Analyse und Design von symmetrischen Blockchiffrierern

Tim Schneider
Matr.Nr.: 1085258
Blendermannweg 44
28279 Bremen

Diplomarbeit
gemäß DPO Mathematik vom 10.06.1985

Universität Bremen
Fachbereich 3
Mathematik und Informatik
13. August 1999

Zusammenfassung

Diese Arbeit befaßt sich mit dem Design und der Analyse von symmetrischen Blockchiffrierern. Es ist für das Verstehen dieser Arbeit nur ein Mathematikvordiplom vonnöten, um die mathematischen Ansätze begreifen zu können. Der erste Teil bereitet die Theorie symmetrischer Blockchiffrierer bis Mitte der 80er Jahre locker und gut lesbar auf. Dies umfaßt die nötigen Grundbegriffe und Strukturen, sowie elementare Designkriterien und Analysen. Zuletzt werden die zwei wichtigsten Blockchiffrierer definiert, der Data Encryption Standard (DES) und FEAL.

Der zweite Teil behandelt die moderne Kryptologie, die durch die Entdeckung zweier wichtiger Analysemethoden begründet wurde. Diese Methoden sind die differentielle und lineare Kryptoanalyse, die im Detail erläutert werden. Aus diesen Angriffen entwickelten sich viele neue Designkriterien, über die ein Überblick gegeben wird. Schließlich wird ein AES-Kandidat vorgestellt, der zu den modernsten Blockchiffrierern zählt.
Schlüsselwörter: symmetrische Blockchiffrierer, Designkriterien, Lineare Kryptoanalyse, Differentielle Kryptoanalyse, DES, FEAL, AES, RC6

Inhalt im Überblick

I Klassische Verfahren und Designkriterien 1

1 Grundbegriffe . . . 3
2 Elementare Analysen und Designkriterien . . . 19
3 Die Blockchiffrierer DES und FEAL . . . 31

II Moderne Verfahren und Designkriterien 43

4 Differentielle Kryptoanalyse . . . 45
5 Lineare Kryptoanalyse . . . 65
6 Designkriterien mit beweisbarer Sicherheit . . . 83
7 Advanced Encryption Standard . . . 93

III Anhang 103

A Das Geburtstagsparadoxon . . . 105
B Hyperlinks und Inhalt der CD . . . 109

Vorwort

Vor der Mitte dieses Jahrhunderts gab es praktisch keine öffentliche Diskussion über Kryptologie, das war Sache von Militärs oder anderer Geheimniskrämer. Die erste ernsthafte Arbeit über Kryptologie stellt das heute grundlegende Werk von C.E. Shannon [Shan49] dar, welches auch heute noch die fundamentalen Prinzipien für die Konstruktion von Kryptosystemen definiert. Die klassische Kryptologie wurde durch das Werk „The Codebreakers" von D. Kahn [Kahn67] aufgearbeitet. Nach diesem Buch begann allmählich das Interesse der Forschung an der Kryptologie zu wachsen. Einige Werke, wie z.B. von H. Feistel [Feis75] Mitte der 70er Jahre, stellten grundlegende Designkriterien und Strukturen für Blockchiffrierer bereit, die heute noch in der Anwendung sind.

Der Durchbruch für eine breite Anwendung der Kryptologie erfolgte durch die Entdeckung sogenannter **Public Key Verfahren**, oder präziser ausgedrückt, durch die Entwicklung **asymmetrischer** Verschlüsselungsmethoden, durch die neben der Wirtschaft auch immer mehr Privatpersonen Datensicherheit in ihrer Kommunikation zu schätzen wissen. Diese Methoden unterscheiden sich von **symmetrischen** Verfahren durch die Art der Schlüssel. Während bei symmetrischen Verfahren der gleiche oder zumindest leicht ableitbare Schlüssel sowohl zur Ver- wie auch Entschlüsselung verwendet wird, so ist der Schlüssel bei asymmetrischen Verfahren bei Ver- und Entschlüsselung verschieden, und eine Ableitung des einen Schlüssels vom anderen beruht auf der Lösung eines schwierigen mathematischen Problems. So wurden in der Vergangenheit asymmetrische Kryptosysteme vorgestellt, die auf dem diskreten Logarithmusproblem beruhen, auf der Primfaktorzerlegung sehr großer Zahlen oder die Gruppenstrukturen auf elliptischen Kurven benutzen. Wenngleich die Mathematik hinter diesen Verfahren sehr raffiniert ist, so spielen asymmetrische Verfahren doch eine eher untergeordnete Rolle, da diese Verschlüsselungsmethoden zu langsam sind, um im großen Stil eingesetzt werden zu können.

Diese Arbeit befaßt sich daher mit dem Design und der Analyse von symmetrischen Chiffrierern, oder genauer, mit symmetrischen Blockchiffrierern. Blockchiffrierer operieren im Gegensatz zu zeichenorientierten Verfahren auf Blöcken von 8 oder 16 Byte Länge. Um möglichst hohe Verschlüsselungsraten zu erzielen, arbeiten diese Verfahren mit einfachen Verfahren, wie z.B. logischen Operationen (bitweises XOR, AND, OR oder Integeraddition und -multiplikation) und durch Wertetabellen realisierte komplexe Boolesche Funktionen. Diese einfachen Operationen werden massiv eingesetzt, um Abhängigkeiten so komplex wie möglich zu gestalten. Vor allem ein hoher Anteil an nichtlinearen Elementen macht die Analyse eines Chiffrierers schwieriger, und alle modernen Verfahren sind gegen naive Angriffe wie Schlüsselsuche oder einfache Analyseverfahren, wie ich sie in Kapitel 2 vorstellen werde, gefeit. In den letzten Jahren wurden dagegen neue Kryptoanalyseverfahren vorgestellt, die mathematisch raffinierter sind; angefangen mit der differentiellen Kryptoanalyse von 1990, die „Differenzen" von Vektoren betrachtet, oder die 1993 vorgestellte Lineare Kryptoanalyse, die versucht, lineare Approximationen des Chiffrierers zu nutzen.

Einige meine Arbeit streifende Begriffe kann ich aus Platzgründen nicht weiter ausführen, da jedes Thema für sich den Umfang einer Diplomarbeit ausmacht, dies betrifft die Klasse der Stromchiffren und Hashfunktionen, sowie kryptographische Protokolle und Schlüsselmanagment.

Tim Schneider
Bremen, 13. August 1999

Inhalt

Meine Arbeit ist in zwei Teile gegliedert; im ersten Teil werde ich mich den Grundbegriffen und klassischen Verfahren widmen, die die Grundlage für die moderne Kryptologie darstellen.

Den grundlegenden Strukturen und Begriffen, wie sie Shannon und Feistel in ihren Werken definiert haben, ist Kapitel 1 gewidmet. Die Feistel-Struktur ist z.B. auch heute noch ein weitverbreiterter Ansatz für die Struktur eines Blockchiffrierers.

Kapitel 2 ist den elementaren Designkriterien und Angriffen gewidmet. Diese einfachen Angriffe machen sich Schwächen im Input/Outputverhalten zunutze, ohne die interne Struktur des Chiffrierers zu kennen. Alle modernen Chiffrierer erfüllen die daraus resultierenden Designkriterien und sind daher gegen diese Angriffe immun.

Mitte der 70er Jahre begann mit dem Aufkommen erschwinglicher Computer das Interesse der Wirtschaft an geschützter Kommunikation zu wachsen, was schließlich in der Ausschreibung um den Data Encryption Standard (DES) gipfelte. Ich werde den DES sowie den 1987 definierten Chiffrierer FEAL in Kapitel 3 vorstellen. Der FEAL war besonders aufgrund seiner einfach zu analysierenden Struktur das erste Angriffsziel der modernen Analysen aus dem zweiten Teil.

Im zweiten Teil meiner Arbeit widme ich mich den modernen Analysen und Designkriterien, wie sie seit Anfang der 90er entwickelt werden.

Die differentielle und lineare Kryptoanalyse und ihre Erweiterungen sind die zur Zeit mächtigsten Werkzeuge zur Brechung von symmetrischen Blockchiffrierern. Auch wenn sie vielleicht eher von akademischen Interesse sind, da sie teilweise enorme Mengen an Speicher benötigen, oder eher unpraktischer Voraussetzungen bedürfen, wie z.B. ein Angriff mit frei wählbarem Klartext, so haben diese Verfahren doch die Designkriterien für Blockchiffrierer wesentlich erweitert und das Verständnis für das „diskrete Chaos", welches Kryptosysteme verwirklichen, vertieft.

Ich werde zunächst in Kapitel 4 die differentielle Kryptoanalyse vorstellen, zwei Beispiele einer Analyse vornehmen und danach Aufwandsabschätzungen für die in dieser Arbeit vorgestellten Chiffrierer angeben. Zuletzt werde ich als Erweiterung dieses Verfahrens die Analyse mit gekürzten Differenzen und Differenzen höherer Ordnung kurz vorstellen.

Kapitel 5 ist der linearen Kryptoanalyse gewidmet. Auch für dieses Verfahren werde ich ein Beispiel präsentieren sowie Aufwandsabschätzungen für die in Kapitel 3 vorgestellten Chiffrierer geben. Ich schließe dieses Kapitel mit der Differentiell-Linearen Kryptoanalyse, einer Erweiterung, die beide Verfahren verbindet.

In Kapitel 6 werde ich Designkriterien für Blockchiffrierer entwickeln, die eine beweisbare Sicherheit gegen die oben genannten Verfahren sichern. Diese Designkriterien richten sich hauptsächlich an die Gestalt der Rundenfunktion und der S-Boxen, der nichtlinearen Anteile eines Chiffrierers.

Diese Designkriterien finden Einzug in eine neue Generation von Blockchiffrierern, wovon einige sich zur Zeit in einer Ausschreibung um den neuen Verschlüsselungsstandard AES befinden. Einen der AES Kandidaten, den RC6, werde ich in Kapitel 7 vorstellen.

Im Anhang werde ich das für die Kryptologie äußerst wichtige Geburtstagsparadoxon erläutern und den Erwartungswert für große Grundmengen herleiten. Anschließend stelle ich für die Online-Version meiner Arbeit im zweiten Anhang einige Hyperlinks zu wichtigen Kryptoseiten und zwei Programme zum Ausprobieren zusammen.

Zum Umgang mit dieser Arbeit: Es empfiehlt sich, die PDF oder DVI-Version meiner Arbeit durchzulesen, da mit einem entsprechenden Betrachter (z.B. Adobe Acrobat oder XDVI) Hyperlinks aktiviert werden, die durch die gesamte Arbeit führen und so ein bequemes Nachschlagen ermöglichen.

Inhaltsverzeichnis

I Klassische Verfahren und Designkriterien 1

1 Grundbegriffe 3
1.1 Was ist Kryptologie ? . . . 3
1.2 Klassische Chiffrierer . . . 3
1.2.1 Substitutionschiffren . . . 3
1.2.2 Transpositionschiffren . . . 4
1.2.3 Produktchiffren . . . 4
1.3 Struktur von symmetrischen Blockchiffrierern . . . 5
1.4 Feistel-Chiffren . . . 7
1.5 Grundlagen der Kryptoanalyse . . . 8
1.6 Die Betriebsmodi eines Blockchiffrierers . . . 12

2 Elementare Analysen und Designkriterien 19
2.1 Vollständige Schlüsselsuche — Tabellensuche . . . 19
2.2 Klartext-Chiffretext Unabhängigkeit . . . 22
2.3 Vollständigkeit . . . 24
2.4 Linearität . . . 25
2.5 Das (strikte) Avalanche Kriterium . . . 25
2.6 Die Abhängigkeitsmatrix . . . 26
2.7 Komplementeigenschaft . . . 27
2.8 Abgeschlossenheit . . . 27
2.9 Lineare Faktoren . . . 28
2.10 Mehrfache Verschlüsselung . . . 29

3 Die Blockchiffrierer DES und FEAL 31
3.1 DES . . . 31
3.1.1 Design des DES . . . 33
3.1.2 Eigenschaften des DES . . . 33
3.1.3 Triple DES . . . 38
3.2 FEAL . . . 39
3.2.1 Design des FEAL . . . 39
3.2.2 Eigenschaften des FEAL . . . 41

II Moderne Verfahren und Designkriterien **43**

4 Differentielle Kryptoanalyse **45**
4.1 Grundlagen . . . 45
4.2 Charakteristiken und Differentiale . . . 51
4.3 Kryptoanalyse des 3-Runden DES . . . 54
4.4 Kryptoanalyse des 6-Runden DES . . . 57
4.5 Aufwandsabschätzung für DES und FEAL-N . . . 59
4.6 Zusammenfassung . . . 61
4.7 Kryptoanalyse mit gekürzten und höheren Differenzen . . . 63

5 Lineare Kryptoanalyse **65**
5.1 Grundlagen . . . 65
5.2 Charakteristiken und Lineare Hüllen . . . 69
5.3 Lineare Kryptoanalyse des FEAL-4 . . . 74
5.4 Aufwandsabschätzung für DES und FEAL-N . . . 78
5.5 Zusammenfassung . . . 80
5.6 Differentiell-Lineare Kryptoanalyse . . . 81

6 Designkriterien mit beweisbarer Sicherheit **83**
6.1 Sicherheit gegen diff. Kryptoanalyse . . . 83
6.2 Sicherheit gegen lin. Kryptoanalyse . . . 85
6.3 Neue Kriterien für Chiffrierer . . . 86
6.3.1 Größe und Art von S-Boxen . . . 87
6.3.2 Schlüsselauswahl . . . 90
6.3.3 Konstruktion der Rundenfunktion . . . 91

7 Advanced Encryption Standard **93**
7.1 Kriterien für die Kandidaten . . . 94
7.2 Ein Bewerber: RC6 . . . 96
7.2.1 Design des RC6 . . . 96
7.2.2 Sicherheit des RC6 . . . 99

III Anhang **103**

A Das Geburtstagsparadoxon **105**

B Hyperlinks und Inhalt der CD **109**
B.1 Hyperlinks zu wichtigen Kryptoseiten im Internet . . . 109
B.2 Der Inhalt der CD . . . 110
B.3 Lineare Kryptoanalyse des FEAL-4: Das Programm . . . 110
B.4 Die Entropie meiner Arbeit . . . 115

Abbildungsverzeichnis

1.1 Schema einer symmetrischen Blockchiffre 5
1.2 Prinzip der Produktchiffre 6
1.3 Prinzip der Feistel-Chiffre 7
1.4 Feistel-Chiffre mit 3 Verschlüsselungsschritten 8
1.5 Das Kryptographische System 9
1.6 Schema des One-Time-Pads 10
1.7 Der ECB-Modus 12
1.8 Der CBC-Modus 13
1.9 Der OFB-Modus 15
1.10 Der OFB-Modus nach ISO/IEC 10116 16
1.11 Der CFB-Modus 17
1.12 Fehlerfortpflanzung in den verschiedenen Betriebsmodi 18

2.1 Ein *time-memory tradeoff* 20
2.2 Linearer Faktor eines Kryptosystems 28

3.1 Der Data Encryption Standard 32
3.2 Die Permutationen *IP* und *PC1* des DES 33
3.3 Struktur des FEAL-8 40
3.4 Struktur der Funktionen F und F_K des FEAL 40

4.1 Die F-Funktion des DES 46
4.2 Die iterative 2-Runden Charakteristik des DES 52
4.3 6-Runden DES Analyse: Charakteristik Ω^1 58
4.4 6-Runden DES Analyse: Charakteristik Ω^2 58
4.5 Die iterative Charakteristik des FEAL-N 62

5.1 Beste 1-Runden Approximation des DES 71
5.2 Informationsfluß in der diff. und linearen Analyse 73
5.3 Modifizierter FEAL-4 und Rundenfunktion F 75

6.1 Ungleichmäßige Feistelnetzwerke 92

7.1 Struktur des RC6 (Verschlüsselung) 98

A.1 Gerichteter Graph einer Funktion 106

Tabellenverzeichnis

1.1 Vor- und Nachteile der Blockbetriebsmodi 18
1.2 Vor- und Nachteile der einzelnen Strombetriebsmodi 18

2.1 Laufzeitenvergleich Schlüsselsuche — Tabellensuche 21
2.2 **Algorithmus:** Abhängigkeitsmatrix 26

3.1 **Algorithmus:** Der Data Encryption Standard 34
3.2 DES Ein- und Ausgangspermutation IP und IP^{-1} 35
3.3 DES Rundenfunktionen: Expansion E und Permutation P 35
3.4 DES Schlüsselauswahlfunktionen PC1 und PC2 35
3.5 Die acht S-Boxen des DES 36
3.6 Die schwachen Schlüssel des DES 37
3.7 Semi-schwache Schlüssel des DES 38
3.8 Die Funktionen F und F_K des FEAL 42
3.9 **Algorithmus:** Der FEAL-8 42

4.1 Prozentsatz an möglichen Outputxors der verschiedenen S-Boxen 48
4.2 Differenzverterteilungstabelle der S-Box S1 49
4.3 S-Box Inputs für Beispiel 4.4 50
4.4 In- und Outputpaare für Beispiel 4.5 56
4.5 Zählerfelder für Beispiel 4.5 56
4.6 **Algorithmus:** Cliquemethode 59
4.7 Aufwand der diff. Analyse beim DES 60
4.8 Aufwand der verbesserten diff. Analyse des DES 60
4.9 Aufwand der diff. Analyse des FEAL-N 62

5.1 **Algorithmus** 1 der lin. Kryptoanalyse 67
5.2 **Algorithmus** 2 der lin. Kryptoanalyse 67
5.3 Lineare Approximationstabelle für S5 70
5.4 Aufwand der lin. Kryptoanalyse beim n-Runden DES 79
5.5 Aufwand der lin. Kryptoanalyse beim FEAL-N 80

7.1 **Algorithmus:** Der RC6 97
7.2 Aufwand der diff. und lin. Kryptoanalyse beim RC6 99
7.3 iteratives Differential und beste Charakteristik des RC6 101

Teil I

Klassische Verfahren und Designkriterien

Kapitel 1

Grundbegriffe

1.1 Was ist Kryptologie ?

Kryptologie [Broc90] ist die Lehre von der Entwicklung und der Bewertung von Verschlüsselungsverfahren zum Schutz von Daten. In der Kryptologie geht es darum, eine von einem Sender durch einen unsicheren (d.h. Unbefugten zugänglichen) Kanal zu übermittelnde Nachricht (Klartext K) so zu verschlüsseln, daß Unbefugte aus dem Chiffretext C den Klartext K nicht rekonstruieren können. Kryptologie unterteilt sich in die beiden Untergebiete Kryptographie und Kryptoanalyse. Die Aufgabe des Kryptographen besteht darin, eine zugleich sichere wie einfache Methode zur Verschlüsselung zu entwickeln, d.h. sowohl die Verschlüsselung als auch die Entschlüsselung sollte möglichst einfach zu realisieren, aber dennoch sicher in einem gewissen Maß sein. Der Kryptoanalytiker ist hingegen bemüht, diese Methoden zu brechen.

1.2 Klassische Chiffrierer

Bereits die alten Griechen und Römer erkannten die Notwendigkeit, wichtige Dokumente oder Botschaften zu verschlüsseln. Die griechische Skytala oder die Caesar-Chiffre sind Beispiele hierfür. Im Laufe der Jahrhunderte wurden diese vergleichsweise primitiven Verschlüsselungsfunktionen weiterentwickelt, aber erst mit der Entwicklung der ersten mechanischen Verschlüsselungsapparate wurde ein Quantensprung in der Komplexität der Chiffrierer erreicht. Die beiden grundlegenden Prinzipien eines Chiffrierers sind Konfusion und Diffusion, wie sie C.E. Shannon in [Shan49] niedergelegt hat.

Das Prinzip der Konfusion (Verwirrung) ist nach Shannon die möglichst komplexe Gestaltung der Verbindung zwischen Schlüssel, Klartext und Chiffretext. Die fortgeschritteneren Typen von **Substitutionschiffren** bzw. **Stromchiffrierer** sind darauf basierende Verschlüsselungsmethoden. Das Prinzip der Diffusion besagt, daß ein Chiffretextzeichen von möglichst vielen Klartextzeichen und möglichst dem gesamten Schlüssel abhängen sollte, dies kann z.B. dadurch erreicht werden, indem die Anordnung der Klartextzeichen verändert wird, also durch Permutation eines Blocks von Klartextzeichen. **Transpositionschiffren** sind die einfachste darauf beruhende Klasse von Verschlüsselungsmethoden.

1.2.1 Substitutionschiffren

Substitutionschiffren ersetzen ein Klartextzeichen durch ein Chiffretextzeichen, wobei meistens die beiden zu Grunde liegenden Alphabete gleich sind. Die älteste bekannte Substitutionschiffre ist die Caesar-Chiffre, sie gehört zu den einfachen Substitutionschiffren. Es gibt verschiedene Typen von Substitutionschiffren; die **einfache Substitutionschiffre** ersetzt jedes Klartextzeichen durch ein eindeutig festgelegtes Chiffretextzeichen. Diese Chiffre ist sehr einfach zu brechen, indem man Häufigkeitsverteilungen der entsprechenden Buchstaben in der jeweiligen Sprache zu Rate zieht, vorausgesetzt, man besitzt eine genügend große Menge an Chiffretext und kennt die Sprache des Klartextes.

Die **homophone Substitutionschiffre** ordnet jedem Klartextzeichen eine Menge von Chiffretextzeichen zu, aus der (pseudo)zufällig eines ausgewählt wird. Sinnigerweise entsprechen die Mächtigkeiten der Bildmengen der Häufigkeit des Klartextzeichens, sodaß die Häufigkeitsanalyse keine Anwendung mehr finden kann. Ein berühmter Vertreter dieser Chiffre ist die **Beale-Chiffre**, die bis heute nicht entschlüsselt werden konnte. Ein selten zu tolerierender Nachteil der homophonen Substitutionschiffren ist die Aufblähung des Chiffretextes aufgrund des größeren Alphabets der Chiffretextzeichen. Einfache Substitutionschiffren sind im Grunde homophone Substitutionschiffren mit einelementigen Bildmengen.

Polyalphabetische Substitutionschiffren ordnen jedem Klartextzeichen je nach Position unterschiedliche Chiffretextzeichen zu. Somit kann einerseits einem Klartextzeichen mehrere Chiffretextzeichen, andererseits mehreren Klartextzeichen dasselbe Chiffretextzeichen zugeordnet werden. Die Auswahl der verschiedenen Chiffretextzeichen erfolgt in der Regel nach einem festen Schema in einem periodischen Muster. **Stromchiffrierer** sind eine Unterklasse von polyalphabetischen Substitutionschiffren.

Polygraphische Substitutionschiffren ersetzen schließlich mehrere Klartextzeichen durch eine Chiffretextzeichenkette. Dies kann auch durch mehrere verschiedene Funktionen erfolgen. **Blockchiffrierer** gehören diesem Typ von Substitutionschiffren an.

1.2.2 Transpositionschiffren

Transpositionschiffren verändern, wie bereits oben erwähnt, den Klartext nicht, sondern verändern nur ihre Reihenfolge, die griechische Skytala ist das älteste Beispiel für eine Transpositionschiffre. Die dafür verwendeten Permutationen werden bei klassischen Chiffrierern mittels geometrischer Figuren wie Mäander, Rechtecke, u.ä. beschrieben. Auch wenn die Zeichen des Klartextes unverändert bleiben, so werden doch die statistischen Eigenschaften zum Teil zerstört. Die Häufigkeitverteilung der einzelnen Buchstaben bleibt natürlich unverändert, aber die Verteilung von Bi- oder Trigrammen (Zweier oder Dreierpaare) wird verändert. Die Kryptoanalyse einer Transpositionschiffre beruht also im Wesentlichen auf der Herstellung der ursprünglichen Bi- und Trigramme. Bei der Konstruktion einer Transpositionschiffre stellt also die Auswahl einer geeigneten Permutation die größte Herausforderung dar:

- Die Permutation sollte möglichst gut „mischen“,
- dabei sollten möglichst wenige Bigramme erhalten bleiben.
- Ein Tupel „xy“ sollte auch nicht auf das Tupel „yx“ abgebildet werden.
- Als mittlerer Abstand zwischen Urbild- und Bildposition ist die Hälfte der gewählten Blocklänge wünschenswert.
- Der mittlere Abstand der Bildpositionen zweier benachbarter Urbilder sollte ebenfalls möglichst groß sein. Eine zufällig gewählte Permutation wird im Mittel die halbe Blocklänge als mittleren Abstand besitzen.

1.2.3 Produktchiffren

Die klassischen Vorgänger moderner Blockchiffrierer sind die Produktchiffren. Diese Klasse von Verschlüsselungsfunktionen schließt Substitutions- und Transpositionschiffren zusammen und vereint damit das Prinzip von Konfusion und Diffusion. Produktchiffren bestehen aus einer Folge von Funktionen $F_1, \ldots, F_n$, die hintereinander ausgeführt werden, wobei F_i entweder eine Substitution oder eine Transposition ist. Aufgrund der Komplexität

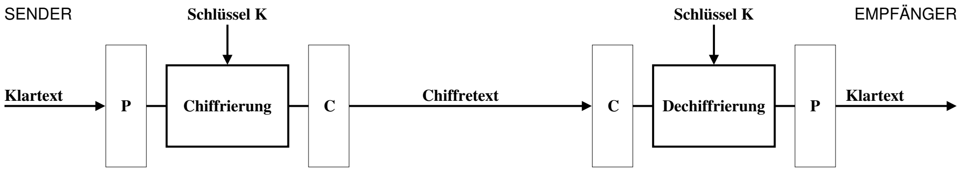

Abbildung 1.1: Schema einer symmetrischen Blockchiffre

konnten solche Chiffrierer nicht mehr mit einfachen Mitteln konstruiert werden, sie wurden erst mit der Erfindung (elektro-)mechanischer Verschlüsselungsapparate realisierbar. Das älteste Beispiel ist der Jefferson-Zylinder, der um 1790 entwickelt, aber erst Anfang dieses Jahrhunderts eingesetzt wurde. Einen weitereren großen Fortschritt stellten die Rotormaschinen dar, die kurz nach Ende des Ersten Weltkrieges erfunden wurden. Bekanntester Vertreter dieser Klasse ist die Enigma, die während des zweiten Weltkrieges von der Reichsarmee eingesetzt wurde. Rotormaschinen sind im Grunde polyalphabetische Substitutionschiffren, wobei der Wechsel zwischen den verschiedenen einfachen Substitutionen durch Weiterschaltung der Rotoren erfolgt. Die Länge der Periode ergibt sich dadurch aus der Anzahl der Rotoren sowie deren Größe, als Schlüssel kommt die Anfangsstellung noch hinzu.

Für eine einfache Einführung in klassische Substitutions-, Transpositions- und Produktchiffren sowie deren Analyse konsultiere man [FuRi94] oder den Klassiker unter den Kryptographiebüchern, D. Kahns „The Codebreakers“ [Kahn67].

1.3 Struktur von symmetrischen Blockchiffrierern

Wie bereits im Vorwort erwähnt, betrachte ich in dieser Arbeit nur symmetrische Blockchiffrierer, d.h. der Schlüssel für die Entschlüsselungsoperation ist derselbe wie der für die Verschlüsselung oder kann zumindest leicht daraus abgeleitet werden. Es hat sich eingebürgert, einen Klartextblock mit P und einen Chiffretextblock mit C zu bezeichnen. Darüberhinaus bezeichnet K einen Schlüssel, $\mathbf{E}$ die Verschlüsselungs- und $\mathbf{D}$ die Entschlüsselungsoperation einer Blockchiffre (siehe Abbildung 1.1). Ist C der aus P mittels $\mathbf{E}$ und K erzeugte Chiffretext, so gelten für symmetrische Blockchiffrierer die Beziehungen

$$C = \mathbf{E}(K,P) \qquad \textit{und} \qquad P = \mathbf{D}(K,C).$$

Damit diese Zuordnung eindeutig wird, müssen die Funktionen $\mathbf{E}(K,\cdot)$ und $\mathbf{D}(K,\cdot)$ Bijektionen sein, weil eine Aufblähung des Chiffretextes unerwünscht ist, d.h. der Klartextraum ist genauso groß wie der Chiffretextraum. Seit die gesamte Kryptographie im Computer stattfindet, ist der zugrunde liegende Klartext- bzw. Chiffretextraum der Vektorraum $\mathbb{Z}_2^n$ mit seiner üblichen komponentenweisen Addition, die ich mit $\oplus$ bezeichnen werde. Der Körper $\mathbb{Z}_2$ wird mit seiner üblichen Addition $(+)$, die der logischen Verknüpfung XOR entspricht, und Multiplikation $(\cdot)$, die der logischen Verknüpfung AND entspricht, betrachtet. Der Raum $\mathbb{Z}_2^n$ enthält 2^n verschiedene Vektoren, d.h. moderne Blockchiffrierer sind im Prinzip nichts anderes als eine vom Schlüssel abhängige Permutation dieser 2^n Blöcke.

Wie bereits bei den Transpositionschiffren erwähnt, stellt die Auswahl geeigneter Permutationen aus den $(2^n)!$ Möglichkeiten die größte Herausforderung dar. Die meisten Blockchiffrierer operieren auf 64-Bit Blöcken, zur Zeit vollzieht sich aber gerade der Wechsel zu 128-Bit Blöcken. Bereits 64 Bit ermöglichen eine ausreichend große Klasse an geeigneten Permutationen, allerdings stellt die Beschreibung einer solchen Permutation sowohl Hardwaredesigner als auch Programmierer vor große Probleme. Im Prinzip ist eine Permutation durch Leitungsvertauschung leicht zu realisieren, allerdings würde dies bei 2^{64} Leitungen unmöglich zu realisieren sein. Softwareseitig ist eine Permutation nicht effektiv

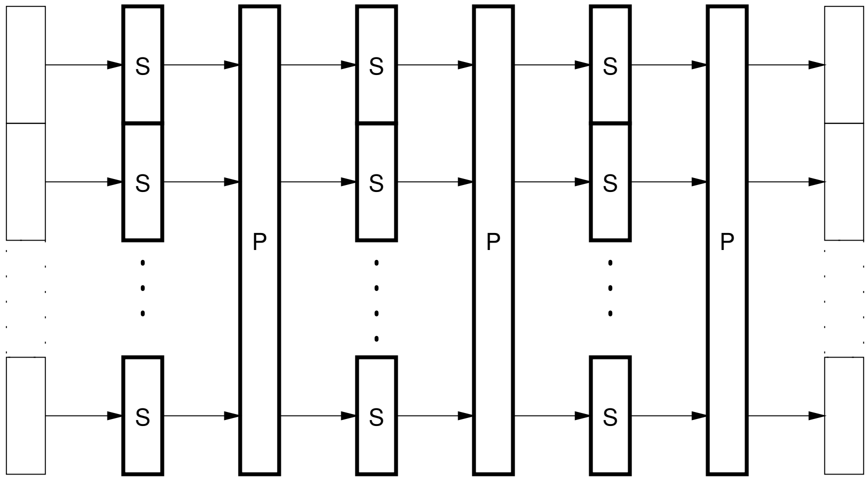

Abbildung 1.2: Prinzip der Produktchiffre

beschreibbar und bereits eine Bitpermutation von 64 Leitungen sehr langsam. Eine andere Möglichkeit wäre das Anlegen einer Tabelle, die zu jedem Inputblock den entsprechenden Outputblock ausgibt. Eine solche Tabelle würde $2^n \cdot n$ Bit benötigen, im Falle von 32-Bit Blöcken wäre also ein Speicher der Größe $2^{37} = 128$ Gigabit erforderlich, selbst eine 16-Bit Substitution würde noch ein Megabit Speicher benötigen. Dies kann also keine geeignete Möglichkeit sein.

In der Praxis wird also versucht, den normalerweise 64-Bit großen Inputvektor so zu permutieren, daß die Permutation aus kleineren, leicht realisierbaren Komponenten zusammengesetzt werden kann. Hierzu wird der Inputvektor in kleinere Teile zerlegt und auf diesen Teilen eine kleine bijektive Substitution durchgeführt (in der Literatur häufig S-Box genannt). Anschließend werden die einzelnen Outputvektoren permutiert und noch einmal substituiert, usw. Durch die wiederholte Anwendung dieses Schemas versucht man, die Komplexität der Verschlüsselungsoperation zu erhöhen. Eine Permutation von z.B. 8-Bit Blöcken ist heute ohne Probleme durch eine Tabelle realisierbar. Natürlich ist nicht jede bijektive Substitution geeignet, um als S-Box zu fungieren, im folgenden versuche ich also, Kriterien zu entwickeln, die eine Kryptoanalyse so schwer wie möglich gestalten. Da die S-Box in diesem Schema der einzige nichtlineare Anteil ist, muß also darauf das größte Augenmerk gelegt werden. Elementare Designkriterien für kryptographisch „gute“ Substitutionen sind unter anderem:

- Jedes Outputbit sollte von jedem Inputbit echt abhängig sein, d.h. die Boolesche Funktion, die jedes Outputbit beschreibt, enthält in ihren minimierten Termen alle Inputvariablen. Eine Funktion mit dieser Eigenschaft heißt vollständig.
- Die Änderung eines Inputbits sollte im Mittel die Hälfte aller Outputbits ändern. Diese Eigenschaft heißt „Avalanche“-Effekt.
- Kein Outputbit sollte linear von einem Inputbit abhängen.
- Auch die Booleschen Funktionen, die Linearkombinationen von Input- oder Outputbits beschreiben, sollten nichtlinear sein.

Diese und weitere Designkriterien werde ich in den folgenden Kapiteln näher analysieren und Kryptoanalysen vorstellen, die etwaige Lücken ausnutzen.

In den 70er und 80er Jahren war es technisch noch nicht möglich, gute und schnelle Blockchiffrierer mit 128-Bit Blöcken zu entwickeln, obwohl eine größere Blockgröße für

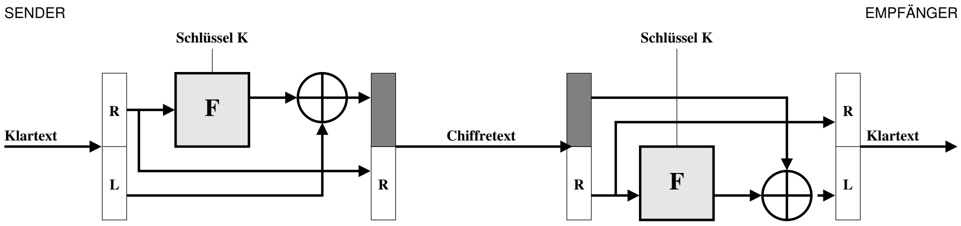

Abbildung 1.3: Prinzip der Feistel-Chiffre

den Klartext und den Schlüssel wünschenswert ist. Mit dem Aufkommen von 32-Bit Prozessoren ist es möglich geworden, schnelle Operationen auf 32-Bit Blöcken auszuführen, was die Vergrößerung erleichterte.

1.4 Feistel-Chiffren

Die Feistel-Chiffre (z.B. zu finden in [Feis75]) stellt die Grundlage für die meisten modernen Blockchiffrierer dar. Die Idee hinter der Feistel-Chiffre ist die Benutzung einer Funktion F sowohl zur Verschlüsselung wie auch für die Entschlüsselung und basiert auf der Hufeisenfunktion aus der Theorie dynamischer Systeme. Die Hufeisenfunktion ist dort bekannt für seine „chaoserzeugende“ Wirkung.

Sei $\mathcal{B} = \mathbb{Z}_2^n$ der Klartext- bzw. Chiffretextraum mit $n = 2k$ gerade. Sei $P, C \in \mathcal{B}$ wobei P ein Klartextvektor, C ein Chiffretextvektor ist. Sei $K \in \mathbb{Z}_2^m$ ein m Bit Schlüsselvektor.

Da n gerade, kann P aufgeteilt werden in zwei Blöcke der Länge k: $P = (L, R)$. Die Verschlüsselungsoperation erfolgt dann durch

$$C = (R, L \oplus F(K, R)).$$

Der Block R bleibt also unverschlüsselt. Diese Operation kann wieder umgekehrt werden aufgrund der Beziehung

$$F(K,R) \oplus (F(K,R) \oplus L) = 0 \oplus L = L,$$

d.h. man erhält seinen Klartextblock P zurück wie in Abbildung 1.3 dargestellt. Bemerkenswert hierbei ist, daß die Eigenschaften der Funktion F nicht von Belang sind, sie muß also nicht bijektiv sein.

Die Struktur der Feistel-Chiffre läßt sich auch als Matrix beschreiben; in dieser Gestalt wird deutlich, daß das Schema dem der Hufeisenfunktion entspricht. Die Rundenoperation und seine Inverse sehen als Matrix wie folgt aus:

$$\begin{pmatrix} C_L \\ C_R \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & F \end{pmatrix} \begin{pmatrix} P_L \\ P_R \end{pmatrix}, \qquad \begin{pmatrix} P_L \\ P_R \end{pmatrix} = \begin{pmatrix} F & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} C_L \\ C_R \end{pmatrix}$$

Das Schema stellt in dieser Form natürlich noch keine gute Chiffre dar, da der rechte Teilblock überhaupt nicht verschlüsselt wird. Deswegen wird dieses Verfahren nun iteriert, um größere kryptographische Stärke zu erhalten. Die bei jedem Verschlüsselungsschritt $\mathbf{E}_i$ verwendeten Funktionen F_i und Schlüssel K_i müssen dabei nicht identisch sein. Jeder dieser Verschlüsselungsschritte (auch **Runden** genannt)

$$\mathbf{E}_i(L_{i-1}, R_{i-1}) = (R_{i-1}, L_{i-1} \oplus F_i(K_i, R_{i-1})) = (L_i, R_i)$$

ist bis auf eine Vertauschung der Teilblöcke idempotent. Definiert man nämlich $V(L,R) = (R,L)$, so gilt:

$$V \circ \mathbf{E}_i \circ V \circ \mathbf{E}_i(L_{i-1}, R_{i-1}) =$$

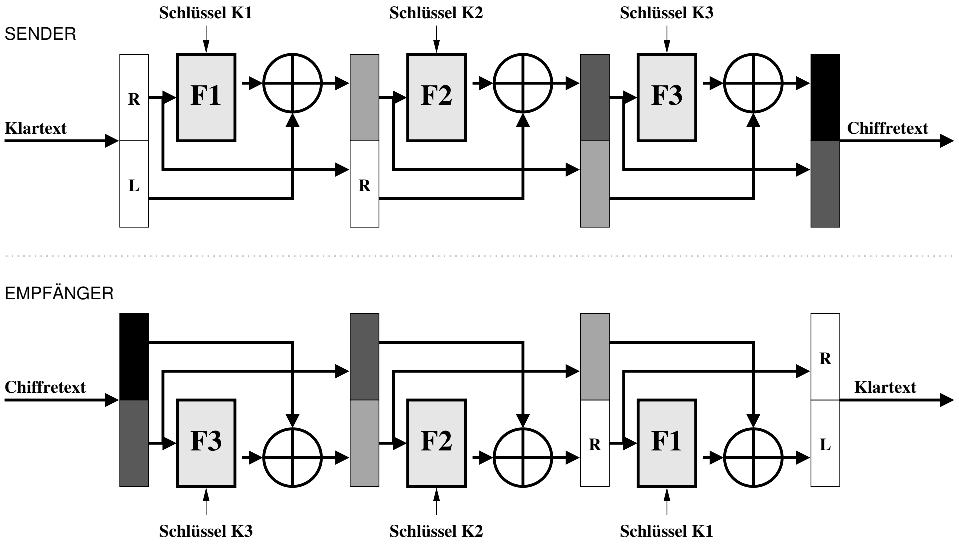

Abbildung 1.4: Feistel-Chiffre mit 3 Verschlüsselungsschritten

$$
\begin{aligned}
&= V \circ \mathbf{E}_i(L_{i-1} \oplus F_i(K_i, R_{i-1}), R_{i-1}) \\
&= V(R_{i-1}, L_{i-1} \oplus F_i(K_i, R_{i-1}) \oplus F_i(K_i, R_{i-1})) \\
&= (L_{i-1}, R_{i-1})
\end{aligned}
$$

Die Inverse einer aus n Runden bestehenden Chiffre

$$\mathbf{E} = V \circ \mathbf{E}_n \circ \mathbf{E}_{n-1} \circ \ldots \circ \mathbf{E}_2 \circ \mathbf{E}_1$$

ist daher die Abbildung

$$\mathbf{D} = V \circ \mathbf{E}_1 \circ \mathbf{E}_2 \circ \ldots \circ \mathbf{E}_{n-1} \circ \mathbf{E}_n,$$

es muß also zur Entschlüsselung nur die Reihenfolge der Verschlüsselungsschritte vertauscht werden. Eine Chiffre dieses Typs heißt **Feistel-Chiffre**, das Schema der Ver- und Entschlüsselung einer Feistel-Chiffre mit drei Runden ist in Abbildung 1.4 dargestellt.

In der Praxis wählt man als Funktionen F_i für jede Runde die gleiche Funktion F, und die Teilschlüssel K_i werden aus einem Gesamtschlüssel K generiert, um die Implementierung zu vereinfachen. Die kryptographische Sicherheit beruht nun allein in der Anzahl der Runden, der Schlüsselauswahl und der Funktion F. Das Problem, eine geeignete Familie von Permutationen auszuwählen, die die Blockchiffre definiert, wird bei einer Feistel-Chiffre so allein auf das Design der Funktion F reduziert. Eine geeignete Funktion könnte leicht mittels eines Zufallsprozesses erzeugt werden; wie im vorigen Abschnitt erwähnt, ist allerdings eine 32-Bit Substitution, wie sie im Falle von den derzeit üblichen 64-Bit Blöcken nötig wäre, nicht realisierbar. Die Funktion F ist daher eine Produktchiffre, wie in Abbildung 1.2 gezeigt, die sich aus kleinen S-Boxen zusammensetzt. Diese S-Boxen müssen nicht bijektiv sein, im Falle des DES sind sie Abbildungen von $\mathbb{Z}_2^6 \to \mathbb{Z}_2^4$.

1.5 Grundlagen der Kryptoanalyse

Die Grundlage der Kryptographie ist das Kryptographische System, wie in Abbildung 1.5 illustriert. Die Aufgabe des Kryptographen ist die Überbrückung des unsicheren Kanals, ohne daß etwaige Abhörer die ursprüngliche Nachricht restaurieren können. Dabei muß der Kryptograph berücksichtigen, daß der Angreifer nicht nur den Chiffretext kennt, sondern mit großer Wahrscheinlichkeit auch den verwendeten Chiffrierer. Außerdem besitzt

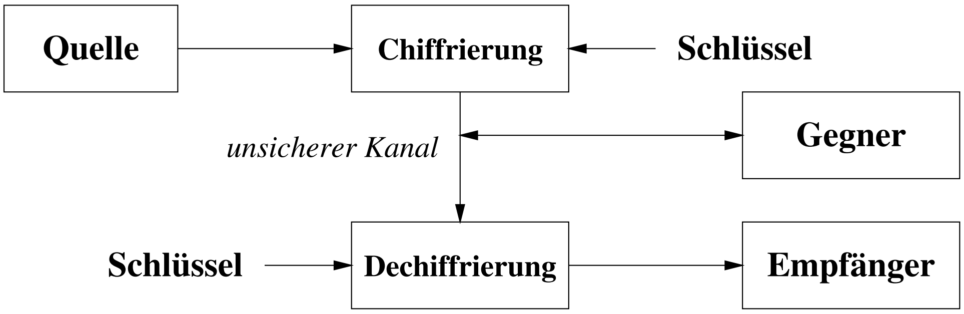

Abbildung 1.5: Das Kryptographische System

ein Angreifer meistens einige Klartext/Chiffretext-Paare oder kann sogar selber Paare mit dem aktuellen Schlüssel erzeugen. Die Sicherheit einer Chiffre darf also nie allein von der Geheimhaltung der Schlüssel herrühren. Das Ziel jeder Kryptoanalyse ist demnach die Rekonstruktion des Klartextes bzw. des benutzten Schlüssels.

Der Angreifer kann drei unterschiedliche Ziele haben:

- Passive Spionage: Der Angreifer versucht nur, die ursprüngliche Nachricht zu restaurieren.

- Aktive Spionage: Der Gegner versucht, die Nachricht zu stören, in dem er die Nachricht verändert. Dies ist z.B. interessant, wenn es um finanzielle Transaktionen geht.

- Tarnung: Kryptographie wird unter anderem dazu benutzt, den Zugriff auf ein System nur legitimen Benutzern zur Verfügung zu stellen. Wird dieser Authentifizierungsmechanismus durchbrochen, so kann sich der Angreifer als ein legitimer Benutzer ausgeben, um das System zu mißbrauchen oder einem Benutzer vorgaukeln, er befinde sich im gewünschten System und nicht im System des Angreifers.

Die eigentliche Analyse mag vielleicht zuerst hoffnungslos erscheinen, sie wird aber dadurch erleichtert, daß die normale Sprache höchst redundant ist; so mag es nur erforderlich sein, einen kleinen Teil des Chiffretextes zu entschlüsseln, um den Rest lesen zu können, da vielleicht ein Teil der Nachricht aus Daten besteht, ein anderer Teil aber aus einzelnen Wörtern. In diesem Sinne durchsucht der Angreifer den Schlüsselraum solange nach dem richtigen Schlüssel, bis ein sinnvoller Text erscheint.

Schwache Chiffrierer können einer Attacke nur widerstehen, solange dem Angreifer bestimmtes Wissen vorenthalten wird; starke Chiffrierer widerstehen, selbst wenn dem Angreifer das genaue Design des Chiffrierers zur Verfügung steht. Demnach wird eine Kryptoanalyse in die folgenden Klassen eingeordnet:

- Reiner Chiffretextangriff: Hier steht dem Kryptoanalytiker nur der Chiffretext zur Verfügung.

- Angriff mit bekanntem Klartext: Der Gegner verfügt über einige Klartext/ Chiffretext-Paare.

- Angriff mit wählbarem Klartext: Der Angreifer kann nach Belieben Klartext/ Chiffretext-Paare mit dem aktuellen Schlüssel erzeugen.

- Angriff mit wählbarem Chiffretext: Der Angreifer kann nach Belieben Chiffretext mit dem aktuellen Schlüssel dechiffrieren.

Ein Chiffrierer, der allein aus Kenntnis eines Stückchens Chiffretext gebrochen werden kann, ist demnach sehr unsicher, während ein Chiffrierer, der selbst mit wählbarem Klartext nicht gebrochen werden kann, als sehr sicher gilt. Der letzte Punkt entspricht in etwa

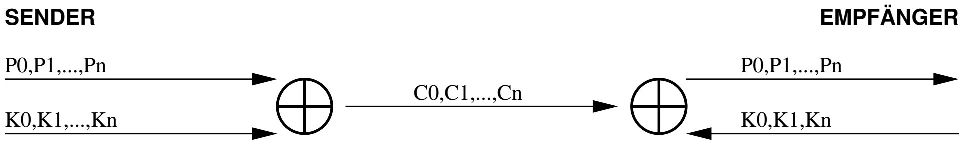

Abbildung 1.6: Schema des One-Time-Pads

dem dritten Punkt bei asymmetrischen Chiffren. Für symmetrische Blockchiffren spielt der letzte Punkt kaum eine Rolle.

Diese Steigerung kann mit den von Shannon [Shan49] begründeten Prinzipien der Diffusion und Konfusion näher erläutert werden. Ein Angriff mit bekanntem Klartext läuft auf das Lösen eines Gleichungssystems hinaus, das den Schlüssel ausgibt. Das Prinzip der Diffusion sorgt nun dafür, daß in den einzelnen Gleichungen viele Variablen enthalten sind, während die Konfusion die Komplexität der Gleichungen erhöht, also nichtlineare Anteile hineinbringt. Bei einem wählbaren Klartextangriff können, um die Komplexität zu verringern, Paare mit bestimmter Struktur betrachtet werden, während bei einem reinen Chiffretextangriff nur wenig Informationen zur Verfügung stehen, wenn überhaupt ein Gleichungssystem aufgestellt werden kann.

Im folgenden gehe ich noch auf zwei wichtige Begriffe ein: Der **absoluten** und **praktischen Sicherheit**. Ich werde diese Begriffe aus der Informationstheorie nur oberflächlich erläutern, da eine genauere Ausführung den Rahmen dieser Diplomarbeit sprengen würde.

Absolute Sicherheit ist dann gegeben, wenn aus dem Chiffretext keinerlei Rückschlüsse auf den ursprünglichen Klartext gezogen werden können. Mit anderen Worten: die Wahrscheinlichkeit, daß eine Nachricht N gesendet wurde, ist gleich der Wahrscheinlichkeit, daß N gesendet wurde, wenn der Empfänger den Chiffretext C empfangen hat; also $\mathcal{P}(N) = \mathcal{P}(N|C)$. Es gibt also zu jedem Chiffretext C einen Schlüssel $K \in \mathcal{K}$, der N in C überführt. Daraus folgt, daß $|\mathcal{K}| \geq |\mathcal{C}|$ und aufgrund der Injektivität der Verschlüsselungsfunktion es mindestens ebenso viele Chiffretexte wie Klartexte geben muß. Zusammengefaßt muß es also in einem absolut sicheren Kryptosystem mindestens ebenso viele Schlüssel wie Klartexte geben. In der Praxis bedeutet dies, daß jeder zu verschlüsselnde Klartextblock von stochastisch unabhängigen Schlüsseln chiffriert werden muß.

Das einfachste Beispiel einer absolut sicheren Chiffre ist der *One-Time-Pad*, wie in Abbildung 1.6 dargestellt. Diese Chiffre verschlüsselt jedes einzelne Bit des Klartextes mit einem Bit des Schlüsselstroms durch Addition. Die Entschlüsselung erfolgt auf einfachste Weise durch erneute Addition des Chiffretextes mit dem Schlüsselstrom. Der hierbei verwendete Schlüsselstrom $K_0 \cdots K_n$ ist dabei aus einer echten Zufallsquelle gewonnene Folge von 0 und 1. Daß diese Chiffre nicht in großem Stil einsetzbar ist, liegt daran, daß zusammen mit der Nachricht auch der Schlüsselstrom übertragen werden muß und zudem jede Nachricht nicht mit demselben Schlüsselstrom verschlüsselt werden darf. Daher wird diese Chiffre nur in sehr begrenzten Fällen eingesetzt, in denen absolute Sicherheit gewährleistet sein muß, wie z.B. bei dem heißen Draht zwischen Moskau und Washington.

Der Umstand, daß absolute Sicherheit nur dann vorliegen kann, wenn der Schlüssel ebenso lang ist wie die Nachricht, bringt das Konzept der praktischen Sicherheit ins Spiel. Jeder moderne Chiffrierer ist zwar theoretisch brechbar, die Analyse kann aber durchaus Ressourcen beanspruchen, die jenseits aller technischen Möglichkeiten liegt.

Es gibt drei verschiedene Ansätze zur Kryptoanalyse:

- Vollständige Schlüsselraumsuche
- Statistische Analyse
- Analyse mit mathematischen Hilfsmitteln aus der (linearen) Algebra oder diskreten Mathematik.

Man erhält sehr leicht eine obere Schranke für die praktische Sicherheit eines Chiffrierers, indem man den Schlüsselraum möglichst groß wählt. Mit heutigen Rechnerkapazitäten sind vollständige Schlüsselsuchen bis zur Größenordnung von 2^{64} machbar. Besitzt der Schlüsselraum aber eine Größe von 2^{100} oder mehr, so ist eine Suche auch in absehbarer Zukunft nicht denkbar. Da sich seit geraumer Zeit die Rechnerleistung alle 18 Monate etwa verdoppelt, wird also nach dieser Annahme eine vollständige Schlüsselsuche eines 100-Bit Schlüssels erst in ca. 54 Jahren machbar sein. Eine Abschätzung von Rechnerleistungen über mehr als 5 Jahre im Voraus sind aber sehr mit Vorsicht zu betrachten. Alle Angriffe, die ich im zweiten Teil vorstellen werde, beruhen auf einer Kombination der Punkte eins bis drei. Die einfache Substitution aus Abschnitt 1.2.1 besitzt zwar einen großen Schlüsselraum (alle möglichen Permutationen auf 26 Elementen, also $26! \approx 4 \cdot 10^{26}$), kann aber mittels statistischer Analyse leicht mit nur wenigen Versuchen gebrochen werden.

Ein Kryptosystem kann dann als **praktisch sicher** bezeichnet werden, wenn die Kryptoanalyse einen Aufwand benötigt, der die Möglichkeiten jedes denkbaren Gegners übersteigt, oder die Analyse so teuer (in Rechenzeit oder Geld) ist, daß die Kosten der Analyse den zu erwartenden Gewinn übersteigt. Hierbei ist es wünschenswert, eine untere Schranke zu finden, die angibt, welcher Aufwand mindestens erforderlich ist, um eine Chiffre zu brechen. Dafür müßte man aber einen Überblick über alle möglichen Angriffe haben, was wohl kaum machbar ist, da stets neue Ansätze zur Kryptoanalyse vorgestellt werden. So konnten einige Blockchiffrierer, die gegenüber konventionellen Attacken weitgehend immun waren, vor neuen Ansätzen wie der differentiellen (Kapitel 4) oder linearen (Kapitel 5) Kryptoanalyse nicht bestehen.

Um ein einheitliches Maß für den Aufwand eines Angriffes anzugeben, wird als Basiseinheit die Zeit einer Verschlüsselung verwendet. Der Aufwand eines Angriffs besteht immer aus zwei Teilen: der **Daten-** und **Berechnungskomplexität**. Beide Teile werden mit dieser Basiseinheit zu einem Gesamtaufwand verknüpft, der dann normalerweise in 2^n Verschlüsselungsoperationen angegeben wird. Diese Einheit ist besser geeignet als absolute Zeitangaben, da sich die Hardware stets weiterentwickelt. Neben dieser Anzahl ist eine Angabe über den benötigten Speicherplatz zwar nicht immer gegeben, bei einigen Angriffen jedoch notwendig, da dieser nicht immer unwesentlich ist.

Ein noch wesentlich wichtigerer Punkt ist das Schlüsselmanagment. Über 90% aller erfolgreichen Hackerversuche gelangen nicht, weil der Algorithmus so schlecht war, sondern weil der Schlüssel zu unsicher war oder unsicher aufbewahrt wurde. Der beste Algorithmus nützt einem nichts, wenn ein Angreifer den Schlüssel leicht erraten kann, Beispiele dafür sind z.B.:

1. Der Schlüssel ist ein Wort einer Sprache.
2. Der Schlüssel enthält Daten, die mit dem Nutzer verbunden sind, also Geburtstage oder Telefonnummern.
3. Der Schlüssel wird nach einem festen Muster vergeben, sei es durch eine Liste oder durch einen schlechten Pseudozufallszahlengenerator.
4. Der Schlüssel wird unverschlüsselt aufgeschrieben, entweder auf einen Zettel oder in eine Datei.
5. Der Schlüssel wird mit der Nachricht zusammen verschickt (solche Kryptoprogramme gibt es tatsächlich!).
6. Der Schlüssel kann durch Bestechung erpreßt werden.

Alle Fälle sind gar nicht so abwegig, und allgemein wird das Schlüsselmanagment als der schwierigste Teil der Kryptologie angesehen. Die Lösung dieser Probleme besteht hauptsächlich aus geeigneten Protokollen zur Schlüsselverwaltung und dem Design zuverlässiger Pseudozufallszahlengeneratoren. Dieser Thematik könnte allein eine Diplomarbeit

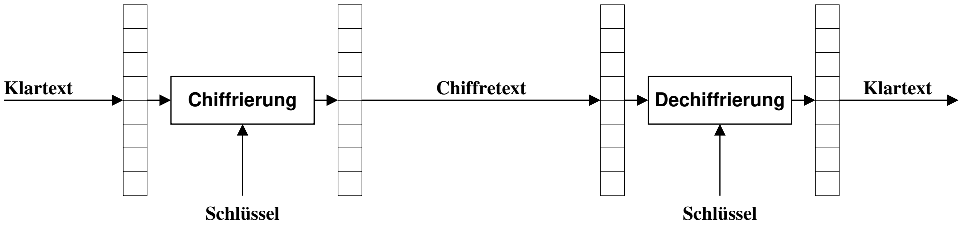

Abbildung 1.7: Der ECB-Modus

gewidmet werden, deswegen sei auf die übliche Literatur verwiesen, z.B. [MOV96] oder [Schn96].

1.6 Die Betriebsmodi eines Blockchiffrierers

Nach der Einführung des *Data Encryption Standard* kurz DES, wurden vom National Bureau of Standards 1980 [Fips81] vier Betriebsmodi für Blockchiffrierer, die den verschiedensten Ansprüchen gerecht werden sollen, veröffentlicht. Diese Modi unterscheiden sich in Hinsicht auf Verschlüsselungsrate, Fehlerfortpflanzung und Fehlererkennung, oder in einer grundlegenderen Form, wie dem Einsatz eines Blockchiffrierers als Stromchiffrierer. Im folgenden stelle ich die einzelnen Betriebsmodi vor, sowie Vor- und Nachteile der einzelnen Modi.

