine Einwirkungen auf die nachfolgenden. Daher gibt es beim OFB Modus keine Fehlerfortpflanzung. Er ist für Ton- oder Videoverschlüsselung daher sehr gut geeignet (Stichwort “ Video on Demand”), ebenso für Satellitenübertragungen.

3. *Synchronisationsfehler*
   Beide Seiten, Sender und Empfänger, müssen unbedingt im Takt sein, ansonsten ist die Entschlüsselung unmöglich. Daher ist auch beim OFB Modus ein Synchronisationsverfahren notwendig.

1.48 Bemerkung (Authentikation)
Fehlerfortpflanzung ist manchmal unerwünscht, wie etwa bei der Satellitenkommunikation. In anderen Anwendungen ist sie aber höchst erwünscht. Die kleinste Änderung im Klartext, etwa ein hinzugefügtes Leerzeichen, soll bereits auffallen. Deswegen sind die Modi mit Fehlerfortpflanzung wie CBC und CFB hier besonders interessant. Mit Hilfe des CBC Modus kann man einen sogenannten *Message Authentication Code* (MAC) erzeugen. Mit einem MAC überprüft man, ob eine Nachricht während der Übertragung verfälscht wurde, etwa durch Leitungsstörungen oder durch einen aktiven Gegner. Die Vorgangsweise ist die folgende:

1. Alice will an Bob eine Nachricht m (m: message) übertragen, die bei der Übertragung auf keinen Fall verfälscht werden darf (z.B. einen wichtigen Vertrag). Dazu benützen Alice und Bob die gleiche Blockchiffre mit dem gleichen geheimen Schlüssel k.

2. Alice erzeugt einen Initialisierungsblock c_0 und verschlüsselt anschließend den Klartext $m = p_1 \dots p_n$ im CBC Modus mit dem Schlüssel k. Sie erhält auf diese Weise die Geheimtextblöcke $c_1, \dots, c_n$.

3. Es gilt die Festlegung $\mathrm{MAC}(m) = c_n$, der MAC der Nachricht m ist bei diesem Verfahren also der letzte Geheimtextblock.

4. Alice überträgt die Nachricht m, also die Klartextblöcke $p_1, \dots, p_n$ zusammen mit dem MAC c_n. (Ob Alice dies in verschlüsselter Form oder im Klartext macht, ist hier unwesentlich)

5. Bob empfängt die Nachricht m', also die Klartextblöcke $p'_1, \dots, p'_n$. Er hofft, dass $m = m'$ gilt, also $p'_i = p_i$, $1 \leq i \leq n$. Um diese Vermutung zu überprüfen, bedient er sich des MAC. Dazu wendet er wie Alice den CBC Modus an und verschlüsselt p'_1 bis p'_n mit dem Schlüssel k. Er erhält so als letzten Geheimtextblock c'_n.

6. Bob vergleicht den MAC c_n, den er von Alice erhalten hat, mit dem von ihm berechneten:
$$\mathrm{MAC}(m) = c_n = c'_n = \mathrm{MAC}(m')?$$

7. Wenn diese Gleichung erfüllt ist, dann geht Bob davon aus, dass die von ihm empfangene Nachricht m' mit der von Alice versendeten Nachricht m übereinstimmt.

Warum ist das Vertrauen von Bob in den MAC gerechtfertigt? Da ein MAC bei DES 64 Bit lang ist und es sehr viel mehr mögliche Nachrichten als 2^{64} gibt, ist die Zuordnung $m \mapsto \mathrm{MAC}(m)$ offensichtlich (hochgradig!) nicht-injektiv. Der soeben vorgestellte kryptographische MAC ist aber so konstruiert, dass bei

Kenntnis der Nachricht m und bei Kenntnis von MAC(m) in der Praxis keine Chance besteht, eine andere Nachricht m' mit dem gleichen MAC-Wert zu erzeugen. Der Angreifer kennt ja den geheimen Schlüssel k nicht, der von Alice zur Berechnung des MAC-Wertes verwendet wurde. Der Angreifer besitzt zwar den Klartext $m = p_1 \dots p_n$ und den Geheimtext c_n, aber er kennt den Schlüssel nicht. Er steht also vor der Aufgabe, die jeweilige Blockchiffre zu knacken. Dies ist bekanntermaßen ein harter Brocken, denn genau so sind diese Chiffrierverfahren konstruiert, dass sie ihren geheimen Schlüssel nur extrem schwer verraten!

