Kapitel 3
Klassische Blockchiffren

Im vorigen Kapitel haben wir gesehen, daß alle dort behandelten Kryptoverfahren mit Ausnahme des *one time pad* nicht einmal einfachsten Sicherheitsanforderungen genügen; der *one time pad* wiederum erfordert einen derart großen Aufwand beim Schlüsselaustausch, daß er für die meisten aktuellen Anwendungen wie etwa dem Handel über das Internet nicht in Frage kommt.

Nun hat sich aber bereits KERCKHOFFS mit *praktischer* Sicherheit begnügt, und in den meisten Fällen werden auch wir uns damit begnügen müssen. Zwei Ansätze bieten sich an:

Als erstes können wir den *one time pad* imitieren, indem wir zwar keinen wirklich zufälligen Einmalschlüssel von der Länge der Nachricht verwenden, dafür aber einen solchen Schlüssel erzeugen aus einem Anfangsschlüssel handhabbarer Länge und einem Algorithmus, der daraus eine zufällig aussehende Folge produziert. Solche Algorithmen bezeichnet man als *Pseudozufallsgeneratoren*, denn tatsächlich ist die erzeugte Folge ja nicht zufällig, sondern wird streng deterministisch aus dem Anfangsschlüssel berechnet.

Solche Folgen von Pseudozufallszahlen spielen nicht nur in der Kryptographie eine wichtige Rolle, sondern vor allem auch bei Simulationen aller Art; die meisten Pseudozufallsgeneratoren wurden für solche Anwendungen entwickelt. Einer der bekanntesten verwendet lineare Kongruenzen, indem er ausgehend von einem Anfangswerts x_0 die Folgewerte gemäß $x_n = ax_{n-1} \bmod p$ berechnet, wobei p eine hinreichend große Primzahl ist. Bei geeigneter Wahl von a und p sind die so erzeugten Zahlen für Simulationen gut geeignet; für kryptographische Anwendungen jedoch sind sie wertlos: Wer im Rahmen einer Attacke mit bekanntem Klartext auch nur wenige Klartext/Chiffretext-Paare bestimmen kann, kennt eine Reihe von Folgengliedern x_n und kann daraus im allgemeinen ohne große Schwierigkeiten auf a und p schließen. Damit kann er den gesamten Schlüsselstrom berechnen. Ein kryptographisch sicherer Pseudozufallsgenerator muß natürlich so beschaffen sein, daß auch eine lange Folge der erzeugten Zahlen nicht ausreicht, um die Parameter des Algorithmus praktisch zu bestimmen. Wir werden solche Generatoren im Zusammenhang mit kryptographisch sicheren Hashfunktionen kennen lernen.

Chiffren, die auf diese Weise eine abgespeckte Version des *one time pad* realisieren, bezeichnet man als *Stromchiffren;* ihr Hauptanwendungsgebiet sind lange Datenströme, wie sie etwa bei der Übertragung zwischen einem Mobiltelephon und dem Sendemasten oder zwischen einem Beobachtungssatelliten und seiner Bodenstation anfallen; auch Pay-TV wird so verschlüsselt.

Bei der Kommunikation zwischen Computern oder zwischen Computern und ihrer Peripherie geht man meist einen anderen Weg: Die Angriffe, die wir im vorigen Kapitel betrachtet haben, beruhten auf der unterschiedlichen Häufigkeit der verwendeten Buchstaben, Buchstabenpaare und so weiter. Bei einem Alphabet aus nur 26 Buchstaben bereitet es auch bei kurzen Kryptogrammen keine Schwierigkeiten, sich eine entsprechende Statistik zu verschaffen; nimmt man aber ein Alphabet, dessen Buchstabenzahl die Anzahl der Buchstaben in einer typischen Nachricht deutlich übersteigt, wird es eher unwahrscheinlich, daß im Chiffretext überhaupt irgendwelche Doubletten zu finden sind, auf jeden Fall aber nicht genug, um eine sinnvolle Statistik aufzustellen. Bei diesem Ansatz spricht man von *Blockchiffren*, da vor der Verschlüsselung jeweils mehrere Buchstaben zu einem Block zusammengefaßt und dieser dann als ganzes verschlüsselt wird.

Da Blockchiffren hauptsächlich in Rechnernetzen eingesetzt werden, bestehen die Blöcke bei den heute üblichen Verfahren nicht aus Buchstaben, sondern aus Bits oder Bytes; bei den ersten kommerziell eingesetzten Blockchiffren arbeitete man mit Blöcken von 64 Bit; heute üblich sind mindestens 128 Bit.

§1: Anforderungen an eine Blockchiffre

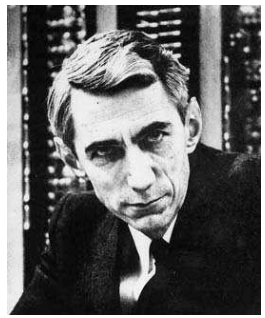
Idealerweise möchten wir erreichen, daß ein Gegner durch ein aufgefangenes Kryptogramm keinerlei Informationen über den Klartext erhält – außer natürlich der offensichtlichen, daß eine Nachricht gesendet wurde, und gewissen Hinweisen auf deren Länge. In der Kryptologie spricht man von *perfekter Sicherheit*, wenn die gegnerische Information über den Klartext ohne Kenntnis des Kryptogramms genauso groß ist wie mit.

Mathematisch läßt sich dieser Begriff wie folgt formalisieren: Der Gegner erwartet, daß irgendeine von r möglichen Nachrichten $m_1, \ldots, m_r$ übertragen wird, und auf Grund seiner Einschätzung der Situation ordnet er diesen Nachrichten Wahrscheinlichkeiten $p(m_i)$ zu: Ein Militärkryptanalytiker beispielsweise wird die Nachricht Angriff im Morgengrauen für wahrscheinlicher halten als Bringe Blumen fuer Mutti, und wer im Internet Kontodaten abgreifen will, wird Banken mit Sitz in der Nähe seines Opfers für wahrscheinlicher halten als solche aus anderen Erdteilen.

Sobald er einen Chiffretext c aufgefangen hat, kann er diesen untersuchen und dann die *bedingten* Wahrscheinlichkeiten $p(m_i|c)$ dafür berechnen, daß sich hinter dem Chiffretext c der Klartext m_i verbirgt. Wir reden von perfekter Sicherheit, wenn $p(m_i) = p(m_i|c)$ ist für alle i, wenn ihm also der aufgefangene Chiffretext nicht erlaubt, seine Anfangseinschätzung zu modifizieren. (Da man aus der Länge eines Kryptogramms fast immer Rückschlüsse auf die Länge der Nachricht ziehen kann, beschränkt man sich bei dieser Forderung meist auf Nachrichten einer festen Länge – sonst würde auch der *one time pad* keine perfekte Sicherheit bieten.)

Leider konnte SHANNON beweisen, daß eine *notwendige* Voraussetzung für perfekte Sicherheit darin besteht, daß die Schlüssellänge mindestens gleich der Summe der Längen aller je mit diesem Schlüssel übermittelten Nachrichten ist. Einen Beweis dieses Satzes in Lehrbuchdarstellung findet man zum Beispiel bei

DOMINIC WELSH: *Codes und Kryptographie*, Wiley-VCH, 1991, Theorem 7.3, oder in einigen der älteren Versionen dieses Skriptums.



CLAUDE ELWOOD SHANNON (1916–2001) wurde in Petoskey im US-Bundesstaat Michigan geboren; 1936 verließ er die University of Michigan mit sowohl einem Bachelor der Mathematik als auch einem Bachelor der Elektrotechnik, um am M.I.T. weiterzustudieren. In seiner 1938 geschriebenen Masterarbeit *A symbolic analysis of relay and switching circuits* entwickelte er die Schaltlogik, die seitdem eine wichtige Grundlage der digitalen Informationsverarbeitung bildet. Seine 1940 fertiggestellte Dissertation befaßte sich mit Anwendungen der Algebra auf die MENDELschen Gesetze. Nach seinem Studium arbeitete er bis 1956 bei den Bell Labs, wo er während des zweiten Weltkriegs insbesondere über die Sicherheit kryptographischer Systeme forschte. Seine *Mathematical theory of cryptography* wurde aus Geheimhaltungsgründen erst 1949 zur Veröffentlichung freigegeben. Seine wohl bekannteste Arbeit ist die 1948 erschienene *Mathematical theory of communication*, in der er die fehlerfreie Übertragung von Nachrichten über einen gestörten Kanal untersuchte. Von 1956 bis zu seiner Emeritierung 1978 lehrte er am M.I.T., das er dadurch zur führenden Universität auf dem Gebiet der Informationstheorie und Kommunikationstechnik machte. Zu seinen zahlreichen Arbeiten zählt auch eine über die mathematische Theorie der Jongliermuster, anhand derer Jongleure eine Reihe neuer Muster gefunden haben; außerdem konstruierte er mehrere Jonglierroboter.

Der zitierte Satz von SHANNON zeigt noch einmal von der theoretischen Seite, warum uns schon KERCKHOFFS riet, uns mit *praktischer* Sicherheit zu begnügen. Dafür müssen wir nicht unbedingt wissen, daß kein Gegner irgendwelche Informationen aus dem Kryptogramm gewinnen kann; wir können an mehreren Stellen abschwächen:

- Wir müssen uns nicht gegen *jeden* Gegner schützen, sondern nur gegen zu erwartende Gegner. Private Aufzeichnungen, die nicht in die Hände neugieriger jüngerer Geschwister fallen sollen, müssen nicht unbedingt auch gegenüber einer Attacke von Regierungsorganisationen sicher sein.
- Viele Geheimnisse haben ein Verfallsdatum, nach dem sie nicht mehr geschützt werden müssen. Entwürfe für eine öffentliche Ankündigung eines Unternehmens etwa müssen nur bis zu dieser Ankündigung geheim gehalten werden und müssen nicht wie etwa eine Kundenkartei gegen eine Attacke geschützt werden, die eine mehrmonatige Kryptanalyse erfordert.
- Wir müssen nicht unbedingt fordern, daß der Gegner *keinerlei* Infor-

mation aus dem Kryptogramm extrahieren kann; falls er nicht mehr als einige kleinere statistische Anomalien des Klartexts findet, kann man in vielen Fällen damit leben.

Diese Abschwächungen sind allesamt wenig konkret, denn natürlich wissen wir nicht, was welcher Gegner in welcher Zeit leisten kann. Um auf der sicheren Seite zu sein, müssen wir seine Fähigkeiten nach oben abschätzen, indem wir im Zweifelsfall mit einem sehr viel gefährlicheren Gegner rechnen, als dem tatsächlich erwarteten.