ECB-Modus

Die klassische Form eines Blockchiffrierers ist der **Electronic Code Book** Modus (siehe Abbildung 1.7). In diesem Modus wird eine Nachricht in n Blöcke gleicher Länge zerlegt, wobei der letzte Block eventuell geeignet aufgefüllt werden muß. Diese Blöcke werden anschließend einer nach dem anderen mit derselben Verschlüsselungsfunktion chiffriert. Die Entschlüsselung läuft nach dem gleichen Muster ab. Ist die Übertragung hingegen mit Fehlern behaftet, so kann dies unter Umständen die ganze Nachricht unlesbar machen, verantwortlich dafür ist der sogenannte Avalanche-Effekt (Abschnitt 2.5). Das Avalanche Kriterium verlangt von einem Blockchiffrierer, daß bei Änderung eines Bits des Schlüssels oder Klartextes im Mittel die Hälfte aller Chiffretextbits eines Blocks geändert werden. Es muß also dafür gesorgt werden, daß ein möglichst fehlerfreier Kanal zur Übertragung zur Verfügung steht, z.B. mittels fehlerkorrigierender Codes. Andererseits ist der ECB-Modus der einzige, der einen wahlfreien Zugriff auf jeden einzelnen Chiffretextblock zuläßt. Dies ist zugleich Vor- wie Nachteil, da wahlfreier Zugriff bei verschlüsselten Datenbanken unerläßlich ist, andererseits könnte ein Angreifer gezielt einzelne Blöcke verändern, um den Sinn einer Nachricht zu ändern. Bei der Kommunikation wird der CBC-Modus dem ECB-Modus deswegen vorgezogen.

Außerdem ist der ECB-Modus sehr anfällig gegenüber Codebuch-Analysen. Dafür sammelt der Angreifer Klartext/ Chiffretext-Paare, die er in einer Bibliothek speichert. Wenn viele Nachrichten mit dem gleichen Schlüssel erzeugt werden, so kann der Angreifer dadurch einen immer größeren Anteil entschlüsseln. Sinnvoll ist dieses Vorgehen nur dann, wenn der entsprechende Klartext eine große Redundanz besitzt, also z.B. ein Text verschlüsselt wird. Bei 64-Bit Blöcken wird eine Codebuchanalyse illusorisch, wenn der Text nur eine sehr geringe Redundanz besitzt, also z.B. 48 Bit des Klartextblocks aus zufälligen Bits besteht. Deswegen wird der ECB-Modus hauptsächlich zum Verschlüsseln von Schlüsseln benutzt. Auch wenn eine Speicherung von 2^{70} Byte zur Zeit so gut wie unmöglich sind, so ist es absehbar, daß in näherer Zukunft diese Menge durchaus verfügbar sein

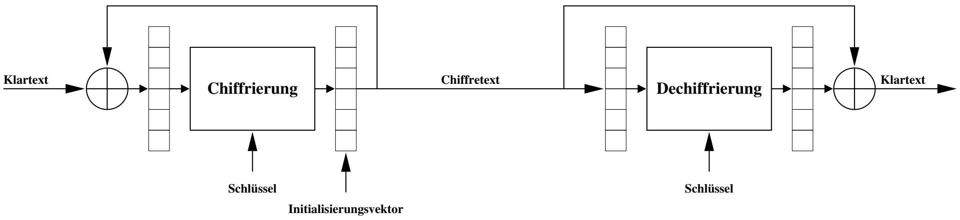

Abbildung 1.8: Der CBC-Modus

wird, weswegen damit begonnen wurde, Blockchiffrierer mit einer Blocklänge von 128 Bit zu entwickeln.

Die Codebuchanalyse wird dadurch ermöglicht, daß identische Klartextblöcke auf identische Chiffretextblöcke abgebildet werden. Dies ermöglicht die aktive Beeinträchtigung von stereotypischen Nachrichten, wie z.B. Briefe oder computergenerierte Nachrichten und Briefe. Als Beispiel seien Überweisungsformulare angeführt, die viel Struktur besitzen und wichtige Daten stets an der gleichen Stelle vermerken. Ein Angreifer kann hier leicht den Inhalt gezielt verändern, z.B. die Kontonummer. Ein solches Eingreifen könnte entweder mit dem CBC-Modus verhindert werden, oder durch Einführung einer Authentifizierung der Nachricht per digitaler Signatur oder Checksumme.

Ein Klartext wird nur selten genau in vorgegebene Blöcke passen, meistens wird der letzte Block unvollständig sein. Dieser letzte Block könnte mit Nullen oder Einsen aufgefüllt werden, dies würde aber eine Codebuchanalyse vereinfachen. Die am häufigsten angewandte Möglichkeit ist die Füllung des letzten Blockes mit zufälligen Nullen und Einsen, dann muß aber ein Füllindikator eingesetzt werden, der die Stelle markiert, wo die Nachricht zu Ende ist. Dies führt zwangsläufig zu einer Expansion des Chiffretextes, die höchstens einen ganzen Block umfäßt, in dem Fall, wo die Nachricht genau in Blöcke unterteilt werden kann. Zumindest dieser Fall kann vermieden werden, wenn am Anfang der Nachricht in einem Bit vermerkt werden kann, ob der letzte Block unvollständig ist oder nicht. Die Expansion ist ein selten erwünschter Effekt, der bei den beiden stromorientierten Verfahren nicht auftreten kann. In Tabelle 1.1 werden die Vor- und Nachteile des ECB-Modus noch einmal zusammengefaßt.

CBC-Modus

Um Codebuchanalysen vorzubeugen, wird im **Cipher Block Chaining** Modus jeder Klartextblock mit dem vorigen Chiffretextblock per bitweiser modulo 2 Addition verknüpft, bevor er verschlüsselt wird. Um den ersten Klartextblock zu verschlüsseln, wird ein zufällig gewählter **Initialisierungsvektor** (IV) zum Klartextblock hinzuaddiert. Hierdurch wird erreicht, daß eine identische Nachricht mit demselben Schlüssel dennoch auf einen unterschiedlichen Chiffretext abgebildet wird. Der IV muß dabei nicht geheimgehalten werden, sondern kann als erster Block des Chiffretextes im Klartext übertragen werden. Eine Nachricht $P_0, P_1, \ldots, P_n$ wird im CBC-Modus mit einem Schlüssel K also abgebildet auf eine Folge

$$\begin{array}{ccccc} \mathbf{E}(K, P_0 \oplus IV), & \mathbf{E}(K, P_1 \oplus C_1), & \ldots, & \mathbf{E}(K, P_i \oplus C_{i-1}), & \ldots \\ = & = & & = & \\ C_1 & C_2 & & C_i & \end{array}$$

von Chiffretextblöcken wobei $C_0 = IV$ ist. Bei fehlerfreier Übertragung kann vom Empfänger die Nachricht zurückgewonnen werden mittels

$$\mathbf{D}(K, C_1) \oplus C_0, \mathbf{D}(K, C_2) \oplus C_1), \ldots, \mathbf{D}(K, C_i) \oplus C_{i-1}, \ldots .$$

Durch die Verkettung ist es nun für den Angreifer nicht mehr möglich, die Reihenfolge der Chiffretextblöcke zu vertauschen oder durch andere zu ersetzen. Andererseits zerstören Übertragungsfehler jetzt zwei Blöcke, den ersten vollständig wieder aufgrund des Avalanche Effekts, den zweiten wegen der Addition nur an der Stelle, wo der Fehler auftrat. Ein wahlfreier Zugriff auf einen Klartextblock ist nun auch nicht mehr so einfach möglich. Bei Zugriff auf den Block C_i muß der Block C_{i-1} hinzugezogen werden; bei einer Modifizierung des Blockes C_i müssen sogar alle nachfolgenden Blöcke entschlüsselt, und anschließend wieder verschlüsselt werden.

Wie bereits erwähnt, muß der IV nicht geheimgehalten werden, sondern kann im Klartext übertragen werden. In den meisten Fällen wird der IV aber auch verschlüsselt werden, um eine Authentizität des IV sicherzustellen, da sonst zumindest der erste Block gezielt verändert werden kann. Eine weitere Möglichkeit eines gezielten Angriffs ist die absichtliche Veränderung eines Blockes, da zwar dieser Block unlesbar ist, der nachfolgende Block aber nur an den Stellen verändert wird, wo der Fehler auftrat. Aus diesem Grund darf der auf einen fehlerhaften Block folgende nicht akzeptiert werden. Also ist auch dieser Modus wieder auf einen fehlerkorrigierenden Code angewiesen. Neben den additiven Fehlern können auch Synchronisationsfehler auftreten. Diese würden im CBC-Modus die gesamte folgende Nachricht unkenntlich machen, es sei denn, diese Fehler würden in Blockgröße auftreten, dann wird nur der folgende Block an den Stellen verändert, die unterschiedlich zum eigentlichen Block sind. Im ECB-Modus richten zumindest Blockverluste keinen solchen Schaden an, der fehlerkorrigierende Code sollte also auch Synchronisationsfehler abfangen können.

Die Verkettung macht ein Auffüllen eines eventuell unvollständigen letzten Blocks mit einem konstanten Muster nicht unbedingt erforderlich. Eine Expansion der Nachricht ist wegen des zu übertragenden IV allerdings fast nicht zu vermeiden. Als IV könnte auch ein beliebiger Zufallswert genommen werden, dann ist aber der Inhalt des ersten Blockes zerstört. Da in den meisten Kommunikationssystemen vorher eine Paketnummer oder eine ähnliche Präambel übertragen wird, ist dies unter Umständen aber verschmerzbar, während eine Expansion des Chiffretextes nicht immer toleriert werden kann. Der CBC-Modus sollte zur Verschlüsselung von längeren Nachrichten eingesetzt werden, wenn kein wahlfreier Zugriff auf die Blöcke erforderlich ist. Da der CBC-Modus sehr fehleranfällig gegenüber Übertragungsfehlern ist, muß aber der Einsatz eines entsprechenden Protokolls unbedingt gewährleistet sein. Die Vor- und Nachteile des CBC-Modus sind noch einmal in Tabelle 1.1 zusammengefaßt.

OFB-Modus

In den beiden folgenden Modi wird der Blockchiffrierer als Stromchiffrierer eingesetzt. Ein Stromchiffrierer funktioniert im Prinzip wie der One-Time-Pad, nur daß statt einer echten Zufallsfolge eine Pseudozufallsfolge zum Klartext hinzuaddiert wird. Die Qualität eines Stromchiffrierers hängt also allein von der Qualität des Pseudozufallsstroms ab, insbesondere von der Vorhersagbarkeit des nächsten zu erwartenden Zeichens. Einen umfassenden Überblick über Design und Analyse von Stromchiffrierern bietet z.B. [FuRi94] oder [Daws91].

Im **Output Feedback Modus** werden jeweils m-Bit Klartextzeichen mit der vom Chiffrierer erzeugten Pseudozufallsfolge per Addition verknüpft und auf die gleiche Weise wieder entschlüsselt (da $P \oplus K \oplus K = P$). Die Zeichen werden meistens aus einem oder acht Bits bestehen. Hierzu wird ein Inputregister benötigt, daß bei Beginn der Verschlüsselung mit einem IV geladen wird. Nach jedem Verschlüsselungsschritt wird in der ursprünglichen Definition [Fips81] das Inputregister um m Bit nach links geshiftet und in die obersten m Bit der Output eingeschoben, wie in Abbildung 1.9 illustriert.

Ein Blockchiffrierer ist im OFB und CFB-Modus am langsamsten, da er nur einen von m abhängigen Bruchteil seiner maximalen Verschlüsselungsrate erreicht. Außerdem ist ein wahlfreier Zugriff im Normalfall nicht möglich. Ebenfalls prinzipbedingt gibt es

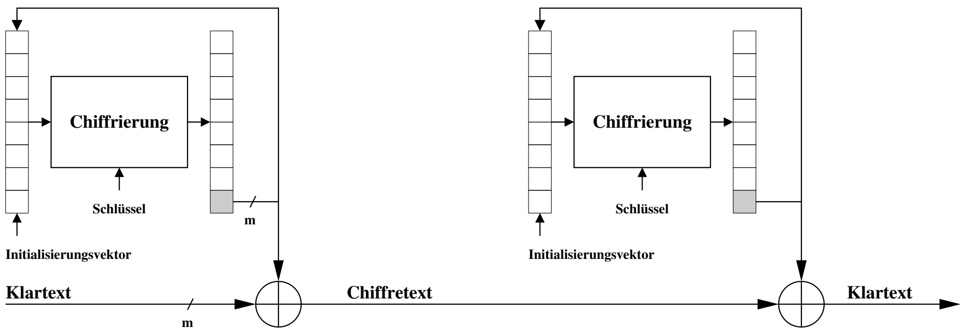

Abbildung 1.9: Der OFB-Modus

wie bei allen Stromchiffrierern im OFB-Modus keine Fehlerfortpflanzung bei additiven (Übertragungs-) Fehlern, das kann z.B. bei Sprachübertragung die Vermeidung von fehlerkorrigierenden Codes ermöglichen, da hier einzelne fehlerhafte Bits nicht weiter stören; andererseits kann die fehlende Fehlerfortpflanzung von einem Angreifer wieder zu gezielten Veränderungen des Klartextes benützt werden. Ein Synchronisationsfehler wird hingegen die gesamte folgende Nachricht zerstören, ein entsprechendes Protokoll muß also implementiert werden.

Eines der wichtigsten Kriterien für einen guten Pseudozufallszahlengenerator ist die Länge der Periode. Eine Blockchiffre kann wie in Abschnitt 1.3 erwähnt, als Permutation der 2^n möglichen Blöcke aufgefaßt werden. Wählt man nun $m = n$, so iteriert man die Funktion $x_{i+1} = \mathbf{E}(x_i)$, wobei $\mathbf{E}$ bijektiv. Die Länge der Periode hängt also von der Länge des Zyklus ab, den x_0 unter der Permutation $\mathbf{E}$ beschreibt. Aus der linearen Algebra wissen wir, daß jede Permutation $\pi \in \mathcal{S}_n$ von n Elementen eindeutig in Zyklen zerlegt werden kann. Sei

$$i_{\pi,c} = \{\text{Anzahl der Zyklen der Länge } c \text{ in } \pi \in \mathcal{S}_n\}.$$

Dann gilt

$$\sum_{\pi \in \mathcal{S}_n} i_{\pi,c} \cdot c = n!, \qquad (1 \le c \le n),$$

d.h. die Summe der Länge aller Zyklen der Länge c aller $n!$ Permutationen aus $\mathcal{S}_n$ ist $n!$[1]. Wählt man nun zu einer vorgegebenen Permutation π einen Startwert k $(1 \le k \le n)$ und iteriert diesen unter π, so ist also die Wahrscheinlichkeit, einen Zyklus der Länge c zu erhalten für alle c gleich. Der Erwartungswert für die Länge eines Zyklus ist also nur von n abhängig, genauer, der Erwartungswert ist $n/2$. Nimmt man also vereinfachend an, daß ein Schlüssel K einer Blockchiffre zufällig eine der $2^n!$ möglichen Permutationen auswählt, so beträgt die zu erwartende Zyklenlänge 2^{n-1}, im Falle eines 64-Bit Blocks und $m = n$ ist die mittlere Zyklenlänge also 2^{63}. Dieses Ergebnis deckt sich mit einer entsprechenden Untersuchung für den DES [DPa82].

Ist $m < n$ (m ist normalerweise 1 oder 8), so ist die Zyklenlänge schwieriger zu ermitteln. Es wird jetzt die Funktion $x_{i+1} = f(x_i)$ iteriert, die aber nicht mehr zwingend bijektiv ist (z.B. ist bei $m = 1$ leicht ein zweites Urbild konstruierbar, indem man nur das höchste Bit ändert, das beim Feedback ja weggeshiftet wird). In diesem Fall wird also die Annahme gemacht, daß durch den Schlüssel zufällig eine der $(2^n)^{2^n}$ möglichen Funktionen ausgewählt wird. Das zyklische Verhalten einer solchen Zufallsfunktion wird in Anhang A hergeleitet. Die durchschnittliche Dauer der Iteration, bis ein Wert x ein zweites Mal angenommen wird, zu ermitteln, ist mit der Lösung des Geburtstagsproblems (siehe Anhang

[1] Dies sieht man so: Es gibt $(c-1)!$ verschiedene Zyklen der Länge c. Die Anzahl an Möglichkeiten, c Elemente aus n auszuwählen, beträgt $\binom{n}{c}$, und $(n-c)!$ Möglichkeiten, die restlichen $n-c$ Elemente zu permutieren. Zusammen ergibt sich also als Anzahl der Zyklen der Länge c in $\mathcal{S}_n$: $\binom{n}{c}(c-1)!(n-c)! = \frac{n!}{c}$. Daraus folgt die Behauptung.

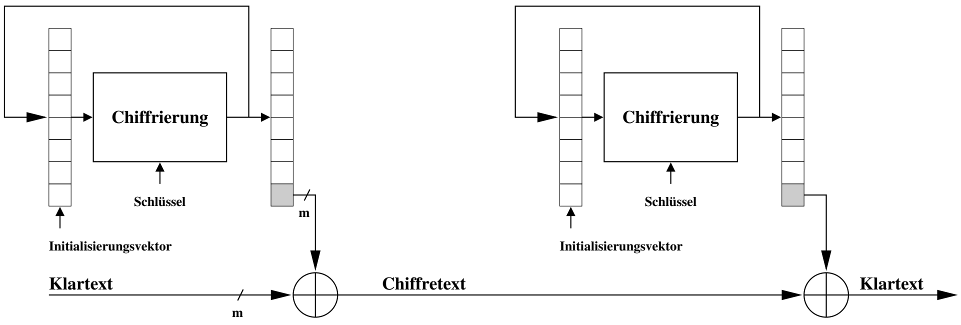

Abbildung 1.10: Der OFB-Modus nach ISO/IEC 10116

A) verwandt. Nach diesem Problem ist zu erwarten, daß nach etwa $\sqrt{t}$ Iterationen ein Wert ein zweites Mal angenommen wird und in einen Zyklus der durchschnittlichen Länge $\sqrt{\frac{\pi t}{8}}$ eintritt, wobei t die Größe des Wertebereiches ist. Im Falle einer 64-Bit Blockchiffre ist also eine mittlere Zyklenlänge von etwa $\sqrt{2^{64}} = 2^{32}$ zu erwarten. Genauere Analysen haben im Falle des DES gezeigt, daß je nach m die mittlere Zyklenlänge zwischen 2^{31} und 2^{32} liegt (siehe [DPa82]). Die Zyklenlänge für $m = 1,2,3,61,62,63$ liegt um etwa den Faktor $\sqrt{2}$ höher, da hier der Effekt der Nichtzufälligkeit am größten ist. Eine solche Periode ist aber für die meisten Anwendungen zu kurz, deswegen hat die ISO [Iso91] eine Modifizierung des OFB Modus vorgeschlagen, der den gesamten Output der Verschlüsselung als neuen Input verwendet (siehe Abbildung 1.10).

Eine weitere Möglichkeit, die Zyklenlänge zu maximieren, besteht darin, statt eines Feedbacks den *IV* einfach nur zu inkrementieren. Dieser „Zählermodus“ hat, wie man sofort sieht, die maximal mögliche Periode von 2^{64}. Ein weiterer Vorteil besteht in der Möglichkeit eines wahlfreien Zugriffs. Um das Chiffretextzeichen c_i zu entschlüsseln, ist bei der Feedbackmethode der gesamte Schlüsselstrom bis zu der gewünschten Stelle erforderlich. Im Zählermodus ist hingegen nur der IV und der Index i nötig. Voraussetzung ist natürlich, daß keine Synchronisationsfehler auftraten.

Der Initialisierungsvektor ist im Gegensatz zum CBC-Modus zwingend erforderlich. Der IV muß nicht verschlüsselt werden, da der IV allein dem Gegner nichts nützt. Allerdings wird dringend davon abgeraten, den IV kleiner als die vollen 64 Bit zu machen, da sonst Codebuchanalysen ermöglicht werden, die den Angreifer die gesamte Nachricht entschlüsseln lassen. Außerdem sollte bei gleichem Schlüssel derselbe IV nicht zweimal verwendet werden. Eine Übersicht über Vor- und Nachteile des OFB-Modus ist in Tabelle 1.2 zusammengestellt.

CFB-Modus

Der **Chained Feedback Modus** operiert wie der OFB-Modus als Stromchiffrierer auf dem Klartext. Typischerweise bestehen die Klartextzeichen aus $m = 1$ oder $m = 8$ Bits, die einzeln mit dem Pseudozufallsstrom verschlüsselt werden. Das Inputregister wird im Gegensatz zum ursprünglichen OFB-Modus (siehe Abbildung 1.9) aber mit dem addierten m-Bit Chiffretextvektor aufgefüllt (Abbildung 1.11). Eine Nachricht $P = P_1, P_2, \ldots$ wird so mit einem IV K_1 und den folgenden Schlüsselzeichen $K_i := K_{i-1} \ll C_{i-1}$ zu dem Chiffretext

$$\begin{array}{ccccc} P_1 \oplus \mathbf{E}(K_1), & P_2 \oplus \mathbf{E}(K_2), & \ldots, & P_i \oplus \mathbf{E}(K_i), & \ldots \\ = & = & & = & \\ C_1 & C_2 & \ldots & C_i & \ldots \end{array}$$

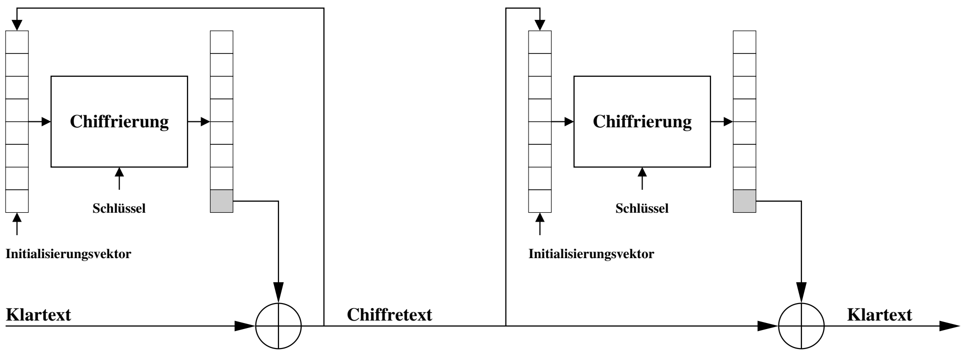

Abbildung 1.11: Der CFB-Modus

verschlüsselt und mittels

$$\begin{array}{ccccc} C_1 \oplus \mathbf{E}(K_1), & C_2 \oplus \mathbf{E}(K_2), & \ldots & C_i \oplus \mathbf{E}(K_i) & \ldots \\ = & = & & = & \\ P_1 & P_2 & \ldots & P_i & \ldots \end{array}$$

wieder entschlüsselt.

Durch die Rückkopplung hängt so ein Chiffretextzeichen vom IV, Schlüssel, Klartextzeichen und allen vorangegangenen Chiffretextzeichen ab.

Der CFB-Modus unterscheidet sich in einigen grundlegenden Eigenschaften von normalen Stromchiffrierern. Die Rückkopplung sorgt dafür, daß ein additiver Fehler nicht nur das aktuelle Zeichen zerstört, sondern auch alle $\frac{n-m}{m}$ folgenden Zeichen, also solange sich das fehlerhafte Zeichen im Inputregister des Empfängers aufhält. Falls $m = 8$, wird das fehlerhafte Zeichen nur an den Stellen verändert, wo der Fehler auftrat; deswegen sollte das auf einer vollständig zerstörten Stelle vorangegangene Zeichen ebenfalls abgelehnt werden, da es von einem Gegner beeinflußt sein könnte. Der CFB-Modus ist wie der ECB und CBC Modus in der Lage, Synchronisationsfehler in m-Bit Blocklänge relativ unbeschadet zu überleben. Der Effekt ist derselbe wie bei additiven Fehlern. Der Einsatz eines fehlerkorrigierenden Codes ist wie bei allen Modi also auch hier erforderlich. Der IV kann wie beim OFB-Modus wieder im Klartext übertragen werden, da er allein einem Angreifer nichts nützt.

Ein Gegner kann aufgrund der Art der Fehlerfortpflanzung nur am letzten m-Bit Block eine unerkennbare Veränderung durchführen. Da das letzte Zeichen einer Nachricht aber äußerst selten wichtige Informationen enthält, ist dies nicht so kritisch, solange $m = 1$ oder $m = 8$. Sollte der CFB-Modus hingegen auf einem vollen 64-Bit Block angewendet werden, so sollte stattdessen der CBC-Modus vorgezogen werden, bei dem dieser Angriff nicht möglich ist. Der CFB-Modus bietet aufgrund seiner Fehlerfortpflanzung einen besseren Schutz vor ungewollter Beeinflussung als der OFB-Modus, falls eine Nachricht durch einen als Stromchiffrierer eingesetzten Blockchiffrierer verschlüsselt werden soll. Falls die Fehlerfortpflanzung unerwünscht ist, so sollte der OFB-Modus vorgezogen werden. Die Vor- und Nachteile des CFB-Modus habe ich in Tabelle 1.2 aufgelistet.

Eine Übersicht über die unterschiedliche Fehlerfortpflanzung bei additiven Fehlern findet sich in Abbildung 1.12. Die Modi OFB und CFB wurden jeweils mit $m = 8$ betrieben.

Auswirkung auf:	Ausmaß im Modus ECB	Ausmaß im Modus CBC
additive Fehler	1 Block	1,5 Blöcke*
Synchronisationsfehler	kein	0,5 Blöcke
Wahlfreier Zugriff	ja	nein
Vertauschung von n Blöcken	kein	$2n$ Blöcke
Ersetzen eines Blocks	kein	0,5 Blöcke
Auffüllen des letzten Blocks	ja	nein
Codebuchanalyse	ja	nein
Expansion des Chiffretextes	eventuell	in der Regel ja
IV im Klartext übertragbar	—	nein
Geschwindigkeit	voll	voll $-\varepsilon$

*Ein halber Block kennzeichnet einen Block, der durch Addition nur an einzelnen Stellen verändert wird.

Tabelle 1.1: Vor- und Nachteile der Blockbetriebsmodi

Auswirkung auf:	Ausmaß im Modus OFB	Ausmaß im Modus CFB
additive Fehler	0,5 Blöcke	8,5 Blöcke
Synchronisationsfehler	total	8,5 Blöcke
Wahlfreier Zugriff	im Zählermodus	nein
Vertauschung von n Blöcken	n Blöcke	$8n$ Blöcke
Ersetzen eines Blocks	kein	8 Blöcke
Auffüllen des letzten Blocks	nein	in der Regel nein
Codebuchanalyse	nein	nein
Expansion des Chiffretextes	ja	ja
IV im Klartext übertragbar	ja	ja
Geschwindigkeit	Bruchteil	Bruchteil

In den beiden Modi wurde $m = 8$ gewählt.

Tabelle 1.2: Vor- und Nachteile der einzelnen Strombetriebsmodi

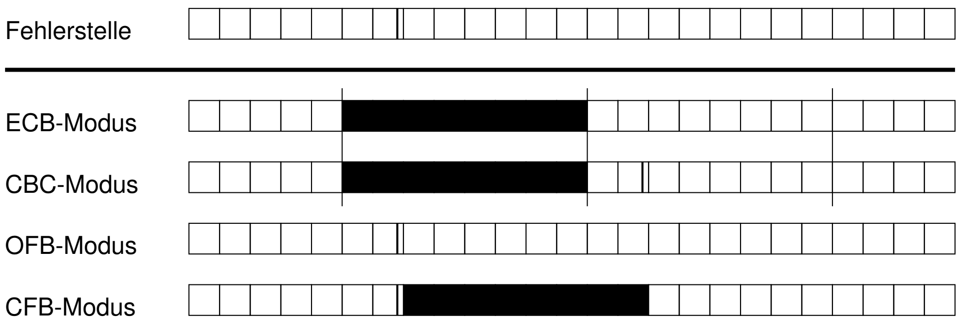

Abbildung 1.12: Fehlerfortpflanzung in den verschiedenen Betriebsmodi

Kapitel 2

Elementare Analysen und Designkriterien

In diesem Kapitel stelle ich einige grundlegende Designkriterien vor, die unter der „Black-Box“ Annahme gemacht werden können. Mit dem Ausdruck „Black Box“ ist hier gemeint, daß der Blockchiffrierer nur als gegebene Funktion vorhanden ist, ohne weitere Details der Verschlüsselungsfunktion zu kennen. Die einzigen Werte, die als bekannt vorausgesetzt werden können, sind die Block- und Schlüssellänge. Daraus folgt, daß alle hier vorgestellten Kryptoanalysen sich Schwächen des Input - Output Verhalten des Chiffrierers zunutze machen. Alle modernen Verfahren sind gegen die folgenden Angriffe immun.

Für die folgenden Attacken betrachte ich den Blockchiffrierer im ECB-Modus (siehe Abschnitt 1.6), um die Darstellung zu vereinfachen. Sofern nur wenige bekannte Klartexte erforderlich sind, sind Angriffe ohne Probleme auch im CBC-Modus möglich.

Besitzt der Angreifer ein Klartext/ Chiffretext-Paar (X,C), so kann er durch Vektoraddition mit dem vorigen Chiffretextblock C' den eigentlichen Klartextblock P herausfinden, da $X = P \oplus C'$ ist. Hat der Angreifer nämlich zu einem festen Schlüssel K sich ein umfangreiches Codebuch angelegt, so findet er mit je nach Umfang unterschiedlicher Wahrscheinlichkeit einen Chiffretextblock in der Nachricht wieder, und erhält somit über dem zugehörigen Klartext X den tatsächlichen Klartext P zurück. Noch einfacher ist natürlich ein Angriff mit wählbarem Chiffretext, so kann er sich direkt aus dem Chiffretext die entsprechenden Paare erzeugen, die zum Angriff auf den Schlüssel nötig sind. Der CBC-Modus verbessert also nicht die Verschlüsselungseigenschaften des Chiffrierers, sondern verhindert nur eine aktive Beeinträchtigung der Nachricht.

2.1 Vollständige Schlüsselsuche — Tabellensuche

Jeder Chiffrierer besitzt stets eine obere Schranke, die den Aufwand einer Kryptoanalyse beschreibt: ihre Schlüssellänge. Diese Analyse kann entweder durch Ausprobieren aller möglichen Schlüssel oder durch Zugriff auf eine vorberechnete Tabelle ermöglicht werden.

Die **vollständige Schlüsselsuche** benötigt hierzu nur ein einziges Klartext/ Chiffretext-Paar (P,C). Besteht der Klartext aus einem Text, so ist unter Umständen nur ein Chiffretextblock erforderlich und man untersucht die entschlüsselten Blöcke auf sinnvolle Worte. Bei dem Angriff wird aus dem Schlüsselraum $\mathcal{K}$ der Mächtigkeit $2^n = k$ für jeden Schlüssel $K \in \mathcal{K}$ untersucht[1], ob $C = \mathbf{E}(K,P)$. Im Mittel sind hierzu $k/2$ Verschlüsselungsoperationen erforderlich. Allerdings stößt man hier schon für kleine n an die Grenze des technisch Möglichen. Geht man davon aus, daß bald eine Verschlüsselungsoperationen

[1] Ich gehe hierbei davon aus, daß die Zuordnung eindeutig ist. Gibt es mehrere Schlüssel K mit $C = \mathbf{E}(K,P)$, so sind mehrere Textpaare erforderlich, die Komplexität wird durch diesen Faktor jedoch nicht erhöht.

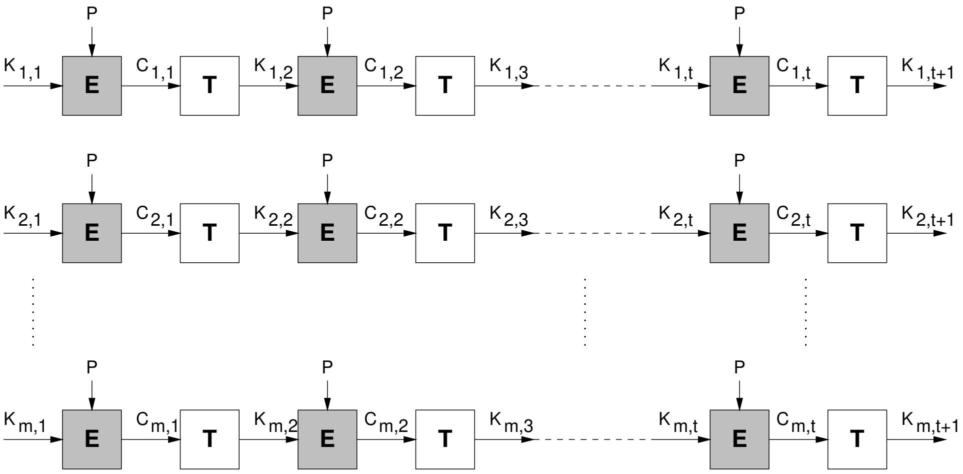

Abbildung 2.1: Ein *time-memory tradeoff*

in $2ns$ möglich ist, also bei einer Taktrate von 500 MHz, so können mit einem Baustein dieser Art in einem Jahr ca $1,57 \cdot 10^{16}$ Schlüssel durchprobiert werden. Der DES besitzt z.B. einen Schlüsselraum der Mächtigkeit $2^{56} \approx 7,2 \cdot 10^{16}$. Setzt man also 1000 dieser Chips ein, so kann ein Schlüssel im Mittel nach etwa 20 Stunden ermittelt werden.

Das andere Extrem ist der **Zugriff auf eine vorberechnete Tabelle**. Dieser Angriff benötigt hierzu ein einziges wählbares Klartext/ Chiffretext-Paar. Zunächst wird der Klartextblock mit jedem Schlüssel $K_i \in \mathcal{K}$ verschlüsselt, $C_i = \mathbf{E}(K_i, P)$. Dieses erfordert k Verschlüsselungsoperationen und Speicher für k Chiffretextblöcke. Dann muß der Angreifer Zugriff auf die Verschlüsselungsfunktion erhalten, die noch mit dem gesuchten Schlüssel geladen ist. Er verschlüsselt nun seinen Klartextblock $C = \mathbf{E}(K, P)$ und durchsucht die Tabelle, solange bis $C = C_i$. Der zu C_i gehörende Schlüssel K_i (z.B. i selber) ist dann der gesuchte Schlüssel K. Da dieser Angriff mehr Ressourcen verwendet, lohnt sich ein Einsatz erst bei wiederholter Anwendung, da die k Verschlüsselungsoperationen nur einmal durchgeführt werden müssen. Kritisch ist allerdings der verwendete Speicherplatz. Man benötigt im Falle des DES mit 64-Bit Blöcken und 56 Bit Schlüssellänge $8 \cdot 2^{56} = 2^{59} = 524.288$ Terabyte. Benutzt man die neuen DVD-ROMs, die 17 GByte speichern können, braucht man also $31,6$ Millionen DVDs.

Viele Probleme lassen sich durch einen sogenannten **time-memory-tradeoff** lösen; dies bedeutet, daß auf Kosten von Speicher Operationen eingespart werden können, und umgekehrt. Allgemein kann man folgendes Modell aufstellen: Soll eine aus N möglichen Lösungen gefunden werden, und benötigt dieser Algorithmus S Speicherzellen und Z Operationen, so gilt die Ungleichung $S \cdot Z \geq N^{2}$. Die beiden vorher besprochenen Ansätze stellen die beiden Extreme dieses Verfahrens dar. Die vollständige Schlüsselsuche benötigte nur einen Block Speicher, aber k Verschlüsselungsoperationen, während die Tabellensuche nur eine Verschlüsselungsoperation aber Unmengen an Speicher brauchte. Der effizienteste time-memory-tradeoff würde nur etwa $\sqrt{k}$ Verschlüsselungsoperationen und Speicher benötigen, im Falle von 64-Bit Blöcken und Schlüsseln wäre man hier in Größenordnungen, die beim heutigen Stand der Technik keine Probleme bereiten würden ($S = Z = 2^{32}$). Es wurde allerdings für dieses Problem bisher kein solcher Algorithmus gefunden, ob es überhaupt einen solchen gibt, konnte ebenfalls noch nicht geklärt werden. Der effizienteste bisher vorgeschlagene Algorithmus (von M.E. Hellman 1980 [Hel80]) erfordert ein Produkt $S \cdot Z$ in der Größenordnung $k^{4/3}$, wobei $S = Z = k^{2/3}$.

[2] Tatsächlich ist die rechte Seite von der Form const$\cdot N \cdot \log(N)$, die Faktoren tragen allerdings nichts zur Komplexität des Problems bei.

Schlüssel-länge	k	vollständige Suche	Tabelle (DVDs)	time-memory tradeoff	
				Zeit	Speicher
32 Bit	$4,3\cdot 10^9$	4,3 sek	2	0,006 s	1
48 Bit	$2,8\cdot 10^{14}$	3,25 Tage	123362	9 s	4
56 Bit	$7,2\cdot 10^{16}$	2,3 Jahre	$31,6\cdot 10^6$	5,8 min	152
64 Bit	$1,8\cdot 10^{19}$	587 Jahre	$8,1\cdot 10^9$	3,9 h	6120

Tabelle 2.1: Laufzeitenvergleich Schlüsselsuche — Tabellensuche

Dieses Verfahren benutzt eine vorab berechnete Tabelle und benötigt eine je nach Größe der Tabelle abhängige Anzahl an Verschlüsselungsoperationen. Der Angriff erfordert ein bestimmtes P, also ist eventuell ein frei wählbarer Klartextangriff vonnöten. Als erstes wählt man einen Klartextblock P und verschlüsselt diesen mit m verschiedenen Startschlüsseln $K_{i,1}$ ($i = 1,\ldots m$) zu Chiffretextblöcken $C_{i,1}$. Diesen Prozeß iteriert man wie in Abbildung 2.1 verdeutlicht, wobei

$$K_{i,j+1} = \mathbf{T}(C_{i,j}), \qquad 1 \leq j \leq t.$$

T bezeichne eine Transformation, die einen Chiffretextblock auf einen Schlüssel abbildet. Nun speichert man die Start- und Endschlüssel $K_{i,t+1}$ in einer Tabelle ab, hierzu sind $m\cdot t$ Verschlüsselungsoperationen und Speicher für $2m$ Schlüssel erforderlich.

Die eigentliche Kryptoanalyse funktioniert nun wie folgt: Man berechnet

$$C_0 = \mathbf{E}(K,P), \qquad Y_0 = \mathbf{T}(C_0)$$

wobei K der gesuchte Schlüssel sei[3]. Stimmt Y_0 mit einem der gespeicherten Endschlüssel $K_{r,t+1}$ ($1 \leq r \leq m$) überein, so kann der gesuchte Schlüssel durch $t-1$ Verschlüsselungsoperationen aus $K_{r,1}$ berechnet und anschließend überprüft werden. Falls nicht $K_{r,t}$, sondern ein anderes Urbild von C_0 unter **T** die korrekte Lösung ist, so wird ein falscher Alarm angenommen. Gibt es keine Übereinstimmung oder lag falscher Alarm vor, so führt man die Suche bis zu $(t-1)$-mal fort, indem man

$$Y_j = \mathbf{T}(\mathbf{E}(Y_{j-1},P))), \quad j = 1,\ldots,t-1$$

bildet. Stimmt ein Y_j mit einem der gespeicherten $C_{r,t}$ überein, so kann $K_{r,t-j}$ als der gesuchte Schlüssel K angenommen und mit Hilfe von $t-1-j$ Verschlüsselungsoperationen zurückgewonnen werden.

Das Hauptproblem dieses Angriffs ist die richtige Wahl von m und t, der Startschlüssel $K_{i,1}$ und die Wahl der Transformation **T**, da bei schlechter Wahl nur mit einer geringen Wahrscheinlichkeit der richtige Schlüssel ermittelt werden kann. Geht man nämlich davon aus, daß bei der Tabellengenerierung die auftretenden Werte $K_{i,j}$ zufällig sind, so wird nur mit einer Wahrscheinlichkeit von ${}^{k!}/_{(k^k)}$ alle k verschiedenen Schlüssel erzeugt. Sind also die Startschlüssel $K_{i,1}$ und die Transformation **T** gut gewählt, so wird im Fall $m = t = k$ nur in 63% aller Fälle der richtige Schlüssel ermittelt. Die Erfolgsquote kann erhöht werden, indem man das Produkt $m\cdot t$ deutlich größer als k wählt (die entsprechende Analyse findet man in [MeMa82]). Da im Falle $K_{i_1,j_1} = K_{i_2,j_2}$ alle folgenden Werte übereinstimmen, kann allerdings nicht von einer zufälligen Folge ausgegangen werden. M.E. Hellmann umgeht dieses Problem durch Generierung von s Tabellen dieser Art, wobei er für jede Tabelle unterschiedliche Transformationen $\mathbf{T}_i$ verwendet werden. Dieser so modifizierte Angriff erfordert nun $m\cdot s$ Speicherblöcke. Die eigentliche Kryptoanalyse benötigt im Mittel etwa $s\cdot t$ Verschlüsselungsoperationen (siehe auch [MeMa82]). Hellmann hat gezeigt, daß mit diesem Verfahren und der Wahl $m = s = t = k^{1/3}$ mehr als 80% aller Schlüssel in den Tabellen enthalten sind und damit die Erfolgswahrscheinlichkeit hoch genug ist.

[3] $E(K,\cdot)$ ist eine Black-Box, auf den internen Schlüssel K kann also nicht zugegriffen werden

Tabelle 2.1 gibt Aufschluß über Laufzeiten der vorgestellten Algorithmen bei verschiedenen Schlüssellängen. Es wird davon ausgegangen, daß eine Verschlüsselungsoperation in $2ns$ durchgeführt werden kann, Speichergrößen werden in benötigten DVD-ROMs angegeben, die jeweils 17 GB speichern können. Zudem sei daran erinnert, daß beim time-memory tradeoff einmalig $m \cdot s \cdot t = k = 2^n$ Verschlüsselungsoperationen durchgeführt werden müssen, ein in der Praxis nicht zu unterschätzender Zeitaufwand.

2.2 Klartext-Chiffretext Unabhängigkeit

Die vielleicht elementarste Eigenschaft eines Blockchiffrierers ist die Klartext — Chiffretext Unabhängigkeit. Überwacht ein Angreifer einen Nachrichtenkanal, so sollte der Angreifer nicht feststellen können, daß eine Übertragung überhaupt stattfindet. Das verlangt von dem Chiffrierer, daß der Chiffretext wie eine zufällige Folge von 0 und 1 aussieht. Da der Klartext normalerweise über eine hohe Redundanz verfügt, ist die Zufälligkeit für den Chiffretext nicht selbstverständlich, sondern verlangt einen Algorithmus, der schnell und effektiv chaotische Einflüsse in die Verschlüsselung hineinbringt. Die hier vorgestellten Verfahren werden normalerweise dazu benutzt, um Pseudozufallsfolgen von Stromchiffrierern auf Zufälligkeit zu testen. Manche dieser Verfahren lassen sich auch für Blöcke fester Länge anwenden, während andere Tests bei Sequenzen von 64 oder 128 Bit Blocklänge keinen Sinn machen, wie z.B. der Run-Test. Um die Darstellung übersichtlich zu gestalten, gehe ich auf die verwendeten Verfahren nur sehr oberflächlich ein, eine genaue Beschreibung dieser Verfahren findet sich z.B. in [Daws91] oder [FuRi94].

Um die Unabhängigkeit zu testen, generiere man zufällig eine hohe Anzahl an Klartextblöcken $P_1, \ldots, P_r$ und verschlüssele sie unter einem zufällig gewählten festen Schlüssel K zu Chiffretextblöcken $C_1, \ldots, C_r$. Nun bilde man Summen $S_i = P_i \oplus C_i, \quad i = 1, \ldots, r$. Die Blöcke S_i sollten nun zufällig erscheinen, wenn die Unabhängigkeit gegeben ist.

Die Zufälligkeit der r Vektoren kann durch mehrere Verfahren getestet werden, die ich im Folgenden vorstellen werde. Ich werde jeweils die erwarteten Werte von Zufallsvektoren berechnen; um diese Werte mit den S_i zu vergleichen, bietet sich der χ^2-Test an.

Dieses Verfahren testet die Übereinstimmung der Verteilung einer Zufallsvariablen X mit einer vorgegebenen Verteilung $F_0(x)$ (Hypothese H_0). Dazu wird der Wertebereich von X in mehrere Abschnitte $\Delta_i, (i = 1, \ldots, k)$ unterteilt. Die „theoretische“ Wahrscheinlichkeit, daß X in Δ_i liegt sei p_i, also unter der Annahme, daß die Hypothese H_0 stimmt. Aus X wird nun eine Stichprobe $(X_1, \ldots, X_r)$ entnommen und die Anzahl M_i der Stichprobenwerte in Δ_i ermittelt. Dann gilt:

$$\sum_{i=1}^{k} p_i = 1 \; ; \; \sum_{i=1}^{k} M_i = r.$$

Bei der Einteilung der Klassen Δ_i muß darauf geachtet werden, daß für die Randklassen $rp_i \geq 1$ und $rp_i \geq 5$ für die restlichen Klassen gilt, da sonst der Test unzuverlässig wird. Um das Problem zu umgehen, können entsprechend einzelne Klassen zusammengelegt werden. Sind alle Voraussetzungen erfüllt, so ist die Testgröße

$$\chi^2 = \sum_{i=1}^{k} \frac{(M_i - rp_i)^2}{rp_i} = \sum_{i=1}^{k} \frac{M_i^2}{rp_i} - r$$

asymptotisch χ^2-verteilt mit $k-1$ Freiheitsgraden, wobei die Dichte der χ^2-Verteilung mit n Freiheitsgraden gegeben ist durch

$$F(x) = \frac{x^{(n/2)-1} e^{-x/2}}{2^{n/2} \Gamma(^n/_2)}, \quad x \geq 0.$$

Überschreitet der Wert für χ^2 einen gewissen Wert χ^2_α, wobei α das Signifikanzniveau bezeichnet (meistens 5% oder 1%), so muß die Hypothese H_0 abgelehnt werden. Die entsprechenden Werte findet man in statistischen Tabellen.

Alle Tests sollten natürlich nicht nur mit einem Schlüssel durchgeführt werden. Dazu wähle man v zufällige Schlüssel, und berechne nach dem Test v χ^2-Werte mit n Freiheitsgraden, wobei der Freiheitsgrad vom durchgeführten Test abhängig ist.

Der Kolmogorov-Smirnov Test überprüft nun die Hypothese, ob die v Werte einer χ^2-Verteilung in n Freiheitsgraden entnommen sein könnten. Dazu werden die theoretischen kumulativen Verteilungswerte mit den beobachteten empirischen Verteilungswerten verglichen. Die Funktionswerte der Verteilung der Zufallsvariablen X sind gegeben durch $F(x) = P(X \leq x)$. Die Werte können aus entsprechenden Tabellen abgelesen werden. Die empirischen Verteilungsfunktion einer aus v Werten bestehenden Testreihe ist die Funktion

$$RF(x) = \frac{\#\{\text{Werte} \leq x\}}{v}.$$

Die Teststatistik des Kolmogorov-Smirnov Tests besteht nun aus dem maximalen Abstand zwischen der theoretischen Verteilungsfunktion $F(x)$ und der empirischen Verteilungsfunktion $RF(x)$,

$$D = \sup_x |F(x) - RF(x)|. \tag{2.1}$$

Ein großer Wert für D bedeutet also, daß die v χ^2-Testwerte nicht aus einer χ^2-Verteilung mit n Freiheitsgraden stammen. Also bestehen Abhängigkeiten zwischen den Klartext- und Chiffretextblöcken. Bei einem Testniveau von 5% oder 1% liegen die kritischen Werte für D bei $1,36/\sqrt{v}$ und $1,63/\sqrt{v}$ für v größer als 35.

Ich habe jetzt nur einseitige Konfidenzintervalle betrachtet, die Tests könnten natürlich auch mit zweiseitigen Konfidenzintervallen durchgeführt werden, um auch große Abweichungen nach unten feststellen zu können; die entsprechende Vorgehensweise findet man in jedem Statistikbuch.

Testmöglichkeiten für die Unabhängigkeit

Sei r die Anzahl der zu testenden Vektoren der Länge n. Sei $x \in \mathbb{Z}_2^n$. Dann sei $w(x)$ das Hamminggewicht von x, also die Anzahl an Einsen in x.

Frequenztest

Sei f_i die Anzahl der Vektoren S_j, $j = 1, \dots, r$, deren Hamminggewicht i ist. Theoretisch gibt es $\binom{n}{i}$ verschiedene Vektoren mit Gewicht i. Werden r zufällige Vektoren betrachtet, so beträgt die erwartete Anzahl an Vektoren mit Gewicht i

$$e_i = \frac{\binom{n}{i} r}{2^n}.$$

Nun vergleiche man die Werte e_i und f_i mittels des χ^2-Tests. Bei $r = 10000$ und $n = 64$ müssen die Klassen $i = 1, \dots, 8$ und $i = 56, \dots, 64$ zusammengelegt werden, wodurch sich also eine Anzahl von 48 Freiheitsgraden ergibt.

Binärdifferenzen

Sei $x = (x_1, \dots, x_n)$ ein n-Bit Vektor. Dann ist die Binärdifferenz $x' = (x_1 \oplus x_2, x_2 \oplus x_3, \dots, x_{n-1} \oplus x_n)$ ein $(n-1)$-Bit Vektor. Sei f_i' die Anzahl der Differenzvektoren der Länge $(n-1)$ und Hamminggewicht i. Es gibt insgesamt $\binom{n-1}{i}$ verschiedene Vektoren der Länge $(n-1)$ und Gewicht i. Also sind die erwarteten Anzahlen bei r Vektoren

$$e_i' = \frac{\binom{n-1}{i}}{2^{n-1}} r.$$

Die Werte e_i' und f_i' können nun wieder mit dem χ^2-Test verglichen werden.

Lineare Komplexität

Die Lineare Komplexität eines n-Bit Vektors ist gegeben durch den kürzesten LFSR[4], der den Vektor erzeugen kann. Die Lineare Komplexität kann leicht durch den Berleykamp-Massey Algorithmus[5] bestimmt werden. Sei $N_n(L)$ die Anzahl der Binärvektoren der Länge n der Linearen Komplexität L, dann ist

$$N_n(L) = \begin{cases} 2^{\min\{2n-2L,2L-1\}} & n \geq L > 0 \\ 1 & n > L = 0 \end{cases}$$

nach Rueppel [Ruep84]. Damit ist die erwartete Anzahl an Vektoren der Länge n mit Komplexität L

$$e(L) = \frac{N_n(L)}{2^n} r.$$

Sei $f(L)$ die beobachtete Anzahl an Vektoren der Komplexität L. Die Werte sollten ungefähr übereinstimmen, was mit dem χ^2-Test überprüft werden kann.

2.3 Vollständigkeit

Definition 2.1 *Eine Funktion $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ heißt* **vollständig**, *wenn jedes Outputbit von allen Inputbits abhängig ist, d.h. in jeder minimierten Booleschen Funktion, die ein Outputbit beschreibt, sind alle Inputvariablen enthalten.*

Wie Webster und Tavares in [Web85] gezeigt haben, folgt aus der Vollständigkeit von f nicht zwingend die Vollständigkeit von f^{-1}. Ist eine Funktion nicht vollständig, so kann aus den entsprechenden unvollständigen Outputs leichter auf die entsprechenden Inputbits geschlossen werden, was eine Chiffre anfälliger für bekannte oder frei wählbare Klartextangriffe macht.

Da in vielen Chiffren bijektive Substitutionen eingesetzt werden, ist die Frage nach der Existenz bijektiver S-Boxen, die vollständig sind, von großem Interesse. Man kann sich leicht davon überzeugen (durch Überprüfung aller 24 möglichen), daß es keine bijektiven 2-Bit S-Boxen gibt, die vollständig sind; während für $n > 2$ solche existieren. Darüber hinaus kann man sofort einsehen, daß es für $n > 2$ keine lineare vollständige bijektive S-Box geben kann, da es für eine Outputvariable y_i, die linear von allen Inputbits abhängt, nur zwei Möglichkeiten gibt: Entweder es gilt

$$y_i = x_1 + x_2 + \ldots + x_n$$

oder

$$y_i = x_1 + x_2 + \ldots + x_n + 1.$$

Dieses Ergebnis ist aber nicht weiter schlimm, da lineare Funktionen in einem Kryptosystem ohnehin unerwünscht sind (siehe Abschnitt 2.4).

Die Eigenschaft, vollständig zu sein, macht aus einer Funktion noch lange keine gute Chiffre, sie ist aber ein notwendiges Kriterium, wie der folgende Angriff demonstriert, der auf bekanntem Klartext beruht.

Angriff auf eine unvollständige Blockchiffre

Sei $\mathbf{E} : \mathbb{Z}_2^l \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ eine Blockchiffre und $C = \mathbf{E}(K,P)$, wobei $K \in \mathbb{Z}_2^l$ einen Schlüssel und $P,C \in \mathbb{Z}_2^n$ einen Klartext- bzw. Chiffretextblock bezeichne. Sei ferner $C = (c_1,\ldots,c_n)$ mit $c_i = f_i(k_1,\ldots,k_l,m_1,\ldots,m_n)$ für $1 \leq i \leq n$.

[4] Ein Lineares Feedback Shift Register ist der grundlegende Baustein aller Stromchiffren, siehe [FuRi94] Abschnitt 3.2

[5] siehe [FuRi94] Abschnitt 3.3

Man nehme nun o.B.d.A. an, daß die Komponenten $c_1,\dots,c_r$ des Chiffretextblocks C von den Schlüsselbits $k_{s+1},\dots,k_l$ unabhängig sind, das heißt für $1 \leq i \leq r$ ist $c_i = f_i(k_1,\dots,k_s,m_1,\dots,m_n)$. Damit kann man nun eine Blockchiffre $\mathbf{E}' : \mathbb{Z}_2^s \times \mathbb{Z}_2^n \to \mathbb{Z}_2^r$ mit $C' = \mathbf{E}'(K',P)$ definieren. Verfügt man nun über ein Klartext/ Chiffretext-Paar (P,C) und will den l-Bit Schlüssel K herausfinden, für den $C = \mathbf{E}(K,P)$ gilt, so kann man die Unvollständigkeit von $\mathbf{E}$ wie folgt ausnutzen:

1. Zuerst bestimme man C' aus C und suche diejenigen s-Bit Schlüssel, für die $C' = \mathbf{E}'(K',P)$ gilt. Dazu sind höchstens 2^s Verschlüsselungsoperationen $\mathbf{E}'$ erforderlich.

2. Nun bestimme man den gesuchten Schlüssel K aus den im ersten Schritt ermittelten K', indem man jeweils alle möglichen Ergänzungen eines s-Bit Schlüssels zu einem l-Bit Schlüssel durchprobiert. Dafür sind pro Schlüssel K' höchstens 2^{l-s} Operationen $\mathbf{E}$ erforderlich.

Erhält man im ersten Schritt nur ein mögliches K', so reduziert sich der Aufwand der Schlüsselsuche von 2^l zu $2^s + 2^{l-s}$ Verschlüsselungsoperationen. Da dies jedoch selten der Fall sein wird, so kann man die Anzahl der Lösungen im ersten Schritt deutlich reduzieren, wenn man über mehrere bekannte Klartexte verfügt. Hat man m solcher Paare, so daß $m \cdot r > s$ gilt, so reduziert sich der Aufwand für diesen Angriff tatsächlich auf ungefähr $2^s + 2^{l-s}$ Verschlüsselungsoperationen.

Um herauszubekommen, ob eine Blockchiffre vollständig ist, müßte man die Möglichkeit haben, die minimierten Booleschen Funktionen zu finden. Dieses Problem ist nicht einfach lösbar, durch einen einfachen Test kann man das Problem aber umgehen. Dieses Verfahren werde ich im Abschnitt 2.6 beschreiben.

2.4 Linearität

Definition 2.2 *Eine Funktion f heißt* **linear***, wenn jedes Outputbit (affin) linear von den Inputbits abhängt. Hängt zumindest ein Outputbit (affin) linear von den Inputbits ab, so nennt man f* **partiell-linear***.*

Aus der Linearen Algebra ist bekannt, daß eine lineare Funktion allein durch die Bilder der Basiselemente bestimmt ist, wogegen eine beliebige Funktion $\mathbb{Z}_2^n \to \mathbb{Z}_2^n$ erst durch Kenntnis aller Zuordnungen eindeutig bestimmt ist. Dadurch sind lineare oder partiell-lineare Funktionen als Chiffrierer denkbar ungeeignet.

Gordon und Retkin haben in [GoRe82] eine Schranke für den Anteil der partiell-linearen bijektiven n-Bit S-Boxen an allen möglichen bestimmt, und gezeigt, daß dieser Anteil schon für kleine n schnell gegen 0 geht, so ist der Anteil bijektiver partiell-linearer 5-Bit S-Boxen bereits kleiner als 10^{-6}.

