Kapitel 3
Blockchiffren und ihre Kryptanalyse

§1: Grundlagen

Bei den sogenannten klassischen oder symmetrischen Kryptosystemen gibt es einen zwischen Sender und Empfänger vereinbarten Schlüssel, der sowohl zur Ver- als auch zur Entschlüsselung benutzt wird. Je nach Vorgehensweise unterscheidet man zwei Arten von Chiffren:

- Stromchiffren *und*
- Blockchiffren.

Bei *Stromchiffren* wird ein Text oder sonstiger Datenstrom buchstaben- oder byteweise oder in welchen Einheiten auch immer die Daten ankommen chiffriert, wobei der Schlüssel natürlich von Buchstabe zu Buchstabe wechselt. *Blockchiffren* fassen die Daten zu Blöcken einer festen Länge zusammen und verschlüsseln diese Blöcke als ganzes.

Stromchiffren werden häufig angewandt zur Übermittlung von Satellitendaten, Satellitenfernsehen und ähnlichen kontinuierlichen Datenströmen; Blockchiffren haben ihre Bedeutung im Bankenbereich, sowohl im elektronischen Zahlungsverkehr wie auch zur Identifikation der Benutzer an Geldautomaten, zur Sicherung der Kommunikation zwischen Computern und in ähnlichen Anwendungen.

a) Die Sicherheit einer Blockchiffre

Typischerweise faßt eine Blockchiffre die zu übermittelnden Daten zusammen zu Blöcken von 64 Bit (bei alten Systemen) oder 128 Bit (bei neueren). Jeder Block wird in derselben (von einem Schlüssel abhängigen) Weise durch einen anderen Block derselben Länge ersetzt; ist $\mathcal{B}$

die Menge aller Blöcke der vorgegebenen Länge, so besteht das Kryptosystem also aus bijektiven Abbildungen

$$T_s\colon \mathcal{B} \to \mathcal{B}; \qquad s \in S\,.$$

Jedes T_s ist damit ein Element der symmetrischen Gruppe $S_{\#\mathcal{B}}$, im Gegensatz zu den monoalphabetischen Buchstabensubstitutionen können wir allerdings nicht mehr alle Permutation zulassen: Falls etwa $\mathcal{B}$ aus Blöcken der Länge 64 Bit besteht, hat $\mathcal{B}$ immerhin schon 2^{64} Elemente, es gibt also $2^{64}!$ Permutationen. Nach STIRLING ist in erster Näherung

$$\log_2(2^{64}!) \approx 2^{64} \log_2 2^{64} = 64 \cdot 2^{64} = 2^{72}\,,$$

zur Festlegung einer Permutation braucht man also ungefähr 2^{72} Bit oder 2^{69} Byte oder 2^{66} kB oder 2^{63} MB oder 2^{60} GB oder 2^{57} TB oder ..., jedenfalls viel zu viel.

Aus diesem Grund muß die Abbildungsvorschrift konkreter definiert werden; allerdings sollten die Substitutionen wenn möglich immer noch wie zufallsverteilt in der symmetrischen Gruppe liegen. Beispielsweise sollten sie keine Untergruppe bilden, denn die Kenntnis von Erzeugenden und Relationen dieser Untergruppe könnte einem Gegner Ansatzpunkte zur Entschlüsselung geben.

Ansonsten hängt die Sicherheit einer jeden Blockchiffre stark vom Inhalt der Klartexte und der Schlüssellänge ab: Für deutschen Klartext mit ASCII-Kodierung der Buchstaben etwa enthält ein Block von 64 Bit acht Buchstaben; geht man von einer Redundanz der deutschen Sprache in der Größenordunung von etwa 70% aus, enthält er also eine Information von

$$8 \times 0{,}3 \times \log_2 26 \approx 11{,}28\,\text{Bit}\,;$$

tatsächlich übermittelt werden 64 Bit Chiffretext, so daß in erster Näherung ein Block etwa 52 Bit Schlüsselinformation liefert. Bei einer Schlüssellänge von 64 Bit reichen dem BAYESschen Gegner also im allgemeinen bereits zwei Blocke zur Dekodierung, bei einer Schlüssellänge von 128 Bit braucht er drei.