Der gefährlichste Gegner überhaupt ist der bereits erwähnte BAYESsche Gegner, der über unbegrenzte Ressourcen für Rechnungen und Probeentschlüsselungen verfügt. Er ist benannt nach dem englischen Theologen und Mathematiker THOMAS BAYES, dessen Formeln wohl den meisten Lesern aus der Wahrscheinlichkeitstheorie bekannt sind. Mit diesen Formeln arbeitet auch der BAYESsche Gegner, indem er für jeden möglichen Klartext m und/oder Schlüssel s die bedingte Wahrscheinlichkeit dafür berechnet, daß der abgefangene Chiffretext eine Verschlüsselung von m *bzw.* mit Schlüssel s ist. Er entscheidet sich dann für den Klartext m und/oder den Schlüssel s mit der größten bedingten Wahrscheinlichkeit.

THOMAS BAYES (1702–1761) wurde in London geboren als ältestes von sieben Kindern eines der ersten nonkonformistischen Pastoren Englands. Da die englischen Universitäten Oxford und Cambridge keine Nonkonformisten akzeptieren, mußte er zum Studium 1719 nach Schottland an die Universität Edinburgh, wo er sich für Logik und Theologie immatrikulierte. Nach seinen späteren Äußerungen muß er sich auch bereits damals oder kurz danach mit Mathematik beschäftigt haben. Wie sein Vater wurde er Geistlicher; seine mathematischen Arbeiten, z.B. über die Grundlagen der Analysis, erschienen zu seinen Lebzeiten nur anonym. Trotzdem wurde er 1742 fellow der Royal Society, die 1764 auch posthum seinen *Essay towards solving a problem in the doctrine of chances* veröffentlichte.

Sicherheit gegenüber dem BAYESschen Gegner ist nur in einem Punkt schwächer als perfekte Sicherheit: Wir können zulassen, daß er etwas

Information über den Klartext (oder den Schlüssel) erhält, allerdings so wenig, daß es ihm nichts nützt.

Mit SHANNONs Ansatz läßt sich genau bestimmen, wieviel Information der BAYESsche Gegner aus einem Chiffretext extrahieren kann; sobald die Schlüssellänge wesentlich kleiner wird als der Klartext, ist das leider deutlich mehr als alles, was wir tolerieren können.

Tatsächlich ist *keine* Blockchiffre sicher gegenüber einem BAYESschen Gegner, denn selbst wenn nur wenige Blöcke Text im ASCII Code verschlüsselt werden, gibt es mit an Sicherheit grenzender Wahrscheinlichkeit nur einen Schlüssel, mit dem die Entschlüsselung zu lesbarem Klartext führt. Der BAYESsche Gegner mit seinen unbegrenzten Ressourcen kann alle Schlüssel durchprobieren und so den einen richtigen identifizieren.

Man könnte ihm das Leben schwerer machen, indem man den Text vor der Verschlüsselung komprimiert; da kein heutiger Komprimierungsalgorithmus so gut ist, daß *jede* mögliche Bitfolge zu lesbarem Klartext dekomprimiert werden kann, heißt das aber nur, daß er eventuell einige Blöcke mehr braucht, um den korrekten Schlüssel zu finden.

Zum Glück sind die Gegner, vor denen wir uns schützen müssen, nur selten BAYESsche Gegner, und SHANNON machte sich auch Gedanken über die Konstruktion praktikabler Chiffren, die zumindest gegenüber den zu erwartenden Gegnern praktisch sicher sind. Nach seinen Vorstellungen beruht die Sicherheit eines Kryptosystems auf zwei Prinzipien, der *Konfusion* und der *Diffusion*.

Die *Konfusion* soll dafür sorgen, daß der Zusammenhang zwischen Schlüssel und Chiffretext möglichst undurchsichtig ist; mit seinen verfügbaren Mitteln soll der Kryptanalytiker aus dem Chiffretext nur wenig Information über den Klartext und den Schlüssel 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 hilfreich ist. Wie das Beispiel der HILL-Chiffre vom zweiten Übungsblatt zeigt, muß Konfusion insbesondere auch für Nichtlinearität

sorgen; andernfalls reicht möglicherweise bereits die Lineare Algebra zur Kryptanalyse.

Die *Diffusion* soll die statistischen Besonderheiten des Klartexts möglichst weiträumig über den Chiffretext verteilen und somit ausmitteln. Jedes Zeichen im Klartext sollte möglichst viele Zeichen im Chiffretext beeinflussen, so daß insbesondere kein signifikanter statistischer Zusammenhang zwischen dem Zeichen an einer speziellen Position des Klartexts mit dem Zeichen an einer (eventuell anderen) speziellen Position des Chiffretexts bestehen sollte.

§2: Der Aufbau einer Blockchiffre

Moderne Blockchiffren arbeiten natürlich nicht über einem Alphabet aus 26 Buchstaben; heute brauchen wir Kryptosysteme, die nicht nur Texte, sondern Dateien aller Art sicher verschlüsseln können, und bei vielen Anwendungen wollen wir, daß die gesamte Kryptographie ohne Zutun des Anwenders im Hintergrund abläuft. Insbesondere muß die entschlüsselte Datei genau so aussehen, wie vor der Verschlüsselung. Daher müssen wir davon ausgehen, daß unsere Blöcke aus beliebigen Bitfolgen (einer festen Länge) bestehen, und daß sie auch als solche wieder entschlüsselt werden müssen.

Wir betrachten die Chiffrierung daher als eine Abbildung von $\mathbb{F}_2^n$ nach $\mathbb{F}_2^n$, wobei $\mathbb{F}_2^n = \{0,1\}$ den Körper mit den beiden Elementen 0 und 1 bezeichnet und n die Blocklänge.

Die Rechenoperationen in $\mathbb{F}_2$ sind die Addition und die Multiplikation modulo zwei:

$\oplus$	0	1
0	0	1
1	1	0

$\odot$	0	1
0	0	0
1	0	1

$\mathbb{F}_2^n$ mit den üblichen Operationen ist ein Vektorraum über $\mathbb{F}_2$; die Vektoraddition bezeichnen wir mit $\oplus$. Sie läßt sich auf Computern problemlos realisieren durch die logische Antivalenz, das exklusive Oder XOR.

Da gängige Computer byteorientiert arbeiten, ist die Blocklänge n praktisch immer ein Vielfaches von acht; bei einigen alten Blockchiffren war $n = 64$, heute sollte $n \geq 128$ gewählt werden.

Getreu dem Prinzip der Konfusion darf die Verschlüsselung keine lineare Abbildung sein; eine solche ließe sich schließlich mit nur wenigen Paaren aus bekanntem Klartext und seiner Verschlüsselung rekonstruieren. Idealerweise sollte die Verschlüsselung irgendeine beliebige Permutation der *Menge* $\mathbb{F}_2^n$ sein.

Diese Menge hat 2^n Elemente; somit gibt es $2^n!$ verschiedene Permutationen. Schon für $n = 64$ ist das eine Zahl, die kein Taschenrechner mehr ausrechnen kann und vor der auch die meisten Computeralgebrasysteme nicht einmal eine Gleitkomma-Näherung berechnen können: Nach der STIRLINGschen Formel ist in erster Näherung $\log n! \approx n \log n$, wobei der Logarithmus links und rechts zur gleichen, aber beliebigen Basis genommen werden kann. Somit ist

$$\log_2(2^{64}!) \approx 2^{64} \log_2(2^{64}) = 2^{64} \cdot 64 = 2^{70} \quad \text{und}$$
$$\log_2(2^{128}!) \approx 2^{128} \log_2(2^{128}) = 2^{128} \cdot 128 = 2^{137}\,;$$

die Anzahl aller möglicher Permutationen hat also etwa $2^{70} \approx 1{,}2 \cdot 10^{21}$ *bzw.* $2^{135} \approx 4{,}3 \cdot 10^{40}$ Binärziffern; die Anzahl der Dezimalziffern ist ungefähr 0,3 mal so viel.

Um nur *eine* Permutation $\mathbb{F}_2^{64} \to \mathbb{F}_2^{64}$ zu spezifizieren, brauchen wir demnach ungefähr 2^{70} Bit oder 2^{67} Byte oder 2^{57} kB oder 2^{47} MB oder 2^{37} GB oder 2^{27} TB oder . . ., jedenfalls viel zu viel.

Damit ist klar, daß wir bei einer modernen Blockchiffre nicht mehr wie bei den monoalphabetischen Substitutionen einfach eine beliebige Permutation als Schlüssel zulassen können; die Übermittlung eines solchen Schlüssels wäre völlig unpraktikabel. Wir müssen uns daher beschränken auf eine deutlich kleinere Menge von Permutation, deren Elemente sich durch Schlüssel handhabbarer Länge beschreiben lassen.

Von einer guten Blockchiffre erwarten wir aber, daß die von ihr realisierten Permutationen wie zufallsverteilt in der Gruppe aller Permutationen liegen. Sie sollen beispielsweise weder eine Untergruppe

bilden noch eine verhältnismäßig kleine Untergruppe erzeugen, denn die Kenntnis von deren Erzeugenden und Relationen könnte einem Gegner Ansatzpunkte zu einer Entschlüsselung geben, die deutlich schneller geht als das Durchprobieren aller Schlüssel.

Eine Blockchiffre gilt nach heutigen Standards als gut, wenn auch nach längerer Untersuchung durch Fachleute keine Attacken auftauchen, die schneller sind als das Durchprobieren aller Schlüssel; als praktisch sicher gilt sie derzeit, wenn die Anzahl möglicher Schlüssel mindestens gleich 2^{100} oder besser 2^{128} ist. Bis zum Ende dieses Kapitels, spätestens aber bis Mitte des Semesters, sollte jedem Hörer klar sein, daß letzterer Wert in 25 Jahren mit ziemlicher Sicherheit nicht mehr als ausreichend betrachtet werden kann.

Fast alle heutigen Blockchiffren arbeiten in mehreren Runden, wobei in jeder Runde eine relativ einfache Permutation in Abhängigkeit von einem sogenannten Rundenschlüssel realisiert wird; erst die Hintereinanderausführung hinreichend vieler Runden führt zu ausreichender Konfusion und Diffusion. Speziell für die Diffusion setzt man hier auf den sogenannten *Lawineneffekt:* Während die Transformationen in jeder einzelnen Runde recht einfach sind und ein verändertes Eingabebit nur wenige Ausgangsbits beeinflußt, sorgt die Hintereinanderausführung vieler Runden dafür, daß sich dieser Einfluß immer mehr ausweitet und schließlich das gesamte Endergebnis beeinflußt – zumindest bei einer gut aufgebauten Blockchiffre.

Bis gegen Ende des vorigen Jahrhunderts dominierten bei der Architektur der Blockchiffren die sogenannten FEISTEL-Netzwerke, bei denen in jeder Runde nur ein halber Block modifiziert wurde.

