Kryptographie I
WS 2005/2006

G. Leander

T. Brecher, M. Goldack, O. Grieb,

S. Hoerder, S. Spitz, C. Wachsmann

21. Februar 2006

Inhaltsverzeichnis

Kapitel 1

Block Ciphers

1.1 DES - Data Encryption Standard

Der Data Encryption Standard war bis zu der Einführung seines Nachfolgers AES (siehe Abschnitt 1.2) - und ist es in weiten Teilen noch immer - der wichtigste symmetrische Algorithmus überhaupt. Nicht zuletzt aus diesem Grunde ist es sinnvoll bei einer weiterführenden Kryptographie Vorlesung mit dem DES zu beginnen. Trotzdem gibt es, zumindest in diesem Kurs, kaum ein undankbareres Kapitel. Das Thema ist nicht nur für die Studenten schwer nachzuvollziehen, auch für den Dozenten ist es eine große Herausforderung den DES sinnvoll darzustellen. Dieser Kurs soll deshalb nicht nur die Funktionsweise des Data Encryption Standard beschreiben, wie es in zahllosen Büchern und Skripten schon geschehen ist, sondern auch die naheliegendste, aber doch oft ignorierte Frage beantworten, die sich bei einer ersten Beschreibung mit DES stellt: „Warum sollen gerade diese Tabellen so gut sein?“

1.1.1 Bezeichnung

Der DES („Data Encryption Standard“) ist ein Verschlüsselungsalgorithmus, der von dem NIST (bis 1988 NBS - „National Bureau of Standards“, danach „National Institute of Standards and Technology“) standardisiert wurde. ([FIPS 46-3]) Das ANSI („American National Standards Institute“) bezeichnet den Algorithmus als DEA („Data Encryption Algorithm“) und die ISO („International Organization for Standardization“) bezeichnet ihn als DEA-1 („Data Encryption Algorithm 1“).

1.1.2 Geschichte des DES

um 1970: Die gesamte Forschung wurde fast ausschließlich von militärischen Organisationen und Geheimdiensten betrieben. In der öffentlichen Forschung war Kryptographie kein Thema. Die NSA („National Security Agency“) wurde von offizieller Seite damals noch dementiert und bekam den Spitznamen „No Such Agency“. Es gab zwar ein paar kommerzielle Verschlüsselungsprodukte, die aber alle Black-Box-Systeme waren und somit nicht analysiert werden konnten.

1972: Das NBS beschließt, einen öffentlichen Verschlüsselungsstandard zu entwerfen.

1973: Das NBS führt erste öffentliche Befragungen durch. Einige der Anforderungen des NBS an den neuen Standard waren:

- eine höhere Sicherheit als die bekannten Systeme
- vollständig beschrieben und einfach zu verstehen
- öffentlich bekannt
- Flexibilität, also in vielen Anwendungen verwendbar
- hohe Effizienz
- leicht verifizierbar und zertifizierbar
- exportierbar

1974: IBM schlägt seinen Verschlüsselungsalgorithmus „Lucifer“ (Blockchiffre, 128 Bit Block- und Schlüssellänge, Feistelchiffre, von Horst Feistel und anderen entwickelt) als Kandidat für den DES vor. Lucifer wird zwar von der NSA, die die US-Regierung in dem Verfahren vertritt, abgelehnt, allerdings entwickeln IBM und das NBS auf Basis des Lucifer-Algorithmus den DES.

1975: Der DES wird erstmals veröffentlicht. Viele Forscher befürchten, dass der DES Hintertüren enthält, die entweder von IBM oder von der NSA dort eingebaut wurden. Weiterhin hat die NSA durchgesetzt, dass die Schlüssellänge auf 56 Bit reduziert wird und hat die S-Boxen modifiziert, ohne die genauen Designkriterien zu veröffentlichen.

1976: Am 23. November 1976 wird der DES als US-Standard eingeführt. Alle 5 Jahre musste die Akkreditierung des DES erneuert werden.

Seitdem: In den Jahren 1982, 1987 und 1993 wurde die Akkreditierung verlängert, da keine wirkungsvollen Attacken gefunden wurden. In den Jahren 1998/1999 konnte die Electronic Frontier Foundation (EFF) mit spezieller Hardware („Deep Crack“) erste Brute-Force-Angriffe in vertretbarer Zeit (56 Stunden 1998 bzw. 22 Stunden 1999) durchführen.

Die beiden wichtigsten kryptanalytischen Angriffe, die „differentielle Kryptanalyse“ (Eli Biham, Adi Shamir) und die „lineare Kryptanalyse“ (Mitsuru Matsui) haben die Sicherheit des DES nicht wesentlich beeinflusst und auch heute gilt der DES noch als sehr sicher, wäre nur der Schlüssel nicht zu kurz.

1.1.3 Beschreibung des DES

Für die Sicherheit des DES gibt es zwei grundlegende Designkriterien:

- **Konfusion:** Es dürfen keine statistischen Auffälligkeiten zwischen dem Klartext und dem Chiffrat erkennbar sein.
- **Diffusion:**
  - Wenn sich am Eingang 1 Bit ändert, soll sich jedes Ausgangsbit mit Wahrscheinlichkeit 1/2 ändern. Dies wird auch als „Avalanche-Effekt“ bezeichnet.
  - DES muss nichtlinear sein.

Zu diesen generellen Kriterien kommen noch weitere hinzu, die insbesondere Schutz vor der differentiellen und linearen Attacke bieten und auf die weiter unten eingegangen wird.

Um diese Eigenschaften zu erreichen, baut der DES auf drei Strukturen auf:

- Blockchiffre
- iterative Chiffre
- Feistelchiffre

Blockchiffren

Als Blockchiffre verschlüsselt der DES nicht einzelne Bits sondern immer ganze Eingabeblöcke. Die Blocklänge beträgt beim DES 64 Bit und die effektive

Schlüssellänge 56 Bit.

$$\begin{aligned} \mathrm{DES} : \mathbb{F}_2^{64} \times \mathbb{F}_2^{56} &\rightarrow \mathbb{F}_2^{64} \\ \mathrm{DES}(m, K) &= c \end{aligned}$$

Iterative Chiffre

Der DES verwendet eine Funktion $F_{k_i} : \mathbb{F}_2^{64} \rightarrow \mathbb{F}_2^{64}$ 16 mal (16 Runden), um die Daten zu verschlüsseln. Das Chiffrat c ergibt sich aus der Nachricht n wie folgt:

$$c = F_{k_{16}}\left(F_{k_{15}}\left(F_{k_{14}}\left(\ldots F_{k_1}\left(n\right)\ldots\right)\right)\right) = F_k(n)$$

Für jede der 16 Runden wird ein eigener Teilsschlüssel k_i verwendet. Die Teilschlüssel werden von dem Schlüssel K abgeleitet.

Feistelchiffre

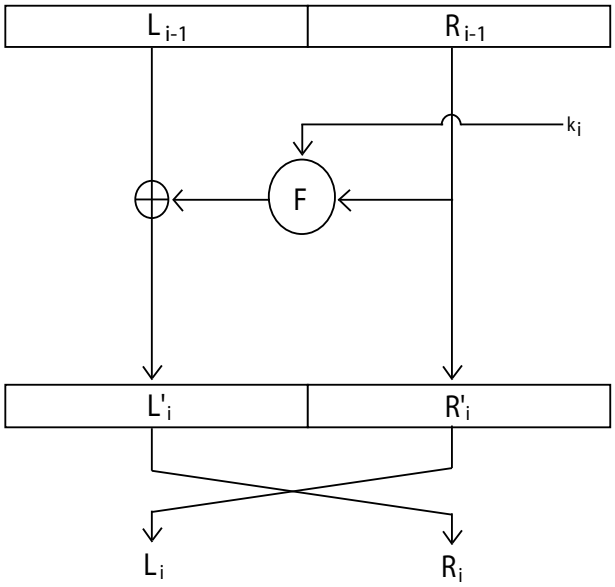

Abbildung 1.1: Eine Feistelrunde

Eine Feistelchiffre (auch „Feistelnetzwerk“) lässt sich allgemein als Funktion $F_k = F(n, k)$ beschreiben, die die Eingabenachricht in einen linken Teil L und einen rechten Teil R gleicher Länge zerlegt. k ist ein Teilschlüssel von K, also $k \subseteq K$. Dann gilt für die Runde i:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= F_{k_i}(R_{i-1}) \oplus L_{i-1} \end{aligned}$$

Für eine Feistelchiffre (mit N Runden) ist also eine Schlüsselableitungsfunktion $K \to k_1, k_2, \ldots, k_N$ notwendig.

1.1.4 Die Rundenfunktion

Die Verschlüsselung des DES beginnt mit der „Initial Permutation“ (IP), danach kommen die 16 Durchläufe der Rundenfunktion. Auf die Ausgabe der 16. Runde wird die inverse Permutation IP^{-1} angewendet, bevor das Chiffrat ausgegeben wird. Die Initial Permutation wurde so gewählt, dass eine schnelle Implementation möglich ist.

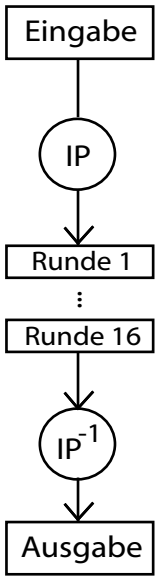

Abbildung 1.2: Die Rundenfunktion im DES

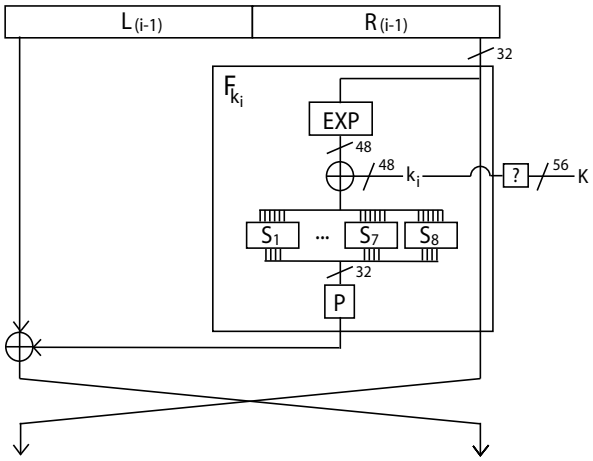

Abbildung 1.3: Die Rundenfunktion

Die Rundenfunktion selbst ist als Feistelnetzwerk organisiert und besteht aus einer Expansionsfunktion (Exp-Box), die die 32-Bit Eingabe auf 48 Ausgabebit abbildet. Das Ergebnis der Exp-Box wird mit dem Rundenschlüssel XOR-verknüpft und in sechs Bit Blöcken als Eingabe für die 8 S-Boxen verwendet. Die Ausgabe der S-Boxen ist nur noch 32 Bit lang und wird erneut permutiert („P-Box“), bevor sie mit der linken Hälfte der Nachricht XOR-verknüpft wird.

1.1.5 Die Expansionsabbildung Exp

Die Exp-Box verlängert die Eingabe, in dem einige Eingabebits redundant wiederholt werden. Die Grafik verdeutlicht dies.

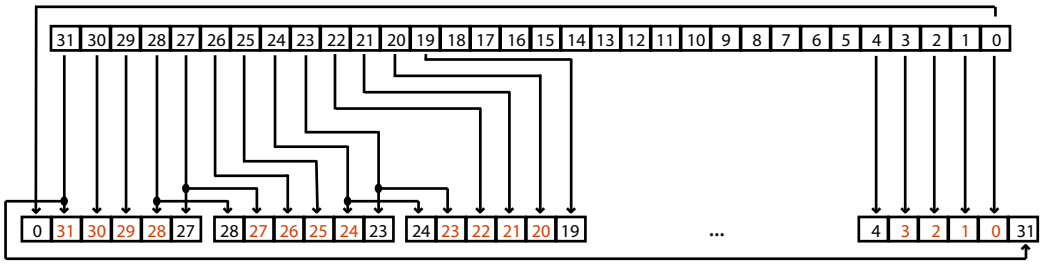

Abbildung 1.4: Die Exp-Box

Insbesondere ist die Exp-Box daher eine *lineare* Abbildung

$$Exp : \mathbb{F}_2^{32} \rightarrow \mathbb{F}_2^{48}$$

Wenn wir die Eingaben für eine S-Box S_i mit (a, b, c, d, e, f) bezeichnen dann ist die Eingabe für die benachbarte S-Box S_{i+1} von der Form (e, f, g, h, i, j). Solche und ähnliche Überlegungen, zusammen mit den Design Kriterien der S-Boxen werden später eine Rolle spielen, wenn die Resistenz von DES gegen die differentielle Attacke untersucht wird.

1.1.6 Die S-Boxen

Die S-Boxen sind Abbildungen

$$S : \mathbb{F}_2^6 \rightarrow \mathbb{F}_2^4$$

es werden also 6 Bits auf 4 Bits abgebildet. Eine Besonderheit der DES S-Boxen ist, dass jede S-Box aus 4 Permutationen $S_{i,j} : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ mit $i, j \in \mathbb{F}_2$ zusammengesetzt wurde. Das erste und sechste Bit der Eingabe wählen eine der 4 Permutationen aus.

$$S : \mathbb{F}_2 \times \mathbb{F}_2^4 \times \mathbb{F}_2 \rightarrow \mathbb{F}_2$$

Also:

$$S(e_0, (e_1, e_2, e_3, e_4), e_5) = S_{e_0,e_5}(e_1, e_2, e_3, e_4).$$

Für 8 S-Boxen sind also insgesamt 32 Permutationen notwendig. Die ursprünglich von IBM und dem NBS vorgeschlagenen S-Boxen wurden von der NSA kommentarlos abgelehnt und durch die heute aktuellen S-Boxen ersetzt.

Auch deshalb waren die S-Boxen lange Zeit das große Mysterium des DES. Die Tabellen scheinen völlig willkürlich gewählt und erst durch die Veröffentlichung der Design-Kriterien durch Coppersmith [Coppersmith 94] kam ein

wenig Licht in das Dunkel. Die Design-Kriterien waren bis zu der Entwicklung der differentiellen Attacke durch Shamir und Biham geheimgehalten worden. Wir geben die Design Kriterien hier an, eine genauere Analyse der Kriterien und des Schutzes, den sie gegen die Differentielle Attacke bieten, erläutern wir später.

S-1: Jede S-Box bildet sechs Eingabebits auf vier Ausgabebits ab.

S-2: Keines der Ausgabebits darf zu dicht an einer linearen Funktion über die Eingabebits liegen. (Genaueres hierzu später.)

S-3: Die S-Boxen müssen aus 4 Permutationen zusammengesetzt sein, die von den beiden Randbits e_0 und e_5 ausgewählt werden. (Siehe oben.)

S-4: Wenn sich zwei Eingaben für eine S-Box in genau einem Bit unterscheiden, müssen sich die Ausgaben in mindestens zwei Bit unterscheiden.

S-5: Wenn sich zwei Eingaben für eine S-Box genau in den beiden mittleren Bits unterscheiden, müssen sich die Ausgaben in mindestens 2 Bit unterscheiden.

S-6: Wenn sich zwei Eingaben für eine S-Box in den ersten beiden Bits unterscheiden und in den letzten beiden Bits übereinstimmen, müssen sich die Ausgaben unterscheiden.

S-7: Für jede feste Ausgabedifferenz und jeder festen, von Null verschiedenen Eingabedifferenz dürfen höchstens 8 der möglichen 32 Eingabe-Paare mit der gegebenen Eingabedifferenz die gegebene Ausgabedifferenz erzeugen. Für jede S-Box darf die Gleichung

$$S_i(x) + S_i(x + \Delta) = \Delta'$$

also für jedes $\Delta \in \mathbb{F}_2^6$ und jedes $\Delta' \in \mathbb{F}_2^4$ höchstens 8 Lösungen x haben.

S-8: Ist eine Verschärfung von S-7 für den Fall von 3 benachbarten S-Boxen. (Genaueres findet sich weiter unten.)

1.1.7 Die Permutation P

Die P-Box ist in allen Runden gleich. Die 32 Ausgabebits der 8 S-Boxen werden durch die P-Box permutiert. Insbesondere ist damit die P-Box eine lineare Abbildung

$$P : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}\,.$$

Auch für die P-Box hat Coppersmith in [Coppersmith 94] Designkriterien veröffentlicht, die insbesondere für den Schutz gegen die differentielle Attacke dienen.

P-1: Die vier Ausgabebits einer S-Box der Runde i werden so verteilt, dass zwei von ihnen die mittleren Bits e_2 und e_3 von S-Boxen der Runde $i+1$ beeinflussen. Die beiden anderen Ausgabebits beeinflussen die „Endbits“ e_0, e_2, e_4 und e_5 von S-Boxen der Runde $i+1$. (Eine S-Box teilt sich ihre Endbits mit den beiden benachbarten S-Boxen.)

P-2: Die vier Ausgabebits jeder S-Box einer Runde beeinflussen genau sechs S-Boxen der nächsten Runde. D.h. dass zwei Ausgabebits einer S-Box nicht die selbe S-Box der nächsten Runde beeinflussen können.

P-3: $\forall j,k\ :\ 1 \leq j,k, \leq 8$ gilt: Wenn ein Ausgabebit einer S-Box S_j in Runde i eines der mittleren Eingabebits der S-Box S_k in Runde $i+1$ beeinflusst, dann darf keines der Ausgabebits der S-Box S_k in Runde $i+1$ die mittleren Eingabebits der S-Box S_j in Runde $i+2$ beeinflussen.

1.1.8 Die Schlüsselableitung

Der Schlüssel K ist bei dem DES 64 Bit lang und teilt sich in 56 zufällige Bits sowie 8 Paritätsbits auf. Wenn man K in acht 8-Bit Blöcke aufteilt, ist das Least-Significant-Bit (LSB) jedes Blockes das Paritätsbit (ungerade Parität) für diesen Block. Zu Beginn werden die Paritätsbit mit Hilfe der Permutation „Permuted Choice 1“ (PC-1) aus dem Schlüssel entfernt.

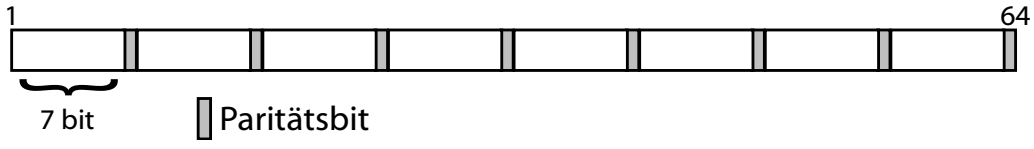

Abbildung 1.5: Die Struktur des DES-Schlüssels

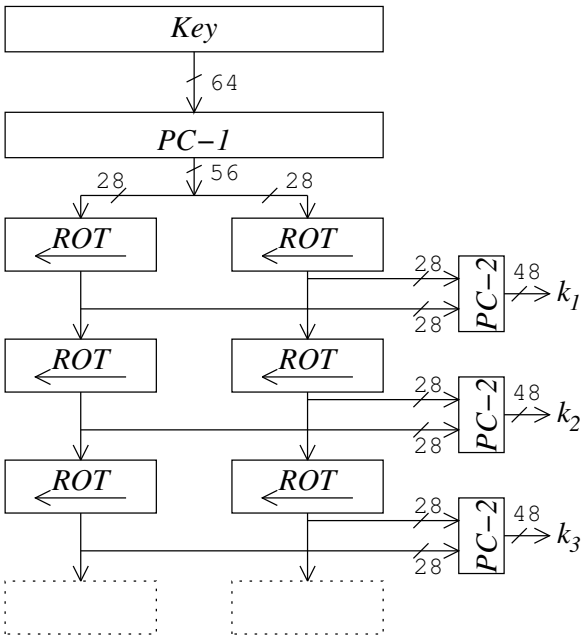

Abbildung 1.6: Die Schlüsselableitung

Der Schlüssel K wird zuerst in der PC-1-Box auf 56 Bit gekürzt. Danach wird der Schlüssel in zwei Hälften geteilt und beide Hälften werden nach links rotiert. Die rotierten Hälften bilden nach einer weiteren Permutation PC-2 den Sitzungsschlüssel. Die PC-2 bildet die 56 Eingabebits auf die benötigten 48 Ausgabebits ab. Für jeden weiteren Sitzungsschlüssel k_i werden die rotierten Hälften des Sitzungsschlüssels k_{i-1} weiter nach links rotiert und durch die PC-2-Box permutiert. Die Zahl der Verschiebungen nach links ist abhängig von der Runde:

- $\forall k_i$, $i \in \{1, 2, 9, 16\}$: Verschiebung um 1 Stelle
- In allen anderen Runden: Verschiebung um 2 Stellen

1.1.9 Entschlüsselung

Ein Feistelnetzwerk ermöglicht eine sehr einfache Umkehrung der Verschlüsselung (also die Entschlüsselung): Die XOR-Operation lässt sich leicht umkehren, mit $L_i = R_{i-1}$ ist ein Teil der Eingabe bekannt und die Funktion F_{k_i} muss nicht einmal injektiv sein, um die Umkehrung zu ermöglichen. Wenn man über die richtigen Rundenschlüssel verfügt, kann man prinzipiell die selben Schaltkreise für die Ver- und die Entschlüsselung verwenden.

Es reicht ein Bit, um zwischen Verschlüsselung und Entschlüsselung zu unterscheiden. Beide Vorgänge können mit dieser Struktur realisiert werden: Für die Entschlüsselung müssen lediglich die Rundenschlüssel in der umgekehrten Reihenfolge verwendet werden und die linke und die rechte Hälfte der Eingabe müssen vertauscht werden. Es gilt also für die Zahl der Verschiebungen:

- Für k_1: Keine Verschiebung

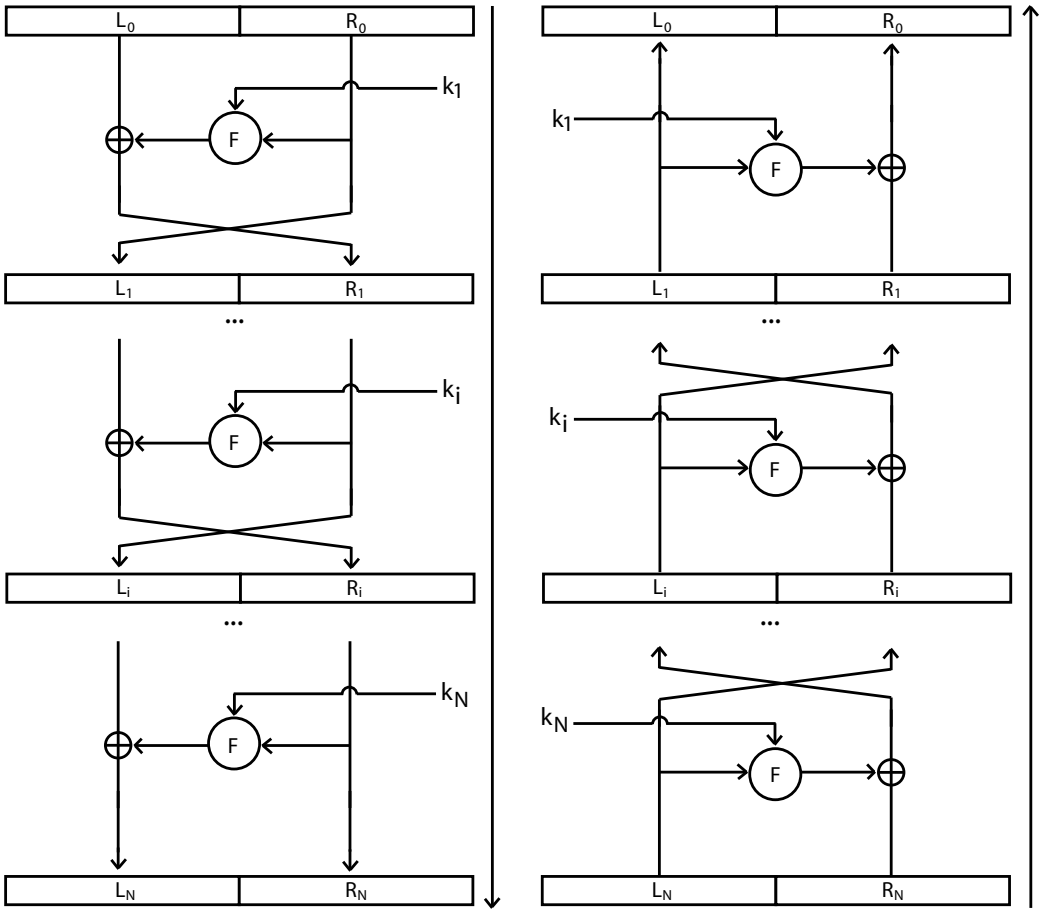

Abbildung 1.7: N Runden einer Feistelchiffre, Ver- und Entschlüsselung

- $\forall k_i$, $i \in \{2, 9, 16\}$: Verschiebung um 1 Stelle
- In allen anderen Runden: Verschiebung um 2 Stellen

1.1.10 Schwache Schlüssel und komplementär Eigenschaft

Es ist bekannt, dass schwache Schlüssel existieren. Dies sind insbesondere Schlüssel, in denen sich Bitketten wiederholen. Da die beiden Hälften des Schlüssels unabhängig voneinander rotieren, wird in jeder Runde der selbe Rundenschlüssel verwendet, wenn in jeder rotierenden Hälfte alle Bits 0 oder 1 sind. Die folgende Liste (siehe [Schneier 96])zeigt alle schwachen Schlüssel vor und nach der PC-1-Box in Hexadezimalnotation:

`0101 0101 0101 0101` $\longrightarrow$ `0000000 0000000`

`1F1F 1F1F 0E0E 0E0E` $\longrightarrow$ `0000000 FFFFFFF`
`E0E0 E0E0 F1F1 F1F1` $\longrightarrow$ `FFFFFFF 0000000`
`FEFE FEFE FEFE FEFE` $\longrightarrow$ `FFFFFFF FFFFFFF`

Dies Schlüssel erzeugen in jeder Runde die selben Rundenschlüssel.

Als „halbschwache Schlüssel“ (im englischen „semiweak keys“) werden Schlüssel bezeichnet, für die es einen anderen Schlüssel gibt, der aus dem selben Klartext das selbe Chiffrat erzeugen. K_1, K_2 sind also halbschwach, wenn $E_{K_1}(m) = E_{K_2}(m) = c$ gilt. Dann gilt auch, dass ein Chiffrat, das mit K_1 erzeugt wurde, mit K_2 entschlüsselt werden kann. (Also: $m = D_{K_2}\left(E_{K_1}\left(m\right)\right)$) Insgesamt gibt es 6 Paare (K_1, K_2), also 12 halbschwache Schlüssel.

Als „möglicherweise schwache Schlüssel“ (im englische „possibly weak keys“) werden Schlüssel bezeichnet, die nur 4 unterschiedliche Rundenschlüssel produzieren. Diese Rundenschlüssel werden jeweils 4 mal verwendet. Insgesamt gibt es 48 möglicherweise schwache Schlüssel. Vollständige Listen der halb schwachen und möglicherweise schwachen Schlüssel sind in [Schneier 96] abgedruckt.

Eine weitere interessante Eigenschaft des DES ist das Problem komplementärer Schlüssel:

Lemma 1 *Für einen Schlüssel k, eine Nachricht n und einen Chiffrat c gilt:*

$$c = \mathrm{DES}_K(m) \Leftrightarrow \overline{c} = \mathrm{DES}_{\overline{K}}(\overline{m}),$$

wobei mit $\overline{x}$ immer das bitweise Komplement von x bezeichnet wird.

Beweis: Bis auf die S-Boxen verhalten sich alle Operationen im DES linear. Wir bezeichnen mit $\overrightarrow{1}_n$ den Vektor mit einer 1 in jeder Komponente in $\mathbb{F}_2^n$.