Sicherer ist ein System, das komprimierten Text übermittelt, da hier die Redundanz bereits deutlich reduziert ist; man muß allerdings beachten, daß bei vielen gängigen Komprimierungsprogrammen eine ganze Reihe von Anfangsbytes auf feste Werte gesetzt werden, so daß der BAYESsche Gegner mit bekanntem Klartext arbeiten kann.

Für die praktischen Beurteilung einer Blockchiffre hat sich heutzutage folgendes Kriterium durchgesetzt:

- Die Schlüssellänge muß so groß sein, daß systematisches Durchprobieren für die zu erwarteten Gegner unrealistisch ist
- Das Verfahren muß so gewählt werden, daß es keine Attacke gibt, die nennenswert schneller ist als systematisches Durchprobieren

Beide Kriterien sind eher vage formuliert: Das erste hängt davon ab, daß man abschätzen kann, über welche Hardware ein Gegner verfügt, was zumindest bei Gegnern, die für Regierungen arbeiten, kaum möglich ist; beim zweiten müßte man, um wirklich sicher zu sein, die untere Schranken für die Laufzeit eines *beliebigen* Algorithmus finden, der die Chiffre entschlüsseln kann, was weit jenseits der heutigen Möglichkeiten der Komplexitätstheorie liegt.

In der Praxis verwendet sollte man sich auf Systeme beschränken, die bereits seit einigen Jahren untersucht sind und dabei keine Schwächen gegenüber kryptanalytischen Attacken zeigten; das Restrisiko, daß jemand nicht dokumentierte bessere Verfahren kennt, muß (soweit das möglich ist) durch zusätzliche Sicherheitsfaktoren bei der Schlüssellänge und der Komplexität des Algorithmus kompensiert werden.

b) Beispiel: Hill-Chiffren

Ein möglicher Ansatz für eine Blockchiffre wäre etwa, daß man die Substitution der Blöcke durch lineare Gleichungen angibt: Für $\mathbf{x} = (x_1, \ldots, x_N)$ sei die Verschlüsselung $\mathbf{y} = (y_1, \ldots, y_N)$ mit

$$y_i = \sum_{j=1}^{N} a_{ij} x_j + b_i, \qquad a_{ij}, b_j \in \mathbb{F}_2 ,$$

wobei N natürlich die Blocklänge bezeichnet und $\mathbb{F}_2 = \{0, 1\}$ den Körper mit zwei Elementen mit Rechenoperationen modulo zwei.

Der Schlüssel hätte dann die Länge $N(N+1)$, was handhabbar ist; man muß aber darauf achten, daß die Matrix $A = (a_{ij})$ invertierbar ist, da es sonst keine eindeutige Entschlüsselung gibt. Solche Chiffren heißen HILL-Chiffren nach LESTER HILL, der sie allerdings nicht über $\mathbb{F}_2^N$ betrachtete, sondern über $(\mathbb{Z}/26)^N$, wo sie eine Verallgemeinerung der CAESAR-Chiffren darstellen.

Unsicher sind sie sowohl über $\mathbb{F}_2$ als auch über $\mathbb{Z}/26$, sofern der Gegner über hinreichend viel korrespondierenden Klartext und Chiffretext verfügt und den GAUSS-Algorithmus zur Lösung linearer Gleichungssysteme kennt: Mit $N+1$ linear unabhängigen Klartextblöcken kann er über einem Körper die Spalten von A sowie auch b berechnen; bei einem Ring wie $\mathbb{Z}/26$ kann es noch Probleme mit gemeinsamen Teilern geben, die aber den Aufwand auch nicht wesentlich größer machen. Auch ohne Kenntnis von Klartext (die man in der Praxis nie ausschließen kann) gibt es Verfahren zur Kryptoanalyse.

c) Diffusion und Konfusion

Nach CLAUDE SHANNON sollte ein Kryptosystem auf zwei Techniken beruhen: **Diffusion** und **Konfusion**.