HORST FEISTEL (1915–1990) wurde in Berlin geboren. Er emigrierte 1934 in die USA, wo er am MIT (BSc) und in Harvard (MSc) Physik studierte. Als Deutscher stand er während des zweiten Weltkriegs zunächst unter Hausarrest, wurde aber Anfang 1944 eingebürgert und arbeitete dann gleich in einem Forschungszentrum der Air Force. Nach dem Krieg arbeitete er kurze Zeit am MIT und bei MITRE, dann wechselte er zu IBM, wo er ab etwa 1960 die ersten Blockchiffren entwickelte, insbesondere auch das Lucifer-System, auf dessen Grundlage der im nächsten Paragraphen vorgestellte DES entwickelt wurde.

Da FEISTEL-Netzwerke jeweils mit halben Blöcken arbeiten, muß die Blocklänge gerade sein; wir bezeichnen sie als $2N$. Kern des Verschlüsselungsalgorithmus ist 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. Sie wird folgendermaßen angewandt: Vor Beginn der Verschlüsselung wird die Nachricht aufgeteilt in ein Paar (L_0, R_0) aus der linken Hälfte L_0 und der rechten Hälfte R_0; in den verschiedenen Runden wird dieses Paar transformiert zu neuen Paaren (L_i, R_i), deren letztes das Ergebnis der Verschlüsselung ist. Konkret wird in der i-ten Runde das Paar (L_i, R_i) zunächst ersetzt durch

$$\big(f(s_i, R_{i-1}) \oplus L_{i-1},\ R_{i-1}\big) \,,$$

wobei s_i den Schlüssel der i-ten Runde bezeichnet; danach werden (außer in der letzten Runde) die beiden Hälften miteinander vertauscht. Die i-te Runde realisiert also die Substitution

$$L_i = R_{i-1} \quad \text{und} \quad R_i = f(s_i, R_{i-1}) \oplus L_{i-1} \,.$$

Die FEISTEL-Funktion ist somit die einzige Quelle von *Konfusion;* die Auswahl der Schlüsselbits für die jeweilige Runde sowie auch die Vertauschung von linker und rechter Seite in jeder Runde dienen (neben der FEISTEL-Funktion selbst) der Diffusion.

Da $\mathbb{F}_2$ nur die beiden Elemente $0, 1$ enthält mit $0 \oplus 0 = 1 \oplus 1 = 0$, ergibt jeder Vektor $v \in \mathbb{F}_2^N$ zu sich selbst addiert den Nullvektor; somit läßt sich das Paar (L_{i-1}, R_{i-1}) aus (L_i, R_i) rekonstruieren durch

$$R_{i-1} = L_i \quad \text{und} \quad L_{i-1} = f(s_i, L_i) \oplus R_{i-1} \,.$$

Zur Entschlüsselung einer als FEISTEL-Netzwerk aufgebauten Blockchiffre kann die Verschlüsselung also einfach rundenweise rückgängig gemacht werden, wobei im wesentlichen derselbe Algorithmus wie zur Verschlüsselung benutzt wird.

§3: Der Data Encryption Standard DES

Logisch gehört dieser Paragraph eigentlich zum vorigen Kapitel: Der Data Encryption Standard DES ist kein Verfahren, das man heute noch

anwenden sollte. Der Standard wurde 1977 eingeführt und war ursprünglich für eine Dauer von zehn Jahren vorgesehen. Spätestens seit 1998, als die *Electronic Frontier Foundation* öffentlich vorführte, wie einfach er geknackt werden kann, sollte jedem klar sein, daß er seine nützliche Lebensdauer inzwischen deutlich überschritten hat.

Tatsächlich aber dürfte er immer noch zumindest eines der in der Praxis am häufigsten eingesetzten Kryptoverfahren sein, wenn auch (hoffentlich) meist in der derzeit wahrscheinlich noch sicheren Variante Triple-DES. Ein Grund dafür dürfte in einer heute als falsch eingeschätzten Rahmenbedingung seines Entwurfs liegen: Er sollte sich nur schwer rein softwaremäßig implementieren lassen und im Regelfall mit Spezialhardware eingesetzt werden. Dies hielt man für einen Sicherheitsvorteil, da dadurch ein Angriff von Amateuren mit begrenzten Mitteln deutlich erschwert wurde. Aus diesem Grund mußten viele Anwender in Spezialhardware investieren, die sich zumindest in manchen Unternehmen nur schwer aussondern läßt, bevor sie nicht mehrfach abgeschrieben ist.

Professionelle Angreifer freilich (zu denen auf jeden Fall auch der Hauptsponsor des DES, die *National Security Agency* NSA der Vereinigten Staaten zählt), sind typischerweise bereit, zum Knacken eines Codes ein Vielfaches des Aufwands einzusetzen, den sparsame Buchhalter für die Verschlüsselung zulassen, so daß sie durch die Notwendigkeit von Spezialhardware nicht abgeschreckt werden können.

Heute, da auch Amateure mit reinen Softwareattacken keine großen Schwierigkeiten mehr haben, DES zu knacken, muß man trotzdem sagen, daß DES nach allem, was in den letzten dreißig Jahren bekannt wurde, abgesehen von der viel zu kleinen Schlüssellänge ein sehr gutes Verfahren war: Trotz vieler Versuche auch der erfahrensten unter den veröffentlichenden Kryptanalytikern ist es keinem von ihnen gelungen, eine Angriffsmöglichkeit zu finden, die schneller wäre als das Durchprobieren aller Schlüssel: Alle erfolgreichen Angriffe basieren darauf. Im Sinne der Sicherheitsdiskussion im ersten Kapitel ist das die ideale Situation für ein Verfahren, dessen Sicherheit wir nicht beweisen können: Wir können mit ziemlich großer Sicherheit sagen, wie groß der Aufwand des Gegners sein muß: Er muß die Schlüssel durchprobieren, was im

Durchschnitt nach 2^{55} Versuchen zum Erfolg führt. Unser einziges Problem besteht darin, daß dieser Aufwand inzwischen mit recht geringen Kosten erbracht werden kann.

Der DES ist im wesentlichen als FEISTEL-Netzwerk aufgebaut, allerdings werden die Nachrichtenblöcke $(x_1, x_2, \ldots, x_{64})$ zunächst einer Anfangspermutation unterzogen, d.h. der Block 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 $(x_{58}, x_{50}, x_{42}, \ldots, x_{23}, x_{15}, x_7)$. Erst dann beginnen die im Falle des DES 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 anscheinend in erster Linie darin, Software-Attacken zu erschweren, denn Permutation sind aufwendig zu programmieren. (Bei Hardware-Implementierungen sind Permutationen natürlich sehr einfach und schnell durch Leitungskreuzungen 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; sie beschreiben also Abbildungen von $\mathbb{F}_2^6$ nach $\mathbb{F}_2^4$.

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 sechzehn 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 in der offensichtlichen Weise zusammengesetzt zu einer Funktion $F\colon \mathbb{F}_2^{48} \to \mathbb{F}_2^{32}$: 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

Die 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; der kryptologische Sinn besteht natürlich wieder in einer Diffusion, die dafür sorgt, 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.

§4: Designkriterien und Kryptanalyse des DES

Die Beschreibung des DES im vorigen Paragraphen läßt die meisten 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 kryptanalytische 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 Kryptanalyse*. 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 rele-

vanten 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 sechzehn 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.)*

Für die Permutation aus S_{32}, die in jeder FEISTEL-Funktion als Abschluß ausgeführt wird, sollten folgende Bedingungen erfüllt sein:

(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 – mehr dazu gleich bei der *linearen Kryptanalyse.*

Die restlichen Kriterien dienen in erster Linie zur Förderung der Diffusion: 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 kryptanalytischen Attacke beschäftigen, vor der dies schützen soll:

c) Differentielle Kryptanalyse

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) Differenzen ΔA bei der ersten S-Box folgende Verteilung:

ΔA	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Fälle	12	0	0	2	2	2	2	0	14	14	2	0	2	6	2	4

Wir haben also hier, wie auch bei den anderen Differenzen und anderen S-Boxen eine ziemlich inhomogene Verteilung. (Die Fallanzahlen für jede S-Box und jede Ausgabedifferenz sind aufgelistet im Anhang von

E. BIHAM, A. SHAMIR: Differential Cryptanalysis of the Data Encryption Standard, *Springer,* 1993;

in diesem Buch ist die differentielle Kryptanalyse des DES und anderer Blockchiffren vollständig beschrieben.)

Auf die Ausgabe der acht S-Boxen wird eine Permutation angewandt; die ist wieder mit Differenzenbildung kompatibel. Falls wir also die Differenzen der Ausgaben der S-Boxen kennen, bereitet diese Permutation keine Schwierigkeiten und wir kennen die Eingabedifferenzen für die zweite Runde, mit denen wir genauso weiter verfahren können.

Tatsächlich kennen wir die Ausgaben der acht S-Boxen der ersten Runde natürlich nicht; wir können nur für jede einzelne S-Box eine Wahrscheinlichkeitsverteilung der Ausgabedifferenzen angeben. Für einzelne Bits oder Bitgruppen kommen wir dabei durchaus auf recht ansehnliche Wahrscheinlichkeiten: Im obigen Beispiel für die Eingabedifferenz 100100 etwa ist mit jeweils einer Wahrscheinlichkeit von 14 : 64, also in mehr als 20% aller Fälle, die Ausgabedifferenz gleich acht oder

neun, binär geschrieben also 0100 oder 0101. Die Wahrscheinlichkeit dafür, daß die ersten drei Bits der Ausgabedifferenz gleich 010 sind, ist also 28 : 64 = 7 : 16, und wenn wir uns auf das erste und dritte Bit beschränken, kommen wir auf eine Wahrscheinlichkeit von 40 : 64 = 5 : 8 dafür, daß die Ausgabedifferenz die Form $0x0y$ hat. Das dritte Bit schließlich ist in 52 der 64 möglichen Fällen gleich Null, so daß wir zumindest dieses eine Bit mit der recht hohen Wahrscheinlichkeit von 13 : 16 kennen.

Eine Blockchiffre geht insbesondere deshalb durch mehrere Runden, daß sie solche Inhomogenitäten durch die Konfusion und Diffusion in den Folgerunden weitestgehend zu zerstören. Man wird erwarten, daß dies nicht für alle Klartextdifferenzen gleich gut gelingt; die Idee hinter der differentiellen Kryptanalyse ist, sich auf die zu konzentrieren, bei denen es möglichst schlecht gelingt. Wir müssen uns also genauer anschauen, wie eine Klartextdiffernz durch die Runden geht.

