Bitblock-Verschlüsselung

Klaus Pommerening
Fachbereich Mathematik
der Johannes-Gutenberg-Universität
Saarstraße 21
D-55099 Mainz

7. April 1997, letzte Revision 6. Juli 2008

Für den Einsatz auf Computern braucht man Chiffrier-Algorithmen, die auf Bitblöcken operieren, schnell in der Anwendung – und damit auch für große Datenmengen und hohe Übertragungsgeschwindigkeiten geeignet – sind und die der Kryptoanalyse optimalen Widerstand leisten. In diesem Kapitel werden daher behandelt:

- Konstruktionsprinzipien für Bitblock-Chiffren –
  - Produkt-Chiffren, SP-Netze, Feistel-Netze, Nichtlinearität,
- exemplarische Chiffren von besonderer Bedeutung –
  - Lucifer, DES, AES,
- die wichtigsten kryptoanalytischen Ansätze –
  - algebraische, lineare und differenzielle Kryptoanalyse.

Ein mathematischer Anhang betrifft das Erkennen, Messen und Verhindern von Linearität: Verschlüsselungsfunktionen sollen „möglichst wenig linear“ sein. Aber was heißt das konkret?

1 Komposition von Chiffren

Ein erstes Konstruktionsprinzip für starke Chiffren, das auch in der klassischen Kryptographie oft angewendet wurde, ist die Nacheinander-Ausführung von Chiffrierschritten. Solche Verknüpfungen können zu beachtlicher Verstärkung der Sicherheit führen; selbstverständlich ist das aber nicht – und es stimmt auch nicht immer. In diesem Abschnitt werden einige grundlegende Überlegungen dazu vorgestellt.

Dabei wird unter einer

Mehrfach-Chiffre die Nacheinanderausführung des gleichen Verfahrens, aber mit verschiedenen Schlüsseln,

Kaskade die Nacheinanderausführung verschiedener Verfahren (auch Produktchiffre genannt)

verstanden.

1.1 Mehrfach-Chiffren und Gruppenstruktur

Mehrfach-Chiffren

Sei $F = (f_k)_{k\in K}$ eine Chiffre über dem Alphabet Σ, also $f_k\colon \Sigma^* \longrightarrow \Sigma^*$ die zugehörige Verschlüsselungsfunktion für jeden Schlüssel $k \in K$. Die gesamte Menge von Verschlüsselungsfunktionen wird mit

$$\tilde{F} = \{f_k \mid k \in K\} \subseteq \mathrm{Abb}(\Sigma^*, \Sigma^*)$$

bezeichnet.

Der Schlüsselraum wird wesentlich vergrößert, nämlich von K zu $K \times K$, durch die Bildung der **Zweifach-Chiffre**

$$F^{(2)} = (f_h \circ f_k)_{h,k\in K}$$

Natürlich kann man ebenso die Dreifach-Chiffre $F^{(3)}$, ..., die n-fach-Chiffre $F^{(n)}$ bilden. Sinnvoll ist das alles nur, wenn

(A) $\tilde{F}$ keine Halbgruppe ist.

Ist nämlich $\tilde{F}$ eine Halbgruppe, so gibt es zu zwei Schlüsseln $h, k \in K$ stets einen weiteren Schlüssel $x \in K$ mit $f_h \circ f_k = f_x$. Durch Komposition entstehen also keine neuen Verschlüsselungsfunktionen, sie ist eine „illusorische Komplikation“.

Noch besser ist, wenn

(B) $\tilde{F}$ eine möglichst große Unter-Halbgruppe von $\mathrm{Abb}(\Sigma^*, \Sigma^*)$ erzeugt.

Und das beste, was man hier erreichen kann, ist:

(C) Die Abbildung $K \times K \longrightarrow \widetilde{F^{(2)}} \subseteq \mathrm{Abb}(\Sigma^*, \Sigma^*)$ ist injektiv;

das kann man für einen endlichen Schlüsselraum K auch so ausdrücken:

(C') $\#\widetilde{F^{(2)}} = \#\{f_h \circ f_k \mid h, k \in K\} = (\#K)^2$.

Die Gruppen-Eigenschaft von Blockchiffren

Eine Blockchiffre ist durch die Wirkung auf einem Σ^r (für einen gegebenen Exponenten r) eindeutig festgelegt. (Um die Details der Fortsetzung auf Zeichenketten beliebiger Länge und des Auffüllens oder „Padding“ von zu kurzen Ketten auf Blocklänge kümmern wir uns im Moment nicht.)

Eine Blockchiffre heißt **längentreu**, wenn sie Σ^r in sich abbildet. Insbesondere ist dann $\tilde{F}$ auf natürliche Weise Teilmenge der symmetrischen Gruppe $\mathcal{S}(\Sigma^r)$, also endlich, und man kann den Schlüsselraum K ohne Einschränkung als endlich annehmen. Für solche Blockchiffren ist die Halbgruppen-Eigenschaft (also die Negation von (A) oben) zur Gruppen-Eigenschaft äquivalent. Das folgt aus dem bekannten einfachen:

Hilfssatz 1 *Sei G eine endliche Gruppe, $H \leq G$ eine Unter-Halbgruppe, d. h., $H \neq \emptyset$ und $HH \subseteq H$. Dann ist H Gruppe, insbesondere $\mathbf{1} \in H$.*

Beweis. Jedes $g \in G$ hat endliche Ordnung, $g^m = \mathbf{1}$. Ist nun $g \in H$, so $\mathbf{1} = g^m \in H$ und $g^{-1} = g^{m-1} \in H$. $\diamond$

Daher ist bewiesen:

Satz 1 *Sei F eine längentreue Blockchiffre über einem endlichen Alphabet. Dann sind folgende Aussagen äquivalent:*

(i) *Zu je zwei Schlüsseln $h, k \in K$ gibt es $x \in K$ mit $f_h \circ f_k = f_x$.*

(ii) *Die Menge $\tilde{F}$ der Verschlüsselungsfunktionen ist eine Gruppe.*

Anmerkung

Die Wahrscheinlichkeit, dass zwei zufällige Elemente der symmetrischen Gruppe $\mathcal{S}_n$ bereits die ganze Gruppe $\mathcal{S}_n$ oder wenigstens die alternierende Gruppe $\mathcal{A}_n$ erzeugen, ist

$$> 1 - \frac{2}{(\ln \ln n)^2} \quad \text{für große } n.$$

Für $n = 2^{64}$, einen typischen Wert bei Blockchiffren, ist diese untere Schranke ≈ 0.86. Eine nicht ganz ungeschickt gewählte Blockchiffre wird also sehr wahrscheinlich die volle oder wenigstens halbe Permutationsgruppe auf den Blöcken erzeugen. Trotzdem ist der konkrete Nachweis davon oft schwer. Jedenfalls kann man davon ausgehen, dass eine Mehrfach-Chiffre „in der Regel“ stärker als die Einfach-Chiffre ist.

Quelle: John Dixon, *The probability of generating the symmetric group.* Mathematische Zeitschrift 110 (1969), 199–205.

1.2 Beispiele für Mehrfach-Chiffren

Beispiele für Gruppen

Jeweils eine Gruppe bilden die folgenden längentreuen Blockchiffren:

- die Verschiebechiffren über Σ bezüglich einer Gruppenstruktur auf Σ,
- die monoalphabetischen Substitutionen über Σ,
- die Belaso-Chiffren fester Periode,
- die Block-Transpositionen fester Länge.

DES

DES ist eine Blockchiffre auf $\mathbb{F}_2^{64}$ mit Schlüsselraum $\mathbb{F}_2^{56}$. Campbell und Wiener haben gezeigt (Crypto 92), dass DES die alternierende Gruppe der Ordnung 2^{64} erzeugt.

Historische Beispiele

Die Komposition einer polyalphabetischen Chiffre der Periode l mit einer der Periode q hat die Periode $\mathrm{kgV}(l,q)$. **Anwendung:** Schlüsselerzeugermaschinen, wie sie in Kapitel I am Ende des Abschnitts über Chiffrierzylinder (I.4.8) kurz erwähnt wurden.

Weiteres historisches Beispiel: Die doppelte Spaltentransposition, die wesentlich schwerer zu brechen ist, als die einfache.

Komposition von Belaso-Chiffren

Die Komposition zweier Belaso-Chiffren der Perioden l und q hat zwar die Periode $\mathrm{kgV}(l,q)$, also im wesentlichen das Produkt lq, die Sicherheit entspricht aber höchstens der Summe $l+q$, wenn man einen Angriff mit bekanntem Klartext berücksichtigt:

Aus einem bekannten Klartext der Länge $l+q$ gewinnt man $l+q$ lineare Gleichungen für die ebensovielen Unbekannten, aus denen die beiden Schlüssel insgesamt bestehen. Sei dazu o. B. d. A. $l<q$. Dann haben wir die Situation

Klartext	a_0	a_1	$\ldots$	a_{l-1}	a_l	$\ldots$	a_{q-1}	$\ldots$
Schlüssel 1	h_0	h_1	$\ldots$	h_{l-1}	h_0	$\ldots$	$\ldots$	$\ldots$
Schlüssel 2	k_0	k_1	$\ldots$	k_{l-1}	k_l	$\ldots$	k_{q-1}	$\ldots$
Geheimtext	c_0	c_1	$\ldots$	c_{l-1}	c_l	$\ldots$	c_{q-1}	$\ldots$

Wir haben also eine Belaso-Chiffre mit dem Gesamtschlüssel

$$(h_0+k_0, h_1+k_1, \ldots)$$

und der Periode $\mathrm{kgV}(l, q)$.

Nehmen wir nun bekannten Klartext der Länge $l+q$ an, etwa o. B. d. A. $(a_0, \ldots, a_{l+q-1})$. Dann haben wir das folgende lineare Gleichungssystem für die $l+q$ Unbekannten $h_0, \ldots, h_{l-1}, k_0, \ldots, k_{q-1} \in \mathbb{Z}/n\mathbb{Z}$:

$$\begin{aligned}
h_0 + k_0 &= c_0 - a_0, \\
h_1 + k_1 &= c_1 - a_1, \\
&\vdots \\
h_{l-1} + k_{l-1} &= c_{l-1} - a_{l-1}, \\
h_0 + k_l &= c_l - a_l, \\
&\vdots \\
h_{l+q-1 \bmod l} + k_{l+q-1 \bmod q} &= c_{l+q-1} - a_{l+q-1}.
\end{aligned}$$

Dieses ist mit Sicherheit nicht eindeutig lösbar: Addiert man einen festen Wert x zu allen h_i und subtrahiert x von allen k_j, so hat man ebenfalls eine Lösung. Daher kann man zunächst der Einfachheit halber $h_0 = 0$ annehmen; sollten die Schlüssel nicht zufällig gewählt, sondern aus Schlüsselwörtern gebildet sein, kann man am Ende mit einer einfachen „CAESAR-Exhaustion“ wie ganz am Anfang von Kapitel I die „wahren“ Schlüssel ermitteln. Für die Dechiffrierung tun es auch die verschobenen Schlüssel. Und da wir eine Unbekannte weniger haben, reichen im allgemeinen sogar $l+q-1$ bekannte Klartext-Zeichen, um die übrigen $l+q-1$ Gleichungen eindeutig aufzulösen. Dies soll hier nicht weiter ausgeführt werden; der Interessierte möge die folgende Aufgabe lösen.

Übungsaufgabe

Gegeben sei der Geheimtext.

```
CIFRX KSYCI IDJZP TINUV GGKBD CWWBF CGWBC UXSNJ LJFMC
LQAZV TRLFK CPGYK MRUHO UZCIM NEOPP LK
```

Für einen Angriff mit bekanntem Klartext

- bestehe die berechtigte Vermutung, dass der Klartext mit „Sehr geehrter ...“ beginnt,
- seien mit Trial & Error plausible Schlüssellängen ausprobiert; es sei $42 = 6 \times 7$ an der Reihe, um zu testen, ob eine doppelte BELASO-Verschlüsselung mit den Schlüssellängen 6 und 7 vorliegt.

(Hierzu braucht man nicht alle Schlüssellängen und Kombinationen durchzuprobieren; die Koinzidenzanalyse ist wegen der Kürze des Textes nicht allzu spezifisch, reicht aber, um alle bis auf wenige plausible Schlüssellängen auszuschließen.)

1.3 Kryptoanalyse von Zweifach-Chiffren

Der Treffpunkt-Angriff

Diese Attacke auf Zweifach-Chiffren wurde 1981 von MERKLE und HELLMAN unter dem Namen „Meet in the Middle“ vorgestellt; sie ist nicht mit einem „Man in the Middle“-Angriff auf kryptographische Protokolle zu verwechseln, sondern ist eine Formalisierung eines Angriffs auf Mehrfach-Chiffren, der schon bei der Analyse von Rotormaschinen üblich war, siehe I.5.8.

Betrachtet wird die Komposition von zweimal der gleichen Chiffre mit verschiedenen Schlüsseln:

$$\begin{array}{ccccc} \Sigma^* & \xrightarrow{f_k} & \Sigma^* & \xrightarrow{f_h} & \Sigma^* \\ a & \mapsto & b & \mapsto & c. \end{array}$$

Sei ein bekanntes Klartext-Geheimtextpaar (a, c) gegeben. Dann bildet der Angreifer zwei Tabellen:

- alle $f_k(a)$, $k \in K$,
- alle $f_h^{-1}(c)$, $h \in K$,

und vergleicht diese. Jede Übereinstimmung ergibt ein mögliches Schlüsselpaar $(h, k) \in K^2$, das weiter getestet werden kann, etwa an einem weiteren bekannten Klartext.

Aufwand

Benötigt werden für diesen Angriff

- $2 \cdot \#K$ Verschlüsselungsoperationen (*nicht etwa* $(\#K)^2$!),
- $2 \cdot \#K$ Speicherplätze,

wobei die Zahl der nötigen Speicherplätze durch die Bemerkung halbiert wird, dass man nur eine der beiden Tabellen abspeichern muss.

Speichergrößen werden bekanntlich so bezeichnet:

2^{10}	2^{20}	2^{30}	2^{40}	2^{50}
Kilo	Mega	Giga	Tera	Peta

Dabei ist der Speicherbedarf – üblicherweise in Byte = Oktetten angegeben – noch mit der Größe eines Blocks des Verschlüsselungsverfahrens, etwa 64 Bit = 8 Byte, zu multiplizieren.

Man sieht, dass man schon mit 50-Bit-Schlüsseln in Größenbereiche kommt, die mit heutigen Speichertechniken nicht realisierbar sind. Da es bei der Kryptoanalyse allerdings mehr auf den Zeit- als auf den Speicherbedarf ankommt, ist die allgemeine Aussage gerechtfertigt:

Eine Zweifach-Chiffre ist nicht wesentlich sicherer als die zugrundeliegende Einfach-Chiffre. Insbesondere ist die Bitlänge für die exhaustive Schlüsselsuche bei weitem nicht verdoppelt sondern im wesentlichen nur um 1 Bit erhöht.

Fehlalarme

Eine Frage ist bei der Analyse offen geblieben: Wieviele der beim Tabellenabgleich gefundenen Übereinstimmungen führen zu einem falschen Schlüsselpaar? D. h., wie groß ist die Wahrscheinlichkeit eines Fehlalarms?

Dazu eine heuristische Überlegung: Gehen wir von einer Blockverschlüsselung von n-Bit-Blöcken mit l-Bit-Schlüsseln aus. Dann haben die Tabellen die Länge 2^l, es gibt also 2^{2l} Vergleiche. Da es 2^n verschiedene mögliche Werte gibt, kann man etwa $N_1 = 2^{2l-n}$ Übereinstimmungen erwarten. (Annahmen über die Zufälligkeit der Werte implizit. Die erste Übereinstimmung ist wegen des Geburtstagsphänomens nach etwa $2^{n/2}$ Versuchen zu erwarten, aber das nützt hier kaum.)

Probiert man die gefundenen Schlüsselpaare mit einem weiteren bekannten Klartext, so bleiben etwa $N_2 = N_1/2^n = 2^{2l-2n}$ Kandidaten übrig. Nach der Prüfung von insgesamt t bekannten Klartextblöcken kann man noch $N_t = 2^{2l-tn}$ Kandidaten erwarten – aber natürlich mindestens einen, nämlich den richtigen.

Eine eindeutige Lösung wird also im allgemeinen erreicht, wenn

$$t \geq \frac{2l}{n}.$$

Beispiele

1. DES, $n = 64$, $l = 56$: $N_1 = 2^{48}$, $N_2 = 2^{-16}$. *Es werden ungefähr* 2 *bekannte Klartextblöcke benötigt.*

2. IDEA, $n = 64$, $l = 128$: $N_1 = 2^{192}$, $N_2 = 2^{128}$, $N_3 = 2^{64}$, $N_4 = 1$. *Es werden ungefähr* 4 *bekannte Klartextblöcke benötigt.*

3. AES, $n = 128$, $l = 128$: $N_1 = 2^{128}$, $N_2 = 1$. *Es werden ungefähr* 2 *bekannte Klartextblöcke benötigt.* Allerdings ist wegen $\#K = 2^{128}$ die Zahl der benötigten Speicherplätze hier sehr weit außerhalb der Möglichkeiten (wie auch bei Beispiel 2).

Time-Memory-Tradeoff

Eine allgemeinere Überlegung führt zu einer Ausbalancierung von Zeit und Speicherplatz („Time-Memory-Tradeoff“): Man kann bei dem Treffpunkt-Angriff Speicherplätze auf Kosten von Rechenzeit sparen, wenn man nur Teiltabellen anlegt:

Hält man in einem Durchgang jeweils s Bits von h und k fest, so benötigt man jeweils 2^{l-s} Speicherplätze für die Tabellen der $f_k(a)$ bzw. $f_h^{-1}(c)$. Zur Kompensation muss man 2^{2s} solche Durchgänge mit je einem Tabellenpaar-Abgleich machen. Der Aufwand beträgt:

$2 \cdot 2^{l-s}$	Verschlüsselungsoperationen für ein Tafelpaar,
2^{2s}	Tafelpaar-Abgleiche, also insgesamt
$2 \cdot 2^{l+s}$	Verschlüsselungsoperationen,
$2 \cdot 2^{l-s}$	Speicherplätze.

Das Produkt aus der Anzahl der Verschlüsselungsoperation und der benötigten Speicherplätze ist $4 \cdot 2^{2l}$, unabhängig von s. *Der Angreifer kann also seine Ressourcen flexibel einsetzen.*

Beispiel DES: Hat der Angreifer 128 Terabyte Speicher zur Verfügung, so kann er 2 Tabellen von je 2^{40} Blöcken anlegen, also $s = 56 - 40 = 16$ wählen. Er benötigt dann insgesamt $2 \cdot 2^{72}$ Verschlüsselungsoperationen. Das liegt für den größten Geheimdienst der Welt zweifellos im Bereich des Machbaren.

Fazit: *Zweifach-Chiffren tragen zur Erhöhung der Sicherheit nicht lohnenswert bei.*

1.4 Dreifach-Chiffren

Die im vorigen Abschnitt gezeigte Schwäche der Zweifach-Chiffren führt dazu, dass man, wenn man eine Verstärkung braucht, zu Dreifach-Chiffren übergeht. Üblicherweise verwendet man das „EDE-Schema“ (Encryption, Decryption, Encryption)

$$f_g \circ f_h^{-1} \circ f_k \quad \text{für } g, h, k \in K.$$

Es hat den Vorteil, dass man mit der Schlüsselwahl $g = h = k$ die Kompatibilität mit der Einfach-Chiffre hat.

Der Treffpunkt-Angriff kann hier natürlich auch durchgeführt werden und ergibt, dass die Bitlänge für die vollständige Suche gegenüber der Einfach-Chiffre zwar nicht verdreifacht, aber doch immerhin (mehr als) verdoppelt ist.

Üblich ist auch ein vereinfachtes Schema: die Dreifach-Verschlüsselung mit zwei Schlüsseln:

$$f = f_k \circ f_h^{-1} \circ f_k \quad \text{für } h, k \in K.$$

Dieses Schema hat eine Schwäche bei einem Angriff mit gewähltem Klartext, die allerdings nur für Paranoiker bedenklich ist. Die Situation sei

$$\begin{array}{ccccccc} \Sigma^* & \xrightarrow{f_k} & \Sigma^* & \xrightarrow{f_h^{-1}} & \Sigma^* & \xrightarrow{f_k} & \Sigma^*, \\ a & \mapsto & b & \mapsto & b' & \mapsto & c. \end{array}$$

Schritt 1: Mit $\#K$ Verschlüsselungsschritten und $\#K$ Speicherplätzen wird für jeden Zwischenwert b_0 die Tabelle

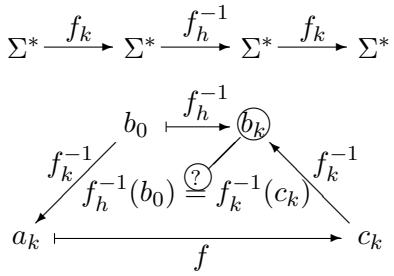
$$\{f_h^{-1}(b_0) \mid h \in K\}$$

vorausberechnet.

Schritt 2: Dann wird für alle Schlüssel $k \in K$ berechnet:

$$\begin{aligned} a_k &:= f_k^{-1}(b_0), \\ c_k &:= f(a_k), \\ b_k &:= f_k^{-1}(c_k); \end{aligned}$$

die zweite Zuweisung ist möglich, weil wir einen Angriff mit gewähltem Klartext durchführen, d. h., wir können f auf beliebige Klartexte anwenden. Das ganze benötigt $5 \cdot \#K$ einfache Verschlüsselungsschritte. Falls $b_k = f_k^{-1}(b_0)$, ist ein Kandidaten-Schlüsselpaar (h, k) gefunden, das weiter untersucht wird.



Der effizienteste bekannte Angriff wird beschrieben in:

- Van Oorschot/Wiener: A known plaintext attack on two-key triple encryption. Eurocrypt 90.

1.5 Kaskaden verschiedener Chiffren

Beispiele

1. Monoalphabetische Substitutionen und Transpositionen sind vertauschbar. Eine Komposition von mehr als jeweils einer solchen Operation bringt also nichts, da sie einzeln für sich je eine Gruppe bilden. Eine Komposition aus einer monoalphabetischen Substitution und einer Transposition ist noch ziemlich leicht sogar ohne bekannten Klartext zu brechen, da die wichtigsten Buchstaben aufgrund ihrer Häufigkeiten erkannt werden.

2. Auch für periodische polyalphabetische Chiffren und Transpositionen gilt diese Bemerkung – es sein denn, man ändert die Periodenlänge nach jedem Schritt. Hier kann man schon recht komplizierte Chiffren erzeugen.

3. Die Enigma führte die Komposition aus einer monoalphabetischen, einigen polyalphabetischen verschiedener Periode und nochmal einer monoalphabetischen Substitution aus. Das ganze war zusammengenommen auch wieder nur eine polyalphabetische Substitution sehr langer Periode.