Diffusion soll die Redundanz des Klartexts möglichst weiträumig über den Chiffretext verteilen. Dies könnte beispielsweise durch eine Permutation der Klartextbuchstaben erreicht werden (was die deutsche Armee im ersten Weltkrieg als hauptsächliches Kryptoverfahren benutzte); allerdings ist diese Permutation allein kryptographisch ziemlich unsicher: Da man beispielsweise weiß, daß auf ein „C“ in deutschen Klartext praktisch immer ein „H“ folgt oder auf ein „Q“ ein „U“, kann man bei hinreichend viel Chiffretext solche Permutationen leicht rückgängig machen. Bei guten Kryptoverfahren werden sie daher nur angewandt als eine von mehreren Komponenten. Diffusion wird auch erreicht, wenn man dafür sorgt, daß jedes Klartextbit möglichst viele Bit des Chiffretexts beeinflußt.

Konfusion soll dafür sorgen, daß der Zusammenhang zwischen Schlüssel und Chiffretext möglichst undurchsichtig ist; der Kryptanalytiker soll aus dem Chiffretext nur wenig Information über die Schlüsselverteilung gewinnen können. Der *one time pad* ist ein Beispiel dafür, wie Konfusion zu perfekter Sicherheit führen kann; die anderen Beispiele aus dem ersten Kapitel zeigen, daß reine Substitution bei kürzeren Schlüssellängen nicht sonderlich sicher ist.

Moderne Kryptosysteme beruhen darauf, daß Diffusion und Konfusion *in Kombination* zu deutlich größerer Sicherheit führen als jedes für sich allein.

§2: Feistelnetzwerke und der Aufbau des DES

a) Feistelnetzwerke

Bis vor kurzem waren fast alle gebräuchlichen Blockchiffren sogenannte FEISTEL-Netzwerke, die in *Runden* arbeiten: In jeder Runde wird nur ein Teil des Blocks und ein Teil des Schlüssels verwendet, aber nachdem alle der (typischerweise sechzehn) Runden abgearbeitet sind, ist sichergestellt, daß die Ausgabe von allen Schlüsselbits und allen Klartextbits abhängt.

HORST FEISTEL entwickelte ab etwa 1960 bei IBM die ersten Blockchiffren, insbesondere auch das Lucifer-System, auf dessen Grundlage DES entwickelt wurde.

Sei $2N$ die Blocklänge des zu betrachtenden Systems. Dann gibt es eine Funktion

$$f\colon \mathbb{F}_2^k \times \mathbb{F}_2^N \to \mathbb{F}_2^N \;,$$

die sogenannte FEISTEL-Funktion, die aus k Schlüsselbits und N Nachrichtenbits wieder N Bits produziert. Diese wird folgendermaßen angewandt: Vor Beginn der i-ten Runde wird die Nachricht aufgeteilt in eine linke Hälfte L und eine rechte Hälfte R. Die i-te Runde ersetzt den Block (L, R) durch

$$\big(F(s_i, R) \oplus L, R\big) \;,$$

wobei $\oplus$ die Addition im Vektorraum $\mathbb{F}_2^N$ bezeichnet und s_i den Schlüssel der i-ten Runde. Nach jeder außer der letzten Runde werden anschließend die linke und rechte Hälfte miteinander vertauscht – ansonsten würde ja die rechte Hälfte den gesamten Algorithmus unverändert durchlaufen.

Die FEISTEL-Funktion dient somit zur *Konfusion;* die Auswahl der Schlüsselbits für die jeweiligen Runden sowie auch die Vertauschung von linker und rechter Seite in jeder Runde zur Diffusion.

b) Aufbau des DES

Als Beispiel betrachten wir den 1977 in USA als Standard eingeführten DES (Data Encryption Standard), der unter anderem auch unserem EC-Kartensystem zugrunde liegt. Er arbeitet mit einem Schlüssel der Länge 56 Bit (der mit acht Prüfbits auf 64 Bit verlängert wird, indem man an je sieben Schlüsselbits ein Paritätsbit anhängt) auf Blöcken der Länge 64.