Für die Eingabe der S-Boxen gilt also:

$$\begin{aligned} Exp\left(\overline{R_{i-1}}\right) + \overline{K_i} &= Exp\left(R_{i-1} + \overrightarrow{1}_{32}\right) + \overline{K_i} \\ &= Exp\left(R_{i-1}\right) + Exp\left(\overrightarrow{1}_{32}\right) + K_i + \overrightarrow{1}_{48} \\ &= Exp\left(R_{i-1}\right) + \overrightarrow{1}_{48} + K_i + \overrightarrow{1}_{48} \\ &= Exp\left(R_{i-1}\right) + K_i \end{aligned}$$

Die Ausgabe der S-Boxen ist also für die Eingabe $(\overline{K}, \overline{m})$ die selbe wie für die Eingabe (K, m). Damit gilt dann:

$$\overline{R_i} = F_{k_i}(R_{i-1}) \oplus \overline{L_{i-1}} = F_{\overline{k_i}}(\overline{R_{i-1}}) \oplus \overline{L_{i-1}}$$

Die Komplementäreigenschaft bleibt also auch in dem Feistelnetzwerk erhalten. □

Dadurch verringert sich der Schlüsselraum von 2^{56} auf 2^{55}.

1.1.11 Mehrfach-DES

Biham und Shamir haben gezeigt, dass es für DES mit weniger als 16 Runden Angriffe gibt, deren Komplexität unter der Komplexität eines Brute-Force-Angriffes liegt. Der Beweis versagt jedoch für DES mit 16 oder mehr Runden.

Im Jahr 1998 gelang es, die „DES Challenge II-2“ mit spezieller Hardware („Deep Crack“) in 56 Stunden zu knacken, im Jahr 1999 konnte die „DES Challenge III“ von der selben Hardware mit Hilfe eines „distributed computing“-Netzwerkes in 22 Stunden gebrochen werden.

Als Reaktion auf die Angriffe kann man versuchen, durch mehrfache DES-Verschlüsselung die Schlüssellänge und damit die Sicherheit zu erhöhen. Die erste Idee ist „Double-DES“ (Schlüssel: $K = (K_1, K_2)$). Hier sind die Ver- und Entschlüsselung folgendermaßen definiert:

$$c = E_{K_2}\left(E_{K_1}(n)\right)$$
$$n = E_{K_1}^{-1}\left(E_{K_2}^{-1}(c)\right)$$

Der Double-DES kann jedoch durch eine „Meet-In-The-Middle-Attacke“ angegriffen werden. Für ein gegebenes Nachrichten-Chiffrat-Paar (m, c) wird hierbei wie folgt verfahren:

Zuerst wird für alle möglichen Schlüssel K_1 eine Tabelle der entsprechenden Chiffrate erstellt.

$$\forall K_1 : \mathbb{C}_{Middle} = [E_{K_1}(m)]$$

Anschließend wird für jeden möglichen Schlüssel K_2 der Wert $E_{K_2}^{-1}(c)$ berechnet. Für jeden dieser Werte wird geprüft, ob der Wert einem Eintrag in der erstellten Tabelle entspricht. Wenn nicht, kann K_2 nicht der richtige (Teil)-Schlüssel sein, wenn ja ist K_2 ein Kandidat für den Teil-Schlüssel. Wenn mehr als nur ein Schlüssel in Frage kommen, muss man die möglichen Schlüssel mit weiteren Paaren (m, c) testen. Diese Attacke reduziert die Anzahl der notwendigen Berechnungen in etwa auf das Niveau von normalem DES, benötigt jedoch sehr viel Speicher.

Der Schlüsselraum des Double-DES ist also nur geringfügig größer als der Schlüsselraum des normalen DES. Es sind aber

$$\underbrace{8}_{\text{Blocklänge}} * \underbrace{2}_{\text{Schlüssel und } \mathbb{C}} * \underbrace{2^{56}}_{\textit{Möglichkeiten}} \text{ Bytes} = 2^{60}\text{Bytes} = 1E\text{Byte}$$

notwendig. Inzwischen gibt es bereits Tape Libraries, die ca. $100P\text{Bytes} = \frac{1}{10}E\text{Bytes}$ speichern können. Die Meet-In-The-Middle-Attacke ist in [Schneier 96] ausführlich beschrieben.

Die nächste Möglichkeit, „Triple-DES“ (3-DES) verwendet den Schlüssel $K = (K_1, K_2, K_3)$. Die Ver- und Entschlüsselung sind definiert als:

$$c = E_{K_3}\left(D_{K_2}\left(E_{K_1}\left(m\right)\right)\right)$$
$$m = D_{K_1}\left(E_{K_2}\left(D_{K_3}\left(c\right)\right)\right)$$

Die alternierende Folge von Ver- und Entschlüsselung erleichtert die Implementierung des 3-DES. Eine Meet-In-The-Middle-Attacke ist prinzipiell möglich, allerdings müssen dabei immer eine Ver- und eine Entschlüsselung zusammen als Black-Box betrachtet werden:

$$c = \underline{E_{K_3}}\left(\underline{D_{K_2}}\left(E_{K_1}\left(n\right)\right)\right)$$
$$n = D_{K_1}\left(\underline{E_{K_2}}\left(\underline{D_{K_3}}\left(c\right)\right)\right)$$

Dadurch umfasst der Schlüsselraum noch immer 2^{112} Schlüssel. Die Zeitkomplexität hat sich im Vergleich zu dem normalen DES (2^{56}) also quadriert. Die Sicherheit des 3-DES verändert sich auch dann nicht, wenn $K_3 = K_1$ gewählt wird. (Da der DES keine Gruppe bildet, gibt es kein K_4 so dass $E_{K_3}\left(E_{K_2}\left(E_{K_1}\left(n\right)\right)\right) = E_{K_4}\left(n\right)$ gilt. Dies ist etwas ausführlicher in [Schneier 96] beschrieben.)

Für den 4-DES kann leicht gezeigt werden, dass die Zeitkomplexität der des 3-DES entspricht. Erst beim 5-DES steigt die Zeitkomplexität (2^{165}) wieder deutlich an.

1.1.12 Literatur

Eine relativ ausführliche Beschreibung und Diskussion des DES bietet [Schneier 96] (auf Englisch). Eine etwas knappere deutschsprachige Beschreibung bietet [Beutelspacher 05]

1.2 AES - Advanced Encryption Standard

1.2.1 Weshalb wurde der AES entwickelt

Ende der 90er Jahre stand fest, dass die Schlüssellänge von DES zu kurz war und nicht mehr lange als sicher gelten würde. Bereits heute sind Brute-Force Angriffe auf DES mit spezieller Hardware (DES-Cracker) relativ günstig zu realisieren. Daher wird statt DES mittlerweile oft 3-DES verwendet, welcher eine Schlüssellänge von 112 Bit, anstelle der 56 Bit von DES, besitzt.

Softwareimplementierungen des DES sind vergleichsweise langsam. Daher ist der Einsatz von 3-DES (dreifache Hintereinanderausführung des DES) in manchen Umgebungen nur bedingt anwendbar.

NIST (US Standardisierungsbehörde) Die *National Institution of Standards and Technology* (NIST) hat am 2. Januar 1997 eine Initiative ins Leben gerufen um einen Nachfolger für den DES, den Advanced Encryption Standard (AES), zu finden. Am 12. September 1997 wurde eine entsprechende Ausschreibung publiziert.

Die Anforderungen an den neuen Verschlüsselungsstandard AES verlangten die Entwicklung einer Blockchiffre mit einer Blockgröße von 128 Bit und die Unterstützung der Schlüssellängen von 128, 192 und 256 Bit.

Am 20. August 1998 standen 15 Kandidaten fest, die in der ersten von drei Runden publiziert wurden. Die zweite Runde fand im März 1999 statt, bei der fünf Verfahren (MARS, RC6, Serpent, Twofish und Rijndael) in die engere Auswahl kamen. Am 2. Oktober 2000 wurde nach eingehender Überprüfung dieser fünf Kandidaten Rijndael als neuer Verschlüsselungsstandard gewählt.

1.2.2 Rahmenbedingungen für den AES

Sicherheit

- Robust gegen Kryptanalyse
- Mathematisch beweisbar gegen bekannte Attacken
- Zufälligkeit der Chiffratausgabe

Kosten

- Keinerlei Patentansprüche
- Effiziente Berechnung auf verschiedenen Rechnerarchitekturen
- Geringe Speicher- und Rechenanforderungen für den Einsatz in mobilen Geräten

Implementierung

- Für verschiedenste Hard- und Softwareumgebungen geeignet
- Einfaches Verfahren, folglich einfach und unkompliziert zu implementieren
- Flexible Schlüssellänge (128, 192, 256 Bit)
- Blockgröße von mindestens 128 Bit

1.2.3 Rijndael

Gemäß den von der NIST gestellten Anforderungen kann der AES mit Schlüssellängen von 128, 192 und 256 Bit eingesetzt werden, um 128 Bit lange Nachrichten zu verschlüsseln. Um die Beschreibung von AES zu vereinfachen, beziehen sich alle weiteren Ausführungen auf AES-128 d.h., AES mit 128 Bit langen Schlüsseln.

Statt eines Feistel-Netzes, wie es beispielsweise der DES verwendet, kommt beim AES ein S-P-Netz (Substitutions-Permutations-Netz) zum Einsatz.

S-P-Netze

Abbildung 1.8 zeigt die Struktur eines S-P-Netzes.

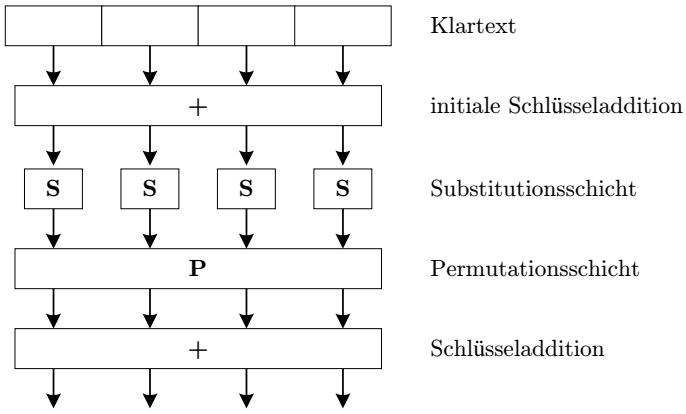

Abbildung 1.8: S-P-Netz

Diese Struktur wird über mehrere Runden wiederholt. Vor der ersten Runde muss jedoch der erste Rundenschlüssel zum Klartext addiert werden (XOR), da die Permutation bzw. Substitution in der ersten Runde schlüsselunabhängig und öffentlich sind. Diese Transformationen sind somit ohne Kenntnis des Schlüssels berechenbar und daher in der ersten Runde nicht sicherheitsrelevant.

1.2.4 Funktionsweise des AES

Man beschreibt die Zustände des AES-128 durch eine 4×4 Matrix:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \quad \text{mit } a_{i,j} \in GF(2^8)$$

Jede Position der Matrix entspricht also einem Byte.

Einschub Endliche Körper

Galoisfeld

$$GF(2^8) = GF(2)[x]/(f)$$

Beschreibt alle Polynome in $GF(2)$ mod f wobei f ist ein irreduzibles Polynom aus $GF(2)[x]$ vom Grad 8 ist. Für AES ist diese Polynom in der Beschreibung spezifiziert: $f(x) = x^8 + x^4 + x^3 + x + 1$

Ein irreduzibles Polynom lässt sich, ähnlich einer Primzahl, nicht in Polynome kleineren Grades zerlegen. Diese Eigenschaft garantiert, das jedes Element in $GF(2)[x]/(f)$ ein multiplikatives Inverses hat.

Elemente $a \in GF(2^8)$ haben eine eindeutige Darstellung

$$a = a_7x^7 + a_6x^6 + \ldots + a_0x^0 \quad \text{mit } a_i \in GF(2) = \mathbb{Z}_2 = \{0, 1\}$$

Addition im $GF(2^8)$ Seien $a, b \in GF(2^8)$, dann berechnet sich $a + b$ aus der Addition der beiden Polynome a und b (normale Polynomaddition).

Multiplikation im $GF(2^8)$ Seien $a, b \in GF(2^8)$, dann berechnet sich $a \cdot b$ aus der Multiplikation der beiden Polynome a und b (normale Polynommultiplikation) mit anschließender Reduktion modulo f. Die Reduktion ist notwendig, da durch die Multiplikation der Grad des Ergebnispolynoms größer als der des irreduziblen Polynoms f werden kann.

Berechnen des Inversen eines Polynoms Die multiplikative Inverse eines Polynoms $a \in GF(2^8)$ lässt sich mittels des erweiterten Euklidischen Algorithmus berechnen. Eine weitere Möglichkeit besteht darin anstelle von a^{-1} einfach $a^{(2^8)-2} = a^{254}$ zu berechnen. Dies begründet sich darin, dass der $GF(2^8)/[0]$ genau $2^8 - 1 = 255$ Elemente enthält und

$$a^{255} \equiv a^0 \equiv 1 \quad \text{und daher} \quad a^{-1} \equiv a^{254}$$

Die Substitutionsschicht bei AES

Die S-Box AES besitzt im Gegensatz zu DES nur eine einzige nicht-lineare S-Box welche auf jedes Element der Zustandsmatrix $(a_{i,j})_{i,j\in[0,\ldots,3]}$ angewandt wird. Dies vereinfacht die Implementierung des AES gegenüber dem DES, da nur noch eine einzige S-Box, entweder in Software (z.B. als Nachschlagetabelle) oder in Hardware (z.B. als feste Verdrahtung), realisiert werden muss.

Die S-Box ist das einzige nicht-lineare Element des AES und trägt somit maßgeblich zur Sicherheit der Verschlüsselung bei. Details hierzu werden im Folgenden noch erläutert.

$$S : GF(2^8) \to GF(2^8)$$

Die durch die S-Box repräsentierte invertierbare Transformation bildet einen Zustand $(a_{i,j})_{i,j}$ auf einen Zustand

$$(b_{i,j})_{i,j} = \big(S(a_{i,j})\big)_{i,j}$$

ab, indem sie auf jedes Element des Zustandes (also byteweise) angewendet wird. Die Abbildung S wird deshalb auch `SubBytes` genannt.

`SubBytes` Die Transformation `SubBytes` besteht aus zwei Teilen:

1. Inversion I
2. Affine Abbildung A

Inversion Die Inversion ist eine Abbildung

$$I : GF(2^8) \to GF(2^8)$$

die definiert ist als

$$I(a) = \begin{cases} a^{-1} & \text{falls } a \neq 0 \\ 0 & \text{falls } a = 0 \end{cases}$$

Man berechnet für die Elemente $a = a_7x^7 + a_6x^6 + \ldots + a_0x^0 \in GF(2^8)$ eines Zustandes also die zugehörigen multiplikativen Inversen in $GF(2^8)$ wobei die Null auf sich selbst abgebildet wird.

Affine Abbildung Die zweite Transformation ist eine affine Abbildung

$$A : GF(2)^8 \to GF(2)^8$$

welche auf die Ergebnisse der Inversion I angewendet wird. Dazu interpretiert man die Elemente des jeweiligen Zustandes (Bytes) nicht mehr als Polynom, sondern als Vektor der Dimension 8 über dem $GF(2)$. Dies wird formal durch einen Isomorphismus Φ beschrieben:

$$\Phi : GF(2^8) \to GF(2)^8$$

wobei Φ definiert ist als

$$\Phi(a) = \Phi(a_7 x^7 + \ldots + a_o x^0) = (a_7, \ldots, a_0)^T$$

Die affine Abbildung A ist definiert als

$$A(a) = L \cdot a + c$$

Hierbei ist L eine bestimmte invertierbare 8×8 Matrix über $GF(2)$ und $c \in GF(2)^8$. Die Konstanten L und c wurden in [FIPS 197] definiert (s. Abbildung 1.9).

$$L := \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad , \quad c := \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Abbildung 1.9: Die Matrix L und der Vektor c der AES Transformation A

Die Abbildung A verknüpft die Elemente im $GF(2)$ (also auf Bitebene). Daher entspricht die Multiplikation einem logischem UND und die Addition dem XOR.

Zusammenfassende Beschreibung der S-Box Die S-Box lässt sich zusammenfassend folgendermaßen beschreiben:

$$\begin{aligned} S(a) &= \Phi^{-1}\Big(A\Big(\Phi\big(I(a)\big)\Big)\Big) \\ &= \Phi^{-1}\Big(L \cdot \big(\Phi(a^{-1})\big) + c\Big) \qquad \text{für } a \neq 0 \end{aligned}$$

Die durch die S-Box beschriebene Transformation ist invertierbar. Um $S^{-1}(a)$ zu berechnen, muss man sowohl die affine Abbildung A als auch die Inversion I rückgängig machen. Dabei ist $A^{-1}(a) = L^{-1} \cdot a + c$ und $I^{-1} = I$.

Die Permutationsschicht bei AES

Die Aufgabe der Permutationsschicht ist es, die Eingänge der S-Boxen geeignet zu "füttern", so dass eine lineare bzw. differentielle Attacke erschwert wird (mehr dazu später).

Die Permutationsschicht besteht aus den beiden $GF(2^8)$-linearen Transformationen `ShiftRows` und `MixColumns` und ist damit insgesamt auch $GF(2^8)$-linear.

Einschub Linearität: Sei $f : GF(2^8)^{16} \to GF(2^8)^{16}$, dann ist f

- linear, wenn $f(x + y) = f(x) + f(y)$
- $GF(2^8)$-linear, wenn zusätzlich $f(\lambda x) = \lambda f(x)$ wobei $\lambda \in GF(2^8)$

`ShiftRows` Die Transformation `ShiftRows` ist eine einfache Rotation[1] der Zeilen der Zustandsmatrix $(a_{i,j})_{i,j}$ nach rechts. Hierbei wird die i-te Zeile um $i - 1$ Positionen rotiert. Das bedeutet, dass die erste Zeile nicht, die zweite um eine Position, die dritte um zwei Positionen und die vierte Zeile um 3 Positionen rotiert wird.

`ShiftRows` lässt sich durch Umkehren der Rotationsrichtung rückgängig machen.

`MixColumns` Bei `MixColumns` wird jede Spalte eines Zustandes mit einer in [FIPS 197] festgelegten 4×4 Matrix (s. Abbildung 1.10) über $GF(2^8)$ multipliziert. Die Elemente dieser Matrix sind Bytes in hexadezimaler Schreibweise, die als Polynome des $GF(2^8)$ interpretiert werden. Das Byte $(02)_{16}$ steht beispielsweise für die Binärdarstellung $(00000010)_2$, welche wiederum das Polynom x repräsentiert und damit als Körperelement interpretiert werden kann.

Um die `MixColumns`-Operation rückgängig zu machen, genügt die Multiplikation mit der entsprechenden Inversen Matrix.

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

Abbildung 1.10: Die Matrix der AES Transformation `MixColumns`

`AddRoundKey` Bei der `AddRoundKey` Operation wird die 4×4-Schlüsselmatrix (Rundenschlüssel) zu dem aktuellem Zustand addiert (XOR).

Auch diese Transformation ist umkehrbar. Ein erneutes Addieren des selben Rundenschlüssels hebt `AddRoundKey` wieder auf.

[1] Rotation bedeutet, dass man ein Matrixelement, dass aus der Matrix "herausgeschoben" wird, auf der anderen Seite wieder "hineinschiebt"

Verschlüsselung

Man kann den gesamten Verschlüsselungsalgorithmus in einer C-ähnlichen Notation beschreiben, wobei die Funktionen (`Round`, `SubBytes`, etc.) mit Arrays arbeiten, die über entsprechende Pointer (`State`, `RoundKey`, etc.) an die Funktionen übergeben werden.

Die Hauptfunktion des AES lässt sich folgendermaßen beschreiben:

```
Rijndael(State,CipherKey)
{
  KeyExpansion(CipherKey,ExpandedKey);
  AddRoundKey(State,ExpandedKey);
  For( i=1 ; i<10 ; i++ ) Round(State,ExpandedKey + 4*i);
  FinalRound(State,ExpandedKey + 40);
}
```

Die Funktion `KeyExpansion` erzeugt das Schlüsselmaterial für alle Runden (d.h. alle Teilschlüssel). Der `ExpandedKey` ist ein Vektor, der (bei AES-128) in den ersten vier 4-Byte-Wörtern[2] den 128 Bit langen `CipherKey` enthält. Die restlichen 4-Byte-Wörter dieses Vektors enthalten die aus dem `CipherKey` abgeleiteten Rundenschlüssel.

Die Anzahl der Runden ist abhängig von der verwendeten Schlüssellänge. Bei AES-128 gibt es 10 Runden (einschließlich der letzte Runde).

Die Rundentransformationen lassen sich folgendermaßen beschreiben:

```
Round(State,RoundKey)
{
  SubBytes(State);
  ShiftRows(State);
  MixColumns(State);
  AddRoundKey(State);
}
```

In der letzten Runde wird die `MixColumns` Operation nicht mehr ausgeführt:

```
FinalRound(State,RoundKey)
{
  SubBytes(State);
  ShiftRows(State);
  AddRoundKey(State);
}
```

[2] auf einer 32 Bit Maschine entsprechen 4 Bytes (32 Bit) üblicherweise der Länge eines Integers

Entschlüsselung

Für die Entschlüsselung müssen alle Transformationen der Verschlüsselung rückgängig gemacht werden. Ähnlich wie bei DES werden dafür auch bei AES keine besonderen Änderungen des Algorithmus oder dessen Struktur benötigt.

1.3 Square-Attacke auf AES

Die Square-Attacke ist für reduzierte Versionen von AES (bis zu 6 Runden) schneller als eine Brute-Force-Attacke. Der Angriff funktioniert, da AES auf Elementen aus $GF(2^8)$ (Bytes) arbeitet und die Permutationsschicht, bestehend aus `ShiftRows` und `MixColumns`, $GF(2^8)$-linear ist. Wir beschreiben hier nur die Attacke auf 4 Runden. Diese kann sowohl durch eine weitere Runde am Anfang als auch eine weitere Runde am Ende erweitert werden.

Definition 2 *Eine Menge von AES-128 Zuständen (4×4 Matrizen) heißt* Delta-Set *(Δ-Set), wenn alle Positionen der Matrizen des Δ-Sets entweder aktiv oder passiv sind. Das bedeutet, dass für alle $a, b \in \Delta$ mit $a \neq b$ gilt:*

$$a_{ij} = b_{ij} \quad \textit{wenn Position } (i,j) \textit{ passiv ist, oder}$$

$$a_{ij} \neq b_{ij} \quad \textit{wenn Position } (i,j) \textit{ aktiv ist.}$$

Beispiel 1

$$\left\{ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right\}$$

ist ein Δ-Set, wobei jede Position passiv ist. □

Beispiel 2

$$\left\{ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right\}$$

ist ein Δ-Set mit nur einer einzigen aktiven Position bei a_{00}. Alle restlichen Positinen sind passiv. □

Beispiel 3

$$\left\{ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right\}$$

ist ebenso ein Δ-Set mit nur einer aktiven Position bei a_{00}. □

Beispiel 4

$$\left\{ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right\}$$

ist hingegen kein Δ-Set, da Position a_{00} und a_{01} sowohl aktiv als auch passiv sind. □

Um festzustellen, ob eine Menge von AES-Zuständen ein Δ-Set bildet, muss man also für jede Position a_{ij} prüfen, ob diese in allen Matrizen der Menge gleich (also passiv), oder in allen Matrizen unterschiedlich (also aktiv) ist.

Lemma 3 *Ein Δ-Set hat maximal 256 Elemente.*

Beweis Für alle Elemente a_{ij} eines Zustands gilt $a_{ij} \in GF(2^8)$. Da der $GF(2^8)$ der Körper mit $2^8 = 256$ Elementen ist, kann jedes a_{ij} maximal 256 verschiedene Werte annehmen.

In einem Δ-Set mit mindestens zwei Elementen gibt es immer mindestens eine aktive Position. Sei diese Position ohne Einschränkung $(0, 0)$. Wenn es ein Δ-Set mit 257 Elementen gäbe, dann gibt es mindestens zwei Matrizen $a, b \in \Delta$ für die gilt, dass $a_{00} = b_{00}$. Damit wäre die Position $(0, 0)$ aber nicht mehr aktiv und die Menge somit kein Δ-Set. □

1.3.1 Auswirkungen der AES-Transformationen auf Δ-Sets

Die AES-Verschlüsselung setzt sich aus mehreren Runden zusammen, welche aus den Operationen `SubBytes`, `ShiftRows`, `MixColumns` und `AddRoundKey` aufgebaut sind. Deren Auswirkung auf Δ-Sets werden im Folgenden genauer analysiert.

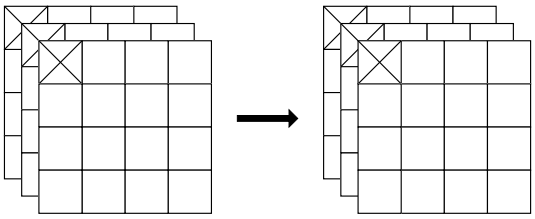

Abbildung 1.11: Die Auswirkung von `SubBytes` auf ein Δ-Set

Ausgangsbasis dieser Betrachtungen ist ein Δ-Set maximaler Größe mit einer einzelnen aktiven Position bei a_{00}, wobei ☒ eine aktive, und □ eine passive Position markiert.

Wie Abbildung 1.11 zeigt, hat die Operation `SubBytes` keine Auswirkungen auf das Δ-Set. Dies liegt daran, dass auf jede Position aller Zustände des Δ-Sets die selbe bijektive Abbildung (S-Box) angewendet wird. Das bedeutet, dass alle Zustände eines Δ-Sets gleichermaßen verändert werden.

Es ändern sich zwar die konkreten Werte der Zustände, jedoch bleibt die Δ-Set-Eigenschaft und dessen Struktur erhalten.

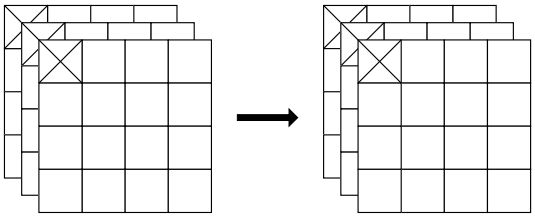

Abbildung 1.12: Die Auswirkung von `ShiftRows` auf ein Δ-Set

Wie Abbildung 1.12 zeigt, ändert auch ShiftRows nichts an der Anordnung der aktiven Positionen. Dies begründet sich darin, dass diese Operation die erste Zeile der Matrix nicht verschiebt. Befindet sich eine aktive Position in einer anderen Zeile, so würde sich diese verschieben.

Da jedoch alle Elemente des Δ-Sets gleichermaßen verschoben werden, hat dies keine Auswirkung auf die Δ-Set-Eigenschaft. Lediglich die Struktur des Δ-Sets wird durch das Verschieben der aktiven Positionen verändert.

Abbildung 1.13 zeigt, dass sich das Δ-Set durch die Operation `MixColumns` offensichtlich verändert.

Bei dieser Transformation wird der jeweils aktuelle Zustand mit einer konstanten 4×4 Matrix multipliziert. Dadurch wirkt sich eine aktive Position auf die gesamte Spalte aus, in der sie sich befindet. Dies hat zur Folge, dass die betroffene Spalte komplett aktiv wird.

Bei `AddRoundKey` wird zu jedem Zustand des Δ-Sets der selbe Schlüssel

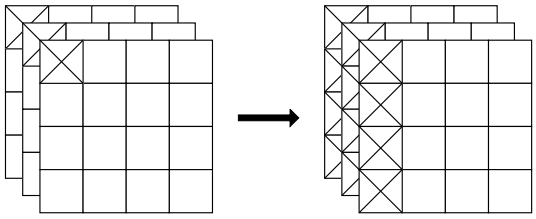

Abbildung 1.13: Die Auswirkung von `MixColumns` auf ein Δ-Set

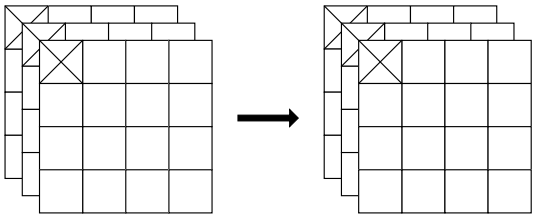

Abbildung 1.14: Die Auswirkung von `AddRoundKey` auf ein Δ-Set

addiert. Da folglich zu jeder Position jedes Zustands der selbe Wert des Rundenschlüssels addiert wird, bleibt diese weiterhin aktiv bzw. passiv.