Definition: Eine *r-Rundencharakteristik* ist eine Folge

$$\Delta = (\delta_0, \delta_1, \ldots, \delta_r)$$

von Elementen aus $\mathbb{F}_2^{64}$.
Ein Klartextpaar $(x_0, y_0) \in \mathbb{F}_2^{64} \times \mathbb{F}_2^{64}$ *gehört zur Charakteristik* Δ, wenn für die Paare (x_i, y_i) der Ausgaben der i-ten Runde gilt: $x_i \oplus y_i = \delta_i$ für $i = 0, \ldots, r$.
Die *Wahrscheinlichkeit* einer Charakteristik ist die Wahrscheinlichkeit dafür, daß ein Klartextpaar (x_0, y_0) mit Differenz δ_0 zur Charakteristik Δ gehört.

Natürlich sind die Wahrscheinlichkeiten der meisten Charakteristiken sehr gering: In einem idealen Kryptosystem wären alle Ausgabewerte einer Runde gleich wahrscheinlich, die Wahrscheinlichkeit einer r-Rundencharakteristik sollte also im Mittel bei $1/2^{-64r}$ liegen, und schon die Wahrscheinlichkeit dafür, daß überhaupt ein Klartextpaar mit gegebener Differenz nach r-Runden eine ebenfalls vorgegebene andere Differenz hat, sollte im allgemeinen bei nur etwa 2^{-64} liegen.

Wenn wir eine gute r-Rundencharakteristik gefunden haben, deren Wahrscheinlichkeit deutlich besser ist als 2^{-64r}, können wir daher ziemlich sicher sein, daß ein zufällig gewähltes Klartextpaar (x_0, y_0) mit Anfangsdifferenz δ_0 und Enddifferenz δ_r in den r Runden so verschlüsselt wurde, wie es der Charakteristik entspricht. Ist etwa $p = 2^{-44}$, sollte die Wahrscheinlichkeit für alles andere bei nur etwa 2^{-20} oder etwa eins zu einer Million liegen. Konkret heißt das: Wir kennen die Differenzen $x_i \oplus y_i$ mit hoher Wahrscheinlichkeit und können die Verschlüsselung durch die r Runden verfolgen.

Zunächst brauchen wir aber gute Charakteristiken. Für nur eine Runde ist das sehr einfach: Für jeden Halbblock $d_0 \in \mathbb{F}_2^{32}$ führt der mit 32 Nullen auf 64 Bit aufgefüllte Block δ_0 auf die Charakteristik (δ_0, δ_0) mit Wahrscheinlichkeit eins: Sind nämlich (m_0, m_1) und (m_0', m_1) zwei Klartexte mit (nach der hier stets ignorierten Anfangspermutation) gleicher rechter Hälfte, so wird die linke Hälfte ersetzt durch

$$m_2 = m_0 \oplus f(s_1, m_1) \quad \textit{bzw.} \quad m_2' = m_0' \oplus f(s_1, m_1)\,,$$

und die Differenz ist

$$m_2 \oplus m_2' = \big(m_0 \oplus f(s_1, m_1)\big) \oplus \big(m_0' \oplus f(s_1, m_1)\big) = m_0 \oplus m_0' = d_0\,,$$

wie wir das schon oben gesehen haben. Nach Durchgang durch die erste Runde haben wir also die beiden Paare (m_2, m_1) und (m_2', m_1), deren Differenz wieder δ_0 ist.

Leider können wir diese Charakteristik nicht iterieren, denn nach der ersten Runde werden ja die beiden Hälften vertauscht, so daß wir dann (m_1, m_2') und (m_1, m_2') haben, worüber wir nicht so viel sagen können, da nun die FEISTEL-Funktionen verschiedene Werte liefern.

Um trotzdem zu einer Zweirundencharakteristik zu kommen, benutzen wir die Nichtinjektivität der FEISTEL-Funktion f: Wir suchen zwei Halbblöcke m_0 und m_0' derart, daß $f(s, m_0) = f(s, m_0')$ für möglichst viele Schlüssel s; die Differenz $m_0 \oplus m_0'$ bezeichnen wir mit d_0.

Zwei beliebige Klartexte der Form (m_0, m_1) und (m_0', m_1) mit Differenz $m_0 \oplus m_0' = d_0$ gehen in der ersten Runde nach (m_2, m_1) und (m_2', m_1) mit

$$m_2 = m_0 \oplus f(s_1, m_1) \quad \text{und} \quad m_2' = m_0' \oplus f(s_1, m_1)\,,$$

danach werden die linke und die rechte Hälfte vertauscht, so daß die Eingaben zur zweiten Runde gleich (m_1, m_2) und (m_1, m_2') sind, wobei

m_2 und m_2' die Differenz d_0 haben. In der zweiten Runde gehen die beiden Klartexte dann nach

$$\big(m_1 \oplus f(s_2, m_2), m_2\big) \quad \text{und} \quad \big(m_1 \oplus f(s_2, m_2'), m_2'\big)\,.$$

Da m_2 und m_2' Differenz d_0 haben, ist dabei mit einer gewissen Wahrscheinlichkeit

$$f(s_2, m_2) = f(s_2, m_2')\,.$$

Falls dem so sein sollte, haben wir $(0, d_0)$ als Differenz zwischen den Ausgabewerten. Es gibt daher eine Zweirundencharakteristik der Form

$$\big((d_0, 0), (d_0, 0), (0, d_0)\big)\,,$$

die mit einer gewissen Wahrscheinlichkeit p auftritt. Diese Charakteristik kann offensichtlich beliebig oft iteriert werden, denn bevor ihre Ausgabewerte in die nächste Runde gehen, werden die linke und die rechte Hälfte vertauscht, so daß wir wieder die Ausgangsdifferenz haben. Damit haben wir für jedes n eine n-Rundencharakteristik gefunden; ihre Wahrscheinlichkeit ist $p^{[n/2]}$, wobei die GAUSS-Klammer $[x]$ die größte ganze Zahl $\le x$ bezeichnet.

Diese Charakteristik ist umso nützlicher, je größer p ist. Wie sich zeigt, kann p nur dann größer als Null sein, wenn mindestens drei benachbarte S-Boxen aktiv sind; die größten Wahrscheinlichkeiten sind also zu erwarten bei *genau* drei aktiven S-Boxen. Systematisches Probieren zeigt, daß der beste erreichbare Wert dann

$$p = \frac{14}{64} \cdot \frac{8}{64} \cdot \frac{10}{64} = \frac{35}{8192} \approx 0{,}004272461 \approx \frac{1}{234} \approx 2^{-7{,}870716983}$$

ist. Er wird erreicht für

$$d_0 = (19\,60\,00\,00)_{\text{hex}} \quad \text{und} \quad d_0 = (1B\,60\,00\,00)_{\text{hex}}\,.$$

Damit haben wir also für beliebiges n eine n-Rundencharakteristik gefunden; leider ist sie aber nicht für jedes n brauchbar: Für 16 Runden ist ihre Wahrscheinlichkeit nur etwa

$$2^{-7{,}870716983 \times 8} \approx 2^{-62{,}96573586}\,,$$

wir bräuchten also mindestens 2^{63} Klartextpaare bekannter Differenz, um eines zu dieser Charakteristik zu finden, während wir mit nur 2^{56}

Versuchen alle Schlüssel durchprobieren könnten. Tatsächlich reichten sogar bereits 2^{55} Versuche, denn nimmt man das Einserkomplement von Schlüssel und Klartext, so entsteht das Einserkomplement des Chiffretexts. Deshalb sind auch die 14- und die 15-Rundencharakteristik, deren Wahrscheinlichkeiten bei

$$2^{-7{,}870716983 \times 7} \approx 2^{-55{,}09501888}$$

liegen, nicht sonderlich interessant; erst die 13-Rundencharakteristik liefert mit

$$p \approx 2^{-7{,}870716983 \times 6} \approx 2^{-47{,}22430190}$$

eine halbwegs interessante Wahrscheinlichkeit.

Wählen wir ein Halbwort m_0 derart, daß $m_0 \oplus f(s, m_1)$ für möglichst viele Schlüssel s gleich ist! Für einen solchen Schlüssel sind dann

$$m_2 = m_0 \oplus f(s, m_1) \quad \text{und} \quad m_2' = m_0' \oplus f(s, m_1)$$

Außerdem interessieren wir uns nicht für die Wahrscheinlichkeitsverteilung der Chiffretexte – Chiffretext ist schließlich das, was wir immer haben – sondern für Klartext oder besser noch den Schlüssel bei gegebenem Chiffretext.

Dazu nutzen wir aus, daß der Eingabewert der S-Boxen der ersten Runde nicht der Klartext ist, sondern der mit gewissen Schlüsselbits geXORte Klartext. Wenn wir nun für ein Klartextpaar mit gegebener Differenz die Ausgabedifferenz kennen, haben wir die möglichen Eingabepaare der S-Boxen, deren Differenzen ja genau dieselben sind wie für das Klartextpaar, von 64 auf eine erheblich kleinere Zahl reduziert. Für jedes dieses möglichen Paare können wir die entsprechenden Schlüsselbits durch XOR mit dem tatsächlichen Klartext berechnen und haben somit eine relativ kleine Anzahl potentieller Schlüsselteile. Wenn wir das ganze für hinreichend viele Klartextpaare wiederholen, sollte der für die jeweilige S-Box zuständige Schlüsselanteil relativ bald eindeutig feststehen.

DES mit nur einer Runde ist auf diese Weise also relativ einfach zu entschlüsseln, falls wir genügend viele Paare von Klartext mit fester

Differenz haben. Diese können wir uns nur verschaffen durch eine Attacke mit wählbarem Klartext, also der schwierigsten Form der Attacke.

Auch diese Attacke liefert aber nicht die Ausgabedifferenzen der ersten Runde, sondern nur die der *letzten.* Der Ansatz der differentiellen Kryptanalyse des DES ist daher folgender:

1. Man wähle eine geeignete Differenz zwischen Klartexten.
2. Dazu erzeuge man hinreichend viele Paare von Klartextblöcken mit dieser Differenz, verschlüssele sie und behalte nur die so berechneten Chiffretextpaare.
3. Durch Analyse der Klartextdifferenzen und des Verhaltens der S-Boxen in den verschiedenen Runden bestimme man die zu erwartende Wahrscheinlichkeitsverteilung der Eingabedifferenzen der letzten Runde.