Zunächst werden die Datenwörter

$$(x_1, x_2, \ldots, x_{64})$$

einer Anfangspermutation unterzogen, d.h. das Wort wird ersetzt durch

$$(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(64)}) \;,$$

wobei die Folge der Zahlen $\pi(1), \ldots, \pi(64)$ der folgenden Tabelle entnommen wird:

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

Das Wort geht also auf

$$(x_{58}, x_{50}, x_{42}, \ldots, x_{23}, x_{15}, x_7) \,.$$

Danach beginnen die sechzehn Runden.

Die Anfangspermutation hat keine kryptographische Funktion: Da sie nicht vom Schlüssel abhängt und allgemein bekannt ist, kann sie jeder Kryptanalytiker leicht rückgängig machen. Ihr Sinn bestand anscheindend in erster Linie darin, Software-Attacken zu erschweren, denn Permutation sind aufwendig zu programmieren. (Bei Hardware-Implementierungen sind Permutationen natürlich sehr einfach und schnell zu realisieren.)

Zur Definition der FEISTEL-Funktion f dienen acht sogenannte S-Boxen (S = Substitution), die als Wertetabellen einem Eingabewort aus sechs Bit einen vier Bit langen Funktionswert zuordnen.

Diese Wertetabellen sind folgendermaßen angeordnet: Das Eingabewort mit seinen sechs Bit wird geschrieben als (a, m, e), wobei a das Anfangsbit, e das Endbit und m die aus vier Bit bestehende Mitte ist. Diese wird als Zahl zwischen null und fünfzehn aufgefaßt, genau wie auch die Ausgabe der S-Box. Die S-Box wird angegeben durch vier Zeilen, die mit den verschiedenen Möglichkeiten für das Paar (a, e) indiziert sind, und die für die sechszehn Werte von m die Ausgabewerte enthalten:

Box 1

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0 1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1 0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1 1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Box 2

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0 1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1 0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1 1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Box 3

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0 1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1 0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1 1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Box 4

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0 1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1 0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1 1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Box 5

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0 1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1 0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1 1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Box 6

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0 1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1 0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1 1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Box 7

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0 1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1 0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1 1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Box 8

$a\ e$	m = 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0 1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1 0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1 1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Diese S-Boxen werden zusammengesetzt zu einer Funktion

$$F\colon \mathbb{F}_2^{48} \to \mathbb{F}_2^{32}$$

in der offensichtlichen Weise: Ein Vektor der Länge 48 wird aufgeteilt in acht Vektoren der Länge sechs; auf den ersten davon wird die erste S-Box angewandt, auf den zweiten die zweite *usw;* dabei entstehen acht Vektoren der Länge vier, die zum Ergebnisvektor der Länge 32 zusammengesetzt werden.

Nach diesem Konfusionsschritt folgt noch ein Diffusionsschritt: Die Komponenten des Vektors werden untereinander permutiert mittels einer Permutation aus S_{32}, die durch folgende Wertetabelle geben ist:

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

Diese Funktion F wird folgendermaßen eingesetzt: Zunächst wird die rechte Hälfte R der Eingabe der jeweiligen Runde auf 48 Bit vergrößert, indem man einen Vektor $(x_1, \ldots, x_{32})$ ersetzt durch

$$(x_{\tau(1)}, \ldots, x_{\tau(48)}),$$

wobei die Werte von τ der folgenden Tabelle entnommen werden:

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						

Das Ergebnis dieser Aufblähung wird zum Schlüssel der i-ten Runde addiert (in $\mathbb{F}_2^{48}$, d.h. die Addition ist ein logisches XOR).

Die kryptographische Funktion dieser Aufblähung ist natürlich wieder eine Diffusion; sie sorgt dafür, daß ein Eingabebit möglichst viele Ausgabebits beeinflußt.