1.3.2 Der Square-Angriff auf AES mit 4 Runden

Wir verfolgen ein Δ-Set mit einer aktiven Position bei a_{00} durch eine auf vier Runden gekürzte Variante von AES. Alle anderen Positionen des Δ-Sets seien passiv.

In Abbildung 1.15 kann man die Modifikationen des Δ-Sets über die vier Runden verfolgen. Man beachte, dass die letzte Runde eine `FinalRound` ist, und daher keine `MixColumns` Operation durchführt.

Das Δ-Set wird den Überlegungen aus dem vorherigen Abschnitt entsprechend durch die AES-Operationen verändert.

In der ersten Runde haben `SubBytes` und `ShiftRows` keine Auswirkungen auf das Δ-Set. Durch das `MixColumns` in der ersten Runde wird die gesamte erste Spalte aktiv. Das darauf folgende `AddRoundKey` und `SubBytes` der zweiten Runde haben wiederum keine weiteren Auswirkungen. In der zweiten Runde werden die aktiven Positionen durch `ShiftRows` über alle Spalten verteilt. Dies hat zur Folge, dass `MixColumns` alle Positionen des Δ-Sets zu aktiven Positionen macht. Die nachfolgenden Transformationen `AddRoundKey`, `SubBytes` und `ShiftRows` verursachen wiederum keine Veränderung. Wir halten fest, dass vor der `MixColumns`-Operation der dritten Runde nach wie vor alle Positionen aktiv sind. Da es vorkommen kann, dass das folgende `MixColumns` (Multiplikation jeder Spalte des Zustands mit einer konstanten

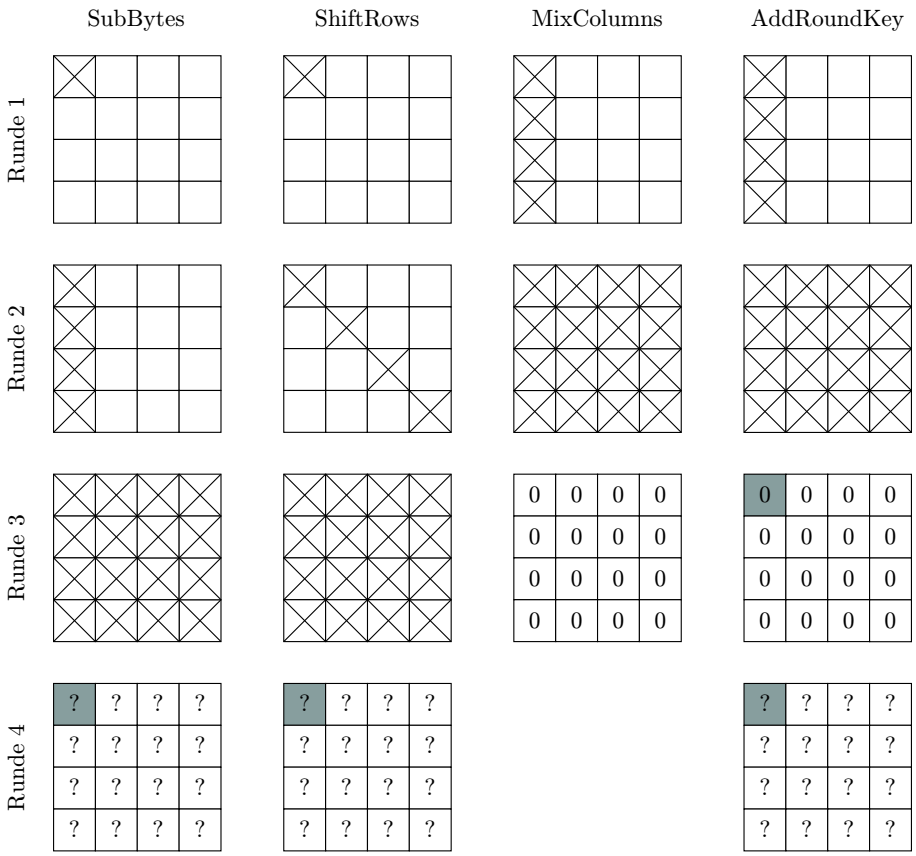

Abbildung 1.15: Der Square-Angriff auf AES mit 4 Runden

Matrix) Elemente auf sich selbst abbildet, kann man an dieser Stelle nicht mehr davon ausgehen, dass es sich bei der betrachteten Menge von Zuständen noch um ein Δ-Set handelt. Wir bezeichnen diese Menge von Zuständen daher als Δ'.

Δ' hat die interessante Eigenschaft, dass die Summe der Elemente an einer Position über alle Zustände gleich Null ist.

Theorem 4 (Balanced Eigenschaft) *Für jede Position nach der* `MixColumns`*-Operation der dritten Runde gilt:*

$$\sum_{a_{ij} \in \Delta'} a_{ij} = 0$$

wobei Δ' die Menge der Zustände bezeichnet.

Für den Beweis dieses Lemmas brauchen wir noch das folgende Lemma.

Lemma 5 *Sei K ein endlicher Körper mit $|K| > 2$, dann gilt*

$$\sum_{x \in K} x = 0$$

Beweis Sei $\alpha \in K \setminus \{0, 1\}$, dann gilt

$$\alpha \cdot \sum_{x \in K} x = \sum_{x \in K} \alpha x = \sum_{x \in K} x$$

subtrahiert man $\sum_{x \in K} x$ auf beiden Seiten, so erhält man

$$(\alpha - 1) \sum_{x \in K} x = 0$$

da $\alpha \neq 1$ folgt die Behauptung. □

Beweis von Theorem 4 Die `MixColumns`-Operation lässt sich formalisieren als

$$a_{ij} = \sum_{i=0}^{3} \lambda_{ji} b_{ij}$$

wobei λ_{ji} die Einträge der `MixClomuns`-Matrix sind. Das heißt,

$$\begin{aligned} \sum_{a_{ij} \in \Delta'} a_{ij} &= \sum_{b_{ij} \in \Delta} \left(\sum_{i=0}^{3} \lambda_{ji} b_{ij} \right) \\ &= \sum_{i=0}^{3} \lambda_{ji} \left(\sum_{b_{ij} \in \Delta} b_{ij} \right) \end{aligned}$$

wobei Δ die Zustände *vor* der `MixColumns`-Operation, also das Δ-Set, bezeichnet.

Da Δ ein maximales Δ-Set mit nur aktiven Positionen ist, gilt nach Lemma 5

$$\sum_{b_{ij} \in \Delta} b_{ij} = \sum_{x \in \mathbb{F}_2^8} x = 0$$

□

Der Ablauf des Angriff

1. Rate ein Schlüsselbyte nach vier Runden AES

2. Berechne den Zustand an dieser Position nach der dritten Runde (über die in Abbildung 1.15 grau hinterlegten Werte)

3. Teste die *Balanced*-Eigenschaft (also ob die Summe der Werte an dieser Position über alle Zustände gleich Null ist)

   - wenn ja ⇒ geratenes Schlüsselbyte könnte richtig sein
   - wenn nicht ⇒ neu Raten (also weiter bei 1.)

Aufwandsabschätzung Da man jedes Schlüsselbyte einzeln raten und testen kann, hat der Square-Angriff einen Aufwand von $16 \cdot 2^8 = 2^{12}$ (im Vergleich zu 2^{128} bei Brute Force).

1.4 Differentielle Attacke auf DES

1.4.1 Geschichte der differentiellen Attacke

Die differentielle Kryptoanalyse ist ein Analyseverfahren für iterierte Blockchiffren wie z.B. den DES, an dem es später demonstriert wurde. Die Idee für dieses Verfahren wurde 1990 von Eli Biham und Adi Shamir veröffentlicht. Diese Methode war den Entwicklern von DES schon lange vorher bekannt, so dass sich für DES kein wesentlich stärkerer Angriff als Brute-Force konstruieren ließ.

1.4.2 Grundlagen

Die Idee des Verfahrens besteht darin, anstatt einzelner Klartext/Ciphertext-Paare XOR-Differenzen von solchen Paaren zu untersuchen. Das dies sinnvoll sein kann, ergibt sich aus den folgenden Überlegungen.

- die Expansion $E(x)$ linear ist, d.h. $E(x + x') = E(x) + E(x')$ [3]
- die Permutation $P(x)$ linear ist, d.h. $P(x + x') = P(x) + P(x')$

[3] Hier entspricht das "+" dem XOR

- für das Key-XOR gilt mit dem jeweiligen Rundenschlüssel k_i:

$$(x + k_i) + (x' + k_i) = x + x' \quad \text{mit } i \in \{1, 2, \ldots, 16\}$$

  $\Rightarrow$ Die Schlüsselabhängigkeit vor den S-Boxen verschwindet bei der Betrachtung von Differenzen.

Da die S-Boxen die einzigen nicht-linearen Funktionen des DES sind, müssen diese gesondert untersucht werden. Im Folgenden wird betrachtet, wie sich Eingabedifferenzen und Ausgabedifferenzen bei den S-Boxen verhalten: Sei Δe eine feste Eingabedifferenz ($\Delta e \in \mathbb{F}_2^6$) und Δa eine feste Ausgabedifferenz ($\Delta a \in \mathbb{F}_2^4$). Wir möchten hier Eingabe-Paare mit der gegebenen Eingabedifferenz bestimmen, deren Bilder unter einer S-Box die gegebene Ausgabedifferenz liefern.

Dazu bestimmt man die Paare (x, x'), so dass gilt:

$$x + x' = \Delta e$$

$$S(x) + S(x') = \Delta a$$

Dies ist äquivalent zu der Bestimmung der Lösungen x von:

$$S(x) + S(x + \Delta e) = \Delta a$$

Die Lösungsmenge dieser Gleichung, und insbesondere die Größe dieser Menge, wird im Folgenden eine entscheidende Rolle spielen.

1.4.3 Angriff auf DES mit einer Runde

Zunächst soll aus Anschauungsgründen ein differentieller Angriff auf einen zu einer Runde reduzierten DES (Final Round) betrachtet werden. Ziel des Angriffs ist es, mit einigen Eingabedifferenzen $\Delta m = (\Delta m_l, \Delta m_r)$ und den dazugehörigen Ausgabepaaren c, c' auf einige Bits des Rundenschlüssels zu schließen. Abbildung 1.16 zeigt den reduzierten DES, wobei F die Rundenfunktion, und k_1 den Rundenschlüssel darstellt. Dabei ist zu beachten, dass bei der Final Round keine Vertauschung stattfindet.

Betrachtung der Rundenfunktion F Betrachtet man die Ein- und Ausgaben der S-Boxen separat für jedes Paar, stellt man fest, dass sich zwar die Ausgabe der S-Boxen a aus den Ausgaben c_l berechnen lässt

$$a = P^{-1}(y), \quad \text{wobei} \quad y = m_l + c_l$$

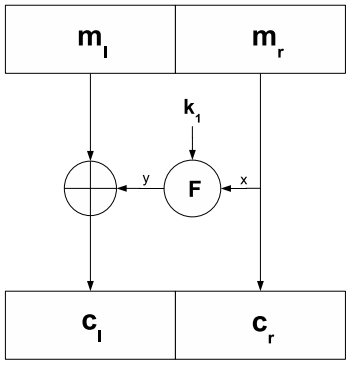

Abbildung 1.16: DES mit einer Runde

die Eingaben der S-Boxen jedoch nicht ermittelbar sind, da diese vom Schlüssel k_1 abhängig sind.

$$e = E(x) + k_1.$$

Betrachtet man hingegen die Differenz der Eingabewerte x, x', so gilt:

$$\Delta e = e + e' = (E(x) + k_1) + (E(x') + k_1) = E(x) + E(x') = E(x + x') = E(\Delta x)$$

Die Eingabedifferenz kann somit ohne Kenntnis von k_1 bestimmt werden und ist ausschließlich von der Differenz von x und x' abhängig.
Für die Ausgabedifferenz gilt analog:

$$\Delta a = a + a' = P^{-1}(y) + P^{-1}(y') = P^{-1}(y + y') = P^{-1}(\Delta y)$$

Wir teilen den Eingabewert $e \in \mathbb{F}_2^{48}$ in die Eingaben $e_i \in \mathbb{F}_2^6$ der einzelnen S-Boxen auf $e = (e_0, \ldots, e_7)$ und die Ausgabe $a \in \mathbb{F}_2^{32}$ in die Ausgaben $a_i \in \mathbb{F}_2^4$ der S-Boxen auf $a = (a_0, \ldots, a_7)$. Um zu bestimmen welche Eingabepaare $(e, e + \Delta e)$ die Ausgabedifferenz Δa erzeugen, müssen wir für jede S-Box S_i die Lösungen der Gleichung

$$S_j(e_j) + S_j(e_j + \Delta e_j) = \Delta a_j$$

bestimmen. Diese können in vorher berechneten Tabellen, so genannten *Differenzentabellen* gespeichert werden.

Einschub: Differenzentabellen In eine S-Box gehen 6 Bit ein, d.h. für jede gibt es 64 verschiedene Eingabedifferenzen. e_j sei der 6-Bit-Eingabewert

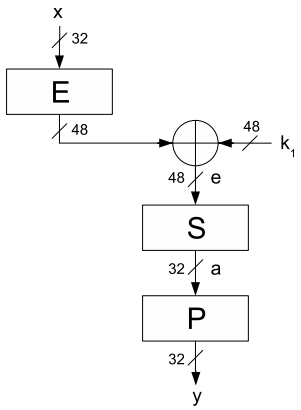

Abbildung 1.17: Rundenfunktion F

der S-Box S_j. Eine Eingabedifferenz kann von 64 Eingabepaaren (e_j, e'_j) erzeugt werden. Zu diesen Eingabedifferenzen bestimmt man die möglichen Eingabepaare und die entsprechenden Ausgabedifferenzen. Diese Zuordnungen legt man für jede Eingabedifferenz und S-Box in einer Differenzentabelle ab. Die S-Boxen bilden 6 auf 4 Bit ab. Damit gibt es nur 16 verschiedene Ausgabedifferenzen. Für eine gegebene Ausgabedifferenz Δa_j und hier ebenfalls bekannte Eingabedifferenz Δe_j kann man für jede S-Box S_j die Eingabepaare $(e_j, e_j + \Delta e_j)$ bestimmen. Dabei gilt für diese Eingabepaare:

$$S_j(e_j) + S_j(e_j + \Delta e_j) = \Delta a_j$$

Berechnung der Schlüsselkandidaten Die Schlüsselkandidaten für k_1 erhält man, indem man Kombinationen für alle möglichen Kandidaten von e_j der S-Box S_j

$$k_1 = e_1||e_2|| \ldots ||e_8 + E(x)$$

berechnet. Die Anzahl der so berechneten Schlüsselkandidaten entspricht also

$$\prod_{j=0}^{7} |\{e_j \in \mathbb{F}_2^6 \mid S_j(e_j) + S_j(e_j + \Delta e_j) = \Delta a_j\}|$$

Bestimmung des Schlüssels Um aus den Schlüsselkandidaten den Schlüssel k_1 zu finden, verschlüsselt man den bekannten Klartext m mit dem Kandidat für k_1 und überprüft, ob der entstandene Geheimtext c entspricht.

1.4.4 Angriff auf DES mit mehreren Runden

Bei einem DES mit mehr als 5 Runden, liefert nicht mehr jedes Eingabepaar mit der dazugehörigen Ausgabedifferenz eine Schlüsselkandidatenmenge, die den richtigen Rundenschlüssel enthält. Die Differenzen lassen sich hier nicht mehr genau bestimmen. Die Grundidee ist, dass wenn die Differenzen schon nicht exakt bestimmt werden können, sie mit möglichst großer Wahrscheinlichkeit vorauszusagen. Für den Angriff auf mehrere Runden nimmt man daher an, dass die Eingabedifferenzen für die letzte Runde bekannt seien und berechnet unter dieser Annahme die möglichen Schlüsselkandidaten. Stimmt die Annahme, ist der richtige Schlüssel in der berechneten Menge enthalten. Stimmt die Annahme nicht, bekommt man zufällig verteilte Schlüsselkandidaten.

Da jedoch der richtige Rundenschlüssel mit größerer Wahrscheinlichkeit in den Schlüsselkandidatenmengen enthalten ist, kann er bei ausreichend vielen Ein- und Ausgabepaaren ermittelt werden. Für dieses Vorgehen definiert man eine Struktur, die im folgenden *Charakteristik* genannt wird.

Zuerst benötigen wir noch einige Bezeichnungen.

Definition 6 *Für eine Abbildung $H : \mathbb{F}_2^n \to \mathbb{F}_2^m$ bezeichen wir die Wahrscheinlichkeit, dass eine Eingabedifferenz ΔI mit $\Delta I \in \mathbb{F}_2^n$ durch Anwendung der Funktion H die Ausgabedifferenz ΔO mit $\Delta O \in \mathbb{F}_2^m$ liefert, mit $\mathcal{P}(\Delta I \to \Delta O|H)$. Es ist*

$$\mathcal{P}(\Delta I \to \Delta O|H) = \frac{|\{x \in \mathbb{F}_2^n \mid \quad H(x) + H(x + \Delta I) = \Delta O\}|}{2^n}$$

Insbesondere gilt mit dieser Notation für die S-Boxen des DES:

$$\mathcal{P}_j(\Delta e \to \Delta a|S_j) = \frac{|\{e_j \in \mathbb{F}_2^6 \mid S_j(e_j) + S_j(e_j + \Delta e) = \Delta a\}|}{2^6}$$

Da die Rundenfunktion F bei DES auch vom Schlüssel abhängig ist, ergibt sich hier für $\Delta x \in \mathbb{F}_2^{32}$ und $\Delta y \in \mathbb{F}_2^{32}$

$$\mathcal{P}(\Delta x \to \Delta y|F) = \frac{|\{(x,k) \in \mathbb{F}_2^{32} \times \mathbb{F}_2^{48} \mid F(x,k) + F(x + \Delta x, k) = \Delta y\}|}{2^{32} \cdot 2^{48}}$$

Theorem 7 *Seien $\Delta x, \Delta y \in \mathbb{F}_2^{32}$ gegeben. Dann gilt unabhängig von dem Rundenschlüssel folgender Zusammenhang zwischen den Wahrscheinlichkeiten bezüglich der S-Boxen und der Wahrscheinlichkeit der Rundenfunktion F:*

$$\mathcal{P}(\Delta x \to \Delta y|F) = \prod_{j=1}^{8} \mathcal{P}_j((E(\Delta x)_j \to (P^{-1}(\Delta y))_j|S_j)$$

Beweis

$$\begin{aligned}
\mathcal{P}(\Delta x \to \Delta y|F) &= \frac{|\{(x,k)|F(x,k)+F(x+\Delta x,k)=\Delta y\}|}{2^{32}\cdot 2^{48}} \\
&= \frac{|\{(x,k)|P(S(E(x)+k))+P(S(E(x+\Delta x)+k))=\Delta y\}|}{2^{32}\cdot 2^{48}} \\
&= \frac{|\{(x,k)|S(E(x)+k)+S(E(x+\Delta x)+k)=P^{-1}(\Delta y)\}|}{2^{32}\cdot 2^{48}} \\
&= \frac{|\{(x,k)|S(E(x)+k)+S(E(x)+k+E(\Delta x))=P^{-1}(\Delta y)|}{2^{32}\cdot 2^{48}}
\end{aligned}$$

Wir ersetzen hier $e = E(x) + k$ und es ergibt sich

$$\begin{aligned}
\mathcal{P}(\Delta x \to \Delta y|F) &= \frac{|\{(x,e)|S(e)+S(e+E(\Delta x))=P^{-1}(\Delta y)\}|}{2^{32}\cdot 2^{48}} \\
&= \frac{|\{e|S(e)+S(e+\Delta e)=P^{-1}(\Delta y)\}|}{2^{48}} \\
&= \prod_{j=1}^{8} \frac{|\{e_j|S_j(e_j)+S_j(e_j+\Delta e_j)=(P^{-1}(\Delta y))_j\}|}{2^6} \\
&= \prod_{j=1}^{8} \mathcal{P}_j((E(\Delta x)_j \to (P^{-1}(\Delta y))_j|S_j)
\end{aligned}$$

□

Dieser Satz zeigt also, dass es ausreicht, die einzelnen S-Boxen zu betrachten, um die Wahrscheinlichkeit für die gesamte Rundenfunktion F zu bestimmen. Insbesondere sind diese Wahrscheinlichkeiten nicht von dem Rundenschlüssel abhängig.

DES-Charakteristik Eine DES_n-Charakterisik $\Gamma = (\Delta m, \lambda, \Delta c)$ für n Runden ist wie folgt definiert:

$$\lambda = (\lambda_1, \ldots, \lambda_n) \quad \text{mit} \quad \lambda_i = (\Delta x_i, \Delta y_i)$$

Δx_i = Eingabedifferenz für die i-te Runde (F-Funktion)
Δy_i = Ausgabedifferenz für die i-te Runde (F-Funktion)
Dabei gelten folgende Abhängigkeiten.

$$\begin{aligned}
\Delta x_1 &= \Delta m_r \\
\Delta x_2 &= \Delta m_l + \Delta y_1 \\
\Delta y_n &= \Delta c_l + \Delta x_{n-1} \\
\Delta y_i &= \Delta x_{i-1} + \Delta x_{i+1} \quad \textit{wobei} \quad 2 \le i \le n-1
\end{aligned}$$

Wahrscheinlichkeiten einer DES_n-Charakteristik Die Wahrscheinlichkeit, dass ein Paar von Nachrichten $(m, m + \Delta m)$ die Charakteristik erfüllt, setzt sich aus den Wahrscheinlichkeiten für jede Runde zusammen. Δm erzeugt Δc gemäß einer DES_n-Charakteristik Γ mit der Wahrscheinlichkeit:

$$p^\Gamma = \prod_{i=1}^{n} p_i^\Gamma = \prod_{i=1}^{n} P(\Delta x_i \rightarrow \Delta y_i | F)$$

wobei p_i^Γ die DES-Charakteristik der i-ten Runde ist. Während sich für wenige Runden noch leicht gute Charakteristiken finden lassen, ist die Konstruktion von guten Charakteristiken für mehrere Runden relativ kompliziert. Statt direkt Charakteristiken für viele Runden zu konstruieren, versucht man daher Charakteristiken für wenige Runden zu verbinden. Dies wird als Konkatenation von DES-Charakteristiken bezeichnet.

Konkatenation zweier DES-Charakteristiken Zwei DES-Charakteristiken $\Gamma' = (\Delta m', \lambda', \Delta c')$ und $\Gamma'' = (\Delta m'', \lambda'', \Delta c'')$ lassen sich verbinden, wenn folgendes gilt:

$$\Delta c'_l = \Delta m''_r \quad \text{und} \quad \Delta c'_r = \Delta m''_l$$

Für die sich ergebende Charakteristik $\Gamma = \Gamma'\Gamma'' = (\Delta m', \lambda, \Delta c'')$ ist $\lambda = (\lambda'_1, \lambda'_2, \ldots, \lambda'_n, \lambda''_1, \lambda''_2, \ldots, \lambda''_m)$.

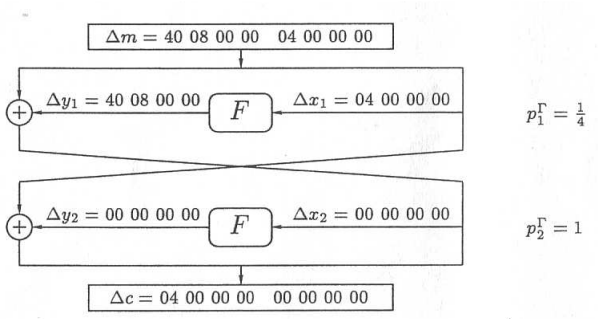

Abbildung 1.18: Konkatenierte Charakteristiken

Noch einfacher geht es mit iterativen Charakteristiken.

Iterative DES-Charakteristik Eine DES-Charakteristik $\Gamma = (\Delta m, \lambda, \Delta c)$ mit $\Delta m = \Delta m_l || \Delta m_r$ und $\Delta c = \Delta c_l || \Delta c_r$ heißt iterativ, wenn gilt:

$$\Delta m_l = \Delta c_r \quad \text{und} \quad \Delta m_r = \Delta c_l$$

Mit einer gegebenen iterativen Charakteristik lassen sich somit Charakteristiken für beliebig viele Runden finden, indem die Charakteristik oft genug wiederholt wird.

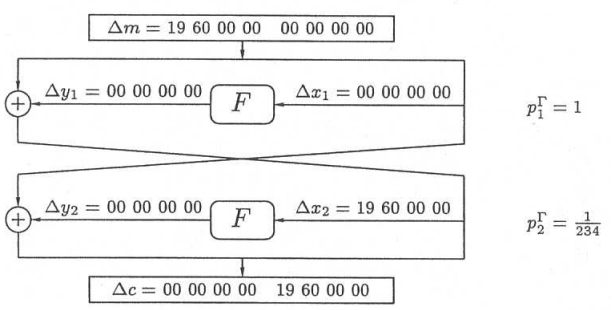

Abbildung 1.19: Iterative Charakteristik

1.4.5 Schlüsselbestimmung mit Hilfe der Charakteristik

Praktisch erfolgt die Durchführung der differentiellen Kryptoanalyse des DES_n mittels einer DES_{n-1} Charakteristik Γ folgendermaßen:

- Man lässt, wie in Abbildung 1.20 ersichtlich, Klartextpaare M, M' mit $M + M' = \Delta m$ DES_n-verschlüsseln ($DES_n(M) = C, DES_n(M') = C'$) und für die zugehörigen Geheimtextpaare ($C = C_l||C_r, C' = C'_l||C'_r$) wird jeweils ΔC mit $\Delta C = C + C'$ bestimmt. Man nimmt nun an, dass die jeweilige Ausgabedifferenz der $(n-1)$-ten Runde der DES_{n-1} Charakteristik $\Gamma = (\Delta m, \lambda, \Delta c)$ entspricht, also gleich $\Delta c = c_l||c_r$ ist. Unter dieser Annahme lässt sich dann die jeweilige Ausgabedifferenz Δy_n der letzen DES_n-Rundenfunktion mit $\Delta y_n = \Delta c_r + \Delta C_l$ bestimmen. Dies bedeutet, dass man sich somit in der gleichen Situation wie bei der Attacke auf nur eine Runde DES befindet. Die Schlüsselkandidaten lassen sich genau wie in Abschnitt 1.4.3 bestimmen.

Das Problem bei diesem Vorgehen ist jedoch, dass die Annahme, ob die Charakteristik erfüllt wird, nur mit der Wahrscheinlichkeit p^Γ wahr ist. Es können somit 2 Fälle auftreten:

1. Ein Klartextpaar M, M' liefert die der Charakteristik Γ entsprechende Ausgangsdifferenz Δc als Ausgansdifferenz der $(n-1)$-ten Runde. Dieser Fall tritt mit der Wahrscheinlichkeit von mindestens p^Γ ein.

2. Ein Klartextpaar M, M' liefert **nicht** die der Charakteristik Γ entsprechende Ausgangsdifferenz Δc als Ausgansdifferenz der $(n-1)$-ten Runde. Dieser Fall tritt mit der Wahrscheinlichkeit höchstens $1 - p^\Gamma$ ein.

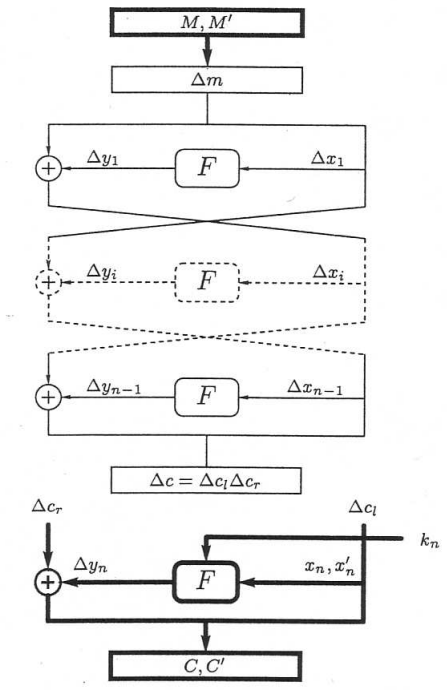

Abbildung 1.20: Verwendung einer Charakteristik

Ob nun bei einer Verschlüsselung der erste oder zweite Fall eingetreten ist lässt sich nicht ermitteln. Da es ohne Kenntnis des Rundenschlüssels der letzen Runde nicht möglich ist, die Annahme, dass die Verschlüsselung des Klartextpaares auch wirklich das Δc gemäß der Charakteristik liefert, zu verifizieren. Je nachdem ob die Annahme richtig oder falsch ist ergibt sich für die Schlüsselkandidaten folgende Situation:

- Ist die Annahme für ein Klartextpaar wahr, so enthält die Schlüsselkandidatenmenge den gesuchten Schlüssel k_n.
- Ist die Annahme falsch, so kann man Annehmen, dass man zufällig verteilte Schlüsselkandidaten erhält.