4. Die ADFGVX-Chiffre der deutschen Wehrmacht im 1. Weltkrieg bestand aus einer Substitution gefolgt von einer Spaltentransposition; die Substitution wurde ausgeführt, indem die 26 Buchstaben und 10 Ziffern in ein 6-mal-6-Quadrat nach Vorgabe eines Schlüssels geschrieben wurden und jedes Zeichen durch seine Koordinaten in diesem Quadrat ersetzt wurde, welche mit A, D, F, G, V, X bezeichnet waren. Es gelang den Franzosen (Painvin und Givierge) zum großen Teil, diese Chiffre zu brechen.

5. Die Komposition von monoalphabetischer Chiffre und einer Autokey-Chiffre werden wir bald als „Betriebsmodus bei Blockverschlüsselung“ kennen lernen. Die Autokey-Chiffre erschwert einige Angriffe, erhöht aber insgesamt die Sicherheit nur unwesentlich.

6. Ferner sei noch einmal daran erinnert, dass die Drehscheiben-Chiffre nach Porta sich als Komposition einer monoalphabetischen Substitution und einer Belaso-Chiffre darstellen ließ – sogar in beiden möglichen Reihenfolgen.

Insgesamt kann man sagen, dass Kaskaden verschiedener Chiffren oft zu Verbesserungen der Sicherheit führen, aber durchaus nicht immer. Es ist in jedem Fall eine sorgfältige Analyse der entstehenden Produkt-Chiffre nötig, bevor man ihr vertraut.

2 Bitblock-Chiffren und Feistel-Netze

In diesem Abschnitt werden grundlegende Überlegungen zu Bitblock-Chiffren sowie erste Konstruktions-Ansätze vorgestellt.

2.1 Bitblockchiffren – Einleitung

Beschreibung

Bitblock-Chiffren arbeiten über dem Alphabet $\Sigma = \mathbb{F}_2 = \{0, 1\}$ und verschlüsseln zunächst Blöcke fester Länge längentreu; sie sind also auf dem Vektorraum $\mathbb{F}_2^n$ definiert mit Schlüsselraum $\mathbb{F}_2^l$. Die Fortsetzung auf Bitketten beliebiger Länge ist Thema des Abschnitts „Betriebsarten“ und kümmert uns vorläufig nicht, ebensowenig wie die Frage, wie man zu kurze Blöcke auffüllt („Padding“).

Achtung: Der Buchstabe n bezeichnet jetzt bis auf weiteres nicht mehr die Größe des Alphabets, sondern die Länge der Bitblöcke.

Man kann eine Bitblock-Chiffre auch als monoalphabetisch über dem Alphabet $\Sigma' = \mathbb{F}_2^n$ ansehen. Zur Konstruktion und Beschreibung wird meist die algebraische Struktur als n-dimensionaler Vektorraum über dem Körper $\mathbb{F}_2$ verwendet, gelegentlich die als Körper $\mathbb{F}_{2^n}$, nur selten die als zyklische Gruppe der Ordnung 2^n.

Eine solche Chiffre wird also beschrieben durch eine Abbildung

$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$$

bzw. als Familie $(F_k)_{k\in K}$ von Abbildungen

$$F_k\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n \quad \text{für alle } k \in K = \mathbb{F}_2^l$$

mit $F_k(a) = F(a, k)$.

Wahl der Blocklänge

Die Blocklänge soll groß genug sein, um die Verfahren zum Brechen monoalphabetischer Substitutionen, insbesondere Muster- und Häufigkeitsanalysen, unmöglich zu machen; besser ist es, jede Art von Informationspreisgabe über den Klartext, z. B. jede Wiederholung im Geheimtext, zu vermeiden.

Bei einer **Codebuch-Attacke** würde der Angreifer bekannte Klartext-Geheimtextpaare bei festem Schlüssel sammeln, sich also sozusagen sein eigenes Codebuch anlegen. Ein vollständiges Codebuch führt, selbst wenn der Schlüssel sich daraus nicht bestimmen lässt, zu einer vollständigen Entschlüsselung. Konsequenzen:

- $\#\Sigma' = 2^n$ sollte größer als die Zahl der im ungünstigsten Fall verfügbaren Speicherplätze sein.
- Der Schlüssel sollte oft genug gewechselt werden.

Wegen des Geburtstagsphänomens sind allerdings strengere Kriterien sinnvoll: Falls der Gegner ca. $\sqrt{\#\Sigma'} = 2^{n/2}$ Klartext-Geheimtextpaare in seinem Codebuch gesammelt hat, ist die Wahrscheinlichkeit einer Kollision schon etwa $\frac{1}{2}$. Daher sollte diese Zahl, also $2^{n/2}$ bei einer Bitblock-Chiffre, schon die Zahl der verfügbaren Speicherplätze überschreiten; und auch Schlüssel sollten oft genug gewechselt werden – deutlich vor dieser Anzahl verschlüsselter Blöcke.

Die bisher meist verwendete Blocklänge 64 ist so gesehen also schon bedenklich; sie ist allenfalls noch bei häufigem Schlüsselwechsel zu rechtfertigen. Besser ist eine Blocklänge von 128 Bit, wie sie auch im neuen Standard AES vorgesehen ist.

Diese Überlegung ist ein typische Beispiel für die Sicherheitsabwägungen in der modernen Kryptographie: Es wird mit breiten Sicherheitsabständen gearbeitet; erkennbare Schwächen werden vermieden, selbst wenn sie noch weit von einer praktischen Auswertbarkeit für den Gegner entfernt sind. Da es effiziente Verfahren gibt, die diese Sicherheitsabstände einhalten, besteht überhaupt keine Notwendigkeit, weniger strenge Verfahren einzusetzen.

2.2 Polynome über endlichen Körpern

Satz 1 *Sei K ein endlicher Körper mit q Elementen und $n \in \mathbb{N}$. Dann wird jede Funktion $F : K^n \longrightarrow K$ durch ein Polynom $\varphi \in K[T_1, \ldots, T_n]$ vom partiellen Grad $\leq q-1$ in jedem T_i beschrieben.*

Beweis. (Skizze) folgt unten. $\diamond$

Korollar 1 *Seien $m, n \in \mathbb{N}$. Dann wird jede Abbildung $F : K^n \longrightarrow K^m$ durch ein m-Tupel $(\varphi_1, \ldots, \varphi_m)$ von Polynomen $\varphi_i \in K[T_1, \ldots, T_n]$ vom partiellen Grad $\leq q-1$ in jedem T_i beschrieben.*

Korollar 2 *Jede Abbildung $F : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ wird durch ein m-Tupel von Polynomen $(\varphi_1, \ldots, \varphi_m)$ von Polynomen $\varphi_i \in \mathbb{F}_2[T_1, \ldots, T_n]$ beschrieben, deren sämtliche partiellen Grade ≤ 1 sind.*

Damit erhalten wir auch die **algebraische Normalform** einer Boolschen Funktion $F : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$: Für eine Teilmenge $I = \{i_1, \ldots, i_r\} \subseteq \{1, \ldots, n\}$ sei x^I das Monom

$$x^I = x_{i_1} \cdots x_{i_r}.$$

Dann lässt sich F eindeutig schreiben als

$$F(x_1, \ldots, x_n) = \prod_I a_I x^I \quad \text{mit } a_I = 0 \text{ oder } 1.$$

Insbesondere bilden die 2^n Monome x^I eine Basis des $\mathbb{F}_2$-Vektorraums $\mathrm{Abb}(\mathbb{F}_2^n, \mathbb{F}_2)$, und es gibt – wie auch anderweitig klar ist – 2^{2^n} solche Funktionen.

Beweisskizze für den Satz – ein etwas elementarerer, vollständiger Beweis im Fall $K = \mathbb{F}_2$ wird im Abschnitt „Die algebraische Normalform Boolescher Funktionen“ gegeben.

Sei zunächst K ein beliebiger (kommutativer) Körper. Dann ist $A := \mathrm{Abb}(K^n, K)$ eine K-Algebra. Sei $K[T]$ der Polynomring in dem n-Tupel $T = (T_1, \ldots, T_n)$ von Unbestimmten. Dann ist

$$\begin{array}{rccl} \alpha \colon K[T] & \longrightarrow & A, & \\ \varphi & \mapsto & \alpha(\varphi) & \text{mit } \alpha(\varphi)(x_1, \ldots, x_n) := \varphi(x_1, \ldots, x_n) \end{array}$$

ein K-Algebra-Homorphismus, der „Einsetzungs-Homomorphismus“. Sein Bild, $\mathrm{Bild}\,\alpha \subseteq A$, ist die Algebra der Polynomfunktionen. Es gibt zwei grundsätzlich unterschiedliche Fälle:

Fall 1: K ist unendlich. Dann ist α

- injektiv, d. h., unterschiedliche Polynome definieren unterschiedliche Funktionen – der Beweis ist der Eindeutigkeitsbeweis von Interpolationsformeln –,
- nicht surjektiv, denn $K[T]$ hat die gleiche Mächtigkeit wie K, dagegen hat A eine echt größere – der Beweis ist elementare Mengenlehre –.

Fall 2: K ist endlich. Dann ist α

- nicht injektiv, denn $K[T]$ ist unendlich, aber $\#A = q^{q^n}$,
- surjektiv, denn $F \in A$ wird vollständig beschrieben durch die q^n Paare $(x, F(x))$, $x \in K^n$, also durch den Graphen; mit Interpolation findet man ein Polynom $\varphi \in K[T]$ mit $\varphi(x) = F(x)$ für alle $x \in K^n$, d. h., $\alpha(\varphi) = F$.

Damit ist der erste Teil des Satzes bewiesen: *Jede Funktion $K^n \to K$ ist Polynomfunktion.*

Für den weiteren Beweisgang bestimmt man am besten $\operatorname{Kern}\alpha$. Sei $\mathfrak{a}$ das von den Polynomen $T_i^q - T_i$ erzeugte Ideal:

$$\mathfrak{a} = (T_1^q - T_1, \ldots, T_n^q - T_n) \trianglelefteq K[T].$$

Da die multiplikative Gruppe $K^\times$ die Ordnung $q-1$ hat, ist $a^q = a$ für alle $a \in K$. Also ist $\mathfrak{a} \subseteq \operatorname{Kern}\alpha$.

Nun hat der Restklassenring $K[T]/\mathfrak{a}$ offensichtlich ein vollständiges Repräsentantensystem aus den Restklassen derjenigen Polynome, die in allen T_i den Grad $\leq q-1$ haben, und die bilden einen K-Vektorraum der Dimension $\leq q^n$, da die Monome ihn aufspannen. Also ist

$$q^{q^n} = \#A = \#(K[T]/\operatorname{Kern}\alpha) \leq \#(K[T]/\mathfrak{a}) \leq q^{q^n}.$$

Da folglich überall in dieser Kette die Gleichheit gilt, ist $\operatorname{Kern}\alpha = \mathfrak{a}$, und $A \cong K[T]/\mathfrak{a}$ wird daher von den Polynomen mit allen partiellen Graden $\leq q-1$ ausgeschöpft. $\diamond$

2.3 Algebraische Kryptoanalyse

Der Angriff mit bekanntem Klartext

Sei (wie hier üblich) eine Bitblock-Chiffre durch eine Abbildung

$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$$

beschrieben. Dann ist F ein n-Tupel $F = (F_1, \dots F_n)$ von Polynomfunktionen in $n + l$ Unbestimmten, deren sämtliche partiellen Grade ≤ 1 sind.

Ein Angriff mit bekanntem Klartext $a \in \mathbb{F}_2^n$ und Geheimtext $c \in \mathbb{F}_2^n$ ergibt ein Gleichungssystem

$$F(a, x) = c$$

von n Polynomgleichungen für den unbekannten Schlüssel $x \in \mathbb{F}_2^l$.

Solche Gleichungssysteme (über beliebigen Körpern) sind Gegenstand der Algebraischen Geometrie. Eine Faustregel besagt

Die Lösungsmenge für x ist „im allgemeinen klein“, wenn $n \geq l$.

(Andernfalls braucht man mehrere bekannte Klartextblöcke.)

Die allgemeine Theorie hierzu ist hochkompliziert, insbesondere, wenn man konkrete Lösungsverfahren haben will. Aber vielleicht hilft die Beobachtung, dass man nur partielle Grade ≤ 1 benötigt?

Beispiele

Beispiel 1, Linearität: Ist F eine *lineare* Abbildung, so ist das Gleichungssystem mit den Methoden der Linearen Algebra effizient lösbar (n lineare Gleichungen in l Unbekannten). Es reicht dazu schon, wenn F linear in x ist.

Beispiel 2: Sei $n = l = 2$, $F(T_1, T_2, X_1, X_2) = (T_1 + T_2X_1, T_2 + T_1X_2 + X_1X_2)$, $a = (0, 1)$, $c = (1, 1) \in \mathbb{F}_2^2$. Dann sieht das Gleichungssystem für den Schlüssel $(x_1, x_2) \in \mathbb{F}_2^2$ so aus:

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 + x_1 \\ 1 + 0 + x_1x_2 \end{pmatrix},$$

die Lösung ist offensichtlich $x_1 = 1$, $x_2 = 0$.

Substitution: Dass man Polynomgleichungen nicht immer auf den ersten Blick ihre Komplexität ansieht, zeigt das Beispiel (über $\mathbb{F}_2$)

$$x_1x_2x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_2 + x_3 = 0.$$

Es geht durch die Substitutionen $x_i = u_i + 1$ über in

$$u_1u_2u_3 + u_1 = 0$$

(umgekehrt sieht man das leichter) mit der Lösungsmenge

$$u_1 = 0,\ u_2, u_3 \text{ beliebig } \textit{oder } u_1 = u_2 = u_3 = 1.$$

Die vollständige Lösung der ursprünglichen Gleichung ist also

$$x_1 = 1,\ x_2, x_3 \text{ beliebig } \textit{oder } x_1 = x_2 = x_3 = 0.$$

Die Komplexität des Angriffs

Was in den Beispielen so einfach war, ist im allgemeinen zu komplex:

Satz 2 (Garey/Johnson) *Das Problem, eine gemeinsame Nullstelle eines Polynomsystems $f_1, \ldots, f_r \in \mathbb{F}_2[T_1, \ldots T_n]$ zu finden, ist* **NP***-vollständig.*

Beweis. Siehe das Buch von Garey/Johnson. ◇

Der Begriff „**NP**-vollständig“ wird später in der Vorlesung erklärt (siehe III.7).

Deutung: Bei günstig gewählter Blockverschlüsselungsfunktion $F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$ ist der Angriff mit bekanntem Klartext nicht effizient durchführbar.

Offene Probleme

Im Grunde besagt der Satz für die praktische Anwendung *gar nichts*:

1. Er bezieht sich nur auf den Fall eines Algorithmus für *beliebige* Polynome. Er macht keine Aussage für ein bestimmtes Polynomsystem.
2. Selbst wenn er das machen würde, wäre immer noch kein konkretes Beispiel eines „schwierigen“ Polynomsystems bekannt.
3. Und selbst dann würde der Satz nichts darüber sagen, ob nur einzelne, wenige Instanzen des Problems schwierig sind oder – was der Kryptologe eigentlich braucht – fast alle.

Weitere Hinweise auf die Schwierigkeit, Polynomgleichungen zu lösen, gibt der Artikel

- D. Castro, M. Giusti, J. Heintz, G. Matera, L. M. Pardo: The hardness of polynomial equation solving. Found. Comput. Math. 3 (2003), 347–420. (Für Uni-Mainz online über die EZB zugänglich.)

Interpolationsangriff

Eine Variante der algebraischen Kryptoanalyse mit bekanntem Klartext ist der Interpolationsangriff, der in

- Thomas Jakobsen, Lars R. Knudsen: The interpolation attack on block ciphers, FSE 1997,

vorgestellt wurde. Die Idee ist ganz einfach: Der Vektorraum $\mathbb{F}_2^n$ kann bei Wahl einer geeigneten Multiplikation als endlicher Körper $K = \mathbb{F}_{2^n}$ der Charakteristik 2 interpretiert werden. Bei festem Schlüssel $k \in \mathbb{F}_2^l$ ist die Bitblock-Chiffre dann einfach eine Funktion $F_k \colon K \longrightarrow K$, also ein Polynom. Ist d sein Grad, so kann es nach der Interpolationsformel aus $d+1$ bekannten Klartext-Blöcken bestimmt werden; gleiches gilt für die Umkehrfunktion. Damit kann dann ver- bzw. entschlüsselt werden, ohne dass der Schlüssel explizit bestimmt wurde.

Um diesen Angriff zu verhindern, muss man also darauf achten, dass Ver- und Entschlüsselungsfunktion auf K bei festem Schlüssel stets einen hohen Grad besitzen; das ist machbar, da ja Grade bis $2^n - 1$ möglich sind.

Allerdings funktioniert der Angriff auch bei hohem Grad, wenn das jeweilige Polynom nur wenige Koeffizienten $\neq 0$ hat, also „dünn besetzt“ ist. Daher muss auch dieses vermieden werden.

Linearisierung überbestimmter Gleichungssysteme

Gleichungssysteme höherer Ordnung kann man manchmal brechen, wenn sie so weit überbestimmt sind, dass man die Monome (oder einige davon) als eigene Unbekannte ansehen kann. Dies wird durch das folgende einfache Beispiel illustriert:

$$\begin{aligned} x^3 + xy + y^5 &= 1, \\ 2x^3 - xy &= 0, \\ xy + 3y^5 &= 3. \end{aligned}$$

Hier substituiert man alle vorkommenden Monome: $u := x^3$, $v := xy$, $w := y^5$, und erhält das lineare Gleichungssystem

$$\begin{aligned} u + v + w &= 1, \\ 2u - v &= 0, \\ v + 3w &= 3. \end{aligned}$$

aus drei Gleichungen mit drei Unbekannten. Die (in diesem Fall sehr einfach auch manuell zu erhaltende) Lösung ist $u = 0$, $v = 0$, $w = 1$; sie ist eindeutig, wenn wir einen Körper der Charakteristik $\neq 7$ annehmen. Daraus ergibt sich als vollständige Lösung des ursprünglichen Systems: $x = 0$, $y = 1$ oder irgendeine im Körper enthaltene 5. Einheitswurzel.

Dieser Angriff wurde im Jahre 2002 populär, als das Gerücht um die Welt lief, der neue AES sei für diesen Angriff anfällig. Bei genauem Hinsehen ließ sich das aber nicht bestätigen; die einzelnen Gleichungen des konstruierten (riesigen) linearen Gleichungssystems waren bei weitem nicht unabhängig.

2.4 SP-Netze

Shannons Konstruktionsprinzipien

Nach Shannon sollen Blockchiffren folgendes leisten:

Diffusion (Durchmischung): Die Bits des Klartextblocks werden über den gesamten Block „verschmiert“. Quantitativ ausdrücken kann man das durch den „Lawinen-Effekt“ (englisch: avalanche effect):

- Jedes Bit des Geheimtextblocks hängt von jedem Bit des Klartextblocks ab.
- Bei Änderung eines Klartextbits ändern sich ca. 50% der Geheimtextbits.

Grundbausteine zur Erreichung von Diffusion sind Transpositionen.

Konfusion (Komplexität des Zusammenhangs): Die Beziehung zwischen Klartextblock und Geheimtextblock soll möglichst kompliziert sein (insbesondere hochgradig nichtlinear). Ähnlich komplex soll auch die Abhängigkeit des Geheimtextblocks vom Schlüssel sein, insbesondere sollen sich bei Änderung eines Schlüsselbits möglichst viele Geheimtextbits ändern, und zwar möglichst unvorhersagbar. Es soll für den Angreifer unmöglich sein zu erkennen, dass er einen Schlüssel „fast“ richtig geraten hat.

Grundbausteine hierfür sind vor allem Substitutionen.

Produktchiffren nach Shannon

Shannon schlug als Konstruktionsprinzip für starke Blockchiffren vor, Produktchiffren aus einer wechselnden Folge von **S**ubstitutionen und Transpositionen (= **P**ermutationen) zu bilden – sogenannte **SP-Netze**. Im einfachsten Fall sieht das so aus:

$$\begin{aligned} \mathbb{F}_2^n \xrightarrow{S_1(\bullet,k)} \mathbb{F}_2^n \xrightarrow{P_1(\bullet,k)} \mathbb{F}_2^n \longrightarrow \ldots \\ \ldots \longrightarrow \mathbb{F}_2^n \xrightarrow{S_r(\bullet,k)} \mathbb{F}_2^n \xrightarrow{P_r(\bullet,k)} \mathbb{F}_2^n \end{aligned}$$

abhängig von einem Schlüssel $k \in \mathbb{F}_2^l$. Dabei ist

$$\begin{aligned} S_i &= i\text{-te Substitution}, \\ P_i &= i\text{-te Permutation}, \\ P_i \circ S_i &= i\text{-te } \textbf{Runde} \text{ –} \end{aligned}$$

wobei insgesamt r Runden nacheinander ausgeführt werden.

Beispiel: Lucifer I (Feistel 1973).

2.5 FEISTEL-Chiffren

Die Kernabbildung

Die Blockgröße $n = 2s$ wird als gerade vorausgesetzt. Blöcke $a \in \mathbb{F}_2^n$ werden in ihre linke und rechte Hälfte zerlegt:

$$a = (L, R) \in \mathbb{F}_2^s \times \mathbb{F}_2^s$$

(groß geschrieben, um die Verwechslung mit der Dimension l des Schlüsselraums zu vermeiden). Hierfür muss man sich auf eine Nummerierung der Bits in einem Block einigen:

- In der **natürlichen Nummerierung**, die auch hier meistens verwendet wird, steht das LSB (Least Significant Bit) immer rechts und trägt die Nummer 0, das MSB (Most Significant Bit) steht links und trägt die Nummer $n - 1$:

  $$b = (b_{n-1}, \ldots, b_0) \in \mathbb{F}_2^n.$$

  Dies entspricht der Darstellung natürlicher Zahlen im ganzzahligen Intervall $[0 \ldots 2^n[$ zur Basis 2:

  $$b_{n-1} \cdot 2^{n-1} + \cdots b_1 \cdot 2 + b_0 \in \mathbb{N}.$$

- In der **IBM-Nummerierung** stehen die Bits umgekehrt und werden von 1 bis n nummeriert:

  $$a = (a_1, \ldots, a_n) \in \mathbb{F}_2^n.$$

  Dies entspricht der üblichen Nummerierung der Komponenten von Vektoren eines Vektorraums. Manchmal wird auch 0 bis $n - 1$ nummeriert.

Eine FEISTEL-Chiffre beruht auf einer **Kernabbildung**

$$f \colon \mathbb{F}_2^s \times \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^s,$$

an die formal keine weiteren Anforderungen gestellt werden; insbesondere brauchen die $f(\bullet, k)$ nicht bijektiv zu sein.