Damit kommen wir zur Verwendung des Schlüssels im Algorithmus. Der Schlüssel hat, wie bereits erwähnt, 56 Bit, wird aber mit 64 Bit gespeichert, wobei jedes achte Bit ein Paritätsbit ist, d.h. die Summe (in $\mathbb{F}_2$) der sieben davorstehenden Bits. Wir numerieren die Schlüsselbits daher von eins bis 64, verwenden aber nur die nicht durch acht teilbaren Indizes.

Aus dem Schlüssel werden zunächst zwei Schlüssel der Länge 28 extrahiert, bestehend aus den folgenden Komponenten:

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

und

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

Die so erhaltenen Teilschlüssel werden vor jeder Runde noch zirkulär nach links verschoben, und zwar vor der i-ten Runde um nochmals a_i Bit gegenüber der vorherigen Runde, wobei a_1 bis a_{16} die Zahlenfolge

$$1\ 1\ 2\ 2\ 2\ 2\ 2\ 2\ 1\ 2\ 2\ 2\ 2\ 2\ 2\ 1$$

ist. Man beachte, daß die Summe dieser Zahlen gleich 28 ist, jeder der Halbschlüssel wird also in den sechzehn Runden einmal komplett zyklisch verschoben.

Nachdem die beiden Teilschlüssel so präpariert sind und aneinandergehängt einen 56 Bit-Schlüssel $(s_1, \ldots, s_{56})$ bilden, wird daraus ein 48 Bit-Schlüssel für die i-te Runde gewählt, bestehend aus den folgenden Komponenten:

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

Die Rundenschlüssel werden also in einem reinen Diffusionsverfahren aus dem Gesamtschlüssel berechnet.

Die FEISTEL-Funktion f berechnet die Summe aus den 48 aufgeblähten Nachrichtenbits und den 48 Schlüsselbits der i-ten Runde und setzt diesen Vektor aus F_2^{48} in F ein; der Funktionswert ist der Wert der FEISTEL-Funktion.

Dieses Spiel wird insgesamt sechzehnmal gespielt, wobei nach der letzten Runde keine Vertauschung von links und rechts mehr stattfindet.

Danach wird nur noch die Anfangspermutation rückgängig gemacht; die inverse Permutation hat die Wertetabelle

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

Man beachte, daß DES (wie jedes (balancierte) FEISTEL-Netzwerk) in jeder Runde nur einen halben Block verändert; die andere Hälfte bleibt erhalten. Schreiben wir den Nachrichtenblock nach Anwendung der Anfangspermutation also in der Form (m_0, m_1), so wird in der i-ten Runde (m_{i-1}, m_i) zu (m_i, m_{i+1}) mit

$$m_{i+1} = m_{i-1} \oplus f(s_i, m_i)\,,$$

wobei wieder s_i den Schlüssel der i-ten Runde bezeichnet. Das Ergebnis der sechzehn Runden, abgesehen von der Endpermutation, ist dann allerdings nicht (m_{16}, m_{17}), sondern (m_{17}, m_{16}), da nach der letzten Runde die Hälften nicht mehr miteinander vertauscht werden.

Der Grund dafür liegt in der Entschlüsselung: Wegen

$$m_{i+1} = m_{i-1} \oplus f(s_i, m_i) \Leftrightarrow m_{i-1} = m_{i+1} \oplus f(s_i, m_i)$$

läßt sich die Verschlüsselung bei Kenntnis des Schlüssels leicht rückgängig machen, sogar mit derselben Hardware. Da aber vor dem ersten Verschlüsselungsschritt die Hälften nicht vertauscht werden, sollten sie es dann auch nach dem letzten nicht mehr werden, denn die Entschlüsselung läuft ja rückwärts durch die Runden.

