Proseminar im WS98/99

# Kryptographische Verfahren und ihre Anwendung

3. Teil: Symmetrische Verfahren II

Richard Atterer

26. November 1998

## Inhaltsverzeichnis



## 1 Einleitung

**Symmetrische Verschlüsselungsverfahren** werden bei Programmen, die Daten vor unautorisiertem Zugriff schützen sollen, sehr häufig verwendet. Ein Hauptmerkmal symmetrischer Verfahren ist, daß die zur Ver- und Entschlüsselung benötigten Schlüssel identisch sind oder voneinander abgeleitet werden können.

Soll eine mithilfe eines solchen Verfahrens verschlüsselte Botschaft über einen unsicheren Nachrichtenweg übertragen werden, müssen sich Sender und Empfänger zunächst auf einen gemeinsamen Schlüssel verständigen und ihn geheimhalten, weswegen auch der Begriff „*secret key*-Verfahren“ verwendet wird.


Abbildung 1: Symmetrische Blockchiffre

Man unterscheidet bei den symmetrischen Algorithmen zwischen Stromchiffren (*stream ciphers*), die Daten Bit für Bit verschlüsseln, und Blockchiffren (*block ciphers*), bei deren Benutzung die Daten in Blöcke von normalerweise 64 oder 128 Bits aufgeteilt werden und jeder Block mit demselben Schlüssel verschlüsselt wird.

Durch eine Reihe von **Betriebsarten** können symmetrische Verfahren für verschiedenste Zwecke eingesetzt werden. Je nach Anwendungsgebiet variieren damit die Eigenschaften des Chiffrierverfahrens.

Es kann eine Betriebsart gewählt werden, die bei etwas geringerer Sicherheit wahlfreien Zugriff auf alle Chiffretextblöcke z.B. in einer Datenbank erlaubt, oder eine, die stattdessen besseren Schutz vor Manipulationen bei der Datenübertragung bietet.

Auf die Wahl der Betriebsart hat auch Einfluß, daß sich die verschiedenen Betriebsarten im Fall von Übertragungsfehlern sehr unterschiedlich verhalten.

In dieser Abhandlung soll die Funktionsweise von symmetrischen Blockchiffren anhand der Algorithmen DES und IDEA verdeutlicht werden. Anschließend werden die Möglichkeiten bei der Wahl einer Betriebsart sowie deren Eigenschaften diskutiert, wobei auch auf die Verwendung von Blockchiffren als Stromchiffren eingegangen wird.

# 2 Symmetrische Verschlüsselungssysteme

## 2.1 Grundkonzepte von Blockchiffren

In Abbildung 1 ist die Arbeitsweise einer Blockchiffre grob dargestellt. $M$ bezeichnet einen Klartextblock (*message*) aus dem gewählten Alphabet (bei der Umsetzung auf Rechnern stets Binärwörter), $C$ einen Chiffretextblock (*ciphertext*), $E$ und $D$ die Ver- bzw. Entschlüsselungsoperation (*en-/decryption*). Für ihre Durchführung wird ein Schlüssel $K$ (*key*) benötigt. Die Blockgröße von $n$ Bit für Klartext und Chiffretext ist üblicherweise gleich.

Es gilt $C = E(K,M)$, sowie, da der Schlüssel für Ver- und Entschlüsselung gleich ist, $M = D(K,C)$. Damit diese Anforderungen bei identischer Größe von Klartext- und Chiffretextblock erfüllt sind, muß die Blockchiffre für jeden Schlüssel eine bijektive Abbildung der $2^n$ Klartextblöcke auf die $2^n$ Chiffretextblöcke darstellen, also eine eindeutige Substitution in beide Richtungen.

Man kann sich $E$ als Zugriff auf eine in Abhängigkeit von $K$ generierte Tabelle vorstellen, deren $2^n$ Einträge für jeden Klartextblock den zugehörigen Chiffretextblock enthalten. In der Praxis läßt sich die Funktion aufgrund des immensen Speicherplatzbedarfs auf diese Weise nicht implementieren.

Für alle möglichen Permutationen ergibt sich eine Anzahl von $(2^n)!$ verschiedener bijektiver Abbildungen. Für den Fall, daß die Schlüssellänge mit der Blockgröße übereinstimmt, werden durch den Schlüssel $2^n$ der $(2^n)!$ Möglichkeiten ausgewählt, also ein Bruchteil $1/(2^n-1)!$.

Ein „guter“ Algorithmus erfüllt die Aufgabe, bei möglichst niedrigem Rechenaufwand einen so komplexen Zusammenhang zwischen Ein- und Ausgabe herzustellen, daß Angriffe auf die Chiffre unmöglich werden.

Zu den an die Ausgabe gestellten Anforderungen zählt die *Vollständigkeit*, d.h. jedes der Outputbits sollte von allen Inputbits abhängen, so daß jede Änderung der Eingabe unter Umständen jedes der einzelnen Outputbits ändern könnte. Darüber hinaus soll die Wahrscheinlichkeit, daß sich das Outputbit tatsächlich ändert, für jedes Outputbit im Idealfall 0,5 betragen – im Durchschnitt ändert sich also beim Ändern eines Inputbits die Hälfte der Outputbits. Dies wird als *Avalanche*-Effekt bezeichnet.

Über die Outputbits sollte nichts bekannt sein, solange nicht die Inputbits bekannt sind. Jeder Ausgabewert sollte die gleiche Häufigkeit haben, um einen Angriff mit statistischen Methoden zu verhindern. Bei bijektiven Substitutionen ist das stets der Fall, weil jeder Ausgabewert nur für genau einen Eingabewert auftreten kann.