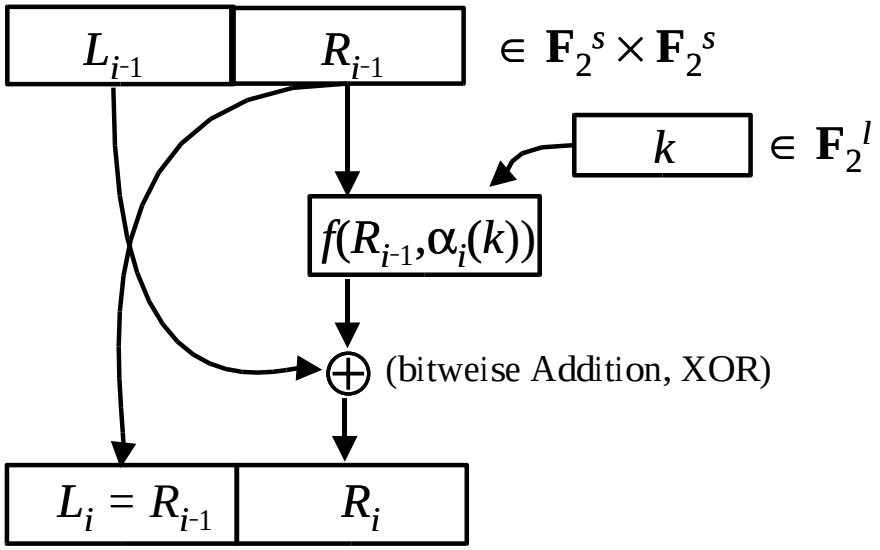
> Um ein brauchbares Verschlüsselungsverfahren zu erhalten, fordert man jedoch, dass f möglichst gute Konfusion und Diffusion bietet, etwa bereits aus Substitutionen und Transpositionen zusammengesetzt und hochgradig nichtlinear ist.

Beschreibung der Runden

Eine FEISTEL-Chiffre besteht aus r Runden, wobei jeweils aus dem Schlüssel $k \in \mathbb{F}_2^l$ ein q-Bit-Rundenschlüssel gebildet wird mit Hilfe der i-ten **Schlüsselauswahl**

$$\alpha_i \colon \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^q \quad \text{für } i = 1, \ldots, r.$$

Die i-te Runde soll dann so aussehen:



Man erkennt in der Addition der linken Hälfte auf die transformierte rechte eine Spur des Autokey-Prinzips.

Algorithmische Beschreibung

Aus der graphischen Beschreibung leitet man leicht eine algorithmische ab:

$$\begin{array}{lrcl}
\textbf{Input} \longrightarrow & a & = & (a_0, a_1) \in \mathbb{F}_2^s \times \mathbb{F}_2^s \\
& a_2 & := & a_0 + f(a_1, \alpha_1(k)) \\
& & & \quad - \text{1. Runde, Ergebnis } (a_1, a_2) \\
& \vdots & & \vdots \\
& a_{i+1} & := & a_{i-1} + f(a_i, \alpha_i(k)) \\
& & & \quad - i\text{-te Runde, Ergebnis } (a_i, a_{i+1}) \\
& & & \quad - [a_i = R_{i-1} = L_i,\ a_{i+1} = R_i] \\
& \vdots & & \vdots \\
\textbf{Output} \longleftarrow & c & = & (a_r, a_{r+1}) =: F(a, k)
\end{array}$$

Die Entschlüsselung

Entschlüsselt wird nach der Formel

$$a_{i-1} = a_{i+1} + f(a_i, \alpha_i(k)) \quad \text{für } i = 1, \ldots, r.$$

Das entspricht dem gleichen Algorithmus, nur werden die Runden in umgekehrter Reihenfolge durchlaufen – mit anderen Worten: Die Schlüsselauswahl wird in umgekehrter Reihenfolge ausgeführt.

Insbesondere ist damit bewiesen:

Satz 3 (FEISTEL) *Sei* $F\colon \mathbb{F}_2^{2s} \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^{2s}$ *die Blockabbildung zur Kernabbildung* $f : \mathbb{F}_2^s \times \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^s$ *und zur Schlüsselauswahl* $\alpha = (\alpha_1, \ldots, \alpha_r)$, $\alpha_i\colon \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^q$.

Dann ist die Verschlüsselungsfunktion $F(\bullet, k) : \mathbb{F}_2^{2s} \longrightarrow \mathbb{F}_2^{2s}$ *für jeden Schlüssel* $k \in \mathbb{F}_2^l$ *bijektiv.*

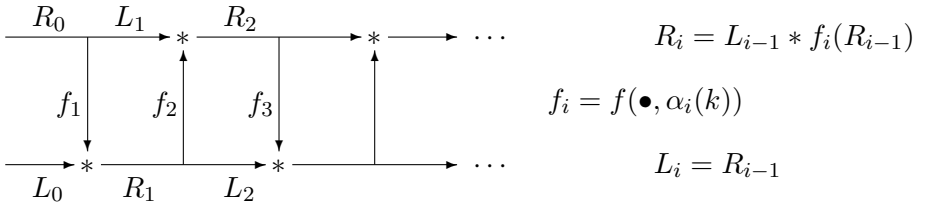
Zusatz. *Die Entschlüsselung ist die Blockabbildung zur gleichen Kernabbildung* f *und zur umgekehrten Schlüsselauswahl* $(\alpha_r, \ldots, \alpha_1)$.

Achtung: Beginnt man die Entschlüsselung mit $c = (a_r, a_{r+1})$, so sind zuerst *die Seiten zu vertauschen*, denn der Algorithmus beginnt mit (a_{r+1}, a_r). Daher wird bei der letzten Runde einer FEISTEL-Chiffre meist die Seitenvertauschung weggelassen.

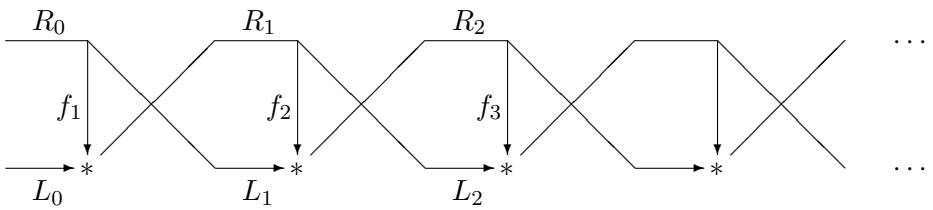
Anmerkungen

- Sind f und die α_i linear, so auch F.
- Für die α_i nimmt man meist nur eine Bitauswahl, also eine Projektion $\mathbb{F}_2^l \longrightarrow \mathbb{F}_2^q$.
- Alternative graphische Beschreibungen:

a) Leiter



b) Verdrehte Leiter



Verallgemeinerungen

1. Die Gruppe $(\mathbb{F}_2^s, \oplus)$ wird durch eine beliebige Gruppe $(G, *)$ ersetzt. Die Formeln für Verschlüsselung und Entschlüsselung werden dann zu:

$$\begin{aligned} a_{i+1} &= a_{i-1} * f(a_i, \alpha_i(k))), \\ a_{i-1} &= a_{i+1} * f(a_i, \alpha_i(k)))^{-1}. \end{aligned}$$

2. Unbalancierte Feistel-Chiffren (Schneier/Kelsey): Hier werden die Blöcke in zwei ungleich große Hälften zerteilt: $\mathbb{F}_2^n = \mathbb{F}_2^s \times \mathbb{F}_2^t$, $x = (\lambda(x), \rho(x))$. Die Formel für die Verschlüsselung wird dann zu:

$$\begin{aligned} L_i &= \rho(L_{i-1}, R_{i-1}) && \in \mathbb{F}_2^s, \\ R_i &= \lambda(L_{i-1}, R_{i-1}) \oplus f(L_i, \alpha_i(k))) && \in \mathbb{F}_2^t. \end{aligned}$$

Beispiele

1. Lucifer II (von Feistel 1971 entwickelt, 1975 veröffentlicht),
2. DES (von Coppersmith u. a. bei der IBM 1974 entwickelt, 1977 als US-Norm veröffentlicht),
3. u. v. a. neuere Bitblock-Chiffren.

Die **Bedeutung** der Feistel-Netze liegt in den empirischen Beobachtungen:

- Die „(s, q)-Bit-Sicherheit“ der Kernfunktion f wird durch die Mehrfach-Ausführung im Runden-Schema zu einer „(n, l)-Bit-Sicherheit“ der Feistel-Chiffre F erweitert.
- Die gesamte Chiffre ist lässt sich aus handhabbaren, auf Sicherheit optimierbaren Stücken zusammensetzen.

Die erste dieser Beobachtungen lässt sich auch theoretisch untermauern: Nach einem Ergebnis von Luby/Rackoff ist eine Feistel-Chiffre mit mindestens vier Runden nicht mehr effizient von einer zufälligen Permutation zu unterscheiden, wenn die Kernfunktion zufällig ist. D. h., eine an sich gute, aber zu kurze zufällige Funktion wird durch diese Konstruktion zu einer ausreichend langen erweitert.

2.6 Algebraische Angriffe bei kleiner Rundenzahl

Formeln für kleine Rundenzahlen

Die Rekursionsformel für eine FEISTEL-Chiffre kann man in der Form

$$(L_i, R_i) = (R_{i-1}, L_{i-1} + f(R_{i-1}, k_i))$$

schreiben mit dem Rundenschlüssel $k_i = \alpha_i(k)$.

Satz 4 *Die Ergebnisse einer* FEISTEL*-Chiffre nach* 2, 3 *und* 4 *Runden erfüllen die Gleichungen*

$$\begin{aligned} L_2 - L_0 &= f(R_0, k_1), \\ R_2 - R_0 &= f(L_2, k_2); \end{aligned}$$

$$\begin{aligned} L_3 - R_0 &= f(L_0 + f(R_0, k_1), k_2), \\ R_3 - L_0 &= f(L_3, k_3) + f(R_0, k_1); \end{aligned}$$

$$\begin{aligned} L_4 - L_0 &= f(R_0, k_1) + f(R_4 - f(L_4, k_4), k_3), \\ R_4 - R_0 &= f(L_4, k_4) + f(L_0 + f(R_0, k_1), k_2). \end{aligned}$$

Die Minuszeichen stehen hier, damit die Formeln auch noch für die Verallgemeinerung auf abelsche Gruppen gut sind, wo nicht, wie „im Binären“, Plus und Minus zusammenfallen. Der Sinn dieser Formeln ist, dass außer den Rundenschlüsseln k_i jeweils nur der Klartext (L_0, R_0) und der Geheimtext (L_r, R_r) vorkommen, also Größen, die bei der algebraischen Kryptoanalyse als bekannt angenommen werden.

Beweis. Im Fall von zwei Runden sind die Gleichungen

$$\begin{aligned} L_1 &= R_0, \\ R_1 &= L_0 + f(R_0, k_1), \\ L_2 &= R_1 = L_0 + f(R_0, k_1), \\ R_2 &= L_1 + f(R_1, k_2) = R_0 + f(L_2, k_2); \end{aligned}$$

und daraus folgt die Behauptung.

Im Fall von drei Runden ist analog

$$\begin{aligned} L_1 &= R_0, \\ R_1 &= L_0 + f(R_0, k_1), \\ L_2 &= R_1 = L_0 + f(R_0, k_1), \\ R_2 &= L_1 + f(R_1, k_2) = R_0 + f(L_2, k_2), \\ L_3 &= R_2 = R_0 + f(L_0 + f(R_0, k_1), k_2), \\ R_3 &= L_2 + f(R_2, k_3) = L_0 + f(R_0, k_1) + f(L_3, k_3). \end{aligned}$$

Die Berechnung für vier Runden bleibt dem Leser überlassen. ◇

Zweirunden-Chiffren

Bei einem Angriff mit einem bekannten Klartextblock sind L_0, R_0, L_2 und R_2 gegeben. Aufzulösen sind die Gleichungen

$$\begin{aligned} L_2 - L_0 &= f(R_0, k_1), \\ R_2 - R_0 &= f(L_2, k_2); \end{aligned}$$

nach k_1 und k_2. Die Sicherheit der Chiffre hängt also ganz von der Kernfunktion f ab. Da q, die Bitlänge der Teilschlüssel, allerdings in der Regel wesentlich kleiner als die Gesamtschlüssellänge l ist, sind die für einen Exhaustionsangriff nötigen 2^{q+1} Auswertungen von f eventuell im Bereich des Möglichen. Bemerkenswert ist, dass diese Überlegung unabhängig von der Schlüsselauswahl α ist – es werden einfach die tatsächlich verwendeten Schlüsselbits (k_1, k_2) bestimmt.

Beispiel: Auf $\mathbb{F}_2^s$ wird die Multiplikation „$\cdot$“ des Körpers $\mathbb{F}_t$ mit $t = 2^s$ verwendet [diese wird in einem späteren Abschnitt genauer erklärt] und

$$f(x, y) = x \cdot y$$

gesetzt. Die Schlüsselauswahl sei durch $l = 2q$ und $k_i =$ linke oder rechte Hälfte von k, je nachdem ob i ungerade oder gerade ist, gegeben. Dann werden die Gleichungen zu

$$\begin{aligned} L_2 - L_0 &= R_0 \cdot k_1, \\ R_2 - R_0 &= L_2 \cdot k_2, \end{aligned}$$

sind also leicht zu lösen. (Falls einer der Faktoren R_0 oder L_2 Null ist, braucht man natürlich einen anderen bekannten Klartextblock.)

Dreirunden-Chiffren

Hier sind die zu lösenden Gleichungen deutlich komplexer, da f bereits iteriert wird. Allerdings ist mit einem bekanntem Klartextblock ein Treffpunktangriff (Meet in the Middle) durchführbar, wenn die Bitlänge q der Teilschlüssel nicht zu groß ist: Man berechnet für alle möglichen Teilschlüssel k_1 das Zwischenergebnis (L_1, R_1) nach der ersten Runde und speichert es in einer Tabelle. Über die letzten beiden Runden macht man eine Exhaustion, wie für die Zweirunden-Chiffre beschrieben. Der Gesamtaufwand beträgt also $3 \cdot 2^q$ Auswertungen von f und 2^q Speicherplätze.

Daraus resultiert die Faustregel: Feistel-*Chiffren sollten stets mindestens vier Runden haben.* Das ist eine passende Ergänzung zu dem zitierten Ergebnis von Luby/Rackoff. Man sieht, wie mit wachsender Rundenzahl, wenn nur f genügend komplex ist, die Resistenz des gesamten Schemas gegen den algebraischen Angriff wächst.

In der **Beispiel**chiffre mit Kernfunktion = Multiplikation im Körper mit 2^s Elementen werden die Gleichungen zu:

$$\begin{aligned} L_3 - R_0 &= [L_0 + R_0 \cdot k_1)] \cdot k_2, \\ R_3 - L_0 &= [R_0 + R_3] \cdot k_1. \end{aligned}$$

Sie sind offensichtlich leicht zu lösen.

Vierrunden-Chiffren

Hier sind die Gleichungen noch komplexer, und selbst im **Beispiel**

$$\begin{aligned} L_4 - L_0 &= [R_0 + R_4 + L_4 \cdot k_2] \cdot k_1, \\ R_4 - R_0 &= [L_4 + L_0 + R_0 \cdot k_1] \cdot k_2, \end{aligned}$$

sind sie schon quadratisch in zwei Unbekannten (wenn auch in diesem Trivialbeispiel noch leicht zu lösen – die Elimination von k_1 führt auf eine quadratische Gleichung für k_2 – **Übungsaufgabe**).

2.7 Lucifer

Geschichte und Bedeutung

Lucifer war die erste öffentlich bekannte und weit diskutierte Bitblock-Chiffre. Sie wurde um 1970 von Horst Feistel bei der IBM entwickelt und ist auch wirklich eine Feistel-Chiffre. Sie gilt als Vorläufer der kurz darauf entwickelten Standard-Chiffre DES, der gegenüber sie einige ins Auge fallende Stärken, aber, wie sich inzwischen herausgestellt hat, auch entscheidende Schwächen hat.

> Es handelt sich bei der Beschreibung hier um die Variante, die wegen der etwas späteren Veröffentlichung als „Lucifer II“ bezeichnet wird; die zuvor veröffentlichte Variante (1973 im *Scientific American*) weicht davon etwas ab.

Folgendes sind die Charakteristika von Lucifer:

- 128-Bit-Schlüssel, d. h., ein auch für heutige Begriffe bei weitem ausreichender Schlüsselraum.
- 128-Bit-Blöcke, auch dies – wie gesehen – heutigen Ansprüchen genügend.
- Für die Verarbeitung werden die 128 Bit von Schlüsseln und Blöcken in 16 Oktette (fälschlicherweise, aber inzwischen eingebürgert, auch als Bytes bezeichnet) unterteilt.
- Es werden 16 Runden durchgeführt.
- In jeder Runde werden die 8 Oktette der rechten Blockhälfte (= 64 Bits) quasi parallel bearbeitet – anders ausgedrückt, besteht jede Runde aus 8 gleichartigen Verarbeitungsblöcken, in denen jeweils 1 Oktett abgearbeitet wird.
- Jeder dieser Verarbeitungsblöcke besteht aus einer Substitution und einer Permutation, dazwischen werden Schlüsselbits binär addiert.
- Nichtlinearität wird also einzig und allein durch die Substitution in den Algorithmus eingeführt.
- Für die Substitution eines Bytes wird dieses in zwei 4-Bit-Hälften zerlegt und jede davon getrennt mit einer Substitution

  $$S_0, S_1 \colon \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4$$

  behandelt. Das sind immer die gleichen beiden Substitutionen; einzig die Zuordnung – welche 4-Bit-Hälfte mit S_0 und welche mit S_1

behandelt wird – variiert; sie wird aufgrund eines Bits des Schlüssels entschieden. Es hat sich eingebürgert, solche elementaren nichtlinearen BOOLEschen Abbildungen wie diese S_0 und S_1 als **„S-Boxen“** zu bezeichnen.

- Das Verfahren ist sehr gut für eine Hardware-Implementierung geeignet – auch auf 8-Bit-Architekturen. In Software ist es wegen vieler Bit-Permutationen weniger effizient.
- Das Prinzip, die Kernfunktion aus vielen kleinen, evtl. gleichartigen, S-Boxen zusammenzusetzen, wird auch heute noch gelegentlich verwendet. *Faustregel*: Je kleiner die S-Boxen, desto mehr Runden sind nötig, um Sicherheit zu erreichen.

Die Darstellung des Verfahren folgt dem Artikel

- Arthur Sorkin: Lucifer, a cryptographic algorithm. Cryptologia 8 (1984), 22–41.

Die Schlüsselauswahl

Die 16 Oktette des Schlüssels $k \in \mathbb{F}_2^{128}$ werden mit

$$k = (k_0, \ldots, k_{15}) \in (\mathbb{F}_2^8)^{16}$$

bezeichnet – IBM-Nummerierung, aber mit 0 beginnend. In der i-ten Runde werden davon die Oktette

$$\alpha_i(k) = (\alpha_{ij}(k))_{0 \le j \le 7} \quad \text{mit } \alpha_{ij}(k) = k_{7i+j-8 \bmod 16}$$

benützt. Diese kompliziert aussehende Formel beschreibt in Wirklichkeit etwas ganz einfaches, nämlich folgendes Auswahl-Schema:

Runde	Position							
	0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6	7
2	7	8	9	10	11	12	13	14
3	14	15	0	1	2	3	4	5
4	5	6	7	8	9	10	11	12
5	12	13	14	15	0	1	2	3
6	3	4	5	6	7	8	9	10
7	10	11	12	13	14	15	0	1
8	1	2	3	4	5	6	7	8
9	8	9	10	11	12	13	14	15
10	15	0	1	2	3	4	5	6
11	6	7	8	9	10	11	12	13
12	13	14	15	0	1	2	3	4
13	4	5	6	7	8	9	10	11
14	11	12	13	14	15	0	1	2
15	2	3	4	5	6	7	8	9
16	9	10	11	12	13	14	15	0

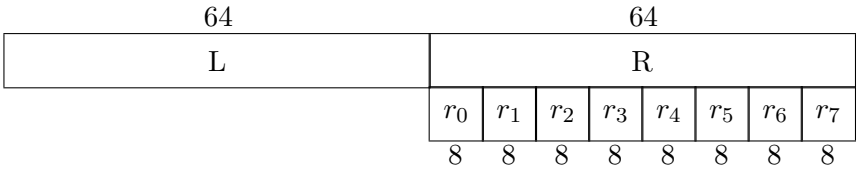
Von Runde zu Runde wird die Auswahl also zyklisch um 7 Plätze weitergeschoben. Bemerkenswert ist, dass jedes Oktett genau einmal in jeder Position vorkommt. Die Position gibt an, für welches Oktett des aktuellen 64-Bitblocks das jeweilige Schlüsseloktett verwendet wird. Das jeweils in Position 0 stehende Oktett $\alpha_{i0}(k)$ wird außerdem in der zugehörigen Runde als „Transformations-Steuerbyte“ eingesetzt.

Die Runden-Abbildung

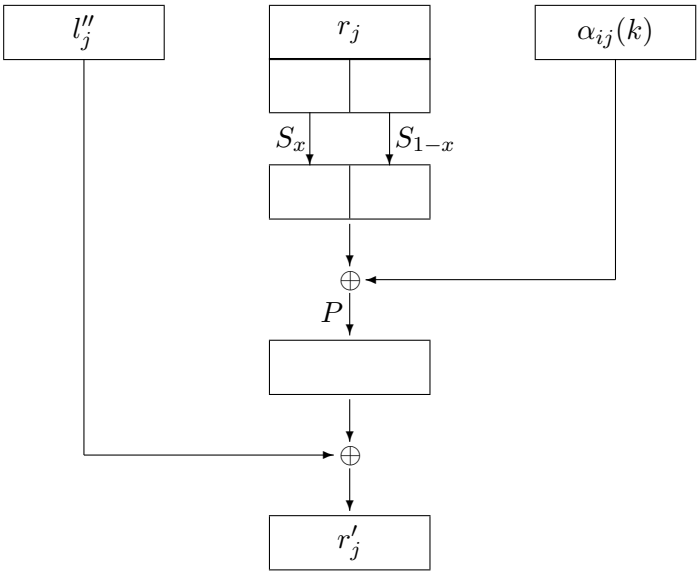
In der i-ten Runde wird der Input, die rechte 64-Bit-Hälfte des aktuellen Blocks, in die acht Oktette

$$r = (r_0, \ldots, r_7)$$

eingeteilt:



Für das j-te davon sieht die Transformation in dieser Runde so aus:



Dabei ist l''_j eine feststehende Auswahl von acht Bits aus der linken Hälfte des aktuellen Blocks. Das Transformations-Steuerbyte

$$\alpha_{i1}(k) = (b_0, \ldots, b_7)$$

wird von rechts nach links abgearbeitet; es ist $x = b_{7-j}$.

Diese grafische Beschreibung der eigentlichen Kern-Abbildung f ist nicht durch eine kompakte explizite Formel wiederzugeben.

Die S-Boxen

Die S-Boxen

$$S_0, S_1 \colon \mathbb{F}_2^4 \longrightarrow \mathbb{F}_2^4$$

werden durch ihre Wertetabellen beschrieben. Dabei werden die 4-Bitblöcke hexadezimal notiert, also z. b. $1011 = B$.