Daraus ergibt sich, dass in der Gesamtmenge aller erhaltenen Schlüsselkandidaten der richtige Teilschlüssel also *mindestens* mit der Wahrscheinlichkeit p^{Γ} vertreten ist.

Den richtigen Teilschlüssel findet man also, indem man viele Klartextpaare auswertet und für jeden möglichen Teilschlüssel die Häufigkeit seines Vorkommens in der Gesamtkandidatenmenge bestimmt. Falls p^{Γ} nun groß genug ist und genügend viele Klartextpaare ausgewertet werden, ist der richtige Teilschlüssel am häufigsten vorhanden und kann somit erkannt werden.

Ein Maß für die Güte der Attacke ist somit das Vorkommen des richtigen Schlüssels im Verhältnis zu den falschen Schlüsseln, das sogenannte Signal-to-Noise Ratio.

1.4.6 Signal-To-Noise-Ratio

Sei $\mathcal{M}$ die Anzahl der betrachteten Klartextpaare mit der Differenz Δm und α die durchschnitlliche Anzahl der Teilschlüsselkandidaten, die ein Klartextpaar liefert. Der gesuchte Teilschlüssel k_n ist bei Betrachtung eines Klartextpaares mit der Wahrscheinlichkeit p^Γ in der Menge der resultierenden Schlüsselkandidaten enthalten. Bei $\mathcal{M}$ Klartextpaaren findet man k_n in der Gesamtmenge der Schlüsselkandidaten erwartungsgemäß $\mathcal{M} \cdot p^\Gamma$ mal. Ein falscher Schlüssel ist erwartungsgemäß $\frac{\mathcal{M}\cdot\alpha}{2^{56}}$ mal in der Gesamtschlüsselkandidatenmenge enthalten. Das Signal-To-Noise-Ratio ist definiert als:

$$\frac{S}{N} = \frac{\mathcal{M} \cdot p^\Gamma}{\frac{\mathcal{M}\cdot\alpha}{2^{56}}} = \frac{2^{56} \cdot p^\Gamma}{\alpha}$$

Bei einem S/N-Wert zwischen 2 und 4 lässt sich die differentielle Attacke effizient anwenden. In der Praxis wurden DES-Varianten verschiedener Runden mit Hilfe der differentiellen Attacke attackiert. Für die Besten bekannten Charakteristiken, und einigen weiteren Tricks, auf die hier nicht eingegangen wurde, ergibt sich hierbei folgender Aufwand:

Anzahl der Runden	Anzahl der Paare	Anzahl der Schlüsselbits	p^Γ
4	2^3	42	1
5	2^3	42	1
6	2^7	30	2^{-4}
8	2^{15}	30	1/10486
9	2^{25}	30	1/1000000
9	2^{26}	48	2^{-24}
10	2^{34}	18	2^{-32}
11	2^{35}	48	2^{-32}
12	2^{42}	18	2^{-40}
13	2^{43}	48	2^{-40}
14	2^{50}	18	2^{-48}
15	2^{51}	48	2^{-48}
16	2^{57}	18	2^{-56}

Für DES mit 16 Runden beträgt die Anzahl der benötigten Klartextpaare demnach 2^{57}. Somit ist die Praxistauglichkeit nicht gegeben, da in der Praxis niemand 2^{57} Klartextpaare mit dem gleichen Schlüssel verschlüsseln wird.

1.4.7 Fazit für das Design von Verschlüsselungsalgorithmen

Nachdem die differentielle Attacke jetzt ausführlich dargestellt wurde, stellt sich die Frage, wie man einen Blockcipher gegen diese Attacke schützen kann. Hierzu zuerst noch eine Definition.

Definition 8 *Für eine Abbildung* $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ *bezeichne*

$$\Delta_{F,c}(x) := F(x) + F(x + c).$$

Der Wert

$$\delta_F := \max_{c \in \mathbb{F}_2^n, c \neq 0, a \in \mathbb{F}_2^m} |\Delta_{F,c}(a)^{-1}|$$

heißt ***Uniformität*** *von* F*.*

Wie wir gesehen haben ist eine differentielle Attacke umso effizienter je bessere Charakteristiken gefunden werden können. Der Designer eines Verschlüsselungsalgorithmus sollte deshalb darauf achten, dass keine Charakteristik mit großer Wahrscheinlichkeit auftritt. Da die Wahrscheinlichkeit einer Charakteristik Γ beschränkt ist durch

$$p^\Gamma = \left(\frac{\delta_S}{2^n}\right)^{\#\text{aktive S-Boxen}},$$

wobei die S-Box hier eine Abbildung von $S : \mathbb{F}_2^n \to F_2^m$ ist, gibt es hierfür zwei Möglichkeiten:

1. Man minimiert die Wahrscheinlichkeit, dass eine feste Eingabedifferenz unter Anwendung einer S-Box eine bestimmte Ausgabedifferenz liefert. Für $c \neq 0$ sollte die Gleichung $S(x)+S(x+c) = \Delta a$ also möglichst wenig Lösungen haben. Dies ist gleichbedeutend damit, dass die Uniformität von S klein ist.

2. Über alle Runden der Charakteristik müssen möglichst viele S-Boxen aktiv sein, d.h. die Eingabedifferenz muss $\neq 0$ sein.

In den folgenden Abschnitten soll erläutert werden, wie diese beiden Punkte bei dem Design von DES und AES umgesetzt wurden.

1.4.8 DES-Kriterien

Die folgenden Anmerkungen sind Auszüge aus [Coppersmith 94]. Das Kriterium (S-7) gibt als obere Schranke für die Uniformität der S-Boxen den Wert 8 vor. Damit ist die Wahrscheinlichkeit

$$\mathcal{P}(\Delta e \to \Delta a | S_i) \leq \frac{\text{Uniformität}(S_i)}{32} = \frac{8}{32} = \frac{1}{4} \quad \text{für jedes} \quad i \in \{1, \ldots, 8\}$$

Die weiteren Design-Kriterien sind zum Großteil darauf angelegt, die Anzahl der aktiven S-Boxen zu maximieren. Wir wollen dies nicht vollständig erläutern, sondern nur an einigen Beispielen deutlich machen, wie diese Kriterien wirken und damit ein wenig Licht in das Dunkel der DES S-Boxen bringen.

Eine aktive S-Box

Lemma 9 *Sei in einer Runde i genau eine S-Box aktiv. In den Runden $(i+1)$ und $(i-1)$ müssen (zusammen) mindestens 2 S-Boxen aktiv sein.*

Beweis Sei S-Box S_j die einzige aktive S-Box in Runde i. Wir bezeichnen die Eingabedifferenz für diese S-Box mit

$$\Delta I_j = \quad a \quad b \quad c \quad d \quad e \quad f \ .$$

Da insbesondere S_{j-1} und S_{j+1} passiv sind, d.h die Eingabedifferenz für beide S-Boxen $(0,0,0,0,0,0)$ ist, folgt mit der Struktur der Expansionsfunktion, dass $a = b = e = f = 0$ gelten muss.

$$\Delta I_j = \quad 0 \quad 0 \quad c \quad d \quad 0 \quad 0$$

Dann folgt aus den DES-Kriterien (S-4) und (S-5), dass die Ausgabedifferenz mindestens das Gewicht 2 hat. Daher müssen in den Runden $(i+1)$ und $(i-1)$ (zusammen) mindestens 2 S-Boxen aktiv sein. □

Drei aktive S-Boxen Es seien genau die S-Boxen S_j, S_{j+1}, S_{j+2} aktiv. Man bestimmt die Wahrscheinlichkeit, dass für diese S-Boxen die Ausgabedifferenz aller S-Boxen 0 ist. Analog wie oben untersuchen wir die möglichen Eingabedifferenzen für diese S-Boxen.

- Eingabedifferenz der S-Box S_j:

$$\Delta I_j = \quad a \quad b \quad c \quad d \quad e \quad f$$

- Eingabedifferenz der S-Box S_{j+1}:

$$\Delta I_{j+1} = \; e \quad f \quad g \quad h \quad i \quad j$$

- Eingabedifferenz der S-Box S_{j+2}:

$$\Delta I_{j+2} = \; i \quad j \quad k \quad l \quad m \quad n$$

Die Werte a, b, m, n müssen 0 sein, da S_{j-1} und S_{j+3} passiv sind. Weiterhin müssen f und i stets 1 sein (siehe Kriterium (S-3)), da sonst die Ausgabedifferenz $\neq 0$ ist. Mit Kriterium (S-6) folgt nun, dass $j = 0$ ist, da sonst die Ausgabedifferenz $\neq 0$ ist. Schliesslich muss e immer 1 sein, da sonst wegen Kriterium (S-3) die Ausgabedifferenz 0 nicht möglich ist. Zusammenfassend ergibt sich:

- Eingabedifferenz der S-Box S_j:

$$\Delta I_j = \; 0 \quad 0 \quad c \quad d \quad 1 \quad 1$$

- Eingabedifferenz der S-Box S_{j+1}:

$$\Delta I_{j+1} = \; 1 \quad 1 \quad g \quad h \quad 1 \quad 0$$

- Eingabedifferenz der S-Box S_{j+2}:

$$\Delta I_{j+2} = \; 1 \quad 0 \quad k \quad l \quad 0 \quad 0$$

Wir führen folgende Notation ein:

$$q_j = \max_{c,d \in \mathbb{F}_2} P(00cd11 \to 0 | S_j)$$

$$q_{j+1} = \max_{g,h \in \mathbb{F}_2} P(11gh10 \to 0 | S_{j+1})$$

$$q_{j+2} = \max_{k,l \in \mathbb{F}_2} P(10kl00 \to 0 | S_{j+2})$$

Die Wahrscheinlichkeit für eine solche Situation ist daher wie folgt beschränkt:

$$d \leq \max_j q_j q_{j+1} q_{j+2}.$$

Das oben noch fehlende Kriterium (S-8) besagt, dass diese Wahrscheinlichkeit $d \leq \frac{1}{234}$ ist.

Es verwundert nicht, dass es speziell für diesen Fall von drei aktiven S-Boxen ein weiteres Kriterium gibt. Da diese Situation sehr einfach die Konstruktion von iterativen Charakteristiken erlaubt, wurde besonderer Wert darauf gelegt, diese Wahrscheinlichkeit zu minimieren. Die besten bekannten Charakteristiken sind genau von diesem Typ. Ohne das zusätzliche Kriterium (S-8) hätte man mit (S-7) nur die Abschätzung $d \leq (1/4)^3 = 1/64$ erhalten.

1.4.9 Resistenz von AES gegen die Differentielle Attacke

Während bei DES die Design-Kriterien noch sehr an eine Ad-Hoc-Konstruktion erinnern, zeigen die AES-Kriterien das tiefere Verständnis der mathematischen Hintergründe, das in der Zwischenzeit erreicht wurde. Auch hier sind wieder beide Ansätze, das Minimieren der Uniformität und das Maximieren der aktiven S-Boxen, verfolgt worden. Zu dem ersten Punkt zeigen wir zuerst den folgenden Satz:

Theorem 10 *Sei $n \in \mathbb{N}$ und*

$$\begin{aligned} S : \mathbb{F}_{2^n} &\rightarrow \mathbb{F}_{2^n} \\ S(x) &= x^{-1} \end{aligned}$$

gegeben. Für Uniformität δ_S von S gilt

$$\delta_S = \begin{cases} 2 & n \text{ ungerade} \\ 4 & n \text{ gerade} \end{cases}$$

Beweis Wir betrachten für $c, d \in \mathbb{F}_{2^n}$ und $c \neq 0$ die Gleichung

$$S(x) + S(x + c) = d$$

Mit der Definition von S ergibt sich

$$x^{-1} + (x + c)^{-1} = d$$

Durch eine Multiplikation mit c ergibt sich

$$\frac{c}{x} + \frac{1}{\frac{x}{c} + 1} = dc$$

Jetzt setzen wir $dc = d$ und $\frac{x}{c} = x$, was zu folgender Formel führt:

$$x^{-1} + (x + 1)^{-1} + d = 0 \tag{1.1}$$

Wir müssen zeigen, dass diese Gleichung für d maximal 2 Lösungen hat, wenn n ungerade ist und maximal 4 Lösungen hat, wenn n gerade ist. Wir nehmen zuerst an, dass $x \notin \mathbb{F}_2$ ist. Dann dürfen wir die Gleichung 1.1 mit $x(x + 1)$ multiplizieren und erhalten

$$(x + 1) + x + dx(x + 1) = 0$$

$$\Leftrightarrow dx^2 + dx + 1 = 0$$

Dies ist eine quadratische Gleichung und hat somit maximal 2 Lösungen. Insbesondere haben wir somit schon bewiesen, dass die Uniformität in jedem Fall kleiner gleich 4 ist, nämlich 2 Lösungen aus der quadratischen Gleichung und evtl. noch 2 aus dem nicht betrachteten Fall $x \in \mathbb{F}_2$. Betrachten wir also jetzt $x \in \mathbb{F}_2$, so ergibt sich beim Einsetzen in die Gleichung 1.1 für d immer 1:

$$(0+1)^{-1} = 1$$
$$1^{-1} + (1+1)^{-1} = 1$$

Zum Bestimmen der Uniformität müssen wir untersuchen, ob die Gleichung

$$x^2 + x + 1 = 0$$

noch weitere Lösungen außer 0 und 1 haben kann. Multiplizieren wir diese Gleichung mit $x(x+1)$ ergibt sich

$$x^4 + x = 0$$

mit Lösungen ungleich 0 und 1 genau dann, wenn der Körper $\mathbb{F}_4$ ein Teil-Körper von $\mathbb{F}_{2^n}$ ist, also wenn n gerade ist. Das beweist die Behauptung. □
Damit hat die S-Box des AES hat die kleinste bekannte Uniformität für Permutationen. Für jede Eingabedifferenz $\Delta I \in \mathbb{F}_{2^8}$ und jede Ausgabedifferenz $\Delta O \in \mathbb{F}_{2^8}$ gilt:

$$\mathcal{P}(\Delta I \to \Delta O | S) \leq \frac{\text{Uniformität(S)}}{2^8} = \frac{4}{2^8} = \frac{1}{64}$$

Es wird auch deutlich, dass die Wahrscheinlichkeit mit größeren S-Boxen kleiner wird. 8-Bit-S-Boxen eignen sich jedoch besonders, da sie auf heutigen Prozessoren gut implementierbar sind.

Auch für den zweiten wichtigen Ansatz, dem Maximieren der aktiven S-Boxen, sind für den AES sehr schöne mathematische Formulierungen gefunden worden. Siehe dazu Übung **??**. Wir stellen hier nur fest, dass beim AES die MixColumns-Transformation garantiert, dass in 2 aufeinander folgenden Runden mindestens 5 S-Boxen aktiv sind. Weiter kann man zeigen, dass in 4 aufeinander folgenden Runden mindestens 25 S-Boxen aktiv sein müssen.

1.5 Lineare Attacke

Es wurde bereits in der Grundlagenvorlesung „Einführung in die Datensicherheit und Kryptographie“ gezeigt, dass lineare Funktionen kryptographisch

schwach sind. Aus diesem Gedanken hat Mitsuru Matsui die „lineare Kryptanalyse“ bzw. die „lineare Attacke“ entwickelt. Bei einer linearen Attacke versucht man, eine Blockchiffre durch eine lineare Funktion zu approximieren.

1.5.1 Lineare Abbildungen

Zur Erinnerung:

Definition 11 *Eine Abbildung $\varphi : \mathbb{F}_2^n \to \mathbb{F}_2^m$ heißt* linear*, wenn folgende Eigenschaften erfüllt sind*

$$\begin{aligned} \varphi(0) &= 0 \\ \varphi(x+y) &= \varphi(x) + \varphi(y) \quad \forall\ x, y \in \mathbb{F}_2^n \\ \varphi(\lambda x) &= \lambda\varphi(x) \quad \forall\ x \in \mathbb{F}_2^n,\ \lambda \in \mathbb{F}_2 \end{aligned}$$

Jede lineare Abbildung $\varphi : \mathbb{F}_2^n \to \mathbb{F}_2^m$ kann als $m \times n$-Matrix beschrieben werden. Es gibt also eine $m \times n$-Matrix A, so dass $\varphi(x) = Ax$ gilt. Alle Einträge der Matrix A stammen aus $\mathbb{F}_2$.

Insbesondere lassen sich lineare Abbildungen $\varphi : \mathbb{F}_2^n \to \mathbb{F}_2$ durch $1 \times n$-Matrizen (eine Zeile, n Spalten) darstellen, es gibt also einen Vektor $(a_0, \ldots, a_{n-1}) \in F_2^n$ so dass für alle $x \in \mathbb{F}_2^n$

$$\varphi(x) = \begin{pmatrix} a_1 & \cdots & a_{n-1} \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix} = \sum_{i=0}^{n-1} a_i x_i$$

gilt. Das bedeutet, dass sich lineare Abbildungen für $\mathbb{F}_2^n \to \mathbb{F}_2$ als Skalaprodukt darstellen lassen:

$$\varphi(x) = \langle a, x \rangle, \qquad a = \begin{pmatrix} a_1 & \cdots & a_{n-1} \end{pmatrix}, x = \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

Wir bezeichnen diese Funktion mit φ_a. Affine Abbildungen für $\varphi' : \mathbb{F}_2^n \to \mathbb{F}_2$ lassen sich analog durch

$$\varphi'(x) = \langle a, x \rangle + c, \qquad a \in \mathbb{F}_2^n,\ c \in \mathbb{F}_2$$

beschreiben.

1.5.2 Approximation durch eine lineare Funktion

Sei $f : \mathbb{F}_2^n \to \mathbb{F}_2$ eine beliebige Funktion. Wir wollen diese Funktion durch lineare (bzw. affine) Funktionen approximieren.

Definition 12 *Eine Funktion f wird durch die lineare Funktion $\varphi : \mathbb{F}_2^n \to \mathbb{F}_2$ mit der Wahrscheinlichkeit $\mathcal{P}_{approx}$ approximiert, wenn gilt*

$$\mathcal{P}_{approx} = \frac{|\{x \in \mathbb{F}_2^n \ |\varphi(x) = f(x)\}|}{2^n}$$

Für eine gegebene Funktion f und eine lineare Funktion φ zählen wir demnach einfach, für wieviele Eingaben beide Funktionen übereinstimmen. Eine gute Möglichkeit diese Wahrscheinlichkeiten auszurechnen bietet die im nächsten Abschnitt vorgestellte Walsh-Transformation.

1.5.3 Die Walsh-Transformation

Um eine Funktion f zu approximieren, kann man die „Walsh-Transformation“ verwenden.

Definition 13 *Für eine Bool'sche Funktion $f : \mathbb{F}_2^n \to \mathbb{F}_2$ bezeichnet für ein $a \in \mathbb{F}_2^n$*

$$f^W(a) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\langle a,x\rangle}$$

den Walsh-Koeffizienten *an der Stelle a*

Wenn man den Abstand von f zu f' mit $d(f, f') = |\{x|f(x) \neq f'(x)\}|$ für zwei Funktionen $f, f' : \mathbb{F}_2^n \to \mathbb{F}_2$ misst, ergibt sich:

$$\begin{aligned} f^W(a) &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\langle a,x\rangle} \\ &= |\{x|f(x) = \langle a, x\rangle\}| - |\{x|f(x) \neq \langle a, x\rangle\}| \\ &= (2^n - d(f, \varphi_a)) - d(f, \varphi_a) \\ &= 2^n - 2d(f, \varphi_a) \end{aligned}$$

Der Walsh-Koeffizient an der Stelle a gibt also an, wie gut sich f durch die lineare Funktion φ_a approximieren lässt. Die Wahrscheinlichkeit, mit der φ_a die Funktion f approximiert ist:

$$\begin{aligned} \mathcal{P}_{approx} &= \frac{2^n - d(f, \varphi_a)}{2^n} \\ &= 1 - \frac{2^n - f^W(a)}{2^{n+1}} \\ &= \frac{1}{2} + \frac{f^W(a)}{2^{n+1}} \end{aligned}$$

1.5.4 Berechnung der Walsh-Koeffizienten

Für die Berechnung der Walsh-Koeffizienten kann man (naiver Weise) einfach die Summe $\sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\langle a,x \rangle}$ durch Einsetzen ausrechnen. Der Aufwand beträgt 2^n für einen Walsh-Koeffizienten. Für die Berechnung aller Walsh-Koeffizienten einer Funktion ergibt sich also der Aufwand $2^n * 2^n = 2^{2n}$.

Einen einfacheren Ansatz bietet die „Fast-Fourier-Transformation“ (FFT). Wir werden an einem Beispiel die FFT-Methode zeigen, mit der man sehr effizient alle Walsh-Koeffizienten einer Funktion berechnen kann. Danach werden wir noch kurz skizzieren, warum die FFT-Methode funktioniert. Eine ausführliche Begründung würde den Umfang der Vorlesung überschreiten.

		FFT:	0:	1:	2:
a	f(x)	$(-1)^{f(x)}$			$f^W(a)$
000	1	-1	0	0	-2
001	0	1	-2	0	2
010	0	1	0	0	2
011	1	-1	2	-4	-2
100	0	1	0	-2	2
101	1	-1	2	2	-2
110	1	-1	-2	2	-2
111	1	-1	0	2	-6
		Abstand:	1	2	4

Abbildung 1.21: Beispiel für die Berechnung der Walsh-Koeffizienten mit der FFT-Methode.

Für den Vektor $a \in \mathbb{F}_2^n$ werden n Schritte benötigt. Die Tabelle wird in jedem Schritt von oben nach unten gefüllt. Die neuen Werte im Schritt j ($\forall j,\ 0 \leq j \leq n-1$) werden kreuzweise aus den Werten des letzten Schrittes berechnet, wobei der Abstand zwischen den gepaarten Werten $d = 2^j$ beträgt. Für die obere Position werden beide Werte addiert, für die untere Position subtrahiert. Im Beispiel ergibt sich im Schritt $j = 1$ (also $d = 2^1 = 2$) für die zweite Position $-2_{pos=2} + 2_{pos=4} = 0_{pos=2}$ und für die vierte Position $-2_{pos=2} - 2_{pos=4} = -4_{pos=4}$.

1.5.5 Die Linearität einer Funktion

Je größer der Walsh-Koeffizient ist, umso besser approximiert die lineare Funktion $\varphi(x) = \langle a, x \rangle$ die Funktion f. Je kleiner der Walsh-Koeffizient ist, umso besser approximiert die affine Funktion $\varphi'(x) = \langle a, x \rangle + 1$ die Funktion f. Aus den Überlegungen zu dem Walsh-Koeffizienten kommt man zu der Definition der Linearität:

Definition 14 *Die Linearität einer beliebigen Funktion f sei:*

$$\mathrm{Lin}(f) := \max_{a \in \mathbb{F}_2^n} |f^W(a)|$$

Das Ziel bei dem Design einer Blockchiffre ist es also, Funktionen mit einer möglichst kleinen Linearität zu finden. Für eine ideale, nichtlineare Funktion

f müsste also $f^W(a) = 0 \quad \forall a \in \mathbb{F}_2^n$ gelten. Der folgende Satz (Satz von Parseval) zeigt aber, dass es keine Funktion gibt, die dieses Ideal erfüllt:

Theorem 15 *Sei $f : \mathbb{F}_2^n \to \mathbb{F}_2$ gegeben, dann gilt:*

$$\sum_{a \in \mathbb{F}_2^n} f^W(a)^2 = 2^{2n}$$

Für den Beweis benötigen wir das folgende Lemma:

Lemma 16 *Sei $a \in \mathbb{F}_2^n$ dann gilt:*

$$\sum_{x \in \mathbb{F}_2^n} (-1)^{\langle a,x \rangle} = \begin{cases} 0 & \forall a \neq 0 \\ 2^n & a = 0 \end{cases}$$

Beweis siehe Übung

Beweis von Satz 15: Es gilt

$$\begin{aligned} \sum_{a \in \mathbb{F}_2^n} f^W(a)^2 &= \sum_{a \in \mathbb{F}_2^n} \left(\sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\langle a,x \rangle} \right)^2 \\ &= \sum_{a \in \mathbb{F}_2^n} \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)+\langle a,y \rangle} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\langle a,x \rangle} \\ &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)} \sum_{a \in \mathbb{F}_2^n} (-1)^{\langle a,y \rangle + \langle a,x \rangle} \\ &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)} \sum_{a \in \mathbb{F}_2^n} (-1)^{\langle a,x+y \rangle} \end{aligned}$$

Mit Lemma 16 ergibt sich

$$\begin{aligned} \sum_{a \in \mathbb{F}_2^n} f^W(a)^2 &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} \sum_{y+x=0} 2^n (-1)^{f(y)} \\ &= 2^n \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+f(x)} \\ &= 2^{2n} \end{aligned}$$

□

1.5.6 Bent-Funktionen

Die besten nichtlinearen Funktionen haben also für alle Stellen a den gleichen Walsh-Koeffizienten $f^W(a) = \pm 2^{\frac{n}{2}}$ und somit die Linearität $Lin(f) = 2^{\frac{n}{2}}$. Diese Funktionen werden als *Bent-Funktionen* (gekrümmte Funktionen) bezeichnet. Da der Walsh-Koeffizient immer eine natürliche Zahl ist ($f^W(a) \in \mathbb{Z}$), muss n bei Bent-Funktionen also immer gerade sein.

Im Folgenden wird gezeigt, dass die sogenannten *Maiorana-McFarland Funktionen* Bent-Funktionen sind.

Theorem 17 *Sei $\pi : \mathbb{F}_2^m \to \mathbb{F}_2^m$ eine Permutation und $h : \mathbb{F}_2^m \to \mathbb{F}_2$ eine beliebige Funktion, dann sind die „Maiorana-McFarland Funktionen“ $f : \mathbb{F}_2^m \times \mathbb{F}_2^m \to \mathbb{F}_2$ mit*

$$f(x, y) = \langle x, \pi(y) \rangle + h(y)$$

Bent-Funktionen.

Beweis: Seien $a, b \in \mathbb{F}_2^m$ gegeben. Dann gilt:

$$\begin{aligned}
f^W(a,b) &= \sum_{x,y \in \mathbb{F}_2^m} (-1)^{f(x,y)+\langle a,x\rangle+\langle b,y\rangle} \\
&= \sum_{x,y \in \mathbb{F}_2^m} (-1)^{\langle x,\pi(y)\rangle+h(y)+\langle a,x\rangle+\langle b,y\rangle} \\
&= \sum_{y \in \mathbb{F}_2^m} \left((-1)^{h(y)+\langle b,y\rangle} * \sum_{x \in \mathbb{F}_2^m} (-1)^{\langle x,\pi(y)\rangle+\langle a,x\rangle} \right) \\
&= \sum_{y \in \mathbb{F}_2^m} \left((-1)^{h(y)+\langle b,y\rangle} * \sum_{x \in \mathbb{F}_2^m} (-1)^{\langle \pi(y),x\rangle+\langle a,x\rangle} \right) \\
&= \sum_{y \in \mathbb{F}_2^m} \left((-1)^{h(y)+\langle b,y\rangle} * \sum_{x \in \mathbb{F}_2^m} (-1)^{\langle \pi(y)+a,x\rangle} \right)
\end{aligned}$$

Für die nächsten Umformungen nutzen wir, dass $\pi(y)$ bijektiv ist, dass also nur ein y existiert, so dass $\pi(y) = a$

$$\begin{aligned} f^W(a,b) &= \sum_{y,\ \pi(y)=a} \left((-1)^{h(y)+\langle b,y\rangle} * 2^m \right) \\ &= \sum_{y=\pi^{-1}(a)} \left((-1)^{h(y)+\langle b,y\rangle} * 2^m \right) \\ &= 2^m \underbrace{(-1)^{h\left(\pi^{-1}(a)\right)+\langle b,\pi^{-1}(a)\rangle}}_{\pm 1} \\ &= \pm 2^m \end{aligned}$$

Da $\mathbb{F}_2^m \times \mathbb{F}_2^m = \mathbb{F}_2^{2m} = \mathbb{F}_2^n \Rightarrow \frac{n}{2} = m$ gilt, ist damit das Theorem 17 bewiesen.