Wenn Alice Authentikation und Vertraulichkeit wünscht, so muß sie wie vorhin den MAC-Wert erzeugen und dann noch alles (üblicherweise mit einem zweiten geheimen Schlüssel) verschlüsseln, bevor sie die Nachricht an Bob schickt. Wie man bei dieser zweifachen Aufgabe geschickt vorgeht, das ist in Schneier [Sch96] nachzulesen.

1.4 AES

Auf Grund der raschen Fortschritte bei der Durchführung von brute-force-Attacken auf DES wurde Anfang 1997 mit der öffentlichen Diskussion um ein geeignetes Nachfolgeverfahren begonnen. Nach einem intensiven öffentlichen Auswahlverfahren wurde am 2.10.2000 das Ergebnis dieses weltweiten Wettbewerbes für den Nachfolger von DES bekannt gegeben: Rijndael, ein belgisches Verfahren. Der Advanced Encryption Standard (AES) wurde am 26. November 2001 ein offizieller Standard der Vereinigten Staaten und als "Federal Information Processing Standard (FIPS)" veröffentlicht, siehe FIPS PUB 197 [Nat01].

Der einzige Unterschied zwischen AES und Rijndael besteht in den zugelassenen Blocklängen für die Klartexte und die Schlüssel (siehe Daemen und Rijmen [DR02]).

AES/Rijndael wurde so entworfen, dass bestimmte Kriterien erfüllt sind: (a) Geschwindigkeit und kompakter Code auf vielen Hardware-Platformen (AES ist ein sehr schneller Algorithmus, effizient genug für viele Zwecke auch des wissenschaftlichen Rechnens, Stichwort: Simulation), (b) Sicherheit gegen alle bekannten Formen der Attacke.

AES ist eine iterierte Blockchiffre, deren Blocklänge b und Schlüssellänge k wie folgt zusammenhängen (siehe Tabelle 1.2). Dabei bezeichnet $r = r(k, b)$ die erforderliche Rundenzahl.

r	$b = 128$	$b = 192$	$b = 256$
$k = 128$	10	12	14
$k = 192$	12	12	14
$k = 256$	14	14	14

Tabelle 1.2: Die Parameter von AES

1.49 Bemerkung Die Zwischenergebnisse des Verschlüsselungsprozesses von AES werden ein *Zustand* genannt. Sie werden mit $S_0, S_1, \dots$ bezeichnet. Ein Zustand S wird als eine Matrix von Bytes geschrieben. Für 128-Bit Zustände

S lautet diese Matrix wie folgt: Eine einzelne Runde von AES/Rijndael hat die

$$S = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

Tabelle 1.3: Ein 128-Bit Zustand von AES

folgende Gestalt:

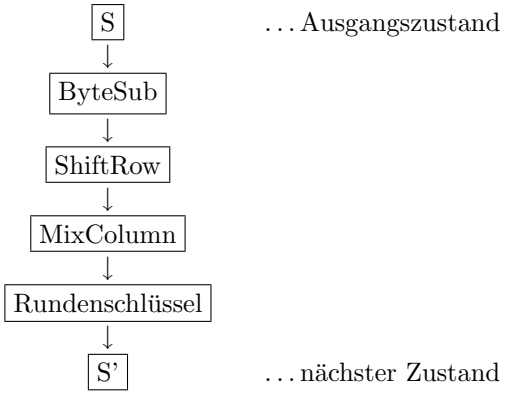


Tabelle 1.4: Eine Runde von AES

Die Struktur von AES ist also wesentlich übersichtlicher als jene von DES. Für die einzelnen Teilalgorithmen gilt:

1. *ByteSub*
   Diese Transformation ist die einzige nichtlineare Abbildung von Rijndael. ByteSub wird auf jedes Byte des Zustandes S angewendet. Sei a ein Byte von S. Wir können a als ein Element des Körpers $\mathrm{GF}(2^8)$ interpretieren. Sei $a^{-1} \in \mathrm{GF}(2^8)$ das inverse Element zu a bezüglich der Multiplikation. Wenn $a = (0, \ldots, 0)$ sein sollte, dann definieren wir a^{-1} als $(0, \ldots, 0)$. Im nächsten Schritt wird a^{-1} mit einer festen binären $(8, 8)$-Matrix A multipliziert und ein fester Vektor $b \in \mathrm{GF}(2^8)$ addiert:

   $$a \mapsto A \cdot a^{-1} + b$$

   Das eigentliche nichtlineare Bauelement von ByteSub und damit von Rijndael ist die Inversion $a \mapsto a^{-1}$. Der Schritt $a \mapsto A \cdot a^{-1} + b$ dient der Konfusion im Sinne von Shannon, indem die relativ einfache algebraische Beschreibung der Abbildung $a \mapsto a^{-1}$ durch die (im Sinne von Permutationspolynomen) wesentlich schwieriger zu beschreibende Permutation $a \mapsto A \cdot a^{-1} + b$ der Menge $\mathrm{GF}(2^8)$ ersetzt wird. Die Matrix A ist regulär.

2. *ShiftRow*
   In ByteSub wurde jedes einzelne Byte a des Zustandes S transformiert. Nun werden die Bytes des Zustandes S durchgemischt. Dies geschieht zuerst mit ShiftRow: Die einzelnen Zeilen des Zustandes S werden zyklisch

nach rechts verschoben. Um wieviel Bytes man verschiebt, hängt von der Größe von S ab: Für $|S| = 128$ Bit wird Zeile 1 um 1 Byte, Zeile 2 um 2 Byte und Zeile 3 um 3 Byte verschoben. Für $|S| = 256$ lauten die Verschiebungen nicht mehr $1, 2, 3$, sondern $1, 3, 4$. Zeile 0 bleibt unverändert.

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

Tabelle 1.5: Die ShiftRow-Transformation

3. *MixColumn*
   Hier werden die einzelnen Spalten des Zustandes S mit Hilfe einer invertierbaren Matrix durchgemischt. Jedes Byte der neuen Spalte hängt von allen Bytes der alten Spalte ab.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = A \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad (x_i, y_j \in \mathbb{F}_2^8)$$

Tabelle 1.6: Die MixColumn-Transformation

1.50 Bemerkung (Einige Eigenschaften von AES)

- Starke Diffusion:
  jedes Zustandsbit hängt von allen Bits des Zustandes vor zwei Runden ab.
- wenn ein Zustandsbit verändert wird, dann ändert sich nach zwei Runden ungefähr die Hälfte der Bits
- aufgrund der einfachen, byte-orientierten Struktur von AES ist diese Blockchiffre auf vielen Platformen verwendbar

1.51 Bemerkung
Auf einem 1 GHz PC liegt der Datendurchsatz bei der Verschlüsselung mit AES mit $|S| = 128$ Bit bei 200 MBit/sec. Die Byte-orientierten Operationen von AES sind auch für Chipkarten sehr gut geeignet.