$x =$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_0(x) =$	C	F	7	A	E	D	B	0	2	6	3	1	9	4	5	8

$x =$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_1(x) =$	7	2	E	9	3	B	0	4	C	D	1	A	6	F	8	5

Die Permutationen

Die Permutation P vertauscht die Bits eines Oktetts wie folgt:

$$\begin{aligned} P \colon \mathbb{F}_2^8 &\longrightarrow \mathbb{F}_2^8, \\ (z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7) &\mapsto (z_2, z_5, z_4, z_0, z_3, z_1, z_7, z_6). \end{aligned}$$

Die bitweise Addition der linken auf die transformierte rechte Hälfte geschieht nach einer Permutation, die so beschrieben wird: Die linke Hälfte des aktuellen Blocks besteht aus acht Oktetten:

$$L = (l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7);$$

diese werden nach jedem Schritt zyklisch rotiert, d. h., für r_j sind die in der Lage

$$(l'_0, \ldots, l'_7) = (l_j, \ldots, l_{7+j \bmod 8}).$$

Dann wird das aufzuaddierende Oktett l''_j so gebildet:

$$l''_j = (\text{Bit 0 von } l'_7, \text{Bit 1 von } l'_6, \text{Bit 2 von } l'_2, \ldots)$$

usw. in der Reihenfolge $(7, 6, 2, 1, 5, 0, 3, 4)$.

3 Betriebsarten bei Blockverschlüsselung

Die Anwendung einer Blockverschlüsselungsfunktion $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ auf längere (oder kürzere) Bitfolgen erfordert zwei Maßnahmen:

1. die Folge in n-Bit-Blöcke aufspalten,
2. den letzten Block auffüllen („padding“) mit
   - Nullen oder
   - Zufallswerten oder
   - Strukturinformationen.

Jeder Block wird dann im Prinzip einzeln verschlüsselt, wobei aber gewisse „Verkettungen“ üblich sind. Hierfür gibt es vier Standardverfahren, die „Betriebsarten“ oder „Modi“ genannt werden:

- ECB,
- CBC,
- CFB,
- OFB,

die zusammen mit dem DES-Verfahren normiert wurden, aber für beliebige Blockchiffren anwendbar sind. Fundstelle dieser Normen im Internet ist

`http://csrc.nist.gov/cryptval/`

Darüber hinaus gibt es natürlich auch Varianten und nicht standardisierte Verfahren; den aktuellen Diskussionsstand findet man in

`http://www.nist.gov/modes/`

Zur Beschreibung der Betriebsarten ist es meist sinnvoll, folgende allgemeine Situation zu betrachten: Σ sei ein „Blockalphabet“, etwa $\mathbb{F}_2^n$, versehen mit einer Gruppenoperation $*$. Ferner sei eine Verschlüsselungsfunktion

$$f : \Sigma \longrightarrow \Sigma$$

gegeben.

Die Abhängigkeit vom Schlüssel spielt hierbei keine Rolle und wird daher in der Notation weggelassen.

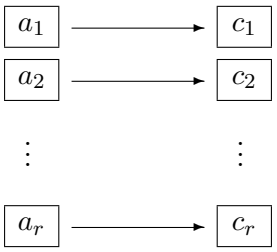
3.1 ECB = Electronic Code Book

Beschreibung

Sei r die Zahl der Blöcke, der Klartext also die Folge $(a_1, \ldots, a_r)$ von Blöcken.

Verschlüsselung: Beim ECB-Modus wird der Reihe nach einfach jeder Block für sich verschlüsselt:

$$a = (a_1, \ldots, a_r) \mapsto c = (c_1, \ldots, c_r) \in \Sigma^r \quad \text{mit } c_i = f(a_i).$$



Entschlüsselung: $a_i = f^{-1}(c_i)$.

Eigenschaften

Es handelt sich um eine monoalphabetische Substitution auf Σ. Falls $\#\Sigma$ sehr groß ist, ist das hinreichend sicher vor einem Geheimtextangriff. Nachteilig ist aber in jedem Fall:

- Information über identische Blöcke wird preisgegeben. Der Klartext ist zwar nicht zufällig, dennoch wird die Faustregel für das Geburtstagsphänomen hier oft interpretiert als: „Nach $2^{n/2}$ Bits beginnt beim ECB Information auszutreten.“ Durch die im folgenden behandelten Betriebsarten wird diese Grenze nach oben verschoben.

- Das Anlegen eines „Codebuchs“ aus bekannten Klartexten ist möglich. Bei strukturierten Nachrichten, z. B. Banktransaktionen, ist es ziemlich leicht, bekannte Klartextblöcke zu gewinnen.

- Ein aktiver Angriff durch Austausch oder Einschub einzelner Geheimtextblöcke (z. B. mit bekanntem, „sympathischen“ Klartext) ist möglich. Beispielsweise könnte ein Angreifer bei einer Banktransaktion, für die er weiß, in welchem Block der Empfänger definiert ist, diesen austauschen, um den Geldfluss umzuleiten. Was er dort hinschreiben muss, hat er aus einer früheren Transaktion als bekannten Klartextblock abgegriffen. Für diesen Angriff muss er den Schlüssel nicht kennen.

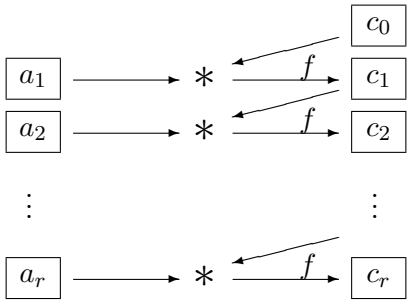
- Erlaubt die Situation einen Angriff mit gewähltem Klartext (Black-Box-Analyse), so ist Probeverschlüsselung bis hin zur Wörterbuch-Attacke möglich.

Besser ist es, eine Diffusion über die Klartextblöcke hinweg zu erzeugen. Das wird durch die im folgenden beschriebenen Betriebsarten erreicht.

3.2 CBC = Cipher Block Chaining

Beschreibung

Mit einem zufällig gewähltem Startwert c_0 (auch IV = „Initialisierungsvektor“ genannt) sieht das Verfahren so aus:



Verschlüsselung: Beim CBC-Modus wird nach folgender Formel verschlüsselt:

$$\begin{aligned} c_i &:= f(a_i * c_{i-1}) \quad \text{für } i = 1, \ldots, r \\ &= f(a_i * f(a_{i-1} * \cdots f(a_1 * c_0) \ldots)). \end{aligned}$$

Entschlüsselung: $a_i = f^{-1}(c_i) * c_{i-1}^{-1}$ für $i = 1, \ldots, r$.

Eigenschaften

- Jeder Geheimtextblock hängt von *allen vorhergehenden* Klartextblöcken ab (Diffusion).
- Ein Angreifer kann Textblöcke nicht unbemerkt ersetzen oder einfügen.
- Gleiche Klartextblöcke werden im allgemeinen verschieden chiffriert.
- Ein Angriff mit bekanntem Klartext ist hingegen, im Vergleich zum ECB, nicht erschwert.
- Jeder Klartextblock hängt von zwei Geheimtextblöcken ab.
- D. h., bei fehlerhafter Übermittlung eines Geheimtextblocks werden (nur) zwei Klartextblöcke unleserlich (≫Selbstsynchronisation≪ des Verfahrens).

Frage: *Kann der Startwert c_0 bei Geheimhaltung als zusätzlicher Schlüssel dienen?* (Das wären im Beispiel DES aus 56 Bits des Schlüssels und 64 Bits des Startwerts insgesamt 120 Bits.)

Antwort: Nein!

Begründung: Nur a_1 hängt beim Entschlüsseln von c_0 ab, d. h., es wird lediglich bekannter Klartext am Anfang etwas besser verschleiert, wenn c_0 geheim bleibt. Ist der zweite oder ein späterer Klartextblock bekannt, kann der Angreifer wie bei EBC den Schlüssel bestimmen (durch vollständige Suche oder einen anderen Angriff mit bekanntem Klartext.

Bemerkungen

1. CBC ist die Komposition $f\circ$ (Geheimtext-Autokey). Ist also $f = \mathbf{1}_\Sigma$, so bleibt das (völlig untaugliche) Geheimtext-Autokey-Verfahren mit Schlüssellänge 1 übrig.

2. (John Kelsey in der Mail-Liste `cryptography@c2.net`, 24 Nov 1999) Falls eine „Kollision“ $c_i = c_j$ für $i \neq j$ auftritt, folgt $f(a_i * c_{i-1}) = f(a_j * c_{j-1})$, also $a_i * c_{i-1} = a_j * c_{j-1}$ und daraus $a_j^{-1} * a_i = c_{j-1} * c_{i-1}^{-1}$. Der Gegner gewinnt also etwas Information über den Klartext.

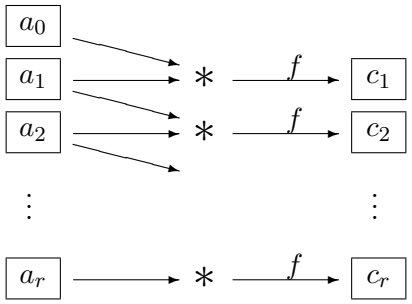
   Erwarten kann man diese Situation wegen des Geburtstagsphänomens nach ca. $\sqrt{\#\Sigma}$ Blöcken.

   Je länger der Text, desto mehr solcher Kollisionen sind zu erwarten. Auch dies bestätigt wieder die Faustregel über die Frequenz nötiger Schlüsselwechsel: rechtzeitig bevor $\sqrt{\#\Sigma}$ Blöcke erreicht sind.

3.3 Varianten des CBC

Klartext-Autokey

Ersetzt man beim CBC das Geheimtext-Autokey-Verfahren durch Klartext-Autokey, so erhält man folgendes Schema:



welches man PBC = Plaintext Block Chaining nennen könnte.

Verschlüsselung: Die Verschlüsselung folgt nach Wahl eines Startwertes a_0 der Formel:

$$c_i := f(a_i * a_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

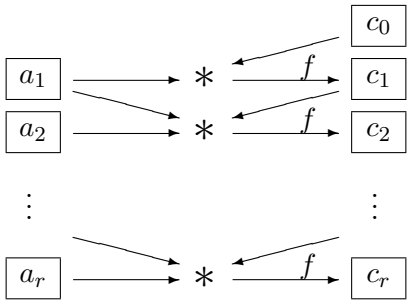
Entschlüsselung: Die Formel für die Entschlüsselung heißt:

$$a_i = f^{-1}(c_i) * a_{i-1}^{-1} \quad \text{für } i = 1, \ldots, r.$$

Dieses Verfahren ist allerdings völlig unüblich, und über seine Sicherheit ist anscheinend nichts bekannt.

PCBC = error-Propagating CBC

Dieses Verfahren ist ein Mix aus CBC und PBC und folgt dem Schema



Verschlüsselung: Die Verschlüsselung folgt (mit dem Startwert $a_0 =$ neutrales Element der Gruppe) der Formel:

$$c_i := f(a_i * a_{i-1} * c_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

Für die Bitblock-Chiffrierung wird a_0 also als Nullblock gewählt.

Entschlüsselung: Die Formel für die Entschlüsselung heißt:

$$a_i = f^{-1}(c_i) * c_{i-1}^{-1} * a_{i-1}^{-1} \quad \text{für } i = 1, \ldots, r.$$

Dieses Verfahren wurde bei älteren Versionen von Kerberos verwendet, wegen gewisser Schwächen inzwischen jedoch aufgegeben.

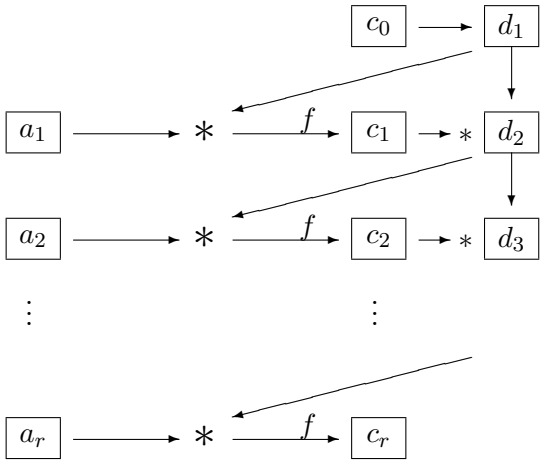
Verallgemeinerung nach MEYER/MATYAS

$$c_i := f(a_i * h(a_{i-1}, c_{i-1})) \quad \text{für } i = 1, \ldots, r,$$

wobei im Falle $\Sigma = \mathbb{F}_2^n$ z. B. für h die Addition modulo 2^n vorgeschlagen wird.

BCM = Block Chaining Mode

Diese Betriebsart folgt dem Schema:



Formel für die Verschlüsselung:

$$\begin{aligned} d_i &:= c_0 * \ldots * c_{i-1}, \\ c_i &:= f(a_i * d_i) \quad \text{für } i = 1, \ldots, r. \end{aligned}$$

Eine Anwendung des CBC

Der CBC-MAC (= „Message Authentication Code“) ist eine schlüsselabhängige „Hash-Funktion“, die zur Integritätsprüfung von Nachrichten verwendet wird. Sie ist in ISO/IEC 9797 normiert und mit dem DES-Verfahren im Bankenbereich verbreitet.

Sender und Empfänger einer Nachricht – die auch identisch sein können, wenn es sich um Nachrichtenspeicherung handelt – haben den Schlüssel k gemeinsam und verwenden die Verschlüsselungsfunktion $f = f_k$.

Der MAC eines Textes $a = (a_1, \ldots, a_r)$ ist der letzte Geheimtextblock, wenn a nach dem CBC verschlüsselt wird, also

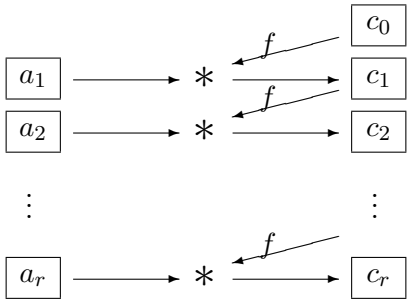
$$\mathrm{MAC}(a) = c_r = f(a_r * f(a_{r-1} * \cdots f(a_1 * c_0) \ldots)).$$

Wird $\mathrm{MAC}(a)$ zusammen mit a verschickt, kann der Empfänger die Echtheit von Absender und Inhalt prüfen, denn nur wer den Schlüssel hat, kann diesen Wert richtig berechnen.

Der Nachteil des geteilten Geheimnisses k ist allerdings, dass in einem Rechtsstreit zwischen den beiden Parteien jeder dem anderen eine Fälschung unterstellen kann.

3.4 CFB = Cipher Feedback

Beschreibung (in der einfachsten Version)



Verschlüsselung: Beim CFB-Modus wird nach folgender Formel verschlüsselt:

$$\begin{aligned} c_i &:= a_i * f(c_{i-1}) \quad \text{für } i = 1, \ldots, r \\ &= a_i * f(a_{i-1} * f(\cdots a_1 * f(c_0) \ldots)). \end{aligned}$$

Entschlüsselung: $a_i = c_i * f(c_{i-1})^{-1}$ für $i = 1, \ldots, r$.

Eigenschaften

- Der Startwert taugt auch hier nicht als zusätzlicher Schlüssel.
- Ein Angriff mit bekanntem Klartext wird auch durch diese Betriebsart nicht erschwert.
- Bemerkenswert ist, dass man auch zum Entschlüsseln nur f braucht, nicht etwa f^{-1}. Im Vorgriff sei hier schon vermerkt:
  - Der CFB-Modus ist für asymmetrische Chiffren ungeeignet.
  - Er kann aber mit einer echten (natürlich schlüsselabhängigen) Einweg- oder Hash-Funktion verwendet werden.
- Der CFB reduziert sich im Falle der identischen Abbildung $f = \mathbf{1}_\Sigma$ ebenfalls auf das Geheimtext-Autokey-Verfahren.
- (David Wagner) ECB $\circ$ CFB = CBC:

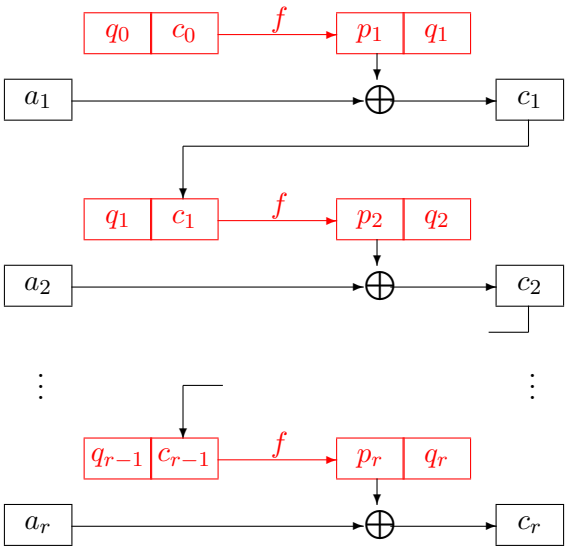
Dazu setzt man c_0 als Startwert für CFB und $c_0' := f(c_0)$ als Startwert für

CBC. Dann ist

$$\begin{aligned} c_1 &= \mathrm{CFB}(a_1) = a_1 * f(c_0), \\ c_1' &= \mathrm{ECB}(c_1) = f(a_1 * f(c_0)) = f(a_1 * c_0') = \mathrm{CBC}(a_1), \\ c_2 &= \mathrm{CFB}(a_2) = a_2 * f(c_1), \\ c_2' &= \mathrm{ECB}(c_2) = f(a_2 * f(c_1)) = f(a_2 * c_1') = \mathrm{CBC}(a_2), \\ &\text{usw.} \end{aligned}$$

Die genormte Version

... verwendet ein Schieberegister und ist daher nur im Falle $\Sigma = \mathbb{F}_2^n$ definiert. Hier ist $1 \leq t \leq n$, und verschlüsselt werden Blöcke $a_i \in \mathbb{F}_2^t$ der Länge t; der aktuelle Geheimtextblock c_i der Länge t wird von rechts in das (hier rot dargestellte) Schieberegister nachgeschoben:

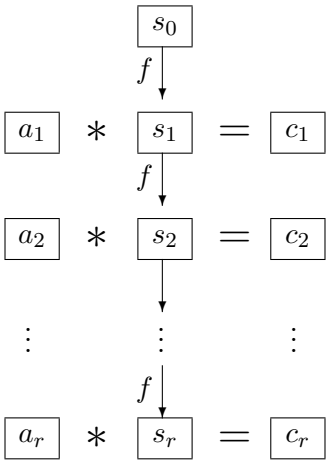


Die q_i sind dabei stets Bitblöcke der Länge $n - t$.

Diese allgemeinere Version ist weniger sicher als die mit $t = n$ und wird daher kaum verwendet.

3.5 OFB = Output Feedback

Beschreibung (in der einfachsten Version)



Auch diese Betriebsart war ursprünglich als Schieberegister-Version definiert; auch hier bedeutet die Verwendung eines $t <$ Blocklänge n eine Schwächung des Verfahrens [Jueneman, Crypto 82].

Verschlüsselung: Beim OFB-Modus wird nach folgender Formel verschlüsselt:

$$c_i := a_i * s_i, \quad s_i := f(s_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

Entschlüsselung: nach der Formel

$$a_i = c_i * s_i^{-1}, \quad s_i := f(s_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

Eigenschaften

- Es gibt keine Diffusion, aber gleiche Klartextblöcke werden im allgemeinen verschieden verschlüsselt.
- Im Falle $\Sigma = \mathbb{F}_2^s$ handelt es sich um eine Bitstrom-Chiffre mit f als „Zufallsgenerator“.
- Wird eine besonders schnelle Ver- und Entschlüsselung benötigt, so kann man den „Schlüsselstrom“ s_i auf beiden Seiten (beim Sender und beim Empfänger) vorausberechnen.
- Auch hier wird zum Entschlüsseln nur f benötigt, nicht f^{-1}.
- Falls $\Sigma = \mathbb{F}_2^s$, ist die Chiffre sogar involutorisch, d. h., Verschlüsselung = Entschlüsselung (als Funktion). Allgemeiner gilt das, wenn Σ eine Gruppe vom Exponenten 2 ist.

- Bei einem Angriff mit bekanntem Klartext liefert ein Paar (a_1, c_1) den Wert s_1, ein weiteres Paar (a_2, c_2) den Wert $f(s_1)$. Damit ist also ein Angriff mit bekanntem Klartext auf f selbst möglich.
- Das Geheimhalten des Startwerts s_0 bringt also auch hier praktisch keine zusätzliche Sicherheit.

Variante: Der Counter-Mode CTR

Hier ist im einfachsten Fall

$$c_i := a_i * f(i) \quad \text{für } i = 1, \ldots, r.$$

4 DES

Der 'Data Encryption Standard' (DES) wurde im wesentlichen bei der IBM von einer Forschungsgruppe um Feistel und Coppersmith entwickelt; die NSA wirkte mit: Sie sorgte für eine Modifikation der S-Boxen und die Reduzierung der Schlüssellänge auf 56 Bit – entgegen allen ursprünglichen Vermutungen ist beides nach heutigen Erkenntnissen keine Schwächung. Der DES wurde 1977 vom NBS ('National Bureau of Standards' – heute NIST) in den USA genormt. Das Ziel der Entwicklung war, für 10 bis 15 Jahre ein zuverlässiges Verschlüsselungssystem für sensible (aber nicht hochgeheime) Daten der Regierung zur Verfügung zu haben. Die Norm verlangt eine Hardware-Implementation des Algorithmus; von 1989 bis 1998 unterlagen DES-Chips der US-Ausfuhrbeschränkung. Der Algorithmus selbst heißt eigentlich DEA, schon lange werden aber auch Software-Implementationen als DES bezeichnet.

Verschlüsselt werden 64-Bit-Blöcke, wobei ein 56-Bit-Schlüssel verwendet wird. Die Verschlüsselung eines Blocks beginnt mit einer festen (bekannten) Permutation und endet mit der Umkehrpermutation. Obwohl diese Permutationen bekannt sind, wird dadurch schon eine gewisse Diffusion erreicht. Dazwischen werden 16 Runden durchgeführt, in denen sowohl Diffusion als auch Konfusion erhöht werden. Die einzelnen Runden unterscheiden sich nur dadurch, dass jeweils eine andere 48-Bit-Gruppe aus dem Schlüssel gewählt wird. Die Entschlüsselung unterscheidet sich von der Verschlüsselung nur dadurch, dass die Runden in umgekehrter Reihenfolge durchlaufen werden.

Im folgenden wird der Algorithmus stufenweise „von innen nach außen“ beschrieben. $\oplus$ ist immer die bitweise Addition modulo 2 (XOR).

4.1 Die Kern-Abbildung

Im Innern des DES steckt die „Kern-Abbildung“

$$f\colon \mathbb{F}_2^{32} \times \mathbb{F}_2^{48} \longrightarrow \mathbb{F}_2^{32},$$

die als Input 32 Textbits und einen 48-Bit-Teilschlüssel hat. Zuerst werden die 32 Textbits durch teilweise Wiederholung zu 48 Bits aufgebläht; die „Expansionsabbildung“

$$E\colon \mathbb{F}_2^{32} \longrightarrow \mathbb{F}_2^{48}$$

wird durch die folgende Tabelle beschrieben:

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

Die Tabelle ist so zu interpretieren dass

$$E(b_1 b_2 \dots b_{32}) = b_{32} b_1 b_2 b_3 \dots b_{31} b_{32} b_1.$$

Die expandierten 48 Bits werden mit dem 48-Bit-Teilschlüssel per $\oplus$ überlagert. Die resultierenden 48 Bits werden in 8 Gruppen zu je 6 Bits zerteilt und auf diese die 1. bis 8. S(ubstitutions)-Box

$$S_j\colon \mathbb{F}_2^6 \longrightarrow \mathbb{F}_2^4 \quad (j = 1, \dots, 8)$$

angewendet. Die S-Boxen werden im nächsten Abschnitt beschrieben.