§ 3: Designkriterien und Kryptoanalyse des DES

Die Beschreibung des DES im vorigen Paragraphen läßt den Leser zunächst wohl mit ziemlicher Verwirrung zurück und es erscheint schwierig, irgendeine Aussage über die kryptographische Sicherheit des Verfahrens zu machen. In der Tat wurde diese von Anfang an sehr kontrovers diskutiert.

a) Geschichtliche Entwicklung

Als das damalige *National Bureau of Standards* der USA (heute National Institute of Standards and Technology, NIST) im Januar 1977 den DES als Standard veröffentlichte (mit einer auf zehn Jahre veranschlagten Laufzeit) enthielt das Dokument im wesentlichen nur die hier reproduzierten Angaben; sowohl IBM als auch die National Security Agency (NSA) lehnten es ab, die Kriterien zu benennen, nach denen die S-Boxen und die Permutationen konzipiert worden waren.

Dies führte schnell auf den Verdacht, daß DES möglicherweise eine nur IBM und NSA bekannte „Falltür“ enthält, mit deren Hilfe eine Entschlüsselung ohne Schlüssel mit vertretbarem Aufwand durchgeführt werden kann. Außerdem gab es bereits damals Kritik an der mit 56 Bit sehr kurzen Schlüssellänge: Das Vorgängersystem LUCIFER hatte eine Schlüssellänge von 128 Bit, im militärischen Bereich waren Systeme mit mehr als zehn mal so langen Schlüsseln nichts Ungewöhnliches.

M.E. HELLMAN: A cryptanalytic time-memory tradeoff, *IEEE Trans. Inf. Theory* **26** (1980), 401–406

schlug ein Verfahren vor, mit dem durch eine Kombination von Vorberechnungen und Probieren die Komplexität der Schlüsselsuche von 2^{56} mit großer Erfolgswahrscheinlichkeit auf ungefähr die Kubikwurzel dieser Zahl reduziert werden konnte; als Baupreis seiner Maschine schätzte er zehn Millionen Dollar, als Zeitrahmen für die Vorberechnungen ungefähr ein Jahr. Da die NSA erheblich größere Geldmittel als nur zehn Millionen Dollar einsetzen kann, bestärkte dies den Verdacht, daß sie DES selbst dann knacken kann, wenn der Algorithmus keine Falltür enthalten sollte.

Im Laufe der Jahre wurden einige der Designkriterien durch *reverse engineering* gefunden; einige dann auch freiwillig veröffentlicht. Erst 1994 veröffentlichte einer der ursprünglichen Entwickler bei IBM die, wie er sagt, vollständige Liste der kryptographisch relevanten Kriterien in

D. COPPERSMITH: The Data Encryption Standard (DES) and its strength against attacks, *IBM J. Res. Develop.* **38** (1994), S. 243–250

– nachdem die kryptoanalytische Technik, gegen die diese Kriterien schützen sollten, auch in der offenen Literatur erschienen war. Dabei zeigte sich, daß DES mit seinen nur 56 Bit zumindest gegen diese Technik eine eher größere Sicherheit bietet als Lucifer mit seinen 128 Bit und daß die Sicherheit von DES nicht unbedingt erhöht würde, indem man für jede der sechzehn Runden einen neuen 48 Bit-Schlüssel verwendet, so daß man insgesamt eine Schlüssellänge von $16 \times 48 = 762$ Bit hätte. NSA hielt diese Technik damals für so wichtig für den Angriff auf gegnerische Systeme, daß die speziell dagegen eingesetzten Designkriterien „aus Gründen der nationalen Sicherheit" geheimgehalten wurden.

Die Technik, um die es hier geht, war bei IBM um 1974 unter dem Namen *T attack* bekannt; in der offenen Literatur erschienen erste Ansätze dazu ab etwa 1988, vollständige Beschreibungen erschienen ab 1990 unter dem Namen *differentielle Kryptoanalyse.* Bevor wir sie genauer betrachten, wollen wir uns zunächst die inzwischen bekannten Designkriterien des DES ansehen.

b) Designkriterien

D. COPPERSMITH nennt in der oben zitierten Arbeit folgende Designkriterien für die S-Boxen (und sagt, daß dies *alle* kryptographisch relevanten gewesen seien; der Rest habe nur mit Implementierungsfragen zusammengehängt):

(S1) Jede S-Box hat sechs Eingabe- und vier Ausgabebits.