Differentielle Kryptanalyse war der erste Ansatz, DES mit geringerem Aufwand als der vollständigen Durchsuchung des Schlüsselraums zu zu brechen. Da aber, wie bereits erwähnt, die Designer des DES die differentielle Kryptanalyse schon kannten lange bevor sie in der offenen Literatur auftauchte und den Algorithmus so gut wie möglich dagegen immun machten, ist diese Attacke nicht sehr praktikabel: Selbst bei Angriffen mit frei wählbarem Klartext braucht man über 2^{40} Paare aus Klartext und Chiffretext, um den Schlüssel zu finden.

d) Lineare Kryptanalyse

Eine leichte Verbesserung bietet die kurz später entdeckte *lineare Kryptanalyse:* Zwar sind die Ausgabebits der S-Boxen nach Design-Kriterium ($S2$) auch nicht näherungsweise lineare Funktionen der Eingabebits, aber es kann dennoch vorkommen, daß man eine Linearkombination von Ausgabebits mit einer Wahrscheinlichkeit, die deutlich über 50% liegt durch eine Linearkombination der Eingabebits vorhersagen kann. Mit hinreichend vielen Paaren aus Klartext und Chiffretext läßt sich dadurch ein niedrigdimensionaler affiner Unterraum des Schlüsselraums $\mathbb{F}_2^{56}$ finden, in dem der Schlüssel mit hoher Wahrscheinlichkeit liegen muß. Die vollständige Durchsuchung dieses Unterraums ist unproblematisch, so daß der Schlüssel mit hoher Wahrscheinlichkeit gefunden werden kann.

Auch für diese Attacke sind allerdings unrealistisch viele Paare aus Klartext und Chiffretext erforderlich (in einer Variante genügen noch mehr reine Chiffretexte), so daß der Gesamtaufwand nicht wirklich geringer sein dürfte als die vollständige Durchsuchung des Schlüsselraums.

Nach allem, was in der offenen Literatur bekannt ist, gibt es also zum Knacken des DES keine wesentlich bessere Alternative zur vollständigen Durchsuchung des Schlüsselraums.

e) DES-Cracker

Der erste in der offenen Literatur dokumentierte realistische Angriff auf DES war denn auch die vollständige Durchsuchung des Schlüsselraums. Eine amerikanische Bürgerrechtsorganisation, die *Electronic Frontier Foundation (EFF),* konstruierte eine Maschine mit Spezialhardware zum Knacken von DES mit Chiffretext allein.

Die *Electronic Frontier Foundation* wurde 1990 nach dem großen *Hacker Crackdown* in den USA gegründet; Initiatoren waren unter anderem JOHN PERRY BARLOW, bekannt vor allem durch die Lieder, die er für *The Grateful Dead* schrieb, JOHN GILMORE, einer der Pioniere sowohl von *Sun Microsystems* als auch der *Free Software Foundation*, MITCHELL KAPOR, der Gründer von *Lotus*, sowie STEVE WOZNIAK, einer der beiden Gründer von *Apple.*

Ihr Ansatz ist im wesentlichen der unseres guten alten Feinds, des BAYESschen Gegners: Kodiert man einen englisch- oder deutschsprachigen Klartext im ASCII-Code sollten dem BAYESschen Gegner ein bis zwei Blöcke Chiffretext ausreichen, um den Schlüssel zu finden.

Natürlich verfügen selbst die vier obengenannten Gründer der *Electronic Frontier Foundation* nicht über die unbegrenzten Mittel, die der BAYESsche Gegner einsetzen kann; verglichen mit vielen anderen Gegnern verfügt aber doch jeder von ihnen über beträchtliche Mittel. Trotzdem war das 1997 begonnene und 1998 beendete DES-Cracker-Projekt kein Angriff ohne Rücksicht auf die Kosten: Die *Electronic Frontier Foundation* wollte gerade zeigen, daß DES auch mit begrenzten Mitteln geknackt werden kann. Aus diesem Grund wurde der Ansatz des BAYESschen Gegners an mehreren Stellen optimiert:

Zunächst ist es nicht notwendig, wirklich für *jeden* Schlüssel die bedingte Wahrscheinlichkeit auf Grund des Chiffretexts zu bestimmen: In vielen Fällen werden bei der Entschlüsselung nicht druckbare Zeichen entstehen, so daß schon nach wenigen Byte klar ist, daß die Wahrscheinlichkeit des Schlüssels Null ist.

DES Cracker beginnt daher mit der Aussonderung unmöglicher Schlüssel durch massiv parallele Hardware: Die Maschine arbeitet mit zwei Blöcken Chiffretext; sie hat als Kern von EFF entwickelte ASICs *application specific integrated circuits)*, die einen 64-Bit-Block mit einem vorgegebenen Schlüssel dechiffrieren können und die Bytes des Ergebnisses auf vom Benutzer einstellbare Bitmuster überprüfen – beispielsweise darauf, ob es sich um ASCII-Codes druckbarer Zeichen handelt. Nur wenn alle Bytes den gewählten Kriterien genügen, wird auch der zweite Block entsprechend untersucht, und wenn auch hier kein im Klartext unmögliches Byte auftaucht, wird der Schlüssel zur weiteren Untersuchung an einen die Maschine steuernden PC weitergegeben, der eine genauere Untersuchung gemäß dem Ansatz des BAYESschen Gegners durchführt.

Von den 256 möglichen ASCII-Werten sind etwa ein Viertel druckbare Zeichen; da DES eine Ausgabe liefert, die sich nur wenig von einer Zufallsfolge unterscheidet, wird ein Block nur mit einer Wahrscheinlichkeit von etwa $1 : 4^8 = 1 : 65536$ den Test bestehen; für zwei Blöcke liegt die Wahrscheinlichkeit entsprechend bei $1 : 4^{16} = 1 : 2^{32}$. Von den 2^{56} zu untersuchenden Blöcken werden also nur etwa

$$2^{56-32} = 2^{24} = 16\,777\,216$$

an den PC weitergegeben, und die Untersuchung von etwa 17 Millionen Klartextkandidaten ist kein Problem für einen Standard-PC.

Der hauptsächliche Rechenaufwand liegt in der Voruntersuchung der Blöcke durch die ASICs; je nachdem, wie viele von diesen parallel arbeiten, kann dies mehr oder weniger schnell gehen.

Die tatsächlich gebaute Maschine enthält $1536 = 3 \times 2^9$ ASICs, von denen jedes aus 24 parallel arbeitenden Sucheinheiten besteht; insgesamt können also jeweils 36 864 Schlüssel parallel untersucht werden.

Jede Sucheinheit kann zweieinhalb Millionen Schlüssel pro Sekunde untersuchen, die gesamte Maschine also etwas über 92 Milliarden.

Insgesamt müssen 2^{56} Schlüssel untersucht werden; im Mittel wird man nach 2^{55} Versuchen den richtigen gefunden haben. Dafür braucht man

$$\frac{2^{55}}{92\,160\,000\,000} \approx 390\,937 \text{ Sekunden} \approx 108{,}5 \text{ Stunden} \approx 4{,}5 \text{ Tage}\,.$$

Die Maschine ist skalierbar: Jeweils 64 Chips sitzen auf einem Board und 12 Boards in einer Chassis (einer ehemaligen SUN); die gebaute Maschine besteht aus dem steuernden PC zusammen mit zwei Chassis; der PC könnte aber auch mit deutlich mehr als zwei Chassis arbeiten und auch mehrere PCs wären denkbar.

In der gebauten Version kostete DES Cracker 210 000 $, wovon 80 000 $ Entwicklungskosten waren; der Bau eines zweiten Exemplars wäre also für 130 000 $ möglich, wobei der Preis bei Serienproduktion wohl deutlich niedriger gewesen wäre. Mit einer Investition in der Größenordnung von einer Million Dollar hätte am also bereits damals einen DES-Schlüssel innerhalb weniger Stunden finden können. Heute kann man das auch rein softwaremäßig mit einem handelsüblichen PC.

Da eine Million Dollar auch für Geheimdienste kleiner Länder und (etwas kreative Buchhaltung vorausgesetzt) Großunternehmen kein Problem sind, war damit endgültig gezeigt, daß die Zeit für DES abgelaufen war.

Die EFF veröffentlichte sowohl die komplette Hardware-Spezifikation als auch die Software von DES-Cracker; in gedruckter Form findet man sie im Buch

ELECTRONIC FRONTIER FOUNDATION: *Cracking DES. Secrects of Encryption Research, Wiretap Politics & Chip Design*, O'Reilly, 1998

Online ist das Buch unter anderem verfügbar unter

`http://cryptome.org/cracking-des.htm` ;

die DES Cracker Seite der EFF ist

`www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/` .

§4: Modifikationen

Viele Unternehmen, insbesondere im Bankenbereich, hatten viel Geld in DES-Hardware investiert und hatten daher wenig Interesse, auf ein neues Verfahren umzusteigen – ganz abgesehen davon, daß 1998 noch kein allgemein anerkannter Nachfolgealgorithmus zur Verfügung stand. (Mit dem inzwischen normierten „offiziellen“ Nachfolger AES werden wir uns in einen späteren Kapitel beschäftigen.)

Deshalb bot sich ein Verfahren an, das auf der Grundlage von DES eine Verschlüsselung mit deutlich mehr als nur 56 Schlüsselbit realisierte.

a) Mehrfacher DES

Die Kritik an der extrem kurzen Schlüssellänge des DES wurde bereits kurz nach dessen Einführung laut, und schon damals wurde vorgeschlagen, ihn zur Erhöhung der Sicherheit mehrfach und mit verschiedenen Schlüsseln anzuwenden.

Eine zweifache Verschlüsselung hängt von 112 statt von nur 56 Schlüsselbit ab; der Aufwand für eine Durchsuchung des gesamten Schlüsselraums steigt also von 2^{56} auf 2^{112}, was eine Maschine nach Art des DES Crackers auch nach heutigem Stand der Technik noch nicht in akzeptabler Zeit durchführen kann.

Eine mehrfache Verschlüsselung bietet allerdings nur dann Vorteile, wenn die Hintereinanderausführung zweier DES-Verschlüsselungen nicht äquivalent zur einfachen DES-Verschlüsselung mit einem anderen Schlüssel ist.

Um dies zu untersuchen, müssen wir noch einmal zurück zur grundsätzlichen Struktur von Blockchiffren: Blockchiffren sind Permutationen auf der Menge aller Blöcke; im Falle von DES also auf der Menge aller bijektiver Abbildungen von der Menge aller 64-Bit-Blöcke auf sich selbst.

Die 2^{56} durch DES definierten Permutationen bilden natürlich nur eine winzige Teilmenge der Gruppe aller $2^{64}!$ solcher Permutationen; falls diese Teilmenge eine Gruppe sein sollte (oder in einer relativ kleinen

Untergruppe liegt), kann die mehrfache Anwendung von DES keine (oder nur wenig) zusätzliche Sicherheit bieten.