Insgesamt erhält man die (polyalphabetisch zusammengesetzte) Substitution

$$S\colon \mathbb{F}_2^{48} \longrightarrow \mathbb{F}_2^{32}.$$

Schließlich wird noch die P(ermutations)-Box

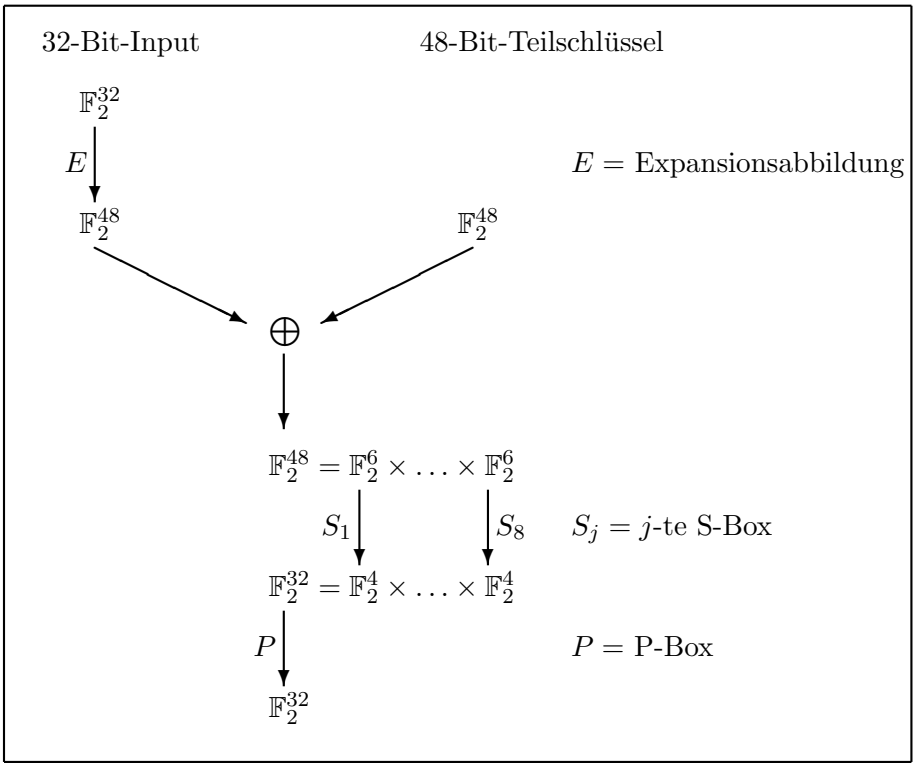
$$P\colon \mathbb{F}_2^{32} \longrightarrow \mathbb{F}_2^{32}$$

ausgeführt, die durch die folgende Tabelle beschrieben wird; das heißt,

$$P(b_1 b_2 \dots b_{32}) = b_{16} b_7 \dots b_4 b_{25}.$$

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

Zusammengefasst wird die Kernabbildung in der folgenden Abbildung:



4.2 Die S-Boxen

Jede der acht S-Boxen S_j wird durch eine 4×16-Matrix beschrieben, siehe die Tabelle:

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

Jede Zeile ist eine Permutation der Zahlen $0, \ldots, 15$. Um $S_j(b_1 \ldots b_6)$ zu bestimmen, deutet man $b_1 b_6$ als Binärdarstellung einer Zahl in $\{0, 3\}$ und $b_2 b_3 b_4 b_5$ als Binärdarstellung einer Zahl in $\{0, 15\}$, liest in der Matrix zu S_j die Zahl in Zeile $b_1 b_6$ und Spalte $b_2 b_3 b_4 b_5$ ab und stellt sie binär dar. Beispiel:

$$S_3(101100) = 0011 \quad \rightarrow \quad \text{Zeile 2, Spalte 6.}$$

4.3 Die Runden

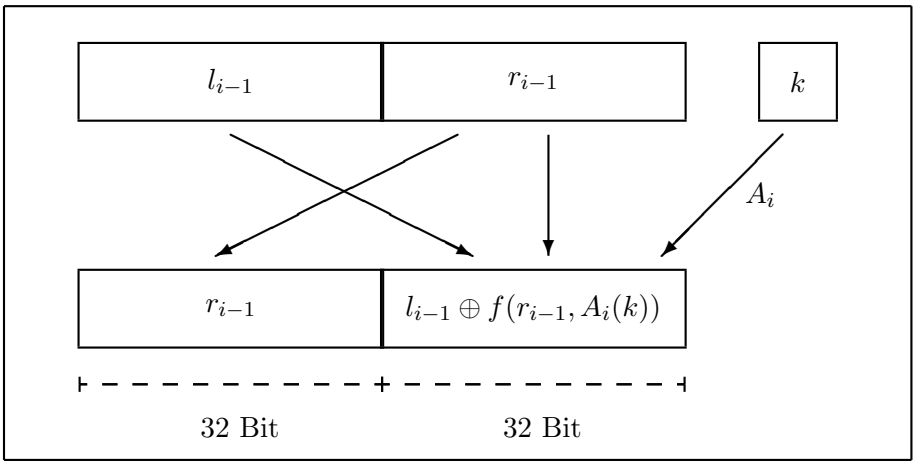
Die 16 Runden im DES bestehen aus je einer Abbildung

$$R_i \colon \mathbb{F}_2^{64} \times \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{64} \quad (i = 1, \ldots, 16),$$

die mit Hilfe der i-ten Schlüsselauswahl

$$A_i \colon \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{48} \quad (i = 1, \ldots, 16),$$

wie in der folgenden Abbildung beschrieben wird.



Die Runden unterscheiden sich also nur durch den verwendeten Teilschlüssel $A_i(k)$. Man erkennt hier das FEISTEL-Schema.

4.4 Die Schlüsselauswahl

Zur Beschreibung der Runden gehört noch die Beschreibung der Schlüsselauswahl. Zunächst wird der 56-Bit-Schlüssel auf 64 Bit aufgebläht, indem nach je 7 Bits ein Paritätsbit eingefügt wird; welches, ist egal, man kann sogar beliebige Bits einfügen, da die zusätzlichen Bits nicht weiter verwendet werden. Jedenfalls ist der erste Schritt eine Abbildung

$$\mathrm{Par}\colon \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{64}.$$

Im zweiten Schritt werden die ursprünglichen 56 Bits wieder extrahiert, allerdings in der Reihenfolge der folgenden Tabelle.

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

Das ist eine Abbildung

$$\mathrm{PC}_1\colon \mathbb{F}_2^{64} \longrightarrow \mathbb{F}_2^{56}$$

('Permuted Choice 1'). Nun werden die 56 Bits in zwei 28-Bit-Hälften geteilt und diese jeweils zyklisch nach links geschoben, insgesamt 16 mal. Das sind also 16 Abbildungen

$$\mathrm{LS}_i\colon \mathbb{F}_2^{28} \longrightarrow \mathbb{F}_2^{28} \quad (i = 1, \ldots 16);$$

wie weit geschoben wird, zeigt die Tabelle:

1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Die ersten beiden Male wird also um ein Bit geschoben, dann 6 mal um zwei Bits usw. Für die i-te Schlüsselauswahl A_i wird nach der i-ten Verschiebung noch die 'Permuted Choice 2',

$$\mathrm{PC}_2\colon \mathbb{F}_2^{56} \longrightarrow \mathbb{F}_2^{48}$$

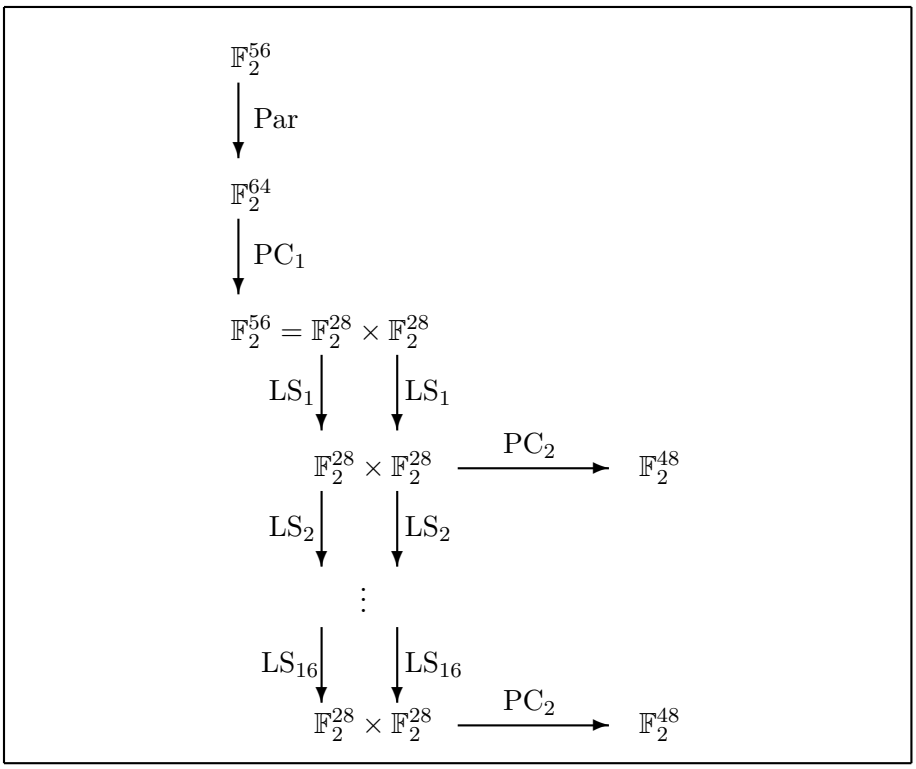
ausgeführt, wobei die Auswahl in der Reihenfolge der folgenden Tabelle geschieht (die Bits 9, 18, 22, 25, 35, 38, 43, 54 entfallen dabei).

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

Insgesamt ist

$$A_i = \mathrm{PC}_2 \circ \mathrm{LS}_i \circ \cdots \circ \mathrm{LS}_1 \circ \mathrm{PC}_1 \circ \mathrm{Par}\,.$$

Diese Konstruktion wird noch einmal in dieser Abbildung zusammengefasst:



4.5 Der gesamte Algorithmus

Nun ist noch die Initial-Permutation

$$\mathrm{IP}\colon \mathbb{F}_2^{64} \longrightarrow \mathbb{F}_2^{64}$$

zu beschreiben; das geschieht durch diese Tabelle:

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

Invers zu IP ist die Ausgabe-Permutation IP^{-1}; der Bequemlichkeit halber wird die zugehörige Tabelle ebenfalls angegeben:

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

Der gesamte DES-Algorithmus DES_k zum Schlüssel $k \in \mathbb{F}_2^{56}$ ist nun die Zusammensetzung

$$\mathbb{F}_2^{64} \xrightarrow{\mathrm{IP}} \mathbb{F}_2^{64} \xrightarrow{R_1(\bullet,k)} \dots \xrightarrow{R_{16}(\bullet,k)} \mathbb{F}_2^{64} \xrightarrow{T} \mathbb{F}_2^{64} \xrightarrow{\mathrm{IP}^{-1}} \mathbb{F}_2^{64}.$$

Dabei ist T die Vertauschung der linken und der rechten 32-Bit-Hälften, die man einschiebt, damit DES_k^{-1} bis auf die umgekehrte Reihenfolge der Runden wie DES_k aussieht.

Anmerkung. Der Sinn von Initial- und Ausgabe-Permutation liegt wohl in einer bequemen Verdrahtung von Input und Output auf kleinen Prozessoren. Kryptologisch ergeben die Permutationen keinen Effekt, da beide ja vom Kryptoanalytiker ohne weiteres abgestreift werden können. Für eine Software-Implementierung wirken sie nur als Bremsen; dennoch dürfen sie nicht weggelassen werden, wenn man eine zum Standard kompatible Implementation will.

5 Kryptoanalyse von Bitblock-Chiffren

Für die Kryptoanalyse von Bitblock-Chiffren sind folgende allgemeine Ansätze bekannt:

1. Exhaustion = vollständige Schlüsselsuche. Sie wird durch Wahl einer genügend großen Schlüssellänge unmöglich gemacht. – Viele der weiteren bekannten Angriffe zielen darauf ab, den Suchraum für die Exhaustion signifikant zu verkleinern.

2. Algebraischer Angriff. Er wird durch die Nichtlinearität der Chiffre erschwert oder verhindert.

3. Statistische Angriffe auf versteckte Linearität:

   (a) Lineare Kryptoanalyse (Matsui/Yamagishi, Eurocrypt 92). Sie ist das Hauptthema dieses Abschnitts.

   (b) Differenzielle Kryptoanalyse (Murphy, Shamir, Biham 1990 – bei IBM und NSA schon 1974 bekannt). Sie wird im Anschluss daran kurz erläutert.

   (c) Verallgemeinerungen und Mischformen, s. u.

Im Gegensatz zur differenziellen Kryptoanalyse war die lineare Kryptoanalyse den DES-Entwicklern wahrscheinlich nicht bekannt; demgemäß ist DES gegen sie nicht optimal resistent. Es gab nur das Design-Kriterium

- Die S-Boxen sollen so nichtlinear wie möglich sein.

Aber Shamir bemerkte schon früh (Crypto 85), dass es „lineare Approximationen“ für die S-Boxen gibt, deren Übereinstimmung besser als zufällig ist. Es dauerte allerdings weitere 8 Jahre, bis es Matsui gelang, diese Beobachtung systematisch auszunutzen.

Darüber hinaus sind in den letzten Jahren verschiedene Verallgemeinerungen und Kombinationen von linearer und differenzieller Kryptoanalyse entwickelt worden:

- Angriff mit verwandten Schlüsseln – ‘related keys’ (Biham 1992, Schneier).

- Differenziale höherer Ordnung (Harpes 1993, Biham 1994, Lai 1994).

- Differenziell-lineare Kryptoanalyse (Langford/Hellman 1994).

- Partielle Differenziale (Knudsen 1995).

- I/O-Summen-Analyse (Harpes/Kramer/Massey 1995).

- S-Box-Paar-Analyse (Davies/Murphy 1995, Mirza 1996).
- Bumerang-Angriff (Wagner 1999).
- Slide-Attacken auf (evtl. versteckte) Periodizität in Chiffren oder Schlüsselauswahl-Schemata (Biryukov/Wagner 1999).
- Unmögliche Differenziale (Biham/Biryukov/Shamir 1999).

Alle diese Verfahren einschließlich der linearen und der differenziellen Kryptoanalyse sind allerdings kaum konkret zum Brechen einer Chiffre im Sinne der klassischen Kryptoanalyse anwendbar. Sie setzen so viele bekannte Klartexte voraus, wie man in realistischen Situationen kaum je erhalten kann. Ihr Sinn liegt aber vor allem darin, sinnvolle Maße für die Sicherheit von Bitblock-Chiffren zu gewinnen. Ein solches Sicherheitsmaß ist z. B. die Anzahl bekannter Klartextblöcke, die man für einen Angriff benötigt. Chiffren, die selbst unter unrealistischen Annahmen über die Kenntnisse des Angreifers sicher sind, können als in der Praxis besonders sicher gelten.

Hier zeigt sich die Kryptoanalyse in ihrem „legalen“ Aspekt – sie dient nicht zu kriminellen Ausspähaktionen, sondern ist ein wichtiges Werkzeug für die Konstruktion starker Chiffren.

Bei Feistel- oder ähnlichen iterierten Chiffren startet man die Angriffe bei den nichtlinearen Bestandteilen der einzelnen Runden – die wie bei LUCIFER und DES meist S-Boxen genannt werden – und versucht, den Angriff über mehrere Runden auszudehnen. Dabei sieht man oft, wie die Schwierigkeit des Angriffs mit der Rundenzahl zunimmt. So erhält man Kriterien, ab wievielen Runden eine Chiffre „sicher“ ist.

Man darf dabei nicht vergessen, dass sich der Angriff immer auf eine bestimmte algebraische Struktur bezieht; hier auf die $\mathbb{F}_2$-Vektorraumstruktur des Klartextblock-Raums. Selbstverständlich kann man ähnliche Angriffsversuche auch auf andere Strukturen ansetzen; eine Abbildung, die komplex aussieht, könnte z. B. plötzlich einfach aussehen, wenn man ihre Wirkung auf die Struktur als zyklische Gruppe $\mathbb{Z}/n\mathbb{Z}$ untersucht – oder gar auf „exotische“ Strukturen, die extra zur Untersuchung dieser einen Abbildung eingeführt werden. Wir beschränken uns hier exemplarisch auf die $\mathbb{F}_2$-Vektorraumstruktur, über die man am meisten weiß.

Kriterien für Bitblock-Chiffren

... bzw. deren Runden-Abbildungen oder deren nichtlineare Bausteine, die S-Boxen. [Sie werden zum großen Teil im mathematischen Einschub „Linearitätsmaße für BOOLEsche Abbildungen“ behandelt.]

- **Diffusion/Lawineneffekt:** Bei Änderung eines Klartextbits ändern sich ca. 50% der Geheimtextbits. Hierdurch sollen statistische Unregelmäßigkeiten vermieden werden.

- **Balanciertheit:** Alle Urbildmengen sind gleich groß, d. h., die Werte der Abbildung sind gleichmäßig verteilt. Unregelmäßigkeiten in der Verteilung würden einen Ansatz zur statistischen Kryptoanalyse bieten.

- **Algebraische Komplexität:** Die Bestimmung von Urbildern oder Teilen davon soll auf möglichst schwer lösbare Gleichungen führen. Diese Forderung hängt mit dem algebraischen Grad der Abbildung zusammen, aber nicht auf leicht zu beschreibende Weise.

- **Nichtlinearität:** Hier gibt es eine Reihe von Kriterien, die auch „versteckte“ Nichtlinearität messen und vergleichsweise leicht zu beschreiben und handzuhaben sind; sie zeigen u. a., wie anfällig die Abbildungen für lineare oder differenzielle Kryptoanalyse sind.

  - Das „lineare Potenzial“ soll möglichst gering, das „Linearitätsprofil“ möglichst ausgeglichen sein.
  - Das „differenzielle Potenzial“ soll möglichst gering, das „Differenzenprofil“ möglichst ausgeglichen sein.
  - Die „Nichtlinearität“, der Abstand (HAMMING-Distanz) zu affinen Abbildungen, soll möglichst groß sein.
  - Die „Linearitätsdistanz“, der Abstand zu Abbildungen mit „linearer Struktur“, soll möglichst groß sein.

Einige dieser Kriterien lassen sich gleichzeitig erfüllen, andere widersprechen sich teilweise, so dass das Design einer Bitblock-Chiffre insbesondere eine Abwägung der verschiedenen Kriterien erfordert; statt der Optimierung nach einem Kriterium ist ein möglichst gleichmäßig hohes Niveau bezüglich aller Kriterien anzustreben.

Zur Zeit wird grundsätzlich der Konflikt zwischen Balanciertheit und Nichtlinearität zu Gunsten der Balanciertheit entschieden. Dafür gibt es aber keinen wirklich stichhaltigen Grund; statistische Angriffe, die die Ungleichverteilung der Bilder bei nicht balancierten Abbildungen ausnutzen, sind einfach leichter zu begreifen und werden daher höher gewichtet. Die Abstriche bei der Nichtlinearität bekommt man durch Erhöhung der Rundenzahl in den Griff.

Die Bezeichnungen und Ergebnisse des Abschnitts „Linearitätsmaße für BOOLEsche Abbildungen“ werden im folgenden, oft ohne expliziten Hinweis, verwendet.

5.1 Die Idee der linearen Kryptoanalyse

Sei

$$F\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$$

eine Bitblock-Chiffre. Wir stellen uns die Argumente von F als Klartexte $a \in \mathbb{F}_2^n$ und Schlüssel $k \in \mathbb{F}_2^l$, die Werte von F als Geheimtexte $c \in \mathbb{F}_2^n$ vor. Dann kann man zu zwei Linearformen

$$\alpha\colon \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2, \quad \text{und} \quad \beta\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$$

die Wahrscheinlichkeit der linearen Relation (α, β) beziehungsweise ihr Potenzial betrachten:

$$\begin{aligned} p_F(\alpha,\beta) &= \frac{1}{2^{n+l}} \cdot \#\{(a,k,c) \in \mathbb{F}_2^n \times \mathbb{F}_2^l \times \mathbb{F}_2^n \mid c = F(a,k), \alpha(a,k) = \beta(c)\}, \\ \lambda_F(\alpha,\beta) &= (2p_F(\alpha,\beta) - 1)^2 = \frac{1}{2^{2n+2l}} \cdot \hat{\vartheta}_F(\alpha,\beta)^2, \end{aligned}$$

wobei in der Notation nicht zwischen einer Linearform und dem zugehörigen Vektor unterschieden wird. Zerlegt man $\alpha(a,k)$ in die Summe $\alpha'(a) + \gamma(k)$ – und schreibt dann statt α' einfach in neuer Bedeutung α –, so kann man sagen, dass $p_F((\alpha,\gamma),\beta)$ die Wahrscheinlichkeit dafür angibt, dass bei bekanntem Klartext a die lineare Relation

$$\gamma(k) = \alpha(a) + \beta(c)$$

für die Schlüsselbits $k_{i_1}, \ldots, k_{i_r}$ gilt, wenn $I = (i_1, \ldots, i_r)$ die Indexmenge ist, die der Linearform γ entspricht. Dabei ist $\gamma(k) = k_{i_1} + \cdots + k_{i_r}$ ein einzelnes Bit, das die durch I definierte Summe einiger Bits des Schlüssels k darstellt. Das Potenzial $\lambda_F((\alpha,\gamma),\beta)$ misst die Abweichung der Wahrscheinlichkeit vom Wert $\frac{1}{2}$, denn eine Wahrscheinlichkeit $< \frac{1}{2}$ ist genauso gut wie eine $> \frac{1}{2}$: Sie sagt, dass die komplementäre Relation

$$\gamma(k) = \alpha(a) + \beta(c) + 1$$

überzufällig oft gilt.

Daraus leitet man folgendes Vorgehen für die Schätzung von $\gamma(k)$ ab (im Fall $p_F > \frac{1}{2}$, sonst komplementär):

1. **[Sammelphase]** Man sammelt N Klartext-Geheimtextpaare $(a_1, c_1), \ldots, (a_N, c_N)$.

2. **[Auszählung]** Man bestimmt die Anzahl

$$t := \#\{i = 1, \ldots, N \mid \alpha(a) + \beta(c) = 0\}.$$

3. **[Mehrheitsentscheidung]** aufgrund von t:

- Ist $t > \frac{N}{2}$, schätzt man $\gamma(k) = 0$.
- Ist $t = \frac{N}{2}$, „randomisiert“ man die Entscheidung, d. h., man entscheidet sich zufällig für 0 oder 1, jeweils mit Wahrscheinlichkeit $\frac{1}{2}$.
- Ist $t < \frac{N}{2}$, schätzt man $\gamma(k) = 1$.

Wenn man ein lineare Relation mit hinreichend hohem Potenzial erwischt hat, wird die Erfolgswahrscheinlichkeit dieses Verfahrens bei hinreichend großem N hinreichend gut sein.