1.5 Zur linearen Kryptoanalyse

Sei der Bitstring $u \in \mathbb{F}_2^n$ gegeben. Ein Bitstring $\alpha \in \mathbb{F}_2^n$ wählt mittels des inneren Produkts

$$\alpha \cdot u = \alpha_1 u_1 + \ldots \alpha_n u_n$$

jene Bits in u aus und addiert diese anschließend, wo die Koordinaten von α ungleich Null sind.

Wir nennen α deshalb eine *Maske*.

Sei $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ eine S-Box mit Eingabevektor $u = (u_1, \ldots, u_n)$ und Ausgabevektor $v = (v_1, \ldots, v_n)$.

Seien $U_1, \ldots, U_n$ unabhängige, identisch in $\mathbb{F}_2$ gleichverteilte Zufallsvariable. Unter einer *linearen Approximation* der S-Box f versteht man ein Ereignis der Form

$$\alpha U \oplus \beta V \ = \ 0, \quad \alpha, \beta \in \mathbb{F}_2^n,$$

mit der Eigenschaft, dass die Wahrscheinlichkeit dieses Ereignisses,

$$P(\alpha U \oplus \beta V \ = \ 0)$$

verschieden ist von $1/2$.

Für eine gegebene Chiffre wie DES oder AES besteht die Kunst des Kryptographen darin, für die gegebenen S-Boxen geeignete lineare Approximationen zu finden, bei denen die Abweichung von $1/2$ möglichst groß ist. Indem weiters die verschiedenen Runden der Blockchiffre geeignet verknüpft werden, erhält man zwischen Klartextblock P, Geheimtextblock C und Schlüssel K ein Ereignis der Form

$$\alpha P \oplus \beta C \oplus \gamma K \ = \ 0, \quad \alpha, \beta, \gamma \in \mathbb{F}_2^n.$$

Indem man viele Klartextblöcke p mit dem unbekannten Schlüssel k verschlüsselt, erhält man für γk einen Schätzwert, aus dem man ein klein wenig Information über den Schlüssel k extrahieren kann:

$$\alpha p \oplus \beta c \oplus \gamma k \ = \ 0, \iff \alpha p \oplus \beta c \ = \ \gamma k.$$

Dies war eine sehr oberflächliche Beschreibung der linearen Kryptoanalyse. Das folgende Beispiel der Verknüpfung von S-Boxen eines Substitutions-Permutations-Netzwerks in dem oben genannten Sinn ist der Diplomarbeit von S. Loiperdinger [Loi07] entnommen, siehe auch Heys [Hey02]. In diesen Arbeiten wird auch die *differentielle Kryptoanalyse* diskutiert, siehe zu diesem Thema auch das ausführliche Beispiel in Trappe und Washington [TW06].