## 2.2 Data Encryption Standard (DES)

Beim DES handelt es sich um eine Blockchiffre, die für lange Zeit mangels anderer offiziell als sicher eingestufter Verfahren eine dominierende Stellung einnahm. Wie kein anderer Algorithmus hatte DES Einfluß auf die Entwicklung der Kryptographie als Wissenschaft.

### 2.2.1 Geschichte des DES

**1972** Das National Bureau of Standards (NBS), heute National Institute of Standards (NIST) startet ein Programm zur sicheren Datenspeicherung/-übermittlung. Gesucht wird ein einheitlicher Algorithmus (Kompatibilität verschiedener Geräte, billigere Implementierung, garantiert sicher und allgemein verfügbar).

**1973** Ausschreibung im Federal Register. Entwicklungsziele: hohe Sicherheit; leichte Implementation, v.a. in Hardware; Sicherheit soll allein im Schlüssel liegen, nicht in der Geheimhaltung des Algorithmus; Algorithmus für alle zugänglich; flexibel für verschiedene Anwendungen; effizient; leichte Validierung; muß exportierbar sein. Die eingereichten Vorschläge sind unzureichend.

**1974** Zweite Ausschreibung – Einreichung eines Vorschlags von IBM, die nachfolgend auf bereits beantragte Patentrechte an dem Algorithmus verzichteten.

**1975** *Veröffentlichung des von der National Security Agency (NSA) modifizierten Algorithmus.* Nie zuvor wurde ein von der NSA untersuchter Algorithmus veröffentlicht – vermutlich gab das NBS aufgrund eines Mißverständnisses zuviele Details bekannt.

**1976** Trotz Kritik, v.a. wegen der geringen Schlüssellänge von 56 Bits und der nicht veröffentlichten Designgrundsätze der NSA, Anerkennung von DES als US-Bundesstandard. DES floß später auch in einige ISO-Standards ein, wurde aber als Algorithmus an sich in keinem anderen Land Standard.

**1983** Alle fünf Jahre Neuzertifizierung des Standards – DES wird weiterhin als sicher eingestuft.

**1987** zunehmende Anzeichen für Mängel von DES vorhanden, dennoch erneute Zertifizierung für weitere fünf Jahre.

**1993** Weiterhin keine Alternative vorhanden, deshalb nochmalige Zertifizierung, obwohl Unsicherheit des Algorithmus offensichtlich ist. Die Kosten einer Maschine, die DES in durchschnittlich 3,5 Stunden knackt, werden auf 1 Mio. Dollar veranschlagt.

**1994** Mittels linearer Kryptoanalyse wird ein DES-Schlüssel von 12 HP9735-Workstations in 50 Tagen ermittelt (durchschnittlich $2^{43}$ benötigte Klartexte).

**1998, 15. Juli** Ein weniger als 250 000 US-Dollar teurer Spezialcomputer knackt eine mit DES verschlüsselte Nachricht in 56 Stunden (brute-force).

### 2.2.2 Beschreibung des Algorithmus

Der *Data Encryption Standard* ist ein symmetrischer Algorithmus, der auf Blöcken zu 64 Bits arbeitet. Die Schlüssellänge beträgt ebenfalls 64 Bits, wobei allerdings jedes achte Bit als Paritätsbit dient, so daß die effektive Mächtigkeit des Schlüsselraums nur $2^{56}$ ist. Die Sicherheit des Verfahrens beruht auf der Kombination von Permutationen und Substitutionen.

Die Ver- bzw. Entschlüsselung eines 64-Bit-Blocks läßt sich grob in drei Schritte aufteilen:

- ◇ Der Eingabeblock wird der Eingangspermutation IP (*initial permutation*) unterworfen, wodurch die Reihenfolge der einzelnen Bits verändert wird. Das Ergebnis wird in zwei 32-Bit-Register L und R geschrieben. Ähnlich wird mit den 64 Bits des Schlüssels verfahren: Die Funktion PC-1 (von *permuted choice*) wählt die 56 vom Algorithmus benutzten Schlüsselbits aus und permutiert sie, bevor sie in zwei 28-Bit-Register C und D geschrieben werden.

- ◇ In 16 Runden werden 16mal dieselben Rechenschritte durchgeführt, wobei in jeder Runde auf den Werten von L und R Operationen ausgeführt werden, die durch einen von 16 Teilschlüsseln beeinflußt werden. Das Ergebnis jeder Runde wird wieder in L und R zurückgeschrieben, in der nächsten Runde wird es dann erneut verarbeitet.

- ◇ Nach der 16. Runde werden L und R wieder zu einem 64-Bit-Wert zusammengefügt, bevor die Ausgangspermutation $\mathrm{IP}^{-1}$ angewendet wird, die zu IP invers ist, d.h. $\mathrm{IP}^{-1}(\mathrm{IP}(M)) = M$. Da IP und $\mathrm{IP}^{-1}$ nur am Anfang und Ende durchgeführt werden, haben sie auf die kryptographische Sicherheit von DES keinen Einfluß.

*Hinweis:* Beim DES werden die Bits aus historischen Gründen $1 \ldots 64$ numeriert, wobei die Wertigkeit von Bit 64 eins ist.