Wir brauchen daher Informationen darüber, wie groß die kleinste Gruppe ist, die alle 2^{56} DES-Permutationen enthält. Über deren genaue Struktur und Elementanzahl ist leider nichts bekannt, man kann aber immerhin untere Schranken angeben: Für jeden einzelnen DES-Schlüssel kann man die zugehörige Substitution so lange wiederholen, bis die Identität entsteht; da $2^{64}!$ eine zwar große, aber endliche Zahl ist, muß dies nach endlich vielen Schritten der Fall sein.

Angenommen, bei solchen Berechnungen mit verschiedenen Schlüsseln ergeben sich die Ordnungen $n_1, n_2, \ldots, n_r$. Dann enthält die kleinste Gruppe, in der alle DES-Substitutionen liegen, zyklische Untergruppen der Ordnungen $n_1, \ldots, n_r$. Nach einem einfachen Satz der Gruppentheorie, dem Satz von LAGRANGE, müssen die Zahlen $n_1, \ldots, n_r$ dann die Ordnung der gesamten Gruppe teilen; diese ist also mindestens gleich dem kleinsten gemeinsamen Vielfachen der n_i.

Experimente zweier Wissenschaftler von Bell Northern Research in Ottawa zeigten, daß die Ordnung der erzeugten Untergruppe größer als $0{,}9 \times 10^{2499}$ sein muß; das liegt sehr deutlich über 2^{56}. Für Einzelheiten sei auf ihre Arbeit

KEITH W. CAMPBELL, MICHAEL J. WIENER: DES is not a Group, *Crypto '92, Springer Lecture Notes in Computer Science* **740** (1993), 512–520

verwiesen. Sie zeigt insbesondere, daß mehrfache Anwendung von DES die Sicherheit erhöhen kann.

b) Doppelter DES

Beim doppelten DES hat man einen Schlüsselraum mit 2^{112} Elementen. Leider muß ein Gegner mit hinreichend viel Speicherplatz diesen aber nicht vollständig durchsuchen, um die Schlüssel zu finden: Falls er nur ein Paar (x, y) aus Blöcken von einander entsprechendem Klartext und Chiffretext hat, reicht es, kann er in einer sogenannten *meet in the middle attack* den Klartext x mit allen 2^{56} möglichen Schlüsseln *ver*schlüsseln und den Chiffretext y mit allen 2^{56} Schlüsseln zu *ent*schlüsseln. Ist

y = DES$\big(s_2, \text{DES}(s_1, x)\}\big)$, so ist DES$^{-1}(s_2, y)$ = DES(s_1, x), dieser Block kommt also in beiden Listen vor. Da die von DES realisierten Permutationen weit von einer Gruppe entfernt sind, ist es sehr unwahrscheinlich, daß es noch einen anderen Block gibt, der in beiden Listen auftaucht; sobald man einen solchen Block gefunden hat, kann man also praktisch sicher sein, daß man s_1 und s_2 kennt.

Der Speicherbedarf dieser Attacke ist sicherlich nichts für Amateure am heimischen PC: Schließlich müssen für 2^{56} Schlüssel jeweils zwei Blöcke à 64 Bit oder 8 Byte gespeichert werden; insgesamt also 2^{60} Byte = 2^{50} Kilobyte = 2^{40} Megabyte = 2^{30} Gigabyte = 2^{20} Terabyte; das ist ungefähr eine Million mal soviel, wie eine große handelsübliche Festplatte heute faßt. Trotzdem handelt es sich hier um eine Kapazität, die nicht nur bei Regierungsorganisationen, sondern auch bei großen Unternehmen durchaus im Bereich des Möglichen liegt; für Google etwa sind das *peanuts.* In der Praxis spielt der doppelte DES daher keine Rolle.

c) Dreifacher DES

Der nächste Schritt zu Erhöhung der Komplexität besteht in einer dreifachen Anwendung von DES. Auch hier kann man wieder eine *meet in the middle attack* anwenden, aber auf dem Weg zur Mitte muß von mindestens einer der beiden Seiten aus DES zweimal mit verschiedenen Schlüsseln angewendet werden, der Aufwand liegt also in der Größenordnung von 2^{112}. Dreifacher DES oder, wie man meist sagt, Triple DES (kurz TDES), gilt daher im Augenblick noch als sicher: Nachdem die DES-Cracker-Attacke publik geworden war, zog die amerikanische Regierung die Zulassung von DES für weniger geheime Nachrichten im Regierungsbereich zurück und verlangte stattdessen Triple DES; entsprechend wurden in Deutschland die Euroscheckkarten ersetzt durch neue Karten, die auf Triple DES anstelle des einfachen DES beruhen. Daran hat sich bis heute wenig geändert: Die großen Kreditkartenunternehmen einigten sich auf einen gemeinsamen Standard, der nach den Initiatoren Europay, Mastercard und Visa als EMV bezeichnet wird, damit die Geldautomaten und Verkaufsstellenterminals einer Firma auch die Karten aller anderer beteiligter Unternehmen lesen können,

und der verwendet weiterhin Triple DES als symmetrisches Kryptoverfahren. (Erst seit 2010 *bzw.* 2011 ist fakultativ auch AES zugelassen.) Die genau Spezifikation des Standards ist via www.emvco.com zu finden.

Triple DES wird praktisch immer so angewendet, daß man mit dem ersten Schlüssel *ver*schlüsselt, mit dem zweiten *ent*schlüsselt und mit dem dritten wieder *ver*schlüsselt. Oft ist dabei der dritte Schlüssel gleich dem ersten; zumindest für einen *meet in the middle* Angriff erleichtert dies die Arbeit des Angreifers nicht wesentlich. Trotzdem gehen inzwischen die Empfehlungen eher dahin, drei verschiedene Schlüssel zu verwenden.

d) DESX

Wie wir gesehen haben, wurde DES bewußt so spezifiziert, daß reine Softwareimplementierungen eher langsam sind. Bei Triple DES macht sich diese Langsamkeit gleich dreifach bemerkbar. Da es sich heute kaum mehr lohnt, neu in DES Hardware zu investieren, suchte man daher früh nach Alternativen, die einerseits das Problem der kurzen DES Schlüssellänge abmildern, andererseits aber keine mehrfache Anwendung von DES erfordern. RON RIVEST, den wir im nächsten Kapitel kennenlernen werden, schlug 1995 eine erstaunlich einfache solche Methode vor: Sein DESX kombiniert den gewöhnlichen DES einfach mit zwei VIGENÈRE-Verschlüsselungen: Zusätzlich zum DES-Schlüssel $s_1 \in \mathbb{F}_2^{56}$ gibt es noch zwei VIGENÈRE-Schlüssel $s_2, s_3 \in \mathbb{F}_2^{64}$, und ein Nachrichtenblock $x \in \mathbb{F}_2^{64}$ wird verschlüsselt als $s_3 \oplus DES(s_1, x \oplus s_2)$.

Nach unseren Erfahrungen mit der VIGENÈRE-Chiffre wäre es naiv zu glauben, daß ein Gegner zum Knacken dieser Chiffre den gesamten Schlüsselraum $\mathbb{F}_2^{56} \times \mathbb{F}_2^{64} \times \mathbb{F}_2^{64} \cong \mathbb{F}_2^{184}$ durchsuchen muß; er könnte beispielsweise durch eine differentielle Kryptanalyse zunächst den Schlüssel s_3 eliminieren, und wenn er – wie auch immer – s_1 und s_2 gefunden hat, genügt ein einziges Klartext/Chiffretext-Paar, um auch noch s_3 zu finden. Trotzdem ist DESX erstaunlich gut: Die beste bekannte Attacke muß auch bei 2^m bekannten Klartext/Chiffretext-Paaren immer noch 2^{119-m} mögliche Schlüssel durchprobieren; sie ist zu finden bei

JOE KILIAN, PHILLIP ROGAWAY: How to protect DES against exhaustive key search (an analysis of DESX), *Journal of Cryptology* **14** (2001), 17–35;
http://www.springerlink.com/content/b1ljx8fky92nlhmk/

e) Alternativen zu DES

Da DES nach heutigen Standard nicht mehr zeitgemäß ist, sind die gerade betrachteten Modifikationen höchstens als Übergangslösungen interessant; sinnvoller sind völlig neu konzipierte Alternativen. Insbesondere eine davon, denn offiziellen Nachfolger AES (Advanced Encryption Standard) werden wir in dieser Vorlesung noch ausführlich diskutieren; da hierzu allerdings mehr Algebra erforderlich ist, als wir bislang kennen, muß das entsprechende Kapitel nach weiter hinten verschoben werden, bis wir im Zusammenhang mit den sogenannten asymmetrischen Kryptoverfahren mehr Hilfsmittel aus der Algebra kennengelernt haben. Zum Abschluß dieses Kapitels wollen wir uns noch kurz mit der Frage befassen, wie man eine Blockchiffre nach Art von DES in der Praxis anwenden sollte.

§5: Operationsmodi

Bislang haben wir DES nur betrachtet für die Verschlüsselung eines einzelnen Blocks; tatsächlich besteht eine Nachricht aber meist aus einer ganzen Folge $x_1, x_2, \ldots, x_n$ von Blöcken. In diesem Paragraphen wollen wir uns überlegen, wie diese Nachricht am besten verschlüsselt wird. Dabei werden zum ersten Mal auf ein Phänomen stoßen, daß uns im Laufe dieser Vorlesung noch mehrfach begegnen wird: Auch eine relativ sichere Chiffre zeigt praktisch immer deutliche Schwächen, wenn sie einfach in der offensichtlichen Weise angewendet wird.

Die Betrachtungen hier beziehen sich nicht speziell auf DES, sondern gelten genauso auch für jede andere Blockchiffre. Daher gehen wir hier aus von *irgendeiner* Blockchiffre

$$B\colon S \times X \to X; \quad (s,x) \mapsto B(s,x)\,,$$

die einem Block x in Abhängigkeit von einem Schlüssel s den Chiffretext $B(s,x)$ zuordnet.

a) Electronic Code Book (ECB)

Die naheliegendste Weise, eine Nachricht $x_1, x_2, \ldots, x_n$ zu verschlüsseln, besteht darin, ihr den Chiffretext $y_1, y_2, \ldots, y_N$ zuzuordnen mit $y_i = B(s, x_i)$, d.h jeder Block wird vor der Übertragung mit s verschlüsselt.

So einfach diese Methode auch ist, in der Praxis sollte man sie besser nicht anwenden. Die große Schwäche von ECB liegt in der Tatsache begründet, daß gleiche Klartextblöcke *immer* zu gleichen Chiffretextblöcken führen. Das bedeutet zwar nicht unbedingt, daß gleiche Klartextteile stets gleich verschlüsselt werden, denn wegen der Blockstruktur der Chiffre hängt der Chiffretext ja auch noch davon ab, wie weit der Textbeginn vom Blockanfang entfernt ist. Bei DES mit ASCII gibt es dafür aber nur acht Möglichkeiten, bei DES mit Unicode sogar nur vier, so daß bei längeren Texten in denen gewisse Namen und/oder Begriffe häufig vorkommen, durchaus die Gefahr einer größeren Anzahl identischer Chiffretextblöcke besteht.