Findet man mehrere solche lineare Relationen mit hinreichender Gewissheit, so hat man den Schlüsselraum auf einen Unter-Vektorraum eingeschränkt und kann über diesen eine Exhaustion versuchen. Das ist die Grundidee der linearen Kryptoanalyse – es gibt je nach dem konkreten Aufbau einer Chiffre verschiedene Varianten, wie in den folgenden Abschnitten deutlich wird.

Als theoretisches Ergebnis aus der Analyse einer Chiffre erhält man dadurch einen Zusammenhang zwischen der Menge von benötigtem Klartext und der Erfolgswahrscheinlichkeit oder auch der Dimension des übriggebliebenen Suchraums.

Damit das Verfahren anwendbar ist, sind folgende Fragen zu klären:

1. Wie findet man lineare Relationen von möglichst großem Potenzial?
2. Da Bitblock-Chiffren meistens aus vielen Runden zusammengestzt, sind, fragt man weiter:
   (a) Wie findet man bei einer iterierten Bitblock-Chiffre brauchbare lineare Relationen für die Rundenfunktion?
   (b) Wie setzt man diese über die Runden hinweg zu linearen Relationen für die ganze Chiffre zusammen, so dass Aussagen über Schlüsselbits resultieren?
   (c) Wie bestimmt man die Wahrscheinlichkeit einer zusammengesetzten linearen Relation für die ganze Chiffre aus der für die einzelnen Runden?
3. Wie hängt die Erfolgswahrscheinlichkeit von der Zahl N der bekannten Klartext-Blöcke ab?

Die Antwort auf die erste Frage und Teil (a) der zweiten heißt: Aus dem linearen Profil, also durch FOURIER-Analyse. Die anschließenden Teilfragen führen zur Untersuchung von „linearen Pfaden“ und „linearen Hüllen“ und der Kumulation von Wahrscheinlichkeiten.

Nun ist die FOURIER-Analyse zwar sehr effizient, wenn man den Aufwand (Zeit und Speicherplatz) als Funktion der Größe des Inputs betrachtet. Leider wächst diese Größe aber exponenziell mit der Dimension; daher wird die

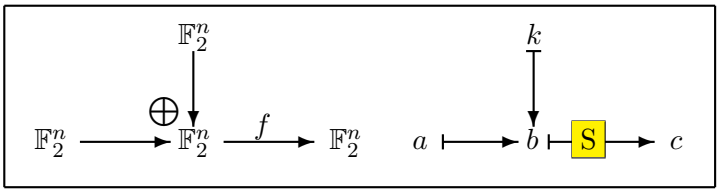
FOURIER-Analyse bei aller Effizienz schon bei Dimensionen von etwas über 10 undurchführbar; die für ernsthafte Blockchiffren relevanten Dimensionen, also Blockgrößen und Schlüssellängen von 64, besser aber 128 Bits, liegen weit darüber.

Dieser Einwand gilt auch für die erste Teilfrage von Frage 2. Da die einzelnen Runden aber meist wiederum im wesentlichen aus einer parallelen Abarbeitung kleinerer Stücke, der S-Boxen, bestehen, wird man versuchen, das Problem auf die Analyse der S-Boxen zurückzuspielen, und diese ist realistisch durchführbar; selbst das AES-Verfahren verwendet nur 8-dimensionale S-Boxen.

5.2 Beispiel: Eine Einrunden-Chiffre

Es werden Beispiele betrachtet, die als ernsthafte Blockchiffren viel zu einfach sind, aber das Prinzip der linearen Kryptoanalyse sehr anschaulich und nachvollziehbar demonstrieren. Dabei werden stets Rundenfunktionen der Gestalt $f(a+k)$ betrachtet, d. h., der Schlüssel wird vor der Anwendung einer bijektiven S-Box $f\colon \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ binär auf den Klartext aufaddiert. Das einfachste denkbare Modell, die Verschlüsselung nach der Vorschrift

$$c = f(a + k),$$

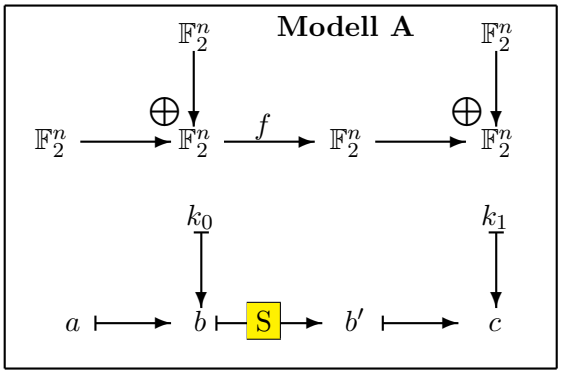


ist dabei witzlos, da bei bekanntem Klartext die Gleichung nach dem Schlüssel k auflösbar ist:

$$k = f^{-1}(c) + a.$$

Dieser einfache Angriff wird bei dem etwas komplizierteren Modell „A“

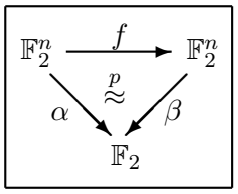
$$c = f(a + k_0) + k_1$$



verhindert [in den grafischen Darstellungen wird die Abbildung f immer durch die S-Box S repräsentiert]; hier ist der Ansatz der linearen Kryptoanalyse bereits sinnvoll: Sei (α, β) ein Paar von Linearformen mit

$$\beta \circ f(x) \overset{p}{\approx} \alpha(x),$$

wobei das Symbol $\overset{p}{\approx}$ gelesen wird als „ist gleich mit Wahrscheinlichkeit p“. Repräsentiert wird dies durch das Diagramm



Dann gilt

$$\begin{aligned}\beta(c) &= \beta(b' + k_1) = \beta(b') + \beta(k_1) \\ &\overset{p}{\approx} \alpha(b) + \beta(k_1) = \alpha(a + k_0) + \beta(k_1) = \alpha(a) + \alpha(k_0) + \beta(k_1).\end{aligned}$$

Hierbei wird k als fest angesehen und zur Spezifikation der Wahrscheinlichkeit über alle Klartexte a gemittelt. Als lineare Relation für die Bits des Schlüssels $k = (k_1, k_2)$ erhalten wir also

$$\alpha(k_0) + \beta(k_1) \overset{p}{\approx} \alpha(a) + \beta(c).$$

Sie gilt genau mit der Wahrscheinlichkeit $p = p_f(\alpha, \beta)$. Ein analoger Schluss lässt sich für die komplementäre Relation

$$\beta \circ f(x) \overset{1-p}{\approx} \alpha(x) + 1$$

durchführen. Insgesamt ist damit gezeigt:

Satz 1 *Im Modell A sei (α, β) eine lineare Relation für f mit der Wahrscheinlichkeit p. Dann ist $p_1 = \max\{p, 1-p\}$ die Erfolgswahrscheinlichkeit der linearen Kryptoanalyse mit* einem *bekannten Klartext.*

Nehmen wir zunächst als konkretes Beispiel $n = 4$ und für f die S-Box S_0 von LUCIFER. Aus der Analyse dieser BOOLEschen Abbildung wissen wir, dass das lineare Potenzial von $\frac{9}{16}$ z. B. von dem Paar $\alpha = 0001$ und $\beta = 1101$ mit $\hat{\vartheta}_f(\alpha, \beta) = 12$ angenommen wird. Die zugehörige Wahrscheinlichkeit ist $p_f(\alpha, \beta) = \frac{7}{8}$. Als konkrete Rundenschlüssel werden $k_0 = 1000$ und $k_1 = 0001$ gewählt. Eine Tabelle über alle 16 möglichen Klartexte sieht dann so aus (unter Verwendung der bekannten Wertetabelle von f):

a	b	b'	c	$\alpha(a)+\beta(c)$
0000	1000	0010	0011	1
0001	1001	0110	0111	1
0010	1010	0011	0010	0
0011	1011	0001	0000	1
0100	1100	1001	1000	1
0101	1101	0100	0101	1
0110	1110	0101	0100	1
0111	1111	1000	1001	1
1000	0000	1100	1101	1
1001	0001	1111	1110	1
1010	0010	0111	0110	1
1011	0011	1010	1011	1
1100	0100	1110	1111	1
1101	0101	1101	1100	1
1110	0110	1011	1010	1
1111	0111	0000	0001	0

Der Wert $1 = \alpha(k_0) + \beta(k_1)$ wird also, wie es sein soll, genau 14-mal angenommen.

Wie groß ist nun die Erfolgswahrscheinlichkeit p_N dafür, diesen Wert richtig zu schätzen, wenn man $N = 1, 2, \ldots$ zufällige bekannte Klartexte aus der Menge der 2^n möglichen zur Verfügung hat? (Zu gegebenen festen Linearformen α und β mit $p = p_f(\alpha, \beta)$ für einen beliebigen – unbekannten, gesuchten – Schlüssel k.) Das ist genau die Fragestellung der hypergeometrischen Verteilung, und daher gilt:

Satz 2 *Im Modell A sei (α, β) eine lineare Relation für f mit der Wahrscheinlichkeit $p = \frac{s}{2^n}$. Dann ist die Erfolgswahrscheinlichkeit der linearen Kryptoanalyse mit N bekannten Klartexten gerade die kumulierte Wahrscheinlichkeit $p_N = p_N^{(s)}$ der hypergeometrischen Verteilung zu den Parametern 2^n, $s = p_1 2^n$ und N mit $p_1 = \max\{p, 1-p\}$.*

Korollar 1 *$p_N = 1$, wenn $N > 2^{n+1} \cdot (1 - p_1)$.*

Im konkreten Beispiel oben wird diese Bedingung zu $N > 32 \cdot \frac{1}{8} = 4$, also $N \geq 5$.

Korollar 2 (Asymptotik) *Ist $p \approx \frac{1}{2}$, $N \ll 2^n$ und N nicht zu klein, so*

$$p_N \approx \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\sqrt{r\lambda}} e^{-t^2/2} dt.$$

5.3 Die hypergeometrische Verteilung

Das Urnenmodell für die hypergeometrische Verteilung ist die „Ziehung ohne Zurücklegen“. Die Urne enthalte n Kugeln, davon s schwarze und $w = n - s$ weiße. Der Anteil

$$p := \frac{s}{n}$$

der schwarzen Kugeln sei bekannt und o. B. d. A. $p > \frac{1}{2}$. (Der Fall $p = \frac{1}{2}$ ist uninteressant, der Fall $p < \frac{1}{2}$ symmetrisch zum ersten.)

> In der Anwendung auf die lineare Kryptoanalyse werden die Kugeln alle möglichen Klartexte sein und die „Ziehung“ die Auswertung einer linearen Relation für einen bekannten Klartext.

Es werden r Kugeln zufällig gezogen ($r \leq n$). Die Wahrscheinlichkeit, dabei genau ν weiße Kugeln zu ziehen, ist

$$q_r^{(s)}(\nu) = \frac{\binom{s}{r-\nu}\binom{w}{\nu}}{\binom{n}{r}}.$$

Die Funktion

$$q_r^{(s)} \colon \mathbb{Z} \longrightarrow \mathbb{R}$$

heißt die **hypergeometrische Verteilung** (zu den Parametern n, s und r). Dabei ist $q_r^{(s)}(\nu) = 0$ für $\nu < 0$ und für $\nu > r$. Die Wahrscheinlichkeit, dass mehr schwarze als weiße Kugeln gezogen werden, ist

$$p_r^{(s)} = \begin{cases} \sum_{\nu=0}^{\frac{r-1}{2}} q_r^{(s)}(\nu), & \text{wenn } r \text{ ungerade,} \\ \sum_{\nu=0}^{\frac{r}{2}-1} q_r^{(s)}(\nu) + \frac{1}{2} q_r^{(s)}(\frac{r}{2}), & \text{wenn } r \text{ gerade,} \end{cases}$$

wenn im Falle des Gleichstands zufällig mit Wahrscheinlichkeit jeweils $\frac{1}{2}$ für schwarz oder weiß entschieden wird.

> Im uninteressanten Fall $p = \frac{1}{2}$ sind offensichtlich alle $p_r^{(s)} = \frac{1}{2}$.

Hilfssatz 1 *Es gilt:*
(i) $p_1^{(s)} = p$.
(ii) $p_2^{(s)} = p_1^{(s)}$ (falls $w \geq 1$).
(iii) $p_3^{(s)} = \frac{s(s-1)}{n(n-1)} \cdot \left[3 - 2 \cdot \frac{s-2}{n-2}\right]$ (falls $w \geq 2$).
(iv) $p_4^{(s)} = p_3^{(s)}$ (falls $w \geq 2$).
(v) $p_r^{(s)} = 1$ für $r > 2w$.

Beweis. (i) ist trivial.

(ii) Da bei jeweils einer weißen und schwarzen Kugel zufällig entschieden wird, ist der Zähler gleich

$$\binom{s}{2} + \frac{1}{2}\binom{s}{1}\binom{w}{1} = \frac{s(s-1)}{2} + \frac{s(n-s)}{2} = \frac{s(n-1)}{2},$$

der Nenner gleich $\frac{n(n-1)}{2}$, der Quotient

$$p_2^{(s)} = \frac{s(n-1)}{n(n-1)} = p.$$

(iii) Hier ist der Zähler

$$\begin{aligned}\binom{s}{3} + \binom{s}{2} \cdot (n-s) &= \frac{s(s-1)(s-2) + 3s(s-1)(n-s)}{6} \\ &= \frac{s(s-1)}{6} \cdot [s-2+3\cdot(n-s)] \\ &= \frac{s(s-1)}{6} \cdot [3\cdot(n-2) - 2\cdot(s-2)].\end{aligned}$$

Der Nenner ist $\frac{1}{6} \cdot n(n-1)(n-2)$, also hat $p_3^{(s)}$ den behaupteten Wert.

(iv) Die Rechnung wird weggelassen, da im nächsten Hilfssatz eine allgemeinere Aussage bewiesen wird.

(v) folgt, weil dann auf jeden Fall mehr schwarze Kugeln gezogen werden. ◇

Hilfssatz 2 *Ist r gerade und $2 \le r \le 2w$, so*

$$p_{r+1}^{(s)} > p_r^{(s)} = p_{r-1}^{(s)}.$$

Beweis. Sei $A_r^{(s)}(\nu) = \binom{n}{r} \cdot q_r^{(s)}(\nu)$ der Zähler von $q_r^{(s)}(\nu)$ und $B_r^{(s)} = \binom{n}{r} \cdot p_r^{(s)}$ der Zähler von $p_r^{(s)}$.

Beim Übergang von r nach $r+1$ wird die Mehrheitsentscheidung „schwarz“ nach $r+1$ Zügen in $B_{r+1}^{(s)}$ Fällen getroffen. Darunter sind:

- $\sum_{\nu=0}^{\frac{r}{2}-1} A_r^{(s)}(\nu)$ Fälle, in denen bereits nach r Zügen mindestens $\frac{r}{2}+1$ schwarze Kugeln gezogen worden waren. Für die $(r+1)$-te Kugel gibt es noch $n-r$ Möglichkeiten, die aber alle an der Entscheidung nichts ändern. Wir haben hier also

$$X_1 = (n-r) \cdot \sum_{\nu=0}^{\frac{r}{2}-1} A_r^{(s)}(\nu)$$

  Fälle, in denen „schwarz“ enschieden wird.

- $A_r^{(s)}(\frac{r}{2})$ Fälle, bei denen nach r Zügen genau $\frac{r}{2}$ schwarze Kugeln gezogen worden waren. Von den $n-r$ Möglichkeiten für die $(r+1)$-te Kugel sind
  - $s-\frac{r}{2}$ schwarz und führen zur Entscheidung „schwarz“,
  - $w-\frac{r}{2}$ weiß und führen zur Entscheidung „weiß“.

  Es kommen also
  $$X_2 = (s-\frac{r}{2}) \cdot A_r^{(s)}(\frac{r}{2})$$
  Fälle hinzu, in denen „schwarz“ enschieden wird.

- In den übrigen Fällen liegen nach r Zügen höchstens $\frac{r}{2}-1$ schwarze Kugeln vor, und die $(r+1)$-te Kugel kann somit die Entscheidung für „weiß“ nicht ändern.

Da von den gezählten Fällen jeweils $r+1$ dieselbe Menge von gezogenen Kugeln ergeben, ist

$$B_{r+1}^{(s)} = \frac{1}{r+1} \cdot (X_1 + X_2) = \frac{n-r}{r+1} \cdot \left[\sum_{\nu=0}^{\frac{r}{2}-1} A_r^{(s)}(\nu) + \frac{s-\frac{r}{2}}{n-r} \cdot A_r^{(s)}(\frac{r}{2}) \right].$$

Für den Koeffizienten des letzten Terms gilt

$$\frac{s-\frac{r}{2}}{n-r} > \frac{1}{2} \Longleftrightarrow 2s-r > n-r \Longleftrightarrow s > \frac{n}{2}.$$

(Da $r \leq 2w$, ist $r < n$.) Also folgt

$$B_{r+1}^{(s)} > \frac{n-r}{r+1} \cdot B_r^{(s)}$$

und somit der erste Teil der Behauptung.

Etwas komplizierter ist der Übergang von $r-1$ nach r. Die Entscheidung „schwarz“ wird nach r Zügen in $B_r^{(s)}$ Fällen getroffen. Darunter sind

- $\sum_{\nu=0}^{\frac{r}{2}-2} A_{r-1}^{(s)}$ Fälle, wo nach $r-1$ Zügen mindestens $\frac{r}{2}+1$ schwarze Kugeln gezogen worden waren. Die $n-r+1$ Möglichkeiten für die r-te Kugel ändern die Entscheidung nicht. Es gibt hier also
  $$Y_1 = (n-r+1) \cdot \sum_{\nu=0}^{\frac{r}{2}-2} A_{r-1}^{(s)}$$
  Fälle, in denen „schwarz“ entschieden wird.

- $A_{r-1}^{(s)}(\frac{r}{2}-1)$ Fälle, wo nach $r-1$ Zügen genau $\frac{r}{2}$ schwarze Kugeln gezogen worden waren. Die $n-r+1$ Möglichkeiten für die r-te Kugel zerfallen in

– $s - \frac{r}{2}$ schwarze, die zu der Entscheidung „schwarz“ führen; hier gibt es also

$$Y_2 = (s - \frac{r}{2}) \cdot A^{(s)}_{r-1}(\frac{r}{2} - 1)$$

zusätzliche Fälle.

– $w + 1 - \frac{r}{2}$ weiße, wo die Entscheidung mit jeweils der Wahrscheinlichkeit $\frac{1}{2}$ zufällig getroffen wird; es kommen also

$$Y_3 = \frac{1}{2} \cdot (w + 1 - \frac{r}{2}) \cdot A^{(s)}_{r-1}(\frac{r}{2} - 1)$$

Fälle hinzu.

- $A^{(s)}_{r-1}(\frac{r}{2})$ Fälle, wo nach $r - 1$ Zügen genau $\frac{r}{2} - 1$ schwarze Kugeln gezogen worden waren. Die $n - r + 1$ Möglichkeiten für die r-te Kugel zerfallen in

  – $s + 1 - \frac{r}{2}$ schwarze, wo die Entscheidung zufällig mit jeweils der Wahrscheinlichkeit $\frac{1}{2}$ getroffen wird – es kommen also

$$Y_4 = \frac{1}{2} \cdot (s + 1 - \frac{r}{2}) \cdot A^{(s)}_{r-1}(\frac{r}{2})$$

  Fälle hinzu –,

  – $w - \frac{r}{2}$ weiße, in denen die Entscheidung bei „weiß“ bleibt.

- In den übrigen Fällen, wo nach $r - 1$ Zügen höchstens $\frac{r}{2} - 2$ schwarze Kugeln gezogen worden waren, bleibt die Entscheidung ebenfalls bei „weiß“.

Da jeweils r der gezählten Fälle dieselbe Menge von gezogenen Kugeln ergeben, gilt

$$\begin{aligned} B^{(s)}_r &= \frac{1}{r} \cdot (Y_1 + Y_2 + Y_3 + Y_4) \\ &= \frac{n - r + 1}{r} \cdot \sum_{\nu=0}^{\frac{r}{2}-2} A^{(s)}_{r-1} + \frac{1}{r} \cdot (s - \frac{r}{2} + \frac{w}{2} + \frac{1}{2} - \frac{r}{4}) \cdot A^{(s)}_{r-1}(\frac{r}{2} - 1) \\ &\quad + \frac{1}{2r} \cdot (s - \frac{r}{2} + 1) \cdot A^{(s)}_{r-1}(\frac{r}{2}) \end{aligned}$$

Da $s + \frac{w}{2} = n - \frac{w}{2}$, ist der Koeffizient des mittleren Terms gleich

$$s - \frac{r}{2} + \frac{w}{2} - \frac{r}{4} + \frac{1}{2} = n - \frac{w}{2} - r + \frac{r}{4} + 1 - \frac{1}{2} = (n - r + 1) - \frac{1}{2} \cdot (w - \frac{r}{2} + 1).$$

Also ist

$$\begin{aligned} B^{(s)}_r &= \frac{n - r + 1}{r} \cdot \sum_{\nu=0}^{\frac{r}{2}-1} A^{(s)}_{r-1} \\ &\quad - \frac{1}{2r}(w - \frac{r}{2} + 1)\binom{s}{\frac{r}{2}}\binom{w}{\frac{r}{2} - 1} + \frac{1}{2r}(s - \frac{r}{2} + 1)\binom{s}{\frac{r}{2} - 1}\binom{w}{\frac{r}{2}}. \end{aligned}$$

Die beiden letzten Terme heben sich weg, und es bleibt

$$B_r^{(s)} = \frac{n-r+1}{r} \cdot B_{r-1}^{(s)}.$$

Daraus folgt der zweite Teil der Behauptung. $\diamond$

Damit ist insbesondere gezeigt:

Satz 3 *Die Wahrscheinlichkeit $p_r^{(s)}$ wächst mit r monoton von $p_1^{(s)} = p$ bis $p_{2w+1}^{(s)} = 1$.*

Wenn die Quotienten

$$\frac{rs}{n}, \frac{rw}{n}, \frac{(n-r)s}{n}, \frac{(n-r)w}{n}$$

hinreichend groß sind (FISHERs Faustregel sagt: ≥ 5 reicht), kann man die hypergeometrische Verteilung durch die Normalverteilung approximieren; das bedeutet insbesondere

$$\sum_{\nu=0}^{x} q_r^{(s)}(\nu) \approx \Phi(\frac{x-\mu}{\sigma}) = \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\frac{x-\mu}{\sigma}} e^{-t^2/2} dt,$$

wobei μ der Mittelwert und σ^2 die Varianz der hypergeometrischen Verteilung (zu den Parametern n, s und r) und Φ die Verteilungsfunktion der Normalverteilung ist. Für Mittelwert und Varianz gilt

Hilfssatz 3

$$\begin{aligned} \mu &= \frac{rw}{n}, \\ \sigma^2 &= \frac{r(n-r) \cdot w(n-w)}{n^2(n-1)}. \end{aligned}$$

Beweis. Bei einer zufälligen Stichprobenziehung von r Kugeln der Reihe nach sei $X_k \colon \Omega \longrightarrow \mathbb{R}$ eine Zufallsvariable, die 0 ist, wenn die k-te Kugel schwarz ist, und 1, wenn sie weiß ist. Dann ist $S = X_1 + \cdots + X_r : \Omega \longrightarrow \mathbb{R}$ eine Zufallsvariable, die die Anzahl der weißen Kugeln in der Stichprobenziehung angibt. Es ist $\mu = \mathrm{E}(S)$ der Erwartungswert und $\sigma^2 = \mathrm{Var}(S)$ die Varianz dieser Zufallsvariablen.