Ein weiteres wichtiges Kriterium für die Linearität einer Funktion ist der Grad der Übereinstimmung mit ihrer besten linearen Approximation. Die **beste lineare Approximation** ist eine Funktion g, deren Funktionswerte für einen maximalen Anteil an Eingangswerten mit den Funktionswerten von f übereinstimmt. Die nicht unbedingt eindeutig bestimmte beste lineare Approximation kann leicht durch die Walsh-Transformation errechnet werden[6]. Ein ähnlicher Ansatz konnte 1993 in einer neuen Art der Kryptoanalyse gewinnbringend eingesetzt werden, der linearen Kryptoanalyse. Dieses Verfahren werde ich in Kapitel 5 genau beschreiben.

2.5 Das (strikte) Avalanche Kriterium

Eine Funktion $f : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ besitzt einen **Avalancheeffekt** (auf Deutsch etwa Lawineneffekt), wenn die Änderung eines Inputbits im Mittel die Hälfte aller Outputbits kippen läßt.

[6] siehe z.B. [Ruep86]

INITIALISIERUNG: Wähle zufällig einen Klartextblock P und r Schlüssel K_i. Sei e_i der i-te kanonische Einheitsvektor aus $\mathbb{Z}_2^m$. Am Anfang besetze man die Abhängigkeitsmatrix $A = (a_{ij})$ mit Nullen.
SCHRITT 1: Berechne $C_k = \mathbf{E}(K_k, P)$ für $1 \leq k \leq r$. Berechne für $i = 1, \ldots, m$ $C_{ki} = \mathbf{E}(K_k \oplus e_i, P)$.
SCHRITT 2: Berechne $U_{ki} = C_k \oplus C_{ki}$ und addiere reell die einzelnen Einträge aus U_{ki} zu den k-ten Spalten und i-ten Zeilen aus A für alle i und k.
SCHRITT 3: Teile alle Einträge aus A durch r.

Tabelle 2.2: **Algorithmus:** Abhängigkeitsmatrix

Diese Eigenschaft wurde von H. Feistel in [Feis73] als wichtiges Merkmal einer Blockchiffre eingeführt. Diese Eigenschaft wurde 1985 von Webster in [Web85] weiter zum **strikten Avalancheeffekt** verschärft, indem nun von der Funktion f gefordert wird, daß bei Änderung eines Inputbit sich **jedes** Outputbit mit der Wahrscheinlichkeit $\frac{1}{2}$ verändert. Diese Definition umfaßt natürlich den einfachen Avalancheeffekt, impliziert nun aber noch Eigenschaften wie Vollständigkeit und Nichtlinearität aus den vorigen Abschnitten.

Da sich die Definition auf **alle** Inputbits ausdehnt, muß zwischen zwei Arten von Avalancheeffekten unterschieden werden: dem Klartext Avalanche und dem Schlüssel Avalanche. Um einen Blockchiffrierer auf den Avalancheeffekt zu testen, hat sich ein Verfahren bewährt, welches die Abhängigkeitsmatrix aus Abschnitt 2.6 benutzt. Dieses Verfahren unterscheidet sich für beide Avalancheeffekte nur minimal.

Webster hat in [Web85] einen wählbaren Klartextangriff konstruiert, falls der Blockchiffrierer in einigen Bits einen schlechten Avalancheeffekt zeigt. In diesem Angriff versucht man, zu einem Chiffretextblock C einen Klartextblock P' zu finden, dessen Chiffretextblock C' eine möglichst geringe Hammingdistanz zu C hat, also das Hamminggewicht von $C' \oplus C$ ist klein. Nun variiere man P' in denjenigen Stellen, von denen man weiß, daß der Chiffrierer dort einen schlechten Avalancheeffekt besitzt. Mit ein wenig Glück findet man schon nach relativ wenigen Versuchen den passenden Klartextblock P. Es sollte dabei bemerkt werden, daß dieser Angriff nur schwer implementierbar ist, und die Komplexität des Angriffs gegenüber einer vollständigen Schlüsselsuche nur unwesentlich abnimmt, da man zumeist mit nur geringen Wahrscheinlichkeiten die richtigen Bits herausfindet.

2.6 Die Abhängigkeitsmatrix

Die Abhängigkeitsmatrix ist eine $m \times n$ Matrix, die einen Blockchiffrierer auf einfache Abhängigkeiten zwischen einzelnen Input- und Outputbits hin untersucht. Dabei ist entweder $m = l$ die Größe des Schlüsselraums oder $m = n$ die Größe des Klartext- bzw. Chiffretextraums. Die einzelnen Einträge a_{ij} kennzeichnen jeweils das Maß der Abhängigkeit zwischen dem i-ten Input- und j-ten Outputbit. Das Testverfahren verdeutliche ich für $m = l$, also die Messung von Abhängigkeiten zwischen Schlüssel und Chiffretext (Tabelle 2.2).

Aus dem Algorithmus erkennt man sofort einige Eigenschaften der Abhängigkeitsmatrix:

- Die Abhängigkeitsmatrix einer konstanten Funktion ist die Nullmatrix.
- Alle Einträge der Abhängigkeitsmatrix liegen zwischen 0 und 1.
- Ist eine Funktion unvollständig, so existieren Einträge gleich Null.
- Ist ein Eintrag a_{ij} gleich Eins, so hängt das j-te Outputbit linear von dem i-ten Inputbit ab.

- Ist eine Spalte a_j ein binärer Vektor, so ist die Funktion partiell - linear. Ist sogar die gesamte Matrix binär, so ist die Funktion linear.
- Ist der Mittelwert aller Matrixelemente nahe bei $\frac{1}{2}$, so besitzt die Funktion den Avalancheeffekt.
- Liegt der Wert jedes Matrixelements nahe bei $\frac{1}{2}$, so erfüllt die Funktion sogar das strikte Avalanche Kriterium.

Um die Qualität des Avalancheeffekts zu testen, bietet sich der χ^2 Test aus Abschnitt 2.2 an. Dazu benutze man die Testcharakteristik

$$\chi^2 = 2r\left(\frac{1}{2} - a_{ij}\right)^2.$$

Dies definiert $m \cdot n$ χ^2-Werte, die nun mit dem Kolmogorov-Smirnov Test (2.1) daraufhin getestet werden können, einer χ^2-Verteilung mit einem Freiheitsgrad zu entstammen, mit einem vorgegebenem Signifikanzniveau α.

Dieser Test sollte natürlich mit mehreren Klartexten/ Schlüsseln wiederholt werden, um eine bessere Aussagekraft des Tests zu ermöglichen.

2.7 Komplementeigenschaft

Definition 2.3 *Sei X ein binärer Vektor. Dann bezeichne $\bar{X}$ das* 1-er Komplement *von X, d.h. $X \oplus \bar{X} = (1,1,\ldots,1)$. Ein Blockchiffrierer besitzt die* **Komplementeigenschaft***, wenn für alle Klartextblöcke P und alle Schlüssel K*

$$\overline{\mathbf{E}(K,P)} = \mathbf{E}(\bar{K},\bar{P}).$$

Diese Eigenschaft halbiert im Falle eines frei wählbaren Klartextangriffs den zu durchsuchenden Schlüsselraum, da der Angreifer sich zwei Paare (P,C), $(\bar{P},C_1)$ erzeugen lassen kann. Um herauszufinden, ob K der richtige Schlüssel ist, erzeugt er $C_2 = \mathbf{E}(P,K)$ und kann dieses Ergebnis nun mit C und C_1 vergleichen. Entspricht $C_2 = C$, so ist K der gesuchte Schlüssel. Ist jedoch $C_2 = \bar{C_1}$, so ist $\bar{K}$ der gesuchte Schlüssel.

Bei einer gewöhnlichen Schlüsselraumsuche ist nur ein bekanntes Klartext/ Chiffretextpaar notwendig; um die Komplementeigenschaft auszunutzen, ist aber ein frei wählbares Paar nötig, deswegen ist diese Eigenschaft nur eine unwesentliche Designschwäche eines Chiffrierers, zumal dies den zu durchsuchenden Schlüsselraum nur halbiert.

2.8 Abgeschlossenheit

Definition 2.4 *Ein Chiffrierer heißt* **abgeschlossen***, wenn die Menge an Permutationen, die durch den Schlüssel ausgewählt wird, eine Untergruppe von S_{2^n} bildet.*

Diese Eigenschaft ist besonders wichtig, wenn man mehrfache Verschlüsselung in Betracht zieht, denn das bedeutet, daß der Chiffrierer abgeschlossen bzgl. Hintereinanderschaltung ist, d.h. für jeden Klartextblock P und je zwei Schlüssel K_1, K_2 gibt es einen Schlüssel K mit

$$\mathbf{E}(K_1,\mathbf{E}(K_2,P)) = \mathbf{E}(K,P). \tag{2.2}$$

Diese Eigenschaft ist in zweierlei Hinsicht sehr schlecht für einen Blockchiffrierer:

1. Mehrfache Verschlüsselung ist dasselbe wie einfache Verschlüsselung.
2. Im Falle einer Schlüsselraumsuche reduziert sich der Aufwand auf $2^{n/2}$, wenn 2^n die Größe des Schlüsselraums ist.

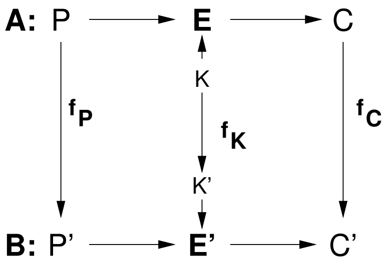

Abbildung 2.2: Linearer Faktor eines Kryptosystems

Dieser Angriff [KRS88] benötigt nur ein einziges bekanntes Klartext/ Chiffretext-Paar. Der eigentliche Schlüssel K wird dabei nicht ermittelt, sondern nur zwei Schlüssel K_1, K_2, sodaß Gleichung (2.2) erfüllt ist. Die Anzahl $2^{n/2}$ beruht auf einem dem Geburtstagsparadoxon verwandten kombinatorischen Problem (Anhang A).

2.9 Lineare Faktoren

Auch wenn eine Funktion f zunächst alle vorher beschriebenen Eigenschaften besitzt, so kann es dennoch vorkommen, daß die Booleschen Funktionen, die Linearkombinationen von Outputvariablen beschreiben, diese Eigenschaften nicht mehr besitzen. Als Beispiel sei die 3-Bit S-Box $f : \mathbb{Z}_2^3 \to \mathbb{Z}_2^3$ gegeben durch

$$\begin{aligned} y_0 &= x_0x_1 + x_1 + x_2 \\ y_1 &= x_1x_2 + x_0 + x_1 + x_2 + 1 \\ y_2 &= x_1x_2 + x_0 + x_2 + 1. \end{aligned}$$

Man sieht sofort, daß f vollständig und nichtlinear, die Summe der Outputs y_1 und y_2 aber nur von x_1 abhängig ist. Mit anderen Worten, die S-Box f', die durch

$$\begin{aligned} y_0 &= x_0x_1 + x_1 + x_2 \\ y_1 &= x_1x_2 + x_0 + x_1 + x_2 + 1 \\ y_2' &= y_1 + y_2 = x_1 \end{aligned}$$

gegeben ist, ist partiell-linear und insbesondere unvollständig. Diese Schwächen können, wie bereits oben beschrieben, zur Kryptoanalyse von f' ausgenutzt werden. Diese Schwäche war aber auch hintergründig bereits eine Schwäche der Funktion f. Solche Eigenschaften werden allgemein als lineare Strukturen bezeichnet, und insbesondere die linearen Faktoren spielen hierbei eine große Rolle.

Definition 2.5 *Ein Kryptosystem* **A** *heißt ein* **Faktor** *eines Kryptosystems* **B**, *wenn es Abbildungen f_P, f_K, f_C der Klartext-, Schlüssel- und Chiffretexträume aufeinander gibt, so daß das Diagramm 2.2 kommutiert, d.h. falls für alle Klartextblöcke P und alle Schlüssel K gilt:*

$$\mathbf{E}'(f_K(K), f_P(P)) = f_C(\mathbf{E}(K,P)).$$

Sind die Abbildungen f_P, f_K, f_C linear, so heißt **A** *ein* **linearer Faktor** *von* **B**.

Äquivalent zur Definition eines linearen Faktors ist die folgende: Ein Blockchiffrierer besitzt einen linearen Faktor, wenn es für alle Klartexte und Schlüssel eine nichtleere Menge an Schlüsselbits gibt, so daß die Komplementierung aller dieser Bits die Vektorsumme der beiden resultierenden Chiffretexte in einer nichtleeren Menge an Chiffretextbits unverändert läßt.

Ist das Kryptosystem **A** schwächer als **B**, so ist es vorteilhaft, das System **B** über den Umweg von **A** zu brechen. Dabei lohnt sich der Aufwand in der Praxis nur bei linearen Faktoren; hier ist ein Komplexitätsgewinn vom Faktor 2^s erreichbar, wenn s die Anzahl der betroffenen Schlüsselbits ist. Lineare Faktoren könnte man auch als verallgemeinerten Begriff der Unvollständigkeit auffassen. Ein gutes Kryptosystem ist für alle möglichen Linearkombinationen seiner Komponentenfunktionen vollständig.

Das Auffinden solcher linearer Faktoren stellt allerdings einen nicht zu unterschätzenden Aufwand dar. Da alle modernen Blockchiffrierer in mehreren Runden operieren, wird diese Arbeit dadurch erleichtert, daß man hier nach Faktoren in den einzelnen Runden suchen und eine geeignete Folge solcher Faktoren dann im Gesamtsystem zur Kryptoanalyse ausnützen könnte, dies wurde z.B. von Chaum und Evertse in [ChEv85] untersucht.

Dann wird ein modifizierter *meet-in-the-middle* Angriff gestartet, der dem Angriff bei einer Unvollständigkeit nicht unähnlich ist. Seien **E**, **E1**, **E2** Blockchiffren mit $\mathbf{E} = \mathbf{E2} \cdot \mathbf{E1}$, also

$$C = \mathbf{E2}(K, \mathbf{E1}(K,P)) \quad \text{und} \quad P = \mathbf{D1}(K, \mathbf{D2}(K,C)).$$

Die Chiffrierer **E1** und **E2** sind hier die Runden 1 bis i und $(i+1)$ bis n des Chiffrierers **E**. Besitzen die Funktionen **E1** und **E2** ähnliche lineare Faktoren, gibt es also lineare Abbildungen f_P, f_K, f_{C1} und f_{C2} mit der Eigenschaft

$$\mathbf{E1}(f_K(K), f_P(P)) = f_{C1}(\mathbf{E1}(K,P))$$

und

$$\mathbf{D2}(f_K(K), f_C(C)) = f_{C2}(\mathbf{D2}(K,C)),$$

so können diese Faktoren zur Kryptoanalyse von **E** ausgenutzt werden, indem man bei einem bekannten Paar (P,C) zuerst die entsprechenden Schlüsselbits herausfindet, die zur Erzeugung der entsprechenden Chiffretextbits (Vektor C' aus C) verantwortlich sind, also den Teilschlüssel K' von K herausfindet, für den

$$f_{C1}(\mathbf{E1}(K,P)) = f_{C2}(\mathbf{D2}(K,C'))$$

gilt. Schließlich erweitert man den Teilschlüssel K' zu denjenigem Schlüssel K, für den $\mathbf{E}(K,P) = C$ gilt.

Dieser Angriff benötigt einen Aufwand von der Komplexität $2^s + 2^{l-s}$ Verschlüsselungsoperationen des Chiffrierers **E**, wenn 2^l die Größe des Schlüsselraums ist.

2.10 Mehrfache Verschlüsselung

Ist die Funktion f abgeschlossen (Abschnitt 2.8), so erübrigt sich eine mehrfache Verschlüsselung, da dadurch die kryptographische Stärke der Funktion nicht erhöht wird. Ansonsten könnte man annehmen, daß eine mehrfache Verschlüsselung die Stärke deutlich erhöht, da dadurch erstens der zu untersuchende Schlüsselraum potenziert wird, und zweitens Abhängigkeiten zwischen einzelnen Input und Outputbits noch stärker verwischt werden. Dies ist jedoch nicht so, wie ich im Folgenden ausführen werde. Eine umfangreiche Analyse der einzelnen Kaskadierungsmöglichkeiten von Chiffrierern werden in [Biha96] untersucht.

Zweifache Verschlüsselung

Die erste Möglichkeit einer Kaskadierung ist der zweifache ECB-Modus mit zwei Schlüsseln. Statt $E(K,P) = C$ wird also die Funktion $E(K_2, E(K_1,P)) = C$ ausgeführt. Dadurch ergibt sich eine effektive Schlüssellänge von 112 Bits im Falle des DES . Würde man also eine vollständige Schlüsselsuche probieren, so müßte man alle 2^{112} Möglichkeiten ausprobieren. Kann ein frei wählbarer Klartextangriff durchgeführt werden, so ist es allerdings leichter, sich alle 2^{64} möglichen Blöcke errechnen zu lassen, um ein Codebuch anzulegen. Andererseits benötigt dieses Verfahren eine mit heutigen Mitteln kaum zu realisierende Speichermenge (2^{70} Byte $= 2^{30}$ Terabyte). Dieser Angriff errechnet zwar nicht den Schlüssel, der ist aber auch nicht notwendig, da man die ganze Abbildung kennt. Eleganter ist der folgende *meet-in-the-middle* Angriff, der zuerst nur zwei bekannte Klartext/ Chiffretext-Paare benötigt:

Angenommen, man verfügt über zwei Blöcke $(P_1, C_1), (P_2, C_2)$. Zunächst verschlüssele man für alle 2^{56} möglichen Schlüssel den Klartextblock P_1 und speichere sie in einer geordneten Liste. Dann entschlüssele man den Block C_1 wieder unter allen möglichen Schlüsseln K_j und überprüfe jeweils, ob $\mathbf{D}(K_j, C_1)$ in der Liste enthalten ist. Dies wird im Mittel bei 2^{48} Schlüsseln der Fall sein. Jede dieser Möglichkeiten muß nun mit dem zweiten Paar (P_2, C_2) getestet werden. Die Wahrscheinlichkeit, daß diese Überprüfung kein eindeutiges Paar an Schlüsseln hinterläßt, beträgt $2^{48}/2^{56} = 2^{-8}$. Gegebenenfalls muß also noch ein drittes Paar hinzugezogen werden. Für diesen Angriff sind also nur 2^{58} Verschlüsselungsoperationen plus Speicherplatz für die Liste in der Größenordnung $2^{62} Byte$ notwendig.

Wie man sieht, ist der Aufwand abgesehen vom Speicherplatz nur unwesentlich größer als der eines Angriffs auf die einfache Verschlüsselung. Ungefähr die Hälfte aller zweifachen Kaskadierungsmöglichkeiten lassen sich so brechen. Der Rest benötigt raffiniertere Methoden, um gebrochen zu werden. Der dazu nötige Aufwand übersteigt allerdings den Aufwand eines Codebuchangriffs, daher sind sie nur von akademischen Interesse.

Dreifache Verschlüsselung

Der *meet-in-the-middle* Angriff der zweifachen Verschlüsselung ist auch beim Tripel ECB-Modus möglich, benötigt aber neben der Liste von 2^{56} Blöcken 2^{112} Operationen, um die beiden äußeren Verschlüsselungen aufzuheben. Damit ist dieser Modus aber immer noch der sicherste aller Tripelmodi. Einige weitere Modi benötigen neben ca. 5 Paaren um die 2^{59} Verschlüsselungen, um gebrochen zu werden. Die meisten Modi benötigen einen wählbaren Klartext oder Chiffretextangriff mit 2^{66} Paaren sowie 2^{67} Verschlüsselungen und Listen in der Größenordnung von 2^{65} Blöcken und sind damit wieder umfassender als ein Wörterbuchangriff.

Eine umfassende Auflistung aller möglichen Kombinationsmöglichkeiten sowie Verfahren, um diese zu brechen, finden sich in [Biha96]. Neben dem Tripel ECB-Modus schlägt Biham noch weitere Kaskadierungsmöglichkeiten mit drei und vier Verschlüsselungsschritten vor, die einen Aufwand von mindestens 2^{112} Verschlüsselungen benötigen, so daß auch in absehbarer Zukunft diese nicht gebrochen werden können.

Auch wenn ein Wörterbuchangriff zur Zeit kaum möglich ist, so ist er nicht gänzlich ausgeschlossen. Deswegen sollten zukünftige Kryptosysteme auf jeden Fall Blöcke der Größe 128 Bit verwenden, um diese Angriffsmöglichkeit zu erschweren, zumal auf diese Weise auch Angriffe, die auf dem Geburtstagsparadoxon beruhen, unhaltbar werden. Die Kaskadierung ist bei einem modernen Chiffrierer mit 128-Bit Blöcken und Schlüsseln zwar nicht notwendig, eine dreifache Verschlüsselung kann aber zumindest für alte Systeme wie den DES ein erstaunlich hohes Maß an Sicherheit garantieren.

Kapitel 3

Die Blockchiffrierer DES und FEAL

In diesem Kapitel stelle ich die wichtigsten Blockchiffrieralgorithmen vor. Im ersten Abschnitt werde ich den **DES** erläutern, den ersten standardisierten Verschlüsselungsalgorithmus. Aufgrund seiner Schlüssellänge von nur 56 Bits ist er allerdings von Anfang an als zu unsicher gebrandmarkt worden. Ich werde daher die heute empfohlene Variante **Triple DES** vorstellen, die den DES in einer dreifachen Verschlüsselung mit drei verschiedenen Schlüsseln verwendet. Konkurrierend zum DES wurde der **FEAL** vorgeschlagen, der vor allem auf eine Softwareimplementation hin optimiert wurde. Die heute mächtigsten kryptoanalytischen Werkzeuge aus dem zweiten Teil wurden an diesen Chiffrierern entwickelt und verfeinert. Um auch einen Algorithmus vorzustellen, der gegen diese Attacken immun ist, werde ich in Kapitel 7 einen Bewerber um den neuen Verschlüsselungsstandard AES beschreiben.

Das Vorgehen, Verschlüsselungsstandards einzuführen, stellt einen bemerkenswerten Schritt dar, da hiermit dem Angreifer ermöglicht wird, auf die genaue Designgrundlage der verwendeten Chiffre inklusive seiner öffentlichen Diskussion und bekannten Schwächen zurückzugreifen. Dieses spricht für die Algorithmen, die ja eine maximale Sicherheit bieten sollen, selbst wenn dem Angreifer ein Angriff mit bekanntem oder wählbarem Klartext möglich ist. Ein weiterer Vorteil sind die gesparten Entwicklungskosten für einen eigenen Verschlüsselungsalgorithmus. Da es sich im Laufe der Jahre gezeigt hat, daß die Entwicklung einer wirklich guten Verschlüsselungsfunktion sehr schwierig ist, ist der Rückgriff auf bewährte Standards wünschenswert. Andererseits bietet die breite Anwendung eines Algorithmus einem Kryptoanalytiker eine lohnende Angriffsfläche für umfangreiche und teure Analysen. Die derzeit rasante Entwicklung der Hardware spielt einem Analytiker hierbei in die Hände. Da aber ein Standard zumindest 10 Jahre halten muß, stellt dies enorme Anforderungen an einen neuen Verschlüsselungsalgorithmus.

3.1 DES

Das Wirtschaftsministerium der USA initiierte 1972 eine Ausschreibung für einen Verschlüsselungsstandard, der für den Gebrauch nichtmilitärischer Kommunikation geeignet sein sollte. Verbunden mit einigen gewünschten Verschlüsselungseigenschaften folgten auf den Aufruf allerdings nur wenige geeignete Vorschläge, so daß die Ausschreibung 1974 wiederholt wurde. Auch dieses Mal gab es nicht viele Einreichungen, und nur ein von IBM eingereichter Vorschlag kam als einziger in die engere Wahl. Dieser Vorschlag war eine Weiterentwicklung der von IBM entwickelten Lucifer-Chiffre, einer auf dem Feistel Prinzip arbeitenden Chiffre, die in den siebziger Jahren in Bankautomaten eingesetzt wurde. Zur ausführlichen Beschreibung der Lucifer-Chiffre siehe [Smit71].

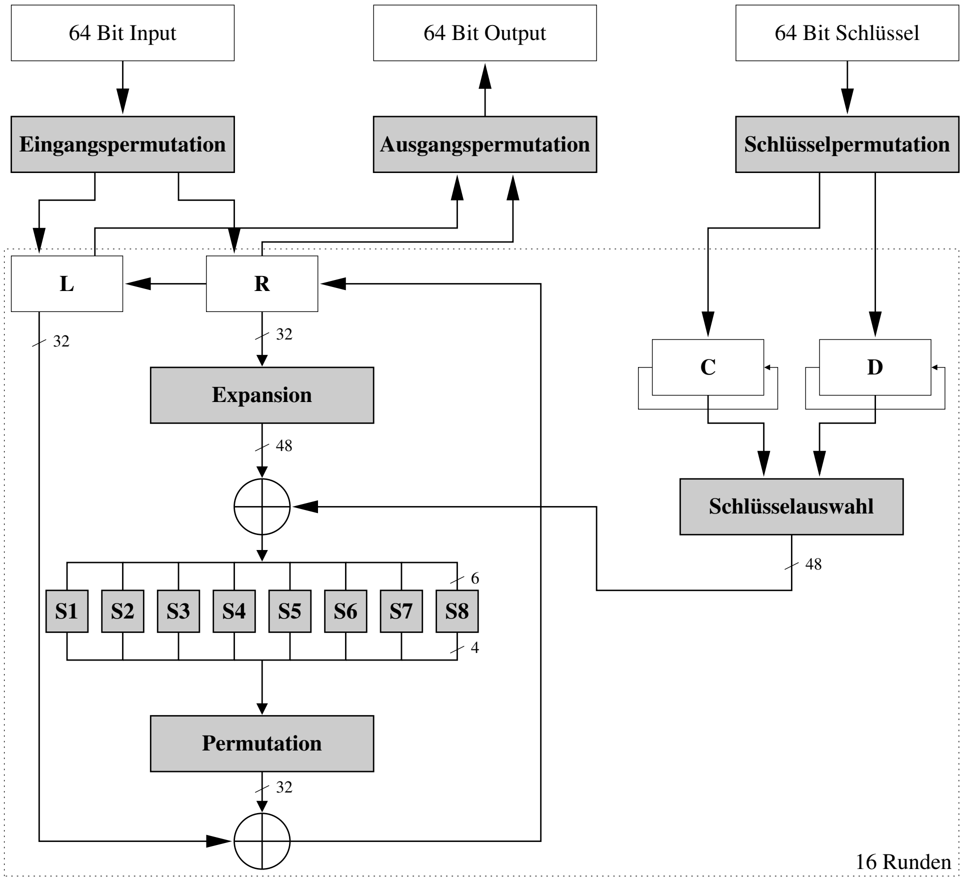

Abbildung 3.1: Der Data Encryption Standard

Der Vorschlag wurde von der *National Security Agency* der USA analysiert, nach einer Reihe von Modifikationen 1975 veröffentlicht und als Standard vorgeschlagen. Die wahren Designkriterien wurden erst nach Bekanntwerden der differentiellen Kryptoanalyse in [Cop94] veröffentlicht, um sich damals einen wissenschaftlichen Vorsprung gegenüber Konkurrenten zu sichern. Hätte man die Designkriterien schon 1975 veröffentlicht, wäre bestimmt jemand schon früher auf diesen mächtigen Angriff gestoßen. Trotz des damit verbundenen 15jährigen Argwohns über Hintertüren im Algorithmus wurden doch keine wesentlichen Schwächen gefunden, abgesehen von dem zu kleinen Schlüsselraum und einer unoptimalen Reihenfolge der S-Boxen für die lineare Kryptoanalyse (Kapitel 5). Schließlich wurde im Januar 1977 die Chiffre als Data Encryption Standard angenommen und in [Fips46] publiziert. Der DES wird wie jeder Standard regelmäßig einem kritischen Test unterworfen und wurde 1982, 1987 und 1992 erneut ratifiziert. Der DES wurde 1998 wie erwartet nicht weiter als Standard empfohlen, deswegen läuft zur Zeit eine neue Ausschreibung zu einem *Advanced Encryption Standard*, kurz AES genannt. Näheres hierzu in Kapitel 7. Die neben dem kleinen Schlüsselraum beim DES am meisten angebrachte Kritik ist die schlechte Implementierbarkeit in Software aufgrund der verwendeten Permutationen. Deshalb gibt es eine große Anzahl an *Special-Purpose-Chips* von verschiedenen Anbietern, die eine hohe Verschlüsselungsrate erzielen. Allerdings sollte erwähnt werden, daß dieser Effekt beabsichtigt war und gezielt auf eine allein durch Hardwareimplementationen effektiv beschreibbare Chiffre hingearbeitet wurde.

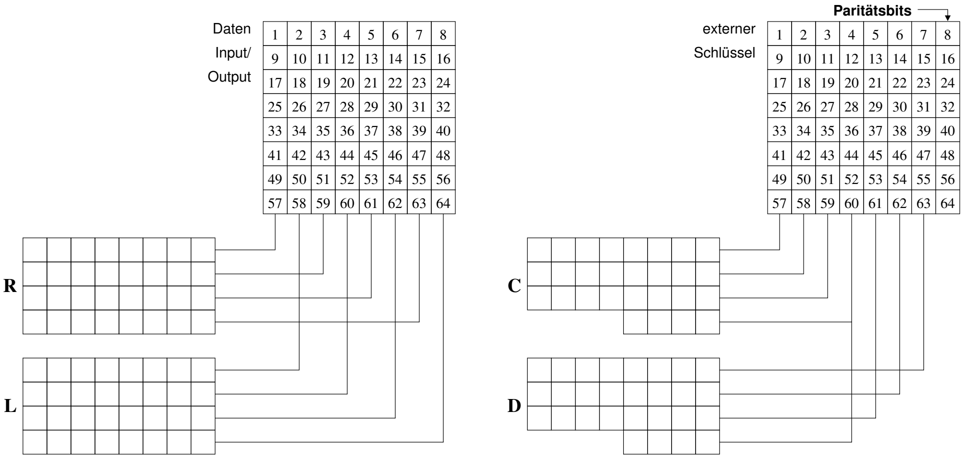

Abbildung 3.2: Die Permutationen *IP* und *PC1* des DES

3.1.1 Design des DES

Der DES ist eine Feistel-Chiffre mit 16 Verschlüsselungsrunden, die auf 64-Bit Blöcken operiert. Als Schlüssel werden aus einem 8-Byte Block jeweils die ersten sieben Bits als Schlüsselbits verwendet, was zusammen eine 56-Bit Schlüssellänge ergibt. Die nicht verwendeten achten Bits können als Parity Bits zur Authentifizierung verwendet werden. Der verwendete Algorithmus ist in Tabelle 3.1 im Detail aufgeführt (zur Struktur siehe auch Abbildung 3.1). Der Mechanismus der Feistel-Chiffre ist gut erkennbar, es wird jeweils nur die rechte Hälfte des 64-Bit Blocks verschlüsselt. Die verwendete Feistel-Funktion $F = P(S(E(R_{i-1}) \oplus K_i))$ ist eine Produktchiffre (siehe auch Abbildung 1.2), bestehend aus einer Permutation P und einer Substitution, die in acht parallel operierenden kleinen S-Boxen realisiert wird. Dieses Geflecht von Permutation und Substitution von kleinen Blöcken wird in der Kryptologie auch *Substitution-Permutation-Network*, kurz **SPN** genannt.

Die verwendeten Permutationen *IP* und *PC1* sowie ihre Umkehrungen tragen aufgrund ihrer alleinigen Anwendung am Anfang und Ende nichts zur Sicherheit des Algorithmus bei, sie erleichtern aber angeblich aufgrund ihrer Struktur das Einlesen der Blöcke in den Algorithmus, wenn sie byte-seriell eingelesen werden (siehe Abbildung 3.2). Dadurch beschleunigen sie zwar den Einsatz in speziellen Chips, eine Softwareimplementation wird dadurch aber deutlich langsamer.

Die Entschlüsselung verläuft wie bei jeder Feistel-Chiffre genauso wie die Verschlüsselung, nur die Unterschlüssel $K_1, \ldots, K_{16}$ müssen in umgekehrter Reihenfolge angewendet werden. Dies könnte durch Vorberechnung gemacht werden, eine andere Möglichkeit ist die Ersetzung der Linksshifts durch Rechtsshifts. In diesem Fall muß v_i geändert werden zu

$$v_i = \begin{cases} 0 & : \quad i = 1 \\ 1 & : \quad i = (2,9,16) \\ 2 & : \quad \text{sonst} \end{cases} \qquad 1 \leq i \leq 16.$$

3.1.2 Eigenschaften des DES

Der DES ist nach sechs Runden vollständig. Dies wurde durch die sorgfältige Wahl der Expansion E, der Permutation P und der Schlüsselauswahlfunktion $PC2$ erreicht. Betrachtet man den Ausgang einer S-Box, so gehen durch die Verknüpfung PE die vier Bits in den Input von sechs anderen S-Boxen. Nach sechs Runden sind die Abhängigkeiten aber

Algorithmus A: Verschlüsselung
INPUT: 64-Bit Klartextblock $M = m_1 \dots m_{64}$, 64-Bit Schlüssel $K = k_1 \dots k_{64}$
OUTPUT: 64-Bit Chiffretextblock $C = c_1 \dots c_{64}$

1. Generiere 16 48-Bit Rundenschlüssel K_i aus Algorithmus B
2. $(L_0, R_0) = IP(M)$ (Definition von IP in Tabelle 3.2. Der permutierte Block wird unterteilt in zwei 32-Bit Blöcke L_0, R_0)
3. Rundenoperation: Für $1 \leq i \leq 16$ berechne (L_i, R_i) mittels $L_i = R_{i-1}$ und $R_i = F(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$, wie folgt:
   (a) Expandiere R_{i-1} von 32 auf 48 Bit mittels Funktion E aus Tabelle 3.3: $E(R_{i-1}) = T$
   (b) $T' = T \oplus K_i$. Sei $T' = (B_1, \dots, B_8)$.
   (c) $T'' = (S_1(B_1), S_2(B_2), \dots, S_8(B_8))$. $S_i(B_i)$ bildet dabei $B_i = b_1 \dots b_6$ auf den 4-Bit Wert in Zeile r und Spalte c von S_i in Tabelle 3.5 ab, wobei $r = 2b_1 + b_6$ und $b_2 b_3 b_4 b_5$ die binäre Darstellung von $0 \leq c \leq 15$ ist. (Z.B. ist bei $S_1(011011)$ $r = 1$ und $c = 13$. Also ist der Wert 5, d.h. $S_1(011011) = (0101)$)
   (d) $P(T'') = R_i$, wobei P Permutation aus Tabelle 3.3.
4. $(R_{16}, L_{16}) \longmapsto b_1 \dots b_{64}$. (Vertausche also letzten Block (L_{16}, R_{16})
5. $IP^{-1}(b_1 \dots b_{64}) = C$, wobei IP^{-1} gemäß Tabelle 3.2.

Algorithmus B: Schlüsselauswahl
INPUT: 64-Bit Schlüssel $K = k_1 \dots k_{64}$ (8 Parity Bits eingeschlossen)
OUTPUT: sechzehn 48-Bit Schlüssel K_i, $1 \leq i \leq 16$

1. Sei $v_i = \begin{cases} 1 & : \quad i = (1,2,9,16) \\ 2 & : \quad \text{sonst} \end{cases} \qquad 1 \leq i \leq 16$
2. $PC1(K) = T = (C_0, D_0)$ mit C_0, D_0 jeweils 28-Bit Blöcke und $PC1$ gemäß Tabelle 3.4.
3. Für $1 \leq i \leq 16$ berechne $K_i = PC2(C_i, D_i)$, wobei $C_i = C_{i-1} \lll v_i$ und $D_i = D_{i-1} \lll v_i$ und '$\lll$' einen zirkulären Linksshift symbolisiert. $PC2$ operiert gemäß Tabelle 3.4 auf der Konkatenation $b_1 \dots b_{56}$ von (C_i, D_i).

Tabelle 3.1: **Algorithmus:** Der Data Encryption Standard

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

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

Tabelle 3.2: DES Ein- und Ausgangspermutation IP und IP^{-1}

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

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

Tabelle 3.3: DES Rundenfunktionen: Expansion E und Permutation P

PC1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
↑ Auswahl für C_i, ↓ für D_i						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tabelle 3.4: DES Schlüsselauswahlfunktionen PC1 und PC2

Zeile	Spalte															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
								S_1								
0	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7
1	0	F	7	4	E	2	D	1	A	6	C	B	9	5	3	8
2	4	1	E	8	D	6	2	B	F	C	9	7	3	A	5	0
3	F	C	8	2	4	9	1	7	5	B	3	E	A	0	6	D
								S_2								
0	F	1	8	E	6	B	3	4	9	7	2	D	C	0	5	A
1	3	D	4	7	F	2	8	E	C	0	1	A	6	9	B	5
2	0	E	7	B	A	4	D	1	5	8	C	6	9	3	2	F
3	D	8	A	1	3	F	4	2	B	6	7	C	0	5	E	9
								S_3								
0	A	0	9	E	6	3	F	5	1	D	C	7	B	4	2	8
1	D	7	0	9	3	4	6	A	2	8	5	E	C	B	F	1
2	D	6	4	9	8	F	3	0	B	1	2	C	5	A	E	7
3	1	A	D	0	6	9	8	7	4	F	E	3	B	5	2	C
								S_4								
0	7	D	E	3	0	6	9	A	1	2	8	5	B	C	4	F
1	D	8	B	5	6	F	0	3	4	7	2	C	1	A	E	9
2	A	6	9	0	C	B	7	D	F	1	3	E	5	2	8	4
3	3	F	0	6	A	1	D	8	9	4	5	B	C	7	2	E
								S_5								
0	2	C	4	1	7	A	B	6	8	5	3	F	D	0	E	9
1	E	B	2	C	4	7	D	1	5	0	F	A	3	9	8	6
2	4	2	1	B	A	D	7	8	F	9	C	5	6	3	0	E
3	B	8	C	7	1	E	2	D	6	F	0	9	A	4	5	3
								S_6								
0	C	1	A	F	9	2	6	8	0	D	3	4	E	7	5	B
1	A	F	4	2	7	C	9	5	6	1	D	E	0	B	3	8
2	9	E	F	5	2	8	C	3	7	0	4	A	1	D	B	6
3	4	3	2	C	9	5	F	A	B	E	1	7	6	0	8	D
								S_7								
0	4	B	2	E	F	0	8	D	3	C	9	7	5	A	6	1
1	D	0	B	7	4	9	1	A	E	3	5	C	2	F	8	6
2	1	4	B	D	C	3	7	E	A	F	6	8	0	5	9	2
3	6	B	D	8	1	4	A	7	9	5	0	F	E	2	3	C
								S_8								
0	D	2	8	4	6	F	B	1	A	9	3	E	5	0	C	7
1	1	F	D	8	A	3	7	4	C	5	6	B	0	E	9	2
2	7	B	4	1	9	C	E	2	0	6	A	D	F	3	5	8
3	2	1	E	7	4	A	8	D	F	C	9	0	3	5	6	B

Jede S-Box ist Abbildung von $\mathbb{Z}_2^6 \to \mathbb{Z}_2^4$, ihre Werte in $\mathbb{Z}_2^4$ werden durch ihre Hexadezimalzahl repräsentiert.

Tabelle 3.5: Die acht S-Boxen des DES

externer Schlüssel								C-Register	D-Register
01	01	01	01	01	01	01	01	0 0 0 0 0 0 0	0 0 0 0 0 0 0
FE	FE	FE	FE	FE	FE	FE	FE	F F F F F F F	F F F F F F F
1F	1F	1F	1F	0E	0E	0E	0E	0 0 0 0 0 0 0	F F F F F F F
E0	E0	E0	E0	F1	F1	F1	F1	F F F F F F F	0 0 0 0 0 0 0

Tabelle 3.6: Die schwachen Schlüssel des DES

noch so simpel, das eine Vielzahl von Attacken bekannt sind, die den DES bis zu 8 Runden in ein paar Minuten brechen (siehe z.B. Kapitel 4 oder [ChEv85]). IBM hat die Rundenzahl so hoch gewählt, daß kein damals bekannter Angriff den DES schneller als mit einer vollständigen Schlüsselsuche brechen kann.

Das strikte Avalanche-Kriterium erfüllt der DES ebenfalls in hohem Maße. Dadurch wird erreicht, daß etwaige Strukturen (Redundanz) im Klartext oder Schlüssel vollkommen verschleiert werden. Durch den Avalanche-Effekt ist der DES natürlich nicht mehr (partiell) linear. In [ChEv85] wurde gezeigt, daß der DES nur bis zu einer Rundenanzahl von acht lineare Faktoren aufweist, die zur Kryptoanalyse benutzt werden können. Bis zur Runde fünf ist dieser Gewinn durch den Faktor 2^9 auch tatsächlich nützlich, während bereits beim DES mit sechs Runden der Gewinn nur noch den Faktor vier beträgt.

Die einzige elementare Designschwäche des DES ist die Komplementeigenschaft. Zur Erinnerung: Dies bedeutet, daß für alle Schlüssel K und alle Klartextblöcke P gilt:

$$\mathbf{E}(K,P) = \overline{\mathbf{E}(\bar{K},\bar{P})}.$$

Dies liegt an dem XOR mit dem Schlüssel in der F-Funktion, da der Input der S-Boxen sich nicht ändert, wenn beide Inputs des XOR komplementiert wurden. Deswegen bleibt ein Input des XOR nach der F-Funktion gleich und das Ergebnis des XOR komplementiert sich. Allerdings nützt einem diese Eigenschaft nichts, wenn man keinen wählbaren Klartextangriff ansetzen kann, und selbst wenn, wird dadurch der zu durchsuchende Schlüsselraum nur halbiert.

Ein weiteres bisher noch nicht besprochenes Kriterium einer Blockchiffre ist die Anzahl (semi-) schwacher Schlüssel. Ein Schlüssel K heißt **schwach**, wenn

$$\mathbf{E}^2(K,P) = P \quad \text{für alle } P,$$

d.h. die Entschlüsselungsoperation ist selbstinvers. Bei einer Feistel-Chiffre bedeutet dies, daß alle Teilschlüssel identisch sind. Beim DES passiert das genau dann, wenn die Register C und D jeweils nur Nullen oder Einsen enthalten. Man überlegt sich leicht, daß die in Tabelle 3.6 angegebenen Schlüssel diese Eigenschaft haben. Schwache Schlüssel haben zudem die Eigenschaft, daß die Verschlüsselung 2^{32} Fixpunkte besitzt (siehe [Cop85]). Dies kommt daher, daß es 2^{32} Klartexte P gibt, so daß nach acht Runden die Register L und R die gleichen Werte enthalten. Dadurch wird die Verschlüsselung der ersten acht Runden in den letzten acht Runden wieder rückgängig gemacht, so daß am Ende wieder der Klartext dasteht.

Darüberhinaus gibt es auch sogenannte semischwache Schlüssel, die in Paaren auftreten. Ein Paar von Schlüsseln (K_1, K_2) heißt **semischwach**, wenn

$$\mathbf{E}(K_1, \mathbf{E}(K_2, P)) = P.$$

Eine Entschlüsselung mittels K_2 ist also dasselbe wie eine Verschlüsselung mit K_1. Dies kommt daher, daß durch die Schlüsselauswahlfunktion die internen Schlüssel des einen Schlüssels in genau der umgekehrten Reihenfolge der internen Schlüssel des zweiten Schlüssels erzeugt werden. Praktisch bedeutet dies, daß es nur zwei verschiedene interne Schlüssel gibt. Die Register C und D enthalten dadurch entweder nur Nullen und Einsen,

externer Schlüssel	C-Register	D-Register
01 FE 01 FE 01 FE 01 FE	A A A A A A A	A A A A A A A
FE 01 FE 01 FE 01 FE 01	5 5 5 5 5 5 5	5 5 5 5 5 5 5
1F E0 1F E0 1F E0 1F E0	A A A A A A A	5 5 5 5 5 5 5
E0 1F E0 1F E0 1F E0 1F	5 5 5 5 5 5 5	A A A A A A A
01 E0 01 E0 01 F1 01 F1	A A A A A A A	0 0 0 0 0 0 0
E0 01 E0 01 F1 01 F1 01	5 5 5 5 5 5 5	0 0 0 0 0 0 0
1F FE 1F FE 0E FE 0E FE	A A A A A A A	F F F F F F F
FE 1F FE 1F FE 0E FE 0E	5 5 5 5 5 5 5	F F F F F F F
01 1F 01 1F 01 0E 01 0E	0 0 0 0 0 0 0	A A A A A A A
1F 01 1F 01 0E 01 0E 01	0 0 0 0 0 0 0	5 5 5 5 5 5 5
E0 FE E0 FE F1 FE F1 FE	F F F F F F F	A A A A A A A
FE E0 FE E0 FE F1 FE F1	F F F F F F F	5 5 5 5 5 5 5

Tabelle 3.7: Semi-schwache Schlüssel des DES

oder die Sequenz (01) oder (10). Es gibt sechs Paare solcher semischwachen Schlüssel, die in Tabelle 3.7 aufgeführt sind. Vier dieser 12 semischwachen Schlüssel besitzen jeweils 2^{32} Antifixpunkte, also Klartexte P, für die $\mathbf{E}(K,P) = \bar{P}$ gilt.

Dieses Prinzip kann man noch dahingehend ausweiten, diejenigen Schlüssel auszusortieren, die nur vier verschiedene Teilschlüssel erzeugen, dies sind weitere 240 Schlüssel. Insgesamt gibt es also 256 Schlüssel, die in kritischen Situationen nicht eingesetzt werden sollten. Bei einer zufälligen Wahl ist die Wahrscheinlichkeit, einen solchen zu wählen, aber verschwindend gering (etwa 1 zu 10^{14}).

Der größte Anteil an der Sicherheit des DES ist in der Wahl der S-Boxen begründet. Die Kriterien, nach denen diese S-Boxen erstellt wurden, sind allerdings erst 1994 in [Cop94] veröffentlicht worden, wodurch immer wieder der Verdacht geäußert wurde, sie wären Funktionen mit Hintertüren. Durch die Designkriterien zeigen die S-Boxen aber durchaus einige Strukturen, die von zufällig gewählten deutlich abweichen.

So erfüllt zwar der DES das strikte Avalanche-Kriterium, die einzelnen S-Boxen ändern aber stets mindestens zwei Bits, sie erfüllen das Kriterium also nicht. Biham und Shamir konnten in [BiSh93] zeigen, daß der DES leichter zu brechen ist, wenn man die Reihenfolge der S-Boxen austauscht oder durch zufällig gewählte ersetzt. Nach heutigem Wissen kann man annehmen, daß der DES keine Hintertüren besitzt. Die Strukturen, die gefunden wurden, scheinen durch die unerwartete Immunität des DES gegenüber der differentiellen Kryptoanalyse begründet worden zu sein. Unerwartet deshalb, weil die differentielle Kryptoanalyse erst 1990 veröffentlicht wurde. Die Designer von IBM haben kurz nach Erscheinen dieses Angriffs zugegeben [Cop94], von diesem Angriff bereits 1975 gewußt zu haben, aber aus Gründen der nationalen Sicherheit diesen nicht bekannt geben durften.

3.1.3 Triple DES

Da der von Anfang als zu klein betrachtete Schlüsselraum von 56 Bits die Hauptkritik am DES war, ist eine zweifache oder dreifache Verschlüsselung die naheliegendste Lösung des Problems, wenn man für jeden Schritt unterschiedliche Schlüssel verwendet. Wie bereits in Abschnitt 2.10 erwähnt, bietet eine zweifache Verschlüsselung keine zusätzliche Sicherheit, aber eine dreifache Verschlüsselung in der Kombination $\mathbf{E}-\mathbf{D}-\mathbf{E}$ mit drei Schlüsseln erhöht die Komplexität auf 2^{112} im Falle des DES bei einem *meet-in-the-middle* Angriff.

Damit man den DES hintereinanderschalten kann, muß man aber Gewißheit haben, daß der DES nicht abgeschlossen ist. Experimente von Rivest, Kaliski und Sherman in [KRS88] haben gezeigt, daß mit hoher Wahrscheinlichkeit dies nicht der Fall ist. Campbell und Wiener haben in [CaWi92] gezeigt, daß eine untere Schranke für die Mächtigkeit der von DES erzeugten Untergruppe von $\mathcal{S}_{256}$ ca. 10^{2499} ist. Diese Größenordnung rechtfertigt eine mehrfache Verschlüsselung, da jeder Angriff, der auf einer kleinen Gruppenordnung beruht, keinen Gewinn gegenüber vollständiger Schlüsselsuche mehr einbringt.

Mit diesen Resultaten bietet also der Triple-DES die versprochene Sicherheit, die man von einer modernen Blockchiffre erwarten kann, bis auf eventuelle Codebuchangriffe, die auf der relativ kleinen Blockgröße von 64 Bit beruhen. Ein Nachteil des Triple-DES ist allerdings die um den Faktor drei kleinere Verschlüsselungsrate aufgrund der Hintereinanderschaltung.

3.2 FEAL

In Japan wurde 1987 von der *Nippon Telephone Corporation* (NTT) ein Alternativvorschlag zum DES der Öffentlichkeit vorgestellt, der eine deutlich bessere Softwareimplementierbarkeit auf 8-Bit Mikroprozessoren und eine hohe Verschlüsselungsrate bieten sollte. Der **Fast Data Encipherment Algorithm** FEAL wurde zuerst mit vier Runden spezifiziert, nachdem diese aber nur unzureichende Sicherheit bot, durch den FEAL Version 2 mit acht Runden ersetzt. Aber auch dieser bot noch keine vergleichbare Sicherheit, so daß FEAL-N und FEAL-NX mit dazu kamen. FEAL-N ist der FEAL mit N Runden, normalerweise 16 oder 32, und FEAL-NX arbeitet mit doppelter Schlüsselbreite. Der FEAL spielt aufgrund seiner einfachen Struktur stets eine tragende Rolle bei der Entwicklung neuer Angriffsmöglichkeiten. So wurde die differentielle und lineare Kryptoanalyse zuerst zur Brechung des FEAL benutzt.

3.2.1 Design des FEAL

Der FEAL-N ist eine Feistel-Chiffre, die einen 64-Bit Klartextblock auf einen 64-Bit Chiffretextblock mit einem 64-Bit Schlüssel abbildet. Die Feistel-Funktion F des FEAL ist allerdings deutlich einfacher als die des DES. Algorithmus 3.9 spezifiziert die Schlüsselauswahl und die eigentliche Verschlüsselung des FEAL-8 (Struktur siehe Abbildung 3.3). Die Funktion F ist eine Abbildung $F : \mathbb{Z}_2^{32} \times \mathbb{Z}_2^{16} \to \mathbb{Z}_2^{32}$, bildet also den üblichen halben Klartextblock mit einem 16-Bit Teilschlüssel auf einen 32-Bit Outputblock ab. Diese Funktion teilt den 32-Bit Block in vier 8-Bit Blöcke auf, die erst unterschiedlich addiert werden, bevor sie zwei verschiedenen S-Boxen $S_d : \mathbb{Z}_2^8 \times \mathbb{Z}_2^8 \to \mathbb{Z}_2^8$ als Input dienen, wie in Abbildung 3.4 illustriert.

Die S-Boxen S_d sind wie folgt definiert:

$$S_d(x,y) = (x+y+d \bmod 256) \lll 2,$$

x und y werden also zuerst natürlich (nicht vektoriell) addiert und das Ergebnis um zwei Stellen zyklisch links verschoben. Die Schlüsselauswahlfunktion $F_K(A,B)$ besitzt eine der Funktion F nicht unähnliche Struktur (siehe Abbildung 3.4). Auch die eigentliche Schlüsselerzeugung besitzt eine Feistel-ähnliche Struktur, wie man in Abbildung 3.3 erkennt.

Da die Operationen XOR und $\lll$ beide linear sind, ist das einzige nichtlineare Element des FEAL die Addition modulo 256. Die Entschlüsselung verläuft, wie bei jeder Feistel-Chiffre, wie die Verschlüsselung, nur mit umgekehrter Anwendung der Teilschlüssel. Genauer werden die Teilschlüssel $((K_{12},K_{13}),(K_{14},K_{15}))$ für den Eingangs-XOR (Schritt 3) und $((K_8,K_9),(K_{10},K_{11}))$ für den Ausgangs-XOR (Schritt 7) verwendet und als Rundenschlüssel die Teilschlüssel K_7 rückwärts bis K_0 (Schritt 5).

Die Addition von Schlüsselmaterial am Anfang und Ende wird als *Whitening* bezeichnet. Dieses von R. Rivest 1984 für seine DES Variante DESX vorgeschlagene Verfahren erschwert beim DESX die differentielle und lineare Kryptoanalyse beträchtlich und macht den DES sogar „immun“ gegen eine vollständige Schlüsselsuche, da der Aufwand für die

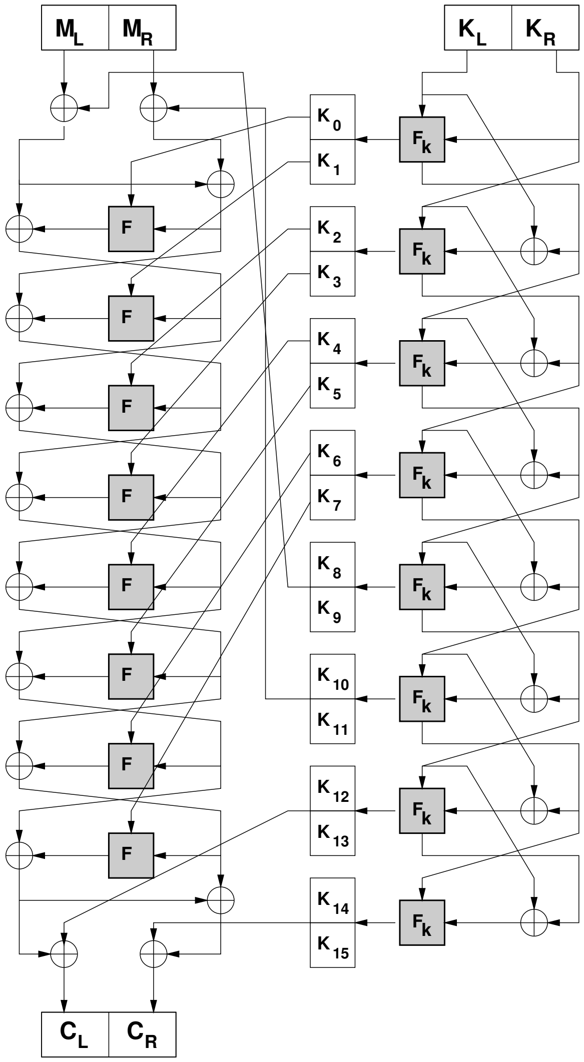

Abbildung 3.3: Struktur des FEAL-8

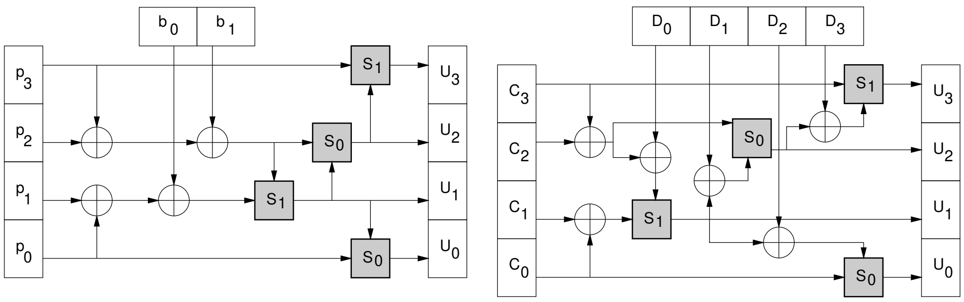

Abbildung 3.4: Struktur der Funktionen F und F_K des FEAL

Suche auf über 2^{100} erhöht wird. Beim FEAL läßt sich allerdings diese Addition vom Anfang und Ende in einer äquivalenten Beschreibung des FEAl (siehe Abbildung 5.3) in die Mitte rücken, so daß eine Analyse nicht mehr behindert wird.

Die Schlüsselerzeugung ändert sich nicht für die Varianten FEAL-N, es werden jedesmal mit Algorithmus B $N+8$ Teilschlüssel erzeugt, wobei die letzten acht für die Eingangs- und Ausgangs-XORs benutzt werden, wobei i von 1 bis $^N/_2+4$ läuft. Da beim FEAL-NX 128-Bit Schlüssel verwendet werden, ändert sich die Schlüsselerzeugung geringfügig. Die Sicherheit wird dadurch aber nicht erhöht, wenn man eine differentielle Kryptoanalyse ansetzt, da bei diesem Angriff direkt die Teilschlüssel ermittelt werden.

3.2.2 Eigenschaften des FEAL

Bereits der FEAL-4 erfüllt das strikte Avalanche-Kriterium, trotz seiner geringen Rundenanzahl. Die Funktion F sorgt dafür, daß der FEAL nach vier Runden seine Vollständigkeit erreicht, eine vollständige Abhängigkeit vom externen Schlüssel wird sogar bereits nach zwei Runden erreicht, da der Teilschlüssel K_1 vom gesamten Schlüssel abhängig ist. Da der FEAL erst nach vier Runden vollständig wird, ist der FEAL-4 wie der DES mit sechs Runden noch leicht zu brechen, da die Abhängigkeiten noch zu einfach sind. S. Murphy stellte 1990 [Mur90] einen Angriff vor, der auf der Lösung eines umfangreichen Gleichungssystem beruhte, der nur 20 frei wählbare Klartexte brauchte. Noch effektiver ist ein Angriff mit der linearen Kryptoanalyse, der fünf oder zehn bekannte Klartexte verwendet, Details hierzu in meinem Beispiel in Abschnitt 5.3.