Der Ablauf eines der 16 Verschlüsselungsschritte ist aus Abbildung 2 ersichtlich. Zunächst wird der 32-Bit-Wert R durch die Expansionsabbildung E auf einen 48-Bit-Wert abgebildet. Abbildung 3 zeigt, daß hierbei die Reihenfolge der meisten Bits unverändert bleibt, es werden lediglich einige Bits verdoppelt. Das Ergebnis wird mit einem 48-Bit-Teilschlüssel XOR-verknüpft.

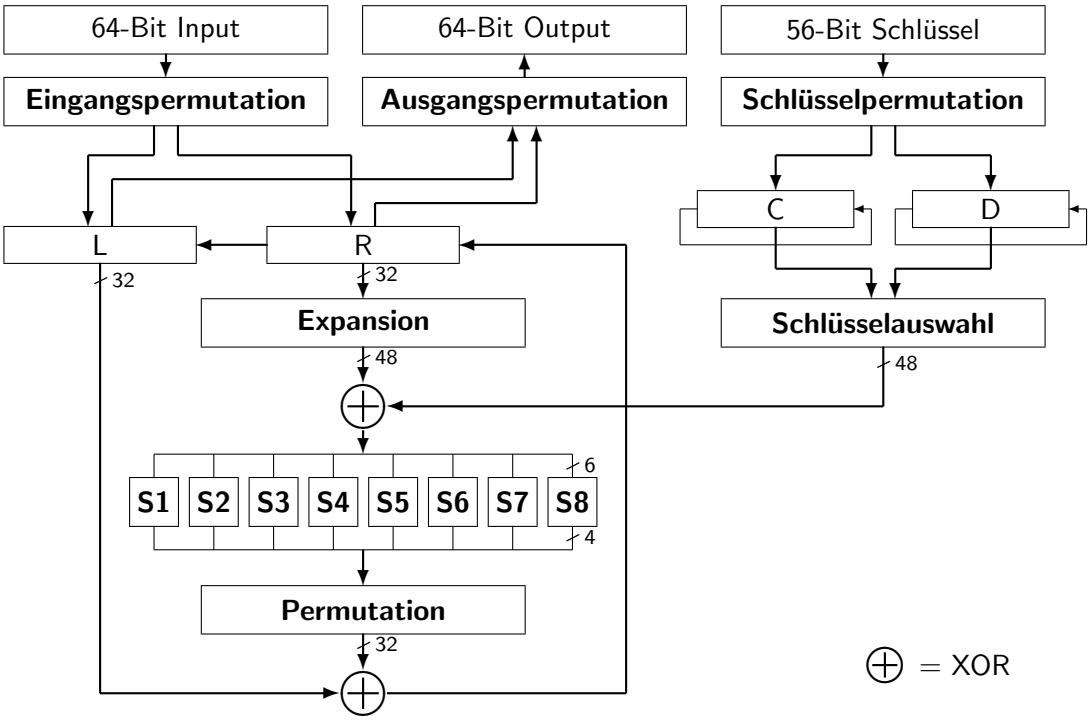


Abbildung 2: Der *Data Encryption Standard*

Jetzt werden die 48 Bits in acht Teilblöcke zu je 6 Bits zerlegt und für jeden der Blöcke wird mithilfe einer der S-Boxen S1 bis S8 eine Substitution durchgeführt. Eine S-Box ist beim Data Encryption Standard nichts weiter als eine Tabelle, die durch den Eingabewert indiziert wird. Die Tabelleneinträge sind konstant und durch den Standard definiert.

Da der Input jeder S-Box 6 Bits breit ist, der Output jedoch nur 4 Bits, beträgt die Breite des gesamten Blocks nach der Substitution wieder 32 Bit. Man kann die Funktion der S-Boxen deswegen auch dahingehend interpretieren, daß durch die redundanten Bits, die durch die Expansionsabbildung E erzeugt wurden, für jede S-Box eine von vier möglichen bijektiven 4-Bit-Substitutionen angesteuert wird. Die S-Boxen gewährleisten aufgrund ihrer Nicht-Linearität mehr als alles andere die Sicherheit des Verschlüsselungsverfahrens.

In Abbildung 4 ist eine S-Box dargestellt. Die Eingabe von S1 sind beispielsweise Bits 1 bis 6 des Ergebnisses von E nach der XOR-Verknüpfung mit Bits 1 bis 6 eines Teilschlüssels. Für den Fall, daß Bit 1 gelöscht und Bit 6 gesetzt ist, wird die zweite der vier möglichen Substitutionen durchgeführt (siehe Abbildung 5). Abbildung 6 zeigt alle vier Substitutionen von S1.

Durch Aneinanderfügen der Ausgaben der S-Boxen wird ein 32-Bit-Wort gebildet, das anschließend der Permutation P unterzogen wird. Diese Permutation ist in Abbildung 7 angegeben; z.B. wird Bit 16 der Eingabe zu Bit 1 der Ausgabe. Nachdem der so erhaltene Wert mit dem Inhalt des Registers L XOR-verknüpft wurde, wird er zurück ins Register R geschrieben. Der alte Block R ersetzt den bisherigen Wert von L.