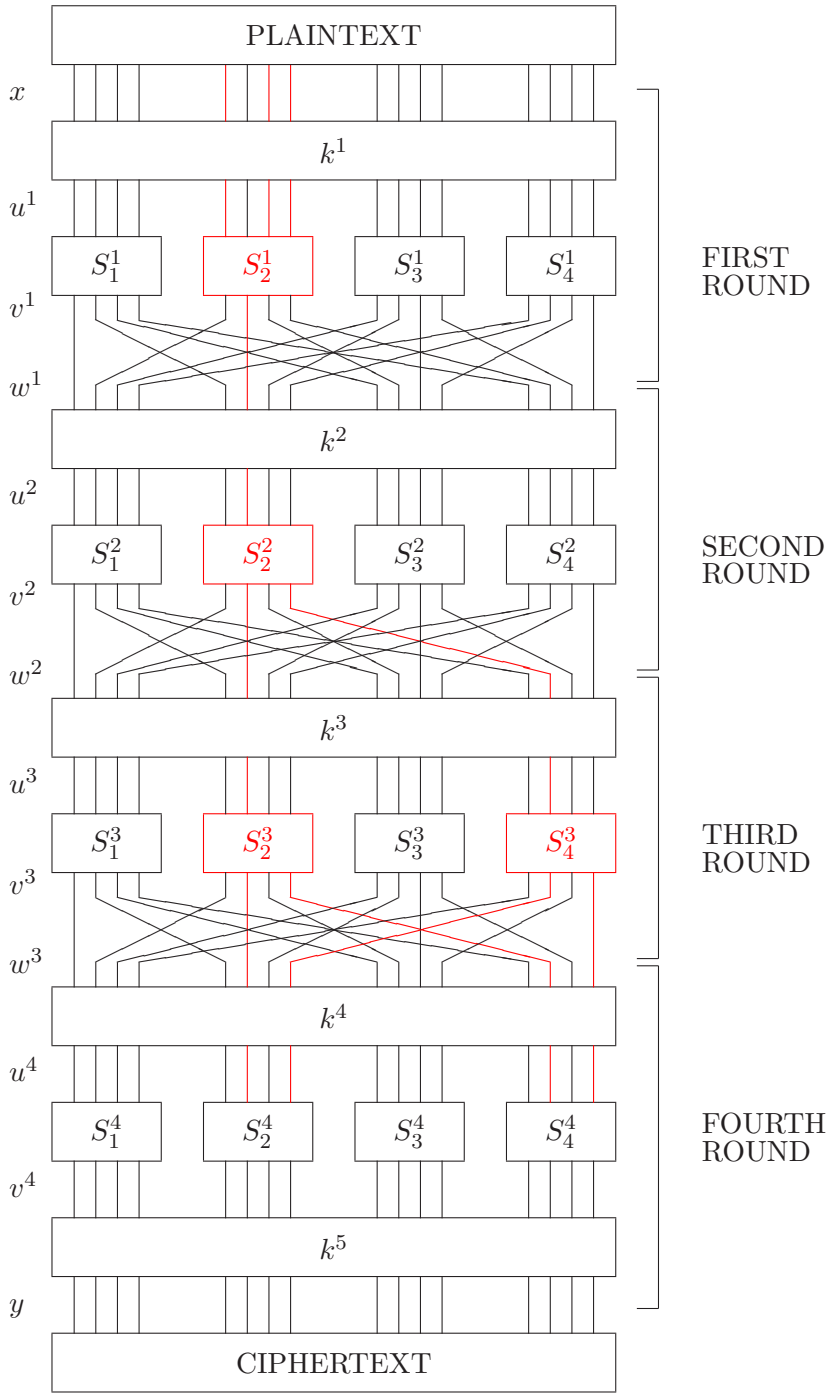


Abbildung 1.8: Zur Linearen Kryptoanalyse

Literaturverzeichnis

[BS93] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, Berlin, 1993.

[DR02] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer Verlag, New York, 2002.

[Ert01] W. Ertel. *Angewandte Kryptographie*. Vieweg, Braunschweig, 2001.

[Hey02] H. M. Heys. A tutorial on linear and differential cryptanalysis. *Cryptologia*, **XXVI**:189–221, 2002.

[Loi07] S. Loiperdinger. Linear and differential cryptanalysis and recent attacks on sha-1. Master's thesis, Universität Salzburg, 2007.

[MH81] R.C. Merkle and M.E. Hellman. On the security of multiple encryption. *Comm. ACM*, **24**:465–466, 1981.

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

[Nat01] National Institute of Standards and Technology (NIST). *Federal Information Processing Standards Publication (FIPS PUB) 197*, 2001. Available from `http://csrc.nist.gov/publications`.

[Sch96] B. Schneier. *Applied Cryptography*. Wiley, New York, second edition, 1996.

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *Bell Sys. Tech. J.*, 28:657–715, 1949.

[Sti06] D. R. Stinson. *Cryptography*. Chapman and Hall/CRC Press, Boca Raton, 3rd edition, 2006.

[TW06] W. Trappe and L. Washington. *Introduction to Cryptography with Coding Theory*. Pearson Prentice Hall, 2nd edition, 2006.