□

1.5.7 Die Lineare Attacke

Die lineare Attacke soll an einer einfachen Chiffre demonstriert werden, in der die Nachricht $m \in \mathbb{F}_2^n$ mit dem Schlüssel $k \in \mathbb{F}_2^n$ XOR-verknüpft und dann in einer S-Box $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ verschlüsselt wird. Das Chiffrat $c \in \mathbb{F}_2^m$ ist die Ausgabe der S-Box. Das Ziel der linearen Attacke ist es, die S-Box durch eine lineare Funktion zu approximieren.
Dabei ergibt sich ein Problem. Da $S(x) \in \mathbb{F}_2^m$ liegt, der Walsh-Koeffizient aber für eine Funktion $f : \mathbb{F}_2^n \to \mathbb{F}_2$ definiert wurde, können wir nicht S approximieren.

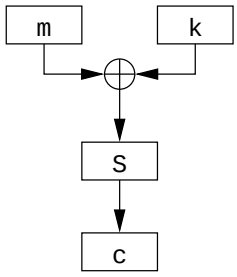

Abbildung 1.22: Die Chiffre für die Demonstration

Wir müssen daher für ein $b \in \mathbb{F}_2^m$ mit $S_b(x) := \langle b, S(x)\rangle = \sum_{i=0}^{m-1} b_i S_i(x)$ approximieren, wobei $b = \begin{pmatrix} b_0 \\ \vdots \\ b_{m-1} \end{pmatrix}$ und $S(x) = \begin{pmatrix} S_0(x) \\ \vdots \\ S_{m-1}(x) \end{pmatrix}$ sind. Für $b = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ entspricht $S_b(x)$ dem ersten Bit von $S(x)$.

Um eine möglichst gute Approximation zu finden, bestimmt man:

$$\max_{b\neq 0,\ b\in\mathbb{F}_2^m} \left(\max_{a\in\mathbb{F}_2^n} |S_b^W(a)| \right) = \max_{b\neq 0} (Lin(S_b))$$

Für das Paar (a, b), so dass $S_b^W(a)$ betragsmäßig maximal ist, ist die optimale Approximation von $S_b : \mathbb{F}_2^m \to \mathbb{F}_2$ durch

$$\langle a, x\rangle = \langle b, S(x)\rangle + d \begin{cases} d = 0 \text{ wenn, } S_b^W(a) > 0 \\ d = 1 \text{ wenn, } S_b^W(a) < 0 \end{cases}$$

gegeben.
Das Maximum $|S_b^W(a)|$ ist immer ungleich Null, da ansonsten $Lin(S_b) = 0$ wäre und das ist, wie in Theorem 15 gezeigt, nicht möglich.

Für unser einfaches Beispiel gilt also

$$\langle a, m + k\rangle = \langle b, c\rangle + d$$

für viele Eingaben $(m + k)$ und Ausgaben c.
Aus der Linearität des Vektorprodukts folgt:

$$\langle a, k\rangle = \langle b, c\rangle + \langle a, m\rangle + d \quad (*)$$

ist für viele Nachrichten $m \in \mathbb{F}_2^n$ und Ciphertexte $c \in \mathbb{F}_2^m$ erfüllt.

Für die Attacke verschlüsselt man nun mehrere Klartexte m und bestimmt mit den entsprechenden Ciphertexten c die rechte Seite der Gleichung $(*)$.

Sei N die Anzahl der Paare (m, c), so dass die rechte Seite der Gleichung $(*)$ 0 ergibt und M die Gesamtzahl der Paare (m, c).

Wenn $N > \frac{M}{2}$ dann wird $\langle a, k\rangle = 0$ gelten.
Wenn $N < \frac{M}{2}$ dann wird $\langle a, k\rangle = 1$ gelten.

Die Erfolgswahrscheinlichkeit ist also abhängig von M und $|S_b^W(a)|$.

Um weitere Informationen über den Schlüssel zu bekommen, kann man weitere $a_1, \ldots, a_t \in \mathbb{F}_2^n$ anstatt des ursprünglichen $a \in \mathbb{F}_2^n$ wählen und erhält so für jedes $a_i \in \mathbb{F}_2^n$ eine **lineare Gleichung** der Schlüsselbits.

Um den gesamten Schlüssel zu bestimmten muss man also ein lineares Gleichungssystem lösen. Dazu sind n **linear unabhängige** Vektoren a_i nötig. Anders ausgedrückt kann man sagen, dass die a_i eine Basis von $\mathbb{F}_2^n$ bilden müssen.

1.5.8 Lineare Attacke auf DES

Der einzige Schutz gegen die lineare Attacke sind die S-Boxen für die es zahlreiche Designkriterien nach Coppersmith [Coppersmith 94] gibt. Eine davon ist folgende:

S-2: Keines der Ausgabebits darf zu dicht an einer linearen Funktion über die Eingabebits liegen.

Dieses Kriterium entspricht aber nur den Werten $b \in F_2^4$ mit:
$b \in \{(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1)\}$,
d.h. für andere $b \in F_2^4$ sind die S-Boxen nicht gegen die lineare Attacke optimiert. Besser wäre folgendes Kriterium:

S-2': Keine Linearkombination der Ausgabebits darf gut durch lineare Funktionen approximiert werden.

Für DES mit 16 Runden ist die lineare Attacke die beste Attacke, die man kennt. Man benötigt ca. 2^{47} Ciphertext/Plaintext-Paare.

1.5.9 Lineare Attacke auf AES

Auch hier stellt die S-Box den einzigen Schutz gegen die lineare Attacke dar. Es gilt:

Theorem 18 *Sei $n \in \mathbb{N}$ und*

$$S: \ F_{2^n} \to F_{2^n}$$
$$x \to x^{-1}$$

dann gilt für alle a,b $\in F_{2^n}$:

$$|S_b^w(a)| \leq 2^{\frac{n}{2}+1}$$

Das bedeutet, dass der Walsh-Koeffizient betragsmäßig immer unter $2^{\frac{n}{2}+1}$ liegt. Somit läßt sich die S-Box des AES nicht gut approximieren und hält damit momentan den Weltrekord.Bevor wir diesen Satz bewiesen, benötigen wir einige Vorbereitungen und Definitionen.

Für die weitere Betrachtung der linearen Attacke gegen AES benötigen wir die Definition der Spur-Funktion:

Definition 19 *Für $n \in \mathbb{N}$ heißt die Funktion*

$$\mathrm{Tr}: \ F_{2^n} \to F_{2^n}$$

mit

$$\mathrm{Tr}(x) := \sum_{i=0}^{n-1} x^{2^i}$$

Spur Funktion

Beispiel: Für F_{2^4} ist $\mathrm{Tr}(x) = x + x^2 + x^4 + x^8$

Der folgende Satz enthält die wichtigsten Eigenschaften der Spur Funktion

Theorem 20 *Die Abbildung $x \to \mathrm{Tr}(x)$ erfüllt folgende Eigenschaften.*

$\mathrm{Tr}(x+y) = \mathrm{Tr}(x) + \mathrm{Tr}(y)$ *(Linearität)*

$\mathrm{Tr}(x^2) = \mathrm{Tr}(x)$

$\mathrm{Tr}(x) \in F_2$

$\mathrm{Tr}(x) = 0 \Leftrightarrow \exists y$ *so dass* $x = y^2 + y$

Beweis (siehe Übung)

Insbesondere ist $\langle x, y\rangle = \mathrm{Tr}(xy)$ $x, y \in F_{2^n}$ ein **Skalarprodukt**, d.h. $(x, y) \to \mathrm{Tr}(xy)$ erfüllt

$\mathrm{Tr}((x+y)z) = \mathrm{Tr}(xz) + \mathrm{Tr}(yz)$ (bilinear)

$\mathrm{Tr}(xy) = \mathrm{Tr}(yx)$ (symmetrisch)

$\mathrm{Tr}(xy) \in F_2$

Also gilt für $S : F_{2^n} \to F_{2^n}$

$$S_b^w(a) = \sum_{x \in F_{2^n}} (-1)^{\mathrm{Tr}(b \cdot S(x)) + \mathrm{Tr}(a \cdot x)} \quad a, b \in F_{2^n}$$

Lemma 21 *Für $S(x) = x^{-1}$ gilt*

$$S_b^w(a) = S_1(ab) \quad \forall a, \forall b \neq 0$$

Beweis:

$$\begin{aligned} S_b^w(a) &= \sum_{x \in F_{2^n}} (-1)^{\mathrm{Tr}(bS(x)) + ax)} \\ &= \sum_{x \in F_{2^n}} (-1)^{\mathrm{Tr}(bx^{-1} + ax)} \\ &= \sum_{x \in F_{2^n}} (-1)^{\mathrm{Tr}((b^{-1}x)^{-1} + ax)} \end{aligned}$$

da $b \neq 0$ gilt können wir x durch bx ersetzen:

$$\begin{aligned} &= \sum_{x \in F_{2^n}} (-1)^{\mathrm{Tr}(x^{-1}+(ab)x)} \\ &= S_1^w(ab) \end{aligned}$$

□

Beweis von Satz 18 Es gilt:

$$S_1^w(a) = \sum_{x \in F_{2^n}} (-1)^{\mathrm{Tr}(x^{-1}+ax)}$$

und nach Lemma 21 reicht es für den Beweis diese Werte für alle a abzuschätzen. Es gilt

$$\begin{aligned} S_1^w(a) &= \#\{x|\,\mathrm{Tr}(x^{-1}+ax)=0\} - \#\{x|\,\mathrm{Tr}(x^{-1}+ax)=1\} \\ &= 2 \cdot \#\{x|\,\mathrm{Tr}(x^{-1}+ax=0\} - 2^n \end{aligned}$$

Aus der vierten Eigenschaft der Spur Funktion folgt

$$\{x|\,\mathrm{Tr}(x^{-1}+ax)=0\} = \{x|\exists y \in F_{2^n}\ s.d.\ y^2+y = x^{-1}+ax\}$$

Es müssen also Lösungen $(x,y) \in F_{2^n} x F_{2^n}$ der Gleichung $y^2+y = x^{-1}+ax$ bestimmt werden. Die Anzahl dieser Lösungen bezeichnen wir mit N und es gilt: $S_1^w(a) = N - 2^m$, da es für jedes Element $x^{-1}+ax$ mit Spur 0 zwei Elemente $y, y+1 \in F_{2^n}$ gibt, so dass $y^2+y = x^{-1}+ax$ gilt.

Für $x \neq 0$ läßt sich diese Gleichung folgendermaßen umformen:

$$\begin{aligned} y^2+y &= x^{-1}+ax \\ \Leftrightarrow y^2x^2+yx^2 &= x+ax^3 \\ \Leftrightarrow (yx)^2+(yx)x &= x+ax^3 \end{aligned}$$

Da uns nur die Anzahl der Lösungen und nicht die Lösungen selbst interessieren können wir y^x durch y ersetzen:

$$\Rightarrow y^2+yx = x+ax^3$$

Diese Gleichung stellt eine elliptische Kurve in F_{2^n} dar. Es muß also die Anzahl der Punkte dieser Kurve abgeschätzt werden.

Nach dem Satz von Hasse gilt:

$$|N - 2^n| \leq 2 \cdot 2^{\frac{n}{2}} + 1$$

$$\Rightarrow |S_1^w(a)| \leq 2^{\frac{n}{2}+1} + 1$$

Da $S_1^w(n)$ immer gerade ist folgt:

$$|S_1^w(a)| \leq 2^{\frac{n}{2}+1}$$

□

Kapitel 2

Der Comp-128

2.1 Comp-128

Comp-128 wurde (und wird eventuell vereinzelt auch noch heute) in der Mobiltelefonie zur Authentifikation gegenüber den Basisstationen und zum Erzeugen von Sitzungsschlüsseln verwendet.

2.1.1 Ablauf der Anmeldung

Abbildung 2.1 bietet eine Übersicht über das Authentifikationsprotokoll.

Das Mobiltelefon sendet zuerst die eindeutige Kennung (IMSI) an die Basisstation welche daraufhin einen 128 Bit langen Zufallswert $RAND$ erzeugt und an das Mobiltelefon sendet.

Die SIM-Karte besitzt einen geheimen Schlüssel K_i, der sonst nur noch dem Mobilfunkanbieter bekannt ist. Mit Hilfe dieses Schlüssels K_i und dem von der Basisstation erhaltenen Zufallswert $RAND$ berechnet die SIM-Karte einen 128-Bit Wert. Die ersten 32 Bit dieses Wertes bilden die Antwort $RESP$ welche an die Basisstation zurück gesendet wird.

Die Basisstation führt die gleiche Berechnung wie die SIM-Karte durch und vergleicht anschließend, ob die empfangene Antwort des Mobiltelefons mit der selbst berechneten übereinstimmt. Ist dies der Fall, hat sich das Mobiltelefon erfolgreich authentifiziert.

2.1.2 Schema des Comp-128

Als Eingabe erhält der Comp-128 den geheimen Schlüssel K_i und den zufälligen Wert $RAND$. Zunächst werden diese beiden Werte zu einem einzigen 32

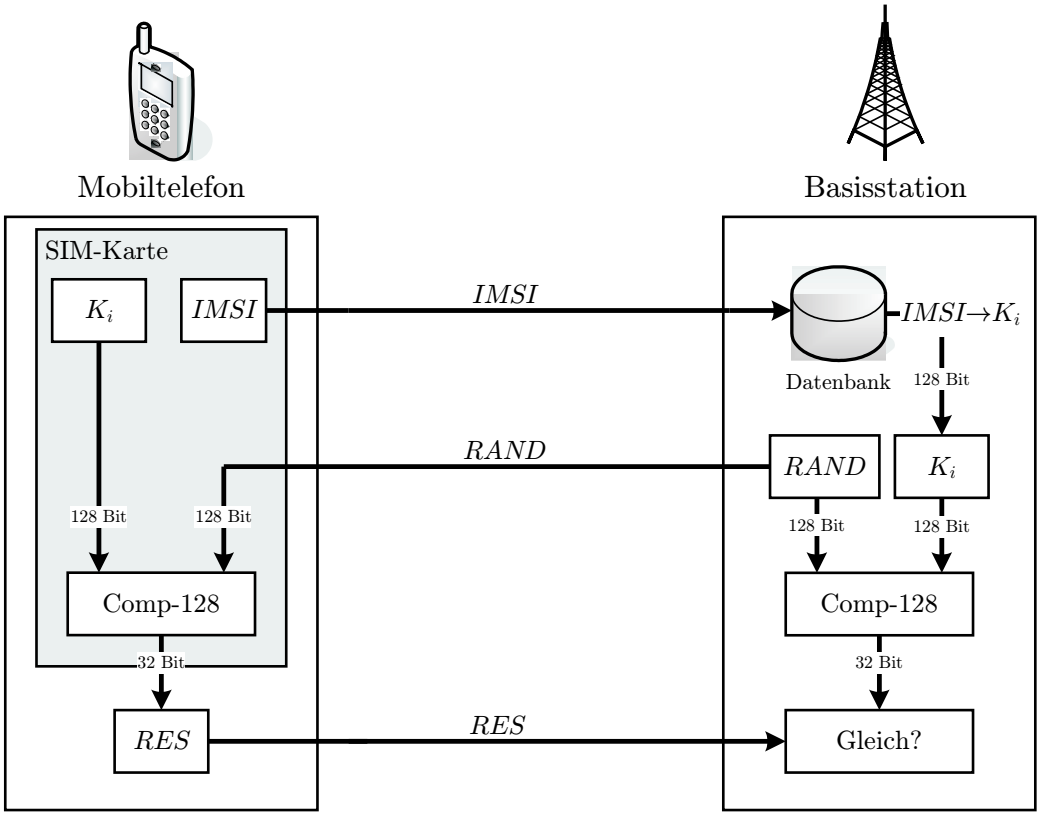

Abbildung 2.1: Ablauf der Authentifikation eines Mobiltelefons gegenüber einer Basisstation

Byte großen Wert konkateniert.

$$RAND, K_i \rightarrow K_i||RAND$$

Dieser Wert dient als Eingabe für eine Hashfunktion, welche einen 16 Byte großen Wert x berechnet. Dieser wird im Anschluss permutiert und wieder mit K_i konkateniert und als Eingabe für die Hashfunktion in der nächsten Runde verwendet.

Nach acht Runden wird der 16 Byte Ausgabewert der Hashfunktion als Ergebnis des Comp-128 ausgegeben.

Das Schema des Comp-128 wird in Abbildung 2.2 dargestellt.

Hierbei ist anzumerken, dass der Zufallswert *RAND* lediglich in der ersten Runde in die Berechnung eingeht. Aufgrund dieser Tatsache wird der im Folgenden beschriebene Angriff möglich.

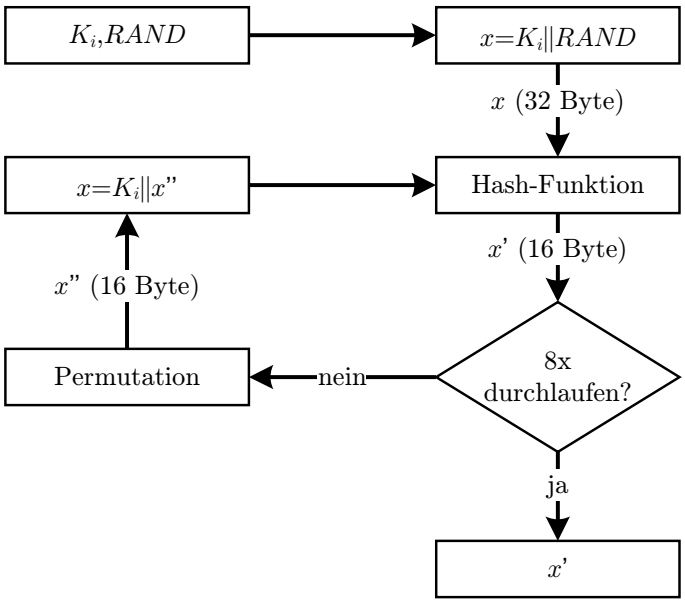

Abbildung 2.2: Schema das Comp-128

Die Hashfunktion des Comp-128

Die Hashfunktion besteht aus fünf Schichten. In jeder Schicht wird jeweils ein Byte-Paar auf ein anderes Byte-Paar abgebildet. Für jede Schicht gibt es eine nicht-lineare Abbildung:

$$S_i : \{0, \ldots, 2^{9-i} - 1\} \to \{0, \ldots, 2^{8-i} - 1\} \quad \text{mit} \quad i \in \{0, \ldots, 4\}$$

Diese Abbildungen S_i werden als Lookup-Tabellen implementiert.

Ein Wertepaar (a, b) wird abgebildet auf (a', b') wobei

$$\begin{aligned} a' &= S_i(a + 2b \bmod 2^{9-i}) \quad \text{und} \\ b' &= S_i(2a + b \bmod 2^{9-i}) \end{aligned}$$

Die Indizes, die angeben welche Paare miteinander verknüpft werden, sind in einer Art "FFT-Schema" organisiert (s. Abbildung 2.3):

- In Schicht 0 werden die Bytes i und $i + 16$ miteinander verknüpft (für $i \in \{0, \ldots, 15\}$).
- In Schicht 1 werden die Bytes i und $i + 8$ miteinander verknüpft (für $i \in \{0, \ldots, 7, 16, \ldots, 23\}$).
- In Schicht 2 werden die Bytes i und $i + 4$ miteinander verknüpft (für $i \in \{0, \ldots, 3, 8, \ldots, 11, 16, \ldots, 19, 24, \ldots, 27\}$). usw.

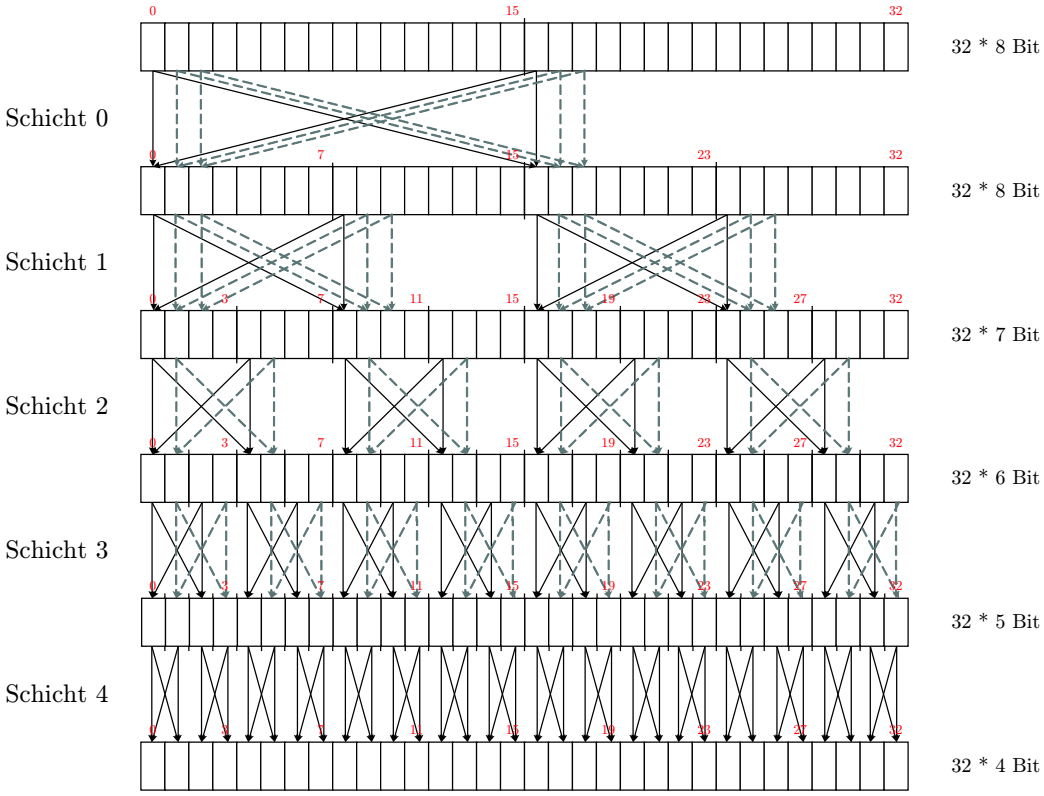

Abbildung 2.3: Schema der Comp-128 Hashfunktion

Permutation

Diese Abbildung bildet 128-Bit Werte auf 128-Bit Werte ab, indem sie das i-te Bit der Eingabe auf das $(17 \cdot i \bmod 128)$-te Bit des Ausgabewertes abbildet. Diese Abbildung ist bijektiv, da $\mathrm{ggT}(17, 128) = 1$ und somit eine Zahl $x \in \{0, \ldots, 127\}$ mit $17 \cdot x = 1 \bmod 128$ existiert. Die Abbildung, die das i-te Bit auf das $(xi \bmod 128)$-te Bit abbildet ist somit die Umkehrfunktion für unsere Abbildung.

2.1.3 Angriff auf Comp-128

Idee: Ziel ist es, eine Kollision in der *zweiten Schicht* der Hashfunktion in der *ersten Runde* von Comp-128 zu erzeugen. Eine solche Kollision führt zu einer Kollision der Ausgabewerte, da der Wert *RAND* nur in der ersten Runde in die Berechnung eingeht und somit alle weiteren Berechnungen von diesem unabhängig sind.

Um eine solche Kollision zu erzeugen wählt man

$$x = (x_{15}, \ldots, x_0) \quad \text{mit} \quad x_i \,\text{Bytes}$$

für $j \in \{0, \ldots, 7\}$, so dass

$$x_i = 0 \quad \text{für} \quad i \neq j \quad \text{und} \quad i \neq j + 8$$

und x_j und x_{j+8} durchlaufen alle 2^{16} möglichen Werte.

Für jeden dieser 2^{16} Werte berechnet man Comp-128(x, K_i) und speichert das entsprechende Ergebnis in einer Tabelle ab. Wenn es zwei Werte x und x' mit $x \neq x'$ gibt, so dass

$$\text{Comp-128}(x, K_i) = \text{Comp-128}(x', K_i)$$

gilt, nutzt man diese Werte um Kandidaten für die Schlüsselbytes $K_i^{(j)}$ und $K_i^{(j+8)}$ zu bestimmen.

$$K_i = (K_i^{(15)}, \ldots, K_i^{(0)}) \quad \text{und} \quad K_i^{(j)} \,\text{Bytes}$$

Berechnung der Schlüsselkandidaten

Tritt eine Kollision in der ersten Runde auf, dann muss für den richtigen Schlüssel gelten, dass die vier 7-Bit-Werte y_i, y_{i+8}, y_{i+16} und y_{i+24} in der zweiten Schicht für x und x' gleich sind.

Für alle Schlüssel K_i mit

$$K_i^{(a)} = 0 \quad \text{für} \quad a \neq j \quad \text{und} \quad a \neq j + 8$$

berechnet man diese vier 7-Bit-Werte für x und x'. Sind diese Werte gleich, dann wurde ein Schlüsselkandidat gefunden. Wenn nicht, dann stimmen diese Schlüsselbytes nicht.

In der Praxis findet man in der Regel nur ein Kandidatenpaar $(K_i^{(j)}, K_i^{(j+8)})$.

Um zu überprüfen, ob die Kollision tatsächlich in der zweiten Schicht der ersten Runde aufgetreten ist, wählt man

$$z = (z_{15}, \ldots, z_0) \quad \text{mit} \quad z_i = c \quad \text{für} \quad i \neq j \quad \text{und} \quad i \neq j + 8$$

mit einem festen 8-Bit-Wert für c und

$$z_j = x_j \quad \text{und} \quad z_{j+8} = x_{j+8}.$$

Analog wählt man z' für x'.

Wenn gilt, dass

$$\text{Comp-128}(z, K_i) \neq \text{Comp-128}(z', K_i)$$

dann trat die Kollision *nicht* in der zweiten Schicht der ersten Runde auf.

2.1.4 Auftrittswahrscheinlichkeit einer Kollision in der ersten Runde

Um die Wahrscheinlichkeit einer Kollision in der zweiten Schicht der Hashfunktion in der ersten Runde zu bestimmen, nehmen wir an, dass die Abbildung

$$f : {\mathbb{F}_{2^8}}^2 \rightarrow {\mathbb{F}_{2^7}}^4,$$

die x_j und x_{j+8} auf y_j, y_{j+8}, y_{j+16} bzw. y_{j+24} abbildet, eine zufällige Abbildung f ist. Die Wahrscheinlichkeit, dass es bei einer zufälligen Abbildung *mindestens* eine Kollision gibt, ist nach der Formel für das Geburtstagsparadoxon

$$P \geq 1 - e^{-\frac{k(k-1)}{2n}}$$

wobei $k = 2^{16}$ und $n = 2^{28}$. Folglich gilt für die Wahrscheinlichkeit

$$P \geq 1 - e^{-\frac{2^{16}(2^{16}-1)}{2^{29}}} \approx 0,9997$$

Der Erwartungswert, der angibt, nach wievielen Versuchen im Schnitt die erste Kollision auftritt, ist

$$E = \sqrt{\pi \frac{n}{2}} \approx 20538$$

Das bedeutet, dass für alle acht möglichen Werte von j im Durchschnitt

$$8 \cdot E \approx 164303$$

Berechnungen des Comp-128 auf der SIM-Karte notwendig sind. Unter der Annahme, dass eine SIM-Karte ca. sieben Werte pro Sekunde berechnen kann, ergibt sich eine Laufzeit von etwa 6,5 Stunden.

Kapitel 3

Stream Ciphers

3.1 Einführung

Eine Stromchiffre (Stream Cipher) verschlüsselt einen Datenstrom bitweise. Dazu wird ein Schlüsselstrom erzeugt und jedes Datenbit mit einem Schlüsselbit verknüpft. Diese Verknüpfung ist in den meisten Fällen ein simples XOR.