Auch *magic bytes*, die bei vieler Dateiformaten als Dateianfang vorgeschrieben sind, führen stets zur selben Verschlüsselung; bei anderen Dateiformaten wie etwa ausführbaren Programmen oder gewissen Büroprogrammen gibt es innerhalb der Datei viele Blöcke von Nullen, *usw.*, so daß jemand, der alle Nachrichten eines Absenders abhört die Empfänger leicht in Klassen einteilen kann, die (ungefähr) dieselbe Information erhalten. Dadurch kennt er zwar noch nicht den Inhalt der Nachrichten, kann aber vielleicht doch sehr nützliche Informationen gewinnen.

Manchmal kann ein Gegner auch einfach dadurch Schaden anrichten, daß er unbemerkt die Reihenfolge von Nachrichtenblöcken vertauscht oder aber einen Nachrichtenblock *mehrfach* übermittelt. Er könnte auch eine neue Nachricht generieren, die aus Teilen bereits übermittelter Nachrichten zusammengesetzt ist; falls er die Struktur der Nachrichten auf Grund gemeinsamer Blöcke erkennt, hat er sogar eine gute Chance, daß die entstehende Nachricht sinnvoll ist. Bei Nachrichten mit festem Format, wie sie beispielsweise im elektronischen Zahlungsverkehr unter Banken üblich sind, hätte er eventuell sogar die Möglichkeit, zwei von ihm selbst initiierte Transaktionen zu identifizieren und zu seinem

Vorteil zu manipulieren. Aber auch das bloße Einschleusen einer nicht als falsch zu erkennenden Nachricht etwa zu Sabotagezwecken kann bereits genügend Schaden anrichten.

Natürlich hat ein Angreifer bei einer guten Blockchiffre auch im ECB-Modus keine Chance, die Nachricht zu dechiffrieren oder gar den Schlüssel herauszufinden, aber wie wir gesehen haben, kann er sich bei gewissen Typen von Nachrichten doch einiges an Information verschaffen.

Man kann in der Kryptographie üblicherweise nicht davon ausgehen, daß ein Anwender über die Stärken und Schwächen des verwendeten Kryptosystems Bescheid weiß: Er verläßt sich darauf, daß das gekaufte oder von einem Experten eingerichtete System seine Geheimnisse zuverlässig schützt, egal worum es sich handelt. Daher sollte man den ECB-Modus im Normalfall nicht benutzen.

b) Cipher Block Chaining (CBC)

Hier wird die Nachricht $x_1, x_2, \ldots, x_n$ übermittelt als $y_1, y_2, \ldots, y_n$ mit

$$y_i = B(s, x_i \oplus y_{i-1}) .$$

Da es für $i = 1$ noch keinen Chiffretextblock y_0 gibt, muß dabei zusaätzlich zum Schlüssel noch ein Anfangsblock y_0 explizit festgelegt werden. Er muß nicht unbedingt geheimgehalten werden, sollte aber zwecks zusätzlicher Sicherheit möglichst für jede Verschlüsselung neu gewählt werden.

Unabhängig von der Wahl des Anfangsblock hängt bei CBC jeder übertragene Block y_i auch noch vom Vorgänger y_{i-1} ab; es ist daher nicht möglich, eine Nachricht durch Auslassen von Blöcken oder durch Zusammensetzen zweier existierender Nachrichten zu manipulieren ohne daß Blöcke verfälscht und damit unentschlüsselbar werden – was den Empfänger (hoffentlich) zum Nachfragen veranlaßt.

Ein weiterer Vorteil besteht darin, daß jeder übertragene Block y_i von *jedem* der Blöcke $x_1, x_2, \ldots, x_i$ abhängt; insbesondere hängt als der letzte Block y_N von jedem einzelnen Klartextblock ab. Falls also die

übermittelte Nachricht noch elektronisch unterschrieben werden soll, reicht es, den Block y_N zu unterschreiben.

(Mit elektronischen Unterschriften werden wir uns im Zusammenhang mit der asymmetrischen Kryptographie beschäftigen. Wir werden dann auch Verfahren kennenlernen, wie man auch bei anderen Übertragungsmodi oder gar der Übertragung von Klartext einen Block finden kann, der von der gesamten Nachricht abhängt. Einen solchen Block bezeichnet man als *message authentication code*, kurz MAC. Im Allgemeinen berechnet man ihn durch sogenannte sichere Hash-Verfahren; eine gute Blockchiffre im CBC-Modus liefert ein, wenn auch vergleichsweise aufwendiges, solches Verfahren.)

Die Abhängigkeit eines jeden Chiffreblocks von allen vorausgehenden Klartextblöcken hat nicht nur Vorteile: Sie führt auch dazu, daß Übertragungsfehler nicht nur einen Block betreffen. Tatsächlich führen sie aber bei CBC nur zur falschen Entschlüsselung zweier Blöcke: Die Entschlüsselungsfunktion ist bei CBC offenbar

$$x_i = B^{-1}(s, y_i) \oplus y_{i-1} ,$$

bereits $x_{i+2} = B^{-1}(s, y_{i+2}) \oplus y_{i+1}$ ist also von einen falsch übermittelten Block y_i nicht mehr betroffen.

Ein großes Problem beim ECB-Modus war, daß gleiche Nachrichten und auch gleiche Blöcke gleich übermittelt werden. Beim CBC-Modus ist dieses Problem zumindest insofern abgemildert, als gleiche Block durch das XOR mit dem vorangegangenen Chiffreblock verschieden chiffriert werden. Falls man allerdings den Anfangsblock y_0 konstant wählt – aus Sicht des Anwenders sicherlich die einfachste Lösung – werden identische Nachrichten weiterhin identisch chiffriert.

Die Sicherheit wird also auf jeden Fall erhöht, wenn für jede Übertragung ein neuer Anfangsblock benutzt wird. Dieser könnte beispielsweise ein Zufallsblock sein, der – damit ihn auch der Empfänger kennt – entweder unverschlüsselt oder ECB-verschlüsselt als erstes übertragen wird. Damit wird die zu übermittelnde Nachricht um einen Block verlängert, was im allgemeinen kein großes Problem ist – außer vielleicht in dem Fall, daß man sehr viele sehr kurze Nachrichten über eine teure oder stark kapazitätsbeschränkte Leitung übertragen muß.

Ein zufälliger Anfangsblock hilft noch nicht gegen das Problem, daß ein Angreifer einfach eine aufgefangene Nachricht ein zweites Mal in die Leitung einspielt. Da so etwas beispielsweise bei elektronischen Finanztransfers unbedingt erkannt werden muß, enthalten entsprechende Nachrichten selbstverständlich eine eindeutige Buchungsnummer.

Auch in anderen Systemen ist es oft üblich, daß jede Nachricht ihre eindeutige Kennzeichnung hat, und das legt es nahe, zumindest in solchen Systemen entweder direkt diese Nachrichtennummer oder aber eine daraus abgeleitete Zahl (eine sogenannte *Nonce;* die Bezeichnung ist eine Kontraktion von *N*umber used *once*) zu verwenden. Da auch Nachrichtennummern Informationen enthalten, sollte diese Nummer zur Sicherheit mit der Blockchiffre verschlüsselt werden.

Ganz perfekt ist die Chiffre auch so noch nicht: Angenommen, der Chiffretextblock y_i ist gleich dem Block y_j. Dann können wir wie folgt argumentieren:

$$y_i = B(s, x_i \oplus y_{i-1}) \wedge y_j = B(s, x_j \oplus y_{j-1})$$
$$\Longrightarrow x_i \oplus y_{i-1} = x_j \oplus y_{j-1} \Longrightarrow x_i \oplus x_j = y_i \oplus y_j \,.$$

Somit läßt sich die Differenz $x_i \oplus x_j$ aus der Differenz der vorangegangenen Chiffretextblöcke $y_{i-1} \oplus y_{j-1}$ berechnen. Falls die Nachrichtenquelle eine ähnlich hohe Redundanz hat wie die deutsche Sprache, sollte diese Information ausreichen, um die beiden Blöcke (bis auf Reihenfolge) zu rekonstruieren.

Dies ist sicherlich ein Schwachpunkt, den man in der besten aller Welten gerne vermeiden würde; andererseits ist die Wahrscheinlichkeit, daß zwei gleiche Chiffretextblöcke auftreten, nicht sonderlich groß: Wenn wir davon ausgehen, daß sich Chiffretext im CBC-Modus wie eine Zufallsfolge verhält (was wahrscheinlich etwas zu optimistisch ist), liegt sie im Falle der Blocklänge N etwa bei $2^{-N/2}$. Bei einer 64-Bit-Blockchiffre wie DES heißt das, daß wir etwa $2^{32} = 4294967296$ oder rund 4,3 Milliarden Blöcke brauchen, bevor wir mit einer Wahrscheinlichkeit von mindestens 50% zwei gleiche Chiffretextblöcke finden. Bei einer Blockchiffre mit 128 Bit (was heute eigentlich Mindeststandard sein sollte, kommt man sogar auf $2^{64} \approx 1{,}8 \cdot 10^{19}$ oder rund 18 Trilliarden

Blöcke. (Für Einzelheiten sei auf das Kapitel über sichere Hashverfahren verwiesen, wo wir das sogenannte *Geburtstagsparadoxon* genauer betrachten werden.)

Da wir in Wirklichkeit natürlich keine Zufallsfolge haben, dürften die tatsächlichen Wahrscheinlichkeiten wohl etwas größer sein, aber bei normalen Textdateien sollten sie weiterhin praktisch vernachlässigbar sein, und bei wirklich großen Dateien wie etwa Videofilmen sollte die Kenntnis einiger weniger einzelner Blöcke für einen Angreifer wohl nutzlos sein. Was bleibt, ist das Restrisiko, daß beispielsweise genau der eine Block, in dem ein besonders streng geheimzuhaltender Name oder Begriff steht zufälligerweise trotzdem genauso verschlüsselt wird wie ein anderer Block und damit einem ohne große Erfolgsaussichten auf genau dieses Restrisiko hoffenden Gegner bekannt wird – dies gehört zum unvermeidbaren Risiko eines jeden nicht absolut sicheren Kryptosystems.