Auch der FEAL-8 bietet noch keine ausreichende Sicherheit, mit 128 frei wählbaren Klartexten kann eine vollständige diff. Kryptoanalyse durchgeführt werden, Details hierzu in Abschnitt 4.5. Der FEAL wird erst ab einer Rundenanzahl von mindestens 24 so sicher wie der DES bei einer diff. Kryptoanalyse. Allerdings ist der FEAL dann so langsam, daß seine ursprüngliche Intention, die Geschwindigkeit, verlorengegangen ist.

Die komplexe Schlüsselerzeugung ist sowohl Vor- wie Nachteil des FEAL. Nachteilig ist der Speicheraufwand, da erst der gesamte erweiterte Schlüssel im Speicher abgelegt werden muß, bevor mit der Verschlüsselung begonnen werden kann. In Smartcard Umgebungen ist dies schwierig, da diese Karten nur über wenige Byte an RAM verfügen. Von Vorteil ist die größere Sicherheit des FEAL gegen diff. und lineare Kryptoanalyse, da man stets gezwungen wird, sämtliche Teilschlüssel komplett zu ermitteln. Beim DES kann darauf verzichtet werden, da die Kenntnis eines Schlüsselbits einer Runde sofort auch die Kenntnis eines Schlüsselbits des Hauptschlüssels und damit auch anderer Teilschlüsselbits nach sich zieht. Dies vereinfacht die Kryptoanalyse des DES *erheblich*.

	$U = F(P,B)$	$U = F_K(C,D)$
$t_1 =$	$(p_0 \oplus p_1) \oplus b_0$	$C_0 \oplus C_1$
$t_2 =$	$(p_2 \oplus p_3) \oplus b_1$	$C_2 \oplus C_3$
$U_1 =$	$S_1(t_1, t_2)$	$S_1(t_1, t_2 \oplus D_0)$
$U_2 =$	$S_0(t_2, U_1)$	$S_0(t_2, U_1 \oplus D_1)$
$U_0 =$	$S_0(p_0, U_1)$	$S_0(C_0, U_1 \oplus B_2)$
$U_3 =$	$S_1(p_3, U_2)$	$S_1(C_3, U_2 \oplus B_3)$

Tabelle 3.8: Die Funktionen F und F_K des FEAL

Algorithmus A: Verschlüsselung
INPUT: 64-Bit Klartext $M = m_1 \dots m_{64}$, 64-Bit Schlüssel $K = k_1 \dots k_{64}$
OUTPUT: 64-Bit Chiffretextblock $C = c_1 \dots c_{64}$

1. Erzeuge 16 16-Bit Teilschlüssel K_i von K aus Algorithmus B (s.u.)
2. Teile M auf in $M_L = m_1 \dots m_{32}$ und $M_R = m_{33} \dots m_{64}$.
3. $(L_0, R_0) = (M_L, M_R) \oplus ((K_8, K_9), (K_{10}, K_{11}))$.
4. $R_0 \leftarrow L_0 \oplus R_0$
5. Rundenoperation: Für $1 \leq i \leq 16$ berechne (L_i, R_i), wobei $L_i = R_{i-1}$ und $R_i = L_{i-1} \oplus F(R_{i-1}, K_{i-1})$. Berechne $F(P,B)$ mit $P = R_i = (p_0, p_1, p_2, p_3)$ und $B = K_{i-1} = (b_0, b_1)$ wie in Tabelle 3.8 angegeben.
6. $L_8 \leftarrow L_8 \oplus R_8$
7. $(R_8, L_8) = (R_8, L_8) \oplus ((K_{12}, K_{13}), (K_{14}, K_{15}))$
8. $C = (L_8, R_8)$ (Vertauschung der beiden Blöcke)

Algorithmus B: Schlüsselerzeugung
INPUT: 64-Bit Schlüssel $K = k_1 \dots k_{64}$
OUTPUT: 256-Bit interner Schlüssel (16-Bit Teilschlüssel $K_i, 0 \leq i \leq 15$).

1. (Initialisierung) $U^{(-2)} = 0$, $U^{(-1)} = k_1 \dots k_{32}$, $U^{(0)} = k_{33} \dots k_{64}$.
2. $U := (U_0, U_1, U_2, U_3)$ mit U_i 8-Bit Vektoren. Berechne $K_0, \dots K_{15}$, wobei i von 1 bis 8 läuft:
   (a) $U = F_K(U^{(i-2)}, U^{(i-1)} \oplus U^{(i-3)})$. Zur Berechnung von $F_K(C,D)$ mit $C = (C_0, C_1, C_2, C_3)$ und $D = (D_0, D_1, D_2, D_3)$ siehe Tabelle 3.8 (C_i, D_i sind jeweils 8-Bit Vektoren).
   (b) $K_{2i-2} = (U_0, U_1)$, $K_{2i-1} = (U_2, U_3)$, $U^{(i)} = U$

Tabelle 3.9: **Algorithmus:** Der FEAL-8

Teil II

Moderne Verfahren und Designkriterien

Kapitel 4

Differentielle Kryptoanalyse

Die differentielle Kryptoanalyse wurde erstmals 1990 von Eli Biham und Adi Shamir in [BiSh90] der Öffentlichkeit vorgestellt. Den Designern des DES war dieser Angriff allerdings schon 1974 als „T-Attack“ bekannt, so daß der DES im Vergleich zu anderen Chiffrierern dieser Zeit (z.B. FEAL) nicht sehr anfällig ist. Die Theorie hinter der diff. Kryptoanalyse wurde von Lai, Massey und Murphy [LMM91] in Verbindung mit Markovketten auf eine mathematisch fundierte Grundlage gestellt, die einige Details der diff. Kryptoanalyse klärte. Ich werde mich mehr an dem Ansatz von Biham und Shamir orientieren [BiSh93], da diese Form es mir erlaubt, einen konkreten Angriff zu erzeugen. Die etwas vagen Punkte in ihrer Argumentation werde ich mit den entsprechenden Aussagen von Lai genauer ausführen (siehe auch [Lai92]). Daher werde ich neben der Entwicklung der nötigen mathematischen Sätze das Verfahren an Beispielen beim DES verdeutlichen.

4.1 Grundlagen

Die diff. Kryptoanalyse ist ein Angriff mit frei wählbarem Klartext, benötigt also Zugang zum Chiffrierer, der noch mit dem zu findenden Schlüssel geladen ist. Es werden stets Paare von Klartexten betrachtet, die eine bestimmte „Differenz“ haben. Diese Differenz wird durch den Verlauf der Rundenoperation auf bestimmte Weise verändert, und durch Auswertung der Outputdifferenzen können so Rückschlüsse auf den Teilschlüssel der letzten Runde gezogen werden. Dafür muß eine geeignete Form von Differenz für das Kryptosystem gefunden werden, bezüglich der das System zu einem „Markovchiffrierer“ wird. In den meisten Fällen ist dies die XOR-Verknüpfung, beim IDEA seine „MA-Struktur“ (siehe [Lai92]).

Bezeichnungen

Hexadezimale Zahlen: kennzeichne ich durch ein x im Index, also $10_x = 16$.

Differenz: Seien X, X^* Klartextblöcke, dann ist die Differenz dieser Blöcke definiert als $X \otimes X^{*-1}$, wobei $\otimes$ eine kommutative Gruppenoperation auf dem Klartextvektorraum ist.

Folge von Differenzen: Sei Y_i der aktuelle Zwischenstand der Verschlüsselung von $Y_0 = X$ nach i von r Runden, $1 < i < r$, X Klartextblock. Sei Y_i^* analog definiert für einen zweiten Klartextblock X^*, dann bezeichne $Y_i' = Y_i \otimes Y_i^{*-1}$ die Differenz dieser Vektoren nach i Runden.

DES-spezifische Bezeichnungen:

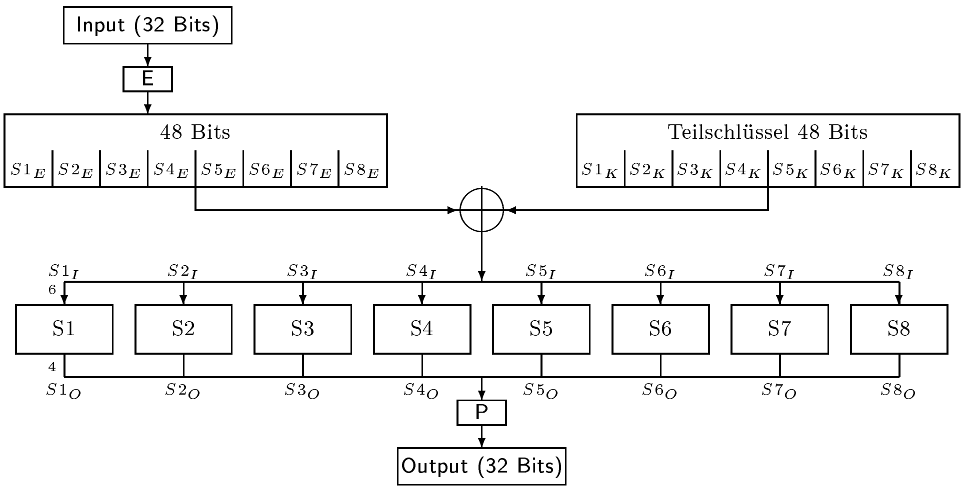

Abbildung 4.1: Die F-Funktion des DES

Klar- und Chiffretexte: Da die Ein- und Ausgangspermutationen des DES von keinerlei Bedeutung für die Sicherheit des DES sind, werde ich diese ignorieren, die Klartexte X, X^* und die Chiffretexte $Y_r = T, Y_r^* = T^*$ bezeichnen die Blöcke also entsprechend nach bzw. vor der Anwendung von IP bzw. IP^{-1}.

Ein- und Ausgänge der F-Funktion: Die 32-Bit Inputs der F-Funktion einer jeden Runde bezeichne ich durch die Kleinbuchstaben $a, b, \ldots, p$ und die 32-Bit Outputs entsprechend mit Großbuchstaben $A, B, \ldots, P$.

Teilschlüssel: Die Teilschlüssel, die zur Verschlüsselung in jeder Runde dienen, bezeichne ich mit Ki, wobei i die Runde kennzeichnet.

Permutation P: Die P Permutation des DES wird mit $P(X)$ bezeichnet.

Expansion E: Die Expansion E des DES wird mit $E(X)$ bezeichnet.

Die S-Boxen: Die S-Boxen des DES bezeichne ich mit $S1, S2, \ldots, S8$. Den Eingang der S-Box Si in Runde X ($X \in \{a, \ldots, p\}$) bezeichne ich mit Si_{IX}, den entsprechenden Ausgang mit Si_{OX}. Si_{IX} wird aus sechs Schlüsselbits und sechs Datenbits berechnet, die mit Si_{KX} und Si_{EX} bezeichnet werden. Siehe auch Abbildung 4.1.

Elementare Definitionen und Bemerkungen

Definition 4.1 *Ein* **unabhängiger Schlüssel** *ist eine Liste von Teilschlüsseln, die nicht notwendigerweise durch einen Schlüsselerzeugungsalgorithmus erzeugt worden sind.*

Beispiel 4.1: Der DES besitzt $2^{16 \cdot 48} = 2^{768}$ unabhängige Schlüssel, aber nur 2^{56} abhängige Schlüssel. □

Bemerkung: Um die mathematische Analyse zu vereinfachen, betrachte ich nur unabhängige Schlüssel. Experimentell wurde gezeigt, daß die entsprechenden Wahrscheinlichkeiten für abhängige Schlüssel ungefähr übereinstimmen, so daß in der Praxis gut damit gearbeitet werden kann. □

Zunächst eine kleine Erinnerung an Markovketten:

Definition 4.2 *Eine Folge von diskreten Zufallsvariablen $v_0, v_1, \ldots, v_r$ heißt* **Markovkette**, *wenn für alle $0 \leq i < r$ gilt:*

$$P(v_{i+1} = \beta_{i+1} | v_i = \beta_i, \ldots, v_0 = \beta_0) = P(v_{i+1} = \beta_{i+1} | v_i = \beta_i).$$

Eine Markovkette heißt **homogen**, *wenn für alle α und β $P(v_{i+1} = \beta | v_i = \alpha)$ unabhängig von i ist.*
Eine Markovkette heißt bzgl. einer Verteilung V **stationär**, *wenn eine V-Verteilung von v_0 auch eine V-Verteilung von v_i für alle $i > 0$ impliziert.*

Definition 4.3 *Ein iterierter Chiffrierer mit Rundenfunktion F heißt* **Markovchiffrierer**, *wenn es eine Gruppenoperation $\otimes$ für Differenzen gibt, so daß für beliebige $\alpha \neq e, \beta \neq e$*

$$P(Y' = \beta | X' = \alpha, X = \gamma)$$

mit $Y = F(X,K)$, $Y^* = F(X^*,K)$ und unabhängigen Schlüssel K unabhängig von γ ist; oder dazu äquivalent, wenn

$$P(Y' = \beta | X' = \alpha, X = \gamma) = P(Y_1' = \beta_1 | X' = \alpha) \tag{4.1}$$

mit unabhängigen Schlüssel K für alle γ gilt.

Der folgende Satz zeigt, warum der Ausdruck Markovchiffrierer gewählt wurde:

Satz 4.1 ([Lai92]) *Ist ein iterierter Chiffrierer ein Markovchiffrierer und die r Teilschlüssel unabhängig und gleichverteilt, dann ist die Folge von Differenzen $Y_0' = X', Y_1', \ldots, Y_r'$ eine homogene Markovkette. Ist die Differenz X' gleichverteilt über alle nicht-neutralen Elemente, so ist die Markovkette sogar stationär.*

Bemerkung: Dieser Satz stellt die Grundlage für die diff. Kryptoanalyse dar: definiert ein Chiffrierer mit einer geeigneten Operation einen Markovchiffrierer, so habe ich eine geeignete Form von Differenz gefunden, da mit dieser Differenz die Wahrscheinlichkeiten von Outputdifferenzen unabhängig vom tatsächlichen Input sind. □
Beweis: Um zu zeigen, daß die Folge von Differenzen eine Markovkette ist, reicht es für die zweite Runde zu zeigen, daß

$$P(Y_2' = \beta_2 | Y_1' = \beta_1, X' = \alpha) = P(Y_2' = \beta_2 | Y_1' = \beta_1).$$

Es gilt:

$$\begin{aligned}
& P(Y_2' = \beta_2 | Y_1' = \beta_1, X' = \alpha) \\
= \quad & \sum_\gamma P(Y_1 = \gamma, Y_2' = \beta_2 | Y_1' = \beta_1, X' = \alpha) \\
= \quad & \sum_\gamma P(Y_1 = \gamma | Y_1' = \beta_1, X' = \alpha) P(Y_2' = \beta_2 | Y_1' = \beta_1, Y_1 = \gamma, X' = \alpha) \\
= \quad & \sum_\gamma P(Y_1 = \gamma | Y_1' = \beta_1, X' = \alpha) P(Y_2' = \beta_2 | Y_1' = \beta_1, Y_1 = \gamma) \\
= \quad & \sum_\gamma P(Y_1 = \gamma | Y_1' = \beta_1, X' = \alpha) P(Y_2' = \beta_2 | Y_1' = \beta_1) \\
= \quad & P(Y_1 = \gamma | Y_1' = \beta_1, X' = \alpha),
\end{aligned}$$

wobei das dritte Gleichheitszeichen daher rührt, daß Y_1 und Y_1' zusammen Y_1^* eindeutig bestimmen und daher Y_2' nicht weiter von X' abhängig ist. Das vierte Gleichheitszeichen ist die Eigenschaft (4.1) eines Markovchiffrierers. Die Homogenität folgt dann aus der Tatsache, daß stets dieselbe Rundenfunktion ausgeführt wird.

Für alle Schlüssel K ist die Rundenfunktion $F(\cdot, K)$ eine Bijektion vom Klartextraum in den Chiffretextraum. Sind also X und $X' \neq e$ gleichverteilt, so ist auch (X, X^*) gleichverteilt. Damit ist auch (Y, Y^*) gleichverteilt und $Y'(\neq e)$ besitzt ebenfalls eine Gleichverteilung. Also ist die Gleichverteilung eine stationäre Verteilung für diese Markovkette. □

S-Box	S1	S2	S3	S4	S5	S6	S7	S8
Prozentsatz	79,4	78,6	79,6	68,5	76,5	80,4	77,2	77,1

Tabelle 4.1: Prozentsatz an möglichen Outputxors der verschiedenen S-Boxen

Definition 4.4 *Sei ein m-Bit Markovchiffrierer gegeben. Dann bezeichne* Π *die* **Übergangsmatrix** *der homogenen Markovkette* $Y'_0 = X', Y'_1, \ldots, Y'_r$. *Sei*

$$\Pi_{i,j} = P(Y'_1 = \alpha_j | X' = \alpha_i)$$

wobei $\alpha_1, \ldots, \alpha_M$ *mit einer vereinbarten Ordnung die M möglichen Werte für* X' *symbolisieren und* $M = 2^m - 1$.

Entsprechend sei Π^r *definiert durch*

$$\Pi^r = \left[p_{ij}^{(r)}\right] = \left[P(Y'_r = \alpha_j | X' = \alpha_i)\right].$$

Anmerkung: Wenn die Verschlüsselung ein Markovchiffrierer ist, so muß dies auch für die Inverse, also die Entschlüsselung gelten, da die Rundenfunktion $F(\cdot, K)$ für jeden Schlüssel K bijektiv ist. Denn folgt aus der Inputdifferenz X' für einen Bruchteil p an Tupeln $\mathbf{E}(X, X^*, K) = (Y, Y^*)$ die Outputdifferenz Y', so verändert sich dieser Bruchteil nicht bei der Inversen, da wegen der Bijektivität entsprechend $\mathbf{D}(Y, Y^*, K) = (X, X^*)$ gilt. Daher ist die Übergangsmatrix der Inversen die transponierte Übergangsmatrix Π^T.

Jede Zeile von Π^r ist eine Wahrscheinlichkeitsverteilung, die Summe über jede Zeile ist demnach 1. Es ist aber auch die Summe über jede Spalte 1, da dies die Zeilen der Übergangsmatrix der Entschlüsselung sind. Also ist Π doppelstochastisch. □

Ich betrachte nun wieder den DES. Definiert man als Differenz $\otimes$ die normale XOR-Verknüpfung, also $X \otimes X^{*-1} = X \oplus X^{*-1} = X \oplus X^*$, dann sind die Funktionen E und P linear bzgl. XOR, ebenso das zwischen den Runden auftretende XOR, eine Inputdifferenz wird also von diesen Operationen nicht beeinflußt. Die S-Boxen sind nichtlinear, so daß sich die Inputdifferenz abhängig vom gewählten Input zu unterschiedlichen Outputdifferenzen verändert. Es sind aber nicht alle Outputdifferenzen möglich, und die auftretenden sind nicht gleichverteilt. Tabelle 4.2 gibt die Differenzenverteilungstabelle der S-Box S1 an. Der Anteil an vorkommenden Outputdifferenzen liegt je nach S-Box zwischen 70% und 85%. Dies motiviert die folgenden Definitionen:

Definition 4.5 *Seien* x, y *zwei Werte, die mögliche Input und Outputdifferenzen einer S-Box repräsentieren.* **y folgt aus x** *oder* $x \to y$, *wenn aus der Inputdifferenz* x *die Outputdifferenz* y *folgen kann. Ist dies unmöglich, so sagt man,* **y folgt nicht aus x**, *oder* $x \not\to y$.

Beispiel 4.2: Sei die Inputdifferenz $S1'_I = 34_x$. Dann gibt es nur acht mögliche Outputdifferenzen, während die restlichen acht nicht auftreten können. In diesem Sinne kann $S1'_I = 34_x$ die Outputdifferenz $S1'_O = 1_x$ zur Folge haben ($34_x \to 1_x$), andererseits kann 34_x die Outputdifferenz 5_x nicht zur Folge haben ($34_x \not\to 5_x$). □

Definition 4.6 **y folgt aus x mit Wahrscheinlichkeit p durch eine S-Box**, *wenn für einen Bruchteil p an Paaren mit der Inputdifferenz x die Outputdifferenz y beträgt.*

Beispiel 4.3: $34_x \to 2_x$ folgt aus 16 aller 64 möglichen Paare in S-Box S1, also folgt 2_x aus 34_x mit Wahrscheinlichkeit $\frac{1}{4}$. □

Mit diesem Wissen kann man nun aus bekannten Inputpaaren auf die entsprechenden Schlüsselbits einer S-Box zurückschließen:

Beispiel 4.4: Angenommen, man weiß, daß $S1_E = 1_x$, $S1^*_E = 35_x$ und $S1'_O = D_x$ ist und will nun auf $S1_K$ schließen. Die Inputdifferenz ist $S1'_I = 34_x$ unabhängig von dem tatsächlichen Wert für $S1_K$. Betrachtet man Tabelle 4.2, so sieht man, daß es acht mögliche Inputwerte

Input	Output XOR															
XOR	0_x	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
0_x	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1_x	0	0	0	6	0	2	4	4	0	10	12	4	10	6	2	4
2_x	0	0	0	8	0	4	4	4	0	6	8	6	12	6	4	2
3_x	14	4	2	2	10	6	4	2	6	4	4	0	2	2	2	0
4_x	0	0	0	6	0	10	10	6	0	4	6	4	2	8	6	2
5_x	4	8	6	2	2	4	4	2	0	4	4	0	12	2	4	6
6_x	0	4	2	4	8	2	6	2	8	4	4	2	4	2	0	12
7_x	2	4	10	4	0	4	8	4	2	4	8	2	2	2	4	4
8_x	0	0	0	12	0	8	8	4	0	6	2	8	8	2	2	4
9_x	10	2	4	0	2	4	6	0	2	2	8	0	10	0	2	12
A_x	0	8	6	2	2	8	6	0	6	4	6	0	4	0	2	10
B_x	2	4	0	10	2	2	4	0	2	6	2	6	6	4	2	12
C_x	0	0	0	8	0	6	6	0	0	6	6	4	6	6	14	2
D_x	6	6	4	8	4	8	2	6	0	6	4	6	0	2	0	2
E_x	0	4	8	8	6	6	4	0	6	6	4	0	0	4	0	8
F_x	2	0	2	4	4	6	4	2	4	8	2	2	2	6	8	8
10_x	0	0	0	0	0	0	2	14	0	6	6	12	4	6	8	6
11_x	6	8	2	4	6	4	8	6	4	0	6	6	0	4	0	0
12_x	0	8	4	2	6	6	4	6	6	4	2	6	6	0	4	0
13_x	2	4	4	6	2	0	4	6	2	0	6	8	4	6	4	6
14_x	0	8	8	0	10	0	4	2	8	2	2	4	4	8	4	0
15_x	0	4	6	4	2	2	4	10	6	2	0	10	0	4	6	4
16_x	0	8	10	8	0	2	2	6	10	2	0	2	0	6	2	6
17_x	4	4	6	0	10	6	0	2	4	4	4	6	6	6	2	0
18_x	0	6	6	0	8	4	2	2	2	4	6	8	6	6	2	2
19_x	2	6	2	4	0	8	4	6	10	4	0	4	2	8	4	0
$1A_x$	0	6	4	0	4	6	6	6	6	2	2	0	4	4	6	8
$1B_x$	4	4	2	4	10	6	6	4	6	2	2	4	2	2	4	2
$1C_x$	0	10	10	6	6	0	0	12	6	4	0	0	2	4	4	0
lD_x	4	2	4	0	8	0	0	2	10	0	2	6	6	6	14	0
$1E_x$	0	2	6	0	14	2	0	0	6	4	10	8	2	2	6	2
$1F_x$	2	4	10	6	2	2	2	8	6	8	0	0	0	4	6	4
20_x	0	0	0	10	0	12	8	2	0	6	4	4	4	2	0	12
21_x	0	4	2	4	4	8	10	0	4	4	10	0	4	0	2	8
22_x	10	4	6	2	2	8	2	2	2	2	6	0	4	0	4	10
23_x	0	4	4	8	0	2	6	0	6	6	2	10	2	4	0	10
24_x	12	0	0	2	2	2	2	0	14	14	2	0	2	6	2	4
25_x	6	4	4	12	4	4	4	10	2	2	2	0	4	2	2	2
26_x	0	0	4	10	10	10	2	4	0	4	6	4	4	4	2	0
27_x	10	4	2	0	2	4	2	0	4	8	0	4	8	8	4	4
28_x	12	2	2	8	2	6	12	0	0	2	6	0	4	0	6	2
29_x	4	2	2	10	0	2	4	0	0	14	10	2	4	6	0	4
$2A_x$	4	2	4	6	0	2	8	2	2	14	2	6	2	6	2	2
$2B_x$	12	2	2	2	4	6	6	2	0	2	6	2	6	0	8	4
$2C_x$	4	2	2	4	0	2	10	4	2	2	4	8	8	4	2	6
$2D_x$	6	2	6	2	8	4	4	4	2	4	6	0	8	2	0	6
$2E_x$	6	6	2	2	0	2	4	6	4	0	6	2	12	2	6	4
$2F_x$	2	2	2	2	2	6	8	8	2	4	4	6	8	2	4	2
30_x	0	4	6	0	12	6	2	2	8	2	4	4	6	2	2	4
31_x	4	8	2	10	2	2	2	2	6	0	0	2	2	4	10	8
32_x	4	2	6	4	4	2	2	4	6,	6	4	8	2	2	8	0
33_x	4	4	6	2	10	8	4	2	4	0	2	2	4	6	2	4
34_x	0	8	16	6	2	0	0	12	6	0	0	0	0	8	0	6
35_x	2	2	4	0	8	0	0	0	14	4	6	8	0	2	14	0
36_x	2	6	2	2	8	0	2	2	4	2	6	8	6	4	10	0
37_x	2	2	12	4	2	4	4	10	4	4	2	6	0	2	2	4
38_x	0	6	2	2	2	0	2	2	4	6	4	4	4	6	10	10
39_x	6	2	2	4	12	6	4	8	4	0	2	4	2	4	4	0
$3A_x$	6	4	6	4	6	8	0	6	2	2	6	2	2	6	4	0
$3B_x$	2	6	4	0	0	2	4	6	4	6	8	6	4	4	6	2
$3C_x$	0	10	4	0	12	0	4	2	6	0	4	12	4	4	2	0
$3D_x$	0	8	6	2	2	6	0	8	4	4	0	4	0	12	4	4
$3E_x$	4	8	2	2	2	4	4	14	4	2	0	2	0	8	4	4
$3F_x$	4	8	4	2	4	0	2	4	4	2	4	8	8	6	2	2

Tabelle 4.2: Differenzverterteilungstabelle der S-Box S1

S-Box Inputs	Mögliche Schlüssel
06_x, 32_x	07_x, 33_x
10_x, 24_x	11_x, 25_x
16_x, 22_x	17_x, 23_x
$1C_x$, 28_x	$1D_x$, 29_x

S-Box Inputs	Mögliche Schlüssel
01_x, 35_x	20_x, 14_x
02_x, 36_x	23_x, 17_x
15_x, 21_x	34_x, 00_x

Tabelle 4.3: Mögliche Inputs für $34_x \to D_x$ und $34_x \to 3_x$ durch S1 mit Inputs $1_x, 35_x$ bzw. $21_x, 15_x$

gibt. Diese acht Möglichkeiten erzeugen acht Möglichkeiten für $S1_K$ per $S1_K = S1_E \oplus S1_I$ wie in Tabelle 4.3.a aufgeführt. Die Werte treten aufgrund der Kommutativität des XOR immer paarweise auf, die beiden Schlüsselwerte entsprechen jeweils $S1_E \oplus S1_I$ oder $S1_E \oplus S1_I^*$. Der richtige Wert muß jedenfalls in der Liste auftauchen.

Indem man nun weitere Paare hinzufügt, bekommt man zusätzliche Kandidaten für $S1_K$. Angenommen man verfügt über ein Paar $S1_E = 21_x$ und $S1_E^* = 15_x$, dessen Outputdifferenz $S1'_O = 3_x$ ist. Die möglichen Inputwerte sind in Tabelle 4.3.b aufgeführt. Der wahre Schlüssel muß nun in beiden Tabellen auftreten. In diesem Fall sind jetzt nur noch zwei mögliche Werte übrig, nämlich 17_x und 23_x. Diese beiden Werte sind unter dieser Inputdifferenz ununterscheidbar, da $17_x \oplus 23_x = 34_x = S1'_E$, so daß ein weiteres Inputpaar hinzugezogen werden muß, welches eine andere Inputdifferenz hat. □

Die nächste Definition erweitert die vorangegangenen Definitionen :

Definition 4.7 *Seien x und y Werte, die mögliche Input- und Outputdifferenzen der F-Funktion repräsentieren.* **y folgt aus x durch F mit Wahrscheinlichkeit p**, *wenn für einen Bruchteil p aller möglichen Inputpaare mit Inputdifferenz x, die mit allen möglichen Teilschlüsseln durch F verschlüsselt worden sind, die Outputdifferenz y gleicht. Ist $p > 0$, so schreibt man $x \to y$.*

Lemma 4.2 ([BiSh93]) *Folgt y aus x durch F mit Wahrscheinlichkeit p beim* DES *, dann folgt für jedes feste Inputpaar X, X^* mit $X \oplus X^* = x$ durch die F-Funktion, daß die Outputdifferenz y mit demselben Bruchteil p aller möglichen Teilschlüssel angenommen wird. Mit anderen Worten, der* DES *ist bezüglich der XOR Verknüpfung ein Markovchiffrierer.*

Anmerkung: Dies liegt daran, daß beim DES $S'_E = S'_I$ und ich durch die XOR Verknüpfung mit S_K für jedes feste S_E durch geeignete Wahl von S_K jedes S_I erzeugen kann, welches die gewünschte Outputdifferenz S'_O liefert. Also ist die Wahrscheinlichkeit p unabhängig von dem tatsächlichen Input Z. □

Korollar 4.3 ([BiSh93]) *Die Wahrscheinlichkeit p von $x \to y$ durch F ist das Produkt der p_i, so daß $x_i \to y_i$ durch S-Box Si $(i \in \{1, \ldots, 8\})$, wobei $x_1 \ldots x_8 = E(x)$ und $y_1 \ldots y_8 = P^{-1}(y)$.*

Anmerkung: Die vorigen Sätze erlauben mir nun eine Erweiterung des Algorithmusses von der Findung der einzelnen Schlüsselbits, die einzelne S-Boxen beeinflussen, auf den gesamten Teilschlüssel einer Runde für die F-Funktion:

1. Wähle eine geeignete Klartextdifferenz.

2. Erzeuge eine angemessene Anzahl an Klartextpaaren mit der gewählten Klartextdifferenz unter dem gesuchten Schlüssel und behalte nur die resultierenden Chiffretextpaare.

3. Leite aus jedem Paar die entsprechende Outputdifferenz für soviele S-Boxen wie möglich aus der Klartextdifferenz und dem Chiffretextpaar ab.

4. Zähle nun für jeden möglichen Schlüssel die Anzahl an Paaren, die die erwartete Outputdifferenz liefern, wenn dieser Schlüssel benutzt wird.

5. Der richtige Schlüssel ist der (hoffentlich einzige) Schlüssel, der von allen Paaren vorgeschlagen wird.

□

4.2 Charakteristiken und Differentiale

Bisher ermöglicht es die diff. Kryptoanalyse, nur einen Teilschlüssel herauszubekommen. Daher benötige ich nun nur noch eine Technik, um das Wissen von Differenzen durch möglichst viele Runden zu transportieren. Biham und Shamir tuen dies mit den Charakteristiken, die sehr einfach zu konstruieren und deren Wahrscheinlichkeiten leicht kontrollierbar sind. Die Wahrscheinlichkeiten sind allerdings stets etwas zu niedrig, da Chakteristiken auch die Differenzen der Zwischenrunden vorschreiben, obwohl diese unwichtig für den Angriff sind. Lai führt daher Differentiale ein, die nur Ein- und Ausgangsdifferenzen vorschreiben. Die Wahrscheinlichkeiten dieser sind zwar schwerer zu bestimmen, geben dafür aber den genauen Wert für den Aufwand einer diff. Kryptoanalyse an. Bestimmt man also die Übergangsmatrix eines Kryptosystems, so kann man aus dem maximalen Element ersehen, wie resistent dieses System gegenüber differentieller Kryptoanalyse ist.

Doch zunächst führe ich die Charakteristiken ein:

Definition 4.8 *Eine* **n-Runden Charakteristik** *ist ein Tupel $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ mit Ω_P und Ω_T m-Bit Zahlen und Ω_Λ eine Liste von n Elementen $\Omega_\Lambda = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$. Die Λ_i sind Paare der Form $\Lambda_i = (\lambda_I^i, \lambda_O^i)$, wobei λ_I^i, λ_O^i $^m/_2$ Bit Zahlen und m die Blockgröße des Kryptosystems ist. Außerdem muß erfüllt sein:*

$$\begin{aligned} \lambda_I^1 &= \Omega_{PR} \\ \lambda_I^2 &= \Omega_{PL} \oplus \lambda_O^1 \\ \lambda_I^n &= \Omega_{TR} \\ \lambda_I^{n-1} &= \Omega_{TL} \oplus \lambda_O^n \end{aligned}$$

und für jedes i mit $2 \leq i \leq n-1$

$$\lambda_O^i = \lambda_I^{i-1} \oplus \lambda_I^{i+1}$$

Definition 4.9 *Ein* **richtiges Paar** *in Bezug auf eine n-Runden Charakteristik $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$ und einen unabhängigen Schlüssel K ist ein Klartextpaar mit $P' = \Omega_P$ und für jede Runde i der ersten n Runden der Verschlüsselung des Paares unter dem Schlüssel K gleicht die Inputdifferenz der i-ten Runde λ_I^i und die Outputdifferenz der F-Funktion λ_O^i. Jedes Paar, welches die Bedingung nicht erfüllt, ist ein* **falsches Paar** *in Bezug zur Charakteristik und dem Schlüssel.*

Bemerkung: Eine n-Runden Charakteristik $\Omega^1 = (\Omega_P^1, \Omega_\Lambda^1, \Omega_T^1)$ und eine m-Runden Charakteristik $\Omega^2 = (\Omega_P^2, \Omega_\Lambda^2, \Omega_T^2)$ kann zu einer $n+m$-Runden Charakteristik $\Omega = (\Omega_P^1, \Omega_\Lambda, \Omega_T^2)$ zusammengefügt werden, wenn $\Omega_T^1 = V(\Omega_P^2)$ (V vertauscht die beiden Hälften eines Blocks). Ω_Λ ist dann die aneindergekettete Liste $(\Omega_\Lambda^1, \Omega_\Lambda^2)$.

Diese Definition einer Charakteristik bezieht sich auf Feistel-Chiffren; für Systeme mit anderen Rundenfunktionen muß die Definition entsprechend umformuliert werden. Für die beiden in diesem Kapitel behandelten Chiffrierer DES und FEAL gelten diese Definitionen jedoch. □

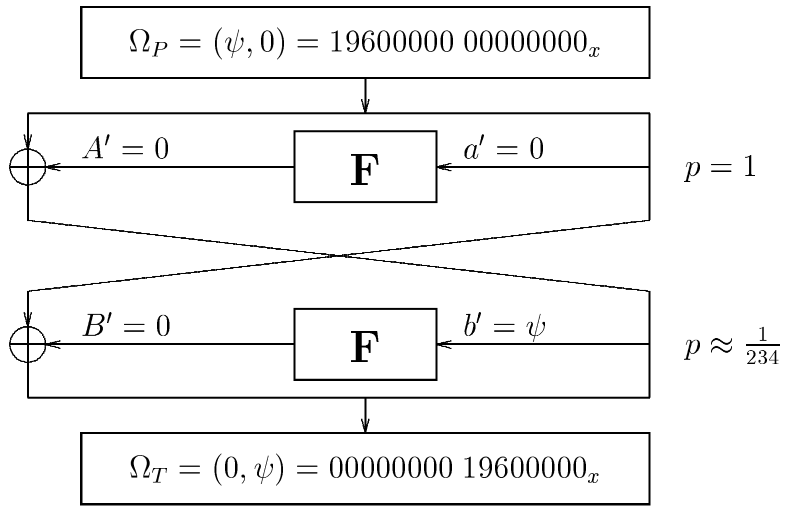

Abbildung 4.2: Die iterative 2-Runden Charakteristik des DES

Definition 4.10 *Runde i einer Charakteristik Ω hat die Wahrscheinlichkeit* $\mathbf{p}_i^\Omega$, *wenn* $\lambda_I^i \to \lambda_O^i$ *mit Wahrscheinlichkeit* p_i^Ω *durch die F-Funktion.*

Definition 4.11 *Die n-Runden Charakteristik Ω hat Wahrscheinlichkeit* $\mathbf{p}^\Omega$, *wenn* p^Ω *das Produkt der Rundenwahrscheinlichkeiten ist:*

$$p^\Omega = \prod_{i=1}^{n} p_i^\Omega$$

Satz 4.4 ([BiSh93]) *Die formal definierten Wahrscheinlichkeiten der beiden vorangegangenen Definitionen ist die tatsächliche Wahrscheinlichkeit einer Charakteristik* $\Omega = (\Omega_P, \Omega_\Lambda, \Omega_T)$, *daß ein fest gewähltes Klartextpaar mit* $P' = \Omega_P$ *ein richtiges Paar bezüglich eines zufällig gewählten unabhängigen Schlüssels ist.*

Beweis: Der Satz folgt aus der Tatsache, daß die Rundenwahrscheinlichkeit unabhängig von dem exakten Input ist (Lemma 4.2) und aus der Annahme, daß die Teilschlüssel unabhängig sind, wodurch die Eingaben der S-Boxen in den einzelnen Runden vollkommen zufällig werden. Aufgrund dieser stochastischen Unabhängigkeit multiplizieren sich die Rundenwahrscheinlichkeiten. □

Definition 4.12 *Eine* **iterative Charakteristik** *ist eine Charakteristik, die mit sich selbst verknüpft werden kann.*

Anmerkung: Diese Charakteristiken haben sich als die nützlichsten Charakteristiken herausgestellt, da man sie beliebig oft aneinanderketten kann und so Charakteristiken für beliebig viele Runden auf einfachste Art und Weise erhält. Die von Biham und Shamir benutzte Charakteristik zum Brechen des DES ist eine iterative 2-Runden Charakteristik, die eine Inputdifferenz ungleich Null zu einer Outputdifferenz gleich Null werden läßt. Dazu müssen beim DES drei benachbarte S-Boxen einbezogen werden. Diese Charakteristik besitzt eine Wahrscheinlichkeit von ungefähr $\frac{1}{234}$ (siehe Abbildung 4.2). □

Die Suche nach geeigneten Charakteristiken muß auf heuristischem Wege bestritten werden, indem man z.B. Computerprogramme nach bestimmten Strukturen suchen läßt. Durch iterative Charakteristiken wird die Suche für einen Angriff auf Systeme mit hoher

Rundenanzahl jedoch sehr erleichtert. Um aber den minimalen Aufwand eines Angriffs für einen Chiffrierer angeben zu können, muß die Übergangsmatrix für das System bestimmt werden. Im folgenden werde ich die nötige Theorie dazu entwickeln.

Definition 4.13 *Ein* **n-Runden Differential** *ist ein Tupel* (α, β)*, wobei* α *eine mögliche Inputdifferenz bezeichne, und* β *eine mögliche Outputdifferenz nach n Runden. Die Wahrscheinlichkeit eines n-Runden Differentials ist die bedingte Wahrscheinlichkeit, daß die Outputdifferenz* Y'_n*, beträgt, wenn die Inputdifferenz* X' *betrug, unter der Voraussetzung, daß der Input X und die Teilschlüssel jeder Runde gleichverteilt zufällig und stochastisch unabhängig ausgewählt wurden. Diese Wahrscheinlichkeit wird mit* $P(Y'_n = \beta | X' = \alpha)$ *bezeichnet.*

Bemerkung: Handelt es sich um einen Markovchiffrierer, so ist die Wahrscheinlichkeit gegeben durch den Eintrag (α, β) in der Übergangsmatrix Π^n. □

Anmerkung: Ich bin bisher davon ausgegangen, daß der Schlüssel zufällig und unabhängig gewählt wurde und der Klartext fest. Tatsächlich ist bei einem Angriff aber der Schlüssel fest (der wird ja gesucht) und die Klartexte sind zufällig gewählt. Damit die differentielle Kryptoanalyse also überhaupt funktionieren kann, muß ich folgende Hypothese aufstellen:

Hypothese der stochastischen Äquivalenz: [Lai92]

Für praktisch alle n-Runden Differentiale (α, β) gilt

$$P(Y'_n = \beta | X' = \alpha) \approx P(Y'_n = \beta | X' = \alpha, K_1 = k_1, \ldots, K_n = k_n) \tag{4.2}$$

für fast alle unabhängigen Schlüssel $(k_1, \ldots, k_n)$.

Ein n-Runden Differential (α, β) heißt *DK-nützlich*, wenn (4.2) gilt. □

Man kann leicht eine obere Schranke für den maximal möglichen Aufwand einer diff. Kryptoanalyse bestimmen, und zwar durch die Blockgröße m: Benötigt die diff. Kryptoanalyse mehr als 2^m Paare, so ist ein Angriff nicht durchführbar, da es nicht genug verschiedene Paare gibt. Damit läßt sich folgendes Theorem aufstellen:

Theorem 4.5 ([Lai92]) *Trifft die Hypothese der stochastischen Äquivalenz zu, dann ist ein n-Runden Kryptosystem genau dann immun gegenüber differentieller Kryptoanalyse, wenn es kein DK-nützliches Differential* (α, β) *gibt mit*

$$P(Y'_{n-1} = \beta | X' = \alpha) \gg \frac{1}{2^m - 1},$$

wobei m die Blockgröße bezeichnet.

Bemerkung: Biham und Shamir formulieren diese Hypothese nicht explizit aus, sondern weisen nur darauf hin, daß die Wahrscheinlichkeiten nicht immer für alle Schlüssel gelten. Sie nehmen jedoch an, daß die Wahrscheinlichkeiten ihrer Charakteristiken die tatsächlichen gut approximieren, wenn zufällige Schlüssel gewählt werden. □

Bemerkung: Die diff. Kryptoanalyse beruht also auf zwei wichtigen Sätzen: Zuerst auf Satz 4.1, der aus dem Chiffrierer mit einer geeigneten Differenz einen Markovchiffrierer macht und die Unabhängigkeit der berechneten Wahrscheinlichkeiten vom exakten Input gewährleistet; zum anderen auf der Hypothese der stochastischen Äquivalenz, die die Wahrscheinlichkeiten auch bei festgehaltenem Schlüssel erhält. □

Es fehlt nun nur noch eine Aussage über den Gesamtaufwand einer diff. Kryptoanalyse. Es gibt stets zwei Faktoren: Die Datenkomplexität und die Berechnungskomplexität. Die Datenkomplexität ist bei der diff. Kryptoanalyse gegeben durch die Anzahl an Paaren, die benötigt werden, so daß der richtige Schlüssel signifikant öfter gezählt wird als der Durchschnitt an falschen Schlüsseln. Diese gibt Lai an als

$$C_d(n) \geq 2 / \left(P(Y'_{n-1} = \beta | X' = \alpha) - \frac{1}{2^m - 1} \right). \tag{4.3}$$

Biham und Shamir führen zu diesem Zweck den Begriff des Rauschabstands ein, der ihnen eine genauere Angabe über die Anzahl der benötigten Paare gibt. Der Rauschabstand ist das Verhältnis der Anzahl an richtigen Paaren zur durchschnittlichen Anzahl an falschen Schlüsseln. Beide Ansätze sind notwendig, da die Differentiale oder Charakteristiken kürzer als das Kryptosystem sind, so daß richtige von falschen Paaren nicht ohne weiteres getrennt werden können. Es ist lediglich bekannt, daß mit der Wahrscheinlichkeit des Differentials die Outputdifferenz der vorletzten Runde den erwünschten Wert liefert. Biham und Shamir geben nun Mechanismen an, wie einige falsche Paare schon möglichst früh aussortiert werden können, indem man z.B. von der Charakteristik nicht berührte S-Boxen zu Rate zieht oder ungültige Input–Outputdifferenzen ermittelt.

Die Berechnungskomplexität ist derjenige Aufwand, der nötig ist, um aus den analysierten Paaren die möglichen Schlüssel für die letzte Runde abzuleiten. Dieser Aufwand ist normalerweise sehr gering, da die Rundenfunktion kryptographisch schwach ist. Es ist daher sehr einfach, die möglichen Teilschlüssel für jedes Paar zu ermitteln. Die Berechnungskomplexität kann deshalb vernachlässigt werden, der wesentliche Anteil bei einer diff. Kryptoanalyse liegt in der Datenkomplexität, da die Anzahl an Paaren die Anzahl der nötigen Verschlüsselungsoperationen angibt.

Der benötigte Speicherplatz ist bei dieser Grundform der diff. Kryptoanalyse allerdings ein weiterer wichtiger Faktor. Um allen möglichen Teilschlüsseln Wahrscheinlichkeiten zuordnen zu können, muß für jeden Schlüssel ein Zähler geführt werden. Bei einem 48-Bit Teilschlüssel müßten also 2^{48} Zähler geführt werden, was praktisch nicht möglich ist. Deswegen müssen zur praktischen Durchführung einer diff. Kryptoanalyse noch andere Faktoren berücksichtigt werden. Die einfachste Möglichkeit ist einfach die Ignorierung einzelner Schlüsselbits, es wird also nur ein Teil des Schlüssels gezählt und der Rest wird entweder mit einer zweiten Chrakteristik oder durch eine vollständige Suche ermittelt. [BiSh93] gibt für einzelne Angriffe noch weitere Optimierungsmöglichkeiten an, die aber nur begrenzt verallgemeinbar sind.

4.3 Kryptoanalyse des 3-Runden DES

Als Beispiel einer ohne Probleme nachzuvollziehenden diff. Kryptoanalyse habe ich einen Angriff auf einen 3-Runden DES gewählt. Zunächst die Methode:

Sei $X = (L_0, R_0), X^* = (L_0^*, R_0^*)$ ein Klartextpaar, $T = (L_3, R_3), T^* = (L_3^*, R_3^*)$ das daraus resultierende Chiffretextpaar. Dann kann R_3 dargestellt werden durch

$$\begin{aligned} R_3 &= L_2 \oplus F(R_2, K_3) \\ &= R_1 \oplus F(R_2, K_3) \\ &= L_0 \oplus F(R_0, K_1) \oplus F(R_2, K_3) \end{aligned}$$

R_3^* kann analog dargestellt werden, und damit ergibt sich R'_3 zu

$$R'_3 = L'_0 \oplus F(R_0, K_1) \oplus F(R_0^*, K_1) \oplus F(R_2, K_3) \oplus F(R_2^*, K_3).$$

Ich wähle nun die triviale 1-Runden Chrakteristik mit Wahrscheinlichkeit 1, es ist also $R_0 = R_0^*$, der linke Block kann beliebig gewählt werden. Damit vereinfacht sich R'_3 auf

$$R'_3 = L'_0 \oplus F(R_2, K_3) \oplus F(R_2^*, K_3).$$

R'_3 kann leicht aus dem Chiffretext errechnet werden, L'_0 aus dem Klartext. Außerdem weiß ich, daß $F(R_2, K_3) = P(S_{OC})$ und $F(R^*_2, K_3) = P(S^*_{OC})$, also

$$P(S_{OC}) \oplus P(S^*_{OC}) = R'_3 \oplus L'_0$$

und damit kenne ich die Outputdifferenzen der S-Boxen in Runde 3 durch

$$S'_{OC} = P^{-1}(R'_3 \oplus L'_0). \tag{4.4}$$

Weiter ist $R_2 = L_3$ und $R^*_2 = L^*_3$ Teil des Chiffretextes, damit kenne ich auch die Inputdifferenzen der S-Boxen in Runde 3 per

$$S_{EC} = E(L_3), \quad S^*_{EC} = E(L^*_3). \tag{4.5}$$

Ich kann nun wie in Beispiel 2.4 für jede S-Box aus der Kenntnis der Inputdifferenz $S'_I = S'_E = S_E \oplus S^*_E$ die möglichen Inputs S_I bestimmen, dazu benötige ich die entsprechenden Tabellen, die leicht erzeugt werden können, sowie $8 \cdot 64$ Zähler, die die entsprechenden Vorschläge von jedem Testpaar für alle möglichen Teilschlüssel Si_K zählen. Damit erhalte ich den kompletten 48-Bit Schlüssel der dritten Runde, die restlichen 8 Bit des Schlüssels kann man schnell durch eine vollständige Schlüsselsuche finden.

Um das Verfahren zu illustrieren, führe ich das Verfahren an einem Beispiel durch (aus [Stin95]):

Beispiel 4.5: Es seien die drei Klartextpaare und ihre zugehörigen Chiffretextpaare in Tabelle 4.4 gegeben, wobei diese Paare unter demselben (unbekannten) Schlüssel erzeugt worden seien.

Zuerst berechne ich für das erste Paar die S-Box Inputs der Runde 3 mit Formel (2):

$$S_{EC} = 00000000\ 01111110\ 00001110\ 10000000\ 01101000\ 00001100$$

$$S^*_{EC} = 10111111\ 00000010\ 10101100\ 00000101\ 01000000\ 01010010.$$

Mit Formel (1) ergibt sich die Outputdifferenz der S-Boxen:

$$S'_{OC} = 10010110\ 01011101\ 01011011\ 01100111.$$

Das Ganze nochmal für das zweite Paar:

$$S_{EC} = 10100000\ 10111111\ 11110100\ 00010101\ 00000010\ 11110110$$

$$S^*_{EC} = 10001010\ 01101010\ 01011110\ 10111111\ 00101000\ 10101010$$

$$S'_{OC} = 10011100\ 10011100\ 00011111\ 01010110.$$

Und schließlich für das dritte Paar:

$$S_{EC} = 11101111\ 00010101\ 00000110\ 10001111\ 01101001\ 01011111$$

$$S^*_{EC} = 00000101\ 11101001\ 10100010\ 10111111\ 01010110\ 00000100$$

$$S'_{OC} = 11010101\ 01110101\ 11011011\ 00101011.$$

Als nächstes muß ich die möglichen Werte für S_{KC} für alle S-Boxen und Paare errechnen und die Ergebnisse in den Zählerfeldern abspeichern. Ich demonstriere die Prozedur anhand des Zählerfelds J_1 für die erste S-Box mit dem ersten Paar. In diesem Paar ist $S1'_E = 101111$ und $S1'_O = 1001$. Mit dem Wert $S1_E = 000000$ ergeben sich folgende Werte für $S1_K$:

$$\{000000,\ 000111,\ 101000,\ 101111\}.$$

Fasse ich die 6-Bit Schlüssel als natürliche Zahlen auf, so muß ich die Zähler 0, 7, 40, 47 im Feld J_1 inkrementieren.

Klartext	Chiffretext
748502CD38451097	03C70306D8A09F10
3874756438451097	78560A0960E6D4CB
486911026ACDFF31	45FA285BE5ADC730
375BD31F6ACDFF31	134F7915AC253457
357418DA013FEC86	D8A31B2F28BBC5CF
12549847013FEC86	0F317AC2B23CB944

Tabelle 4.4: In- und Outputpaare für Beispiel 4.5

J_1															
1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
J_2															
0	0	0	1	0	3	0	0	1	0	0	1	0	0	0	0
0	1	0	0	0	2	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0	1	0	2	0	0	0
J_3															
0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	1
0	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
J_4															
3	1	0	0	0	0	0	0	0	0	2	2	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	1
1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
J_5															
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	2	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	1	0	0	0	0	2	0
J_6															
1	0	0	1	1	0	0	3	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0
J_7															
0	0	2	1	0	1	0	3	0	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	2	0	0	0	2	0	0	0	0	1	2	1	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
J_8															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1
0	3	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Tabelle 4.5: Zählerfelder für Beispiel 4.5

In Tabelle 4.5 habe ich den Stand am Ende der Auswertung angegeben. Wie man sieht, zeigen alle Felder genau einen Zähler mit dem Wert 3 an, d.h. dieser Schlüssel wurde als einziger von allen drei Paaren als möglicher Schlüssel vorgeschlagen. Damit erhalte ich die einzelnen Schlüsselbits aller S-Boxen:

$$\begin{array}{llll} S1_{KC} = 101111 & S2_{KC} = 000101 & S3_{KC} = 010011 & S4_{KC} = 000000 \\ S5_{KC} = 011000 & S6_{KC} = 000111 & S7_{KC} = 000111 & S8_{KC} = 110001. \end{array}$$

Indem ich einen Blick auf den Schlüsselerzeugungsalgorithmus werfe, kann ich zumindest 48 Bits des Gesamtschlüssels rekonstruieren:

$$K = 0001101\ 0110001\ 01?01?0\ 1?00100\ 0101001\ 0000??0\ 111?11?\ ?100011,$$

wobei die Paritätsbits weggelassen wurden und ? noch unbekannte Schlüsselbits kennzeichnet. Durch vollständige Schlüsselsuche kann ich auch den Rest des Schlüssels rekonstruieren, er lautet (mit Paritätsbits):

$$K = 1A624C89520DEC46_x.$$

□

4.4 Kryptoanalyse des 6-Runden DES

Der 6-Runden DES ist nicht mehr ganz so einfach zu analysieren wie der 3-Runden DES. Für den Angriff verwenden Biham und Shamir zwei 3-Runden Charakteristiken mit Wahrscheinlichkeit $\frac{1}{16}$ und wählen am Schluß denjenigen Schlüssel, der am häufigsten gezählt wurde. Jede der Charakteristiken erlaubt es, 30 Schlüsselbits zu untersuchen, beide Charakteristiken zusammen bringen die Anzahl auf 42 Bits. Die restlichen 14 Schlüsselbits können entweder durch eine vollständige Suche oder durch weitere Analysen gefunden werden.

Die erste Charakteristik Ω^1 (Abbildung 4.3) erzeugt in der anschließenden vierten Runde in den S-Boxen $S2, S5, \ldots, S8$ eine Null als Inputdifferenz, so daß die Outputdifferenz ebenfalls Null sein muß. Verfolgt man die Differenzen weiter, so erhält man für die Outputdifferenz der sechsten Runde F' die Beziehung $F' = c' \oplus D' \oplus T_L'$. Der Wert von c' wird durch die Charakteristik vorgegeben und durch die Inputdifferenz Null sind 20 Bits von D' Null. Dadurch kann ich die Outputdifferenz der fünf S-Boxen in Runde sechs aus der Charakteristik und dem Chiffretext rekonstruieren. Die Inputdifferenz f' erhält man wie üblich aus der Differenz T_R' der rechten Teilblöcke der Chiffretexte.

Im Gegensatz zur 3-Runden Analyse erhalte ich nur mit einer Wahrscheinlichkeit von $\frac{1}{16}$ die nötige Outputdifferenz nach der dritten Runde, um die Analyse zu starten. Deswegen kann ich die richtigen Paare nicht von den falschen Paaren trennen, und im Laufe der Analyse werden auch falsche Schlüssel vorgeschlagen. Um zu verhindern, daß am Ende ein falscher Schlüssel vorgeschlagen wird, muß ich die nötige Anzahl an Paaren so wählen, daß sich der richtige Schlüssel deutlich abhebt. Dazu haben Biham und Shamir den Rauschabstand eingeführt, der es ihnen ermöglicht, die Anzahl an Paaren zu bestimmen, so daß genügend richtige Paare vorliegen. Zur Erinnerung: der Rauschabstand ist das Verhältnis von der Anzahl an richtigen Paaren zu der durchschnittlichen Häufigkeit von falschen Schlüsseln, also dem „Rauschen". Um dieses Verhältnis zu verbessern, wird nicht jede S-Box einzeln betrachtet, sondern alle 30 zu ermittelnden Schlüsselbits simultan gezählt. Dies benötigt allerdings eine Menge von 2^{30} Zählern, was nicht mehr so ohne weiteres möglich ist. Betrachten wir 30 Bits simultan, so ist der Rauschabstand dieser Analyse so groß, daß Biham und Shamir 7–8 richtige Paare für ausreichend halten, d.h. aufgrund der Wahrscheinlichkeit der Charakteristik werden $2^4 \cdot 2^3 = 2^7 = 128$ Paare benötigt.