(S2) Kein Ausgabebit einer S-Box sollte zu nahe bei einer linearen Funktion der Eingabebits liegen.

(S3) Bei festgehaltenem linken und rechtem Bit der Eingabe sollte jeder der sechszehn möglichen Ausgabewerte genau einmal vorkommen.

(S4) Wenn sich zwei Eingaben einer S-Box um genau ein Bit unterscheiden, müssen sich die Ausgaben um mindestens zwei Bit unterscheiden.

(S5) Wenn sich zwei Eingaben einer S-Box genau in den beiden mittleren Bits unterscheiden, müssen sich die Ausgaben um mindestens zwei Bit unterscheiden.

(S6) Wenn sich zwei Eingaben einer S-Box in ihren beiden Anfangsbits, nicht aber in ihren beiden Endbits unterscheiden, müssen die Ausgaben verschieden sein.

(S7) Für jede von null verschiedene Differenz Δ zwischen zwei Eingaben dürfen höchstens acht der 32 Paare mit Differenz Δ auf dieselbe Differenz zwischen den Ausgaben führen.

(S8) Ähnlich zu *(S7),* aber mit stärkeren Eigenschaften für den Fall gleicher Ausgaben, wenn in einer Runde drei S-Boxen „aktiv" sind. *(s.u.)*

Die Permutation aus S_{32}, die in jeder FEISTEL-Funktion als Abschluß ausgeführt wird, sollte folgenden Kriterien genügen:

(P1) Die vier Ausgabebits einer S-Box werden so verteilt, daß in der nächsten Runde zwei von ihnen mittlere Bits der Eingabe einer S-Box sind und die beiden anderen nicht (d.h. die kommen an Position 1, 2, 5 oder 6).

(P2) Die vier Ausgabebits einer S-Box sind in der nächsten Runde Eingaben zu sechs verschiedenen S-Boxen; keine zwei von ihnen sind Eingabe derselben S-Box.

(P3) Für zwei (nicht notwendigerweise verschiedene) S-Boxen j, k gilt: Wenn ein Ausgabebit von j als eines der beiden mittleren Bits an k weitergegeben wird, kann kein Ausgabebit von k als mittleres Bit an j weitergegeben werden. Insbesondere darf also kein Ausgabebit von j an j selbst als mittleres Bit weitergegeben werden.

Der Sinn einiger dieser Kriterien ist unmittelbar einsichtig: *(S1)* etwa kommt daher, daß mit der Technologie von 1974 größere S-Boxen dazu geführt hätten, daß man den Algorithmus nicht auf einem Chip untergebracht hätte.

(S2) ist selbstverständlich: Da die S-Boxen der einzige nichtlineare Bestandteil des Algorithmus sind, müssen sie nichtlinear sein; ansonsten hätten wir eine (leicht zu knackende) HILL-Chiffre. Wenn einzelne Bits lineare Funktionen der Eingabebits wären, hätten wir möglicherweise für einzelne Ausgabebits des Algorithmus lineare Zusammenhänge mit den Eingabebits, was dazu führen würde, daß man zumindest einen Teil der Chiffre als HILL-Chiffre betrachten kann und damit die Komplexität des Algorithmus reduziert. Ähnlich verhält es sich, wenn Funktionen nicht exakt, aber doch ungefähr linear sind.

Die restlichen Kriterien sind in erster Linie dazu da, um die Diffusion zu fördern: Für zwei verschiedene Eingaben W, W' in Runde i sagen wir, eine S-Box sei *aktiv,* wenn sie für W und W' verschiedene Ausgaben liefert. Es muß nicht in jeder Runde aktive S-Boxen geben, aber die obigen Kriterien sollen dafür sorgen, daß im Durchschnitt über alle Runden möglichst viele S-Boxen pro Runde aktiv sind; wie man zeigen kann, sind es im Durchschnitt mindestens 1,6.