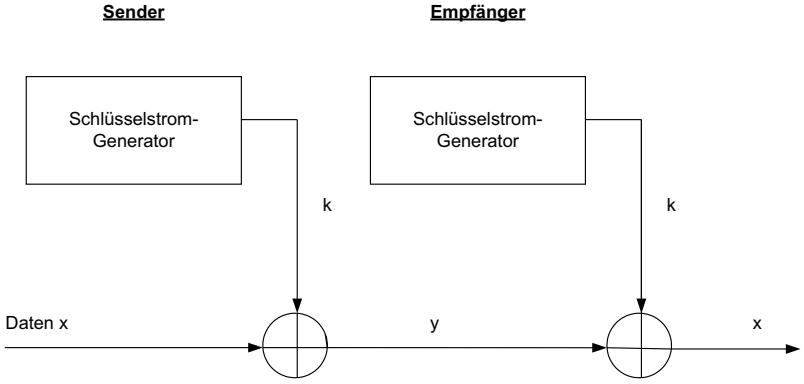

Abbildung 3.1: Schema eines Stream-Cipher

Man kann sich also eine Stromchiffre als Versuch, das *One-Time-Pad* (OTP) effizient zu machen, vorstellen. Das OTP ist eine spezielle Stromchiffre, bei der ein **echt zufälliger** Schlüsselstrom erzeugt wird. Noch eine Bemerkung am Rande: Eine Stromchiffre ist keine Blockchiffre mit der Blocklänge 1, da eine Stromchiffre einen veränderlichen, internen Zustand besitzt. Man kann sich eine Stromchiffre auch nicht als Blockchiffre mit Blocklänge 1 im CBC Modus (oder ähnlichen Betriebsmodi) vorstellen, da bei diesen der Zustand von den verschlüsselten Daten abhängt, was bei den Stromchiffren die wir betrachten nicht der Fall ist.

Die Frage warum Stromchiffren überhaupt interessant sind, wo es doch sehr gute Blockciffren gibt, die sehr gut bezüglich ihrer Sicherheit untersucht sind, läßt sich hauptsächlich mit dem (erhofften) Geschwindigkeitsgewinn begründen.

Vorteile von Stromchiffren

- in Bezug auf den Datendurchsatz, ist eine Stromchiffre häufig schneller als eine Blockchiffre
- eine Stromchiffre ist üblicherweise kleiner und billiger in Soft- und insbesondere in Hardware umzusetzen, da lediglich einfache Operationen auf Bitebene benötigt werden und diese mit wenigen Gattern umzusetzen sind
- die Latenzzeit ist im Vergleich zur Blockchiffre häufig kleiner

Nachteile von Stromchiffren

- es gibt kaum gut untersuchte, sichere, schnelle Stromchiffren
- die Sicherheit von Stromchiffren ist heutzutage schwerer zu untersuchen als bei Blockchiffren[1]

Im folgenden wollen wir uns auf die wichtigste Klasse von Stromchiffren beschränken, solche die mit Hilfe von Linearen Schiebe Registern (LFSR) konstruiert werden. Bei der mathematischen Untersuchung von LFSRs wählen wir die Darstellung mit Hilfe der Spurabbildung, die aus vorherigen Kapiteln schon bekannt ist. Diese Darstellung hat den Vorteil, das sich Eigenschaften wie die Periode der Erzeugten Sequenzen sehr einfach folgen.

3.2 LFSRs

Linear Feedback Shift Register (LFSR) werden heutzutage in vielen Schlüsselstrom-Generatoren verwendet. Abbildung 3.2 zeigt das Prinzip eines LFSR.

In einem LFSR, realisiert als n-Bit-Register, werden Bits an festen Positionen per XOR durch Rückkopplung verknüpft. Der so berechnete Wert wird links in das Register eingefügt, nachdem der Inhalt nach rechts geshiftet wurde. Sei $s_{n-1}, s_{n-2}, \ldots, s_0 \in \mathbb{F}_2$ der Initialzustand des Registers und

[1] Zur Zeit läuft eine europäische Ausschreibung, um sichere Stromchiffren zu entwickeln.

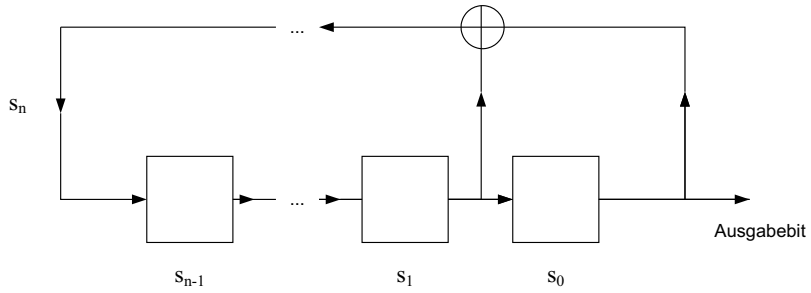

Abbildung 3.2: Prinzip eines LFSR

$a_0, \ldots, a_{n-1} \in \mathbb{F}_2$ die Rückkopplungs-Koeffizienten ($0 \rightarrow$ Bit wird nicht rückgekoppelt, $1 \rightarrow$ Bit wird rückgekoppelt), dann wird in jedem Takt das rechte Bit des Registers (beim Rechts-Shift) ausgegeben. Der neu berechnete Wert

$$s_n = a_0 s_{n-1} + a_1 s_{n-2} + \ldots + a_{n-1} s_0$$

wird das neue linke Bit des Registers. Eine Gleichung dieser Form nennt man *Lineare Rekursionsgleichung.*

3.2.1 Ein prominentes Beispiel

Wir betrachten als Einstieg ein sehr bekanntes Beispiel über den reellen Zahlen.

Ein Beispiel über $\mathbb{R}$ Sei folgende Rekursionsgleichung

$$s_t \quad = \quad s_{t-1} + s_{t-2} \tag{3.1}$$

mit

$$s_0 = 0 \text{ und } s_1 = 1$$

gegeben. Die ersten Folgenglieder dieser Rekursion Gleichung lauten

$$(s_0, s_1, s_2, \ldots) = (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots)$$

Diese Sequenz nennt man *Fibonacci-Zahlen.* Wir sind an einer geschlossenen Formel für das t.te Folgenglied interessiert, die folgenden Überlegungen werden wir später auf den Fall einer linearen Rekursionsgleichung über $\mathbb{F}_2$ übertragen.

Wir nehmen an, es gibt eine Lösung der Rekursionsgleichung (3.1) der Form, so dass für alle $t \in \mathbb{N}$ gilt

$$s_t = \alpha^t$$

mit $\alpha \in \mathbb{R} \setminus \{0\}$. Setzen wir diese Gleichung für s_t in (3.1) ein, ergibt sich

$$\alpha^t = \alpha^{t-1} + \alpha^{t-2} \qquad t \geq 2$$

Geteilt durch α^{t-2}, erhält man das *charakteristische Polynom*

$$\alpha^2 - \alpha - 1 = 0.$$

Die gesuchte reelle Zahl α muss also eine Nullstelle des Polynoms $X^2 - X - 1$ sein. Die beiden (reellen) Lösungen sind

$$\alpha_1 = \frac{1+\sqrt{5}}{2} \qquad \alpha_2 = \frac{1-\sqrt{5}}{2}$$

Daraus folgt, dass sowohl (α_1^t) und (α_2^t), und, wegen der Linearität der Gleichung (3.1), auch jede Linearkombination dieser beiden Sequenzen unserer Rekursionsgleichung genügen. Somit entspricht die Sequenz

$$s_t \;=\; A \cdot \alpha_1^t + B \cdot \alpha_2^t$$

mit den zwei Konstanten $A, B \in \mathbb{R}$ der Gleichung (3.1). Um nicht nur die Rekursionsgleichung, sondern auch den gegebenen Anfangszuständen gerecht zu werden, müssen die Konstanten A, B für die Fibonacci-Sequenz mit den Initialwerten

$$s_0 = 0 \qquad s_1 = 1.$$

bestimmt werden. Es muss also gelten:

$$s_0 = A + B = 0$$

$$s_1 = A \cdot \alpha_1 + B \cdot \alpha_2 = 1$$

Daraus folgt:

$$A = \frac{1}{\sqrt{5}}$$

$$B = -\frac{1}{\sqrt{5}}$$

Wir haben zusammenfassend gezeigt, dass die Fibonacci-Zahlen die Gleichung

$$s_t = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^t - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^t$$

erfüllen.

Das gleiche Beispiel, diesmal über $GF(2)$ Da wir für die Konstruktion von Stromchiffren sicher nicht mit reellen Zahlen, sondern mit Bits beziehungsweise mit Elementen aus $\mathbb{F}_2$ rechnen werden, übertragen wir nun die aus dem Beispiel gewonnen Erkenntnisse auf $\mathbb{F}_2$. Wir betrachten wieder die Rekursionsgleichung

$$s_t = s_{t-1} + s_{t-2} \tag{3.2}$$

mit

$$s_0 = 0 \text{ und } s_1 = 1$$

mit dem einzigen Unterschied, dass wir die Elemente jetzt als Elemente im $\mathbb{F}_2$ auffassen. In diesem Fall lauten die ersten Folgen der Sequenz

$$(s_0, s_1, s_2, \ldots) = (0, 1, 1, 0, 1, 1, 0, 1, 1, 0, \ldots),$$

Insbesondere ist die Folge periodisch mit Periodenlänge 3. Das jede solche Sequenz irgendwann periodisch werden muss ist klar. Warum die Periodenlänge von 3 aber kein Zufall ist, werden wir weiter unten klären.

Wie oben nehmen wir an, dass es ein α gibt, so das

$$s_t = \alpha^t \qquad \alpha \neq 0$$

für alle $t \in \mathbb{N}$ gilt. Analog zum Beispiel über $\mathbb{R}$ muss α eine Nullstelle von

$$f(x) = x^2 - x - 1$$

oder im $\mathbb{F}_2$ äquivalent hierzu eine Nullstelle von $f(x) = x^2 + x + 1$ sein. Da f über $\mathbb{F}_2$ irreduzibel ist, liegen die Nullstellen von f im Erweiterungskörper

$$\mathbb{F}_{2^2} = \mathbb{F}_2[x]/f(x) = \mathbb{F}_2[\alpha] \quad \text{mit} \quad \alpha^2 + \alpha + 1 = 0.$$

Die Elemente des F_{2^2} sind $\{0, \alpha^0 = 1, \alpha^1, \alpha^2 = \alpha + 1\}$. Die Nullstellen von $f(x)$ sind $\alpha_1 = \alpha$ und $\alpha_2 = \alpha^2 = \alpha + 1$. Damit erfüllen die Werte

$$s_t = A \cdot \alpha_1^t + B \cdot \alpha_2^t$$

die Rekursionsgleichung für konstante $A, B \in F_{2^2}$. Für die gegebenen Initialwerte muss analog zum obigen Beispiel gelten:

$$s_0 = A + B = 0$$

$$s_1 = A \cdot \alpha_1 + B \cdot \alpha_2 = 1$$

Daraus folgt:

$$A = B = 1$$

Damit wird folgende Gleichung erfüllt:

$$s_t = \alpha^t + (\alpha + 1)^t = \alpha^t + (\alpha^t)^2 = Tr(\alpha^t)$$

Unterschiedliche Initialwerte ergeben andere Sequenzen und damit andere Konstanten A, B. Wählt man zum Beispiel die Initialwerte

$$s_0 = 0 \quad \text{und} \quad s_1 = 0$$

erhält man

$$s_t = 0 = Tr(0 \cdot \alpha^t).$$

Als letztes Beispiel betrachten wir

$$s_0 = 1 \quad \text{und} \quad s_1 = 0$$

und für die Koeffizienten A, B muss daher

$$s_0 = A + B = 1$$

$$s_1 = A \cdot \alpha + B \cdot (\alpha + 1) = 0$$

gelten. Man erhält

$$A = \alpha + 1 \qquad B = \alpha,$$

und somit

$$\begin{aligned} s_t &= (\alpha + 1) \cdot \alpha^t + \alpha \cdot (\alpha + 1)^t \\ &= (\alpha + 1) \cdot \alpha^t + \alpha \cdot (\alpha^t)^2 \\ &= (\alpha + 1) \cdot \alpha^t + (\alpha^2 + 1) \cdot (\alpha^t)^2 \\ &= (\alpha + 1) \cdot \alpha^t + ((\alpha + 1) \cdot \alpha^t)^2 \\ &= Tr((\alpha + 1) \cdot \alpha^t) \end{aligned}$$

Das sich die Sequenzen für jeden Startwert als $\mathrm{Tr}(\theta\alpha^t)$ für ein $\theta \in \mathbb{F}_{2^2}$ schreiben ließen, werden wir im Folgenden allgemein untersuchen.

3.2.2 Darstellung der Folgen mit Hilfe der Spur-Funktion

Zuerst eine grundlegende Definition

Definition 22 *Für eine lineare Rekursionsgleichung*

$$s_t = a_1 s_{t-1} + \ldots + a_{m-1} s_{t-m} \qquad a_i \in \mathbb{F}_2$$

heißt

$$f(x) = x^m - a_1 x^{m-1} - \ldots - a_m x^0$$

charakteristisches Polynom *der Rekursionsgleichung.*

Ganz analog zu den Überlegungen bei unserem Einführungsbeispiel beweist man folgenden Satz.

Theorem 23 *Seien $\alpha_1, \ldots, \alpha_r$ verschiedene Nullstellen des charakteristischen Polynoms, dann erfüllen die Werte*

$$s_t = \lambda_1 \alpha_1^t + \ldots + \lambda_r \alpha_r^t$$

für beliebige $\lambda_i \in \mathbb{F}_{2^m}$ die Rekursionsgleichung.

Interessant wird es, wenn das Polynom $f(x)$ irreduzibel ist. In diesem Fall besitzt $f(x)$ m verschiedene Nullstellen und mit dem folgenden Satz lassen sich aus einer Nullstelle alle anderen leicht ermitteln.

Theorem 24 *Sei $g \in \mathbb{F}_2[x]$ und $\alpha \in F_{2^n}$ eine Nullstelle von g. Dann sind $\alpha^2, \alpha^4, \alpha^8, \ldots, \alpha^{2^{m-1}} \in \mathbb{F}_{2^n}$ ebenfalls Nullstellen von f.*

Beweis: Sei

$$g(x) = \sum_{i=0}^{m} \lambda_i x^i \quad \text{mit} \quad \lambda_i \in \mathbb{F}_2$$

Dann ist

$$\begin{aligned} g(\alpha^2) &= \sum_{i=0}^{m} \lambda_i (\alpha^2)^i = \sum_{i=0}^{m} \lambda_i (\alpha^i)^2 \\ &= \sum_{i=0}^{m} (\lambda_i \alpha^i)^2 \quad \text{mit} \quad \lambda_i^2 = \lambda_i \quad \text{da} \quad \lambda_i \in \mathbb{F}_2 \\ &= (\sum_{i=0}^{m} \lambda_i \alpha^i)^2 = (g(\alpha))^2 = (0)^2 = 0 \end{aligned}$$

Per induktion folgt damit $g(\alpha^{2^i}) = 0$ für alle natürlichen Zahlen i. □

Das folgende Resultat ist der Hauptsatz in unseren Behandlungen von LFSRs und erklärt die Phänomene aus dem Einführungsbeispiel.

Theorem 25 *Sei eine lineare Rekursionsgleichung mit irreduziblem charakteristischen Polynom*

$$f = x^m + \sum_{i=1}^{m} a_i \alpha^{m-i}$$

mit Nullstelle $\alpha \in \mathbb{F}_{2^m}$ gegeben. Für alle $\theta \in \mathbb{F}_{2^m}$ erfüllen die Werte $s_t = \mathrm{Tr}(\theta \alpha^t)$ die Rekursionsgleichung. Umgekehrt gilt: Für jede Lösung s_t der Rekursionsgleichung gibt es ein $\theta \in \mathbb{F}_{2^m}$, so dass $s_t = \mathrm{Tr}(\theta \alpha^t)$ gilt.

Beweis Der Beweis besteht aus zwei Teilen. Sei zunächst $\theta \in \mathbb{F}_{2^m}$ gegeben. Wir müssen zeigen, dass für $s_t = \mathrm{Tr}(\theta\alpha^t)$ die Rekursionsgleichung erfüllt ist. Es gilt:

$$\begin{aligned} s_t &= \mathrm{Tr}(\theta\alpha^t) \\ &= \mathrm{Tr}(\theta\alpha^{t-m}\alpha^m) \end{aligned}$$

und da α eine Nullstelle von f ist, gilt $\alpha^m = \sum_{i=1}^{m} a_i\alpha^{m-i}$. Wir erhalten

$$\begin{aligned} s_t &= \mathrm{Tr}\left(\theta\alpha^{t-m}\left(\sum_{i=1}^{m} a_i\alpha^{m-i}\right)\right) \\ &= \sum_{i=1}^{m} \mathrm{Tr}(\theta\alpha^{t-m}a_i\alpha^{m-i}) \end{aligned}$$

wobei wir für die letzte Umformung genutzt haben, dass die Spur-Funktion linear ist und $\mathrm{Tr}(ax) = a\,\mathrm{Tr}(x)$ für alle $a \in \mathbb{F}_2$ und $x \in \mathbb{F}_{2^m}$ gilt. Daraus folgt:

$$\begin{aligned} s_t &= \sum_{i=1}^{m} a_i\,\mathrm{Tr}(\theta\alpha^{t-i}) \\ &= \sum_{i=1}^{m} a_i s_{t-i} \end{aligned}$$

Damit ist bewiesen, dass für jedes $\theta \in \mathbb{F}_{2^m}$ die Rekursionsgleichung erfüllt wird. Um die Umkehrung zu beweisen, also zu zeigen, dass auch jede Lösung s_t ein $\theta \in \mathbb{F}_{2^m}$ gibt, so dass $s_t = \mathrm{Tr}(\theta\alpha^t$ gilt, nutzen wir ein Abzählargument. Da es genau $2^m - 1$ von Null verschiedene Startwerte $s_0, \ldots, s_{m-1}$ gibt, gibt es auch genau $2^m - 1$ verschiedene Lösungen der Rekursionsgleichung. Da es auch genau $2^m - 1$ viele Elemente $\theta \neq 0$ gibt, bleibt zu zeigen, dass verschiedene θ immer unterschiedliche Sequenzen produzieren. Um den Beweis abschließen zu können benötigen wir noch folgende Lemmata:

Lemma 26 *Sei f ein irreduzibles Polynom mit der Nullstelle α. Dann sind die Elemente $1, \alpha, \alpha^2, \ldots, \alpha^{m-1}$ über $\mathbb{F}_2$ linear unabhängig.*

Beweis Wir führen einen Widerspruchsbeweis. Annahme: Seien die α^i **nicht** linear unabhängig, d.h. $\exists\lambda_i \in \mathbb{F}_2$ so dass

$$\sum_{i=0}^{m-1} \lambda_i\alpha^i = 0.$$

Dies bedeutet jedoch, dass α eine Nullstelle von

$$g(x) = \sum_{i=0}^{m-1} \lambda_i x^i \quad \text{mit} \quad grad(g) \leq m-1$$

ist. Da α aber eine Nullstelle von f **und** g ist, folgt $ggT(f,g) \neq 1$. Zusätzlich ist $grad(g) < grad(f)$, und somit gilt $ggT(f,g) \neq f$. Damit ist $ggT(f,g)$ ein nicht-trivialer Teiler von f, was jedoch der Irreduzierbarkeit von f widerspricht.

Lemma 27 *Die Abbildung* $\mathrm{Tr} : \mathbb{F}_{2^n} \to \mathbb{F}_2$ *ist nicht die konstante Null-Abbildung, d.h für jedes n gibt es Elemente mit* $\mathrm{Tr}(x) = 1$.

Beweis Wie wir bei der Einführung der Spur-Funktion gesehen haben, sind die Elemente mit $\mathrm{Tr}(x) = 0$ genau die Elemente mit $x = y^2 + y$. D.h. ein Element hat $\mathrm{Tr}(x) = 0 \Leftrightarrow x \in Bild(f)$ mit $f(y) = y^2 + y$. Wir müssen zeigen, dass f nicht bijektiv ist ($\Leftrightarrow Bild(f) \neq \mathbb{F}_{2^n}$). Da f linear ist, reicht es zu zeigen, dass es Elemente $y \neq 0$ gibt mit $f(y) = 0$.

$$\Leftrightarrow y^2 + y = 0 \Leftrightarrow y = 0, \underline{1}$$

□ Als letzten Hilfssatz benötigen wir noch eine Folgerung aus Lemma 27.

Lemma 28 *Seien* $1, \alpha, \alpha^2, \ldots, \alpha^{m-1}$ *linear unabhängig über* $\mathbb{F}_2$*, dann sind für* $\theta \in \mathbb{F}_{2^m}$ *folgende Aussagen äquivalent.*

1. $$\mathrm{Tr}(\theta\alpha^i) = 0 \quad \forall \quad i \in \{0, \ldots, m-1\}$$

2. $$\theta = 0$$

Beweis Es ist klar, das aus $\theta = 0$ folgt, dass $\mathrm{Tr}(\theta\alpha^i) = 0$ gilt. Sei nun $\beta \in \mathbb{F}_{2^m}$. Da $1, \alpha, \alpha^2, \ldots, \alpha^{m-1}$ linear unabhängig sind - und damit eine Basis von $\mathbb{F}_{2^m}$ über $\mathbb{F}_2$ bilden - gibt es Koeffizienten $\lambda_i \in \mathbb{F}_2$, so dass

$$\beta = \sum_{i=0}^{m-1} \lambda_i \alpha^i.$$

Dann gilt

$$\begin{aligned}\mathrm{Tr}(\theta\beta) &= \mathrm{Tr}\left(\theta\left(\sum_{i=0}^{m-1}\lambda_i\alpha^i\right)\right)\\ &= \sum_{i=0}^{m-1}\mathrm{Tr}(\theta\lambda_i\alpha^i)\\ &= \sum_{i=0}^{m-1}\lambda_i\,\mathrm{Tr}(\theta\alpha^i) = 0\end{aligned}$$

nach Voraussetzung. Damit muss wegen Lemma 27 $\theta = 0$ gelten. □

3.2.3 Die Periode eines LFSRs

Es ist klar, dass jedes LFSR periodisch sein muss und höchstens die Periode $2^m - 1$ haben kann. Für die Nutzung von LFSRs in Stromchiffren ist es wichtig, LFSRs mit maximaler Periode zu nutzen. Sei $s_t = \mathrm{Tr}(\theta\alpha^t)$ die Folge eines LFSRs mit der Periode n. Das heisst für alle $t \in \mathbb{N}$ gilt $s_t = s_{t+n}$, also

$$\mathrm{Tr}(\theta\alpha^t) = \mathrm{Tr}(\theta\alpha^{t+n}).$$

Daraus folgt

$$\begin{aligned}0 &= \mathrm{Tr}(\theta(\alpha^t + \alpha^{t+n})\\ &= \mathrm{Tr}(\theta\alpha^t(\alpha^n + 1))\end{aligned}$$

und nach Lemma 28 gilt somit

$$\theta(\alpha^n + 1) = 0.$$

Für $\theta \neq 0$ muss daher $\alpha^n = 1$ gelten. Die minimale Periode entspricht damit der Ordnung von $\alpha \in \mathbb{F}_{2^m}^* = \mathbb{F}_{2^m} \setminus \{0\}$. Damit haben wir folgenden Satz bewiesen:

Theorem 29 *Sei ein LFSR mit irreduzibelem Polynom f vom Grad m mit Nullstelle α gegeben. Für die Periode n eines LFSRs gilt:*

$$n = \mathrm{ord}(\alpha)$$

Insbesondere ist die Periode immer ein Teiler von $2^m - 1$. Das LFSR hat maximale Periode genau dann, wenn α ein primitives Element in $\mathbb{F}_{2^m}^*$ ist.

Das Theorem motiviert daher auch die folgende Definition.

Definition 30 *Ein Polynom $f \in \mathbb{F}_2[x]$ mit $grad(f) = m$ heisst primitiv, wenn die Nullstellen von f die Ordnung $2^m - 1$ haben, also primitiv sind.*

3.2.4 Stromchiffren mit LFSRs

LFSRs haben Eigenschaften, die für Stromchiffren sehr wichtig sind:

- gute statistische Eigenschaften (insbesondere Gleichverteilung)
- die Periode von LFSRs ist bekannt
- LFSRs sind in Hardware sehr schnell und klein

LFSRs alleine eignen sich nicht als Stromchiffren, da sie *linear* sind. Es gibt Algorithmen, zum Beispiel Berlekamp-Massey, die sehr effizient aus einer Sequenz den zugehörigen LFSR berechnen können. Hierfür reicht eine Sequenz der Länge $2 \cdot g$, wobei g der Grad des LSFRs ist. Daher müssen die LFSRs mit nichtlinearen Komponenten verknüpft werden, um eine sichere Stromchiffre zu erhalten.

3.2.5 Nicht-lineare Combination Generator

Die Idee besteht darin, die Ausgaben von mehreren LFSRs mit Hilfe einer nichtlinearen Funktion zu verknüpfen. Diese Funktion ist in Abbildung 3.3 dargestellt.
Der geheime Schlüssel bei einer solchen Konstruktion besteht aus:

- den Anfangszuständen der LFSRs
- evtl. den charakteristischen Polynomen

Ein Nachteil bei der zweiten Variante sind die höheren Kosten für die Implementierung in Hardware und die Schwierigkeit der Sicherstellung einer maximalen Periode. Die Funktion f soll die Linearität des Schlüsselstroms zerstören.

Im nächsten Abschnitt soll eine sehr mächtige Attacke beschrieben werden, aus der sich insbesondere Auswahlkriterien für die Funktion f ergeben.

3.2.6 Korrelationsattacke auf Combination Stromchiffren

Wir nehmen für die weiteren Untersuchungen an, dass charakteristische Polynome der LFSRs bekannt sind und der geheime Schlüssel aus den Anfangszuständen der LFSRs besteht.

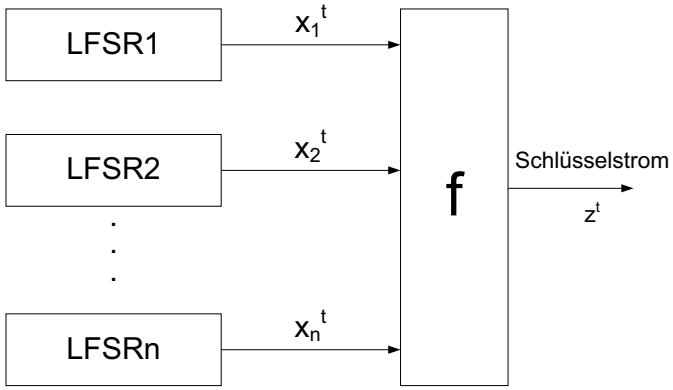

Abbildung 3.3: Funktion f

Beobachtung: Wenn die Wahrscheinlichkeit, dass $f(x_1, \ldots, x_n) = x_i$ für ein $i \in \{1, \ldots, n\}$ ungleich $\frac{1}{2}$ ist, dann ist folgender Angriff möglich: Für jeden möglichen Anfangszustand des i-ten LFSRs zählt man für einen bekannten Teil des Schlüsselstroms, wie oft der Schlüsselstrom mit dem jeweiligen Ausgabebit des i-ten LFSR übereinstimmt.

- Wenn diese Anzahl mit der erwarteten Anzahl übereinstimmt, dann ist ein Kandidat für den Anfangszustand vom i-ten LFSR gefunden.
- Wenn nicht, dann war der Anfangszustand wahrscheinlich nicht richtig.

Wenn die Länge des bekannten Schlüsselstroms mit der Periode des i-ten LFSRs übereinstimmt, dann führt dieser Angriff auf den richtigen Anfangszustand. Wir betrachten folgendes Beispiel

Beispiel für die Funktion f

$$f : \mathbb{F}_2^3 \to \mathbb{F}_2$$

$$f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3$$

Wertetabelle:

x_1	x_2	x_3	z	$z = x_1$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

In diesem Fall ist die Wahrscheinlichkeit, dass $z^t = x_1^t$.

$$\mathcal{P}(z^t = x_1^t) = \frac{6}{8} = \frac{3}{4}$$

Daher kann man hier durch Raten des Initialzustandes des ersten LFSRs eine Korrelationsattacke durchführen.