Um den Speicherbedarf zu reduzieren, setzen Biham und Shamir einen Trick ein, der immer dann funktioniert, wenn sehr wenige Paare analysiert werden. Bei dieser Methode

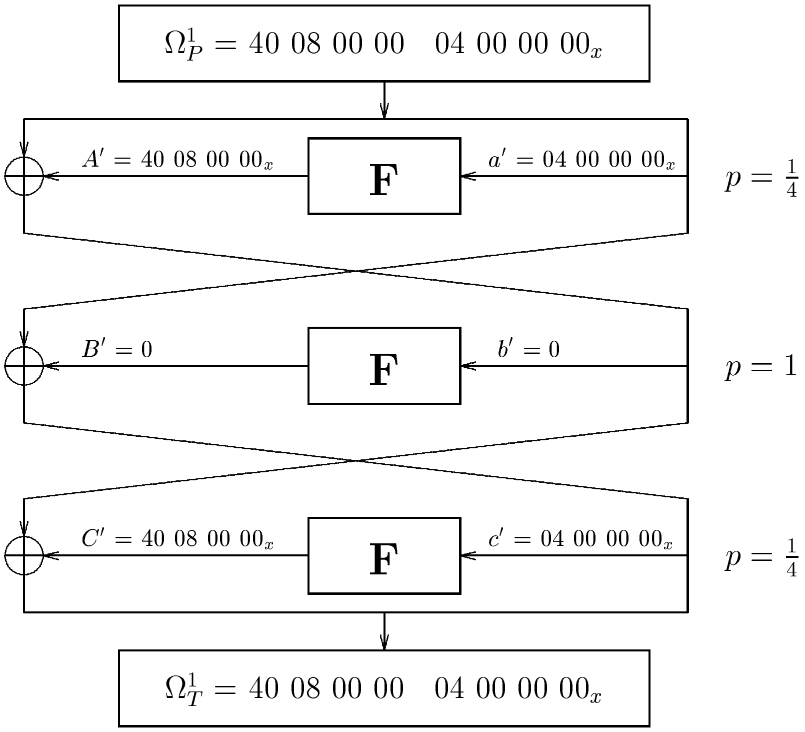

Abbildung 4.3: 6-Runden DES Analyse: Charakteristik Ω^1

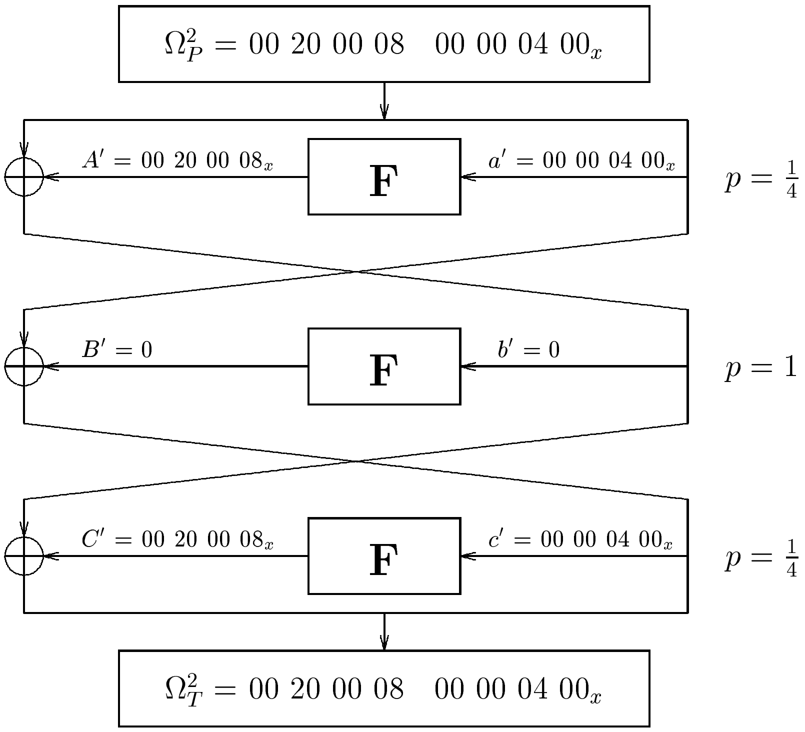

Abbildung 4.4: 6-Runden DES Analyse: Charakteristik Ω^2

INPUT: Anzahl k der betrachteten S-Boxen, Anzahl n Paare
OUTPUT $k \cdot n$ Schlüsselbits.

1. (Initialisierung) Führe für jedes der n Paare k 64-Bit Bitmasken, wobei jedes Bit mit einem der möglichen Schlüssel Si_K korrespondiert.
2. (Kryptoanalyse) Sind die Werte Si_E, Si_E^*, Si_O' gegeben, berechne daraus die möglichen Schlüssel Si_K und setze die entsprechenden Bits in der entsprechenden Maske für jede S-Box und jedes Paar.
3. (Erzeugung der Cliques) *Eine Clique ist eine Menge an Paaren, so daß in jeder der k Bitmasken genau ein gemeinsames Bit in allen Paaren der Menge gesetzt ist.* Für jedes Paar vergleiche die Bitmasken dieses Paares mit denjenigen aller folgenden Paare. Solange in allen Bitmasken noch gemeinsame Bits gesetzt sind, füge weitere Paare durch Vergleich hinzu, bis in jeder Bitmaske genau ein Bit gesetzt ist. Speichere diese Clique mit seinem entsprechenden Schlüsselwert und der Anzahl an Paaren in dieser Clique. Führe das Verfahren für alle weiteren Möglichkeiten hindurch, bis alle Paare durchprobiert wurden.
4. (Ergebnis) Wähle die (hoffentlich) eindeutige Clique mit maximaler Größe und gebe den entsprechenden Schlüsselwert zurück.

Tabelle 4.6: **Algorithmus:** Cliquemethode

werden die Paare und die vorgeschlagenen Schlüssel als Graph betrachtet. Die Paare bilden die Ecken des Graphen, und wird ein Schlüssel von zwei Paaren vorgeschlagen, so werden diese beiden Ecken durch eine Kante verbunden und die Kante mit dem Schlüssel beschriftet. So bildet jeder vorkommende Schlüssel eine Menge mit den ihn vorschlagenden Paaren. Die größte dieser Mengen („cliques“) kennzeichnet dann den gesuchten Schlüssel. Dieses Verfahren wird nun wie in Algorithmus 4.6 beschrieben durchgeführt. Werden 240 Chiffretexte analysiert, so benötigt die Durchführung des Algorithmusses keine Zehntelsekunde.

Derselbe Algorithmus kann auch für die zweite Charakteristik Ω^2 (Abbildung 4.4) mit Wahrscheinlichkeit $\frac{1}{16}$ verwendet werden, welches mich die 30 Schlüsselbits von $S1, S2, S4, S5, S6$ finden läßt, da die Inputdifferenz dieser S-Boxen in der vierten Runde gleich Null ist. Die errechneten Schlüsselbits von $S2, S5, S6$ sollten in beiden Charakteristiken übereinstimmen, wenn der gefundene Schlüssel richtig sein soll. Ist dies geglückt, so kenne ich den Wert von 42 Schlüsselbits von K6. Die restlichen Schlüsselbits können am einfachsten durch eine vollständige Schlüsselsuche gefunden werden, dies wird im Mittel ca. 8000 Verschlüsselungen erfordern.

4.5 Aufwandsabschätzung für DES und FEAL-N

Biham und Shamir haben in ihrem Buch [BiSh93] ihr Verfahren auf eine Vielzahl von Chiffrierern angewendet, darunter auch für verschiedene DES Varianten und den FEAL mit unterschiedlichen Rundenzahlen. Ich werde hier kurz auf die von ihnen erzielten Ergebnisse eingehen.

n-Runden DES

Der Angriff auf den DES mit 10 oder mehr Runden basiert auf der iterativen Charakteristik aus Abbildung 4.2. Diese wird, um eine maximale Wahrscheinlichkeit zu garantieren, stets in eine Charakteristik mit ungerader Rundenanzahl verwandelt, indem am Ende noch einmal die erste Hälfte der Charakteristik mit der Inputdifferenz gleich Null angefügt wird.

Anzahl Runden	Benötigte Paare	Analysierte Paare	Gefundene Bits	Charakteristik Runden	p^Ω
10	2^{34}	4	18	9	$2^{-31.5}$
11	2^{35}	2^{11}	48	9	$2^{-31.5}$
12	2^{42}	4	18	11	$2^{-39.4}$
13	2^{43}	2^{19}	48	11	$2^{-39.4}$
14	2^{50}	4	18	13	$2^{-47.2}$
15	2^{51}	2^{27}	48	13	$2^{-47.2}$
16	2^{57}	32	18	15	$2^{-55.1}$

Tabelle 4.7: Aufwand der diff. Analyse beim DES

Anzahl Runden	Benötigte Paare	Analysierte Paare	Berechnungs-komplexität
10	2^{24}	2^{14}	2^{15}
11	2^{31}	2	2^{32}
12	2^{31}	2^{21}	2^{21}
13	2^{39}	2	2^{32}
14	2^{39}	2^{29}	2^{29}
15	2^{47}	128	2^{37}
16	2^{47}	2^{36}	2^{37}

Tabelle 4.8: Aufwand der verbesserten diff. Analyse des DES

Anschließend kann bei ungerader Rundenzahl des DES mit der um zwei Runden kürzeren Charakteristik eine sogenannte 2R-Attacke gestartet werden, die vom Prinzip gleich dem Angriff auf den 3-Runden DES ist. Sie erlaubt es, alle 48 Bits des letzten Teilschlüssels zu ermitteln, wobei eventuell sogar jede S-Box einzeln betrachtet werden kann, was eine enorme Einsparung an Speicherplatz für die Zähler bedeutet. Bei einem DES mit gerader Rundenanzahl ist eine 1R-Attacke mit der um 1 Runde kürzeren Charakteristik am sinnvollsten. Es können allerdings immer nur die Schlüsselbits ermittelt werden, dessen Inputdifferenz in der letzten Runde ungleich Null ist. Die iterative Charakteristik hat in den S-Boxen S1 bis S3 Differenzen ungleich Null, wodurch sich 18 Schlüsselbits ergeben, die berechnet werden können. Tabelle 4.7 gibt eine Übersicht über den Aufwand der diff. Kryptoanalyse bei einem DES mit verschiedener Rundenzahl.

Biham und Shamir haben diese Ergebnisse beim 16-Runden DES wesentlich verbessert, indem sie die 13-Runden Charakteristik in Runde 2–15 einsetzen und anschließend eine modifizierte 2R-Attacke starten. Anstatt Zähler für alle möglichen Schlüssel, die jedes Paar vorschlägt, zu führen, wird an jedem Paar noch härter gearbeitet, um von jedem Paar einen vollständigen 56-Bit Schlüssel vorschlagen zu lassen, der sofort getestet wird. Dieser Angriff ist im Gegensatz zur Grundform nicht mehr einsetzbar, wenn der DES mit unabhängigen Teilschlüsseln eingesetzt wird. Tabelle 4.8 gibt den Aufwand für diesen modifizierten Angriff für verschiedene Rundenzahlen an. Da der Angriff neben der benötigten Anzahl an Paaren zum Testen der Schlüssel weitere Verschlüsselungsoperationen benötigt, ist jetzt auch die Berechnungskomplexität nicht mehr unwesentlich und muß ebenfalls in Betracht gezogen werden.

In ihrem Buch analysieren Biham und Shamir [BiSh93] auch modifizierte DES Varianten mit veränderten S-Boxen und Permutationen. Bemerkenswert ist dabei die Tatsache, daß der DES nur in seiner bekannten Form eine maximale Sicherheit gegen einen differentiellen Angriff bietet, schon die grundlegende Form des Angriffs reduzierte den Aufwand auf 2^{20} bis 2^{47} Paare. Matsui hat in [Mat94b] gezeigt, daß die iterative Charakteristik 4.2

der beste Angriffspunkt für den DES mit 10 und mehr Runden ist.

FEAL-N

Als Reaktion auf die diff. Kryptoanalyse des FEAL-8 erweiterten die Entwickler den FEAL auf FEAL-N mit N Runden. Der FEAL-8 ist unter Benutzung zweier Charakteristiken mit Wahrscheinlichkeit $\frac{1}{16}$ mit gerade einmal 128 Chiffretexten brechbar. Der Angriff auf den FEAL-N beruht auf der folgenden iterativen 4-Runden Charakteristik (Abbildung 4.5) mit Wahrscheinlichkeit $^1/_4$ pro Runde. Danach wird eine 2R-Attacke gestartet, die nicht direkt den letzten Teilschlüssel ermittelt, sondern vielmehr sogenannte „actual subkeys", die einer äquivalenten Beschreibung des FEAL entstammen und geeigneter für die diff. Kryptoanalyse sind. Danach wird das System Runde für Runde rückwärts aufgerollt, bis alle AKi berechnet worden sind. Aus der Kenntnis der AKi kann dann unter Verwendung des Schlüsselerzeugungsalgorithmus der tatsächliche Schlüssel berechnet werden.

Tabelle 4.9 gibt schließlich eine Übersicht über den Aufwand der diff. Kryptoanalyse beim FEAL-N für $N = 8, 12, 16, 20, 24, 28$ an. Als Vergleich ist für $N = 8$ der Aufwand des speziellen Angriffs für dieses System zusätzlich aufgeführt. Die diff. Kryptoanalyse lohnt sich nur für Rundenzahlen bis 30, darüberhinaus benötigt man mehr Chiffretexte als verfügbar sind (2^{64}). Die Spalte „wählbare Klartexte" ist deswegen aufgeführt, da für den Angriff vier Charakteristiken benötigt werden. Die entsprechenden Paare können zu je acht Stück zusammengefaßt werden, die dann je zwei Paare für jede Charakteristik liefern, deswegen werden nur halbsoviele Klartexte wie Paare benötigt. Der Angriff auf den FEAL-8 verwendet zwei Charakteristiken, deren Paare in Quartetten angeordnet werden können, so daß 4 Klartexte 2 Paare für jede Charakteristik liefern.

4.6 Zusammenfassung

Die diff. Kryptoanalyse ist ein Angriff mit frei wählbarem Klartext. Dies macht den Angriff vielleicht nicht praktikabel, da ja Zugang zum Chiffrierer mit dem aktuellen Schlüssel gegeben sein muß, dennoch ist dieses Verfahren das erste wirklich mächtige Verfahren, um iterierte Kryptosysteme wie Feistelchiffren zu brechen. Der Ansatz ist die Beobachtung, daß die Rundenfunktion eines Chiffrierer kryptographisch schwach ist, und es daher ohne allzu großen Aufwand aus der Kenntnis der Inputs möglich ist, alle möglichen Teilschlüssel zu errechnen, die den beobachteten Output liefern.

Für die Analyse betrachtet man „Differenzen" von Klartexten, wobei die Differenz eine geeignete Gruppenoperation ist, so daß der Chiffrierer zu einem Markovchiffrierer (Definition 4.3) wird. Diese Differenz verändert sich aufgrund der (bzgl. der Differenz) nichtlinearen Einflüsse des Chiffrierers von Runde zu Runde, wobei die Folge der Differenzen eine homogene Markovkette beschreibt. Die diff. Kryptoanalyse versucht nun, aus der Differenz der Chiffretexte und einer vorgegebenen Outputdifferenz der vorletzten Runde den Teilschlüssel der letzten Runde zu berechnen. Die Outputdifferenz der vorletzten Runde wird durch ein Differential [Lai92] oder eine Charakteristik [BiSh93] mit einer gewissen Wahrscheinlichkeit vorgegeben. Diese Wahrscheinlichkeit kann zumindest näherungsweise durch die Übergangsmatrix der Markovkette für eine Runde errechnet werden, indem man die Wahrscheinlichkeiten für jede einzelne Runde multipliziert. Ist der letzte Teilschlüssel berechnet worden, so können entweder die restlichen Teilschlüssel ermittelt oder, falls möglich, durch vollständige Schlüsselsuche alle restlichen Möglichkeiten ausprobiert werden.

Die diff. Kryptoanalyse kann den FEAL bis zu einer Rundenanzahl von 30 schneller als eine vollständige Schlüsselsuche brechen, den DES in seiner grundlegenden Form nur bis zu 15 Runden. Der erweiterte Angriff kann den DES bis zu 19 Runden schneller als eine vollständige Suche brechen.

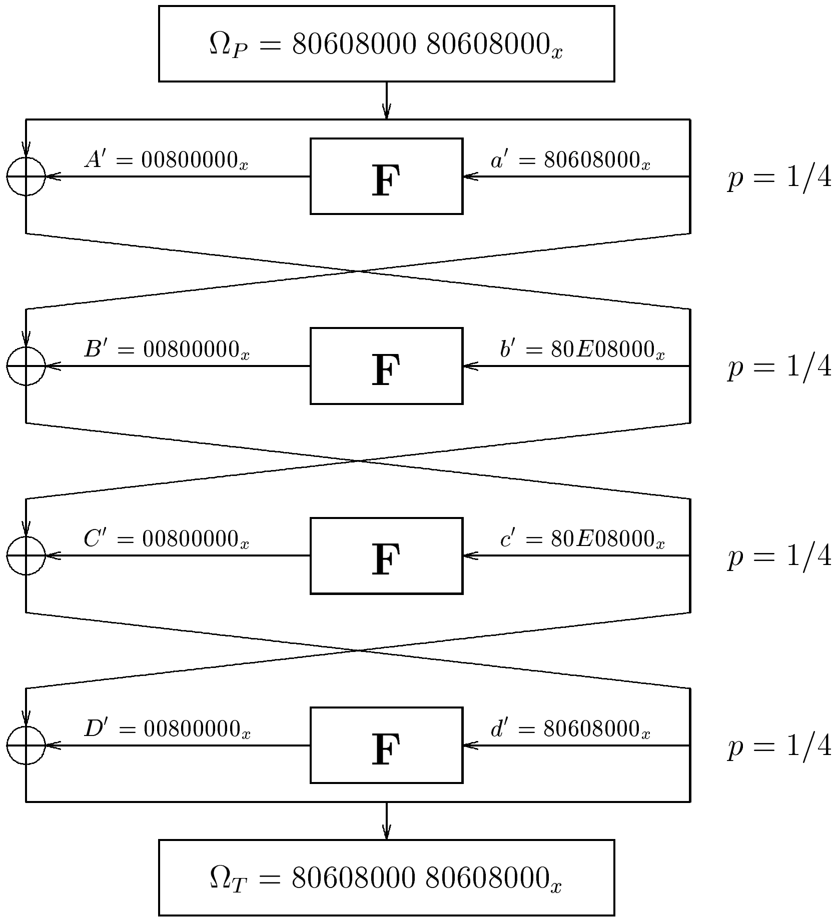

Abbildung 4.5: Die iterative Charakteristik des FEAL-N

Anzahl Runden	p^Ω	Benötigte Paare	wählbare Klartexte
8	2^{-4}	128	128
8	2^{-12}	2^{14}	2^{13}
12	2^{-20}	2^{22}	2^{21}
16	2^{-28}	2^{30}	2^{29}
20	2^{-36}	2^{38}	2^{37}
24	2^{-44}	2^{46}	2^{45}
28	2^{-52}	2^{54}	2^{53}
30	2^{-58}	2^{61}	2^{60}

Tabelle 4.9: Aufwand der diff. Analyse des FEAL-N

4.7 Kryptoanalyse mit gekürzten und höheren Differenzen

Lai hat als erster in [Lai94] das Konzept von Differenzen höherer Ordnung vorgestellt. Knudsen hat dieses Konzept erweitert und um gekürzte Differenzen ergänzt [Knud94]. Es gibt Beispiele für Chiffrierer [KnBe95, Knud94], die gegen normale diff. Kryptoanalyse immun sind, aber gegen dieses erweiterte Verfahren nicht bestehen können.

Gekürzte Differenzen

Bei der Verwendung von gekürzten Differenzen versucht man nur Teile der Differenzen eines iterierten Chiffrierers vorherzusagen. Eigentlich betrachtet man bei einem gekürzten Differential (a', b') eine Familie von Differentialen (a, b), wobei der entsprechende Teil von a und b den Werten von a' und b' gleicht. Bei differentiell-gleichförmigen Rundenfunktionen (siehe Abschnitt 6.3.1) ist die Wahrscheinlichkeit eines vollen Differentials über mehrere Runden sehr schnell sehr klein, so daß entweder mehr Paare benötigt werden als überhaupt verfügbar sind, oder der Rauschabstand wird so klein, daß der Angriff nicht mehr zuverlässig arbeitet. In diesem Fall kann es helfen, nur einen Teil der Differenz vorherzusagen. Knudsen führt in seiner Arbeit dazu ein einfaches Beispiel einer Rundenfunktion $F(x,k) = (x \oplus k)^{-1}$ einer Feistel-Chiffre an, die für ungerade n differentiell-2-uniform ist. Dieser Chiffrierer ist für mehr als 3 Runden sicher gegen diff. Kryptoanalyse.

Knudsen bricht diesen Chiffrierer mit relativ wenigen Paaren [Knud94], indem er nur ein Bit der Outputdifferenz nach drei Runden vorhersagt. Jedes Paar reduziert die Suche auf die Hälfte aller Möglichkeiten für die Teilschlüssel der Runden vier und fünf. In einem weiteren Experiment bricht er den 6-Runden DES mit gekürzten Differenzen, was eine Reduzierung der erforderlichen 128 Paare bei der normalen Analyse auf 46 Paare nach sich zieht.

Höhere Differenzen

Definition 4.14 ([Lai94]) *Seien $(S,+), (T,+)$ abelsche Gruppen. Für eine Funktion $f : S \to T$ ist die* **Ableitung von f im Punkte a** *gegeben durch*

$$\Delta_a f(x) := f(x+a) - f(x).$$

Die **i-te Ableitung von f im Punkt** $a_1, \ldots, a_i$ *ist definiert als*

$$\Delta^{(i)}_{a_1,\ldots,a_i} f(x) := \Delta_{a_i}(\Delta^{(i-1)}_{a_1,\ldots,a_{i-1}} f(x)).$$

Bemerkung: Die Definition der ersten Ableitung entspricht Biham's und Shamir's Definition von Differenz. Es erscheint so nur natürlich, die Definition von Differentialen auf diese Weise auch auf Differentiale höherer Ordnung auszudehnen. □

Definition 4.15 *Ein* **1-Runden Differential der Ordnung i** *ist ein $(i+1)$-Tupel $(a_1, \ldots, a_i, b)$, so daß $\Delta^{(i)}_{a_1,\ldots,a_i} f(x) = b$.*

Bemerkung: Lai hat in [Lai94] folgende Aussagen bewiesen:

- Damit die i-te Ableitung nicht trivial Null wird, müssen die Punkte $a_1, \ldots, a_i$ bei Funktionen über $\mathbb{Z}_2^n$ linear unabhängig sein.
- Jede Ableitung verkleinert die nichtlineare Ordnung von f (siehe Abschnitt 6.3.1) jeweils um eins.

□

Die Einführung höherer Differentiale ist besonders bei 5-Runden Feistelchiffrierern mit niedriger Ordnung und unabhängigen Teilschlüsseln interessant. Sofern die Ordnung r klein ist, kann unter Verwendung von Differentialen der Ordnung r ein einfacher Angriff gestartet werden, der 2^{r+1} wählbare Klartexte benötigt und ca. 2^{n+r} Verschlüsselungsoperationen, wenn n die Blockgröße des Chiffrierers ist. Dieser Angriff nutzt den Fakt aus, daß bei dieser Ordnung die r-te Ableitung konstant ist. Bei 64-Bit Blockgrößen ist dieser Angriff allerdings nicht mehr sehr praktisch.

Wie bei den gekürzten Differenzen gibt es allerdings noch ein ungelöstes Problem in Bezug der Ausweitung des Angriffs auf mehr Runden. Bislang konnte keiner der Angriffe auf mehr als 5 Runden ausgedehnt werden, ob eine Ausweitung überhaupt noch eine Verbesserung gegenüber dem normalen Angriff erreicht, ist ebenfalls nicht bekannt.

Kapitel 5

Lineare Kryptoanalyse

Der Vorläufer der linearen Kryptoanalyse wurde 1992 von Mitsuru Matsui in [MaYa92] vorgestellt. In diesem Experiment versuchte er, den FEAL mit einem bekannten Klartextangriff zu brechen, der lineare Approximationen des FEAL benutzt. Ein Jahr später stellte er in [Mat93] die weiterentwickelte Methode auch für den DES vor. In [Mat94] verbesserte er seinen Angriff auf den DES, der nur noch 2^{43} bekannte Klartexte benötigt und damit besser als eine differentielle Kryptoanalyse ist, zumal nur bekannte Klartexte benötigt werden, die leichter zu beschaffen sind als frei wählbare Klartexte. Weitere Forschungen [Mat94b] haben gezeigt, daß der DES aber durchaus immun gegen die lineare Kryptoanalyse hätte sein können, wenn die S-Boxen in einer anderen Reihenfolge vorgelegen hätten, was die Vermutung bestätigt, daß die Designer des DES zumindest nichts von der lin. Kryptoanalyse wußten. Erstaunlicherweise ist der FEAL-N besser gegen lineare Kryptoanalyse gewappnet als der DES. So ist der FEAL-16 praktisch genauso sicher wie der DES, während bei einer differentiellen Analyse erst der FEAL-26 so sicher wie der DES ist.

Inzwischen gibt es eine Vielzahl von Veröffentlichungen zur lin. Kryptoanalyse, allerdings keine umfassende Beschreibung des Themas, wie es Biham und Shamir in [BiSh93] mit ihrer diff. Kryptoanalyse getan haben. Ich werde in meiner Beschreibung daher die Theorie aus verschiedenen Quellen entwickeln. Eine Verbindung zwischen der lin. Kryptoanalyse und diff. Kryptoanalyse stellt Eli Biham in [Biha94] her, indem er auch für die lin. Kryptoanalyse den Begriff der Charakteristik einführt. Kaisa Nyberg stellt in [Nyb94], dem Ansatz von Lai u.a. ähnlich, die lin. Kryptoanalyse auf eine abstraktere Grundlage und erhält so untere Schranken für den Aufwand eines Angriffs. In Kapitel 6 werde ich auch für diesen Angriff eine Markovkette definieren, so daß Sicherheit gegen den Angriff mathematisch gleich behandelt werden kann. Noch interessanter ist allerdings, daß die lin. Kryptoanalyse strukturell der diff. Kryptoanalyse sehr ähnlich ist. Auf diese Tatsache, die zuerst von Biham in [Biha94] erwähnt wurde, gehe ich in diesem und dem nächsten Kapitel näher ein.

Neben diesen Werken sollte noch die Arbeit von Kaliski und Robshaw [KaRo94] erwähnt werden, die durch Betrachtung mehrerer Approximationen gleichzeitig die Anzahl der nötigen Klartexte deutlich reduzieren können, wenn auch nur durch einen Anstieg des Berechnungsaufwands in gleichem Maße. Beim RC6 (siehe Abschnitt 7.2) und FEAL ist dieser Angriff am wirksamsten. Ich gehe auf diese Variante jedoch nicht näher ein, da sie keine neuen Erkenntnisse liefert, sondern eher ein Optimierungstrick ist, wenn man den Datenaufwand einer lin. Kryptoanalyse reduzieren will.

5.1 Grundlagen

Keiner der modernen Blockchiffrierer enthält einen linearen Faktor, wie er in Abschnitt 2.9 eingeführt wurde, jedoch können viele Chiffrierer zumindest bedingt linear approximiert werden. So gibt es für die meisten Chiffrierer Linearkombinationen von Input- und Outputbits, sodaß daraus mit einer gewissen Wahrscheinlichkeit der Wert eines Schlüsselbits gefolgert werden kann. Doch zunächst die Bezeichnungen:

Bezeichnungen

Klar- und Chiffretexte: werden wie üblich mit $P = (P_L, P_R)$ und $C = (C_L, C_R)$ bezeichnet (im Falle des DES, wie sie nach bzw. vor der Permutation IP bzw IP^{-1} vorliegen).

Eingänge der F-Funktion: Den Zwischenwert der Verschlüsselung vor der i-ten Runde kennzeichne ich mit X_i, den Teilschlüssel der i-ten Runde mit K_i.

Bits: Die Bits werden gemäß ihrer Signifikanz von rechts nach links von 0 bis $n-1$ durchgezählt, wenn ein Vektor A aus n Bits besteht. Das i-te Bit kennzeichne ich dann mit $A[i]$ (im Gegensatz zu meiner bisherigen Notation, in der ich die Bits von links nach rechts durchnummeriert habe).

Bitmengen: Γ bezeichnet eine Menge paarweise verschiedener Zahlen $\{i, j, \ldots, k\}$ zwischen 0 und $n-1$. Ich werde Γ auch als n-Bit Zahl auffassen, wobei genau diejenigen Bits gesetzt sind, deren Stelle in der Menge enthalten sind. Γ bezeichnet also eine Bitmaske.

$a \bullet b$: ist die Parität (XOR-Summe aller Bits) von dem (bitweisen) Produkt a AND $b = a \cdot b$ oder besser das binäre Skalarprodukt $a \bullet b := \sum_{i=0}^{n-1} a[i] \cdot b[i]$.

Summen: $A[i, j, \ldots, k]$ kennzeichnet die Bitsumme $A[i] \oplus A[j] \oplus \ldots \oplus A[k]$. Wenn $\Gamma = \{i, j, \ldots, k\}$, so kann auch $A \bullet \Gamma$ geschrieben werden oder noch kürzer ΓA.

Der grundlegende Angriff

Damit die lin. Kryptoanalyse möglich wird, muß es möglichst stark korrelierte Linearkombinationen von Input und Outputbits geben, damit ein Gleichungssystem der Form

$$\Gamma_1 P \oplus \Gamma_2 C = \Gamma_3 K \tag{5.1}$$

gebildet werden kann, was mir ein Bit an Schlüsselinformationen verschafft. Sind die entsprechenden Input- und Outputbits unkorreliert, so folgt aus genau der Hälfte aller möglichen Kombinationen, daß die linke Seite gleich 0 oder 1 ist, und ich kann daraus keine Informationen für den verwendeten Schlüssel ziehen. Folgt aus allen möglichen Kombinationen für die linke Seite immer ein eindeutiges Ergebnis, so besitzt das Kryptosystem einen linearen Faktor. Ist die linke Seite stets gleich 0, so hängen die entsprechenden Bits linear zusammen, bei einer 1 sind die Bits affin linear.

Existiert jedoch eine Linearkombination wie (5.1), so daß aus signifikant mehr Paaren als der Hälfte aller Möglichkeiten die linke Seite o.E. gleich 0 wird, so kann ich „raten“, daß $\Gamma_3 K$ diesen Wert tatsächlich angenommen hat, wenn die vorliegenden bekannten Klartexte auch diesen Wert ergeben, andernfalls rate ich eine 1. Dieses verschafft mir zwar nicht ein Schlüsselbit, wenn auf der rechten Seite $m > 1$ Schlüsselbits addiert werden, aber immer ein Bit an Information, da die rechte Seite dann nur noch $m-1$ Freiheitsgrade besitzt.

Der Bruchteil aller möglichen Kombinationen, die eine 0 auf der linken Seite erzeugen, wird als Wahrscheinlichkeit p dieser Approximation definiert. Je größer die Abweichung $\tilde{p} := p - \frac{1}{2}$ von p zu $^1/_2$ ist, desto effektiver ist die Approximation und desto weniger Paare wird man benötigen, um die Information für den Schlüssel zu gewinnen. Aus diesem Ansatz kann der folgende Algorithmus 1 (Tabelle 5.1) entwickelt werden.

Das folgende Lemma gibt eine Aussage über die Erfolgsrate des Algorithmusses:

Algorithmus 1:
INPUT: Lineare Approximation $(\Gamma_P, \Gamma_C, \Gamma_K)$, N Klartext – Chiffretextaare
OUTPUT: vorgeschlagener Wert für $\Gamma_K K$.

1. Sei T die Anzahl an Paaren (P,C), so daß $\Gamma_P P \oplus \Gamma_C C = 0$.
2. Ist $T > \frac{N}{2}$,
   dann rate $\Gamma_K K = 0$ wenn $\tilde{p} > 0$, sonst 1,
   sonst rate $\Gamma_K K = 1$ wenn $\tilde{p} > 0$, sonst 0.

Tabelle 5.1: **Algorithmus** 1 der lin. Kryptoanalyse

Algorithmus 2:
INPUT: Lineare Approximation $(\Gamma_P, \Gamma_C, \Gamma_F, \Gamma_K)$, N Paare (P,C)
OUTPUT: vorgeschlagener Wert für K_n und $\Gamma_K K$.

1. Für alle möglichen Kandidaten K_n^i von K_n sei T_i die Anzahl von Paaren, so daß die linke Seite von (5.2) gleich 0 ist.
2. Sei T_{max} bzw. T_{min} der maximale bzw. minimale Wert aus Schritt 1. Ist
   - $|T_{max} - \frac{N}{2}| > |T_{min} - \frac{N}{2}|$, dann schlage den Schlüssel K_n^i vor, der mit T_{max} korrespondiert und
     wenn $\tilde{p} > 0$, so rate $\Gamma_K K = 0$, sonst $\Gamma_K K = 1$,
   - $|T_{max} - \frac{N}{2}| < |T_{min} - \frac{N}{2}|$, dann schlage den Schlüssel K_n^i vor, der mit T_{min} korrespondiert und
     wenn $\tilde{p} > 0$, so rate $\Gamma_K K = 1$, sonst $\Gamma_K K = 0$.

Tabelle 5.2: **Algorithmus** 2 der lin. Kryptoanalyse

Lemma 5.1 ([Mat93]) *Sei N die Anzahl der bekannten zufälligen Klartexte und p die Wahrscheinlichkeit, daß Gleichung (5.1) erfüllt ist und sei $|\tilde{p}|$ klein. Dann kann die Erfolgswahrscheinlichkeit von Algorithmus 1 angenähert werden durch*

$$\int_{-2\sqrt{N}|\tilde{p}|}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx.$$

Bemerkung: Die Formel folgt aus der Approximation der Binomialverteilung, die den Erfolg des Testes vorgibt, durch eine Normalverteilung mit anschließender Standardisierung. Die folgende Tabelle beschreibt die Erfolgswahrscheinlichkeit in Abhängigkeit von N:

N	$\frac{1}{4}\lvert\tilde{p}\rvert^{-2}$	$\frac{1}{2}\lvert\tilde{p}\rvert^{-2}$	$\lvert\tilde{p}\rvert^{-2}$	$2\lvert\tilde{p}\rvert^{-2}$
Erfolgsrate	84,1%	92,1%	97,7%	99,8%

Das heißt, die Erfolgsrate von Algorithmus ist umgekehrt proportional zu $\tilde{p}^2$. □

Algorithmus 1 verschafft mir lediglich ein Bit an Information, daher ist er noch wenig geeignet, um einen Angriff durchzuführen. Daher wird nach linearen Approximationen gesucht, die die Ergebnisse der Rundenfunktion F nach der ersten und/oder letzten Runde mit einbeziehen. Dies resultiert in Approximationen der Form

$$P \bullet \Gamma_1 \oplus C \bullet \Gamma_2 \oplus F_n(C_L, K_n) \bullet \Gamma_3 = K \bullet \Gamma_4 \tag{5.2}$$

und einem entsprechend modifizierten Algorithmus 2 (Tabelle 5.2), dessen Erfolgsrate allerdings nicht mehr so leicht bestimmt werden kann und im Allgemeinen von der Anzahl der zu bestimmenden Schlüsselbits abhängt.

Anmerkung: Dieser Schritt entspricht dem Vorgehen bei der diff. Kryptoanalyse, ein um ein bis zwei Runden kürzeres Differential zu verwenden, um möglichst hohe Wahrscheinlichkeiten zu erhalten. Nur kommt hier zusätzlich die praktische Eigenschaft hinzu, daß die Anzahl der ermittelten Schlüsselbits deutlich erhöht wird, da man eigentlich mehrere Analysen gleichzeitig durchführt. □

Bemerkung: Es müssen z.B. beim DES nicht alle Möglichkeiten für K_n im Algorithmus 2 ausprobiert werden, sondern nur diejenigen Bits, die für die Berechnung von $F_n(C_L, K_n) \bullet \Gamma_3$ erforderlich sind. Diese Bits werden in der Literatur zur lin. Kryptoanalyse auch *effektive* Schlüsselbits genannt. Im Falle des DES sind die effektiven Schlüsselbits sehr leicht zu bestimmen, da nur die Schlüsselbits der betroffenen S-Boxen in der Approximation (5.2) eine Rolle spielen, also sechs Bits pro S-Box. Beim FEAL ist dies ein wenig schwieriger, da hier die S-Boxen sich gegenseitig beeinflussen. Daraus resultieren beim FEAL größere Zahlen an effektiven Schlüsselbits, da stets mindestens 16 Schlüsselbits betrachtet werden müssen, was die Anzahl der nötigen Zähler in Algorithmus 2 an die Grenze des Machbaren bringt. □

Besteht der zu analysierende Chiffretext allein aus ASCII Texten in englischer Sprache oder allgemein in 7-Bit Kodierung, so ist es möglich, den Angriff zu einem reinen Chiffretextangriff zu verwandeln. Matsui stellt in [Mat94] eine Approximation vor, die nur die achten Klartextbits benutzt, die bei solchen Texten ja Null sind. Dadurch ist die Approximation nicht mehr von Klartextbits abhängig und es wird nur noch der Chiffretext zur Analyse benötigt. Da diese Approximation nicht besonders gut ist, steigert sich natürlich der Aufwand der lin. Kryptoanalyse. Eine wesentlich bessere Approximation verwendet zwar auch Bits, die 1 werden können, wo aber aufgrund der Eigenschaften der englischen Sprache mit einer Wahrscheinlichkeit von 0,35 eine 0 geraten werden kann. Dieser Angriff kann einen 8-Runden DES mit 2^{29} Chiffretextblöcken brechen, also unter Verwendung von 4 GB Chiffretext im Gegensatz zu den 2×16 MB an bekannten Klartexten für den normalen Angriff.

Ich übertrage die Idee der lin. Kryptoanalyse nun auf den DES. Da bezüglich des XOR alle Operationen bis auf die S-Boxen linear sind, reduziert sich die Suche nach geeigneten Approximationen auf die Untersuchung dieser Funktionen.

Jede S-Box nimmt einen 6-Bit Input und gibt einen 4-Bit Output aus, damit gibt es 2^6 mögliche Linearkombinationen von Inputbits und 2^4 LK von Outputbits. Anhand der Definitionen der S-Boxen kann ich nun für jede Kombination untersuchen, ob die Summe der Inputbits mit der Summe der Outputbits übereinstimmt.

Definition 5.1 *Für eine gegebene S-Box Si,* $(i = 1, \ldots, 8)$*,* $1 \le a \le 63$ *und* $1 \le b \le 15$ *sei* $\mathbf{NSi(a,b)}$ *definiert als die Anzahl der 64 möglichen Inputs von Si, so daß die Summe der Inputbits maskiert durch a mit der Summe der Outputbits maskiert durch b übereinstimmt, oder in Formeln:*

$$NSi(a,b) := \#\{x | 0 \le x \le 63, x \bullet a = Si(x) \bullet b\}. \qquad (5.3)$$

Korollar 5.2 ([Mat93]) *Aufgrund der Definition der S-Boxen (Bit 0 und 5 wählen eine 4 Bit Permutation aus) sind die folgenden Aussagen trivial:*

1. $NSi(a,b)$ *ist gerade.*
2. *Ist* $a = 1, 32, 33$*, dann ist* $NSi(a,b) = 32$ *für alle i und b.*

Beispiel 5.1: In Tabelle 5.3 sind für S-Box S5 alle $NS5(a,b) - 32$ angegeben (Matsui nennt sie Lineare Approximationstabellen). So ist z.B. $NS5(16, 15) = 12$, d.h. in nur 12 aller 64 möglichen Fälle gilt die Gleichung $x[4] = S5(x)[0, 1, 2, 3]$. Mit anderen Worten, mit einer Wahrscheinlichkeit von $\frac{12}{64} \approx 0,19$ stimmt der Wert von $x[4]$ mit der Summe aller Outputbits überein. Dieser Wert ist der extremste aller S-Boxen und ist auch der Ansatz für den Angriff auf den DES. Bezieht man die gesamte Rundenfunktion des DES in Betracht, so ergibt sich daraus die folgende Gleichung, die mit Wahrscheinlichkeit $\frac{12}{64}$ gilt:

$$X[15] \oplus F(X,K)[7,18,24,29] = K[22] \quad \text{mit } p \approx 0,19. \tag{5.4}$$

Da $NS5(16,15)$ der maximale Wert für alle S-Boxen ist, ist diese Gleichung die beste Approximation für die Rundenfunktion F. □

Bemerkung: Ist $NSi(a,b) = 32$, so ist die entsprechende Linearkombination unkorreliert und man kann daraus keine weiteren Schlüsse ziehen. Deswegen wurden von allen $NS5(a,b)$ in Tabelle 5.3 32 abgezogen, um den Abstand zu dieser Marke zu verdeutlichen. Ist der Wert > 0, so ist diejenige LK positiv korreliert oder „linearer"; ist der Wert negativ, so ist die LK „affiner". Beide Extreme können zur lin. Kryptoanalyse ausgenutzt werden, indem man Algorithmus 1 und 2 verwendet. □

Zum Angriff auf ein n-Runden Kryptosystem fehlt noch ein Formalismus, der eine Approximation für beliebig viele Runden erlaubt. Darüberhinaus werden noch Erfolgswahrscheinlichkeiten für den zweiten Algorithmus benötigt, also die Anzahl der benötigten Paare. Der wichtigste Punkt ist aber das Finden von geeigneten oder besser noch, den besten Approximationen mit maximalem $|\tilde{p}|$.

5.2 Charakteristiken und Lineare Hüllen

Um die linearen Approximationen über mehrere Runden auszudehnen, sucht Matsui nach Approximationen, so daß die Outputbits einer Runde mit den Inputbits der nächsten Runde übereinstimmen, damit sich die Bits von Zwischenergebnissen gegenseitig aufheben ($a \oplus a = 0$). Biham bringt in [Biha94] diese Konstruktion in Verbindung mit seinen Charakteristiken, was die lin. Kryptoanalyse näher an die diff. Kryptoanalyse heranrücken läßt. Wie bei der diff. Kryptoanalyse wird zuerst angenommen, daß die Daten- und Schlüsselinputs einer jeden Runde stochastisch unabhängige Zufallsvariablen sind, um die Theorie zu vereinfachen. Die Wahrscheinlichkeiten für abhängige Schlüssel sind aber *praktisch* gleich, wie bereits bei der diff. Kryptoanalyse erwähnt.

Definition 5.2 *Eine* **1-Runden Charakteristik** *ist ein Tupel* $(\Gamma_P, \Gamma_C, \Gamma_K, p)$ *mit* $(\Gamma_P)_L = (\Gamma_C)_L = A'$, $(\Gamma_P)_R \oplus (\Gamma_C)_R = a'$, *so daß mit Wahrscheinlichkeit p die Gleichung* $\Gamma_P P \oplus \Gamma_C C \oplus \Gamma_K K = 0$ *erfüllt ist.* Γ_P *ist die Teilmenge an Inputbits vor der Runde,* Γ_C *die Teilmenge an Outputbits nach der Runde und* Γ_K *die Teilmenge der beteiligten Schlüsselbits, dessen Parität mit Wahrscheinlichkeit p approximiert wird.*

Beispiel 5.2: Wie bei der diff. Kryptoanalyse ist es sehr einfach, eine 1-Runden Charakteristik zu erzeugen, die nur eine S-Box beeinflußt, indem man sich einen Eintrag ungleich 32 in der entsprechenden linearen Approximationstabelle heraussucht und danach Γ_P, Γ_C und Γ_K gemäß Definition wählt. Abbildung 5.1 beschreibt z.B. Gleichung (5.4) als Charakteristik. □

Bemerkung: Man kann auch mehr als eine aktive S-Box betrachten, allerdings gibt es jetzt einige Unterschiede zur diff. Kryptoanalyse: Eine S-Box wird nicht automatisch dadurch aktiviert, indem man Inputbits benutzt, die durch die Expansion auch benachbarte S-Boxen beeinflussen. Und selbst falls man die benachbarte S-Box mit einbezieht, heben sich in beiden Gleichungen verwendete Bits gegenseitig auf. Sind beide S-Boxen nicht benachbart, so multiplizieren sich die Wahrscheinlichkeiten gemäß Matsuis Piling-Up-Lemma (siehe Lemma 5.3). Benachbarte S-Boxen beeinflussen sich jedoch, wie oben geschildert, so daß keine geschlossene Formel für diesen Fall angegeben werden kann. Die besten Approximationen sind deshalb diejenigen, die möglichst wenige S-Boxen benutzen. □

Durch Konkatenation von Charakteristiken kann nun der Begriff auf n-Runden Charakteristiken ausgedehnt werden:

$a \setminus b$	1_x	2_x	3_x	4_x	5_x	6_x	7_x	8_x	9_x	A_x	B_x	C_x	D_x	E_x	F_x
1_x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2_x	4	-2	2	-2	2	-4	0	4	0	2	-2	2	-2	0	-4
3_x	0	-2	6	-2	-2	4	-4	0	0	-2	6	-2	-2	4	-4
4_x	2	-2	0	0	2	-2	0	0	2	2	4	-4	-2	-2	0
5_x	2	2	-4	0	10	-6	-4	0	2	-10	0	4	-2	2	4
6_x	-2	-4	-6	-2	-4	2	0	0	-2	0	-2	-6	-8	2	0
7_x	2	0	2	-2	8	6	0	-4	6	0	-6	-2	0	-6	-4
8_x	0	2	6	0	0	-2	-6	-2	2	4	-12	2	6	-4	4
9_x	-4	6	-2	0	-4	-6	-6	6	-2	0	-4	2	-6	-8	-4
A_x	4	0	0	-2	-6	2	2	2	2	-2	2	4	-4	-4	0
B_x	4	4	4	6	2	-2	-2	-2	-2	-2	2	0	-8	-4	0
C_x	2	0	-2	0	2	4	10	-2	4	-2	-8	-2	4	-6	-4
D_x	6	0	2	0	-2	4	-10	-2	0	-2	4	-2	8	-6	0
E_x	-2	-2	0	-2	4	0	2	-2	0	4	2	-4	6	-2	-4
F_x	-2	-2	8	6	4	0	2	2	4	8	-2	8	-6	2	0
10_x	2	-2	0	0	-2	-6	-8	0	-2	-2	-4	0	2	10	-20
11_x	2	-2	0	4	2	-2	-4	4	2	2	0	-8	-6	2	4
12_x	-2	0	-2	2	-4	-2	-8	4	6	4	6	-2	4	-6	0
13_x	-6	0	2	-2	4	2	0	4	-6	4	2	-6	4	-2	0
14_x	4	-4	0	0	0	0	0	-4	-4	4	4	0	4	-4	0
15_x	4	0	-4	-4	4	-8	-8	0	0	-4	4	8	4	0	4
16_x	0	6	6	2	-2	4	0	4	0	6	2	2	2	0	0
17_x	4	-6	-2	6	-2	-4	4	4	-4	-6	2	-2	2	0	4
18_x	6	0	2	4	-10	-4	2	2	0	-2	0	2	4	-2	-4
19_x	2	4	-6	0	-2	4	-2	6	8	6	4	10	0	2	-4
$1A_x$	2	2	-8	-2	4	0	2	-2	0	4	2	0	-2	-2	0
$1B_x$	2	6	-4	-6	0	0	2	6	8	0	-2	-4	-6	-2	0
$1C_x$	0	-2	2	4	0	-6	2	-2	6	-4	0	2	-2	0	0
$1D_x$	4	-2	6	-8	0	-2	2	10	-2	-8	-8	2	2	0	4
$1E_x$	-4	-8	0	-2	-2	-2	2	-2	2	-2	6	4	4	4	0
$1F_x$	-4	8	-8	2	-6	-6	-2	-2	2	-2	-2	-8	0	0	-4
20_x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21_x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22_x	-4	-2	2	-2	2	-4	8	-4	0	-6	6	2	-2	-16	-12
23_x	0	-2	-2	6	-2	-4	4	0	0	-2	-2	-2	6	4	-4
24_x	-2	6	4	0	6	-2	4	4	-6	-2	4	0	14	2	0
25_x	6	2	0	0	6	2	0	-4	-6	2	-8	0	-2	6	-4
26_x	2	4	-2	-2	0	2	-4	4	-2	-4	-2	6	0	-2	0
27_x	-10	0	-2	6	4	6	-4	0	6	-12	2	2	0	6	-4
28_x	4	-2	-2	0	4	-6	2	2	-6	4	0	6	-2	-4	0
29_x	0	2	6	0	0	6	2	2	-2	-8	0	-2	-6	0	0
$2A_x$	0	-4	-8	6	6	6	-6	6	2	-2	-2	-8	4	-4	4
$2B_x$	8	0	4	6	-2	-6	6	2	6	-2	6	-4	0	4	4
$2C_x$	2	4	-6	0	-6	0	6	-2	-4	2	-4	-2	4	6	0
$2D_x$	-2	-4	-2	0	-2	-8	2	-2	0	-6	-8	-2	0	-2	4
$2E_x$	6	2	-4	6	4	4	-2	-10	-8	0	-2	4	-2	2	0
$2F_x$	6	-6	-4	6	-4	4	-2	2	4	4	-6	0	2	-2	-4
30_x	2	-2	0	-4	-6	-2	-4	4	2	2	0	0	2	2	4
31_x	2	-2	0	0	-2	2	0	0	-2	-2	-4	0	2	2	4
32_x	6	0	-2	-2	8	2	4	0	10	0	2	-2	4	2	0
33_x	-6	0	10	2	0	-2	-4	0	6	0	-10	2	4	-2	0
34_x	0	-12	4	-4	0	4	-8	-4	0	-4	0	-4	-4	0	0
35_x	-8	0	0	8	-4	4	0	0	-4	-4	0	4	4	-4	4
36_x	4	-2	-6	-2	-2	8	0	4	-4	-2	-2	6	2	-4	0
37_x	-8	-6	-6	-6	6	0	4	12	0	2	-2	2	2	4	-4
38_x	2	4	-6	0	-2	4	-2	-6	4	-6	0	6	4	-2	0
39_x	-2	8	2	-4	6	-4	-6	-2	-4	2	4	-2	0	2	0
$3A_x$	6	-10	0	2	4	0	-2	6	-4	0	2	4	-2	-2	-4
$3B_x$	-2	-6	-4	-10	0	-8	-2	-10	4	4	-2	0	2	-2	4
$3C_x$	-8	-6	-2	0	-4	2	2	-6	2	4	0	10	-2	4	4
$3D_x$	4	2	2	4	4	-2	2	-2	10	0	0	2	2	4	0
$3E_x$	-4	4	-4	2	2	-2	2	2	-2	-2	-2	4	-4	0	4
$3F_x$	-4	-4	-4	14	6	-6	-2	2	-2	6	-2	0	0	-4	0

Tabelle 5.3: Lineare Approximationstabelle für S5

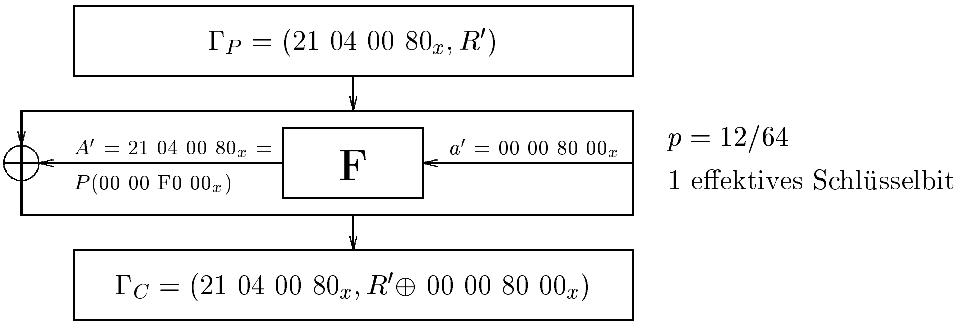

Abbildung 5.1: Beste 1-Runden Approximation des DES

Definition 5.3 *Eine n-Runden Charakteristik* $\Gamma^1 = (\Gamma_P^1, \Gamma_C^1, \Gamma_K^1, p_1)$ *kann mit einer m-Runden Charakteristik* $\Gamma^2 = (\Gamma_P^2, \Gamma_C^2, \Gamma_K^2, p_2)$ **konkateniert** *werden, wenn* $\Gamma_C^1 = V(\Gamma_P^2)$ *(V vertauscht beide Hälften eines Blocks). Die* **Konkatenation** *der Charakteristiken* Γ^1 *und* Γ^2 *ist dann die* $(n+m)$*-Runden Charakteristik* $\Gamma = (\Gamma_P^1, \Gamma_C^2, \Gamma_K^1 \oplus \Gamma_K^2, \frac{1}{2} + 2\tilde{p}_1\tilde{p}_2)$.

Lemma 5.3 (Matsuis Piling-Up-Lemma) *Seien* X_i $(1 \le i \le n)$ *unabhängige Zufallsvariablen, so daß der Wert 0 mit Wahrscheinlichkeit* p_i *und der Wert 1 mit Wahrscheinlichkeit* $(1-p_i)$ *angenommen wird. Dann ist die Wahrscheinlichkeit, daß* $X_1 \oplus X_2 \oplus \ldots \oplus X_n = 0$ *gegeben durch*

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \tilde{p}_i.$$

Beweis: Ich zeige die Behauptung für $n = 2$, das Lemma folgt dann leicht per Induktion.

Die Summe $X_1 \oplus X_2$ wird genau dann 0, wenn beide Variablen X_1 und X_2 den gleichen Wert annehmen. Der erste Fall tritt mit Wahrscheinlichkeit $p_1 p_2$ auf, der zweite mit Wahrscheinlichkeit $(1-p_1)(1-p_2)$. Damit:

$$\begin{aligned}
P(X_1 \oplus X_2 = 0) &= P(X_1 = 0 \wedge X_2 = 0) + P(X_1 = 1 \wedge X_2 = 1) \\
&= p_1 p_2 + (1-p_1)(1-p_2) = 2p_1p_2 + 1 - p_1 - p_2 \\
&= 1 - \left(\tilde{p}_1 + \frac{1}{2}\right) - \left(\tilde{p}_2 + \frac{1}{2}\right) + 2\left(\tilde{p}_1 + \frac{1}{2}\right)\left(\tilde{p}_2 + \frac{1}{2}\right) \\
&= -\tilde{p}_1 - \tilde{p}_2 + 2\left(\frac{1}{4} + \tilde{p}_1\tilde{p}_2 + \frac{\tilde{p}_1}{2} + \frac{\tilde{p}_2}{2}\right) = \frac{1}{2} + 2\tilde{p}_1\tilde{p}_2
\end{aligned}$$

□

Korollar 5.4 ([Biha94]) *Unter der Voraussetzung, daß die Inputs jeder Runde stochastisch unabhängig und gleichverteilt zufällig sind, ist die Wahrscheinlichkeit p einer Charakteristik, die durch Konkatenation von l Charakteristiken mit Wahrscheinlichkeiten* p_i *entstanden ist, gegeben durch* $p = 1/2 + 2^{l-1} \prod_{i=1}^{l} \tilde{p}_i$.

Bemerkung: Die Wahrscheinlichkeit wurde unter der Benutzung von zwei tatsächlich nicht vorhandenen Aussagen berechnet:

1. Die Dateninputs sind nicht wirklich unabhängig zufällig. Die Klartexte sollten zufällig gewählt werden, obwohl dies bei vorliegenden Klartexten nicht unbedingt gegeben ist, aber die Inputs jeder Runde sind dies meistens nicht. Werden unabhängige Schlüssel verwendet, so sind die Dateninputs jeder Runde tatsächlich unabhängig aufgrund der Verknüpfung mit den Teilschlüsseln, bei abhängigen Schlüsseln gilt dies aber nicht. Diese Einschränkung ist allerdings in der Praxis akzeptabel und wurde bei der diff. Kryptoanalyse auch gemacht.

2. Der zweite Punkt ist die aus der diff. Kryptoanalyse bekannte Hypothese der stochastischen Äquivalenz (Seite 53). Die Formel aus Korollar 5.4 geht davon aus, daß die Schlüssel unabhängig und gleichverteilt zufällig gewählt wurden. Tatsächlich ist bei der Analyse aber der Schlüssel fest. Deswegen ist die Wahrscheinlichkeit eigentlich ein Erwartungswert über alle Schlüssel; interessant ist aber nur die Wahrscheinlichkeit für den gesuchten Schlüssel. Trifft die Hypothese der stochastischen Äquivalenz für den Chiffrierer zu, so ist für fast alle Schlüssel die Wahrscheinlichkeit nahe beim Erwartungswert und für die Praxis kann von der Gleichheit ausgegangen werden.

□

Matsui bemerkt in [Mat93] indirekt, daß die Hypothese für die Berechnung der Erfolgsrate von Algorithmus 2 wesentlich ist. Falls die Hypothese zutrifft, erhält er mit $N \approx 8|\tilde{p}|^{-2}$ bekannten Klartexten das richtige Ergebnis mit einer Wahrscheinlichkeit von über 95%. Die tatsächlich beobachtete Wahrscheinlichkeit einer 7-Runden Approximation war beim DES sogar noch höher, während Experimente beim FEAL gezeigt haben, daß die Wahrscheinlichkeit niedriger ist. Aufgrund der Komplexität der Formel verzichte ich hier darauf und verweise auf den Quelltext.