Als Randbemerkung sollte erwähnt werden, daß obige Rechnung natürlich auch zeigt, daß verschiedene Klartextblöcke zu verschiedenen Chiffretextblöcken führen, falls die Chiffretextblöcke in den Vorgängerpositionen verschieden sind. Dies mag zwar auf den ersten Blick als nicht sehr informativ erscheinen, aber die Enigma wurde im zweiten Weltkrieg geknackt eben wegen der Beobachtung, daß sie nie einen Buchstaben durch sich selbst verschlüsselt. Bei einer Blockchiffre von 64 oder 128 Bit kann man mit so einer Information zwar sehr viel weniger anfangen, aber es handelt sich doch Information für den Gegner, von der wir nicht sicher sein können, was er damit anfangen kann.

c) Cipher Feedback (CFB)

Die nun folgenden Modi sind nützlich, wenn Daten in Echtzeit übertragen werden sollen, die kürzer sind als die Blocklänge; hier verwenden wir die Blockchiffre, um einen Schlüsselstrom zu erzeugen, der nach Art des *one time pad* verwendet wird. Der große Unterschied ist natürlich, daß die Entropie dieses Schlüsselstroms nur die des Schlüssels und des (ähnlich zu CBC verwendeten) Anfangsblocks ist: Unser guter alter Feind, der BAYESsche Gegner, hätte also keinerlei Schwierigkeiten, die Chiffre zu entschlüsseln. Unsere Hoffnung und der publik gewordene

Teil der Erfahrung im Umgang mit Blockchiffren wie DES und AES beruht darauf, daß der Schlüsselstrom zu komplex ist für einen realen Gegner.

Bei CFB gehen wir davon aus, daß die Daten nicht als Blöcke anfallen, sondern in eventuell kleineren Einheiten zu k Bit. Typisch für Anwendungen ist der Wert $k = 8$, d.h. wir verschlüsseln einen Strom von Bytes, aber selbst der Fall $k = 1$, bei dem einzelne Bits verschlüsselt werden, kommt gelegentlich vor. Falls k kleiner ist als die Blocklänge N des Codes, ist dieser Modus also um den Faktor N/k langsamer als die bislang betrachteten Modi.

Auch hier gehen wir aus von einem Anfangsblock; er ist allerdings durch k fast vollständig festgelegt: In einem Register R, dessen Länge gleich der Blocklänge des verwendeten Codes ist, stehen rechts k Bit, zum Beispiel lauter Einsen, die restlichen Bits des Registers werden auf Null gesetzt. Sodann werden die *ersten* k Bit von $B(s, R)$ zu den ersten k Bit der Nachricht addiert und dies wird übertragen. Man beachte, daß das Verschlüsselungsergebnis nur für die Übertragung benutzt wird; der Inhalt des Registers behält seinen Wert.

In jedem der folgenden Schritt wird der Inhalt des Registers um k Bit (nichtzyklisch) nach links verschoben, und die k zuletzt übertragenen Bits werden am rechten Ende eingesetzt. Sodann werden die ersten k Bit des mit dem neuen Registerinhalt berechneten Blocks $B(s, R)$ zum nächsten Nachrichtenblock addiert und übertragen, *usw.*

Sofern die ersten k Bit, die ins Register geschrieben werden, konstant sind, werden gleiche Texte stets gleich verschlüsselt, und – was schlimmer ist – zum ersten Block der Nachricht wird stets derselbe Schlüssel addiert, so daß die statistischen Angriffe aus dem ersten Kapitel anwendbar sind. Falls k gleich der Blocklänge ist, könnte ein damit erfolgreicher Angreifer sogar den Wert $B(s, R)$ für den Anfangszustand des Registers rekonstruieren und, zumindest im Falle DES mit Hilfe von DES-Cracker oder einem ähnlichen Werkzeug den Schlüssel s ermitteln. Hier liefert also die Wahl eines deutlich unterhalb der Blocklänge liegenden Werts von k einen zusätzlichen Sicherheitsfaktor. Bei einer guten und zeitgemäßen Blockchiffre ist es natürlich unmöglich, aus einem Paar von

Klar- und Chiffretextblöcken den Schlüssel zu rekonstruieren – es sei denn, man verfügt über die Rechenkraft des BAYESschen Gegners.

Auch in der praktischen Anwendung gibt es ein Problem, denn wie bei CBC hängt wieder jedes übertragene Wort aus k Bit von allen Vorgängern ab. Aus Sicht des Empfängers allerdings hängt der Inhalt des Registers nur ab von den letzten r empfangenen Chiffretextblöcken, wobei r die kleinste ganze Zahl ist mit $rk \geq n$, denn nach r Übertragungen fällt jeder Chiffreblock wegen der zyklischen Verschiebung aus dem Register heraus. Ein Übertragungsfehler beeinflußt hier also insgesamt $r + 1$ Nachrichtenblöcke.

d) Output feedback (OFB)

Typische Anwendungen von Stromchiffren sind Satellitenübertragungen. Hier sind Bitfehler auf Grund atmosphärischer Störungen relativ häufig; obwohl sie natürlich durch fehlerkorrigierende Codes so weit wie möglich kompensiert werden, muß man doch immer wieder mit auch längerfristigen erhöhten Fehlerraten rechnen. Dabei ist die Eigenschaft des CFB-Modus, jeden Fehler gleich auf $r + 1$ Blöcke durchschlagen zu lassen, höchst unwillkommen.

Ein für solche Anwendungen nützlicher Modus ist *output feedback* (OFB). Auch dieser Modus erzeugt einen Schlüsselstrom mit Hilfe eines Registers R, allerdings hängt dessen Inhalt weder vom Klartext noch vom Chiffretext ab. Da der Schlüsselstrom zum Nachrichtenstrom addiert wird, betrifft daher ein Bitfehler bei der Übertragung hier nur ein einziges Bit.

Das Register R wird zu Beginn auf einen Anfangswert gesetzt. Im Gegensatz zu CFB wird das Register selbst in jedem Schritt verschlüsselt, sein Inhalt also durch $B(s, R)$ ersetzt. Danach werden die ersten k Bit zum Nachrichtenblock addiert, und vor dem nächsten Schritt wird das Register *zyklisch* um k Positionen nach links verschoben.

Bei dieser Vorgehensweise *muß* man natürlich für jede Übertragung einen neuen Anfangsblock und/oder Schlüssel wählen, denn ansonsten wird mehrfach derselbe Schlüsselstrom verwendet, ein Gegner kann also schon mit einer relativ kleinen Anzahl von Chiffretexten durch

Häufigkeitsanalysen Informationen über den Klartext bekommen, die bis zur völligen Entschlüsselung führen können. Da die Blockchiffre hier nur zur Erzeugung eines Schlüsselstroms verwendet wird, ist die zu betrachtende Einheit aus Sicht des Kryptanalytikers kein Block, sondern die kleinste Dateneinheit der Nachricht, typischerweise also ein Byte, so daß die Verfahren aus dem ersten Kapitel problemlos angewandt werden können. Außerdem muß darauf geachtet werden, daß der Schlüsselstrom natürlich periodisch ist. Die Periode ist zwar, abgesehen von einigen wenigen sogenannten *schwachen* Schlüsseln der Blockchiffre, sehr groß, aber je nach zu übertragendem Datenvolumen kann es trotzdem Probleme geben.

e) Counter mode (CTR)

Dieser Modus wird im Standard für DES nicht erwähnt, wurde aber 2001 vom NIST (dem *National Institute of Standards* der Vereinigten Staaten) als eine Methode zur Anwendung von Blockchiffren standardisiert.

Ausgangspunkt ist eine *Nonce*, d.h. eine Zahl a, die für genau eine Nachricht und danach nie wieder während der Gültigkeitsdauer des Schlüssels verwendet wird. Sie wird beispielsweise aus der Nummer oder dem Übertragungsdatum der Nachricht nach einem vorher definierten Verfahren erzeugt.

An diese Zahl wird wie bei OFB ein von der Nachricht unabhängiger Schlüsselstrom erzeugt, hier nach der Vorschrift

$$s_i = B(s, a\|i)\,,$$

wobei $a\|i$ für eine Vorschrift steht, wie die Blocknummer i hinter die Zahl a geschrieben wird. Konkret geht es also darum, daß a eine gewisse maximale Bitlänge hat, und die restlichen Bits werden für i reserviert.

Wie bei OFB *muß* auch hier natürlich sichergestellt sein, daß keine zwei Blöcke mit demselben s_i verschlüsselt werden, d.h. die Anzahl möglicher Werte für i muß größer sein als die maximale Länge einer zu übertragenen Nachricht. Bei 64 Bit-Chiffren kann dies den Wertebereich von i deutlich einschränken; bei einer Blocklänge von 128 Bit sollte es jedoch mit realistischen Nachrichten keine Probleme geben.

Die Verschlüsselung geschieht auch hier wie beim *one time pad*, d.h.

$$y_i = x_i \oplus s_i = x_i \oplus B(s, a\|i)\,.$$

Auch hier muß wieder unbedingt sichergestellt werden, daß derselbe Schlüsselstrom nur einmal benutzt wird, d.h. die Zahl a darf auf keinen Fall mehrfach benutzt werden, da sonst die Attacken aus dem ersten Kapitel greifen würden.

Sofern dies wirklich sichergestellt ist, dürfte CTR wohl der sicherste unter den hier diskutierten Modi sein.

§6: Literatur

Da DES rund 25 Jahre lang *das* Standardverfahren für ernsthafte symmetrische Verschlüsselung im zivilen Bereich war, ist er natürlich in praktisch jedem Lehrbuch der Kryptologie aus der damaligen Zeit ausführlich beschrieben, z.B. in

JAN C.A. VAN DER LUBBE: *Basic Methods of cryptography*, Cambridge University Press, 1998

oder, besonders ausführtlich in

ALAN G. KONHEIM: *Cryptography – A Primer*, Wiley, 1981

Auch in

JOHANNES BUCHMANN: *Einführung in die Kryptographie*, Springer, ^{5}2010

ist noch ein Kapitel über DES zu finden.

Eine ausführliche Diskussion der Operationsmodi findet man unter anderem in

A.J. MENEZES, P.C. VAN OORSCHOT, S.A. VANSTONE: *Handbook of applied cryptography*, CRC Press 1997

sowie in

NIELS FERGUSON, BRUCE SCHNEIER: *Practical Cryptography*, Wiley, 2003

und

NIELS FERGUSON, BRUCE SCHNEIER, TADAYOSHI KOHNO: *Cryptography Engineering – Design Principles and Practical Applications*, Wiley,2010

Speziellere Fragen sind außer in den bereits im Text zitierten Arbeiten und Büchern auch in fast jeder Konferenz über Kryptologie behandelt; insbesondere gilt dies für Tagungen wie *Crypto, Eurocrypt* und *Asiacrypt*, deren Proceeding jeweils in den Springer Lecture Notes in Computer Science erscheinen.