Die Vorgehensweise bei der Auswahl der Teilschlüssel wurde bisher noch nicht vollständig erklärt. Wie bereits erwähnt werden die Bits des externen Schlüssels zunächst durch die Funktion PC-1 permutiert und dann in die zwei 28-Bit-Register C und D geschrieben. Diese Register werden

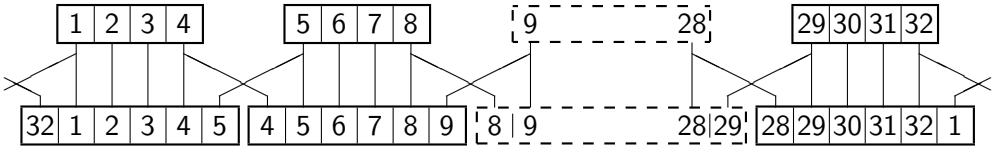


Abbildung 3: Die Expansionsabbildung E

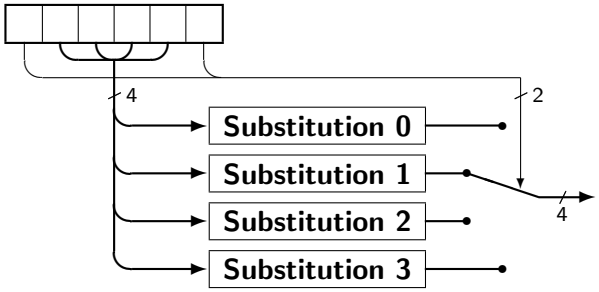


Abbildung 4: Arbeitsweise einer S-Box

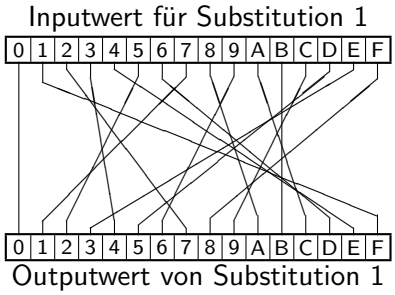


Abbildung 5: Eine der vier 4-Bit-Substitutionen von S1

| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **S1** | 0: | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |
| | 1: | 0 | F | 7 | 4 | E | 2 | D | 1 | A | 6 | C | B | 9 | 5 | 3 | 8 |
| | 2: | 4 | 1 | E | 8 | D | 6 | 2 | B | F | C | 9 | 7 | 3 | A | 5 | 0 |
| | 3: | F | C | 8 | 2 | 4 | 9 | 1 | 7 | 5 | B | 3 | E | A | 0 | 6 | D |

Abbildung 6: Die S-Box S1 als Tabelle

nun vor jeder der 16 Runden zyklisch um 1 oder 2 Bits nach links verschoben. Da die Anzahl der Schiebeoperationen über alle 16 Runden hinweg 28 beträgt, enthalten C und D am Ende einer Ver- oder Entschlüsselungsoperation wieder den ursprünglichen Wert, so daß die Register für weitere Datenblöcke nicht neu geladen werden müssen.

Um in jeder Runde einen Teilschlüssel für die XOR-Verknüpfung mit dem expandierten Register R zu gewinnen, ist eine weitere Permutationsfunktion PC-2 vonnöten. Sie wählt 48 Bits der 56 Bits aus und ändert deren Reihenfolge.

Beim Data Encryption Standard unterscheidet sich eine Entschlüsselungsoperation nur geringfügig von einer Verschlüsselungsoperation. Es müssen lediglich die Teilschlüssel der 16 Runden in umgekehrter Reihenfolge verwendet werden, der Rest des Algorithmus bleibt unverändert.

DES ist in Hardware wesentlich einfacher und schneller zu implementieren als in Software – tatsächlich war dies eines der Designkriterien. Der größte Nachteil bei einer Softwareimplementation sind die Permutationen, nicht zuletzt, weil die Eingangs- und Ausgangspermutationen IP und $IP^{-1}$ ebenso wie die Schlüsselpermutation PC-1 nichts zur Sicherheit des Algorithmus beitragen. Bei der Realisierung in Hardware erleichtern sie dagegen den Entwurf eines Bausteins für den Fall, daß Daten und Schlüssel Byte für Byte eingelesen werden.

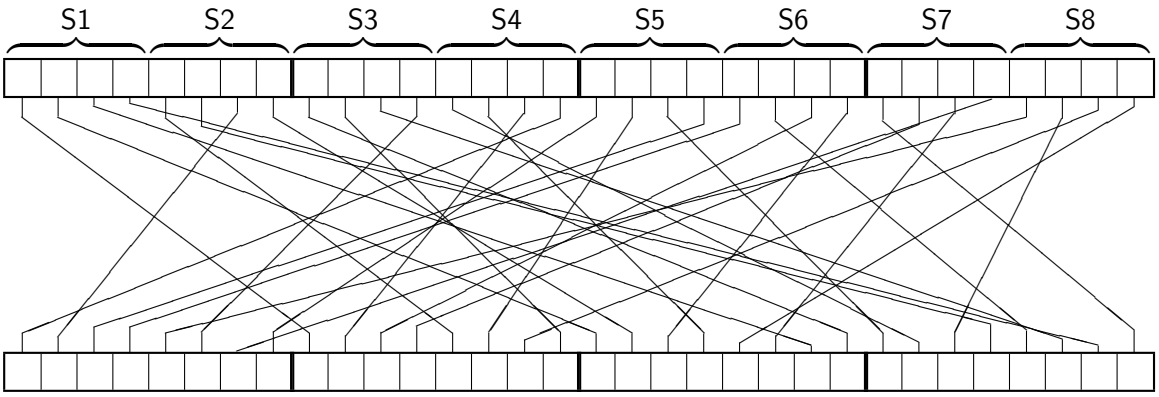


Abbildung 7: Die Permutation P

Als in den 80er Jahren trotz wachsenden Zweifeln an der Sicherheit des Algorithmus immer noch keine Alternative vorhanden war, ging man bei Anwendungen mit hohen Sicherheitsansprüchen dazu über, die Daten dreifach zu verschlüsseln (sogenanntes *Triple-DES*). Bei diesem Verfahren wird zunächst mit einem Schlüssel verschlüsselt, mit einem zweiten entschlüsselt und schließlich nochmals mit dem ersten Schlüssel verschlüsselt:

$$C = \mathrm{DES}(K_1, \mathrm{DES}^{-1}(K_2, \mathrm{DES}(K_1, M)))$$

Die dreifache Anwendung des Algorithmus ist nötig, weil eine doppelte Verschlüsselung die Sicherheit nur geringfügig erhöht.