Die Charakteristiken der lin. Kryptoanalyse unterscheiden sich grundlegend von den Charakteristiken der diff. Kryptoanalyse. Man sollte sich stets vor Augen halten, daß die Werte in den Charakteristiken keine tatsächlichen Werte sind oder irgendwelche Differenzen von Werten, sondern Linearkombinationen von Input- und Outputbits. Differenzen wurden durch die XOR-Operation am Ende einer Runde nicht beeinflußt, aber um die Parität einer Linearkombination von Outputbits der XOR-Operation zu kennen, muß man die Parität der selben LK von Inputbits von beiden Variablen kennen. Deswegen steckt die wichtige Information einer Charakteristik in den linken Teilblöcken von Γ_P, Γ_C, um den Informationsfluß durch alle Runden zu gewährleisten, wogegen die rechten Teilblöcke nur entsprechend gewählt werden müssen, damit der Output der F-Funktion die gewünschte LK für die XOR-Operation am Ende der Runde erzeugt. Es gelten folgende Beziehungen für die beiden Operationen vor und nach der F-Funktion:

XOR-Verknüpfung $\oplus$ **:**	**Duplikation** $\dashv$ **:**
Ist $X = Y \oplus Z$,	Ist $X \dashv (Y, Z)$ und $X = Y = Z$,
dann gilt bei der	dann gilt bei der
DK: $X' = Y' \oplus Z'$	DK: $X' = Y' = Z'$
LK: $\Gamma X = \Gamma Y = \Gamma Z$	LK: $\Gamma X = \Gamma Y \oplus \Gamma Z$

Besonders deutlich wird dies an der trivialen 1-Runden Charakteristik der lin. Kryptoanalyse $(\Gamma_P = (0, R), \Gamma_C = (0, R), \Gamma_K = 0)$ mit Wahrscheinlichkeit 1, die nur leere Teilmengen an Input- und Outpubits betrachtet und daher keine Schlüsselbits ermitteln kann. Die triviale 1-Runden Charakteristik der diff. Kryptoanalyse besaß ihre freie Variable in der linken Hälfte, während in der lin. Kryptoanalyse die freie Variable in der rechten Hälfte sitzt. Abbildung 5.2 verdeutlicht den Unterschied im Informationsfluß einer 1-Runden Charakteristik bei der diff. und linearen Kryptoanalyse.

Ein weiterer Unterschied ist die Nutzung von Differentialen. Bei der diff. Kryptoanalyse konnten alle Charakteristiken mit gleichen Input- und Outputdifferenzen zu Differentialen zusammengefaßt werden, da die Zwischenwerte einer Charakteristik für die Analyse nicht von Belang waren. Bei der lin. Kryptoanalyse hingegen stecken in den Zwischenwerten die Informationen über die ermittelbaren Schlüsselbits. Gibt es also zwei Charakteristiken mit gleichen Γ_P, Γ_C, so beeinflussen sich diese gegenseitig und könnten sich sogar schlimmstenfalls aufheben. Derartige Charakteristiken können in zwei Kategorien aufgeteilt werden:

1. die Teilmengen der Schlüsselbits sind nicht identisch,

2. die Teilmengen der Schlüsselbits sind identisch, aber die Wahrscheinlichkeiten der Charakteristiken sind unterschiedlich.

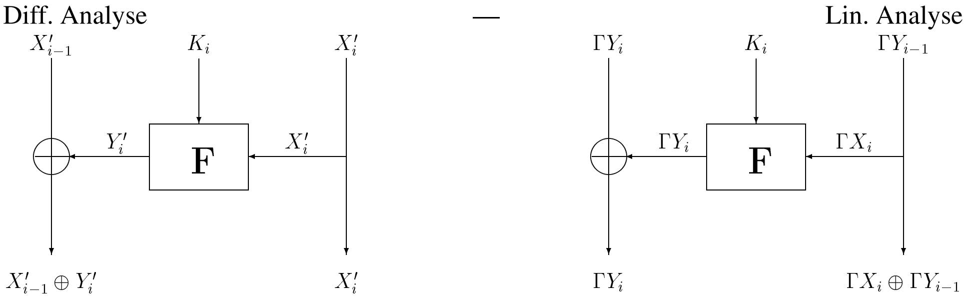

Abbildung 5.2: Informationsfluß in der diff. und linearen Analyse

Diese sich gegenseitig beeinflussenden Charakteristiken hat Kaisa Nyberg in [Nyb94] zu sogenannten „linearen Hüllen“ zusammengefaßt und gezeigt, daß die Erfolgsrate einer Charakteristik und damit die Erfolgsrate von Algorithmus 2 an ihrer linearen Hülle zu messen ist. Wie bei den Differentialen ist die Erfolgsrate einer Charakteristik in der Regel geringfügig höher als Matsuis Schätzung aus [Mat93].

Definition 5.4 *Eine* **lineare Hülle** $ALH(a,b)$ *ist eine Familie von n-Runden Charakteristiken* $(\Gamma^i_P, \Gamma^i_C, \Gamma^i_K, p_i)$ *mit* $\Gamma^i_P = a, \Gamma^i_C = b \quad \forall i$. *Sei*

$$\begin{aligned} pot(a,b,;k) &:= |P_X(a \bullet X + b \bullet F(X,k) + c \bullet k = 0) - \frac{1}{2}|^2 \\ &= |P_X(a \bullet X + b \bullet F(X,k) = 0) - \frac{1}{2}|^2 \\ pot(a,b,c) &:= |P_X(a \bullet X + b \bullet F(X,K) + c \bullet K = 0) - \frac{1}{2}|^2. \end{aligned}$$

Der Wert $pot(a,b;k)$ *ist das* Potential der linearen Approximation $a \bullet X + b \bullet F(X,k)$ für den Schlüssel k. *Der Wert* $pot(a,b,c)$ *ist das* Potential der linearen Approximation $a \bullet X + b \bullet F(X,K) + c \bullet K$. *Mit diesen Bezeichnungen sei*

$$pot(a,b) := \sum_c pot(a,b,c)$$

das **Potential der linearen Hülle** $ALH(a,b)$.

Das folgende Theorem, eine Verallgemeinerung der Parsevalschen Gleichung, wird mir bei der weiteren Analyse der Potentiale helfen (der Beweis kann in [Nyb94] gefunden werden):

Theorem 5.5 ([Nyb94]) *Seien* $X \in \mathbb{Z}_2^m$ *und* $K \in \mathbb{Z}_2^l$ *unabhängige Zufallsvariablen und K gleichverteilt, dann gilt für alle* $a \in \mathbb{Z}_2^m, b \in \mathbb{Z}_2^n$ *und* $\gamma \in \mathbb{Z}_2^l$

$$2^{-l} \sum_{k \in \mathbb{Z}_2^l} |P_X(a \bullet X + b \bullet F(X,k) = 0) - \frac{1}{2}|^2 =$$

$$2^{-l} \sum_k |P_X(a \bullet X + b \bullet F(X,k) + \gamma \bullet k = 0) - \frac{1}{2}|^2 =$$

$$\sum_{c \in \mathbb{Z}_2^l} |P_{X,K}(a \bullet X + b \bullet F(X,K) + c \bullet K = 0) - \frac{1}{2}|^2.$$

Das Ergebnis des Theorems kann wie folgt ausgedrückt werden: Das durchschnittliche Potential einer linearen Approximation über alle Schlüssel entspricht dem Potential ihrer linearen Hülle. Da die Summe der Potentiale $pot(a,b,c)$ über alle c in der Regel größer ist als das Potential $pot(a,b;k)$ der linearen Approximation, ist die Erfolgsrate der Charakteristik größer als durch die von Matsui gegebene Formel (N ist ja umgekehrt proportional zum Potential zu wählen!). Gleichheit besteht nur dann, wenn es genau ein c mit einer nichtverschwindenden Wahrscheinlichkeit $\tilde{p}$ gibt, und selbst dann könnte es aufgrund der Hypothese der stochastischen Äquivalenz zu Abweichungen kommen, aber das betrifft lineare und differentielle Kryptoanalyse gleichermaßen.

Bemerkung: Ich hatte im vorigen Abschnitt bemerkt, daß es auch möglich ist, die lin. Kryptoanalyse in einen reinen Chiffretextangriff zu verwandeln, wenn die vorliegenden Chiffretexte aus englischer Sprache bestehen. Eine wesentliche Voraussetzung für die Berechnung der Wahrscheinlichkeit einer Charakteristik oder der zugehörigen linearen Hülle ist aber die Gleichverteilung der Klartexte, wie im vorigen Theorem deutlich sichtbar. Da Sprache aber eine sehr hohe Redundanz besitzt, ist es fraglich, ob die von Matsui gemachten Angaben über den Aufwand der entsprechenden Analysen wirklich richtig sind. Matsui bemerkt selbst, daß noch weitere Forschungen nötig sind, um das Problem der Nichtzufälligkeit zu verstehen, bisher wurden aber noch keine weiteren Ergebnisse bekanntgegeben. □

5.3 Lineare Kryptoanalyse des FEAL-4

Um den Angriff praktikabler zu gestalten, muß wie bei der diff. Kryptoanalyse der FEAL umgeschrieben werden, um das *Pre- und Postwhitening*[1] am Anfang und am Ende des FEAL zu umgehen. Abbildung 5.3 zeigt die zum FEAL äquivalente Struktur, die das Whitening in die Mitte des FEAL schiebt[2]. Die Rundenfunktion F ändert sich auch geringfügig, der wichtigste Unterschied sind die nun 32 Bit großen Teilschlüssel, von Biham und Shamir auch *actual subkeys* genannt. Aus diesen AKi kann, abhängig von der Rundenzahl, der ursprüngliche erweiterte Schlüssel unterschiedlich zurückgewonnen werden, um den eigentlichen 64-Bit Schlüssel zu berechnen. Dazu benötigt man aber alle Teilschlüssel, wodurch man es genauso gut auch bei dem erweiterten Schlüssel belassen kann. Damit ist auch bei der lin. Kryptoanalyse kein Unterschied zwischen FEAL-N und FEAL-NX zu verzeichnen, da wieder direkt die Teilschlüssel ermittelt werden.

Da die S-Boxen nicht parallel zueinander agieren, ist die Anzahl an effektiven Schlüsselbits beim FEAL größer, nämlich ca. 16 im Gegensatz zu den minimalen sechs Bits beim DES. Dadurch ist es nicht möglich, Algorithmus 2 mit einer Approximation hindurchzuführen, die die erste *und* letzte Runde entfernt, was sich beim DES als sehr wirksam erwiesen hat. Außerdem bekommt man keine Information über Schlüsselbits des Gesamtschlüssels, sondern nur die des entsprechenden Teilschlüssels. Da als nichtlineare Komponente lediglich eine Addition modulo 256 eingesetzt wird, beeinflussen im Gegensatz zum DES nicht alle Schlüsselbits gleichermaßen das Ergebnis einer Approximation, wodurch sich unter Umständen einige Bits aus einer Approximation entfernen lassen. So wird sich meistens der Anteil der effektiven Schlüssel- und Textbits auf ein erträgliches Maß reduzieren lassen. Weiterhin ist wegen dieser Eigenschaft die Erkennung der von der untersuchten Stelle weit entfernten Bits nicht sehr zuverlässig. Mit sehr wenigen Klartexten können beim FEAL-4 nur Bits zuverlässig erkannt werden, die nicht weiter als 2–3 Bits unterhalb der approximierten Stelle liegen.

Doch nun zum Angriff: Ich verwende als Grundlage die 3-Runden Approximation aus [MaYa92], der Rest des Angriffs beruht auf eigener Forschung. Da diese Approximation

[1] Whitening nennt man das Addieren von Schlüsselmaterial zum Datenblock. Dieses zuerst beim DESX eingeführte Verfahren erschwert erheblich vollständige Schlüsselsuchen, aber auch diff. und lin. Kryptoanalyse

[2] [BiSh93] benutzt eine Version des FEAL, die nur das Postwhitening entfernt

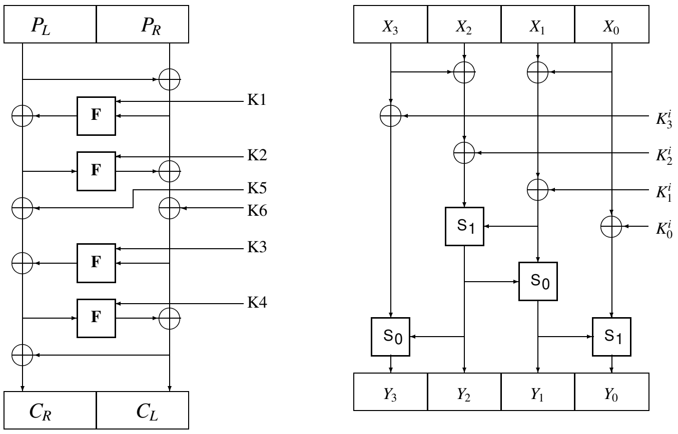

Abbildung 5.3: Modifizierter FEAL-4 und Rundenfunktion F

die Linearität der Addition im niederwertigsten Bit ausnutzt, ist die lin. Kryptoanalyse des FEAL-4 sogar deterministisch anstatt probabilistisch; d.h. ich brauche nur wenige Klartexte, um zu überprüfen, ob die Approximation stets den gleichen Wert für ΓK vorschlägt. Diesen Wert werde ich jedoch ignorieren, da er mir ohnehin nur ein Bit an Information veschafft. Die meisten Informationen erhalte ich aus der Einbeziehung der Rundenfunktion von Runde eins und vier in Algorithmus 2, so daß ich die 3-Runden Approximation sowohl in Runde 1–3 als auch in Runde 2–4 ansetzen werde.

Mein erster Versuch einer lin. Kryptoanalyse ermittelte zuerst die Teilschlüssel K_1 und K_4 und führte dann eine Analyse der inneren beiden Runden durch. Meine programmierte Version konnte allerdings bei diesem Angriff den Schlüssel so nie berechnen. Der Grund dafür war, daß Algorithmus 2 nicht auf einem einzigen Feistelzyklus operieren kann. Ich versuchte eine lineare Approximation in einer Runde und die Überbrückung der anderen Runde durch einen Ausdruck in der Rundenfunktion. Da die lineare Approximation nur einen halben Zyklus überbrückte, habe ich keinerlei Informationen erhalten, die mir die Erzeugung eines zuverlässigen Schlüsselkandidaten in Algorithmus 2 erlaubte. Daher habe ich das Verfahren modifiziert und entferne nach der Berechnung von K_1 oder K_4 nur eine Runde.

Äquivalente Schlüssel

Im Laufe meiner Programmierung der lin. Kryptoanalyse des FEAL-4 bin ich auf die interessante Tatsache gestoßen, daß nicht alle erweiterten Schlüssel wirklich verschieden sind. Vielmehr liegt jeder Schlüssel in einer Äquivalenzklasse von 256 Schlüsseln. Dies liegt daran, daß im Laufe der Verschlüsselung nicht der Wert einiger Schlüsselbits von Bedeutung ist, sondern vielmehr die Summe (XOR) bestimmter Bits. Besonders leicht zu sehen sind die Summen

$$K_2[7] + K6[1], \qquad K_2[31] + K_6[25],$$
$$K_3[7] + K5[1], \qquad K_3[31] + K_5[25],$$

die unmittelbar die Berechnung der Whitening Schlüssel beeinflussen. Ist z.B. beim gesuchten Schlüssel $K_2[7] = 0$ und $K_6[1] = 1$, so wird der Schlüssel, der genau die entgegengesetzten Werte enthält, dasselbe Verschlüsselungsergebnis erzeugen, da die Änderung in K_2 bei der Verschlüsselung durch das Kippen des Bits in K_6 wieder kompensiert wird. Neben diesen Summen gibt es noch vier weitere, die auch Schlüsselbits der äußeren Runden einbeziehen:

$$K_1[7] + K_2[1;9] + K_5[1], \qquad K_1[31] + K_2[17;25] + K_5[25],$$

$$K_4[7] + K_3[1;9] + K_6[1], \qquad K_4[31] + K_3[17;25] + K_6[25].$$

Ändere ich z.B. das siebte Bit in K_1, so zieht dies die Kippung der Bits 1 und 9 in K_2 nach sich, als auch von $K_5[1]$, wenn die Verschlüsselung wieder gleich sein soll. Alle acht Gleichungen können beliebig miteinander kombiniert werden, wodurch sich 256 Möglichkeiten der Variation ergeben.

Da der Übertrag der Summe der achten Bits bei der Addition modulo 256 ignoriert wird, waren diese Beziehungen überhaupt möglich. Analog zu den 16 1-Runden Approximationen mit maximaler Wahrscheinlichkeit, die ich im nächsten Abschnitt vorstellen werde, gibt es auch 16 Einrunden Charakteristiken der diff. Kryptoanalyse, die genau diese Eigenschaft ausnutzen, die Elimination des Übertrags im achten Bit.

Die verwendeten Gleichungen

Es gibt 16 Approximationen der Rundenfunktion mit maximaler Wahrscheinlichkeit, davon 15 nichttrivial, die auf der Kombination der vier folgenden grundlegenden Approximationen beruhen:

$$X[0] + Y[2,8] = K[0] \tag{5.5}$$

$$X[8] + Y[2,8,10,16] = K[0,8] \tag{5.6}$$

$$X[16] + Y[10,18,26] = K[16,24] \tag{5.7}$$

$$X[24] + Y[16,26] = K[24] \tag{5.8}$$

Diese Gleichungen können zu einer 3-Runden Approximation ausgebaut werden, indem die zweite Runde durch die triviale Approximation ($\Gamma_X = \Gamma_Y = \Gamma_K = 0$) überbrückt wird. Der Input der ersten Runde wird mit $X_{L,R}$ bezeichnet und der Ausgang der letzten Runde mit $Y_{L,R}$. Das Whitening wird hier noch weggelassen.

$$X_L[2,8] + X_R[0] + Y_L[2,8] + Y_R[0] = K_1[0] + K_3[0] \tag{5.9}$$

$$X_L[2,8,10,16] + X_R[8] + Y_L[2,8,10,16] + Y_R[8] = K_1[0,8] + K_3[0,8] \tag{5.10}$$

$$X_L[10,18,26] + X_R[16] + Y_L[10,18,26] + Y_R[16] = K_1[16,24] + K_3[16,24] \tag{5.11}$$

$$X_L[16,26] + X_R[24] + Y_L[16,26] + Y_R[24] = K_1[24] + K_3[24] \tag{5.12}$$

Der Angriff

Der Angriff erfolgt in mehreren Phasen:

Phase 1: ($K_1[0 \sim 5; 8 \sim 29]$)

Ich setze die 3-Runden Approximationen (5.9) bis (5.12) von Runde 2–4 und überbrücke die erste Runde durch einen Ausdruck in der F-Funktion für Algorithmus 2. Dadurch wird Gleichung (5.9) bis (5.12) zu:

$$\begin{aligned} &P_L[0,2,8] + P_R[2,8] + C_L[0,2,8] + C_R[0] + F(P_L \oplus P_R, K_1)[0] \\ &\qquad = K_2[0] + K_4[0] + K_6[2,8] = T_1 \end{aligned} \tag{5.13}$$

$$\begin{aligned} &P_L[2,10,16] + P_R[2,8,10,16] + C_L[2,10,16] + C_R[8] + F(P_L \oplus P_R, K_1)[8] \\ &\qquad = K_2[0,8] + K_4[0,8] + K_6[2,8,10,16] = T_2 \end{aligned} \tag{5.14}$$

$$\begin{aligned} &P_L[10,16,18,26] + P_R[10,18,26] + C_L[10,16,18,26] + C_R[16] \\ &+ F(P_L \oplus P_R, K_1)[16] = K_2[16,24] + K_4[16,24] + K_6[10,18,26] = T_3 \end{aligned} \tag{5.15}$$

$$P_L[16,24,26] + P_R[16,26] + C_L[16,24,26] + C_R[24] + F(P_L \oplus P_R, K_1)[24] = K_2[24] + K_4[24] + K_6[16,26] = T_4 \quad (5.16)$$

Die effektiven Schlüsselbits in Gleichung (5.15) sind $K_1[8 \sim 14]$ und $K_1[16 \sim 22]$, wobei $K_1[14]$ und $K_1[22]$ nur als XOR in der Approximation der Rundenfunktion auftreten, so daß diese Terme ebenfalls auf die rechte Seite der Gleichung gebracht werden können. Damit kann also eine vollständige Suche über zwölf Bits erfolgen, die die Werte von $K_1[8 \sim 13]$ und $K_1[16 \sim 21]$ ermittelt. Gleichung (5.14) verschafft mir zusätzlich zu den bereits bekannten Werten die drei Bits $K_1[15;22;23]$, die ich allerdings auch durch Gleichung (5.13) erhalte, deswegen werde ich diese Gleichung nicht für die Analyse benötigen. Gleichung (5.13) gibt mir also die zehn Bits $K_1[0 \sim 5;14;15;22;23]$. Zum Schluß untersuche ich Gleichung (5.16) (sechs Bits $K_1[24 \sim 29]$), sodaß ich am Abschluß von Phase 1 die Schlüsselbits $K_1[0 \sim 5;8 \sim 29]$ kenne.

Phase 2: ($K_2[0 \sim 5;8 \sim 29]$, $K_1[6;30]$)
Da der Angriff ohnehin nur einen zum gesuchten Schlüssel äquivalenten finden kann, nehme ich einfach an, daß die Bits 7 und 31 von K_1 null sind und probiere alle Möglichkeiten für die verbliebenen Schlüsselbits von K_1 durch (zwei Bits). Für jede dieser Möglichkeiten verschlüssele ich alle vorliegenden Klartexte mit dem jeweiligen K_1 und benutze die ersten beiden Runden der Approximationen (5.9) bis (5.12) für Runde 3 und 4. Runde 2 überbrücke ich mit einem Ausdruck in der Rundenfunktion. Damit erhalte ich folgende Gleichungen (mit $X_{L,R}$ der Input in Runde 2 nach der Vertauschung der beiden Hälften):

$$X_L[0] + X_R[2,8] + C_L[2,8] + C_R[2,8] + F(X_R, K_2)[0] = T_5 \quad (5.17)$$

$$X_L[16] + X_R[10,18,26] + C_L[10,18,26] + C_R[10,18,26] + F(X_R, K_2)[16] = T_6 \quad (5.18)$$

$$X_L[24] + X_R[16,26] + C_L[16,26] + C_R[16,26] + F(X_R, K_2)[24] = T_7 \quad (5.19)$$

Die Anwendung von Algorithmus 2 auf die Gleichungen (5.18), (5.17) und (5.19) gibt mir so die Schlüsselbits $K_2[0 \sim 5;8 \sim 29]$.

Phase 3: ($K_4[0 \sim 5;8 \sim 29]$)
Ich setze nun die 3-Runden Approximation von Runde 1–3 und überbrücke die letzte Runde mit der Rundenfunktion. Damit ergibt sich z.B. anhand Gleichung (5.9):

$$P_L[0,2,8] + P_R[0] + C_L[0,2,8] + C_R[2,8] + F(C_L \oplus C_R, K_4)[0] = T_8 \quad (5.20)$$

In derselben Reihenfolge wie in Phase 1 führe ich nun mit den drei Gleichungen Algorithmus 2 durch, was mir schließlich die Schlüsselbits $K_4[0 \sim 5;8 \sim 29]$ verschafft.

Phase 4: ($K_3[0 \sim 5;8 \sim 29]$, $K4[6;30]$)
Indem ich alle Möglichkeiten für die restlichen zwei Schlüsselbits von K_4 durchprobiere (ich nehme wieder Bit 7 und 31 als null an), entschlüssele ich alle Chiffretexte jeweils mit K_4 und benutze analog zu Phase 2 die letzten beiden Runden der 3-Runden Approximation in Runde 1 und 2. Runde 3 überbrücke ich wieder mit einem Ausdruck in der Rundenfunktion. Das gibt mir z.B. anhand Gleichung (5.17) (mit $Y_{L,R}$ der Output von Runde 3 nach der Vertauschung der beiden Hälften):

$$P_L[2,8] + P_R[2,8] + Y_L[2,8] + Y_R[0] + F(Y_L, K_3)[0] = T_9 \quad (5.21)$$

Wie in Phase 2 erhalte ich mittels Algorithmus 2 mit den drei Gleichungen die Schlüsselbits $K_3[0 \sim 5;8 \sim 29]$.

Phase 5: (K_5, K_6, $K_2[6;30], K_3[6;30]$)
Zum Abschluß ver- bzw. entschlüssele ich alle vier Runden für jede Möglichkeit der restlichen Schlüsselbits von K_2 und K_3 (die Bits 7 und 31 werden wieder auf Null gesetzt) und erhalte durch XOR jeweils K_5 und K_6 komplett. Danach überprüfe ich anhand der vorliegenden Paare die korrekte Wahl der fehlenden Bits von K_2 und K_3.

Anmerkung: Aufgrund der Natur der Schlüssel gibt mir das Programm nur mit einer Wahrscheinlichkeit von 1 : 256 den gesuchten Schlüssel zurück, aber immer einen dazu äquivalenten Schlüssel; die Erfolgsrate ist also 100%.

Matsui empfiehlt in [MaYa92] die Durchführung des Algorithmusses mit zehn bekannten Klartexten. Dieses Ergebnis konnte ich nicht bestätigen, mein Programm läuft mit 70–150 Paaren am schnellsten, ca. 5 Sekunden im Durchschnitt auf einem Celeron 450 MHz. Meiner Erfahrung nach sind zehn bekannte Klartexte zuwenig, um möglichst wenige mögliche Schlüsselkandidaten zu erhalten. Mein Programm funktioniert zwar auch mit bis zu 20 Paaren, benötigt dann allerdings schnell über eine Minute, daher empfehle ich eine Anwendung von 50–70 Paaren für einen Angriff. Um den Angriff auch mal in der Praxis erleben zu können, befindet sich auf der CD meine eigene C++ Implementierung des Angriffs zum Ausprobieren (siehe Anhang B). □

5.4 Aufwandsabschätzung für DES und FEAL-N

Die Angabe des Gesamtaufwands einer lin. Kryptoanalyse ist schwieriger als bei der diff. Kryptoanalyse, da die Berechnungskomplexität der lin. Kryptoanalyse nicht mehr unwesentlich ist. Allerdings handelt es sich hierbei nicht nur um Verschlüsselungsoperationen, sondern auch um einfache Operationen wie XOR und AND, die wesentlich schneller durchgeführt werden können. So dauerte die Analyse des 16-Runden DES unter Verwendung von 2^{43} bekannten Klartexten 50 Tage, wobei das Chiffrieren der Klartexte 40 Tage benötigte und 10 Tage die eigentliche Analyse. Dieses Verhältnis ist allerdings sehr stark von dem zu brechenden Chiffrierer und der Berechnungskomplexität des verwendeten Algorithmusses abhängig.

n-Runden DES

Der erste Angriff auf den DES [Mat93] benötigte 2^{47} Klartexte und ermittelte 7 Schlüsselbits mit der linearen Approximation

$$\begin{aligned}&P[12,16,39,50,56] \oplus C[7,18,24,29,47] \oplus F_{16}(C_R, K_{16})[15] =\\&K_1[19,23] \oplus K_3[22] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \oplus\\&K_9[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \oplus K_{15}[22]\end{aligned}$$

unter Verwendung der besten 15-Runden Approximation für die Runden 1–15. Eine zweite Approximation erreichte er durch Anwendung der besten 15-Runden Approximation auf die Runden 2–16, was ihm sechs Bits von K1 brachte. Alle ermittelten Bits sind unterschiedliche Bits des Gesamtschlüssels. Danach wurde eine vollständige Schlüsselsuche für die restlichen $56 - 2 \cdot 7 = 42$ Bits durchgeführt. Die Berechnungskomplexität des Angriffs ist 2^{54} einfache Operationen, zusätzlich kommen noch 2^{42} Verschlüsselungsoperationen für die vollständige Schlüsselsuche und eine Datenkomplexität von 2^{47} bekannten Klartexten hinzu.

In [Mat94] verbessert Matsui seinen Angriff, indem er die beste 14-Runden Approximation vorwärts und rückwärts von Runde 2–15 einsetzt, was ihm je 13 Schlüsselbits bringt, da wieder alle ermittelten Bits verschieden sind. Eine weitere Neuerung ist das *key-ranking*, welches nicht nur die *maximum-likelihood* Lösung von Algorithmus 2 betrachtet, sondern die r wahrscheinlichsten Lösungen ihrer Wahrscheinlichkeit nach ausgibt. Für jede dieser Lösung wird eine vollständige Schlüsselsuche für die restlichen 30 Bits durchgeführt, bis das richtige Ergebnis gefunden wird. Matsui hat umfangreiche Tests durchgeführt, um zu errechnen, wie groß die Erfolgswahrscheinlichkeit des Angriffs ist, wenn man die Anzahl der Klartexte variiert und die Anzahl r der betrachteten Lösungen.

Sein empfohlenes Ergebnis ist eine Durchführung des Angriffs mit 2^{43} Klartexten und Betrachtung der 2^{13} wahrscheinlichsten Lösungen, sodaß der richtige Schlüssel mit einer

Runden	$\lvert\tilde{p}\rvert$	bekannte Klartexte
8	$1.95 \cdot 2^{-9}$	2^{18}
10	$1.22 \cdot 2^{-11}$	2^{23}
12	$1.53 \cdot 2^{-15}$	2^{30}
14	$1.19 \cdot 2^{-17}$	2^{35}
16	$1.19 \cdot 2^{-21}$	2^{43}
18	$1.49 \cdot 2^{-24}$	2^{48}
20	$1.86 \cdot 2^{-26}$	2^{52}

Tabelle 5.4: Aufwand der lin. Kryptoanalyse beim n-Runden DES

Wahrscheinlichkeit von 85% berechnet wird. Mit derart gewähltem r beträgt die Berechnungskomplexität pro Approximation 2^{43} einfache Operationen. Anschließend kommen maximal 2^{43} Verschlüsselungsoperationen für die Schlüsselsuche hinzu. Der Speicherbedarf beider Angriffe ist minimal, jedoch wird Festspeicher für die Klartexte benötigt, beim verbesserten Angriff $2 \cdot 8 \cdot 2^{43} = 2^{47}$ Byte= 128 Terabyte. Hat man Zugang zum Chiffrierer, dann kann man Schritt 1 von Algorithmus 2 direkt mit jedem erzeugten Paar hindurchführen und benötigt so keinen Speicher, allerdings wird dadurch der Angriff zu einem wählbaren Klartextangriff. Da eigentlich nur 13 Bits aus dem Klartext und Chiffretext benötigt werden, könnte man auch von den bekannten Klartexten nur diese Bits behalten, wodurch sich der Speicheraufwand auf 13 Terabyte bei einem bekannten Klartextangriff reduzieren würde.

In Tabelle 5.4 gebe ich einen Überblick über den benötigten Aufwand einer lin. Kryptoanalyse für unterschiedliche Rundenzahlen mit dem verbesserten Angriff. Die besten n-Runden Approximationen für die Angriffe auf $n+2$ Rundensysteme finden sich in [Mat93]. Die Berechnungskomplexität belief sich auf denselben Wert wie die Anzahl der benötigten Klartexte in Verschlüsselungsoperationen und die doppelte Anzahl an einfachen Operationen, deswegen verzichte ich auf diese Werte in der Tabelle.

Nachdem die diff. Kryptoanalyse veröffentlicht wurde, gaben die Designer des DES zu, von diesem Angriff gewußt zu haben und ihn deswegen so definierten, daß er möglichst immun gegen diesen Angriff sei. Nachdem die lin. Kryptoanalyse bekannt wurde, gab D. Coppersmith zu, daß die lin. Kryptoanalyse kein Bestandteil der Designkriterien des DES waren. Das kann entweder heißen, daß sie davon nichts wußten, oder den DES gegen einen noch mächtigeren Angriff schützen mußten; wahrscheinlicher ist eher der erste Fall.

Biham und Shamir hatten in [BiSh93] gezeigt, daß bereits eine Veränderung der Anordnung der S-Boxen den DES in Bezug auf die diff. Kryptoanalyse schwächt. Diesen Fakt nahm Matsui in [Mat94b] genauer unter die Lupe und verglich die Anordnung der S-Boxen mit den entsprechenden Wahrscheinlichkeiten der besten Charakteristiken für beide Angriffe. Er hat dabei herausgefunden, daß die vorliegende Permutation der S-Boxen zu den besten 3% an Permutationen gehört, die maximale Sicherheit des DES gegen diff. Kryptoanalyse garantieren. Die Wahrscheinlichkeiten der Charakteristiken der besseren Permutationen sind nur noch unwesentlich höher als die der ursprünglichen.

Ganz anders sieht es hingegen bei der lin. Kryptoanalyse aus. Die vorliegende Permutation gehört zu den 9% – 16% schlechtesten Permutationen, was die Sicherheit gegen eine lin. Kryptoanalyse betrifft. Unter den 256 besten Permutationen für die lin. Kryptoanalyse hat er sogar eine Permutation gefunden, die besser als die ursprüngliche auch im Hinblick auf die diff. Kryptoanalyse ist. Wäre der DES in der Reihenfolge **24673158** der S-Boxen definiert worden, so hätte eine diff. Kryptoanalyse ca. doppelt so viele wählbare Klartexte erfordert, und gegen eine lin. Kryptoanalyse wäre der DES sogar immun gewesen.

Runden	bekannte Klartexte	Dauer in s
4	10	ca. 1
6	100	ca. 1
7	2^{15}	45
8	2^{25}	3300
12	$1.49 \cdot 2^{29}$	—
16	2^{42}	—
24	$1.37 \cdot 2^{62}$	—

Tabelle 5.5: Aufwand der lin. Kryptoanalyse beim FEAL-N

FEAL-N

Wie bereits erwähnt, ist der FEAL-N besser gegen die lin. Kryptoanalyse gewappnet als der DES. Aufgrund der großen Anzahl an effektiven Schlüsselbits dauerte es einige Jahre, bis endlich die besten Approximationen für den FEAL gefunden werden konnten [AOM96]. Die gefundenen Approximationen sind erheblich besser als die von Biham in [Biha94] verwendete iterative 4-Runden Charakteristik, allerdings ist nicht bekannt, ob sie ebenfalls einer iterativen Charakteristik entstammen. Der nach wie vor beste Angriff auf den FEAL-N benutzt eine $N-1$-Runden Approximation von Runde 1–$N-1$. Die Gleichung enthält etwa 13 effektive Schlüsselbits und 15 effektive Textbits auf der linken Seite der Gleichung (5.2), die Schlüsselbits auf der rechten Seite stammen größtenteils aus dem Schlüssel K_{N+1}, also einem der zwei Schlüssel, die in der Mitte des FEAL zum Klartext addiert werden. Die Schlüsselbits auf der rechten Seite (ein Bit an Information) werden bei dem Angriff auf den FEAL-N ignoriert, da sie leicht komplett bestimmt werden können, wenn erst einmal alle äußeren Runden entfernt worden sind. Von den 13 effektiven Schlüsselbits können neun tatsächlich bestimmt werden, da nicht alle Schlüsselbits von gleicher Bedeutung für das Ergebnis der Approximation sind und einige Bits nicht von ihren komplementierten Werten unterschieden werden können.

Durch einige weitere Hilfsmittel sowie teilweisen Schlüsselsuchen werden einzelne Teilschlüssel der äußeren Runden ermittelt und damit der FEAL-N auf ein System mit immer weniger Runden reduziert. Der Hauptanteil des Berechnungsaufwands liegt dabei stets im ersten Schritt, also der Entfernung der beiden äußersten Runden. Da kein unmittelbarer Zusammenhang zwischen dem Hauptschlüssel des FEAL und den Teilschlüsseln besteht, müssen stets alle Teilschlüssel vollständig ermittelt werden, was die Kryptoanalyse des FEAL sehr aufwendig gestaltet.

Für den FEAL-4,6,7 und FEAL-8 gibt es Angaben über den Aufwand einer kompletten Analyse [MaYa92, AOMM98], für andere Rundenzahlen sind mir nur Aufwandsschätzungen für die erste Angriffsphase bekannt [AOM96]. Alle Werte sind in Tabelle 5.5 aufgeführt. Soweit bekannt, enthält die dritte Spalte Zeitangaben für die komplette Analyse auf einer Sparcstation 10 model 30 mittels C-Programmen. Die Erfolgswahrscheinlichkeit aller Angriffe beträgt 70%, die Erfolgswahrscheinlichkeit kann aber durch Verwendung von doppelt so vielen Klartexten auf 100% erhöht werden.

5.5 Zusammenfassung

Die lineare Kryptoanalyse ist ein Angriff mit bekanntem Klartext. Sie versucht, annähernd lineare Faktoren in den nichtlinearen Komponenten eines Chiffrierers auszunutzen. Diese linearen Faktoren sind Summen aus einzelnen Daten- und Schlüsselbits, die mit einer bestimmten Wahrscheinlichkeit zusammen 0 ergeben. Ist die Summe unter allen möglichen Wertekombinationen häufiger 0, so besteht ein annähernd linearer Zusammenhang zwischen den entsprechenden Bits, bei einer 1 besteht ein annähernd affin linearer Zusammenhang. Beide Extreme können dazu ausgenutzt werden, um mit ausreichend Klartext/Chiffretext-Paaren mittels eines *maximum-likelihood* Tests einzelne Schlüsselbits zu bestimmen.

Um eine lineare Approximation für ein n-Runden Kryptosystem zu erhalten, gibt es einen zur diff. Kryptoanalyse vergleichbaren Begriff einer Charakteristik [Biha94] oder den allgemeineren Begriff einer linearen Hülle [Nyb94] als Analogon zu den Differentialen. In vielen Punkten verhalten sich Charakteristiken beider Analysen ähnlich, oder besser, fast invers zueinander. Dies wird in Abbildung 5.2 besonders deutlich, da die diff. Kryptoanalyse den Informationsfluß „stromabwärts" der F-Funktion beobachtet, also die Entwicklung von Differenzen durch eine Runde, während die lin. Kryptoanalyse den Informationsfluß „stromaufwärts" analysiert. Das liegt daran, daß die beiden Strukturen, die XOR-Operation nach der F-Funktion und die Duplikation des rechten Blockes vor der F-Funktion, sich auf Differenzen und Linearkombinationen genau entgegengesetzt auswirken (siehe Argumentation in Abschnitt 5.2 und [Biha94]). Durch diese Beziehungen kann ein- und derselbe Algorithmus für die Suche nach den besten Charakteristiken für beide Analysen durchgeführt werden, wenn man auf die Unterschiede im Detail Rücksicht nimmt.

Der Aufwand einer diff. Kryptoanalyse wurde durch Differentiale bestimmt, bei der lin. Kryptoanalyse geschieht dies durch lineare Hüllen. Beides sind Familien von Charakteristiken mit gleichen Ein- und Ausgangswerten, aber unterschiedlichen Informationen in den Zwischenwerten. Aufgrund all dieser Gemeinsamkeiten in beiden Analysen spricht Biham von einer „Dualität" der diff. und linearen Kryptoanalyse.

Die Argumentation in Abschnitt 5.4 hat gezeigt, daß der DES nicht sehr sicher in Bezug auf die lineare Kryptoanalyse ist, auch wenn die erforderliche Anzahl an bekannten Klartexten die praktikable Grenze übersteigt. So ist der DES über 20 Runden schneller mittels einer vollständigen Schlüsselsuche zu brechen, während der FEAL ab einer Rundenanzahl von 24 immun gegenüber lin. Kryptoanalyse ist.

5.6 Differentiell-Lineare Kryptoanalyse

Die Differentiell-Lineare Kryptoanalyse wurde 1994 von Langford und Hellmann in [HeLa94] präsentiert. Dieses Verfahren verbindet beide Angriffe, indem es Charakteristiken beider Analysen verwendet, um einzelne Schlüsselbits zu finden. Es ist daher ebenfalls ein Angriff mit wählbarem Klartext.

Wie schon bei der Analyse mit gekürzten und höheren Differenzen ist es bisher nicht möglich gewesen, den Angriff auf mehr als acht Runden auszudehnen. Eigentlich fehlt, um genau zu sein, für beide erweiterten Varianten nur ein statistisches Rahmenwerk, das die entstehenden Probleme hoher Rundenzahlen auffängt. Es wurden allerdings bisher keine weiteren Erkenntnisse veröffentlicht. Im Gegensatz zu den erweiterten Differenzen gibt es allerdings keine Beispiele für Chiffrierer, die erst mit diesem Verfahren brechbar werden, da es ja immer noch auf den grundlegenden Verfahren beruht.

In [HeLa94] wird das Verfahren auf den 8-Runden DES angewendet, und obwohl es den Aufwand nicht verringert, so benötigt es doch wesentlich weniger wählbare Klartexte. Aoki und Ohta übertrugen dieses Verfahren in [AoOh96] auf den FEAL-8, was die erforderliche Anzahl an wählbaren Klartexten auf ganze zwölf Stück reduzierte.

Die Differentiell-Lineare Analyse verwendet stets zwei kurze Charakteristiken beider Analysen, meistens drei bis vier Runden, wobei die differentielle Charakteristik so gewählt wird, daß die Parität der Inputbits der linearen Charakteristik bekannt ist. In der Praxis wird die differentielle Charakteristik so gewählt, daß sich die entsprechenden Bits nie verändert haben, also durch eine Nulldifferenz. Da solche Nulldifferenzen z.B. beim DES nicht lange mit hoher Wahrscheinlichkeit erhalten bleiben, ist die Einschränkung auf kleine Rundenzahlen verständlich.

Die letzte Runde wird wie in Matsuis Algorithmus 2 durch einen Ausdruck der letzten Rundenfunktion überbrückt, so daß der entsprechende Teilschlüssel teilweise bestimmt werden kann. Ebenso muß die erste Runde durch eine teilweise Schlüsselsuche überbrückt werden, um die korrekte Inputdifferenz für die differentielle Charakteristik zu erhalten. Auf diese Weise erhielten Hellmann und Langford beim 8-Runden DES zehn Hauptschlüsselbits, durch Anwendung einer anderen linearen Approximation erhielten sie weitere sechs Schlüsselbits unter Verwendung von 512 wählbaren Klartexten.

Der Angriff von Aoki und Ohta verwendet eine ähnliche Struktur, da es beim FEAL 3-Runden Charakteristiken beider Angriffe mit maximalen Wahrscheinlichkeiten gibt, so daß extrem wenige wählbare Klartexte benötigt werden. Das Problem beim FEAL ist allerdings die Überbrückung der letzten beiden Runden, da die Anzahl an effektiven Schlüsselbits sehr groß ist. Durch raffinierte Optimierungen können sie allerdings diese Anzahl auf 25 Bits verkleinern, was den Angriff praktikabel macht.

Kapitel 6

Designkriterien mit beweisbarer Sicherheit

Ziel dieses Abschnittes ist es, Kriterien für einen Chiffrierer bereitzustellen, die dem Kryptographen Sicherheit gegenüber differentieller und linearer Kryptoanalyse garantieren. Diese Kriterien sind allerdings nur für die Grundform geeignet. Die erweiterten Verfahren sind davon nicht betroffen und benötigen jeweils eigene Abschätzungen, siehe hierzu die entsprechenden Veröffentlichungen [Knud94]. So gibt es für alle Verfahren Beispiele für Chiffrierer, die immun gegen differentielle oder lineare Kryptoanalyse sind, aber anfällig für die erweiterten Verfahren. Je sicherer aber ein Chiffrierer gegen die Grundformen ist, desto schwieriger wird auch ein Angriff mit den erweiterten Verfahren.

6.1 Sicherheit gegen diff. Kryptoanalyse

Theorem 4.5 stellte ein erstes Kriterium für die Durchführbarkeit einer diff. Kryptoanalyse zur Verfügung, das auf der Blockgröße des Chiffrierers beruhte. Eine weitere Schranke stellt die Größe des Schlüsselraums dar, denn wenn die Anzahl der Verschlüsselungsoperationen größer als die Hälfte des Schlüsselraums ist, so ist eine vollständige Suche schneller.

Zunächst einige grundlegende Begriffe und Aussagen aus der Theorie der Markovketten:

Definition 6.1 *Sei eine Markovkette mit Übergangsmatrix $\Pi = [p_{ij}], 1 \leq i, j \leq M$ gegeben. Dann heißt sie*

$$\textbf{irreduzibel} \iff \forall i, j\ \exists n \in \mathbb{N} : p_{ij}^n > 0,$$

sie heißt

$$\textbf{aperiodisch} \iff ggT\Big(n_i = \min_n \{p_{ii}^n > 0\}; 1 \leq i \leq M\Big) = 1.$$

Die Kette besitzt eine **ergodische** *(engl.* steady-state*) Verteilung, wenn es eine von der Anfangsverteilung unabhängige Verteilung $(p_j^*)_{1\leq j\leq M}$ gibt, so daß*

$$\lim_{n\to\infty} p_{ij}^n = p_j^*, \qquad \forall 1 \leq i, j \leq M.$$

Die folgenden Aussagen finden sich in jedem Buch über endliche Markovketten:

1. Eine endliche Markovkette hat eine ergodische Verteilung genau dann, wenn es ein $n > 0$ gibt, so daß Π^n keine Nullen enthält. In diesem Fall heißt Π *primitiv*. Eine Matrix Π ist primitiv genau dann, wenn es ein n_0 gibt, so daß Π^{n_0} eine Spalte ohne Nullen enthält.

2. Eine endliche Markovkette ist irreduzibel genau dann, wenn die Kette genau eine stationäre Verteilung besitzt.

3. Eine endliche Markovkette besitzt eine ergodische Verteilung genau dann, wenn sie irreduzibel und aperiodisch ist. Eine solche Kette nennt man dann auch **ergodisch**.

Die Sicherheit eines Chiffrierers basiert auf dem „Glauben“, daß es ausreicht, eine kryptographisch schwache Funktion nur hinreichend oft zu iterieren, um sie sicher zu machen. Diese Aussagen können wir nun präzisieren.

Theorem 6.1 ([Lai92]) *Sei ein Markovchiffrierer mit Blockgröße m und unabhängigen und gleichverteilten Teilschlüsseln gegeben. Besitzt die unendliche Markovkette $Y_0', Y_1', \ldots$ eine ergodische Verteilung, dann muß diese Verteilung die Gleichverteilung sein, d.h.*

$$\lim_{n\to\infty} P(Y_n' = \beta | Y_0' = \alpha) = \frac{1}{2^m - 1}$$

für alle nichtneutralen Differentiale (α, β). Gilt die Hypothese der stochastischen Äquivalenz für diesen Chiffrierer, dann ist er nach einer ausreichenden Anzahl an Runden sicher gegenüber differentieller Kryptoanalyse.

Beweis: Die Existenz einer ergodischen Verteilung impliziert die Existenz einer eindeutigen stationären Verteilung (Aussage drei und zwei). In Satz 4.1 wurde bewiesen, daß die Gleichverteilung stationär ist. Damit sind beide Behauptungen bewiesen. □

Bemerkung: Wenn das Theorem für einen Markovchiffrierer gilt, so ist die Folge der Differenzen irreduzibel und aperiodisch. Eine hinreichende Bedingung für die Aperiodizität ist die Existenz eines 1-Runden Differentials (α, α) mit einer Wahrscheinlichkeit ungleich Null. Die iterative 2-Runden Charakteristik von Biham und Shamir (siehe Abbildung 4.2) basiert auf einem solchen Differential. Damit ist der DES aperiodisch. Existiert also ein solches Differential, dann ist die wesentliche Bedingung für die Erfüllung des Theorems die Irreduzibilität der Folge von Differenzen. Dieses könnte man z.B. an der Primitivität der Übergangsmatrix zeigen, sofern möglich. □

Ist ein Markovchiffrierer irreduzibel, dann liegt die einzige Schwierigkeit nur noch in einer geeigneten Wahl der Rundenanzahl, so daß alle Einträge der Übergangsmatrix hinreichend klein sind. Lai kann in [Lai92] mittels der Perron-Frobenius Theorie nichtnegativer Matrizen dafür ein Kriterium anhand der Eigenwerte der Übergangsmatrix herleiten.

Diese Theorie besagt, daß nichtnegative Matrizen stets genau einen Eigenwert vom Betrag 1 und weitere Eigenwerte strikt kleiner als Betrag 1 besitzen. Bei Markovchiffrierern ist der größte Eigenwert 1 zum Eigenvektor $(\frac{1}{2^m-1}, \ldots, \frac{1}{2^m-1})$ der Gleichverteilung. Da die Gleichverteilung die ergodische Verteilung ist, nähern sich mit wachsendem r alle Einträge von Π^r $\frac{1}{2^m-1}$ und sie tuen dies laut Lai sogar exponentiell. Angenommen, λ_2 ist der zweitgrößte Eigenwert mit Vielfachheit t_0, und alle weiteren Eigenwerte haben Vielfachheiten größergleich t_0, dann gibt es eine Konstante $a > 0$, so daß für alle (α, β) gilt:

$$P(Y' = \beta | X' = \alpha) - \frac{1}{2^m - 1} \leq a r^{t_0 - 1} |\lambda_2|^r.$$

Aus dieser Ungleichung und Ungleichung (4.3) kann unter Berücksichtigung des Fakts, daß die Datenkomplexität nach oben durch 2^m beschränkt ist, mit dem kleinsten r_0, so daß die Abschätzung

$$a r_0^{t_0 - 1} |\lambda_2|^{r_0} \leq 2^{m-1}$$

erfüllt ist, die minimale Rundenzahl gewonnen werden, so daß der Chiffrierer sicher gegen eine diff. Kryptoanalyse ist.

Eli Biham stellte im Zuge der Ausschreibung um den AES die Empfehlung auf, als Rundenzahl die doppelte Rundenzahl zu wählen, die den Chiffrierer immun gegen differentielle und lineare Kryptoanalyse macht, um auch gegen zukünftige Angriffe gewappnet zu sein. Diese Hypothese ist allerdings in der Fachwelt heftig umstritten, vor allem aufgrund der geringeren Verschlüsselungsrate, obwohl mehr Runden auf jeden Fall eine größere Sicherheit bieten.

6.2 Sicherheit gegen lin. Kryptoanalyse

Um ähnliche Aussagen über die Sicherheit eines Chiffrierer gegens eine lin. Kryptoanalyse zu gewinnen, wurde von O'Connor und Golic in [GoOC94] versucht, auch die lin. Kryptoanalyse in Verbindung mit Markovketten zu bringen. Da beide Angriffe strukturell sehr ähnlich sind, ist ihnen dies auch gelungen.

Definition 6.2 *Für eine Boolesche Funktion $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ sei ihr reelwertiges Pendant $\hat{f} : \mathbb{Z}_2^n \to \{-1, 1\}$ gegeben durch $\hat{f}(x) := (-1)^{f(x)}$. Für diese Funktionen sei ein* **inneres Produkt** *definiert durch*

$$< \hat{f}(x), \hat{g}(x) >:= \sum_{x \in \mathbb{Z}_2^n} \hat{f}(x)\hat{g}(x).$$

Definition 6.3 *Seien $f, g : \mathbb{Z}_2^n \to \mathbb{Z}_2$ Boolesche Funktionen. Dann ist der* **Korrelationskoeffizient c(f,g)** *gegeben durch*

$$c(f,g) := 2^{-n} \sum_{x \in \mathbb{Z}_2^n} \hat{f}(x)\hat{g}(x) = P(f = g) - P(f \neq g) = 2P(f = g) - 1.$$

Die lin. Kryptoanalyse beruht auf Gleichungen der Form $\Gamma_P P + \Gamma_C C = \Gamma_K K$, sodaß mit einer Wahrscheinlichkeit p $\Gamma_K K = 0$. Solche Gleichungen können für jede Runde erstellt und zu einer Gleichung für das ganze System zusammengefügt werden.

Sei $Y_0 = \Gamma_P P$ und $Y_r = \Gamma_C C$ für ein r-Runden Kryptosystem. Y_i kennzeichne die Summe der Inputbits in Runde $i+1$, die für die Approximation in Runde $i+1$ benötigt werden. Y_i ist gleichzeitig auch die Summe der Outputbits von Runde i, damit alle mittleren Terme sich gegenseitig wegkürzen und nur eine Summe von Klar- und Chiffretextbits übrig bleibt. Die Approximation in Runde i, $1 \leq i \leq r$ ist also von der Form $Y_{i-1} + Y_i = \Gamma K_i$ mit Wahrscheinlichkeit p_i. Die Wahrscheinlichkeit der r-Runden Approximation ist dann mit Matsuis Piling-Up Lemma gegeben durch $p = \frac{1}{2} + 2^{r-1} \prod_{i=1}^{r} \tilde{p}_i$.

Faßt man die Y_i als Boolesche Funktionen auf (die lineare Funktion $f(x) = Y_i \bullet x$), so ist die Wahrscheinlichkeit p_i jetzt mit Korrelationskoeffizienten beschreibbar, es gilt nämlich

$$\tilde{p}_i = \frac{c(Y_{i-1}, Y_i)}{2}.$$

Auf die gesamte r-Runden Approximation erweitert, ergibt sich für p die Beziehung

$$p = \frac{1}{2} + \frac{1}{2} \prod_{i=1}^{r} c(Y_{i-1}, Y_i).$$

Mit diesen Vorbereitungen kann ich jetzt eine Markovkette für die lineare Kryptoanalyse definieren.

Theorem 6.2 ([GoOC94]) *Sei $F : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ die Rundenfunktion des Chiffrierers. Sei $N = 2^n - 1$ und X ein Inputvektor.*

Dann bilden die Y_k, aufgefaßt als Zufallsvariablen, eine homogene Markovkette mit der $N \times N$-Matrix $P = [p_{ij}]_{1 \leq i,j \leq N}$ als Übergangsmatrix, wobei

$$p_{ij} = c(i \bullet X, j \bullet F(X))^2.$$

Die Einträge der Übergangsmatrix sind also die Quadrate aller möglichen Korrelationskoeffizienten, in [GoOC94] wird die Matrix daher auch als Korrelationsmatrix bezeichnet. Korrelationsmatrizen werden in [DGV95] genauer untersucht, insbesondere wird der Zusammenhang mit der Walsh-Hadamard Transformation herausgestellt, was zu der Bemerkung führt, daß die lineare und diff. Kryptoanalyse sich im „Zeit — Frequenzbereich" entsprechen.

Bemerkung: Um die Darstellung übersichtlich zu gestalten, gehe ich auf die Walsh-Hadamard Transformation nur kurz ein: Die Walshtransformierte einer Booleschen Funktion $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ ist gegeben durch

$$F(w) = \sum_{x \in \mathbb{Z}_2^n} \hat{f}(x) \hat{L}_w(x)$$

mit $\hat{L}_w(x) := (-1)^{w \bullet x}$ linear. Die linearen Funktionen bilden eine orthogonale Basis für den Raum der reelwertigen Booleschen Funktionen und damit auch für den Raum der walshtransformierten Funktionen, da bzgl. des bestehenden inneren Produktes

$$< (-1)^{u \bullet x}, (-1)^{v \bullet x} >= 2^n \delta(u+v),$$

wobei $\delta(w) = 1$, wenn $w = 0$ und 0 sonst. Aus dieser Gleichung folgt auch dieselbe Beziehung für das innere Produkt der wahlshtransformierten. □

Theorem 6.3 (Vollständigkeitsrelation) *Für die Walshtransformation gilt die Vollständigkeitsrelation oder* **Parsevalsche Gleichung**

$$\sum_w F(w)^2 = 2^n \sum_x \hat{f}(x)^2 = 2^{2n}, \quad \textit{für alle } f : \mathbb{Z}_2^n \to \mathbb{Z}_2.$$

Anmerkung: Das erste Gleichheitszeichen ist die eigentliche Parsevalsche Gleichung. Das zweite Gleichheitszeichen gilt stets für reelwertige Boolesche Funktionen, wie man sofort sieht. □

Satz 6.4 ([GoOC94]) *Die Matrix P ist doppelstochastisch.*

Beweis: Jede Zeile von P gibt die totale Korrelation zwischen der Funktion $f = j \bullet F$ und der Menge aller linearen Funktionen an. Nach dem vorigen Theorem gilt $\sum_{L \text{ lin.}} c(f,L)^2 = 1$, da $c(f, L_w) = 2^{-n} F(w)$ und damit ist die Behauptung für die Zeilensumme bewiesen. Die Behauptung für die Spaltensumme folgt wieder aus der Bijektivität der Rundenfunktion, da dessen Inverse P^T als Übergangsmatrix besitzt. □

Mit dieser Markovkette kann nun mit denselben Mitteln wie aus dem vorigen Abschnitt untersucht werden, ob die Kette irreduzibel und aperiodisch ist. Ist die Kette somit ergodisch, kann wieder die Immunität des Chiffrierers nach einer ausreichenden Rundenzahl angenommen werden, da die Gleichverteilung wieder stationär und damit die ergodische Verteilung ist.

6.3 Neue Kriterien für Chiffrierer

Wie bereits erwähnt, hat die Entdeckung der differentiellen und linearen Kryptoanalyse die Anforderungen an das Design von Blockchiffrierern wesentlich erweitert. Ziel dieses Abschnittes ist es, kurz auf einige neue Designkriterien einzugehen, die beweisbare Sicherheit gegen diese Angriffe garantieren.