Aufwand dieser Attacke: Der Aufwand bezieht sich auf die Größe der Periode des i-ten LFSRs, unter der Annahme, dass das charakteristische Polynom primitiv ist. Das heisst, der Aufwand entspricht dem Aufwand eines Brute-Force-Angriffs auf einen Teilschlüssel.

Kriterien für die Funktion f Um diesen Angriff zu verhindern, sollte die Funktion f die Eigenschaft haben, dass

$$\mathcal{P}\left(f(x_1, \ldots, x_n) = \sum_{i \in I} x_i\right) = \frac{1}{2} \quad \forall I \subseteq \{1, \ldots, n\} \quad \text{mit} \ \mid I \mid \leq B$$

Für den Designer am Besten, und für einen Angreifer am Schlechtesten, wäre $B = n$. Mit $B = n$ wäre die Attacke nicht effizienter als ein Brute-Force-Angriff. Die folgende Definition gibt der gewünschten Eigenschaft einen Namen.

Definition 31 *Eine Funktion* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *heisst* m-correlation-immune*, wenn gilt:*

$$f^W(a) = 0 \quad \forall a \in \mathbb{F}_2^n, \ \textit{mit } wt(a) \leq m$$

Mit $wt(a) = \mid \{i \mid a_i - 1\} \mid$ *bezeichnet man das Hamming-Gewicht von* a*.* f *heisst* m-resilient*, wenn* f *balanced ist, d.h.* $f^W(0) = 0$*.*

Das diese Definition genau die gewünschten Eigenschaften wiederspiegelt, ist nicht ganz offensichtlich. Das folgende Theorem gibt den entsprechenden Zusammenhang wieder.

Theorem 32 *Sei* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *m-correlation-immune. Dann gilt* $\forall I \subseteq \{1, \ldots, n\}$ *mit* $\mid I \mid \leq m$*:*

$$\mathcal{P}\left(f(x_1, \ldots, x_n) = \sum_{i \in I} x_i\right) = \frac{1}{2}$$

Beweis Befindet sich auf dem Übungsblatt 11 □

Ist die Funktion f, die als Filter für die LFSRs genutzt wird, m-correlation immune, müssen also mindestens die Zustände von m LFSRs geraten werden um einen Korrelations Angriff möglich zu machen. Der Aufwand dieser Attacke entspricht also einer Brute Force Suche auf diese Teilschlüssel. Für einen Designer wäre daher eine m-correlation immune Funktion optimal. Man kann sich aber leicht überlegen, das es eine solche Funktion nicht geben kann. Das nächstbeste wäre demnach eine $m-1$-correlation immune Funktion. Für unsere Traum-Funktion $f : \mathbb{F}_2^m \to \mathbb{F}_2$ gilt also

$$f^W(a) = 0 \quad \forall a \neq 0 \text{ mit } wt(a) \leq m-1$$

Das heisst, $f^W(a) \neq 0$ gilt nur für $a = 0$ oder $a = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$. Wir nehmen nun an, dass f zusätzlich gleichverteilt (balanced) ist. Das ist eine wichtige Eigenschaft, um nicht die guten statistischen Eigenschaften der LFSRs zu zerstören. Dann gilt für alle $a \neq \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$, dass $f^W(a) = 0$. Mit der in den Übungen behandelten Gleichung

$$\sum_{a \in \mathbb{F}_2^n} f^W(a)^2 = 2^{2n}$$

folgt damit $f^W \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \pm 2^n$ und damit

$$f(x) = \sum_{i \in I} x_i + c \quad \text{mit} \quad c \in \mathbb{F}_2$$

Also ist f affin und damit nicht geeignet als Kombinations-Funktion. Die Linearität der LFSRs wird nicht zerstört und der Keystream ist linear abhängig von dem Schlüssel. Insbesondere ist damit gezeigt, dass es auf solche Stromchiffren immer Angriffe gibt, die schneller als Brute-Force-Angriffe sind.

Eine Konstruktion für m-correlation-immune Funktionen In diesem Abschnitt wollen wir eine Konstruktion von m-correlation-immune Funktionen vorstellen, die den weiter oben eingeführten Maiorana McFarland Bent Funktionen sehr ähnlich ist.

Theorem 33 *Seien $r, s \in \mathbb{N}$ und $\pi : \mathbb{F}_2^r \to \mathbb{F}_2^s$ eine Funktion mit der Eigenschaft, dass $\forall a \in \mathbb{F}_2^s$ mit $wt(a) \leq m$ gilt:*

$$\pi^{-1}(a) = \{x \in \mathbb{F}_2^r \mid \pi(x) = a\} = \emptyset$$

Weiterhin sei $h : \mathbb{F}_2^r \to \mathbb{F}_2$ eine beliebige Boolesche Funktion. Dann ist

$$f : \mathbb{F}_2^s \times \mathbb{F}_2^r \to \mathbb{F}_2 \quad mit \quad f(x,y) =< x, \pi(y) > +h(y)$$

m-correlation-immune.

Beweis Sei $a \in \mathbb{F}_2^r, b \in \mathbb{F}_2^s$. Dann gilt:

$$\begin{aligned} f^W(a,b) &= \sum_{x \in \mathbb{F}_2^r, y \in \mathbb{F}_2^s} (-1)^{f(x,y)+<a,x>+<b,y>} \\ &= \sum_{y \in \mathbb{F}_2^s} (-1)^{<b,y>} \sum_{x \in \mathbb{F}_2^r} (-1)^{f(x,y)+<a,x>} \\ &= \sum_{y \in \mathbb{F}_2^s} (-1)^{<b,y>+h(y)} \sum_{x \in \mathbb{F}_2^r} (-1)^{<x,\pi(y)>+<a,x>} \\ &= \sum_{y \in \mathbb{F}_2^s} (-1)^{<b,y>+h(y)} \sum_{x \in \mathbb{F}_2^r} (-1)^{<a+\pi(y),x>} \end{aligned}$$

Es gilt:

$$\sum_{x \in \mathbb{F}_2^r} (-1)^{<a+\pi(y),x>} = \begin{cases} 0, \text{ sonst} \\ 2^r, \text{ wenn } \pi(y) = a \Leftrightarrow y \in \pi^{-1}(a) \end{cases}$$

Damit folgt:

$$f^W(a,b) = 2^r \sum_{y \in \pi^{-1}(a)} (-1)^{<b,y>+h(y)}$$

Nehmen wir nun an, dass der Vektor $(a,b) \in \mathbb{F}_2^r \times \mathbb{F}_2^s$ ein Gewicht $\mathrm{wt}((a,b)) \leq m$ hat. Dann folgt insbesondere $\mathrm{wt}(a) \leq m$. Mit der Konstruktion von $\pi : \mathbb{F}_2^r \to \mathbb{F}_2^s$ folgt für $\mathrm{wt}(a) \leq m$, dass das Urbild von a unter π leer ist, also

$$\pi^{-1}(a) = \emptyset$$

und in diesem Fall ist

$$f^W(a,b) = \sum_{y \in \emptyset} (-1)^{<b,y>+h(y)} = 0$$

□

Diese Konstruktion erlaubt auch die Konstruktion von nicht-linearen Funktionen, indem zum Beispiel die Funktion $h : \mathbb{F}_2^r \to \mathbb{F}_2$ nicht-linear ist.

3.2.7 Algebraische Attacken

In diesem Abschnitt wird auf eine relativ neue und sehr mächtige Klasse von Attacken eingegangen. Bevor jedoch die eigentliche Idee der Attacke beschrieben werden kann benötigen wir noch einige Grundlagen über Boolsche Funktionen

3.2.8 Boole'sche Funktionen

Funktionen der Form $f : \mathbb{F}_2^n \to \mathbb{F}_2$ werden als *Boole'sche Funktionen* bezeichnet. Die wichtigste Eigenschaft von Boolschen Funktionen für uns in diesem Abschnitt ist die Möglichkeit solche Funktionen als (multivariate) Polynome zu beschreiben, die sogenannte Algebraische Normalform einer Boole'schen Funktion.

Definition 34 *Sei* $\forall u \in \mathbb{F}_2^n$ *und* $\lambda^u \in \mathbb{F}_2$. *Dann heißt die Darstellung von*

$$f(x_1, \ldots, x_n) = \sum_{u \in \mathbb{F}_2^n} \left(\lambda_u \prod_{i=1}^{n} x_i^{u_i} \right)$$

Algebraische Normalform *(ANF) von* f.

Im Fall von Boole'schen Funktionen kann man die Multiplikation als logisches UND und die Addition als logisches XOR auffassen. Damit können wir den folgenden Satz (ohne Beweis) formulieren:

Theorem 35 *Jede Boole'sche Funktion hat eine eindeutige Algebraische Normalform.*

Die Darstellung von Boolschen Funktionen mittels der Algebraischen Normalform werden wir im nächsten Abschnitt für die Algebraische Attacke auf Stromchiffren nutzen.

Für die effiziente Berechnung der ANF nutzen wir ein Verfahren, dass der FFT zur Berechnung der Walschkoeffizienten sehr ähnlich ist. Das vorgehen wird am Beispiel einer Funktion $f : \mathbb{F}_2^2 \to \mathbb{F}_2$, die durch ihre Wertetabelle gegeben ist, demonstriert.

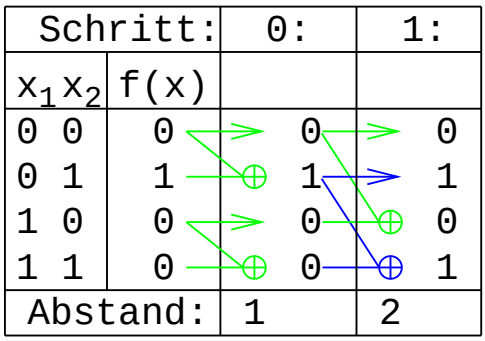

Abbildung 3.4: Beispiel für die Berechnung der Algebraischen Normalform.

Die Anzahl der Schritte ist wie in der FFT n, der Abstand der gepaarten Werte beträgt im Schritt j wie in der FFT 2^j. Der einzige Unterschied zur FFT liegt in der Berechnung der Werte: Für die „obere“ Position wird der Wert direkt übernommen, die „untere“ Position ergibt sich aus dem XOR der beiden Werte. Die ANF der Funktion lässt sich aus der letzten Spalte ablesen

$$\begin{aligned} f(x_1, x_2) &= \lambda_{00} x_1^0 x_2^0 + \lambda_{01} x_1^0 x_2^1 + \lambda_{10} x_1^1 x_2^0 + \lambda_{10} x_1^1 x_2^1 \\ &= 0 * 1 + 1 * 1 * x_2 + 0 * x_1 * 1 + 1 * x_1 x_2 \\ &= x_1 x_2 + x_2 \end{aligned}$$

Das Verfahren hat also das richtige Ergebnis geliefert, wie sich leicht durch eine Probe verifizieren lässt. Auf den Beweis, dass dies immer so ist, verzichten wir an dieser Stelle. Der Aufwand beträgt analog zu dem Aufwand der FFT $\frac{1}{2} n 2^n$.

Definition 36 *Der Grad einer Boole'schen Funktion* $f : \mathbb{F}_2^n \to \mathbb{F}_2$ *mit Algebraische Normalform*

$$f(x_1, \ldots, x_n) = \sum_{u \in \mathbb{F}_2^n} \left(\lambda_u \prod_{i=1}^{n} x_i^{u_i} \right)$$

ist definiert als

$$grad(f) = \max\{\mathrm{wt}(u) \mid \lambda_u \neq 0\}$$

Allgemeines Vefahren

Wir nehmen für die algebraischen Attacken an, dass der folgende Aufbau

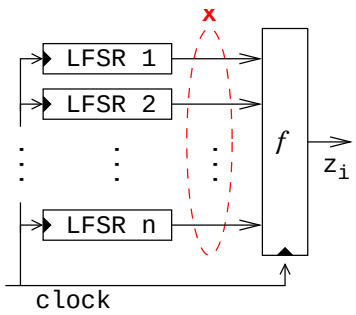

Abbildung 3.5: LFSR-Schaltung für die algebraischen Attacken

verwendet wird. Dabei bekommt die Boole'sche Funktion $f : \mathbb{F}_2^n \to \mathbb{F}_2$ mit jedem Takt aus jedem LFSR ein Eingabebit und berechnet daraus genau ein Bit z_i für den Keystream. Man kann aus dem aktuellen Zustand i eines LFSR den nachfolgenden Zustand $i+1$ mit einer quadratischen Matrix L wie folgt berechnen:

$$\begin{pmatrix} s_1 \\ \vdots \\ s_m \end{pmatrix} = L \begin{pmatrix} s_1 \\ \vdots \\ s_{m-1} \end{pmatrix}$$

Aus dem Anfangszustand lässt sich die Ausgabe s_{m+i} mit

$$s_{m+i} = L^i \begin{pmatrix} s_1 \\ \vdots \\ s_m \end{pmatrix}$$

berechnen, wobei L die Form

$$L = \begin{pmatrix} 0 & 1 & 0 & & \dots & & 0 \\ 0 & 0 & 1 & 0 & \dots & & \\ \vdots & \vdots & & \ddots & & & \vdots \\ 0 & & & \dots & & 1 & 0 \\ 0 & & & \dots & & & 1 \\ a_1 & & & \dots & & & a_m \end{pmatrix}$$

hat und die a_i durch die lineare Rekursionsgleichung $s_t = \sum_{i=1}^{m} a_i s_i$ gegeben sind.

Für die Konstruktion aus n LFSR's gibt es also eine lineare Abbildung (als Matrix M dargestellt), so dass sich der Keystream als

$$f\left(M^j\left(s_1^{(1)}, \dots, s_{m_1}^{(1)}, s_1^{(2)}, \dots, s_{m_2}^{(2)}, \dots, s_1^{(n)}, \dots, s_{m_n}^{(n)}\right)\right) = z_j$$

schreiben lässt, wobei $\left(s_1^{(i)}, \dots, s_{m_i}^{(i)}\right)$ den Initialzustand des i-ten LFSR's bezeichnet.

Bemerkung: Diese Beschreibung ist analog zur Literatur gewählt, allerdings ist sie nicht ganz sauber: Um Potenzen einer Matrix M bilden zu

können, muss die Matrix von einem Raum $\mathbb{F}_2^m$ in den selben Raum $\mathbb{F}_2^m$ abbilden. In der obigen Notation bildet M aber von $\mathbb{F}_2^{m_{ges}}$ nach $\mathbb{F}_2^n$ ab, wobei $m_{ges} = \sum_{\forall i} m_i$.

Wenn ein Teil des Keystreams bekannt ist, kann man für jedes bekannte Bit z_j des Keystreams eine Gleichung der Form $f\left(M^j\left(s_0^{(1)}, \ldots, s_{t_n}^{(n)}\right)\right) = z_j$ aufstellen. Insgesamt erhält man so ein Gleichungssystem vom Grad $d = grad(f)$. Der Aufwand zum Lösen dieses Systems mit dem Gauss'schen Linearisierungsverfahren ist:

$$\left(\sum_{i=0}^{d} \binom{m_{ges}}{i}\right)^3$$

Interessant ist hierbei, dass der Aufwand polynomiell in m ist.

Verfahren zur Reduzierungen des Grades von f

Unser Ziel ist es, den Grad einer Funktion f zu reduzieren. Dafür suchen wir zwei Funktionen g und h, so dass $h(x) = f(x) * g(x)$ gilt. Dabei betrachten wir drei Szenarien, die die Reduzierung des Grades erlauben:

Szenario 1:
Annahme: Es gibt eine Funktion g mit $grad(g) < grad(f)$, so dass für $h = f * g$ gilt:

$$grad(h) < grad(f)$$

Methode: In diesem Fall betrachtet man die Gleichung

$$\begin{aligned} & f(x)g(x) && = z_i g(x) \\ \Leftrightarrow \quad & h(x) && = z_i g(x) \end{aligned} \tag{3.3}$$

an Stelle von $f(x) = z_i$. Die Gleichung 3.3 hat maximal den Grad:

$$\max\left[grad(g), grad(h)\right] < grad(f)$$

Szenario 2:
Annahme: Es gibt eine Funktion g mit $grad(g) < grad(f)$, so dass gilt:

$$0 = f * g$$

Methode: Hier betrachtet man für $z_i = 1$ die Gleichung

$$\begin{aligned} & f(x)g(x) && = z_i g(x) \\ \Leftrightarrow \quad & 0 && = g(x) \end{aligned}$$

Der Grad dieser Gleichung ist $grad(g) < grad(f)$.

Szenario 3:

Annahme: Angenommen es gibt eine Funktion g mit beliebigem Grad, so dass für $h = f * g$ gilt:

$$grad(h) < grad(f)$$

Methode: Hier betrachtet man für $z_i = 0$ die Gleichung

$$\begin{aligned} & f(x)g(x) && = z_i g(x) \\ \Leftrightarrow \quad & h(x) && = 0 \end{aligned}$$

Der Grad der Gleichung ist durch $grad(h) < grad(f)$ gegeben.

Diese Szenarien sind jedoch redundant. Für Szenario 3 gilt (unter Verwendung von $f = f^2$, da wir uns für Erweiterungskörper des $\mathbb{F}_2$ interessieren):

$$h = f * g = f^2 g = f(fg) = f * h$$

Damit entspricht Szenario 3 dem ersten Szenario. Wenn in Szenario 1 $g \neq h$ gilt, gilt auch für Szenario 1

$$fg = f^2 g = fh \rightarrow f(g + h) = 0$$

. Wenn man nun $g' = g + h$ mit $grad(g') \leq \max[grad(g), grad(h)]$ einsetzt, entspricht Szenario 1 dem zweiten Szenario. Für den anderen in Szenario 1 möglichen Fall, $g = h$, folgt aber:

$$h = f * h \Rightarrow (f + 1)h = 0$$

Dann entspricht Szenario 1 dem zweiten Szenarion für die Funktion $f' = f+1$.

Wir benötigen nun ein Kriterium, das die Widerstandsfähigkeit einer Funktion gegen die Reduzierung des Grades misst. Die Menge der Funktionen g, die in Szenario 2 verwendet werden können, ist als

$$An(f) = \left[g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2 \,|g(x)f(x) = 0, \ \forall x\right]$$

gegeben. Die hinreichende und notwendige Bedingung, damit g in $An(f)$ enthalten ist, ist $f(x) = 1 \Rightarrow g(x) = 0$. Damit gilt insbesondere, dass $g = f + 1 \in An(f)$.

Definition 37 *Die* Algebraische Immunität *von f ist als*

$$AI(f) = \min\left[grad(g) | g \neq 0, \ g \in An(f) \vee g \in An(f + 1)\right]$$

definiert.

Die Algebraische Immunität entspricht genau dem Grad des Gleichungssystems, dass man durch Reduzieren erhalten kann. Unser Ziel ist also, Funktionen mit einer großen Algebraischen Immunität zu finden.

Bestimmen der Algebraischen Immunität

Leider gibt es keine guten theoretischen Aussagen zu der Algebraischen Immunität. Daher kann oft nur die Frage beantwortet werden, ob f eine Algebraische Immunität kleiner als t hat. Dies kann man durch (cleveres) Lösen eines relativ großen Gleichungssystems beantworten. Für $f : \mathbb{F}_2^n \to \mathbb{F}_2,\ n = 32,\ t \leq 5$ lässt sich dieses Verfahren effizient durchführen. Für $n = 32$, und einer algebraischen Immunität $AI \geq 6$ ergibt sich für einen 128 Bit Schlüssel (d.H. $m_{ges} = \sum m_i = 128$) ein Aufwand von $\sim 2^{96}$ Operationen.

3.3 Alternative Konstruktionen von Streamciphern mit LFSR

An dieser Stelle wollen wir kurz auf alternative Möglichkeiten eingehen, die Stromchiffre möglichst nicht-linear zu gestalten.

3.3.1 Nicht-lineare Filter

Idee: Verschiedene Bits **eines** LFSRs werden über eine nicht-lineare boolsche Funktion ($F_2^n \to F_2$) kombiniert. Diese Funktion heißt Filter. Die folgende Abbildung stellt die Idee grafisch dar.

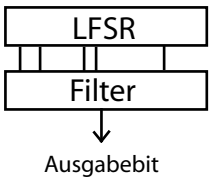

Abbildung 3.6: Prinzip der nicht-linearen Filter

Eigenschaften

- Bei geeigneter Wahl von f hat diese Konstruktion gute statistische Eigenschaften und eine große Periode.
- Angriff, die auf einer Hypothese für Teilschlüssel basieren lassen sich nicht einfach für Filter-Generatoren übernehmen.
- Algebraische Attacken funktionieren für diese Konstruktion genauso wie bei den Combination-Generatoren.

3.3.2 Clock Controlled

Idee: Nichtlinearität durch irreguläres Takten der LFSRs.

Allgemein Es gibt 3 LFSRs ($R1$, $R2$, $R3$). $R1$ wird getaktet, den Ausgang nennen wir z_1. Wenn $z_1 = 0$ ist, dann wird $R2$ getaktet, analog zu $R1$ nennen wir den Ausgang von $R2$ z_2. $R3$ wird nicht getaktet, sondern nur das letzte Ausgabebit von $R3$ wiederholt. Für $z_1 = 1$ gilt der umgekehrte Fall. Der Keystream setzt sich zusammen aus $z_2 \oplus z_3$,

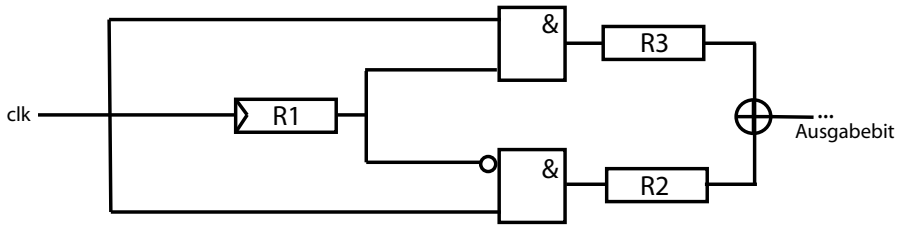

Abbildung 3.7: Clock-Controlled Streamcipher

Eigenschaften

- Bei teilerfremden Perioden der LFSRs ist die Periode (fast immer) das Produkt der Perioden.
- Die Ausgabe hat gute statistische Eigenschaften.
- Die Länge der LFSRs sollte ungefähr gleich groß sein.
- Es gibt Divide-and-Conquer-Angriffe, die den Aufwand 2^l haben, wobei l die Länge des kürzesten LFSR ist $\Rightarrow l \approx 128\ Bit$.

3.3.3 Shrinking Generator

Idee: Der Keystream wird teilweise verzögert ausgegeben, d.h. es gibt Takte in denen keine Ausgabe erfolgt.

Allgemein Es gibt 2 LFSRs ($R1$, $R2$). Beide werden getaktet. Man unterscheidet 2 Fälle:
1. Ausgabe von $R1 = 0 \Rightarrow$ verwerfe Ausgabe von $R2$.
2. Ausgabe von $R1 = 1 \Rightarrow$ Keystream = Ausgabe von R2.

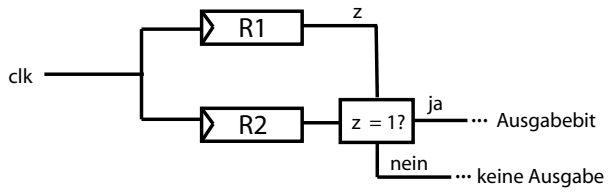

Abbildung 3.8: Prinzip des Shrinking-Generators

Eigenschaften

- Bei teilerfremden Perioden der LFSRs ist die Periode des Keystreams das Produkt der Perioden.
- Für eine zufällige Wahl der (primitiven) charakteristischen Polynome der LFSRs hat der Keystream gute statistische Eigenschaften.

Kapitel 4

Hashfunktionen

4.1 Hashfunktionen

Sei $H : \{0,1\}^* \to \{0,1\}^n$ eine Funktion, die Bitstrings beliebiger Länge auf Bitstrings fester Länge abbildet.

Anwendungsgebiete von Hashfunktionen

Sei sk ein privater (geheimer) Schlüssel und pk der dazugehörige öffentliche Schlüssel. Den Algorithmus um digitale Signaturen zu erzeugen bezeichnen wir mit Sig_{sk} und den entsprechenden Verifikationsalgorithmus mit Ver_{pk}.

Ohne Hashfunktion bestände die Signatur einer Nachricht m aus

$$m||\mathsf{Sig}_{sk}(M)$$

Daraus ergeben sich folgende Nachteile:

- Da die Signatur $\mathsf{Sig}_{sk}(M)$ genauso groß ist, wie die Nachricht selbst, verdoppelt sich das Datenvolumen
- Da die Algorithmen Sig_{sk} und Ver_{pk} im Allgemeinen relativ langsam sind, benötigt die Erzeugung bzw. Verifikation von digitalen Signaturen bei großen Nachrichten sehr viel Zeit

Daher signiert man nicht die Nachricht selbst, sondern man berechnet $\mathsf{Sig}_{sk}\big(H(M)\big)$ und verwendet

$$M||\mathsf{Sig}_{sk}\big(H(M)\big)$$

als Signatur einer Nachricht M.

Da die Signatur nur vom Hashwert der Nachricht abhängt, haben Nachrichten mit dem gleichen Hashwert auch die gleiche Signatur.

Daher ergeben sich folgende Anforderungen an die Hashfunktion.

4.1.1 Eigenschaften guter Hashfunktionen

One-Way Eigenschaft Zu einem gegebenem Hashwert $h \in \{0,1\}^n$ ist es praktisch unmöglich eine Nachricht M zu finden, so dass $H(M) = h$. Diese Eigenschaft wird oft auch "Preimage-Resistance" genannt.

Second-Preimage-Resistance Eigenschaft Zu einer gegebenen Nachricht M ist es praktisch unmöglich eine Nachricht $M' \neq M$ zu finden, so dass $H(M) = H(M')$. Diese Eigenschaft wird auch als "Kollisionsresistenz" bezeichnet.

Collision-Resistance Eigenschaft Es ist praktisch unmöglich beliebige Nachrichten M, M' mit $M \neq M'$ zu finden, so dass $H(M) = H(M')$. Diese Eigenschaft bezeichnet man oft auch als "starke Kollisionsresistenz".

Die Collision-Resistance impliziert die Second-Preimage Eigenschaft. Im Allgemeinen impliziert die Second-Preimage Eigenschaft jedoch nicht die One-Way Eigenschaft.

4.1.2 Generische Angriffe auf Hashfunktionen

Unter der Annahme, dass sich die Funktion $H : \{0,1\}^* \to \{0,1\}^n$ wie eine zufällige Funktion verhält, ergibt sich mit dem Geburtstagsparadoxon ein Angriff gegen die Collision-Resistance Eigenschaft. Der mittlere Aufwand dieses Angriffs beträgt $O(2^{n/2})$.

Gegen die One-Way und die Second-Preimage Eigenschaft sind keine besseren Angriffe als Brute Force bekannt.

Um gegen generische Angriffe in der Praxis immun zu sein, sollte der Hashwert daher mindestens 160 Bit lang sein, d.h. $n \geq 160$.

4.1.3 Iterierte Hashfunktionen

Idee Nutze eine Funktion $g : \{0,1\}^k \times \{0,1\}^l \to \{0,1\}^n$ mit guten Eigenschaften (s. Abschnitt 4.1.1) um eine Funktion $H : \{0,1\}^* \to \{0,1\}^n$ zu konstruieren.

Vorgehendweise Abbildung 4.1 gibt einen graphischen Überblick über die Funktionsweise.