### 2.2.3 Eigenschaften von DES

DES weist einige Besonderheiten auf. Da es sich um eine Blockchiffre handelt, die Klartextblöcke in Chiffretextblöcke gleicher Größe überführt, ist es nicht verwunderlich, daß sie bijektiv ist. Dagegen scheint es zunächst außergewöhnlich, daß der Algorithmus bezüglich der bitweisen Invertierung invariant ist, d.h. $\mathrm{DES}(K,M) = \overline{\mathrm{DES}(\overline{K},\overline{M})}$. Dies ist darauf zurückzuführen, daß bei gleichzeitiger Invertierung des Datenblocks und des Teilschlüssels der Eingabewert für die S-Boxen gleich bleibt. Diese Eigenschaft kann dazu genutzt werden, den Aufwand eines Angriffs zu halbieren, wenn dem Angreifer ein Klartext und zwei Chiffretexte mit der Eigenschaft $C_1 = \mathrm{DES}(K,M)$ und $C_2 = \mathrm{DES}(K,\overline{M})$ bekannt sind. Der Avalanche-Effekt wird von DES in hohem Maße erreicht. Bereits nach fünf Runden ist DES vollständig.

Da es sich beim DES um den ersten von der National Security Agency mitentwickelten Algorithmus handelte, der veröffentlicht wurde, weckte er in der Fachwelt großes Interesse und wurde detailliert untersucht. Vor allem die mysteriösen Konstanten der acht S-Boxen erweckten Mißtrauen, da eine Hintertür in ihnen vermutet wurde. Verschiedene Versuche, die Konstanten durch andere Werte zu ersetzen, zeigten jedoch, daß die Original-Werte sehr gut gewählt sind; die Alternativvorschläge führten alle zu schwächeren Algorithmen. Besonders interessant ist in dieser Hinsicht, daß die Konstanten so gewählt wurden, daß die Chiffre einem Angriff mittels der sogenannten differentiellen Kryptoanalyse bestmöglich widerstehen kann, was bedeutet, daß der NSA dieses Verfahren bereits 1974 bekannt war – die internationale Fachwelt „entdeckte“ die differentielle Analyse erst 1990! Gegen die lineare Kryptoanalyse ist DES dagegen nicht geschützt.

Ein mögliches Sicherheitsproblem stellen die schwachen Schlüssel des DES dar. Hierbei handelt es sich um externe Schlüssel, die dazu führen, daß nach der Anwendung der Permutationsfunktion PC-1 die Register C und D jeweils nur gesetzte oder nur gelöschte Bits enthalten, so daß die 16 internen Teilschlüssel zwangsläufig alle identisch sind. Mit solchen Schlüsseln verschlüsselte Nachrichten sind erheblich leichter zu knacken. Neben den vier schwachen Schlüsseln existieren auch noch 12 semi-schwache Schlüssel, bei denen die internen Schlüssel nur zwei verschiedene Werte annehmen.

Mit Sicherheit der gravierendste Schwachpunkt des Algorithmus ist jedoch die geringe Mächtigkeit des Schlüsselraums von $2^{56}$. Dadurch wird ein *exhaustive search* nach dem richtigen Schlüssel, d.h. Durchprobieren aller Möglichkeiten, erlaubt, was die anderen positiven Eigenschaften der Chiffre zunichte macht. Da es die NSA war, die die 128 Schlüsselbits des ursprünglichen Vorschlags von IBM auf nur 56 Bits verringerte, kann spekuliert werden, daß dieser Effekt erwünscht ist.

Zusammenfassend muß festgestellt werden, daß die Mängel von DES inzwischen so schwerwiegend sind, daß das Verfahren heute nicht mehr als sicher eingestuft werden kann. Auch das Ausweichen auf Triple-DES stellt nur eine Notlösung dar, weil bei seiner Verwendung Geschwindigkeitseinbußen in Kauf genommen werden müssen.

## 2.3 International Data Encryption Algorithm (IDEA)

### 2.3.1 Geschichte von IDEA

Als Alternative zu DES entwickelten Xuejia Lai und James L. Massey eine Blockchiffre, die neben einer höheren Sicherheit, v.a. durch einen größeren Schlüsselraum, auch schnellere Implementationen in Hardware und Software bot. Der Algorithmus wurde 1990 als *Proposed Encryption Standard* (PES) vorgestellt, es zeigte sich jedoch, daß er gegen die kurz darauf veröffentlichte differentielle Kryptoanalyse nicht immun war, so daß der Algorithmus geringfügig modifiziert wurde. Diese Variante bezeichnete man als *Improved PES* (IPES), später IDEA.

### 2.3.2 Beschreibung des Algorithmus

IDEA arbeitet ebenso wie DES auf 64-Bit-Datenblöcken, die Schlüssellänge beträgt dagegen 128 Bit. Eine Verschlüsselungsoperation besteht aus acht Runden gefolgt von einer Ausgabetransformation. Da der Algorithmus schnell in Software zu implementieren sein soll, wird auf Permutationen verzichtet und die Grundoperationen beschränken sich auf XOR, Addition modulo $2^{16}$ sowie Multiplikation modulo $2^{16}+1$, wobei der Operand 0 als $2^{16}$ interpretiert wird.