6.3.1 Größe und Art von S-Boxen

S-Boxen sind die wichtigsten Bestandteile eines Chiffrierers. Sie sind normalerweise die einzigen nichtlinearen Komponenten eines Chiffrierers, also der Dreh- und Angelpunkt für die Sicherheit. Diese S-Boxen werden meistens durch *look-up tables* realisiert, also die Beschreibung der Booleschen Funktion $S : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ durch eine Wertetabelle. Der Vorteil dieser Methode ist die einfache und vor allem extrem schnelle Realisierung von komplexen Funktionen mit möglichst hoher Nichtlinearität. Der Nachteil ist die bereits in Abschnitt 1.3 angesprochene Größe einer solchen Tabelle, die $2^n \cdot m$ Bits erfordert.

Die Sicherheit eines Chiffrierers ist wesentlich von den Eigenschaften der S-Boxen abhängig. In der diff. und linearen Kryptoanalyse wurde diese Sicherheit durch Differenzenverteilungs- und lineare Approximationstabellen bestimmt, oder besser, durch die maximalen Einträge dieser Tabellen. Die beiden wichtigsten Werte von S-Boxen sind in der folgenden Definition zusammengefaßt:

Definition 6.4 *Sei $S : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ eine Boolesche Funktion und $x', \Gamma_x \in \mathbb{Z}_2^n$, $y', \Gamma_y \in \mathbb{Z}_2^m$. Dann sei*

$$\begin{aligned} DP^S_{\max} &:= \max_{x' \neq 0, y'} DP^S(x' \to y') \\ LP^S_{\max} &:= \max_{\Gamma_x, \Gamma_y \neq 0} LP^S(\Gamma_x \to \Gamma_y) \end{aligned}$$

wobei

$$\begin{aligned} DP^S(x' \to y') &:= \frac{\#\{x \in \mathbb{Z}_2^n | S(x) \otimes S(x \otimes x'^{-1})^{-1} = y'\}}{2^n}, \\ LP^S(\Gamma_x \to \Gamma_y) &:= \left(2 \frac{\#\{x \in \mathbb{Z}_2^n | x \bullet \Gamma_x = S(x) \bullet \Gamma_y\}}{2^n} - 1 \right)^2. \end{aligned}$$

Die zuerst etwas untypisch wirkende Definition von LP^S entspricht den Korrelationskoeffizienten des vorigen Abschnittes und erlaubte mir eine mathematisch einheitlichere Analyse beider Verfahren. Mit diesen Werten eng verwandt sind die Begriffe der *(Nicht-)Linearität*, die ich nun definieren werde und so etwas wie den Abstand zur nächsten affinen Funktion angeben:

Definition 6.5 *Die* **Linearität** *und* **Nichtlinearität** *einer Booleschen Funktion $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$ ist gegeben durch*

$$\mathcal{L}(f) := \max_{L \text{ lin.}} |c(f, L)|$$

$$\mathcal{N}(f) := \min_{A \text{ affin}} \#\{x \in \mathbb{Z}_2^n | f(x) \neq A(x)\} = 2^{n-1} - 2^{n-1} \mathcal{L}(f)$$

Aus der Parsevalschen Gleichung kann man leicht folgern, daß für die Linearität $2^{-n/2} \leq \mathcal{L}(f) \leq 1$ gelten muß. Um die definierten Werte auf vektorwertige Funktionen auszudehnen, wird entsprechend das Maximum bzw. Minimum über alle möglichen Linearkombinationen von Outputkomponenten betrachtet:

Definition 6.6 *Sei $S : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ eine vektorwertige boolesche Funktion. Dann ist*

$$\mathcal{L}(S) := \max_{b \neq 0} \mathcal{L}(b \bullet S)$$

$$\mathcal{N}(S) := \min_{b \neq 0} \mathcal{N}(b \bullet S).$$

Neben diesen Werten ist der minimale Grad des Polynoms von Bedeutung, der die S-Box möglichst vollständig beschreiben kann. Denn je niedriger der Grad dieses Polynoms ist, desto besser lassen sich Angriffe algebraischer Natur ansetzen, die niedrige Ordnungen ausnutzen. Angriffe dieser Art sind z.B. die Lineare Kryptoanalyse mit quadratischen und höheren Approximationen [SK97] oder die diff. Kryptoanalyse mit Differenzen höherer Ordnung ([Knud94] und Abschnitt 4.7).

Definition 6.7 *Die Ordnung einer Funktion $F : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ ist der minimale Grad der Polynomialform aller möglichen Kombinationen von Outputkomponenten von F.*

$$O(F) = \min_{b \neq 0} \operatorname{grad}(b \bullet F)$$

Nach der Veröffentlichung der diff. Kryptoanalyse wurde von vielen Kryptographen nach geeigneten Funktionsklassen für S-Boxen gesucht. Um die Angriffsfläche für differentielle Angriffe zu minimieren, wurde hauptsächlich nach Funktionen gesucht, die alle Outputdifferenzen gleich wahrscheinlich machen. Diese Klasse von Funktionen wird allgemeinhin als **differentially-uniform mappings** bezeichnet. Bei geeigneter Struktur der Rundenfunktion läßt sich so sehr einfach die Wahrscheinlichkeit für das maximale Differential bei gegebener Rundenzahl kontrollieren. Funktionen mit dieser Eigenschaft sind z.B. die sogenannten **fast perfekt nichtlinearen Funktionen** (siehe z.B. [Nyb90, Nyb91]).

Definition 6.8 *Eine Funktion $S : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ heißt* **differentiell δ-uniform***, wenn $2^n \cdot DP^S_{\max} \leq \delta$.*

Von besonderem Interesse sind natürlich Funktionen mit minimalem δ. Ist $m = n$, so sollten alle Einträge in der Differenzenverteilungstabelle nur 0 oder 2 sein. Es gibt eine ganze Reihe an geeigneten Bijektionen, die *differentiell-2-uniform* sind [Nyb93], die Beispiele aus [Nyb93] sind allerdings nicht unbedingt für die Definition eines Chiffrierers geeignet. So besitzt z.B. die Funktion x^{-1} hervorragende Eigenschaften, was die Werte von DP^S,LP^S und die Ordnung betrifft. Eine Beschreibung eines solchen Chiffrierers durch eine Gleichung ist jedoch nicht sehr komplex, und wird höchstwahrscheinlich durch eine algebraische Herangehensweise leicht brechbar sein.

Definition 6.9 *Eine Boolesche Funktion $f : \mathbb{Z}_2^n \to \{-1, 1\}$ ist genau dann* **perfekt nichtlinear** *oder* **bent***, wenn $|F(w)|$ konstant ist für alle w. Da $\sum_x f(x)^2 = 2^n$, gilt $|F(w)| = 2^{n/2}$.*

Bemerkung: Perfekt nichtlineare Funktionen kann es nur für gerade n geben (siehe z.B. [Nyb95]). Sie bieten den maximalen Abstand zu linearen Funktionen, d.h ihre Linearitäten $\mathcal{L}(f)$ erreichen die untere Schranke $2^{-n/2}$. Außerdem sind sie hochgradig korrelationsimmun, eine wichtige Eigenschaft für Pseudozufallszahlengeneratoren von Stromchiffrierern. □

Für Blockchiffrierer müssen diese Begriffe noch auf vektorwertige Funktionen ausgedehnt werden. Demnach heißt eine vektorwertige Funktion $F : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ **perfekt nichtlinear**, wenn ihre Linearität $\mathcal{L}(F)$ den minimalen Wert $2^{-n/2}$ erreicht. Wie schon bei den booleschen Funktionen kann diese Grenze nur für gerade n erreicht werden, zusätzlich muß jedoch auch $n \geq 2m$ gelten, was schlecht für die lineare Kryptoanalyse sein kann, wie ich später zeigen werde.

Die besten Eigenschaften in Hinsicht auf Sicherheit gegen beide Analysen besitzen die sogenannten *almost bent functions* oder *fast perfekt nichtlinearen Funktionen*.

Definition 6.10 *Eine Funktion $F : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ heißt* **fast perfekt nichtlinear***, wenn ihre Linearität $\mathcal{L}(f)$ $2^{-\frac{n-1}{2}}$ beträgt.*

Bemerkung: Solche Funktionen existieren für $m = n$ ungerade und besitzen neben der immer noch guten differentiellen Gleichförmigkeit auch Korrelationsmatrizen mit hervorragenden Eigenschaften, d.h. die Einträge bestehen nur aus Nullen oder $2^{-\frac{n-1}{2}}$. Eine Funktion mit dieser Eigenschaft ist z.B. $f(x) = x^{2k+1}$ in $\mathbb{Z}_{2^n}$ mit n ungerade und $\mathrm{ggT}(k,n) = 1$. □

Neben diesen Funktionen gibt es aber eine wesentlich einfachere Art, die Wahrscheinlichkeit von Outputdifferenzen so klein wie möglich zu machen: durch geeignete Wahl von $m > n$. Ist der Bildraum größer als der Werteraum, so folgt mit hoher Wahrscheinlichkeit, daß eine zufällig gewählte Funktion *differentiell-2-uniform* ist. Es gibt eine ganze Reihe an Chiffrierern, die solche S-Boxen benutzen, z.B. die CAST-Familie von Blockchiffrierern [Ada97].

Die lineare Kryptoanalyse setzt dieser einfachen Möglichkeit allerdings wieder einen Dämpfer auf. Biham untersuchte in [Biha94] die linearen Eigenschaften solcher S-Boxen und konnte leicht zeigen, daß eine zu starke Vergrößerung des Bildraumes lineare Faktoren nach sich ziehen **muß**. Sofort einsichtig wird das aus der Betrachtung der unterschiedlichen Informationsflüsse von diff. und linearer Kryptoanalyse. Da die lin. Kryptoanalyse die Veränderung von $\Gamma_y \to \Gamma_x$ betrachtet, bewirkt eine Vergrößerung des Bildraumes dasselbe wie eine Wahl von $n > m$ für die Differenzenverteilungstabelle: größere durchschnittliche Einträge in der Tabelle.

Um diese Aussage zu präzisieren, fasse ich eine S-Box als Matrix mit 2^n Spalten und $n + m$ Zeilen auf. Die Spalten korrespondieren mit allen möglichen Inputs der S-Box und die Zeilen der Matrix geben für jedes Input- und Outputbit den Wert dieses Bits bei dem jeweiligen Input bzw. Output der S-Box an. Eine lineare Approximation gemäß Gleichung (5.1) ist danach eine Linearkombination der entsprechenden Zeilen dieser Matrix.

Definition 6.11 *Sei eine S-Box $S : \mathbb{Z}_2^n \to \mathbb{Z}_2^m$ gegeben. Dann sei*

$$\mathcal{M} := \begin{pmatrix} 0[0] & 1[0] & \dots & 2^n-1[0] \\ 0[1] & 1[1] & \dots & 2^n-1[1] \\ \dots & & & \dots \\ 0[n-1] & 1[n-1] & \dots & 2^n-1[n-1] \\ S(0)[0] & S(1)[0] & \dots & S(2^n-1)[0] \\ \dots & & & \dots \\ S(0)[m-1] & S(1)[m-1] & \dots & S(2^n-1)[m-1] \end{pmatrix}.$$

Zeilen von Input- und Outputbits dieser Matrix bilden eine **affine (lineare) Kombination***, wenn die Spaltensumme dieser Zeilen über alle Spalten stets 1 (0) ist.*

Sei $\mathcal{M}' := [\mathcal{M}|1]$ die Matrix, die sich durch Hinzufügen einer Spalte aus Einsen zu $\mathcal{M}$ ergibt.

Ist der Rang von $\mathcal{M}'$ $m+n$, so gibt es keine lineare Kombination in der S-Box, ist der Rang sogar $m+n+1$, so gibt es auch keine affine Kombinationen der S-Box. Ist der Rang kleiner, so muß es eine affine oder lineare Kombination geben.

Der maximale Rang ist durch die Spaltenzahl auf 2^n begrenzt. Gilt also $m \geq 2^n - n$, so *muß* die S-Box eine affine Kombination besitzen, also gibt es eine Approximation mit maximaler Wahrscheinlichkeit $\frac{1}{2} \pm \frac{1}{2}$. Gilt sogar $m \geq 2^n$, so muß es eine affine Kombination geben, die nur aus Outputbits besteht, also gar nicht mehr vom Input abhängig ist. Dies resultiert in Approximationen von der Form $0 \to X$ mit Wahrscheinlichkeit $\frac{1}{2} \pm \frac{1}{2}$, so daß ein Angriff nur noch wenige Klartexte benötigt.

Wie bereits erwähnt, ist der einfachste Weg, um schnell kryptographisch gute S-Boxen zu erhalten, die zufällige Generierung dieser. Es gibt hierzu Aussagen über die erwartete Linearität und Ordnung solcher Funktionen, die mit garantieren, daß bei ausreichend großem n,m praktisch alle Funktionen sehr hohe Sicherheit gegen diff. und lin. Kryptoanalyse bieten (siehe z.B. [LHT97]).

6.3.2 Schlüsselauswahl

Alle Chiffriersysteme erzeugen aus einem Hauptschlüssel einen erweiterten Schlüssel, der die einzelnen Rundenschlüssel stellt. Die Methode der Erzeugung dieser Schlüssel ist nicht ganz unbedeutend für die Sicherheit eines Chiffrierers. Der zweite Punkt ist die Art, mit der das Schlüsselmaterial mit dem Klartext verknüpft wird. Beide Aspekte haben Konsequenzen auf die Sicherheit und sind deswegen nach den S-Boxen das wichtigste Designkriterium für ein Kryptosystem.

Die einfachste Schlüsselauswahlfunktion besteht aus der Auswahl einer Teilmenge an Hauptschlüsselbits mit anschließender zyklischer Verschiebung des Hauptschlüssels um einen bestimmten Betrag. Dies resultiert in einem direkten Zusammenhang zwischen Rundenschlüsseln und Hauptschlüsselbits. In [Biha93] beschreibt Biham einen Angriff, der eine solche Schlüsselauswahlfunktion angreifen kann. Obwohl der DES auch seine Schlüssel auf diese Weise generiert, bringt dieser Angriff hier keinen Vorteil gegenüber vollständiger Schlüsselsuche. Auch hier wußten die Designer des DES von diesem Angriff und schützten den DES davor. Ein modifzierter DES, der nach jeder Runde einheitlich um eine Stelle shiftet, ist jedoch angreifbar. Dies zeigt, wie eine subtile Änderung, wie z.B. eine unsymmetrische Shiftoperation beim DES, solche Angriffe unterbinden kann. Biham's Angriff wurde in [KSW96] erweitert und mögliche Schutzmethoden präsentiert.

Diese einfache Methode hat allerdings noch einen weiteren Nachteil im Zusammenhang mit diff. und lin. Kryptoanalyse: Durch die Kenntnis einzelner Hauptschlüsselbits ist es nicht mehr nötig, alle Rundenschlüssel zu ermitteln, sondern nur noch eine ausreichende Anzahl, so daß der Rest per vollständiger Suche in kurzer Zeit gefunden werden kann. Wie man in Abschnitt 5.3 sieht, ist eine Analyse wesentlich aufwendiger, wenn erst alle Rundenschlüssel ermittelt werden müssen.

Neben diesen Nachteilen sollte allerdings der nicht unwesentliche Vorteil einer einfachen Schlüsselauswahl herausgestellt werden: Es ist nicht nötig, erst alle Rundenschlüssel zu generieren, um mit der Verschlüsselung zu beginnen, sondern können bei Verwendung erzeugt werden. Je nach Verknüpfung mit dem Klartext ist es sogar eventuell möglich, auch bei der Entschlüsselung den Schlüssel erst bei Gebrauch zu erzeugen, was den verwendeten Speicher bei Ent- bzw. Verschlüsselung stark reduziert. Dies ist vor allem in Smartcard Anwendungen wichtig, da sie nur über wenige Bytes an RAM verfügen.

Die Art der Schlüsselerzeugung kann unter Umständen die Existenz **schwacher** Schlüssel nach sich ziehen. Schwache Schlüssel gibt es in zwei Varianten:

- Schwache Schlüssel im Sinne derer des DES,
- Schlüssel, die die Wahrscheinlichkeit von Charakteristiken erhöhen.

Im ersten Fall heben sich die erzeugten Rundenschlüssel gegenseitig wieder auf, was eine große Anzahl an Fixpunkten der Verschlüsselung nach sich zieht. Solange die Anzahl solcher Schlüssel gering ist, stellt dies keine erhebliche Schwächung des Chiffrierers dar, da sie nicht ohne weiteres erkannt werden können.

Der zweite Fall ist wesentlich gefährlicher. Die Hypothese der stochastischen Äquivalenz (siehe Kapitel 4 und 5) fordert von einem Chiffrierer, daß die Qualität der Verschlüsselung nicht vom Schlüssel abhängig sein darf, da sonst Charakteristiken beider Analysen bei bestimmten Schlüsseln wahrscheinlicher werden könnten. Für den Blockchiffrierer IDEA z.B. existieren Teilmengen der Größe 2^{51} des Hauptschlüssels, so daß ein Differential mit höherer Wahrscheinlichkeit existiert. Ob ein solcher Schlüssel verwendet wurde, kann leicht anhand weniger bekannter Klartexte überprüft werden und damit die Schlüsselsuche erheblich eingegrenzt werden. Allerdings ist diese Teilmenge angesichts von 128 Bit Schlüssellänge verschwindend klein und die Überprüfung funktioniert natürlich auch für den Sender, der somit die Verwendung eines solchen Schlüssels verhindern kann.

Gefährlich werden solche schwachen Schlüssel bei der Verwendung eines Blockchiffrierers als Hashfunktion, da durch entsprechende Wahl des Schlüssels leichter eine Kollision verursacht werden kann, Details hierzu in [Pren93].

Die Existenz solcher schwachen Schlüssel ist meistens auf die Art der Verknüpfung mit dem Klartext zurückzuführen. Es hat sich gezeigt ([DGV95]), daß die einfachste Art der Verknüpfung, das XOR mit dem Klartext vor den S-Boxen, die beste Wahl ist. Ist jedes Outputbit der S-Box auch noch gleichabhängig von jedem Inputbit, so garantiert dies einen gleichförmigen Einfluß des Schlüssels auf das Ergebnis der Rundenfunktion und läßt die Hypothese der stochastischen Äquivalenz eher erfüllt sein.

Diese Gleichförmigkeit wird auch als *schwache Schlüsselabhängigkeit* bezeichnet, im Gegensatz zur *starken Schlüsselabhängigkeit*, die bei manchen Blockchiffrierern sogar als Designkriterium angeführt wird. Der Vorteil liegt auf der Hand: Schwankt die Wahrscheinlichkeit einer Charakteristik mit jedem Schlüssel, so wird eine diff. und lin. Kryptoanalyse schwieriger, da keine endgültigen Aussagen über die Anzahl der zu verwendenden Klartexte gemacht werden kann. Andererseits könnte aber die durchschnittliche Wahrscheinlichkeit zwar gut sein, für bestimmte Schlüssel aber sehr schlecht. Können diese Schlüssel erkannt werden, so besitzt der Chiffrierer die oben genannten Probleme. Um die Sicherheit des Chiffrierers insgesamt abschätzen zu können, ist die schwache Abhängigkeit sinnvoller, da eine geringe Varianz in den Übergangsmatrizen beider Angriffe von Schlüssel zu Schlüssel die Erfolgsgarantie für einen Angriff leichter kontrollierbar macht.

6.3.3 Konstruktion der Rundenfunktion

Die am weitesten verbreitete Konstruktion einer Rundenfunktion habe ich bereits vorgestellt: die Feistelchiffre (Abschnitt 1.4). Dieses Verfahren wird auch heute noch in vielen Chiffrierern verwendet, da es gründlich erforscht und seine Eigenschaften dadurch bestens bekannt sind. Die Konstruktion von Feistelchiffren wurde von Schneier und Kelsey zu sogenannten **ungleichmäßigen Feistelnetzwerken** [KeSch96] erweitert.

In ungleichmäßigen Feistelchiffren wird nicht, wie sonst üblich, der Datenblock halbiert, sondern in beliebigen Verhältnissen aufgeteilt. Dies erlaubt z.B. die Erweiterung des Schemas auf 128-Bit Klartextblöcke unter Verwendung einer guten Rundenfunktion, die nur 32-Bit Blöcke verarbeiten kann. Das Schema für die Erweiterung eines Feistelnetzwerks von zwei Hälften bis auf vier Viertel ist in Abbildung 6.1 dargestellt. Der Nachteil des erweiterten Vorgehens ist die Notwendigkeit einer eigenen Verschlüsselungsfunktion, da es nicht mehr selbstinvers ist. Der CAST-256, der das rechte Schema in Abbildung 6.1 verwendet, umgeht dies, indem in der ersten Phase der Verschlüsselung rechtsgeshiftet wird und in der zweiten Phase links; das vergrößert dennoch den benötigten Programmcode.

Die beste differentielle Charakteristik für den Angriff auf eine Feistelchiffre ist eine iterative 2-Runden Charakteristik, die eine Inputdifferenz ungleich Null zu Null werden läßt, sofern dies möglich ist. Damit erhält man eine Charakteristik, die mit Wahrscheinlichkeit $1 \cdot p$ gilt. Dies erweitert sich auf eine iterative 4-Runden Charakteristik im rechten Beispiel von Abbildung 6.1, wo in einer Runde eine Inputdifferenz ungleich Null zu Null wird und in den restlichen drei Runden die triviale Charakteristik zum Zuge kommt. Dies resultiert dann in einer Charakteristik mit Wahrscheinlichkeit $1 \cdot 1 \cdot 1 \cdot p$.

Zusammenfassend kann man sagen, daß die Sicherheit eines Feistelnetzwerks stets von der besten Charakteristik eines vollen Zyklus abhängig ist. Einen Zyklus nenne ich hier die Anzahl der Runden, bis jeder Block mindestens einmal die Rolle des rechten und linken Blockes eines klassischen Feistelnetzwerks angenommen hat. In Abbildung 6.1 wurde jeweils ein kompletter Zyklus betrachtet. [KeSch96] untersucht alle möglichen Arten von Kombinationsmöglichkeiten und gibt Abschätzungen für die bestmöglichen Charakteristiken der diff. und lin. Kryptoanalyse.

Neben dieser Rahmenstruktur ist die Gestalt der eigentlichen Rundenfunktion F von höherer Bedeutung. Das Feistel-Schema wurde hauptsächlich deshalb erfunden, um die Rundenfunktion einfacher und kleiner zu gestalten, da in den 70er Jahren es technisch nicht möglich war, kryptographisch gute Funktionen auf 64-Bit Blöcken zu entwerfen.

Die Rundenfunktion eines Blockchiffrierers ist immer eine Produktchiffre, da große S-Boxen nicht machbar sind. Sie besteht fast immer aus irgendwelchen Operationen rund

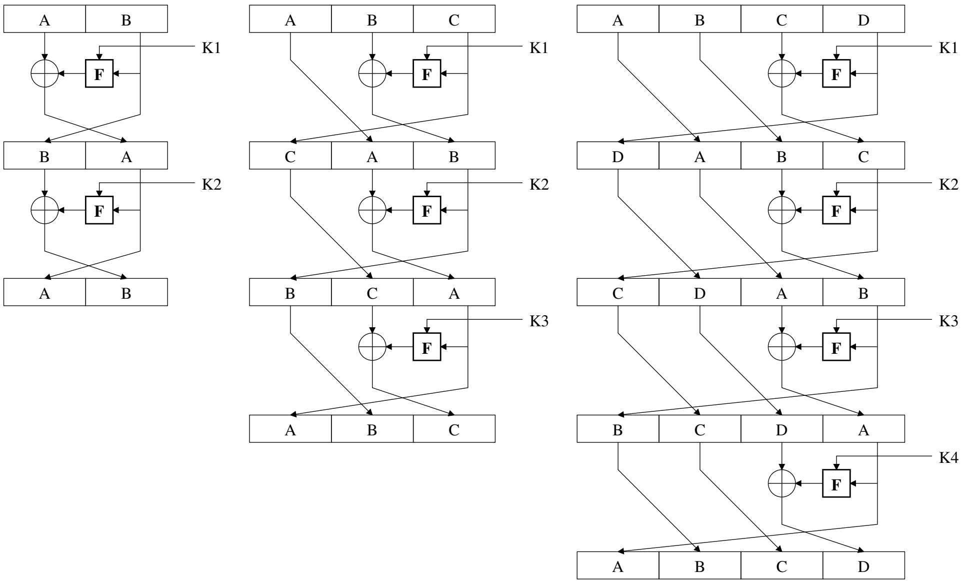

Abbildung 6.1: Ungleichmäßige Feistelnetzwerke

um die zentralen S-Boxen, wobei diese Operationen normalerweise Expansionen oder Permutationen sind. Dieser Typus von Produktchiffren wird SPN (Substitution-Permutation Networks) genannt, es gibt aber auch andere Ansätze für die Konstruktion einer Produktchiffre[1].

Nyberg hat in [Nyb95] die Auswirkung von verschiedenen Operationen auf die (Nicht)-Linearität der Rundenfunktion gegenüber den S-Boxen bei SPN's betrachtet. Beim DES wurde um die S-Boxen herum nur zwei lineare Funktionen, die Expansion und Permutation, konstruiert. Lineare Funktionen verkleinern nicht diese Schlüsselwerte der S-Boxen (sofern der Bildraum nicht verkleinert wird), wie Nyberg gezeigt hat, sodaß von diesen Funktionen keine Gefahr ausgeht.

Matsui betrachtet in [Mat96] Rundenfunktionen, die die Sicherheit gegen diff. und lin. Kryptoanalyse noch stärken können. Seine Haupterkenntnis ist, daß eine serielle Anordnung von S-Boxen, verbunden mit der Addition von Schlüsselmaterial dazwischen, die Sicherheit signifikant stärken kann. Dieses Ergebnis war aber auch zu erwarten, da sich die entsprechenden Wahrscheinlichkeiten für $DP_{\max}$ und $LP_{\max}$ multiplizieren, weil diese Anordnung praktisch zwei Runden entspricht.

Eine rein parallele Anordnung von S-Boxen ist deshalb nur zu empfehlen, wenn anschließend eine gründliche Permutation der Outputbits stattfindet, um den Avalanche-Effekt zu beschleunigen und Bitänderungen schneller zu verteilen. Bei modernen Blockchiffrierern, die auch softwareseitig gut implementierbar sein sollen, wird deswegen eine zum Teil serielle Anordnung der S-Boxen bevorzugt, da so bei guter Konstruktion nie nur eine S-Box aktiv sein kann.

Je nach Anordnung der S-Boxen könnte der letzte Punkt wieder umgangen werden, indem gekürzte Differenzen betrachtet werden (Abschnitt 4.7). Dieser Angriff hat schon so manchen gegen diff. Kryptoanalyse immunen Chiffrierer wieder verwundbar gemacht.

Für eine ganze Reihe interessanter neuer Ansätze zur Konstruktion von Rundenfunktion empfiehlt sich die Lektüre der Dokumentationen der AES Kandidaten der ersten Runde, die auf der CD unter `/Papers/aes` zu finden sind.

[1] siehe z.B. den Chiffrierer Rijndael, auf der CD unter /Papers/AES/Rijndael

Kapitel 7

Advanced Encryption Standard

Als der **Data Encryption Standard** 1992 erneut ratifiziert werden mußte, wurde ihm zwar noch ausreichende Sicherheit bescheinigt, allerdings wurde empfohlen, einen neuen Standard auszuschreiben, der als Blockchiffrierer für das nächste Jahrhundert geeignet sein würde. Zuerst schlug die NIST den Triple-DES als neuen Standard vor, was allerdings von allen namhaften Kryptologen sehr kritisiert wurde und sich in dem Wunsch nach einem komplett neuen Chiffrierer äußerte. Daher begann Mitte 1997 die Ausschreibung um den sogenannten **Advanced Encryption Standard**, kurz **AES**.

Bis Mitte Juni 1998 trafen 21 Bewerbungen um diesen Standard ein, wovon 15 für die erste Runde zugelassen wurden. In der ersten Runde, die im Juni 1999 endete, ging es um die Benennung von ca. fünf Finalisten, die in die zweite Runde eingehen durften. In diesem Zeitraum hatten alle Teilnehmer Gelegenheit, ihre Chiffrierer vorzustellen und der Öffentlichkeit Zeit zu geben, sich an einer Kryptoanalyse der Kandidaten zu versuchen. Um die Ergebnisse zu präsentieren, fanden in diesem einen Jahr zwei AES Konferenzen statt. In der zweiten Runde, die mit der Benennung der fünf Finalisten MARS, RC6, Rijndael, Serpent und Twofish am 9. August 1999 begann [AES99a], werden die Finalisten einer umfangreichen Kryptoanalyse durch die Öffentlichkeit und die NSA unterzogen, wobei neben der Analyse auch Hardwareeffizienz in den Vordergrund treten soll. Die zweite Runde wird mit der Präsentation der von der NSA gefundenen Ergebnisse auf einer dritten AES Konferenz am 13. April 2000 abgeschlossen. Bis 15. Mai 2000 werden noch öffentliche Kommentare akzeptiert, so daß nach einer abschließenden Analyse bis Sommer 2001 der neue Standard präsentiert werden kann (für einen Überblick über den Terminplan siehe auch [AES99b]). Die NIST strebt dabei stets eine öffentliche Diskussion aller gefundenen Erkenntnisse an, um die Glaubwürdigkeit des Standards nicht wieder einzuschränken, wie es beim DES geschah.

Eine wesentliche Voraussetzung für die Teilnahme an der Ausschreibung war der Verzicht aller Rechte an dem Chiffrierer, um Lizenzstreitereien vorzubeugen. Der Standard soll vollkommen offen sein und nicht durch irgendwelche Zahlungen an den Erfinder behindert werden, wie z.B. durch Patente. Die absolute Offenheit bei der Beschreibung des Chiffrierers und bei der anschließenden Anwendung soll das Vertrauen in den Standard stärken und den Gebrauch des Standards uneingeschränkt gewährleisten. Diese Politik wird zwar von den Kryptologen an den Universitäten und anderen geteilt, aus den Reihen der wirtschaftlich arbeitenden Kryptologen mit kommerziellen Algorithmen wurde dieser Vorgang aber heftig kritisiert (z.B. von Terry Ritter, siehe Anhang B.1), da ihnen damit jede Verdienstmöglichkeit an dem Standard beraubt wurde und sie eventuell sogar neue Erkenntnisse im Design von Chiffrierern hätten preisgeben müssen.

7.1 Kriterien für die Kandidaten

Im Gegensatz zum DES soll der AES auf möglichst vielen Plattformen effektiv implementierbar sein. Die folgende Tabelle beschreibt die grundlegenden Kriterien, wie von der NIST gefordert [AES97]:

- Die Geschwindigkeit sollte in etwa der des Triple-DES entsprechen, sowohl in Hardware als auch in Software.
- Gefordert ist neben einer effektiven Hardwareimplementation auch die effektive Implementation in Software auf 8- und 32-Bit Rechnern.
- Die Schlüssellänge soll wahlweise 128, 192 oder 256 Bit betragen, wobei keine Paritätsbits enthalten sein dürfen.
- Die Blockgröße soll 128 Bit betragen.
- Der Chiffrierer sollte möglichst sicher gegen alle bekannten Angriffe sein.

Neben diesen grundlegenden Kriterien wurden einige weitere Empfehlungen ausgesprochen, welche die Kandidaten erfüllen sollten:

- Ununterscheidbarkeit eines Chiffretextstroms von zufälligen Folgen,
- ausgeprägter Avalanche-Effekt,
- fundierte mathematische Grundlage für die Designkriterien des Chiffrierers.
- Möglichst geringer RAM-Bedarf (insbesondere für Smart-Card),
- möglichst große Flexibilität, um möglichst vielen Ansprüchen gerecht zu werden. Dies umfaßt z.B.
  - weitere erlaubte Block- und Schlüsselgrößen,
  - effektive Implementationen auf weiteren Plattformen.
  - Einsatz als Hash-Funktion, Stromchiffrierer oder Authentifizierer (MAC).
- Möglichst einfache Struktur,
- möglichst schneller Wechsel zwischen verschiedenen Schlüsseln, also hohe Geschwindigkeit des Schlüsselerzeugungsalgorithmusses,
- möglichst schnelle Ver- und Entschlüsselung unter dem gleichen Schlüssel,
- möglichst geringe Codegröße.

Biham hat zu diesen Kriterien in [Bih99a] Stellung bezogen und eine Ordnung der Kriterien vorgeschlagen, nach welcher die NIST die Finalisten auswählen sollte. Da der AES Dokumente mindestens 100 Jahre lang schützen muß, sollte er so sicher wie möglich sein. Daher sollte dieses Kriterium über alles andere gestellt werden.

Damit die Sicherheit auch gegen zukünftige Angriffe ausreichend ist, schlägt er vor, die doppelte Rundenzahl zu wählen, die den Chiffrierer sicher gegen diff. und lin. Kryptoanalyse macht. Der beweisbaren Sicherheit tritt er mit großer Skepsis gegenüber, da meistens nur die Sicherheit gegen die Standardverfahren gemessen wird, aber nicht die Sicherheit gegen erweiterte Verfahren.

Der Geschwindigkeit mißt er keine hohe Bedeutung zu, da alle Kandidaten die erforderliche Geschwindigkeit unterboten haben und im Laufe der Benutzung große Fortschritte in der Rechnerarchitektur gemacht werden. Einfachheit im Design ist ein wünschenswertes Ziel und es sollte auf jeden Fall vermieden werden, daß ein kompliziertes Design gewählt wird, da sich Komplexität nur schwer untersuchen läßt. Je einfacher das Design, umso besser und schneller kann man die Sicherheit eines Chiffrierers abschätzen. Durch ein einfacheres Design werden auch mehr Leute zu einer Analyse des Algorithmusses ermutigt, was den Fortschritt in der Kryptologie beschleunigt.

Er empfiehlt nicht uneingeschränkt die Auswahl eines innovativen Designs, da neue Strukturen bisher noch nicht ausreichend untersucht wurden und man diesen Strukturen noch Zeit geben sollte, bis ihre Mechanismen ausreichend verstanden worden sind. Andererseits könnte eine Ernennung eines solchen Algorithmusses als Finalisten die Untersuchung beschleunigen und daher durchaus auch positive Impulse für die Forschung geben.

In [Bih99b] vergleicht er alle Erstrundenkandidaten miteinander und gibt Kommentare zu diesen ab (siehe auch [Biha98]). Da er selber ein Bewerber mit seinem Kandidaten Serpent[1] ist, sollten seine Empfehlungen trotz seiner Reputation allerdings mit einem gewissen Maß an Vorsicht genossen werden, da nicht alle bekannten Kryptologen seine Meinung teilen. Er hat daher in seinem Schreiben eine Diskussion seiner Thesen angeregt.

Neben Biham's Kommentar sind eine Vielzahl weiterer Kommentare eingegangen, die auf der AES Homepage unter www.nist.gov/aes zu finden sind. Nach Beendigung der ersten Runde hat die NIST ein umfangreiches Dokument [AES99c] zur Benennung der Finalisten herausgebracht, indem alle 15 Kandidaten verglichen und alle gefundenen Ergebnisse präsentiert werden.

Die fünf Finalisten sind, wie bereits erwähnt, MARS, RC6, Rijndael, Serpent und Twofish. Bei allen Kandidaten wurden keine wesentlichen Schwächen gefunden und alle zeichnen sich durch unterschiedliche Vor- und Nachteile aus.

MARS: Besitzt in der vorliegenden Form ein hohes Maß an Sicherheit ud ist recht schnell, ähnlich wie der RC6 ist er sehr flexibel, was Schlüssellängen angeht. Er ist, wie der RC6, aufgrund der datanabhängigen Rotationen relativ stark auf 32-Bit Prozessoren beschränkt. Im Gegensatz zum RC6 können die Teilschlüssel nach und nach berechnet werden, was eine Implementation in Smartcards erleichert. Da die Vorteile die Nachteile überwiegen, wurde MARS als Finalist nominiert.

RC6: Der größte Pluspunkt des RC6 ist seine Einfachheit und Schnelligkeit. Die Einfachheit begünstigt eine schnellere Analyse des RC6. Obwohl die Sicherheit bei 20 Runden kleiner ist als bei seinen Konkurrenten, könnte dieser Punkt korrigiert werden. Wie MARS ist der RC6 nur wenig für 8-Bit Rechner optimiert und seine RAM Erfordernisse für die Teilschlüssel erschweren eine Implementation in Smartcards. Auch hier überwogen die Vorteile die Nachteile.

Rijndael: Rijndael verwendet ein sehr neues Designkriterium für Rundenfunktionen, lateinische Quadrate. Er ist auf allen Plattformen gleich schnell und benötigt nur sehr wenig RAM für die Teilschlüssel. Rijndael besitzt vermutlich die größten Chancen, der künftige AES zu werden.

Serpent: Obwohl nicht der schnellste, ist er ultra-konservativ auf Sicherheit eingestellt. Die minimale Rundenzahl, ab der der Serpent sicher ist, wurde verdoppelt, so daß er die höchste Sicherheit aller Kandidaten bietet. Dadurch ist er nur mäßig schnell auf allen Plattformen. Durch weitere Optimierungen benötigt er nur wenig RAM und ROM für Smartcardimplementationen. Da die Vorteile überwiegen, wurde Serpent nominiert.

Twofish: Ähnlich wie Rijndael besitzt auch Twofish fast nur Vorteile, er ist schnell, flexibel und gut geeignet für Smartcardanwendungen. Nachteil ist seine relativ hohe Komplexität, die eine Analyse erschwert.

[1] auf der CD unter /Papers/AES/Serpent

7.2 Ein Bewerber: RC6

Der RC6 [RRSY98] von RSA Inc. ist eine Weiterentwicklung des bekannten RC5, einem Blockchiffrierer mit variablen Block und Schlüssellängen. Der RC5 operiert auf einer Blocklänge von $2w$, wobei $w = 16, 32, 64$ möglich sind. Die genaue Spezifikation ist eigentlich RC5-$w/r/b$, wobei r die Rundenzahl und b die Anzahl der Schlüsselbytes angibt. Für die häufigste Variante $w = 32$ wird $r = 12$ und $b = 16$ empfohlen, dies entspricht einer Blockgröße von 64 Bit mit 12 Runden und 128-Bit Schlüssel. Die Beschreibung des RC5 findet sich in [Riv95], eine erste Analyse in [KaYi95], eine überarbeitete Version der Analyse ist [KaYi98], die alle bisher gefundenen Angriffe wie z.B. [KnMe96, Sel98, BiKu98] enthält.

Der RC5 war nicht direkt als verwendbarer Chiffrierer konzipiert, sondern sehr einfach gestaltet, um eine umfassende Analyse der in dem Chiffrierer verwendeten datenabhängigen Rotationen voranzutreiben. Diese Rechnung ist auch aufgegangen, so daß Rivest u.a. dieses Wissen nutzen konnten, um den RC6 so zu gestalten, daß alle Schwächen des RC5 ausgemerzt wurden. Dabei konnte die Einfachheit des RC5 weitgehend beibehalten werden.

7.2.1 Design des RC6

Wie bereits erwähnt, baut der RC6 auf dem RC5 auf, d.h. das Design entspricht teilweise dem des RC5. Um die Anforderungen des AES zu erfüllen, wurden einige kleinere Änderungen im Aufbau vorgenommen sowie die gefundenen Schwächen des RC5 entfernt. Eine kleine Historie der Entwicklung vom RC5 zum RC6 findet sich in Abschnitt 5.1 von [RRSY98].

- Um bei einer typischen Wordlänge von 32 Bit auf 128-Bit Blöcke zu kommen, operiert der RC6 auf $4w$-Bit anstelle der $2w$-Bit Blöcke. Dadurch erhöhte sich die Anzahl der benötigten Register von zwei auf vier. Dieses Vorgehen war die effizienteste Möglichkeit, da 64-Bit Operationen nur von sehr wenigen Rechnern effizient vorgenommen werden können.

- Eine der wesentlichen nichtlinearen Operationen des RC5, die Addition modulo 2^w, wurde durch die Multiplikation modulo 2^w ergänzt, die eine wesentlich bessere Diffusion bietet und in aktuellen Prozessoren schnell durchgeführt werden kann.

- Die datenabhängigen Rotationen hängen durch die Verwendung der Multiplikation von mehr Inputbits ab, was die Diffusion beschleunigt und in höherer Sicherheit bei gleicher Rundenzahl resultiert.

Wie der RC5 kann auch der RC6 parametrisiert werden durch RC6-$w/r/b$, allerdings ist durch die Anforderungen $w = 32$ und $b = 16, 24, 32$ festgelegt, obwohl natürlich auch andere Werte möglich sind. Für die geforderten Werte empfehlen die Autoren als Rundenzahl $r = 20$. Für sehr große Werte von b ($b > 175$) gibt es allerdings äquivalente Schlüssel, sodaß b nicht zu groß gewählt werden sollte.

Der RC6 verwendet sechs Basisoperationen, für die folgende Bezeichnungen verwendet werden (der Logarithmus zur Basis 2 wird durch ld gekennzeichnet):

$\mathbf{a} + \mathbf{b}$ Addition modulo 2^w

$\mathbf{a} - \mathbf{b}$ Subtraktion modulo 2^w

$\mathbf{a} \oplus \mathbf{b}$ Vektoraddition von w-Bit Vektoren oder bitweises XOR von w-Bit Zahlen

$\mathbf{a} \times \mathbf{b}$ Multiplikation modulo 2^w

$\mathbf{a} \lll \mathbf{b}$ Rotation der w-Bit Zahl a um den Wert der niedrigsten ld w Bits von b nach links.

Algorithmus A: Schlüsselerzeugung
INPUT: Parameter $w/r/b$, Konstanten P_w, Q_w, b Byte Schlüssel, geladen in ein w-Bit Wordarray L[0..c-1] mit $c = \frac{b}{4}$ Feldern.
OUTPUT: $2r+4$ w-Bit Rundenschlüssel S[0..2r+3]

1. $S[0] = P_w$
2. Für i von 1 bis $2r+3$ berechne $S[i] = S[i-1] + Q_w$
3. Setze $A = B = i = j = 0$, $v = 3 \cdot \max\{c, 2r+4\}$
4. Für s von 1 bis v führe folgende Schritte durch
   a. $A = S[i] = S[i] + A + B) \lll 3$
   b. $B = L[j] = L[j] + A + B) \lll (A+B)$
   c. $i = (i+1) \bmod (2r+4)$, $j = (j+1) \bmod c$

Algorithmus B: Verschlüsselung
INPUT: Parameter $w/r/b$, erweiterter Schlüssel S[0..2r+3], $4w$-Bit Datenblock geladen in das Feld (A,B,C,D).
OUTPUT: Chiffretext gespeichert im Feld (A,B,C,D).

1. $B = B + S[0], D = D + S[1]$ (Prewhitening)
2. Für i von 1 bis r führe folgende Schritte durch:
   a. $t = (B \times (2B+1)) \lll \mathrm{ld}\, w$.
   b. $u = (D \times (2D+1)) \lll \mathrm{ld}\, w$.
   c. $A = ((A \oplus t) \lll u) + S[2i]$
   d. $C = ((C \oplus u) \lll t) + S[2i+1]$
   e. $(A,B,C,D) = (B,C,D,A)$
3. $A = A + S[2r+2], C = C + S[2r+3]$ (Postwhitening)

Algorithmus C: Entschlüsselung
INPUT: Parameter $w/r/b$, erweiterter Schlüssel S[0..2r+3], Chiffretext gespeichert im Feld (A,B,C,D)
OUTPUT: Klartext gespeichert im Feld (A,B,C,D).

1. $C = C - S[2r+3], A = A - S[2r+2]$ (Entfernen des Postwhitenings)
2. Für i von r bis 1 führe folgende Schritte durch:
   a. $(A,B,C,D) = (D,A,B,C)$
   b. $t = (B \times (2B+1)) \lll \mathrm{ld}\, w$.
   c. $u = (D \times (2D+1)) \lll \mathrm{ld}\, w$.
   d. $A = ((A - S[2i]) \ggg u) \oplus t$
   e. $C = ((C - S[2i+1]) \ggg t) \oplus u$
3. $D = D - S[1], B = B - S[0]$ (Entfernen des Prewhitenings)

Tabelle 7.1: **Algorithmus:** Der RC6

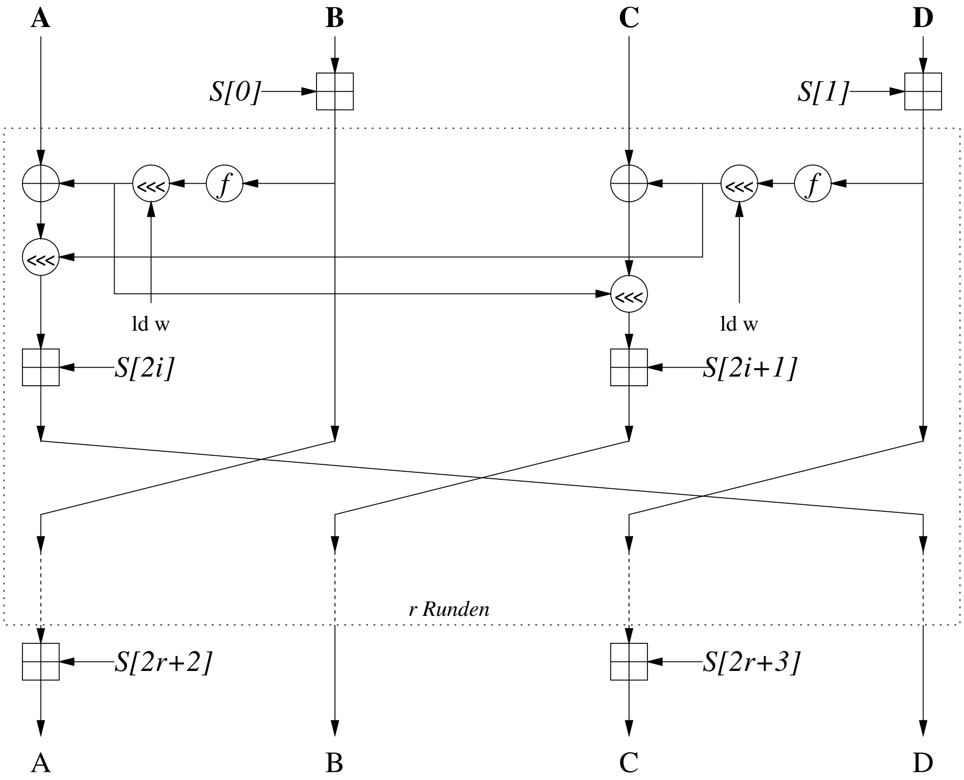

Abbildung 7.1: Struktur des RC6 (Verschlüsselung)

a ⋙ b Rotation der w-Bit Zahl a um den Wert der niedrigsten ld w Bits von b nach rechts.

Eine Beschreibung des RC6 wird in Algorithmus 7.1 gegeben, schematisch dargestellt wird er in Abbildung 7.1. Ähnlich zur Feistelstruktur wird in jeder Runde eine Hälfte des Blocks mit der anderen verknüpft und anschließend beide Hälften vertauscht. Die Rundenoperation arbeitet dabei weitestgehend getrennt auf zwei Vierteln. Dies läßt sich auf vielen modernen Prozessoren parallel berechnen, da sie typischerweise über mehrere Integereinheiten verfügen. Im Prinzip kann man sagen, daß der RC6 aus zwei parallel arbeitenden Feistelchiffren besteht, die sich gegenseitig beeinflussen. Dadurch ist die Entschlüsselung strukturell gleich, nur die die Rundenstruktur umgebenden nichtlinearen Operationen verlangen eine leichte Modifikation des Algorithmusses bei der Entschlüsselung.

Die Schlüssel- und Datenbytes werden in Little-Endian Weise eingelesen, d.h. das erste Byte gelangt in das niederwertigste Byte des ersten Feldelements und das letzte Byte in das höchstwertige Byte des letzten Feldelements. Da die meisten PC's auf diese Weise arbeiten, stellt dies kein Problem dar, nur bei der Anwendung des Chiffrierers auf Sun oder Macintosh Computern muß darauf geachtet werden, daß die Bytes richtig zugeordnet werden.

Wie alle Kandidaten besitzt auch der RC6 ein Pre- und Postwhitening, also eine Addition von Schlüsselmaterial am Anfang und Ende der Verschlüsselungsoperation. Leider sind die Ver- und Entschlüsselung nicht ganz identisch, bei der Entschlüsselung müssen die Additionen durch Subtraktionen ersetzt werden, die beiden Linksshifts bei der Verknüpfung der beiden Hälften müssen durch Rechtsshifts ersetzt werden. Die Struktur bleibt im Wesentlichen erhalten, die Permutation am Ende jeder Runde verschiebt sich an den Anfang und wird durch ihre Inverse ersetzt. Dadurch werden natürlich mehr Gatter bei einer Hardwareimplementation benötigt, aber dieses Manko ist heute kein Problem mehr.

Angriffsvariante / Runden		8	12	16	20	24
diff. Analyse	iteratives Differential	2^{91}	2^{147}	2^{210}	2^{273}	2^{329}
	verbessertes Differential	2^{56}	2^{117}	2^{190}	2^{238}	2^{299}
lin. Analyse	Grundangriff	2^{62}	2^{102}	2^{142}	2^{182}	2^{222}
	erweiterter Angriff	2^{47}	2^{83}	2^{119}	2^{155}	2^{191}

Tabelle 7.2: Aufwand der diff. und lin. Kryptoanalyse beim RC6

Ein größeres Problem stellen die $2r+4$ Teilschlüssel dar, da diese nicht „on-the-fly" erzeugt werden können. Bei $r = 20$ werden daher neben den zur Verschlüsselung benötigten Variablen auch noch 176 Bytes RAM für die Teilschlüssel benötigt, eine Menge, die den Speicher heutiger Smartcards übersteigt. Dadurch eignet sich der RC6 noch nicht zu einer Implementation in Smartcards, in zukünftigen Generationen von Chips dürfte dieses Problem aber behoben werden.

Die verwendeten „magischen" Konstanten P_w und Q_w bei der Schlüsselerzeugung werden aus der Binärdarstellung der ersten w Nachkommastellen der Zahlen $e-2$ und $\phi - 1$ gewonnen, wobei $\phi = \frac{\sqrt{5}+1}{2}$ der goldene Schnitt ist. Im Falle $w = 32$ ist somit $P_{32} = 0xB7E15163$ und $Q_{32} = 0x9E3779B9$. Diese Konstanten können auch durch andere ersetzt werden, dies könnte aber nur eine vollständige Schlüsselsuche abwenden, solange diese Werte nicht bekannt sind oder sonstwie benutzt werden können.

Die datenabhängigen Rotationen werden intensiv genutzt, dies trägt wesentlich zur Sicherheit des RC6 mit bei. Die Multiplikation wird in der Funktion $f(x) = x(2x+1) \bmod 2^w$ benutzt. Diese Funktion ist bijektiv und ihre höchsten Bits, die für die Rotation benutzt werden, sind hochgradig von allen Bits abhängig, so daß diese Funktion als wirksame nichtlineare Komponente verwendet wurde. Da die Funktion bijektiv ist, wird die Entropie von x nicht verändert; neben der hohen Diffusion ein weiterer Grund, daß diese Funktion gewählt wurde.

Bemerkung: In [CRRY98] wird ein kleiner Beweis für die Bijektivität von f gegeben; die Aussage gilt sogar für alle Funktionen $f(x) = x(rx+s) \bmod 2^w$ mit $w > 0$, wenn r gerade und s ungerade. □

7.2.2 Sicherheit des RC6

Die erste Referenz für die Sicherheit des RC6 ist die umfangreiche Analyse von Rivest u.a. in [CRRY98]. Obgleich dort kein praktischer Angriff vorgestellt wird, wird in einem „worst-case" Szenario der minimale Aufwand für einen Angriff hergeleitet. Die in dem Report vorgestellten Zahlen werden wahrscheinlich bei einem realen Angriff um einiges höher sein, solange keine wesentlichen Verbesserungen gefunden werden.

In Tabelle 7.2 gebe ich einen Überblick über die Anforderungen an wählbaren bzw. bekannten Klartexten für die diff. und lin. Kryptoanalyse, wie sie in [CRRY98] berechnet worden sind. In dem verbesserten differentiellen Angriff wurde das iterative 6-Runden Differential am Anfang und Ende durch bessere ersetzt. In der erweiterten lin. Kryptoanalyse wurden mehrfache Approximationen und lineare Hüllen verwendet. Der RC6 ist vermutlich einer der ersten Chiffrierer, der die Bedeutung der Begriffe Differential und lineare Hülle verdeutlicht.

Differentielle Kryptoanalyse

Als Differenz kommen beim RC6 zwei Verknüpfungen in Frage, das XOR und die Integersubtraktion. In Abschnitt 5.2 bis 5.4 von [CRRY98] werden beide Möglichkeiten untersucht und es stellt sich heraus, daß die Subtraktion geringfügig besser ist, was die Wahrscheinlichkeiten der Charakteristiken betrifft. Eine wichtige Beobachtung ist, daß der RC6 bezüglich dieser Differenzen kein Markovchiffrierer ist. Dies sieht man sofort, wenn man bedenkt, daß für einen Markovchiffrierer die Wahrscheinlichkeit einer Charakteristik unabhängig von dem exakten Input sein muß. Dies kann aber aufgrund der datenabhängigen Rotationen nicht erfüllt sein, da die Rotationszahl von dem exakten Wert der Inputs abhängt und nicht von der Differenz. Die wichtigste Eigenschaft von Markovchiffrierern, die Existenz einer ergodischen Verteilung, gilt allerdings auch in gewissem Rahmen für den RC6. Obwohl auch für hohe Rundenzahlen nicht alle Differenzen möglich sind, so werden dennoch die entsprechenden Wahrscheinlichkeiten klein, sodaß nach einer ausreichenden Rundenzahl der RC6 sicher gegen eine diff. Kryptoanalyse ist.

Der sinnvollste Ansatz für die Konstruktion von Charakteristiken ist die Annahme, daß innerhalb der Charakteristik keine Rotationen stattfinden; dieser Fall tritt bei einer zufälligen Wahl der Klartexte mit Wahrscheinlichkeit $\frac{1}{w}$ auf. Außerdem darf die Differenz keine Bits enthalten, die die Rotation beeinflussen. Werden Rotationen zugelassen, so wird eine weitere Analyse unmöglich gemacht, da sich diese Änderungen danach extrem schnell weiterentwickeln. Die datenabhängigen Rotationen sorgen also dafür, daß die Wahrscheinlichkeiten der Charakteristiken schnell sehr klein werden.

Weiterhin haben sich Differenzen, die nur aus der Änderung eines einzelnen Bits bestehen, als die praktischsten herausgestellt, um iterative Charakteristiken zu konstruieren. Typischerweise haben diese Differenzen die höchsten Wahrscheinlichkeiten. Beim RC5 konnten einige gute Charakteristiken gefunden werden, die Differenzen mit höheren Hamminggewichten verwenden, die Designer des RC6 sind aber zuversichtlich, daß die Einführung der quadratischen Funktionen diese Charakteristiken verhindert.

Um die diff. Kryptoanalyse des RC6 zu verstehen, wurden in [CRRY98] zuerst leichtere Varianten des RC6 gebrochen, die die quadratische Funktion durch die Identität ersetzen und/oder die Rotation um ld w Stellen am Anfang weglassen. Die Rundenstruktur erlaubt die Konstruktion einer iterativen 6-Runden Charakteristik auf drei verschiedene Weisen, wovon zwei am vielversprechendsten sind und jeweils die Wanderung von Bitdifferenzen in einer der beiden Hälften verfolgen. Diese iterative Charakteristiken lassen sich leicht zu einem Differential ausbauen, indem man die während der Wanderung der Differenz vorkommenden freien Variablen variabel läßt; dies verkleinert die Anzahl der nötigen Paare erheblich. Führt man die Rotation am Anfang um die w Stellen wieder ein, so verändert sich nur der Kurs der Differenzen, die drei Typen bleiben aber mit gleicher Wahrscheinlichkeit bestehen. Außerdem werden die freien Variablen in ihrem Wertebereich weiter eingeschränkt, was in einer etwas kleineren Wahrscheinlichkeit für das entsprechende Differential resultiert. Führt man statt der Rotation die quadratische Funktion ein, so ergibt sich ein ähnliches Bild: Es gibt zwar nach wie vor die größere Freiheit in den freien Variablen, durch die quadratische Funktion variiert nun aber die Wahrscheinlichkeit der Charakteristik bei unterschiedlichen Wahlen für die Variablen. Beide Methoden zusammen, die Einführung der Rotation und der quadratische Funktion, ergeben schließlich nur noch einen Gewinn von Faktor 4 bei Betrachtung des Differentials anstelle der Charakteristik. In Tabelle 7.3 gebe ich die bestmögliche iterative Charakteristik für den RC6 an, sowie das allgemeinere Differential. Die Werte für t, s, v können zwischen 0 und 26 variieren, u muß zwischen 15 und 26 liegen. Maximal wird die Wahrscheinlichkeit des Differentials für $t = u = 11$, sie beträgt dann 2^{-91}.