Zuerst wird die zu hashende Nachricht M auf ein vielfaches der Bitlänge l aufgefüllt ("Padding"). Die aufgefüllte Nachricht X wird anschließend in

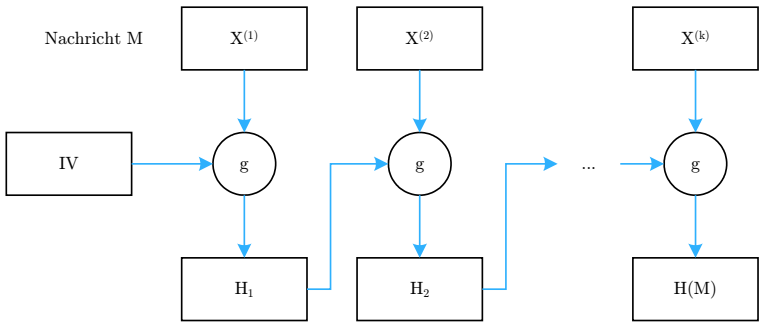

Abbildung 4.1: Funktionsweise einer iterierten Hashfunktion

Blöcke der Länge l aufgeteilt.

$$X = X^{(1)}||X^{(2)}||\ldots||X^{(k)}$$

Zu jedem spezifizierten Wert $IV \in \{0,1\}^K$ berechnet man

$$H_1 = g(IV, X^{(1)})$$

und anschließend

$$H_i = g(H_{i-1}, X^{(i)}) \qquad \text{für} \qquad i \in \{2, \ldots, k\}$$

Der resultierende Hashwert ist dann

$$H(M) := H_k$$

Padding Padding ist das Auffüllen einer Nachricht, so dass diese eine bestimmte Länge erreicht.

Nach einer Idee von Merkel und Damgard füllt man eine Nachricht auf ein Vielfaches der Blocklänge auf, indem man die (binäre) Länge der Nachricht anhängt und mit Nullen auffüllt:

$$M \rightarrow \underbrace{M||0\ldots0||}_{k \cdot l \text{ Bit}}|M|$$

Definition 38 *Eine Funktion* $g : \{0,1\}^k \times \{0,1\}^l \rightarrow \{0,1\}^n$ *heißt* pseudo kollisionsresistent*, falls es praktisch unmöglich ist, Werte* (IV, h) *und* (IV, h') *zu finden, so dass*

$$g(IV', h) = g(IV', h') \qquad \textit{und} \qquad (IV, h) \neq (IV', h')$$

(d.h. $IV \neq IV'$ *oder* $H \neq H'$*).*

Satz 39 *g pseudo kollisionsresistent* $\Rightarrow$ *h kollisionsresistent*

4.2 Hashfunktionen der MD4-Familie

Die Hashfunktionen der MD4-Familie sind iterative Hashfunktionen.

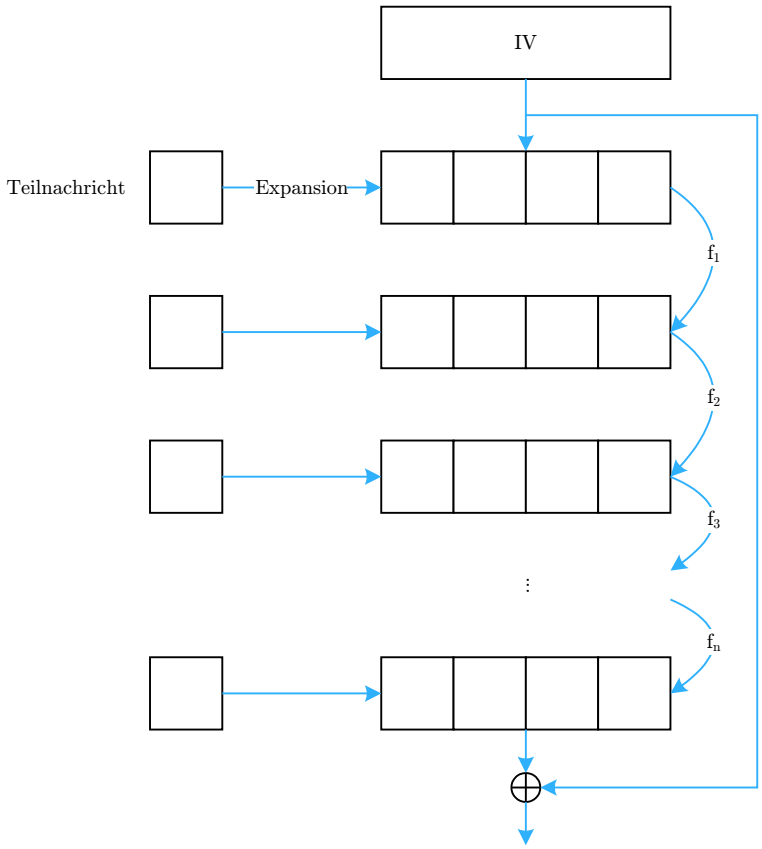

Abbildung 4.2: Aufbau der Hashfunktionen der MD4-Familie

Die (Teil-) Nachricht wird durch Wiederholung (MD4, MD5, RIPEMD, RIPEMD-160) oder Rekursion (SHA-X) verlängert, so dass für jede Runde ein Nachrichtenblock zur Verfügung steht. In jeder Runde wird auf den aktuellen Zustand eine Rundenfunktion f_i angewendet. Diese Funktion f_i besteht aus einfachen boolschen Funktionen, zyklischen Shifts und Additionen modulo 2^{32} bzw. 2^{64}.

4.3 Aktueller Stand der Sicherheit von Hashfunktionen

MD4 Entwickelt von Rivest (1990) und gebrochen (d.h. Kollisionen wurden gefunden) von Dobbertin in 1995.

RIPEMD Entwickelt zwischen 1988 und 1992. 1995 gebrochen von Dobbertin (Kollisionen für zwei von drei Runden gefunden).

RIPEMD-128, 160, 256, 320 1996 entwickelt von Dobbertin Preenel und Bosselares. Bisher noch nicht gebrochen.

MD5 Entwickelt von Rivest in 1990 und gebrochen von Wang et. al. in 2004 (Kollisionen können innerhalb weniger Stunden gefunden werden).

SHA-1 1993 entwickelt von der National Security Agency (NSA). Wang et. al. entwickelten Angriffe mit Aufwand 2^{69} und (ungeprüft) 2^{63} SHA-1 Operationen.

SHA-256, 384, 512 Entwickelt von der NSA in 2002 und 2004. Bisher noch nicht gebrochen.

4.4 Praxisrelevanz von zufälligen Kollisionen

Die gefundenen Kollisionen sind Bitstrings ohne Bedeutung. Es ist bisher nicht möglich, sinnvolle Bitstrings zu erzeugen, die den selben Hashwert haben. Dennoch haben diese Kollisionen, sogar eine einzige, große praktische Bedeutung.

Bsp. mit PostScript

Anhang A

Kurze Wiederholung bekannten Stoffes

A.1 Wiederholung endlicher Körper

Wir wollen eine kurze Wiederholung endlicher Körper bieten. Der Stoff sollte bereits bekannt sein, da dies keine ausführliche Behandlung des Themas ist.

A.1.1 Zentrale Definitionen

Definition 40 *Sei das Tupel $G = \langle M, \circ \rangle$ gegeben, wobei M eine Menge und $\circ$ eine Operation auf M ist. G wird als* Gruppe *bezeichnet, wenn folgende Eigenschaften erfüllt sind:*

- *Das Assoziativgesetz gilt für $\circ$.*
- *Für $\circ$ auf M ist e das neutrale Element.*
- *Für jedes Element $a \in M$ gibt es ein inverses Element bezüglich $\circ$.*

Wenn die Verknüpfung $\circ$ auf M kommutativ ist, wird G als *abelsche Gruppe* bezeichnet.

Definition 41 *Sei das Tripel $\mathbb{F} = \langle M, \oplus, \odot \rangle$ gegeben, wobei M eine Menge ist und $\oplus$ sowie $\odot$ zweistellige Operationen auf M sind. $\mathbb{F}$ wird als* Körper *bzw.* Field *bezeichnet, wenn folgende Bedingung erfüllt sind:*

- *Das Assoziativgesetz gilt für $\oplus$ und $\odot$.*
- *Das Kommutativgesetz gilt für $\oplus$ und $\odot$.*

- *Das Distributivgesetz gilt für* $\oplus$ *und* $\odot$
- *Für* $\oplus$ *auf* M *ist* e *das neutrale Element.*
- *Für jedes Element* $a \in M$ *gibt es ein inverses Element bezüglich* $\oplus$*.*
- *Für* $\odot$ *auf* $M \setminus \{0\}$ *ist 1 das neutrale Element.*
- *Für jedes Element* $a \in M \setminus \{0\}$ *gibt es ein inverses Element* a^{-1} *bezüglich* $\odot$*.*

Man hätte äquivalent auch fordern können, dass $\langle M, \oplus \rangle$ und $\langle M \setminus \{0\}, \odot \rangle$ abelsche Gruppen sind und dass das Distributivgesetz gilt. Wenn die Ordnung von $\mathbb{F}$ endlich ist, also $\mid \mathbb{F} \mid < \infty$ gilt, spricht man von einem *endlichen Körper* bzw. einem *Galois Field.*

Definition 42 *Ein Polynom* $h(x) \in \mathbb{F}[x]$ *heißt in dem Körper* $\mathbb{F}[x]$ irreduzibel*, wenn für alle Polynome* $f(x), g(x) \in \mathbb{F}[x]$ *aus der Gleichung* $h(x) = f(x) * g(x)$ *folgt, dass entweder* $grad(f) = 0$ *oder* $grad(g) = 0$ *gilt.*

Dies bedeutet auch, dass irreduzible Polynome in ihrem Körper keine Nullstellen haben. Das Polynom $h(x) = x^2 + x + 1$ hat in $\mathbb{Z}_2$ keine Nullstellen, ist dort also irreduzibel. In $\mathbb{Z}_3$ hat $h(x)$ jedoch eine Nullstelle für den Wert 1. Das bedeutet, dass $(x+2)|(x^2+x+1)$ in $\mathbb{Z}_3$. Tatsächlich gilt in $\mathbb{Z}_3$: $\quad x^2 + x + 1 = (x+2)(x+2)$.

A.1.2 Wichtige Eigenschaften von Gruppen

G1 Sei G eine Gruppe. Dann ist für ein Element $a \in G$ die Ordnung von a als $ord(a) = min\{r | a^r = e\}$ definiert. Wenn kein solches r existiert, wird $r = \infty$ gesetzt. Wenn G endlich ist, ist auch $\forall a \in G$ die Ordnung von a endlich.

G2 In einer Gruppe G gilt für alle Elemente a mit $ord(a) \neq \infty$: $a^k = e \Leftrightarrow ord(a)|k$

G3 In einer abelschen Gruppe G mit zwei Elementen a, b endlicher Ordnung und $ggt\left(ord\left(a\right), ord\left(a\right)\right) = 1$ gilt $ord(a \circ b) = ord(a) * ord(b)$.

G4 In einer endlichen, abelschen Gruppe G mit zwei Element $a, b \in G$ wobei $ord(a) = max\{ord(c) | \forall c \in G\}$ gilt, gilt $\forall b \in G : \ ord(b)|ord(a)$.

G5 Für eine endliche Gruppe G gilt: $ord(a) \Big| |G| \qquad \forall a \in G$.

G6 Für eine endliche Gruppe G und ein Element $a \in G$ bildet $\langle \{a^0, a^1, ..., a^{ord(a)-1}\}, \circ \rangle$ die kleinste Untergruppe von G, die a enthält.

G7 Für zyklische, endliche Gruppen G und ein Element $a \in G$ gilt $ord(a) = ord(a^d)$ wenn $ggt\,(d, |G|) = 1$.

A.1.3 Wichtige Eigenschaften von endlichen Körpern

F1 $\mathbb{F}_q$ mit $q = p^n$ ist genau dann ein Körper, wenn p eine Primzahl und n eine natürliche Zahl ist.

F2 Für den Körper $\mathbb{F}_{p^n}$ mit der Primzahl p und der natürlichen Zahl n wird p als *Chararkteristik* bezeichnet.

F3 Kleiner Satz von Fermat, auf alle Körper verallgemeinert: Sei $q = p^n$, p eine Primzahl und $n \in \mathbb{N}$. Dann gilt:

$$\forall a \in \mathbb{F}_q,\ a \neq 0 : \qquad a^q = a$$

und

$$\forall a \in \mathbb{F}_q,\ a \neq 0 : \qquad a^{q-1} = 1$$

F4 Es gilt $\mathbb{F}_q = \frac{\mathbb{Z}_p}{(f)}$ genau dann, wenn (f) ein beliebiges irreduzibeles Polynom vom Grad n in $\mathbb{Z}_p$ ist.

F5 Die Ordnung jedes endlichen Körpers ist eine Primzahlpotenz $q > 1$.

F6 Für jede Primzahlpotenz $q > 1$ existiert ein Körper der Ordnung q.

F7 $\langle M \setminus \{0\}, \odot \rangle$ bildet eine zyklische Gruppe.

F8 Für alle normierten, irreduziblen Polynome f in $\mathbb{Z}_p$, deren Grad n teilt, gilt: $\prod_f = X^{p_n} - X$

F9 Für zwei Körper K_1 und K_2 gilt: $|K_1| = |K_2| \Leftrightarrow K_1 \cong K_2$. Zwei Körper gleicher Größe sind also isomorph zu einenader.

F10 Für zwei Körper $G = \mathbb{F}_{p^n}$ und $H = \mathbb{F}_{q^m}$ (p, q sind Primzahlen, $n, m \in \mathbb{N}$) ist G genau dann ein Teilkörper von H ($G \subseteq H$) wenn $p = q$ und $n|m$ gilt.

Wir wollen diese Eigenschaften hier nicht beweisen. Die Beweise sollten bereits bekannt sein. Man kann sie auch unter [McEliece 87], [Honold 04/05] oder [Steger 02] nachlesen.

A.1.4 Aufgabe: Komponentenfunktionen

Sei $f : \mathbb{F}_{2^2} \to \mathbb{F}_{2^2}$ mit

$$f(x) = \begin{cases} x^{-1} & x \neq 0 \\ 0 & x = 0 \end{cases}$$

gegeben.

1 Berechnen sie zwei Komponentenfunktionen f_1 und f_2, mittels derer f als Abbildung von $\mathbb{F}_2^2 \to \mathbb{F}_2^2$ berechnet werden kann. Nutzen Sie hierfür das irreduzible Polynom

$$p(\alpha) = \alpha^2 + \alpha + 1$$

2 Bestimmen sie die Menge

$$Bild(f) := \{f(x) | x \in \mathbb{F}_{2^2}\}$$

Lösung für 1: Wir müssen die Zahl aus dem Körper $\mathbb{F}_{2^2}$ in den Vektorraum $\mathbb{F}_2^2$ abbilden, dort invertieren und das Ergebnis wieder in den Körper $\mathbb{F}_{2^2}$ abbilden:

$$\begin{array}{ccc} & \overset{f}{} & \\ \mathbb{F}_{2^2} & \longrightarrow & \mathbb{F}_{2^2} \\ f_1 \downarrow & & \uparrow f_2 \\ \mathbb{F}_2^2 & \longrightarrow & \mathbb{F}_2^2 \\ & \underset{g}{} & \end{array}$$

Dazu definieren $f_1(x) : \ \forall x \in \mathbb{F}_{2^2},\ x = a_1\alpha + a_0$ folgendermaßen:

$$f_1(x) = \begin{pmatrix} a_1 \\ a_0 \end{pmatrix}$$

Das bedeutet für die Funktion g:

$$g \circ f_1(x) = \begin{pmatrix} g_1(a_1, a_0) \\ g_2(a_1, a_0) \end{pmatrix}$$

Und für die Funktion f_2 bedeutet das:

$$\begin{aligned} f_2 \circ g \circ f_1(x) = &\ f_1^{-1} \circ g \circ f_1 = &\ f(x) \\ f_2 \begin{pmatrix} b_1 \\ b_0 \end{pmatrix} = &\ f_1^{-1} \begin{pmatrix} b_1 \\ b_0 \end{pmatrix} = &\ b_1\alpha + b_0 \end{aligned}$$

Nun müssen wir also noch g bestimmen. Dazu benutzen wir die Tatsache, dass im $\mathbb{F}_{2^2}$ die Gleichung $x^{-1} = x^2$ gilt. (Wem das nicht klar ist, soll sich den $\mathbb{F}_{2^2}$ mal konstruieren und dort $x * x^2$ berechnen!) Wir berechnen also:

$$\begin{aligned} x^2 &= (a_1\alpha + a_0)(a_1\alpha + a_0) \\ &= (a_1\alpha)^2 + \underbrace{2a_1a_0\alpha}_{=0} + a_0^2 \\ &= a_1\alpha^2 + a_0 \\ &= a_1(\alpha + 1) + a_0 \\ &= \underbrace{a_1}_{b_1}\alpha + \underbrace{(a_1 + a_0)}_{b_0} \end{aligned}$$

(Anmerkung: $a_{1,0} \in \mathbb{F}_2 \Rightarrow a_{1,0}^2 = a_{1,0}$) Damit haben wir nun g ausreichend bestimmt:

$$\begin{aligned} g \circ f_1(x) &= \begin{pmatrix} g_1(a_1, a_0) \\ g_2(a_1, a_0) \end{pmatrix} \\ &= \begin{pmatrix} a_1 \\ a_1 + a_0 \end{pmatrix} \\ g_1(a_1, a_0) &= a_1 \\ g_2(a_1, a_0) &= a_1 + a_0 \end{aligned}$$

Lösung für 2: Wir können relativ einfach zeigen, dass f bijektiv ist (die Elemente sind binär kodiert):

x	**Bild**(f)
00	00
01	01
10	11
11	10

A.1.5 Aufgabe: Isomorphe Körper

Seien die in $\mathbb{F}_2[X]$ irreduziblen Polynome $f(X) = X^3 + X + 1$ und $g(X) = X^3 + X^2 + 1$ gegeben. Zeigen Sie, dass die beiden Körper

$$K_1 = \frac{\mathbb{F}_2[X]}{(f)}$$

und

$$K_2 = \frac{\mathbb{F}_2[X]}{(g)}$$

isomorph sind ohne sich auf Eigenschaft F9 zu berufen.

Lösung: Zuerst betrachten wir die Elemente der beiden Körper:

$\mathbf{K_1}$	$\mathbf{K_2}$
0	0
1	1
α	α
α^2	α^2
$\alpha+1$	α^2+1
$\alpha^2+\alpha$	$\alpha^2+\alpha+1$
$\alpha^2+\alpha+1$	$\alpha+1$
α^2+1	$\alpha^2+\alpha$

Hierbei kann man intuitiv nachvollziehen, dass es einen Isomorphismus geben muss. Ein Möglicher Isomorphismus ist:

$$\phi:\ K_1 \longrightarrow K_2, x \mapsto y+1$$

Die Umkehrfunktion existiert natürlich auch:

$$\phi^{-1}:\ K_2 \longrightarrow K_1, y \mapsto x+1$$

Nun prüfen wir, ob die Nullstellen der irreduziblen Polynome tatsächlich im anderen Körper enthalten sind:

$$\begin{aligned}
\phi(x) &= y+1 \\
\phi(x^2) &= y^2+1 \\
\phi(x^2+x) &= y^2+y+1 \\
\phi^{-1}(y) &= x+1 \\
\phi^{-1}(y^2) &= y^2+1 \\
\phi^{-1}(y^2+y+1) &= y^2+y
\end{aligned}$$

Dass sich die Abbildung bijektiv verhält, ist durch leichtes nachrechnen zu überprüfen.

A.1.6 Aufgabe: Teilkörper

Welche Teilkörper K sind in $\mathbb{F}_{2^{12}}$ enthalten?

Lösung: Die Lösung ist mit Körpereigenschaft F10 relativ einfach:

$$\begin{aligned}
\mathbb{F}_{2^1} &\subset \mathbb{F}_{2^{12}} \\
\mathbb{F}_{2^2} &\subset \mathbb{F}_{2^{12}} \\
\mathbb{F}_{2^3} &\subset \mathbb{F}_{2^{12}} \\
\mathbb{F}_{2^4} &\subset \mathbb{F}_{2^{12}} \\
\mathbb{F}_{2^6} &\subset \mathbb{F}_{2^{12}}
\end{aligned}$$

A.1.7 Aufgabe: Irreduzible Polynome

Bestimmen Sie alle irreduziblen Polynome vom Grad kleiner gleich 3 über die Körper $\mathbb{F}_2$ und F_3.

Lösung: Die Polynome sind binär kodiert, für den $\mathbb{F}_2$ sind alle Polynome angegeben, für den $\mathbb{F}_3$ nur die irrduziblen:

$\mathbf{x^3}$	$\mathbf{x^2}$	$\mathbf{x^1}$	$\mathbf{x^0}$	F_2	$\mathbf{x^3}$	$\mathbf{x^2}$	$\mathbf{x^1}$	$\mathbf{x^0}$	F_3
0	**0**	**0**	**0**		**0**	**0**	**0**	**1**	√
0	**0**	**0**	**1**	√	**0**	**0**	**0**	**2**	√
0	**0**	**1**	**0**		**0**	**1**	**0**	**1**	√
0	**0**	**1**	**1**		**0**	**1**	**1**	**2**	√
0	**1**	**0**	**0**		**0**	**1**	**2**	**2**	√
0	**1**	**0**	**1**		**0**	**2**	**0**	**2**	√
0	**1**	**1**	**0**		**0**	**2**	**1**	**1**	√
0	**1**	**1**	**1**	√	**0**	**2**	**2**	**1**	√
1	**0**	**0**	**0**		**1**	**0**	**2**	**1**	√
1	**0**	**0**	**1**		**1**	**0**	**2**	**2**	√
1	**0**	**1**	**0**		**1**	**1**	**0**	**2**	√
1	**0**	**1**	**1**	√	**1**	**1**	**1**	**2**	√
1	**1**	**0**	**0**		**1**	**1**	**2**	**1**	√
1	**1**	**0**	**1**	√	**1**	**2**	**0**	**1**	√
1	**1**	**1**	**0**		**1**	**2**	**1**	**1**	√
1	**1**	**1**	**1**		**1**	**2**	**2**	**2**	√
					2	**0**	**1**	**1**	√
					2	**0**	**1**	**2**	√
					2	**1**	**0**	**2**	√
					2	**1**	**1**	**1**	√
					2	**1**	**2**	**2**	√
					2	**2**	**0**	**1**	√
					2	**2**	**1**	**2**	√
					2	**2**	**2**	**1**	√

A.1.8 Aufgabe: Rechnen in endlichen Körpern

Berechnen Sie für den Körper $\mathbb{F}_{2^3}$ eine Additions- und eine Multiplikationstabelle. Nutzen Sie für die Darstellung einers der irreduziblen Polynome aus der vorhergehenden Aufgabe.

Lösung: Wir nutzen für die Darstellung des $\mathbb{F}_{2^3}$ das irreduzible Polynom $x^3 + x + 1$. Das bedeutet, dass $\alpha^3 = \alpha + 1$. Die Elemente des Körpers sind

also: $\{0,\ 1,\ \alpha,\ \alpha^2,\ \alpha+1,\ \alpha^2+\alpha,\ \alpha^2+\alpha+1,\ \alpha^2+1\}$. Daraus ergeben sich die gesuchten Tabellen (in binärer Kodierung):

+	**000**	**001**	**010**	**100**	**011**	**110**	**111**	**101**
000	000	001	010	100	011	110	111	101
001	001	000	011	101	010	111	110	100
010	010	011	000	110	001	100	101	111
100	100	101	110	000	111	010	011	001
011	011	010	001	111	000	101	100	110
110	110	111	100	010	101	000	001	011
111	111	110	101	011	100	001	000	010
101	101	100	111	001	110	011	010	000

*	**000**	**001**	**010**	**100**	**011**	**110**	**111**	**101**
000	000	000	000	000	000	000	000	000
001	000	001	010	100	011	110	111	101
010	000	010	100	011	110	111	101	001
100	000	100	011	110	111	101	001	010
011	000	011	110	111	101	001	010	100
110	000	110	111	101	001	010	100	011
111	000	111	101	001	010	100	011	110
101	000	101	001	010	100	011	110	111

A.1.9 Aufgabe: Erweiterter Euklidischer Algorithmus

Berechnen Sie für die Polynome $a, b \in \mathbb{F}_2[X]$ mit

$$a = x^4 + x^3 + 1$$

und

$$b = x^3 + x^2 + 1$$

Polynome $c, d \in \mathbb{F}_2[X]$ so dass

$$ac + bd = ggt(a, b)$$

gilt.

Lösung: Zunächst fällt auf, dass beide Polynome in $\mathbb{F}_2[X]$ irreduzibel sind. Daher muss $ggt(a, b) = 1$ gelten.

$$\begin{aligned} a = b * x + (x+1) &\Rightarrow (x+1) = a + bx \\ b = (x+1) * x^2 + 1 &\Rightarrow 1 = b + (x+1) * x^2 \\ &\Rightarrow 1 = b + (a + bx) * x^2 \\ &\Rightarrow 1 = a * \underbrace{x^2}_{c} + b * \underbrace{(x^3+1)}_{d} \end{aligned}$$

Anmerkung: Wir nutzen hier aus, das im $\mathbb{F}_2[X]$ plus und minus äquivalent sind. In anderen Grundkörpern verrechnet man sich bei der Polynomdivision mit den additiven Inversen sehr leicht.

Anhang B

Wörterbuch

Hier sammeln wir einige wichtige Ausdrücke, die entweder vorrausgesetzt werden oder sonst nirgends richtig hinpassen, nicht triviale Übersetzungen ins Englische und Abkürzungen.

ANF *Algebraische Normalform*, siehe Kapitel 3.2.8

bijektiv Eine bijektive Funktion ist gleichzeitig injektiv und surjektiv.

FFT *Fast Fourier Transformation*, kann für die Berechnung der Walsh-Koeffizienten verwendet werden.

injektiv Eine Funktion ist injektiv, wenn alle Elemente der Ausgangsmenge paarweise unterschiedliche Bilder haben.

Lemma Ein Lemma verhält sich wie ein Theorem, ist aber nicht ganz so wichtig. Lemmas werden manchmal auch „Hilfssatz“ genannt.

Monom In der Algebra ist ein Monom ein Polynom, das nur aus Produkten und Potenzen der Variablen und Koeffizienten besteht; zum Beispiel ein Term der Form ax^iy^j. Jedes Polynom ist aus Monomen zusammengesetzt. (Kopiert von 'de.wikipedia.org')

root Die Nullstellen eines Polynoms in einem endlichen Körper werden im Englischen *root* genannt.

surjektiv Eine Funktion ist surjektiv, wenn für jedes Element der Zielmenge mindestens ein Urbild existiert.

Theorem Ein Theorem (engl.: „theorem “) ist mit einem Satz gleich bedeutend.

Literaturverzeichnis

[FIPS 46-3] NIST: *Data Encryption Standard (DES)*, FIPS PUB 46-3, http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, 1999

[Schneier 96] Bruce Schneier: *Applied Cryptography, Second Edition*, John Wiley & Sons, Inc., 1996

[Coppersmith 94] Don Coppersmith: *The data encryption standard (DES) and its strength against attacks.*, IBM Journal of Research and Development, 38(3), Seite 243?250, http://www.research.ibm.com/journal/rd/383/coppersmith.pdf, 1994

[Beutelspacher 05] Albrecht Beutelspacher u.a.: *Kryptografie in Theorie und Praxis*, Vieweg, 2005

[FIPS 197] NIST: *Specification for the Advanced Encryption Standard (AES)*, FIPS PUB 197, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001

[Daemen 02] Joan Daemen and Vincent Rijmen, *The design of Rijndael: AES – The Advanced encryption Standard*, Springer-Verlag, 2002

[Daemen 99] Joan Daemen and Vincent Rijmen, *AES proposal: Rijndael*, http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf, 1999

[Miller 03] Michael Miller, *Symmetrische Verschlüsselungsverfahren*, Teubner, 2003

[McEliece 87] Robert J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Springer, 1987

[Steger 02] Prof. Dr. Angelika Steger, *Diskrete Strukturen 1*, Springer-Verlag Berlin Heidelberg New York, 2002

[Honold 04/05] Prof. Dr. Thomas Honold, *Endliche Körper und ihre Anwendungen*, http://gd.tuwien.ac.at/uni/skripten/collection-rbenedik/Algebra/Endliche_Koerper_und_Anwendugen_001_g_122p.pdf, 2004/2005