Die Funktionsweise von IDEA ist in Abbildung 8 für die erste Runde und die abschließende Ausgabetransformation dargestellt. Der 64 Bits breite Klartextblock $M$ wird zunächst in vier Teilblöcke $M_1$ bis $M_4$ zu je 16 Bits aufgeteilt. In jeder Runde $r = 1 \ldots 8$ werden nun beeinflußt von sechs Teilschlüsseln $K_{r,1}$ bis $K_{r,6}$ verschiedene Operationen durchgeführt, deren Ergebnis nach einer Permutation der Teilblöcke von der nächsten Runde weiterverarbeitet wird.

Insgesamt werden sechs Schlüssel für jede der acht Runden zuzüglich weiterer vier für die Ausgabetransformation benötigt, also zusammen 52 Teilschlüssel zu je 16 Bits. Jeweils vier der Teilschlüssel werden aus dem 128-Bit-Schlüssel generiert, indem dieser in acht 16-Bit-Blöcke auf-

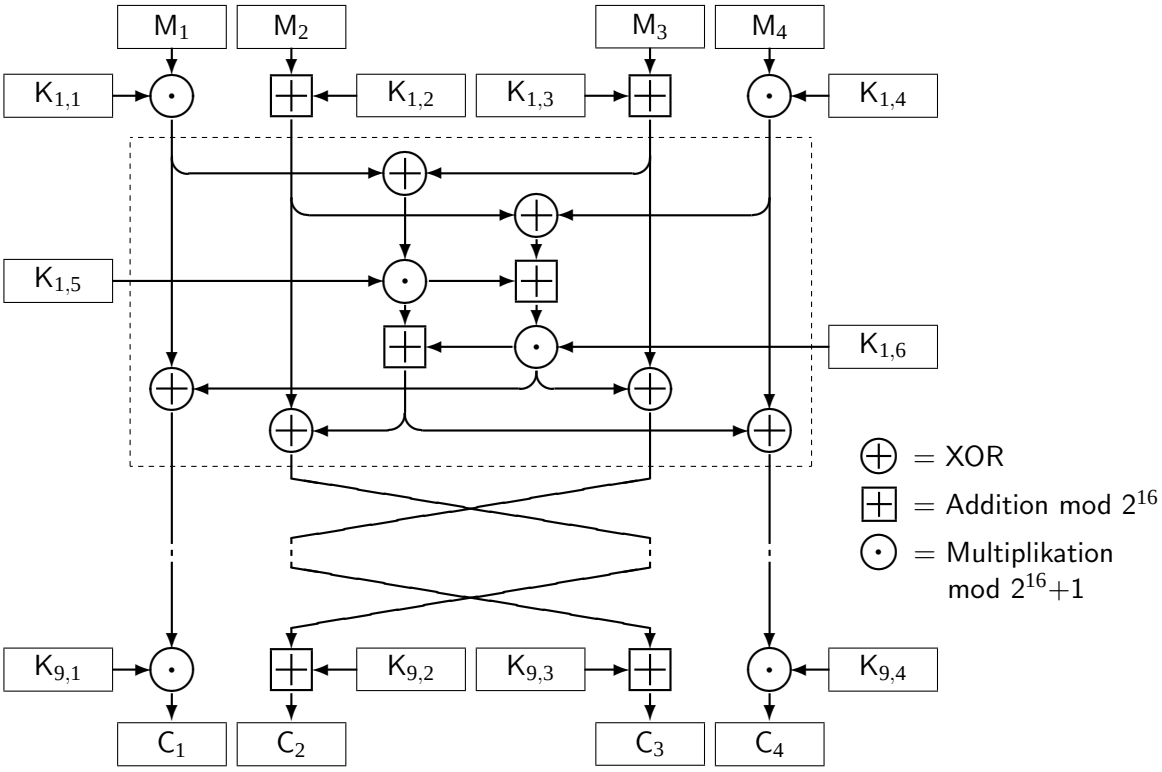


Abbildung 8: Der *International Data Encryption Algorithm*

geteilt wird. Nachdem der 128-Bit-Schlüssel zyklisch um 25 Bits nach links verschoben wurde, steht er dazu bereit, für die nächsten vier Teilschlüssel aufgeteilt zu werden.

Die Entschlüsselung gestaltet sich größtenteils analog zur Verschlüsselung, die Teilschlüssel müssen lediglich in umgekehrter Reihenfolge verwendet werden. Darüber hinaus werden einige der Teilschlüssel verändert: Der in Abbildung 8 gestrichelt umrandete Teil der Operationen ist selbstinvers, so daß $K_{r,5}$ und $K_{r,6}$ unverändert bleiben können. Für $r = 1 \ldots 9$ müssen jedoch $K_{r,1}$ sowie $K_{r,4}$ bezüglich der Multiplikation modulo $2^{16}+1$ und außerdem $K_{r,2}$ sowie $K_{r,3}$ bezüglich der Addition modulo $2^{16}$ invertiert werden. (Es ist stets gewährleistet, daß für die Multiplikation ein Inverses existiert, da es sich bei $2^{16}+1$ um eine Primzahl handelt.)