Um die diff. Kryptoanalyse des 20-Runden RC6 zu verbessern, können am Anfang und am Ende die Differentiale durch bessere ersetzt werden. In Abschnitt 7.2 von [CRRY98] werden einige Varianten erwähnt und die besten Ergebnisse präsentiert.

Lineare Kryptoanalyse

In [CRRY98] werden zwei verschiedene Typen von linearen Approximationen betrachtet, was die Approximation der datenabhängigen Rotationen betrifft.

allg. Differential					beste Charakteristik			
e_{t+5}	e_t	0	0		e_{16}	e_{11}	0	0
e_t	0	0	0		e_{11}	0	0	0
0	0	0	e_s		0	0	0	e_{26}
0	e_u	e_s	0		0	e_{26}	e_{26}	0
e_u	e_{u-5}	0	e_v		e_{26}	e_{21}	0	e_{26}
e_{u-5}	e_{u-10}	e_v	0		e_{21}	e_{16}	e_{26}	0
e_{u-10}	e_{u-15}	0	0		e_{16}	e_{11}	0	0

e_i kennzeichnet den i-ten Einheitsvektor oder 2^i als Integer.

Tabelle 7.3: iteratives Differential und beste Charakteristik des RC6

- Typ I: Die Rotation $A = B \lll C$ wird durch $A \bullet \Gamma_A = B \bullet \Gamma_B \oplus C \bullet \Gamma_C$ approximiert.
- Typ II: Die Rotation $A = B \lll C$ wird durch $A \bullet \Gamma_A = B \bullet \Gamma_B$ approximiert.

Typ I Approximationen haben sich als besser erwiesen; eine Kombination beider Typen ist aber auch möglich, obwohl dies die Analyse nicht verbessert. Bei den Typ I Approximationen ist Γ_C auf Werte kleiner w beschränkt, da höhere Bits die Rotation nicht beeinflussen. Damit die Approximation überhaupt eine wird, sollte $\Gamma_A = \Gamma_B$ sein, bis auf eventuelle Rotationen. Durch die parallele Struktur des RC6 bedingt, muß auch Γ_A kleiner als w sein. Mit diesen Resultaten ist die Abweichung der Approximation $\rho = 2^{-6}$.

Typ I Approximationen umgehen die quadratische Funktion, wodurch neben der Rotation als weiteres nichtlineares Element nur noch die Addition modulo 2^w berücksichtigt werden muß. Die Addition erzwingt die Anwendung von Approximationen, die nur einzelne Bits verwendet. Die Wahrscheinlichkeit einer Approximation der Rotation wird zwar verbessert, wenn man mehrere Bits verwendet, die Addition verhindert dies jedoch. Insgesamt ist die Wahrscheinlichkeit am besten, wenn man sich auf einzelne Bits beschränkt.

Wie man sich leicht verdeutlichen kann, muß bei Typ II Approximationen auch die quadratische Funktion approximiert werden. Obwohl sie in Bit 0 sogar linear ist, nützt einem diese Eigenschaft aufgrund der fixen Rotation um ld w Stellen nur wenig, wodurch sich diese Art der Approximation als nicht sehr vorteilhaft herausgestellt hat. Auch hier gilt, daß aufgrund der Funktion nicht gut mehrere Bits zugleich betrachtet werden können.

Basis der lin. Kryptoanalyse des RC6 ist die iterative 2-Runden Approximation

$$A \bullet e_t \oplus C \bullet e_s = A'' \bullet e_u \oplus C'' \bullet e_v,$$

wobei A und C die Werte in den entsprechenden Registern am Anfang der Approximation kennzeichnen und A'' und C'' die Inhalte der Register A und C nach zwei Runden. e_i bezeichnet wieder den i-ten Einheitsvektor. Die Abweichung $|\tilde{p}|$ errechnet sich mittels des Piling-Up-Lemmas zu

$$|\tilde{p}| = \alpha_u \cdot \rho \cdot \alpha_v \cdot \rho \cdot 2^3,$$

wobei α_i die Abweichung der Approximation über die Addition modulo 2^w ist. Für $u = v = 0$ ergibt sich so insgesamt für $|\tilde{p}| = 2^{-11}$, 2^{-12}, falls eine der beiden Variablen Null, die andere kleiner als fünf ist. Gilt $1 \leq u, v \leq 5$, so ist $|\tilde{p}| = 2^{-13}$.

Die erste Zeile von Tabelle 7.2 gibt die Abweichung der entsprechend häufig hintereinandergeschalteten Approximation für verschiedene Rundenzahlen an. Wie bei der diff. Kryptoanalyse wird davon ausgegangen, daß es möglich ist, einen Angriff mit einer um zwei Runden kürzeren Approximation durchzuführen.

Durch Variation der Variablen t, s, u und v ist es möglich, mehrere Approximationen zugleich zu betrachten, gemäß der Arbeit von Robshaw u.a. in [KaRo94]. Durch die parallele Betrachtung ist es zwar möglich, die Anzahl der nötigen Klartexte proportional zu verkleinern, allerdings auf Kosten eines entsprechenden Anstiegs beim Berechnungsaufwand. Im Falle des RC6 liegt der reduzierte Klartext- / gesteigerte Berechnungsaufwand bei Betrachtung von mehrfachen Approximationen bei Faktor 2^{11}.

Ein ähnliches Vorgehen erlaubt auch eine Abschätzung des Gewinns durch Berücksichtigung der entsprechenden linearen Hülle. Hält man die Eingangs- und Ausgangswerte der r-Runden Approximation fest, so können die Zwischenwerte entsprechend variieren, also zwischen Null und Fünf. Je nach Rundenzahl können so eine wachsende Anzahl an Approximationen zusammengefaßt werden. Der Gewinn liegt bei 20 Runden bereits bei 2^{16}. Werden beide Verfahren zusammen angewendet, so ergeben sich die Zahlen aus der letzten Zeile von Tabelle 7.2.

Anmerkung: Während die Zahlen der diff. Kryptoanalyse bis auf eine Beachtung des Rauschabstands relativ nahe an den tatsächlichen Erfordernissen an wählbaren Klartexten liegen, so sind die Zahlen für die lin. Kryptoanalyse sehr grobe und vor allem sehr konservative Schätzungen, die bei einem tatsächlichen Angriff wesentlich höher liegen können. Die einzelnen Annahmen können in den Kapiteln 8 und 9 von [CRRY98] nachgelesen werden. Rivest u.a. gehen davon aus, daß eine Unterschreitung ihrer konservativen Schätzungen nicht ohne weiteres möglich ist, wollen sie jedoch nicht grundsätzlich ausschließen. Aufgrund ihrer Schätzungen haben sie die Rundenzahl auf 20 festgelegt, da so beide Angriffe mehr Klartexte erfordern, als überhaupt verfügbar sind. □

Erweiterte Verfahren

In [CRRY98] wird bezweifelt, ob die erweiterten Verfahren einen geringeren Aufwand als die Standardverfahren benötigen. Die erweiterten Verfahren haben, wie bereits beschrieben, das Problem, sich nicht einfach auf viele Runden ausweiten zu lassen. So konnten keine geeigneten Charakteristiken gefunden werden, die z.B. für eine differentiell-lineare Analyse geeignet wären. Auch höhere Differenzen oder Interpolationsattacken sind nur wenig für Chiffrierer mit hoher Rundenzahl geeignet, das gilt auch für den RC6. Gekürzte Differenzen werden teilweise in [CRRY98] verwendet, allerdings ist es kein reiner Angriff mit gekürzten Differenzen, als vielmehr die Einführung kleiner Optimierungen zur Reduzierung des Aufwands.

Schwache Schlüssel

Schwache Schlüssel im ersten Sinne (siehe Abschnitt 6.3.2) existieren nicht, beim RC5 gibt es allerdings für bestimmte Schlüssel Charakteristiken und Approximationen mit höherer Wahrscheinlichkeit, obwohl diese Klassen relativ klein sind. Diese Schlüssel resultieren aber nicht aus dem Schlüsselerzeugungsalgorithmus, sondern aus der Struktur des RC5 [KnMe96]. Ob solche schwachen Schlüssel im zweiten Sinne auch für den RC6 existieren, ist bisher nicht bekannt.

Teil III

Anhang

Anhang A

Das Geburtstagsparadoxon

Das Geburtstagsparadoxon ist ein bekanntes Problem aus der Kombinatorik, welches aufgrund seines verblüffenden Ergebnisses eine ganze Reihe von Anwendungen in der Kryptologie hat. Das ursprüngliche Problem ist folgendes:

Wie groß ist die Wahrscheinlichkeit, daß in einer Gruppe von r Personen mindestens zwei Leute am gleichen Tag Geburtstag haben, wenn man davon ausgeht, daß die Wahrscheinlichkeit, an einem bestimmten Tag geboren worden zu sein, gleichverteilt ist? Was ist der Erwartungswert?

Es ist kaum intuitiv, daß bereits in einer Gruppe von 23 Personen die Wahrscheinlichkeit ca. 51% groß ist.

Modellentwicklung

Drückt man das Problem mit Urnen und Bällen aus, so sieht es wie folgt aus:

Seien n Urnen gegeben, und eine beliebige Anzahl an Bällen. Nun werfe man solange Bälle zufällig in eine der Urnen, bis eine Urne einen zweiten Ball enthält. Die Wartezeit bis zum Eintreten des Ereignisses ist der Erwartungswert des oben geschilderten Problems. Zur Berechnung der Wahrscheinlichkeit betrachte ich dieses Ereignis von der anderen Seite, also die Wahrscheinlichkeit, daß bis zum r-ten Wurf keine Urne einen zweiten Ball enthält.

Für den ersten Ball ist die Bedingung immer erfüllt, die Wahrscheinlichkeit ist also 1. Für den zweiten Ball bestehen nur noch $n-1$ aus n Möglichkeiten, die Wahrscheinlichkeit, daß die Urne noch nicht belegt ist, ist also $\frac{n-1}{n}$. Für den dritten Ball gibt es noch $n-2$ aus n Möglichkeiten, also ist die Wahrscheinlichkeit für keinen Treffer

$$\frac{n}{n} \cdot \frac{n-1}{n} \cdot \frac{n-2}{n},$$

da die Ballwürfe stochastisch unabhängig sind.

Also ist die Wahrscheinlichkeit p_r, daß nach r Ballwürfen keine Urne zwei oder mehr Bälle enthält, entsprechend

$$p_r = n(n-1)(n-2)\cdots(n-r+1)/n^r = \frac{n!}{(n-r)!n^r}.$$

Die Wahrscheinlichkeit, daß es mindestens einen Treffer gab, ist also $P_r = 1 - p_r$. Für $n = 365$, $r = 23$ ist $p_r = 0.493$, also $P_r = 0.507$ die Lösung des Geburtstagsproblems.

Um diese Wahrscheinlichkeit für große n zu approximieren, also die asymptotische Verteilung zu finden, betrachte ich im Folgenden Zufallsfunktionen.

Definition A.1 *Ein* **Zufallsfunktionenraum** *ist ein Tripel $(X, \mathcal{J}, \mathcal{P})$, wobei X eine endliche Menge mit Mächtigkeit n, $\mathcal{J}$ der Raum aller Funktionen $F : X \to X$ und $\mathcal{P}$ ein Wahrscheinlichkeitsmaß. Die Wahrscheinlichkeit, eine bestimmte Funktion F zu wählen, sei gegeben durch $\mathcal{P}(F)$. Eine so gewählte Funktion heiße* **Zufallsfunktion**.

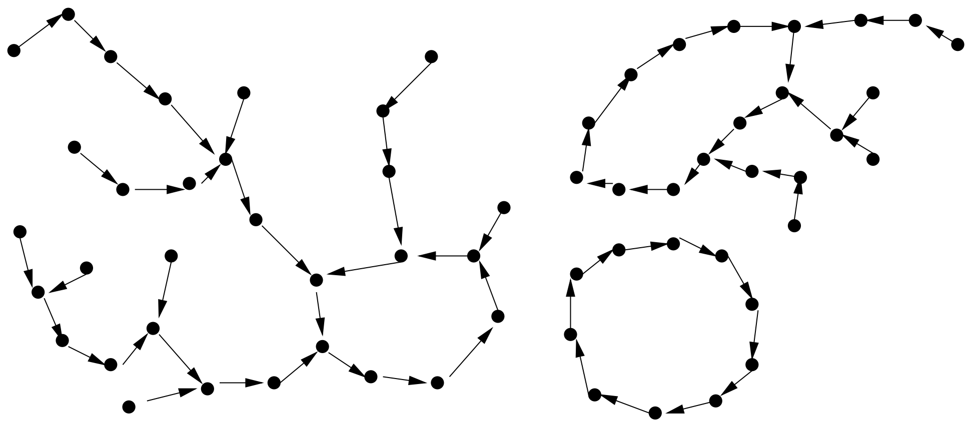

Abbildung A.1: Gerichteter Graph einer Funktion

Im Folgenden nehme ich an, daß $\mathcal{P}$ eine Gleichverteilung besitzt. Die Wahrscheinlichkeit eine Funktion F zu wählen, ist also für alle F gleich $(1/_{n^n})$. Mit dem so gesetzten Raum $\mathcal{J}$ und der Gleichverteilung von $\mathcal{P}$ nimmt das Bild $F(x)$ einer Zufallsfunktion mit gleicher Wahrscheinlichkeit jeden Wert y seines Bildraums für jedes x an.

Um Funktionen auf endlichen Mengen zu visualisieren, bietet sich die Darstellung als gerichteter Graph an; ein Beispiel für einen solchen Graph ist in Abbildung A.1 gegeben. Ein gerichteter Graph besteht aus n Knotenpunkten, wobei ein Pfeil von jedem Knoten x zu demjenigen Knoten y weist, für den $F(x) = y$ gilt. Da die Wertemenge endlich ist, ergibt sich für eine Funktion aus $\mathcal{J}$ immer ein ähnliches Bild: Es gibt einige Zyklen, in die mehr oder minder lange und verzweigte Äste hineinmünden.

Diese Art der Visualisierung beschreibt sehr gut die Dynamik einer Funktion F, also die Eigenschaften der Funktion F unter Iterierung eines beliebigen Startwerts. Ich werde nun die durchschnittliche Zyklen- und Astlänge errechnen. Hierzu führe ich einige Begriffe ein:

Definition A.2 *Sei $x \in X$, $F \in \mathcal{J}$, Dann bezeichne $F^k(x)$ die k-te Iteration von x unter F $(k > 0)$. Sei $F^0(x) = x \ \forall x \in X$. Sei $F^{(k)}(x) := \{y \in X | F^{-k}(y) = x\}$ $(k < 0)$ die Menge aller k-ten Inversen von x (die Menge kann ruhig leer sein).*

Definition A.3 *Es bezeichne*

(i) $S_F(x) := \{x, F(x), F^2(x), \ldots, F^{n-1}(x)\}$ *die Menge aller F-Bilder von x,*

(ii) $P_F(x) := \bigcup_{k=-n}^{0} F^{(k)}(x)$ *die Menge aller Vorgänger von x unter F,*

(iii) $C_F(x) := \{x, F(x), \ldots, F^{m-1}(x)\}$, *falls x zyklisch, es also ein m gibt mit* $F^m(x) = x$,

(iv) $K_F(x) = P_F(x) \cup S_F(x)$ *die* **Komponente** *von F, die x enthält,*

(v) $s_F(x) = |S_F(x)|$, $p_F(x) = |P_F(x)|$ *und schließlich* $l_F(x)$ *die Länge des Zyklus, der in* $K_F(x)$ *enthalten ist.*

Ich werde nun die Verteilungen von s und l bestimmen und dadurch den Erwartungswert für die Mächtigkeit der Mengen $S_F(x), C_F(x')$ errechnen können. Der Erwartungswert von s wird mir auch eine Approximation für den gesuchten Erwartungswert des Geburtstagsproblems liefern.

Sei also F zufällig gewählt. Die Wahrscheinlichkeit, daß $s = k$ und $l = j$ ist gegeben durch:

$$P(s = k, l = j) = \{F^r(x) \neq x, Fx, \ldots, F^{r-1}x \; (0 \leq r \leq k-1) \text{ und } F^k x = F^{k-j}x\}$$

Diese Wahrscheinlichkeit ähnelt der des klassischen Geburtstagsproblems, denn es ist

$$P(s = k, l = j) = \frac{(n-1)!}{(n-k)!n^k}, \quad 1 \leq j \leq k \leq n. \tag{A.1}$$

Interessanterweise beeinflußt die Länge des Zyklus die Wahrscheinlichkeit nicht, mit anderen Worten, der Erwartungswert der Länge des Zyklus und der des Astes, der in den Zyklus hineinführt, ist gleich. Daraus folgt, daß

$$E(l) = E[(s+1)/2]$$

Nun approximiere ich die Wahrscheinlichkeit, indem ich $k = \sqrt{n}x, j = \sqrt{n}y$ setze $(x, y > 0)$ und die Fakultäten mittels der Stirlingschen Formel $n! \approx \sqrt{2\pi} n^{n+1/2} e^{-n}$ nähere. Also habe ich:

$$\begin{aligned} P(s = \sqrt{n}x, l = \sqrt{n}y) &\approx \frac{n^{n-\sqrt{n}x-1/2} e^{-\sqrt{n}x}}{(n-\sqrt{n}x)^{n-\sqrt{n}x+1/2}} \\ &= \frac{n^{n-\sqrt{n}x-1/2} e^{-\sqrt{n}x}}{n^{n-\sqrt{n}x+1/2}\left(1-\frac{x}{\sqrt{n}}\right)^{n-\sqrt{n}x+1/2}}. \end{aligned} \tag{A.2}$$

Schreibe nun

$$\left(1 - \frac{x}{\sqrt{n}}\right)^{n-\sqrt{n}x+1/2} = \exp[(n - \sqrt{n}x + 1/2)\log(1 - x/\sqrt{n})].$$

Zur Erinnerung:
$\log(1-x)$ kann für $-1 \leq x < 1$ in die Taylorreihe $-\left(x + \frac{x^2}{2} + \ldots + \frac{x^n}{n} + \ldots\right)$ entwickelt werden. Ich betrachte den Fall, daß x klein im Vergleich zu n, also $x = O(1)$, d.h. $k = O(\sqrt{n})$, dann folgt:

$$\begin{aligned} \exp[(n - \sqrt{n}x + 1/2)\log(1 - x/\sqrt{n})] &= \exp[(n - \sqrt{n}x + 1/2)(-\frac{x}{\sqrt{n}} - \frac{x^2}{2n} - \ldots)] \\ &= \exp[-x\sqrt{n} + x^2 - \frac{x}{2\sqrt{n}} - \frac{x^2}{2} + \frac{x^3}{2\sqrt{n}} - \frac{x^2}{4n} \pm \ldots] \\ &\overset{n\to\infty}{\approx} \exp(-\sqrt{n}x + \frac{x^2}{2}) \end{aligned}$$

Damit folgt nun:

$$P(s = \sqrt{n}x, l = \sqrt{n}y) \approx \frac{e^{-\sqrt{n}x}}{ne^{-\sqrt{n}x+\frac{x^2}{2}}} = \frac{e^{-\frac{x^2}{2}}}{n}. \tag{A.3}$$

Also ist die asymptotische Dichte von $(s/\sqrt{n}, l/\sqrt{n})$ gegeben durch

$$f(x,y) = e^{-\frac{1}{2}x^2}, \qquad 0 < y \leq x < \infty \tag{A.4}$$

Um die einzelnen Randverteilungen $s/\sqrt{n}, l/\sqrt{n}$ zu bestimmen, integriere ich jeweils über die andere Variable:

$$f_1(x) = \int_0^x e^{-\frac{1}{2}x^2} dy = xe^{-\frac{1}{2}x^2}, \qquad 0 < x \tag{A.5}$$

$$f_2(x) = \int_0^y e^{-\frac{1}{2}x^2} dx = \sqrt{2\pi}(1 - \Phi(y)), \qquad 0 < y \tag{A.6}$$

Der Erwartungswert der Verteilung von $s/\sqrt{n}$ ist also:

$$\int_0^\infty x^2 e^{-\frac{1}{2}x^2} dx = \int_0^\infty \sqrt{2t}\, e^{-t} dt = \sqrt{2} \int_0^\infty t^{\frac{3}{2}-1} e^{-t} dt = \sqrt{2}\, \Gamma(\frac{3}{2}) = \sqrt{\frac{\pi}{2}}$$

Gehe ich zu den ursprünglichen Zufallsvariablen zurück, so erhalte ich

$$E(s) = \sqrt{\frac{\pi n}{2}} \quad \text{und damit} \quad E(l) = \frac{1}{2}\sqrt{\frac{\pi n}{2}} = \sqrt{\frac{\pi n}{8}}.$$

Der Erwartungswert $E(s)$ stellt auch die gewünschte Approximation für das Geburtstagsproblem: Nach ca. $\sqrt{n}$ Ballwürfen wird eine Urne einen zweiten Ball enthalten. □

Anhang B

Hyperlinks und Inhalt der CD

Anmerkung: Das Lesen dieses Kapitels macht nur mit der Online Version meiner Arbeit Sinn, da es Hyperlinks zu Internetseiten und den Inhalten der CD enthält. Ich empfehle das Arbeiten mit der CD auf einer Windows9x Plattform, da meine Programme darauf entwickelt worden sind. □

Dieser Anhang ist in vier Teile gegliedert:

- Links zu wichtigen Kryptoseiten im Internet
- Links zu den Inhalten der CD
- Das Programm zu meiner lin. Kryptoanalyse des FEAL-4
- Ein Programm zu Shannon's Entropie

B.1 Hyperlinks zu wichtigen Kryptoseiten im Internet

Homepages von wichtigen Kryptologen

Eli Biham: `http://www.cs.technion.ac.il/~biham/`

Lars Knudsen: `http://www.ii.uib.no/~larsr/`

Luke O'Connor: `http://www.fit.qut.edu.au/ILE/ile/Staff/oconnor.htm`

Terry Ritter: `http://www.io.com/~ritter`

Ron Rivest: `http://www.theory.lcs.mit.edu/~rivest/homepage.html`

S.E. Tavares: `http://http.ee.queensu.ca:8000/www/dept/facpages/set.html`

Serge Vaudenay: `http://www.ens.fr/~vaudenay/index_en.html`

Wichtige Kryptoseiten im Net

`http://www.io.com/~ritter` Terry Ritters Crypto Pages, enthält eine Vielzahl an Informationen, auch ein Lexikon mit allen wichtigen Begriffen.

`http://www.nist.gov/aes/` Die AES Homepage mit dem aktuellen Stand der Entwicklung

`http://www.ii.uib.no/~larsr/bc.html` Lars Knudsen's Block Cipher Lounge, mit großem Abschnitt zum Vergleich aller AES Kandidaten `nntp://sci.crypt` Usenet Newsgroup zu kryptologischen Themen

`http://www.iya.com/crypto.htm` Seite mit einer Vielzahl an Kryptodokumenten

`http://www.ftech.net/~monark/crypto/main.hts` Beginner's Guide to Cryptology

`http://www.mit.edu:8008/menelaus.mit.edu/cpunks`
Cipherpunks Mailingliste zum Mitlesen

`ftp://soda.berkeley.edu/pub/cipherpunks/Home.html`
Homepage der Cipherpunks in Berkeley

`http://www.pgp.com` Homepage von PGP

`http://www.rsa.com` Homepage von RSA Inc.

`ftp://ftp.tu-clausthal.de/pub/docs/security` FTP-Server mit Kryptodokumenten

`ftp://ftp.tu-clausthal.de/pub/unix/security` FTP-Server mit Unix Kryptotools

`ftp://ftp.darmstadt.dmg.de/pub/crypto` FTP-Server mit Kryptotools

`ftp://ftp.funet.fi/pub/crypt` FTP-Server mit Kryptotools

B.2 Der Inhalt der CD

Dieser Abschnitt enthält Links zu den einzelnen Verzeichnissen auf der CD. Da die Programme und Dokumente normalerweise nicht direkt aus dem Browser heraus gestartet werden können, verzweige ich nur zu dem Index jedes Verzeichnis, welches eine Beschreibung der in dem Verzeichnis vorhandenen Dateien enthält.

acrobatreader Adobe Acrobat Reader 4.0 (PDF Viewer)
- ⟼ Win9x — für Windows 9x
- ⟼ unix — für verschiedene Unix Varianten
- ⟼ mac — für Power Macintosh Computer

Entropie siehe Beschreibung in Abschnitt B.4

Libraries Kryptobibliotheken für C-Programmierer

Papers Online-Dokumente einiger Quellen meiner Arbeit
- ⟼ aes — AES Dokumente

feallina siehe Beschreibung in Abschnitt B.3
- ⟼ Bonus
  - fealprot.cpp — Protokollversion des Analyseprogramms
  - fea4ana.txt — Ein Probelauf des Programms
  - feallina.tar — unter Linux lauffähige Version der Analyse

B.3 Lineare Kryptoanalyse des FEAL-4: Das Programm

In Abschnitt 5.3 habe ich meine lin. Kryptoanalyse des FEAL-4 vorgestellt, die mit einer ausreichend großen Anzahl an bekannten Klartexten einen zum gesuchten Schlüssel äquivalenten finden kann. Auf der CD findet sich dieses Programm im Verzeichnis `/feallina`.

Ich habe das Programm unter C++ implementiert, wobei ich den Programmcode so gehalten habe, daß er auf jeder Plattform lauffähig sein sollte. Auf der CD finden sich kompilierte Versionen für Windows 9x und Linux mit glibc. Die Programmbeschreibung versuche ich so zu gestalten, daß keine C Kenntnisse für das Verstehen des Programms erforderlich sind.

Neben den wesentlichen Funktionen, die die lin. Kryptoanalyse implementieren, gibt es auch einige Datentypen und kleine Funktionen, die für den Ablauf notwendig sind.

Datentypen

Die Datentypen `PAIR`, `BLOCK` und `HALF` sind für die Darstellung der Daten zuständig, wobei ein `PAIR` aus zwei `BLOCK`s besteht, ein `BLOCK` aus zwei `HALF`s und ein `HALF` schließlich ein 32-Bit Integer, der aufgrund der 8-Bit Operationen auch byteweise angesprochen werden kann.

Für die Schlüsselsuche sind die Datentypen `KEY`, `EXTKEY` und `LISTE` zuständig. `KEY` ist ein 192-Bit Schlüssel, der gemäß der Spezifikation des modifizierten FEAL-4 aus sechs 32-Bit Integern besteht. `EXTKEY` ist eine Liste von `CAND` 32-Bit Werten, wobei ich `CAND` standardmäßig auf 100 gesetzt habe. Dieses Feld fängt alle während der Analyse anfallenden möglichen Kandidaten für einen Teilschlüssel auf. `LISTE` ist eine dynamische Liste, die jeweils für Phase zwei und vier des Angriffs notwendig ist. Siehe dazu die Beschreibung der Funktion `Analyse`.

Notwendige Funktionen

Die ersten fünf Funktionen dienen nur der nötigen Definition von Ausgaben und elementaren Rechnungen für die Datentypen `BLOCK` und `KEY`. Für `BLOCK` und `KEY` brauche ich neben der Ausgabe auch die Überprüfung der Gleichheit und für `KEY` zusätzlich noch das XOR. Doch nun zu den Funktionen:

`KEY mask(KEY)` Maskierung von Schlüsseln
: Diese Funktion wird am Ende der Analyse benötigt, um zu überprüfen, ob der gefundene Schlüssel in der richtigen Äquivalenzklasse liegen kann. Dazu werden nur die Bits behalten, die alle Repräsentanten der Klasse gemein haben. Unterscheiden sich der gesuchte Schlüssel und der gefundene nicht in diesen Bits, dann wird mit hoher Wahrscheinlichkeit der gefundene Schlüssel tatsächlich ein zum gesuchten Schlüssel äquivalenter sein.

`unsigned char rot2(unsigned char)` Rotation
: Diese Funktion implementiert den zyklischen Linksshift um zwei Stellen, der für die S-Boxen der F-Funktion benötigt wird.

`uint par(uint)` Parität eines 32-Bit Integers
: Diese Funktion wird immer mit maskierten Werten aufgerufen und wird für den Algorithmus 2 der lin. Kryptoanalyse benötigt.

Die Funktionen mf und FEAL

`HALF mf(HALF data, uint key)` Die Rundenfunktion des FEAL-4
: Die Rundenfunktion nimmt einen halben `BLOCK` und einen 32-Bit Integer als Schlüssel und verknüpft diese gemäß Abbildung 5.3. Diese Funktion funktioniert so nur auf Little-Endian Maschinen (Intel Prozessoren), da bei diesen Rechnern die einzelnen Bytes von rechts nach links im Speicher abgelegt werden. Für Big-Endian Maschinen (Suns, Macintosh) müssen die Bytes von links nach rechts durchnummeriert, also die entsprechenden Indizes vertauscht werden. Diese Funktion wird sowohl von der Funktion `FEAL` als auch von `algo2` aufgerufen.

`BLOCK FEAL(KEY inkey, BLOCK inplain)` Der eigentliche FEAL
: Diese Funktion implementiert die Struktur des modifizierten FEAL-4 und verschlüsselt damit einen Klartextblock unter dem Schlüssel *inkey* und gibt den Chiffretext zurück.

Die Funktion algo2

Die Funktion `algo2` implementiert eine leicht modifizierte Version des Algorithmus 2, die den Bedürfnissen der Analyse des FEAL-4 optimal angepaßt ist. In dieser Abwandlung wird zunächst die Parität aller Textbitmasken errechnet. Da dieses Ergebnis für alle zu untersuchenden Schlüsselbits stets konstant ist, können diese Operationen eingespart werden, was bei größeren Anzahlen an Blöcken die Durchführung von Algorithmus 2 wesentlich beschleunigt.

Anschließend werden alle Möglichkeiten für die betrachteten Schlüsselbits durchprobiert, wobei ein eventuell schon teilweise vorhandener Schlüssel berücksichtigt wird. Da bei der Analyse des FEAL-4 nur Approximationen verwendet werden, die Wahrscheinlichkeit Eins oder Null haben, nehme ich jeden Schlüsselkandidaten in eine Liste auf, bei dem sich für alle vorliegenden bekannten Klartexte stets die gleiche Parität ergeben hat. Es können maximal `CAND` Kandidaten zurückgegeben werden, also normalerweise 100. Die Funktion gibt schließlich diese Liste und die Anzahl der gefundenen Kandidaten zurück.

```
// Algorithmus 2 der linearen Kryptoanalyse
/* Input:
paare: bekannte Klartexte
input: Input der Rundenfunktion
keymask: Bitmaske der effektiven Schlüsselbits
pl-,prmask: Bitmaske der Klartextbits
cl-,crmask: Bitmaske der Chiffretextbits
outmask: Bitmaske der Rundenbits
bits: Anzahl der zu untersuchenden effektiven Schlüsselbits
key: Bereits ermittelter Rundenschlüssel

Output: Schlüsselkandidaten und check gibt Anzahl zurück */
```

Anmerkung: In der Regel wird erst ab einer Anzahl von mindestens 300 bekannten Klartexten nur noch genau ein Kandidat für den Teilschlüssel zurückgegeben. Dies liegt daran, daß die Approximation die siebten Bits benutzt. Die niedrigsten drei Bits beeinflussen die Summe der siebten Bits nur äußerst wenig, daher werden diese Bits bei wenigen Klartexten nicht zuverlässig erkannt. □

Die Funktion algocand

Diese Funktion setze ich als Startplattform für Algorithmus 2 bei der zweiten und dritten Approximation jeder Phase ein. Da der Algorithmus 2 bei der ersten Approximation höchstwahrscheinlich mehrere Schlüsselkandidaten zurückgegeben hat, müssen die beiden folgenden Approximationen mit allen Kandidaten durchgeführt werden, die bis dahin ermittelt worden sind. Daher hat die Funktion statt eines Schlüssels eine Schlüsselliste als Eingangsparameter. Für jeden dieser Kandidaten wird die entsprechende Approximation hindurchgeführt und alle neuen Kandidaten wieder in einer Liste gespeichert. Danach gibt es die neue Liste und die neue Anzahl der Kandidaten zurück.

Die Funktion Analyse

Diese Funktion ist der Dreh- und Angelpunkt des gesamten Programms. Sie führt die lin. Kryptoanalyse des FEAL-4 gemäß Abschnitt 5.3 durch und gibt schließlich einen zum gesuchten Schlüssel äquivalenten zurück, sofern der Angriff nicht bereits vorher abbricht.

In Phase 1 und 3 wird jeweils zuerst Algorithmus 2 mit den entsprechenden Bitmasken der dritten Gleichungen gestartet und als initialer Rundenschlüssel Null übergeben. Die zurückgegebene Liste wird nun jeweils als Grundlage an die Funktion `algocand` übergeben und Algorithmus 2 mit den Approximationen der ersten und vierten Gleichung durchgeführt. Sollten im Verlauf von Phase 1 und 3 keine Kandidaten mehr übrig sein, so wird die Analyse abgebrochen.

Phase 2 und 4 wird nun mit jedem Kandidaten für die Teilschlüssel K_1 bzw. K_4 durchgeführt. Um für jeden Kandidaten eigenständige Listen für die Teilschlüssel K_2 bzw. K_3 führen zu können, benutze ich eine dynamische Liste vom Typ `LISTE`, welches neben dem Kandidaten für Schlüssel K_1 bzw. K_4 eine Schlüsselliste für K_2 bzw. K_3 führt, sowie deren Größe und einen Link zum nächsten Glied der dynamischen Kette. Ein neues Glied in dieser Kette wird nur dann angelegt, falls am Ende von Phase 2 bzw. 4 noch Kandidaten übrig sind. Im Gegensatz zu Phase 1 bzw. 3 wird die Analyse nicht abgebrochen, falls keine Kandidaten für K_2 bzw. K_3 mehr übrig sind, sondern nur zum nächsten Kandidaten aus Phase 1 bzw. 3 übergegangen. Die Analyse wird erst abgebrochen, falls am Ende von Phase 2 und 4 die dynamische Kette leer ist.

Da am Ende von Phase 1 und 3 der Teilschlüssel K_1 und K_4 nicht komplett ermittelt worden ist, werden Phase 2 und 4 für alle Möglichkeiten der verbliebenen Bits durchgeführt. Da die Werte der Bits 7 und 31 von K_1 und K_4 nicht wichtig sind, setze ich sie pauschal auf Null. Dadurch schränke ich natürlich die Art der ausgegebenen Repräsentanten der Äquivalenzklasse ein, aber die Klasse wird ja durch jedes Element gleich gut repräsentiert. Diese Vereinfachung beschleunigt jedenfalls die Durchführung der gesamten Analyse auf ungefähr das zwei- bis dreifache.

Phase 5 wird schließlich mit allen Kandidaten aus Phase 2 und 4 durchgeführt. Zur Überprüfung des gefundenen Gesamtschlüssels verschlüssele ich alle Klartexte mit diesem Schlüssel und überprüfe, ob das Ergebnis mit den gegebenen Chiffretexten übereinstimmt. Der erste Kandidat, der diesen Test besteht, wird schließlich von der Funktion als gefundener Schlüssel zurückgegeben.

Modifikationen

Im Laufe der Entwicklung hatte ich das Programm so modifiziert, daß es mir alle Elemente der Äquivalenzklasse ausgibt, d.h. eigentlich die Unterschiede zum gesuchten Schlüssel in der Reihenfolge ihrer Entdeckung. Da es sich um 256 Schlüssel handelt, wird das Ergebnis in eine Datei ausgegeben. Durch Änderung der entsprechenden Programmzeile könnte man sich auch alle Schlüssel direkt ausgeben lassen, anstatt nur die Unterschiede aufzulisten. Dieses Programm sowie ein Probelauf ist auf der CD unter Programme/feallina/Bonus zu finden.

Testläufe und Empfehlungen

Um den Angriff in Aktion zu erleben, empfehle ich einige Probeläufe zu starten, drei werde ich im Folgenden auflisten.

```
Testlauf 1
----------
Lineare Kryptoanalyse des FEAL-4

unter Verwendung von 70 bekannten Klartexten
mit dem Schluessel
10df9f59 99c7353d eda98aeb 521afa92 2edf4c31 276222b1

Beginne mit Phase 1 des Angriffs
```

```
Beginne mit Phase 2 des Angriffs
Beginne mit Phase 3 des Angriffs
Beginne mit Phase 4 des Angriffs
Beginne mit Phase 5 des Angriffs mit
4 Kandidaten fuer K1 und K2,
3 Kandidaten fuer K3 und K4.
Ein aequivalenter Schluessel
10df9f59 19c7353d 6da98869 521afa12 2cdf4c33 256222b3
wurde gefunden!
Der gefundene Repraesentant unterscheidet sich in den Bits
00000000 80000000 80000282 00000080 02000002 02000002
von dem eigentlichen Schluessel
Benoetigte Zeit fuer die Analyse: 4 Sekunden


Testlauf 2
----------
Lineare Kryptoanalyse des FEAL-4

unter Verwendung von 70 bekannten Klartexten
mit dem Schluessel
a28c8f4b 6ebbc351 c07990a1 1d338efd d14df3d4 0c6f422f

Beginne mit Phase 1 des Angriffs
Beginne mit Phase 2 des Angriffs
Beginne mit Phase 3 des Angriffs
Beginne mit Phase 4 des Angriffs
Beginne mit Phase 5 des Angriffs mit
4 Kandidaten fuer K1 und K2,
4 Kandidaten fuer K3 und K4.
Ein aequivalenter Schluessel
228c8f4b 6cb9c351 40799223 1d338e7d d14df3d6 0c6f422d
wurde gefunden!
Der gefundene Repraesentant unterscheidet sich in den Bits
80000000 02020000 80000282 00000080 00000002 00000002
von dem eigentlichen Schluessel
Benoetigte Zeit fuer die Analyse: 6 Sekunden


Testlauf 3
----------
Lineare Kryptoanalyse des FEAL-4

unter Verwendung von 70 bekannten Klartexten
mit dem Schluessel
b1119dc6 0dc426d3 091b5647 6b90584d 0fe0115c d7c6f067

Beginne mit Phase 1 des Angriffs
Beginne mit Phase 2 des Angriffs
Beginne mit Phase 3 des Angriffs
Beginne mit Phase 4 des Angriffs
Beginne mit Phase 5 des Angriffs mit
1 Kandidaten fuer K1 und K2,
2 Kandidaten fuer K3 und K4.
```

```
Ein aequivalenter Schluessel
31119d46 0fc62451 091b5647 6b90584d 0de0115e d7c6f065
wurde gefunden!
Der gefundene Repraesentant unterscheidet sich in den Bits
80000080 02020282 00000000 00000000 02000002 00000002
von dem eigentlichen Schluessel
Benoetigte Zeit fuer die Analyse: 7 Sekunden
```

Ich empfehle eine Durchführung der Analyse mit 70 bekannten Klartexten, obwohl auch eine Durchführung mit weniger Klartexten möglich ist. Zu diesem Zweck müssen lediglich die Konstanten `PAIRS` und `CAND` in Zeile 12 und 13 geändert werden. Für eine Durchführung mit 30 Klartexten sind 100 Kandidaten noch ausreichend, möchte man aber nur 10–20 Klartexte benutzen, so sollte `CAND` deutlich erhöht werden, mindestens auf 1000. Allerdings steigt die durchschnittliche Dauer des Angriffs bei Reduzierung der verwendeten Klartexte exponentiell an. Die Analyse funktioniert so eventuell auch mit 10 Klartexten, dauert allerdings einige Minuten, wenn sie überhaupt erfolgreich ist. Es sollten mindestens 15 Klartexte verwendet werden, damit der Angriff stets gelingt.

B.4 Die Entropie meiner Arbeit

Nur den Rand meiner Diplomarbeit streifend ist der Begriff der Entropie von Shannon. Dieser Wert gibt an, wieviel Bits an Information ein Text pro Zeichen enthält. Da die meisten Computertexte den 8-Bit ASCII-Code verwenden, ist der maximale Informationsgehalt 8 Bit. Dieser Wert wird am nächsten von komprimierten Dateien erreicht; auch verschlüsselte Texte sollten diesen Wert erreichen, da beide ungefähr die Gleichverteilung als Wahrscheinlichkeitsmaß besitzen (sollten).

Definition B.1 *Zu einer Nachricht X soll das Wahrscheinlichkeitsmaß $\mathcal{P}_X$ gegeben sein durch*

$$\mathcal{P}_X(a) = \frac{\{\text{Häufigkeit des Zeichens } a \text{ in } X\}}{\{\text{Länge der Nachricht } X\}}.$$

Dann ist die **Entropie** *der Nachricht X definiert als*

$$E(X) := -\sum_a \mathcal{P}_X(a) \cdot \log_2 \mathcal{P}_X(a).$$

Das Programm `entropie.exe` auf der CD unter `/Entropie` berechnet diese Entropie einer Nachricht. Die Entropie des Quelltextes meiner Diplomarbeit beträgt z.B. 5,1.

Der Quelltext

```
//////////////////////////////////
//         Berechnung von       //
//       Shannon's Entropie     //
//           (c) 1999           //
//        by Tim Schneider      //
//////////////////////////////////
#include <iostream.h>
#include <stdio.h>
#include <math.h>

void main(int argc, char* argv[])
{
  // Checken, ob alles richtig
```

```
14    if (argc != 2)
15    {
16      cout << "Falsche Anzahl an Parametern!\n";
17      cout << "Aufruf mit:   entropie <Dateiname> \n";
18      return;
19    }
20    else {
21      int i, size=0; FILE *f; unsigned char c;
22      double entr=0, ar[256];
23      if ((f = fopen(argv[1],"rb")) == NULL)
24      {
25        cout << "Datei nicht gefunden!\n";
26        return;
27      }
28      // Jetzt kann es losgehen
29      for (i=0;i<256;i++) ar[i] = 0;
30      while (!feof(f))
31      {
32        fread(&c,1,1,f);
33        ar[c]++;size++;
34      }
35      fclose(f);
36      for (i=0;i<256;i++) ar[i] = ar[i] / (double)size;
37      for (i=0;i<256;i++) if (ar[i] !=0)
38        entr += (ar[i] * (log(ar[i]) / log(2)));
39      entr = - entr;
40      cout << "Die Entropie der Datei betraegt " << entr << '\n';
41    }
42  }
```

Beschreibung

Das Programm wird mit einem Kommandozeilenparameter gestartet, der die zu untersuchende Datei angibt. Die erste Hälfte des Programms bis Zeile 28 ist diesem Teil gewidmet. Existiert die angegebene Datei, wird in der While-Schleife die Größe der Datei und die Zeichenhäufigkeiten ermittelt. Mit diesen Informationen wird schließlich das Wahrscheinlichkeitsmaß in Zeile 36 berechnet.

In der zweiten Schleife wird für alle vorkommenden Buchstaben die Summe berechnet und zuletzt das Vorzeichen umgedreht. Die letzte Zeile gibt schließlich die Entropie aus.

Literaturverzeichnis

[Ada97] C. Adams: *Constructing Symmetric Ciphers Using the CAST Design Procedure*, Selected Areas in Cryptography, Kluwer Academic Publishers 1997, Seiten 71–104

[AES97] NIST Announcement: *Announcing Request for Candidate Algorithm Nominations for the AES*, Federal Register: September 12, 1997 (Vol. 62, Nr. 177), siehe beiliegendes Dokument `/Papers/aes/aes9709.htm`

[AES99a] NIST Pressemitteilung: *NIST Announces Encryption Standard Finalists*, Federal Register: August, 9, 1999 (G99-111), siehe beiliegendes Dokument `/Papers/aes/AESpressrelease-990809.pdf`

[AES99b] NIST Webpage: *AES Development Effort*, Stand 9. August 1999, auf der CD unter `/Papers/aes/aes_home.htm`

[AES99c] NIST Report: *Status Report on the First Round of Development of the AES*, Status Report vom 9. August 1999, siehe beiliegendes Dokument `/Papers/aes/r1report.pdf`

[AoOh96] K. Aoki und K. Ohta: *Differential-Linear Cryptanalysis of FEAL-8*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E79-A, No.1 Januar 1996

[AOMM98] K. Aoki, K. Ohta, M. Matsui und S. Moriai: *Linear Cryptanalysis of FEAL*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E81-A, No.1 Januar 1998, korrigierte Version siehe `/Papers/Linana/e-ieice.ps`

[AOM96] K. Aoki, K. Ohta und S. Moriai: *The Best Linear Expression Search of FEAL*, IEICE Transactions on Fundamentals, Vol E79-A, Nr. 1 Januar 1996

[Biha93] Eli Biham: *New Types of Cryptanalytic Attacks Using Related Keys*, Proceedings of Eurocrypt '93, Springer LNCS 765 (1994), Seiten 398–409

[Biha94] Eli Biham: *On Matsui's Linear Cryptanalysis*, Techreport CS0813, Technion - Computer Science Department, Haifa, Israel 1994

[Biha96] Eli Biham: *Cryptanalysis of Triple-Modes of Operation*, Techreport CS0885, Technion - Computer Science Department, Haifa, Israel 1996

[Biha98] Eli Biham: *Design Tradeoffs of the AES Candidates*, Slideshow auf der Asiacrypt '98, Oktober 1998, siehe beiliegendes Dokument `/Papers/aes/asiacrypt98-slides.ps`

[Bih99a] Eli Biham: *Comment on Selecting the Ciphers for the AES Second Round*, offener Brief an das NIST, April 1999, siehe beiliegendes Dokument `/Papers/aes/aes-comment-to-nist.ps`

[Bih99b] Eli Biham: *A Note on Comparing the AES Candidates*, revidierte Version April 1999, zuerst präsentiert auf der Asiacrypt '98, siehe beiliegendes Dokument `/Papers/aes/aes-comparing-revised.ps`

[BiSh90] Eli Biham und Adi Shamir: *Differential Cryptanalysis of DES-like Cryptosystems*, Proceedings of Crypto '90, Springer LNCS 537 (1991), Seiten 2–21

[BiSh93] Eli Biham und Adi Shamir: *Differential Cryptanalsis of the* DES , Springer Verlag, New York 1993

[BiKu98] A. Biryukov und E. Kushilevitz: *Improved Cryptanalysis of RC5*, Proceedings of Eurocrypt '98, Springer LNCS 1403 (1998), Seiten 85–99

[Broc90] Brockhaus Enzyklopädie Band 12, F.A. Brockhaus GmbH, Mannheim 1990

[CaWi92] K.W. Campbell und M.J. Wiener: DES *is not a Group*, Proceedings of Crypto '92, Springer LNCS 740 (1993), Seiten 512–520

[ChEv85] D. Chaum und J.-H. Evertse: *Cryptanalysis of DES with Reduced Number of Rounds - Sequences of Lineas Factors in Block Ciphers*, Proceedings of Crypto '85, Springer LNCS 218 (1986), Seiten 192–211

[CRRY98] S. Contini, R.L. Rivest, M.J.B. Robshaw und Y.L. Yin: *The Security of the RC6 Block Cipher*, RSA Laboratories, 20. August 1998, auf der CD unter `/Papers/aes/rc6/security.ps`

[Cop85] Don Coppersmith: *The Real Reason for Rivest's Phenomenon*, Proceedings of Crypto '85, Springer LNCS 218 (1986), Seiten 535f

[Cop94] Don Coppersmith: *The DES and its Strength Against Attacks*, IBM Journal of Research and Development, Vol. 38 No.3 Mai 1994, Seiten 243–250

[DGV95] J. Daemen, R. Govaerts und J. Vandewalle: *Correlation Matrices*, 2nd International Workshop on Fast Software Encryption 1994, Springer LNCS 1008 (1995), Seiten 275–285

[Daws91] E.P. Dawson: *Design and Cryptanalysis of Symmetric Ciphers*, Doktorarbeit am Information Security Research Centre, Faculty of Information Technology, Queensland University of Technology, Februar 1991

[DPa82] D.W. Davies und G.I. Parkin: *The Average Cycle Size of the Keystream in Output Feedback Encipherment*, Proceedings of Eurocrypt '82, Springer LNCS 149 (1983), Seiten 263–279

[DPr89] D.W. Davies und W.L. Price: *Security for Computer Networks*, 2nd Edition, Wiley & Sons, Chichester 1989

[Feis73] H. Feistel: *Cryptography and Computer Privacy*, Addison Wesley, 1982

[Feis75] H. Feistel, W.A. Notz und J.L. Smith: *Some cryptographic techniques for machine-to-machine data communications*, Proceedings of the IEEE, Vol. 63 (1975), Seiten 1545–1554

[Fell68] W. Feller: *An Introduction to Probability Theory and its Applications*, Wiley & Sons, 1968

[Fips46] *Data Encryption Standard*, Federal Information Processing Standards Publication 46, US Department of Commerce / National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1977

[Fips81] *DES Modes of Operation*, Federal Information Processing Standards Publication 81, US Department of Commerce / National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1980

[FuRi94] Fumy/Riess: *Kryptographie: Entwurf, Einsatz und Analyse symmetrischer Kryptoverfahren*, 2. Auflage Oldenbourg, München 1994

[GoOC94] J.D. Golić und L. O'Connor: *A Unified Markov Approach to Differential and Linear Cryptanalysis*, Proceedings of Asiacrypt '94, Springer LNCS 917 (1995), Seiten 387–397

[GoRe82] J. Gordon und H. Retkin: *Are Big S-Boxes Best?*, Proceedings of Eurocrypt '82, Springer LNCS 149 (1983), Seiten 257-262

[Hel80] M.E. Hellman: *A Cryptanalytic Time-Memory Tradeoff*, IEEE Transactions on Information Theory 26 (1980), Seiten 401–406

[HeLa94] M.E. Hellmann, S.K. Langford: *Differential-Linear Cryptanalysis*, Proceedings of Crypto '94, Springer LNCS 839 (1994), Seiten 17–25

[Iso91] ISO/IEC 10116: *Information Processing — Modes of operation for an n-bit block cipher algorithm*, International Organization for Standardization, Genf, Sschweiz, 1991

[Kahn67] D. Kahn: *The Codebreakers: The Story of Secret Writing*, Macmillan Publishing Co. New York 1967

[KaRo94] B.S. Kaliski, M.J.B. Robshaw: *Linear Cryptanalysis Using Multiple Approximations and FEAL*, 2nd International Workshop on Fast Software Encryption 1994, Springer LNCS 1008 (1995), Seiten 249–264

[KRS88] B.S. Kaliski, R.L. Rivest und A.T. Sherman: *Is the DES a group?*, Journal of Cryptology, vol.1 (1988), Seiten 3–36

[KaYi95] B.S. Kaliski und Y.L. Yin: *On Differential an Linear Cryptanalysis of the RC5 Encryption Algorithm*, Proceedings of Crypto '95, Springer LNCS 963 (1995), Seiten 171–184

[KaYi98] B.S. Kaliski und Y.L. Yin: *On the Security of the RC5 Encryption Algorithm*, RSA Laboratories Tech Report TR-602, September 1998, auf der CD unter `/Papers/rc5-report.pdf`

[KeSch96] J. Kelsey, B: Schneier: *Unbalanced Feistel-Networks and Block Cipher Design*, 3rd International Workshop on Fast Software Encryption 1996, Springer LNCS 1039 (1996), Seiten 121–144

[KSW96] J. Kelsey, B: Schneier und D. Wagner: *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER and Triple-DES*, Proceedings of Crypto '96, Springer LNCS 1109 (1996), Seiten 237–251

[Knud94] L.R. Knudsen: *Truncated and Higher Order Differentials*, 2nd international Workshop on Fast Software Encryption 1994, Springer LNCS 1008 (1995), Seiten 196–211

[KnBe95] L.R. Knudsen und T. Berson: *Truncated Differentials of SAFER*, 3rd International Workshop on Fast Software Encryption 1995, Springer LNCS 1039 (1995), Seiten 15–26

[KnMe96] L.R. Knudsen und W. Meier: *Improved Differential Attacks on RC5*, Proceedings of Crypto '96, Springer LNCS 1109 (1996), Seiten 216–228

[Lai92] X. Lai: *On the Design and Security of Block Ciphers*, ETH Series in Information Processing, J.L. Massey (Editor), vol. 1, Hartung-Gorre Verlag Konstanz, Technische Hochschule Zürich 1992

[Lai94] X. Lai: *Higher Order Derivatives and Differential Cryptanalysis*, Communication and Cryptography: Two Sides of One Tapestry, Kluwer Academic Publishing 1994

[LMM91] X. Lai, J.L. Massey und S. Murphy: *Markov Ciphers and Differential Cryptanalysis*, Proceedings of Eurocrypt '91, Springer LNCS 547 (1991), Seiten 17–38

[LHT97] J. Lee, H. Heys und S. Tavares: *Resistance of a CAST-like Encryption Algorithm to Linear and Differential Cryptanalysis*, Designs, Code and Cryptography, Vol. 12, Nr. 3 1997, Seiten 267–282

[Mat93] M. Matsui: *Linear Cryptanalysis Method for DES Cipher*, Proceedings of Eurocrypt '93, Springer LNCS 765 (1994), Seiten 386–397

[Mat94] M. Matsui: *The First Experimental Cryptanalysis of the DES*, Proceedings of Crypto '94, Springer LNCS 839 (1994), Seiten 1–11

[Mat94b] M. Matsui: *On Correlation Between the Order of S-Boxes and the Strength of DES*, Proceedings of Eurorypt '94, Springer LNCS 950 (1995), Seiten 366–375

[Mat96] M. Matsui: *New Structure of Block Ciphers with Provable Security Against Differential and Linear Cryptanalysis*, 3rd International Workshop on Fast Software Encryption, Springer LNCS 1039 (1996), Seiten 205–218

[MaYa92] M. Matsui, A. Yamagishi: *A New Method for Known Plaintext Attack of FEAL Cipher*, Proceedings of Eurocrypt '92, Springer LNCS 658 (1993), Seiten 81–91

[MeMa82] C.H. Meyer und S.M. Matyas: *Cryptography: A New Dimension in Computer Data Security*, Wiley & Sons, New York 1982

[MOV96] A.J. Menezes, P.C. van Oorschot, S. A. Vanstone: *Handbook of Applied Cryptography*, CRC Press, New York 1996

[Mur90] S. Murphy: *The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts*, Journal of Cryptology, vol.2 (1990), Seiten 145–154

[Nyb90] K. Nyberg: *Constructions of Bent Functions and Difference Sets*, Proceedings of Eurocrypt '90 (1991), Springer LNCS 473, Seiten 151–160

[Nyb91] K. Nyberg: *Perfect Nonlinear S-Boxes*, Proceedings of Eurocrypt '91, Springer LNCS 547 (1991), Seiten 378–386

[Nyb93] K. Nyberg: *Differentially Uniform Mappings for Cryptography*, Proceedings of Eurocrypt '93, Springer LNCS 765 (1994), Seiten 55–64

[Nyb94] K. Nyberg: *Linear Approximation of Block Ciphers*, Proceedings of Eurocrypt '94, Springer LNCS 950 (1995), Seiten 439–444

[Nyb95] K. Nyberg: *S-Boxes and Round Functions With Controllable Linearity and Differentially Uniformity*, 2nd International Workshop on Fast Software Encryption 1994, Springer LNCS 1008 (1995), Seiten 111–130

[Pren93] B. Preneel: *Analysis and Design of Cryptographic Hash-Functions*, PhD Thesis, Katholieke Universiteit Leuven, Januar 1993

[Riv95] R.L. Rivest: *The RC5 Encryption Algorithm*, 2nd International Workshop on Fast Software Encryption 1994, Springe LNCS 1008 (1995), Seiten 86–96

[RRSY98] R.L. Rivest, M.J.B. Robshaw, R. Sidney und Y.L. Yin: *The RC6 Block Cipher*, RSA Laboratories, 14. April 1998, auf der CD unter
`/Papers/AES/rc6/cipher.pdf`
revidierte Version 1.1 vom 20. August 1998 auf der CD unter
`/Papers/aes/rc6/rc6v11.pdf`

[Ruep84] R.A. Rueppel: *New Approaches to Stream Ciphers*, Doktorarbeit am Schweizer Institut für Technologie, 1984

[Ruep86] R.A. Rueppel: *Analysis and Design of Stream Ciphers*, Springer Verlag, 1986

[Schn96] B. Schneier: *Applied Cryptography*, 2nd Edition, Wiley & Sons, New York 1996

[Sel98] A.A. Selcuk: *New Results in Linear Cryptanalysis of RC5*, 5th International Workshop on Fast Software Encryption 1997, Springer LNCS 1372 (1998), Seiten 1–16

[Shan49] C.E. Shannon: *Communication Theory of Secrecy Systems*, Bell System Technical Journal, Vol. 28, Oktober 1949, Seiten 656–715

[SK97] T: Shimoyama und T. Kaneko: *Quadratic Relations of S-Boxes and its Application to the Linear Attack of Full Round DES*, Proceedings of Crypto '98, Springer LNCS 1462, Seiten 200–211

[Smit71] J.L. Smith: *The Design of Lucifer: A Cryptographic Device for Data Communications*, IBM Research Report RC 3326, IBM T.J. Wtson Watson Research Center, Yorktown Heights NY, 1971

[Stin95] D.R. Stinson: *Cryptography*, CRC Press, Boca Raton 1995

[Web85] A.F. Webster: *Plaintext / Ciphertext Dependencies in Cryptographic Algorithms*, Doktorarbeit an der Queen's University in Kingston, 1985