Klar ist $\mathrm{E}(X_k) = \frac{w}{n}$ also $\mathrm{E}(S) = r \cdot \frac{w}{n}$.

Für die Berechnung der Varianz bemerken wir zuerst, dass $X_k^2 = X_k$, also

$$\mathrm{Var}(X_k) = \mathrm{E}(X_k^2) - \mathrm{E}(X_k)^2 = \frac{w}{n} - \frac{w^2}{n^2} = \frac{w(n-w)}{n^2}.$$

Da $X_j X_k(\omega) = 1 \iff X_j(\omega) = 1$ *und* $X_k(\omega) = 1$, ist die Wahrscheinlichkeit dafür $\frac{w(w-1)}{n(n-1)}$, der Erwartungswert also $\mathrm{E}(X_j X_k) = \frac{w(w-1)}{n(n-1)}$. Daher ist die Covarianz

$$\begin{aligned}\mathrm{Cov}(X_j, X_k) &= \mathrm{E}(X_j X_k) - \mathrm{E}(X_j)\mathrm{E}(X_k) = \frac{w(w-1)}{n(n-1)} - \frac{w^2}{n^2} \\ &= \frac{w(n(w-1) - w(n-1))}{n^2(n-1)} = \frac{w(w-n)}{n^2(n-1)}.\end{aligned}$$

Die Varianz von S ist also

$$\begin{aligned}\mathrm{Var}(S) &= \sum_{k=1}^{r} \mathrm{Var}(X_k) + 2 \cdot \sum_{1 \le j < k \le r} \mathrm{Cov}(X_j, X_k) \\ &= \frac{rw(n-w)}{n^2} + r(r-1) \cdot \frac{w(w-n)}{n^2(n-1)} = \frac{rw(n-w)}{n^2} \cdot \left[1 - \frac{r-1}{n-1}\right] \\ &= \frac{rw(n-w)}{n^2(n-1)} \cdot [n-r],\end{aligned}$$

wie behauptet. $\diamond$

Satz 4 (Asymptotische Verteilung) *Die Wahrscheinlichkeit, mehr schwarze Kugeln zu ziehen, ist*

$$p_r^{(s)} \approx \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\sqrt{r\lambda}} e^{-t^2/2} dt$$

mit $\lambda = (2p-1)^2$*, wenn* $p \approx \frac{1}{2}$*,* $r \ll n$ *und* r *nicht zu klein.*

[Nach FISHERs Faustregel reicht $10 \le r \le n - 10$ für $p \approx \frac{1}{2}$.]
Beweis. Die obere Grenze des Integrals ist für $x = \frac{r}{2}$ zu berechnen:

$$\begin{aligned}\frac{x-\mu}{\sigma} &= \frac{(\frac{r}{2} - \frac{rw}{n}) \cdot n \cdot \sqrt{n-1}}{\sqrt{r(n-r)w(n-w)}} = \frac{(rn - 2rw)\sqrt{n-1}}{2 \cdot \sqrt{r(n-r)w(n-w)}} \\ &= \frac{\sqrt{r}\sqrt{n-1}}{\sqrt{n-r}} \cdot \frac{s-w}{2\sqrt{sw}} = \frac{\sqrt{n-1}}{\sqrt{n-r}} \cdot \sqrt{r} \cdot \frac{2p-1}{2\sqrt{p(1-p)}} \\ &\approx 1 \cdot \sqrt{r} \cdot \frac{2p-1}{2 \cdot \sqrt{\frac{1}{4}}} = \sqrt{r\lambda},\end{aligned}$$

wie behauptet. $\diamond$

5.4 Die Erfolgswahrscheinlichkeit

Die Überlegung aus Abschnitt 5.2 sieht im allgemeinen Rahmen von Abschnitt 5.1 genauso aus und liefert eine zufriedenstellende Antwort auf die dortige Frage 2:

Hauptsatz 1 (Formel von MATSUI**)** *Sei (α, β) eine lineare Relation mit Wahrscheinlichkeit $p = p_F(\alpha, \beta)$ und Potenzial $\lambda = \lambda_F(\alpha, \beta)$ für die Bitblock-Chiffre $F : \mathbb{F}_2^n \times \mathbb{F}_2^l \longrightarrow \mathbb{F}_2^n$. Dann ist die Erfolgswahrscheinlichkeit $P_{\alpha\beta N}$ der linearen Kryptoanalyse mit N bekannten Klartexten gerade die kumulierte Wahrscheinlichkeit $p_N^{(s)}$ der hypergeometrischen Verteilung zu den Parametern 2^n, $s = 2^n \cdot \max\{p, 1-p\}$ und N. Ist $p \approx \frac{1}{2}$, $N \ll 2^n$ und N nicht zu klein, so*

$$P_{\alpha\beta N} \approx \frac{1}{\sqrt{2\pi}} \cdot \int_{-\infty}^{\sqrt{N\lambda}} e^{-t^2/2} dt.$$

Für die exakte Verteilung haben wir die Bedingung

$$p_{\alpha\beta N} = 1, \quad \text{wenn } N > 2^{n+1}(1 - p_1).$$

Diese nützt für $p \approx \frac{1}{2}$ nichts – der Aufwand für die lineare Kryptoanalyse ist genauso groß wie der für die Exhaustion. Verzichtet man aber auf die hundertprozentige Gewissheit, so ergibt die Näherungsformel zusammen mit den bekannten Regeln für die Normalverteilung die Tabelle

$N\lambda$	1	2	3	4	...	8	9
$P_{\alpha\beta N}$	84.1%	92.1%	95.8%	97.7%	...	99.8%	99.9%

D. h., um eine Erfolgswahrscheinlichkeit von 97.7% zu erreichen, braucht man $N \approx \frac{4}{\lambda}$ bekannte Klartexte.

Zahlenbeispiel für DES: Das höchste Potenzial einer linearen Relation ist $\lambda \approx (3 \cdot 2^{-24})^2$ (siehe später), die entsprechende Wahrscheinlichkeit $p \approx \frac{1}{2} - 3 \cdot 2^{-25}$. Für eine Erfolgswahrscheinlichkeit von 97.7% benötigt man also

$$N \approx \frac{4}{\lambda} \approx \frac{4 \cdot 2^{48}}{3^2} \approx 2^{47}$$

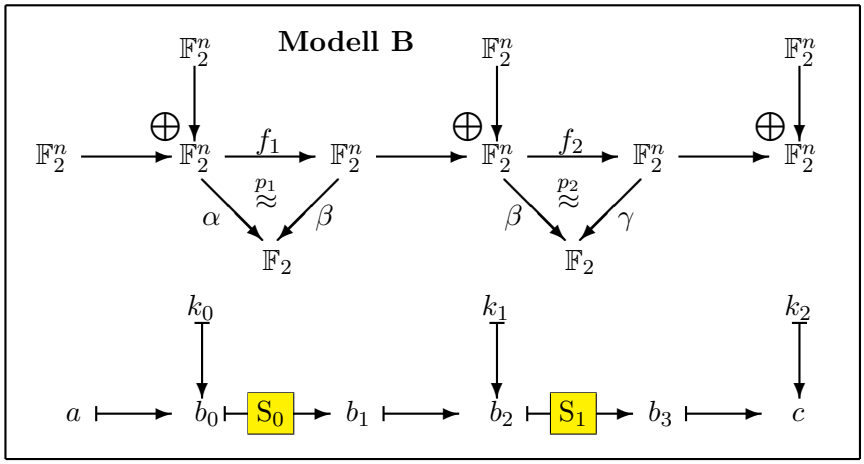
bekannte Klartexte. Damit hat man dann *ein* Schlüsselbit mit hoher Wahrscheinlichkeit bestimmt.

> Leichte Verbesserungen: Mit einer linearen Relation für die Runden 2 bis 15 und einer vollständigen Suche über die relevanten Schlüsselbits der Runden 1 und 16 konnte MATSUI die Anzahl der benötigten Klartexte auf 2^{43} drücken; durch gleichzeitiges Betrachten mehrerer linearer Relationen konnte er ferner die Anzahl der gewonnenen Schlüsselbits auf 14 erhöhen. Es bleibt die vollständige Suche nach den übrigen 42 Schlüsselbits, die auf einem PC nur noch wenige Sekunden dauert.

Dies ist der effizienteste bekannte Angriff auf das DES-Verfahren; die Anzahl der benötigten Klartexte ist allerdings immer noch so groß, daß dieser Angriff kaum als realistische Gefahr eingestuft werden kann. Dennoch offenbart er eine leichte Schwäche, die beim Design des DES übersehen wurde. Die Stabilität des DES gegen differenzielle Kryptoanalyse ist deutlich besser; diese Angriffsmöglichkeit war ja, wie heute bekannt ist, beim Design berücksichtigt worden.

5.5 Beispiel: Eine Zweirunden-Chiffre

Für die Analyse von Chiffren über mehrere Runden beginnen wir wieder mit einem einfachen Beispiel, dem Modell „B“:



Die Verschlüsselung geschieht also sukzessive nach den Formeln

$$b_0 = a + k_0,\ b_1 = f_1(b_0),\ b_2 = b_1 + k_1,\ b_3 = f_2(b_2),\ c = b_3 + k_2,$$

oder in einem Schritt:

$$c = f_2[f_1(a + k_0) + k_1] + k_2.$$

[Dabei wird f_1 durch die S-Box S_0 und f_2 durch die S-Box S_1 beschrieben.]

Nach Analyse der S-Boxen wissen wir über die linearen Relationen für die Rundenabbildungen f_1 und f_2 Bescheid. Was können wir daraus über lineare Relationen für die ganze Chiffre herleiten?

Sei (α, β) eine lineare Relation für f_1 mit Wahrscheinlichkeit p_1 und (β, γ) eine für f_2 mit Wahrscheinlichkeit p_2. Dann gilt

$$\begin{aligned} \gamma(c) &= \gamma(b_3) + \gamma(k_2) \overset{p_2}{\approx} \beta(b_2) + \gamma(k_2) = \beta(b_1) + \beta(k_1) + \gamma(k_2) \\ &\overset{p_1}{\approx} \alpha(b_0) + \beta(k_1) + \gamma(k_2) = \alpha(a) + \alpha(k_0) + \beta(k_1) + \gamma(k_2) \end{aligned}$$

Wir erhalten also eine Relation für das eine Schlüsselbit $\alpha(k_0)+\beta(k_1)+\gamma(k_2)$ in der Form

$$\alpha(k_0) + \beta(k_1) + \gamma(k_2) \overset{p}{\approx} \alpha(a) + \gamma(c)$$

mit noch unbekannter Wahrscheinlichkeit p. Diese ist im allgemeinen sehr schwer explizit zu bestimmen. Betrachten wir das folgende *konkrete Beispiel*:

Es sei $n = 4$, und S_0 und S_1 seien die beiden S-Boxen von Lucifer. Die Linearformen $\alpha = 0001$ und $\beta = 1101$ seien wie in Abschnitt 5.2 gewählt. Passend dazu sei $\gamma = 1100$ gewählt, so dass das Paar (β, γ) das maximale Potenzial $\frac{1}{4}$ für S_1 annimmt, und $\hat{\vartheta}_{f_2}(\beta, \gamma) = -8$. Als konkrete Rundenschlüssel werden $k_0 = 1000$, $k_1 = 0001$ – wie in 5.2 – und $k_2 = 0110$ gewählt. Die Tabelle über alle 16 möglichen Klartexte ist:

a	b_0	b_1	b_2	b_3	c	$\alpha(a)+\gamma(c)$
0000	1000	0010	0011	1001	1111	0
0001	1001	0110	0111	0100	0010	1
0010	1010	0011	0010	1110	1000	1
0011	1011	0001	0000	0111	0001	1
0100	1100	1001	1000	1100	1010	1
0101	1101	0100	0101	1011	1101	1
0110	1110	0101	0100	0011	0101	1
0111	1111	1000	1001	1101	1011	0
1000	0000	1100	1101	1111	1001	1
1001	0001	1111	1110	1000	1110	1
1010	0010	0111	0110	0000	0110	1
1011	0011	1010	1011	1010	1100	1
1100	0100	1110	1111	0101	0011	0
1101	0101	1101	1100	0110	0000	1
1110	0110	1011	1010	0001	0111	1
1111	0111	0000	0001	0010	0100	0

Da $\hat{\vartheta}_{f_2}(\beta,\gamma) = -8$, soll das Bit $\alpha(k_0)+\beta(k_1)+\gamma(k_2)+1 = 1$ erkannt werden; dies geschieht in 12 von 16 Fällen korrekt, also mit Wahrscheinlichkeit $p = \frac{3}{4}$ bzw. mit Potenzial $\lambda = \frac{1}{4}$.

Es gibt auch andere „Pfade“ von α nach γ, z. B. über $\beta' = 0001$ mit $\hat{\vartheta}_{f_1}(\alpha,\beta') = -4$, $\lambda'_1 = \frac{1}{16}$, $p'_1 = \frac{1}{4}$ und $\hat{\vartheta}_{f_2}(\beta,\gamma) = 4$, $\lambda'_2 = \frac{1}{16}$, $p'_2 = \frac{3}{4}$. Hier wird also versucht, das Bit $\alpha(k_0)+\beta'(k_1)+\gamma(k_2)+1 = 1$ zu finden. Auch hierfür ist die Erfolgswahrscheinlichkeit also $p' = \frac{3}{4}$.

5.6 Binäre Summen binärer Zufallsvariablen

Satz 5 („Piling Up Lemma“) *Sei Ω ein Wahrscheinlichkeitsraum und*

$$X_1, \ldots, X_r : \Omega \longrightarrow \mathbb{F}_2$$

unabhängige Zufallsvariablen mit $p_i := P(X_i = 0) = P(X_i^{-1}(0))$. Sei $X = X_1 + \cdots + X_r$ (binäre Summe in $\mathbb{F}_2$) und $p := P(X = 0)$. Sei $\lambda_i = (2p_i - 1)^2$ für $i = 1, \ldots r$ und $\lambda = (2p - 1)^2$. Dann gilt $\lambda = \lambda_1 \cdots \lambda_r$.

Beweis. Es reicht, den Beweis für $r = 2$ zu führen. Für $\omega \in \Omega$ ist $X(\omega) = 0$ genau dann, wenn $X_1(\omega)$ und $X_2(\omega)$ *beide* 0 *oder beide* 1 sind. Also ist

$$\begin{aligned} p &= p_1p_2 + (1 - p_1)(1 - p_2) = 1 - p_1 - p_2 + 2p_1p_2, \\ 2p - 1 &= 4p_1p_2 - 2p_1 - 2p_2 + 1 = (2p_1 - 1)(2p_2 - 1), \\ \lambda &= \lambda_1\lambda_2, \end{aligned}$$

wie behauptet. $\diamond$

Für die Wahrscheinlichkeiten ist die Formel etwas komplizierter:

Korollar 1 *Mit den Bezeichnungen von Satz 5 gilt*

$$p = \frac{1}{2} + 2^{r-1} \cdot \prod_{i=1}^{r} (p_i - \frac{1}{2}).$$

Korollar 2 *Ist ein $p_i = \frac{1}{2}$, so $\lambda = 0$, $p = \frac{1}{2}$.*

Korollar 3 *Mit wachsendem r ist λ monoton fallend.*

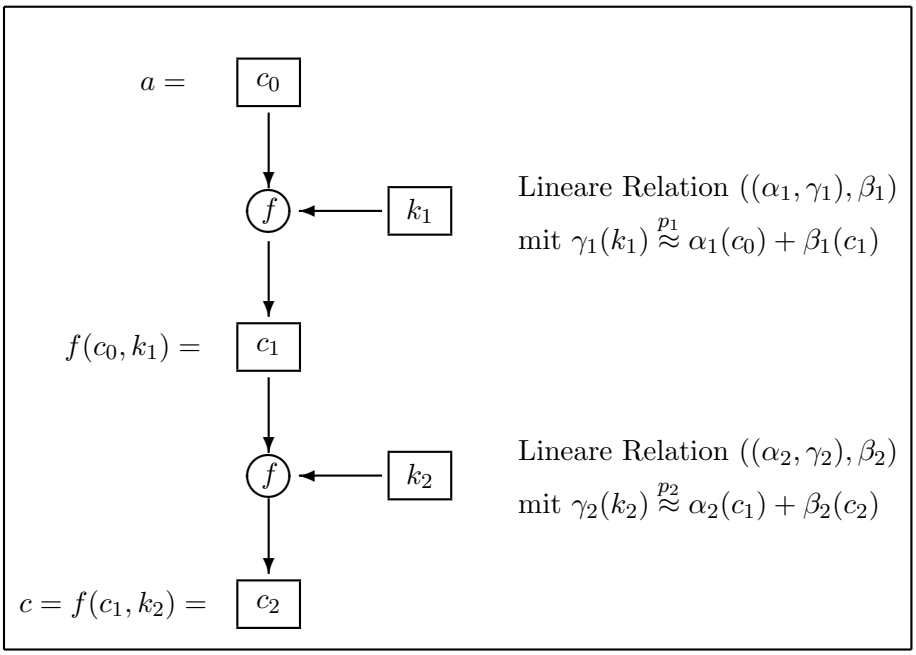
5.7 Lineare Pfade und lineare Hüllen

Die Rundenabbildung

$$f\colon \mathbb{F}_2^n \times \mathbb{F}_2^q \longrightarrow \mathbb{F}_2^n$$

einer Bitblock-Chiffre werde jetzt über mehrere Runden iteriert mit *unabhängigen* Rundenschlüsseln $k_i \in \mathbb{F}_2^q$.

Zunächst wird der Fall von zwei Runden behandelt.



Es gelten also die linearen Relationen

$$\gamma_1(k_1) \overset{p_1}{\approx} \alpha_1(c_0) + \beta_1(c_1)$$

mit Wahrscheinlichkeit p_1 und Potenzial $\lambda_1 = (2p_1 - 1)^2$ und

$$\gamma_2(k_2) \overset{p_2}{\approx} \alpha_2(c_1) + \beta_2(c_2)$$

mit Wahrscheinlichkeit p_2 und Potenzial $\lambda_2 = (2p_2 - 1)^2$. Die beiden linearen Relationen sind **kombinierbar**, wenn $\alpha_2 = \beta_1$. Dann gilt

$$\gamma_1(k_1) + \gamma_2(k_2) \overset{p}{\approx} \alpha_1(c_0) + \beta_2(c_2)$$

mit einer Wahrscheinlichkeit p und einem Potenzial λ, die unter der Annahme, dass die Relationen stochastisch unabhängig, insbesondere k_1 und k_2 unabhängig gewählt sind, aus dem Piling-Up-Lemma folgen. Danach wäre $\lambda = \lambda_1\lambda_2$. Das Beispiel in Abschnitt 5.5 zeigt, dass das nicht gilt; die Annahme unabhängiger Rundenschlüssel reicht nicht. Die Situation wird dadurch

verkompliziert, dass es von der Relation α_1 zur Relation β_2 mehrere „Pfade“, d. h., mehrere Zwischenschritte gibt. Ferner werden, wenn die Rundenzahl größer ist, dabei jedesmal andere Schlüsselbits der Zwischenrunden herausgepickt.

Im Beispiel war $\lambda_1 = \frac{9}{16}$, $\lambda_2 = \frac{1}{4}$ und $\lambda = \frac{1}{4} = \frac{16}{64}$ deutlich größer als $\lambda_1\lambda_2 = \frac{9}{64}$.

Um wenigstens den begrifflichen Rahmen, wenn schon nicht die Ergebnisse, zu präzisieren, definiert man: Gegeben sei eine über r Runden iterierte Bitblock-Chiffre. Sei $((\alpha_i, \gamma_i), \beta_i)$ eine lineare Relation für die i-te Runde mit Potenzial λ_i. Es sei $\alpha_i = \beta_{i-1}$ für $i = 2, \ldots, r$. Sei $\beta_0 := \alpha_1$. Dann heißt die Kette $(\beta_0, \ldots, \beta_r)$ ein **linearer Pfad** für die Chiffre mit Potenzial $\lambda := \lambda_1 \cdots \lambda_r$. Die **lineare Hülle** [Nyberg 1994] zu dem Paar (β_0, β_r) ist die Menge aller linearen Pfade, die β_0 mit β_r verbinden. Ein linearer Pfad heißt **dominant**, wenn sein Potenzial maximal unter allen linearen Pfaden der zugehörigen linearen Hülle ist.

Achtung: Das Potenzial eines linearen Pfades ist im allgemeinen *nicht* das Potenzial der resultierenden linearen Relation. Vielmehr gilt (ohne Beweis):

Satz 6 (Matsui) *Gegeben sei eine über r Runden iterierte Bitblock-Chiffre mit unabhängigen Rundenschlüsseln. Es sei eine lineare Relation für jede Runde gegeben, die das Potenzial $\lambda_1, \ldots, \lambda_r$ haben und zusammen einen linearen Pfad bilden. Dann hat die kombinierte lineare Relation das Potenzial $\lambda \geq \lambda_1 \cdots \lambda_r$. Ist der lineare Pfad dominant, so gilt $\lambda \approx \lambda_1 \cdots \lambda_r$.*

Dieses Ergebnis vermittelt eine konkrete Vorstellung davon, wie der Nutzen von linearen Approximationen mit jeder Runde, wo es keine lineare Relation mit Wahrscheinlichkeit 1 oder 0 gibt, weiter abnimmt, d. h., wie die Sicherheit der Chiffre vor linearer Kryptoanalyse mit zunehmender Rundenzahl steigt.

Die Methode der linearen Kryptoanalyse beruht also auf der Faustregel:

> *Entlang eines linearen Pfades multiplizieren sich die linearen Potenziale (nach Definition). Das Potenzial einer linearen Hülle wird durch das Potenzial des dominanten linearen Pfades ausreichend approximiert.*

Allerdings muss man beachten, dass der Satz nur eine Untergrenze für das Potenzial angibt, also eine obere Schranke für den Aufwand der linearen Kryptoanalyse. Für den Sicherheitsnachweis der Chiffre bräuchte man eine untere Schranke für den Aufwand, also eine obere Schranke für das Potenzial einer kombinierten linearen Relation. Hier gilt nur die empirisch ermittelte Näherungsaussage des Satzes; ferner sind grobe obere Schranken bekannt, die allerdings nicht zur Beruhigung des Kryptographen ausreichen.

Weiter ist bei der Anwendung zu beachten, dass bei konkreten Chiffren die Rundenschlüssel nicht unabhängig sind. Allerdings ist (nach empirischen

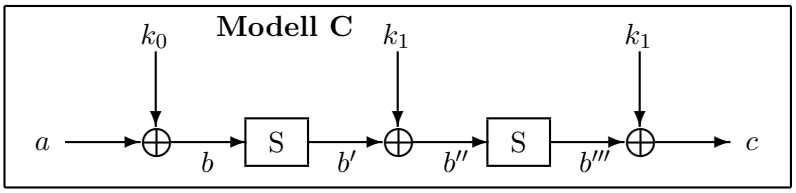
Erfahrungen) wie so oft in der Statistik der Effekt dieser Abhängigkeit vernachlässigbar, wenn die Schlüsselauswahl für die einzelnen Runden wenigstens ein bisschen komplex ist.