Beim IDEA existieren ähnlich wie beim DES schwache Schlüssel, die sich dadurch auszeichnen, daß interne Teilschlüssel den Wert Null oder Eins annehmen. Null zeigt bei der Addition und XOR-Verknüpfung keine Wirkung, bei der Multiplikation ist Null selbstinvers, da laut Konvention $0 = 2^{16} = -1$. Ein Teilschlüssel mit dem Wert 1 hat bei der Multiplikation keinen Effekt. Dies bedeutet v.a., daß die Anwendung von IDEA mit einem externen Schlüssel, bei dem alle Bits gelöscht sind, selbstinvers ist. Es existieren weitere schwache Schlüssel, die einen Angriff auf den Algorithmus ermöglichen. Andere Schwächen wurden in der Chiffre nicht gefunden.

# 3 Betriebsarten

Die Betriebsart legt fest, wie sich die Verschlüsselung mehrerer Klartextblöcke vollzieht. Je nach den Anforderungen der Anwendung variiert die Fehleranfälligkeit und Sicherheit.

## 3.1 Electronic Codebook (ECB)

In der einfachsten Betriebsart wird jeder Klartextblock einzeln mit demselben Schlüssel verschlüsselt. Dies hat den Vorteil, daß die Entschlüsselung der Chiffretextblöcke in beliebiger Reihenfolge stattfinden kann.

Allerdings bringt diese Betriebsart ein großes Sicherheitsrisiko mit sich, weil Klartextblöcke stets auf dieselben Chiffretextblöcke abgebildet werden: Ist z.B. jemand in der Lage, die zwischen zwei Banken übermittelten Daten abzuhören und zu verändern, kann er zwei Konten bei diesen Banken eröffnen und zwischen ihnen Geldbeträge überweisen. Wenn die von einer Bank zur anderen übertragene Nachricht für eine identische Überweisung immer mit dem gleichen Schlüssel verschlüsselt ist, kann er die zugehörigen Chiffretextblöcke identifizieren und schließlich selbst mehrfach einspeisen. Eine andere Möglichkeit bestünde darin, bei bekanntem Datenformat nur die Blöcke anderer Überweisungen zu ändern, die die Kontonummer enthalten.

Ändert sich durch einen Übertragungsfehler der Chiffretextblock, wird er nicht korrekt entschlüsselt, die Störung beschränkt sich aber auf diesen einen Block. Bei Synchronisationsfehlern, die zusätzliche Bits einfügen oder löschen, känn dagegen keiner der folgenden Blöcke korrekt dechiffriert werden.

In der Praxis wird ECB hauptsächlich für die Verschlüsselung von Schlüsseln verwendet.

## 3.2 Cipher Block Chaining (CBC)

Im CBC-Modus findet eine Verkettung der einzelnen Blöcke statt; wie aus Abbildung 9 ersichtlich wird jeder Klartextblock vor der Verschlüsselung mit dem vorhergehenden Chiffretextblock XOR-verknüpft. Der Empfänger der verschlüsselten Daten kehrt das Verfahren um und verknüpft jeden Block, nachdem er entschlüsselt wurde, mit dem vorangegangenen Chiffretextblock.

Für den ersten zu verschlüsselnden Block steht kein Vorgänger-Chiffretextblock zur Verfügung. Man könnte zwar einfach den Wert Null verwenden, dies würde jedoch dazu führen, daß für identische Nachrichten gleicher Chiffretext erzeugt würde, und darüber hinaus zulassen, daß ein Angreifer erkennen kann, wenn verschiedene verschlüsselte Nachrichten den gleichen Anfang besitzen.

Aus diesen Gründen wird der erste Klartextblock mit einem meist zufällig gewählten Initialisierungsvektor verknüpft. Dieser Wert muß auch an den Empfänger übermittelt werden, wofür keine

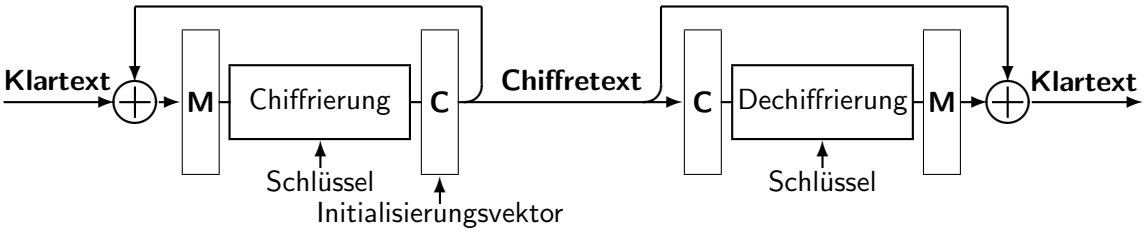


Abbildung 9: Der CBC-Modus

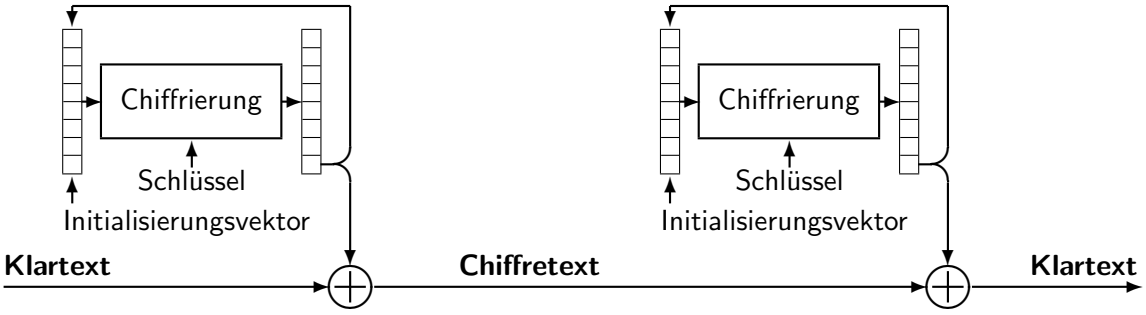


Abbildung 10: Der OFB-Modus

Verschlüsselung nötig ist. Allerdings wird der Initialisierungsvektor dennoch oft im ECB-Modus verschlüsselt, um zu verhindern, daß ein Angreifer den ersten Klartextblock gezielt abändert – bei den nachfolgenden Blöcken ist dies aufgrund der Rückkopplung nicht mehr möglich.