(Bei *(S7)* sind die Zahlen, so wie sie genannt wurden, offensichtlich um den Faktor zwei zu klein: Es gibt 64 Paare mit vorgegebener Differenz Δ, und für $\Delta \neq 0$ dürfen dann wohl höchstens 16 davon auf denselben Ausgabewert führen.)

Bevor wir solche Fragen vertiefen können, müssen wir uns zunächst mit der kryptoanalytischen Attacke beschäftigen, vor der dies schützen soll:

c) Differentielle Kryptoanalyse

Ihre Grundidee besteht darin, daß man nicht von einzelnen Klartextblöcken ausgeht, sondern von Paaren (W, W') aus zwei Klartextblöcken. Diese werden aufgefaßt als Elemente von $\mathbb{F}_2^{64}$; da über dem Körper mit zwei Elementen Addition gleich Subtraktion ist, bezeichnen wir die Differenz zwischen den beiden Nachrichten als $W \oplus W'$. Praktisch handelt es sich hier einfach um das bitweise XOR zwischen den beiden Blöcken.

DES unterzieht die beiden Worte zunächst der Anfangspermutation; da XOR eine bitweise Operation ist, wird dabei auch die Differenz $W \oplus W'$ dieser Permutation unterzogen. Danach werden die rechten Hälften der entstandenen Nachrichten betrachtet; ihre Differenz ist natürlich einfach

die rechte Hälfte der permutierten Differenz. Die entstandenen 32 Bit-Worte werden durch Bitauswahl auf 48 Bit Worte V, V' aufgebläht; auch diese Aufblähung ist kompatibel mit der Differenzbildung.

Als nächstes kommt der Schlüssel ins Spiel; sowohl V als auch V' werden zum 48-Bit Schlüssel s_1 der ersten Runde addiert; dann gehen die Ergebnisse $V \oplus s_1$ und $V' \oplus s_1$ in acht 6 Bit Stücke aufgespalten in die acht S-Boxen. Die Differenz zwischen den beiden Eingaben ist

$$(V \oplus s_1) \oplus (V' \oplus s_1) = (V \oplus V') \oplus (s_1 \oplus s_1) = V \oplus V' ,$$

d.h. der Schlüssel ist herausgefallen.

Nun kommen die S-Boxen ins Spiel. Falls diese linear wären, wäre die Differenz ihre Ausgabe für zwei gegebene Eingabewerte nur von der Differenz der Eingabewerte abhängig, aber da es gerade der Zweck der S-Boxen ist, die Verschlüsselung nichtlinear zu machen, können wir natürlich nicht erwarten, daß wir hier auch nur bei einer einzigen S-Box eine lineare Funktion finden: Schon die ersten experimentellen Untersuchungen von DES befaßten sich mit etwaigen linearen Zusammenhängen zwischen einzelnen Ausgabebits sowohl einer S-Box wie auch des gesamten DES und der jeweiligen Eingabe, und keine konnte eine lineare Funktion finden.

Nach der Anwendung der S-Boxen können wir also nicht mehr sagen, was die Differenz der Ausgabewerte ist, obwohl wir die Differenz der Eingabewerte auch unabhängig vom Schlüssel kennen.

Trotzdem zeigt sich, daß wir zumindest gewisse Informationen über die Differenz haben: Bei sechs Eingabebits und vier Ausgabebits pro S-Box müssen von den $2^6 = 64$ Eingabepaaren (E, E') mit einer gegebenen Differenz ΔE im Durchschnitt jeweils vier auf jede der sechzehn möglichen Differenzen ΔA der Ausgabewerte A, A' führen. Im Einzelnen gibt es allerdings beträchtliche Schwankungen:

Für $\Delta E = 0$ ist natürlich auch $\Delta A = 0$, denn dasselbe Wort kann nicht auf zwei verschiedene Weisen verschlüsselt werden. Aber auch für andere Werte von ΔE gibt es keine Gleichverteilung der Ausgabewerte: Für $\Delta E = 100100$ etwa ergibt sich für die (dezimal geschriebenen)