Auf diese Weise kommt auch das in Abschnitt 5.4 genannte Ergebnis für DES zustande.

5.8 Beispiel: Ein einfaches SP-Netz

Bisher haben wir in den Beispielen nur S-Boxen, keine Permutationen betrachtet. Das hat bei nur einer S-Box auch keinen Sinn, denn

Modell „C“:



5.9 Die Idee der differenziellen Kryptoanalyse

Bei der differenziellen Kryptoanalyse wird analog zur linearen Kryptoanalyse die Approximation durch lineare Strukturen verwendet. Man betrachtet einen Differenzenvektor vor Anwendung einer Rundenabbildung und seine möglichen Werte nach Anwendung der Rundenabbildung. Zusammenpassende Folgen von Differenzenvektoren über die Runden einer iterierten Bitblock-Chiffre werden als **differenzieller Pfad** oder **Charakteristik** [Biham/Shamir 1990] bezeichnet; das Potenzial eines differenziellen Pfades ist nach Definition das Produkt der Potenziale der einzelnen Schritte. Eine **differenzielle Hülle** oder ein **Differenzial** [Lai/Massey/Murphy 1991] ist die Menge aller Pfade von einer gegebenen Input-Differenz der gesamten Chiffre zu einer gegebenen Output-Differenz. Es gilt eine analoge Faustregel, auf der die Methode der differenziellen Kryptoanalyse beruht:

> *Entlang eines differenziellen Pfades multiplizieren sich die differenziellen Potenziale (nach Definition). Das Potenzial einer differenziellen Hülle wird durch das Potenzial des dominanten differenziellen Pfades ausreichend approximiert.*

Dieses Potenzial wiederum ergibt die Wahrscheinlichkeit, mit der eine Gleichung für Schlüsselbits hergeleitet werden kann.

5.10 Charakteristiken und Differentiale

– Differenzielle Pfade und differenzielle Hüllen –

6 AES

Der AES („Advanced Encryption Standard“) ist der Nachfolger des obsoleten DES. Er wurde 2001 nach einem intensiven Auswahlprozess verabschiedet, aus dem der belgische Algorithmus Rijndael als Sieger hervorging. Näheres zu diesem Auswahlprozess auf den Webseiten zur Vorlesung.

Rijndael ist eine Mehrfach-Ciffre, basiert aber nicht auf dem FEISTEL-Schema. Seine Kernabbildung beruht auf einer S-Box, die im wesentlichen nichts anderes ist als die multiplikative Inversion in dem endlichen Körper $\mathbb{F}_{256}$, deren Nichtlinearitäts-Eigenschaften in dem mathematischen Abschnitt über Linearitätsmaße ausführlich behandelt wurden.

Eine sehr ausführliche, gründliche und verständliche Beschreibung des Verfahrens geben die Erfinder DAEMEN und RIJMEN selbst in ihrem Buch, siehe das Literaturverzeichnis zur Vorlesung.

6.1 Die Struktur von Rijndael

Repräsentation der Blöcke

Rijndael operiert auf Oktetten (8-Bit-Bytes), also auf dem $\mathbb{F}_2$-Vektorraum $\mathbb{F}_2^8$. Da an mehreren Stellen die Struktur dieses Vektorraums als Körper mit 256 Elementen ausgenützt wird, liegt es nahe, ihn von vorherein mit diesem Körper $\mathbb{F}_{256}$ zu identifizieren. Wie das genau gemacht wird, steht in Abschnitt 6.2.

Die Blöcke, mit denen Rijndael umgeht, sind zunächst Bit-Blöcke, deren Länge ein Vielfaches von 32 ist; spezifiziert ist der Algorithmus für die Block- und Schlüssellängen 128, 160, 192, 224, 256.

> Hier liegt der einzige Unterschied zwischen Rijndael und AES: AES ist nur für die Blocklänge 128 und die Schlüssellängen 128, 192 und 256 spezifiziert. Der Standardisierungsprozess macht daher keine Aussagen über die Sicherheit von Rijndael bei den übrigen Größen dieser Parameter.

Die 32 Bit werden jeweils als eine Spalte aus 4 Oktetten, also als ein Element des 4-dimensionalen $\mathbb{F}_{256}$-Vektorraums $\mathbb{F}_{256}^4$ aufgefasst. Ein Block der Länge $n = 32 \cdot N_b$ wird dann als N_b-Tupel solcher Spalten, also als $4 \times N_b$-Matrix

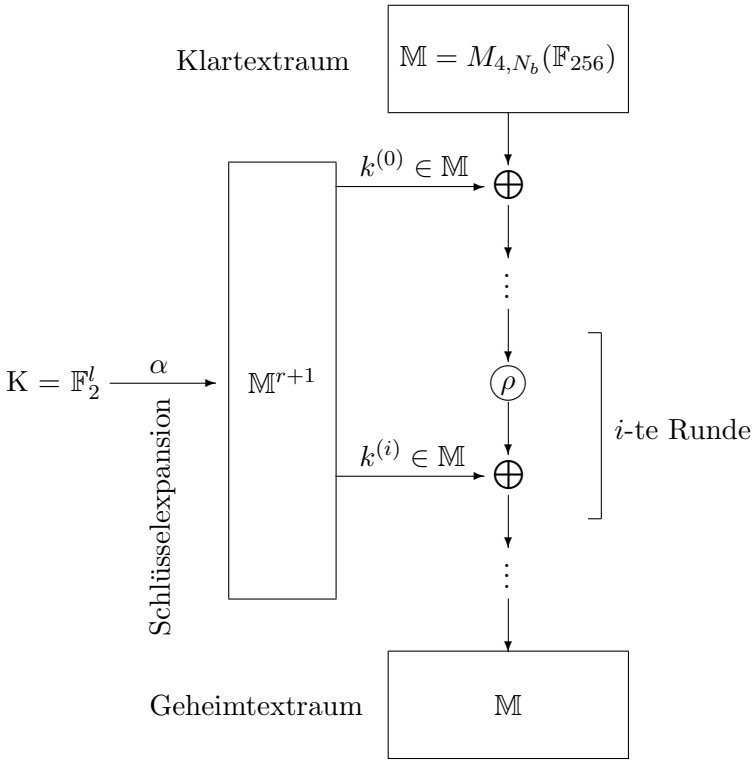
$$a = \begin{pmatrix} a_{0,0} & \dots & a_{0,N_b-1} \\ \vdots & & \vdots \\ a_{3,0} & \dots & a_{3,N_b-1} \end{pmatrix} \in M_{4,N_b}(\mathbb{F}_{256}) =: \mathbb{M}$$

interpretiert. Analog sei die Schlüssellänge $l = 32 \cdot N_k$; entsprechende Matrizen werden aber nicht verwendet, sondern Schlüssel werden erst expandiert und dann in Rundenschlüssel $k^{(i)} \in \mathbb{M}$ zerlegt.

> Nach der Spezifikation von Rijndael ist $N_b, N_k \in \{4, 5, 6, 7, 8\}$, nach der Spezifikation von AES $N_b = 4$, $N_k \in \{4, 6, 8\}$.

Das Iterationsschema

Der Verschlüsselungs-Algorithmus von Rijndael durchläuft r Runden, die jeweils aus der Kernabbildung ρ und der (binären) Addition eines Teilschlüssels (Rundenschlüssel) bestehen; in der letzten Runde ist die Kernabbildung etwas verkürzt – mehr dazu später. Vor Beginn der ersten Runde wird schon ein Teilschlüssel addiert; die Schlüsselexpansion muss also aus den gegebenen l Schlüsselbits $r + 1$ Teilschlüssel aus je $32N_b$ Bits produzieren. Die Kernabbildung ist invertierbar. Sie ist von Runde zu Runde (mit Ausnahme der letzten) gleich.



Die Zahl der Runden ist wie folgt festgelegt:

	N_b				
N_k	4	5	6	7	8
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14

Bei AES mit Schlüssellänge 128, 192 oder 256 ist die Rundenzahl also 10, 12 oder 14, das sind die grün markierten Tabelleneinträge.

Die Kernabbildung

Jede Runde von Rijndael besteht aus den vier Schritten

1. `SubBytes`, einer Substitution, die aus paralleler Anwendung der S-Box auf jedes Oktett besteht, siehe Abbildung 1,

2. `ShiftRows`, einer Permutation jeder Matrixzeile in sich, siehe Abbildung 3,

3. `MixColums`, einer linearen Abbildung (also komplizierter als nur Permutation) jeder Matrixspalte in sich, siehe Abbildung 4,

4. `AddRoundKey`, der Addition des Rundenschlüssels, siehe Abbildung 2.

Die Kernabbildung ρ umfasst also die ersten drei dieser Schritte. In der letzten Runde wird der dritte Schritt `MixColums` weggelassen; dadurch wird für die Umkehrabbildung, den Entschlüsselungsalgorithmus, ein strukturell gleicher Aufbau erreicht, und zur kryptographischen Stärke trägt dieser eine lineare Schritt am Ende ohnehin nichts mehr bei.

Die Abbildungen 1 bis 4 stammen aus der Wikipedia (für den Fall $N_b = N_k = 4$).

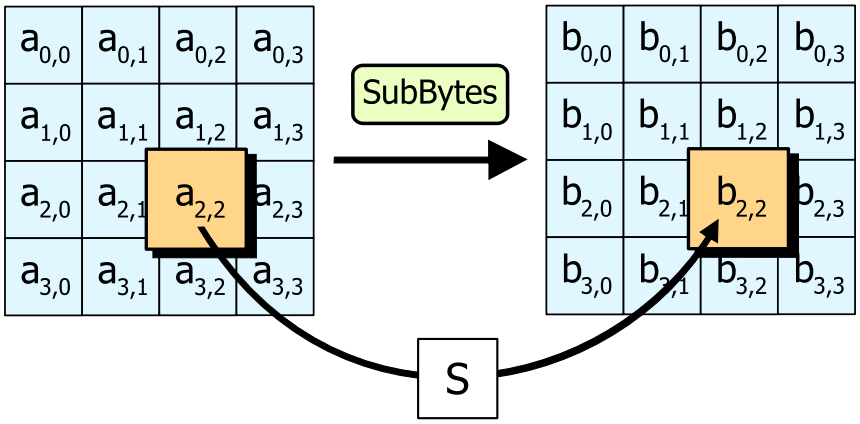


Abbildung 1: Die Wirkung von `SubBytes`

Der Substitutionsschritt `SubBytes`

Auf alle Oktette des aktuellen Zustands, also alle Einträge einer Matrix $b \in \mathbb{M}$, wird die S-Box S_{RD} einzeln angewendet. Diese ist Komposition $\mathrm{S}_{\mathrm{RD}} = f \circ g$ zweier Abbildungen, nämlich der Inversion $g = f_{-1}$ in $\mathbb{F}_{256}$,

$$g\colon \mathbb{F}_{256} \longrightarrow \mathbb{F}_{256}, \quad g(x) = \begin{cases} x^{-1} & \text{für } x \neq 0, \\ 0 & \text{für } x = 0, \end{cases}$$

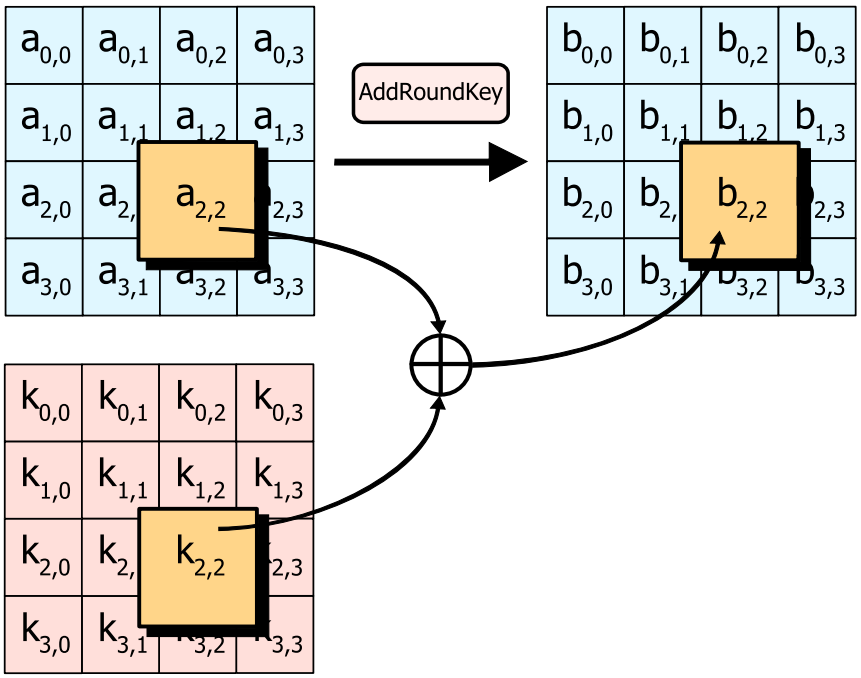


Abbildung 2: Die Wirkung von `AddRoundKey`

deren Nichtlinearitäts-Eigenschaften bestens bekannt sind, sowie der affinen Abbildung

$$f\colon \mathbb{F}_2^8 \longrightarrow \mathbb{F}_2^8, \quad \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix},$$

die so gewählt ist, dass

- S_{RD} keinen Fixpunkt hat, also $\mathrm{S}_{\mathrm{RD}}(b) \neq b$ für alle $b \in \mathbb{F}_{256}$,
- S_{RD} keinen „Anti-Fixpunkt“ hat, also $\mathrm{S}_{\mathrm{RD}}(b) \neq \bar{b}$ für alle $b \in \mathbb{F}_{256}$;

dabei ist $\bar{b} = (1 - b_7, \ldots, 1 - b_0)$ das BOOLEsche Komplement, also die bitweise logische Negation.

Nachteilig an dieser Abänderung der Inversionsabbildung ist, dass die involutorische Eigenschaft von f_{-1} verloren geht; für die Entschlüsselung muss daher eine andere S-Box $\mathrm{S}_{\mathrm{RD}}^{-1}$ implementiert werden.

Die Permutation der Zeilen `ShiftRows`

Jede der vier Zeilen des Zustands – der aktuellen $4 \times N_b$-Matrix – wird individuell zyklisch nach links geschoben, und zwar Zeile i um C_i Stellen. Die Werte der C_i hängen wie folgt von der Blocklänge ab:

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

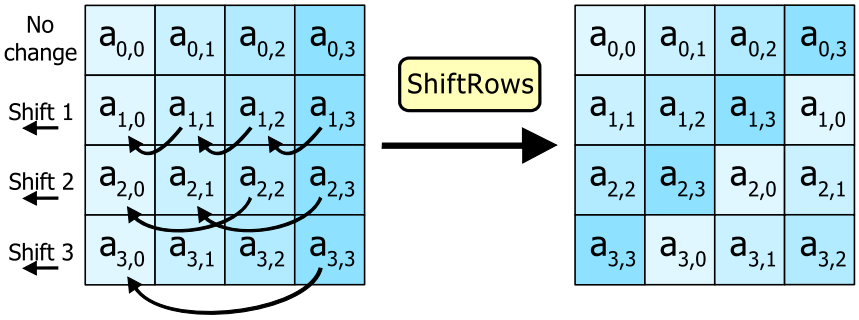


Abbildung 3: Die Wirkung von `ShiftRows`

Die lineare Transformation der Spalten `MixColumns`

Der Zustand wird Spalte für Spalte – d. h. die Zustandsmatrix $\in \mathbb{M}$ wird – von links mit einer festen 4×4-Matrix multipliziert; das Ziel dieses Schritts ist, erhebliche Diffusion zu erzeugen. Es handelt sich also um eine $\mathbb{F}_{256}$-lineare Abbildung

$$\mu \colon \mathbb{F}_{256}^4 \longrightarrow \mathbb{F}_{256}^4,$$

und zwar um die, die durch die Matrix

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

beschrieben wird; die Einträge sind als Oktette in Hexadezimal-Darstellung zu interpretieren, also z. B. $03 = (00000011) \in \mathbb{F}_2^8$.

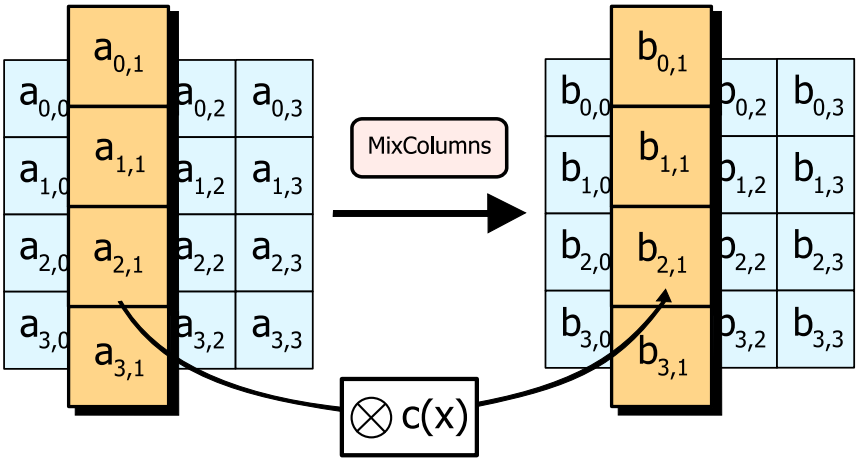


Abbildung 4: Die Wirkung von `MixColumns`

Die Entschlüsselung

Die gesamte Rijndael-Verschlüsselung ist die Komposition

$$F_k = \underbrace{\kappa_r \circ \tau \circ \sigma}_{i=r} \circ \ldots \circ \underbrace{[\kappa_i \circ \mu \circ \tau \circ \sigma]}_{i=1,\ldots,r-1} \circ \ldots \circ \kappa_0$$

von Abbildungen $\mathbb{M} \longrightarrow \mathbb{M}$; dabei ist κ_i die Addition des i-ten Rundenschlüssels, τ = `ShiftRows`, σ = `SubBytes` und μ = `MixColumns`. Die Kernabbildung ist also $\rho = \mu \circ \tau \circ \sigma$.

Für die Entschlüsselung brauchen wir die Umkehrabbildung, die zunächst so aussieht:

$$F_k^{-1} = \kappa_0^{-1} \circ \ldots \circ [\sigma^{-1} \circ \tau^{-1} \circ \mu^{-1} \circ \kappa_i^{-1}] \circ \ldots \circ \sigma^{-1} \circ \tau^{-1} \circ \kappa_r^{-1}.$$

Die versprochene Strukturgleichheit mit F_k ergibt sich aus einigen Umformungen:

- Es ist $\kappa_i^{-1} = \kappa_i$ für alle i.
- Da σ nur auf den Matrix-Einträgen wirkt, und zwar auf allen gleich, ist $\tau \circ \sigma = \sigma \circ \tau$.
- Schließlich ist $\kappa_i \circ \mu(x) = \mu(x) + k^{(i)} = \mu(x + \mu^{-1}(k^{(i)})) = \mu \circ \tilde{\kappa}_i(x)$, da μ linear ist. Also ist $\mu^{-1} \circ \kappa_i = \tilde{\kappa}_i \circ \mu^{-1}$, wobei $\tilde{\kappa}_i$ die binäre Addition von $\mu^{-1}(k^{(i)})$ ist; das wird für $i = 1, \ldots, r$ verwendet.

Damit ergibt sich

$$F_k^{-1} = \kappa_0 \circ \tau^{-1} \circ \sigma^{-1} \circ \ldots \circ [\tilde{\kappa}_i \circ \mu^{-1} \circ \tau^{-1} \circ \sigma^{-1}] \circ \ldots \circ \kappa_r.$$

Die Entschlüsselungsalgorithmus ist also genauso zusammengesetzt wie der Verschlüsselungsalgorithmus, nur dass

- das Schlüsselauswahlschema modifiziert ist: $\tilde{\kappa}_i$ statt κ_i und umgekehrte Reihenfolge der Teilschlüssel,
- `MixColumns` μ durch die lineare Umkehrabbildung μ^{-1} ersetzt wird,
- `Shiftrows` τ durch die Verschiebung nach rechts ersetzt wird,
- die S-Box S_{RD} durch die inverse Abbildung S_{RD}^{-1} ersetzt wird.

Die Schlüsselexpansion

Hier soll nur erwähnt werden, dass die Schlüsselexpansion zyklische Verschiebungen von Bytes innerhalb von Viererblöcken, die S-Box S_{RD}, sowie die Addition fester Konstanten verwendet.

6.2 Die Arithmetik des Grundkörpers

Für den AES-Algorithmus wird der 8-dimensionale $\mathbb{F}_2$-Vektorraum $\mathbb{F}_2^8$ mit dem Körper $\mathbb{F}_{256}$ gleichgesetzt. Dies bedarf einer Erläuterung.

Algebraische Darstellung des Grundkörpers

Endliche Körper werden über dem jeweiligen Primkörper $\mathbb{F}_p$ am einfachsten als Restklassenringe des Polynomrings $\mathbb{F}_p[X]$ nach einem Hauptideal konstruiert, das von einem irreduziblen Polynom $h \in \mathbb{F}_p[X]$ erzeugt wird. Dann ist $h\mathbb{F}_p[X]$ Primideal, also

$$K := \mathbb{F}_p[X]/h\mathbb{F}_p[X]$$

ein endlicher Körper, der über $\mathbb{F}_p$ den Grad (= die Dimension) $n = \operatorname{Grad} h$ hat. Für die Identifikation von K mit dem Vektorraum $\mathbb{F}_p^n$ werden die Restklassen der Potenzen von X mit den n Einheitsvektoren gleichgesetzt, also für $x = X \bmod h$:

$$x^0 = 1 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad x^1 = x = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \quad \ldots, \quad x^{n-1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

Ist $h = X^n + a_1X^{n-1} + \cdots + a_{n-1}X + a_n$ (o. B. d. A. normiert), so folgt aus $h \bmod h = 0$, dass

$$x^n = -a_1x^{n-1} - \cdots - a_{n-1}x - a_n$$

in K. Darüber hinaus zeigt diese Gleichung, wie die Restklasse eines jeden Polynoms f durch $1, x, \ldots, x^{n-1}$, also durch die kanonische Basis ausgedrückt werden kann. Algorithmisch läuft das auf den Rest der Polynom-Division „f geteilt durch h“ hinaus.

Für AES wird das Polynom

$$h = X^8 + X^4 + X^3 + X + 1 \in \mathbb{F}_2[X]$$

verwendet.

Multiplikationstafel

Die Multiplikationstafel für die Basis $(1, x, \ldots, x^{n-1})$ ergibt sich aus der durch h gegebenen Relation. Für AES ist etwa

$$x^2 \cdot x^7 = x^9 = x \cdot x^8 = x \cdot (x^4 + x^3 + x + 1) = x^5 + x^4 + x^2 + x.$$

Effiziente Inversion

Bei AES wird eine vollständige Wertetabelle für S-Box verwendet; das ist effizient implementierbar, da es sich ja nur um 256 Werte handelt.