Ändert sich bei der Übertragung ein Bit des Chiffretexts, werden zwei Blöcke unbrauchbar, denn ein Chiffretextblock wird nicht nur selbst entschlüsselt, sondern findet auch für die Verknüpfung mit dem nachfolgenden Block Verwendung. Wie schon beim ECB-Modus führen auch bei Benutzung von CBC Synchronisationsfehler dazu, daß keiner der folgenden Blöcke korrekt entschlüsselt wird.

Der CBC-Modus bietet eine höhere Sicherheit als ECB, hat jedoch den Nachteil, daß für einen Zugriff auf einen Block alle vorhergehenden Blöcke ebenfalls entschlüsselt werden müssen. Er wird häufig zum Verschlüsseln von Dateien verwendet.

## 3.3 Output Feedback (OFB) und Cipher Feedback (CFB)

Jede Blockchiffre kann mithilfe des OFB- bzw. CFB-Modus auch als Stromchiffre benutzt werden. Stromchiffren werden z.B. eingesetzt, wenn einzelne Zeichen übertragen werden sollen, ohne daß mit der Übertragung gewartet werden kann, bis genug Zeichen für einen ganzen Block vorhanden sind.

In diesen Betriebsmodi wird der Verschlüsselungsalgorithmus als Pseudozufallsfolgengenerator verwendet. Sowohl beim Sender als auch beim Empfänger arbeitet die Chiffre nur verschlüsselnd und erzeugt denselben Schlüsselstrom, der mit den zu übertragenden Daten XOR-verknüpft wird.

In beiden Modi ist der erste verschlüsselte Inputblock ein Initialisierungsvektor, der zunächst zum Empfänger übertragen werden muß (dies kann unverschlüsselt geschehen).

Soll nun bei Verwendung des OFB-Modus (siehe Abbildung 10) ein $m$-Bit-Zeichen übertragen werden, wird zunächst eine Verschlüsselungsoperation der Blockchiffre durchgeführt. $m$ Bits des resultierenden Outputblocks werden vor der Übertragung mit den $m$ Klartextbits XOR-verknüpft, gleichzeitig ersetzen sie auch $m$ Bits des alten Inputblocks.

Bei Synchronisationsfehlern kann der Empfänger keines der nachfolgenden Zeichen korrekt entschlüsseln, Übertragungsfehler, die den Wert einzelner Bits ändern, wirken sich dagegen nur auf die Stelle aus, an der sie auftreten. Deswegen bietet diese Betriebsart keinen Schutz vor Manipulationen einzelner Chiffretextzeichen.

Der CFB-Modus arbeitet ähnlich wie der OFB-Modus, mit dem Unterschied, daß für das Ersetzen von $m$ Bits des alten Inputblocks nicht auf einen Teil des Outputblocks, sondern auf das erzeugte

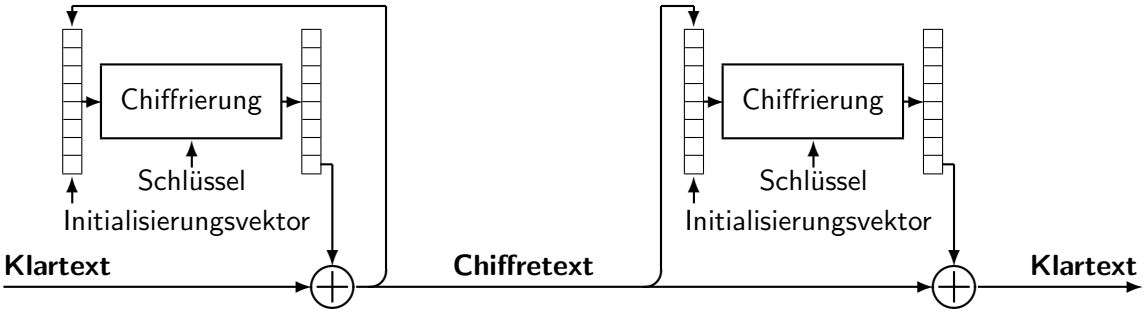


Abbildung 11: Der CFB-Modus

Chiffretextzeichen zurückgegriffen wird. Wie in Abbildung 11 gezeigt wird, muß dieses Verfahren ähnlich wie beim CBC-Modus auf der Empfängerseite umgekehrt werden – das Chiffretextzeichen ersetzt einen Teil des Inputblocks und wirkt sich so auf die Entschlüsselung des nachfolgenden Zeichens aus.

Aufgrund der Rückkopplung beeinflußt eine Änderung eines Chiffretextzeichens alle folgenden Zeichen, so daß Übertragungsfehler und Manipulationen erkannt werden können. Bei Synchronisationsfehlern kann auch mit dieser Betriebsart keines der folgenden Zeichen entschlüsselt werden, außer für $m = 1$: In diesem Fall wird der Bitstrom wieder korrekt entschlüsselt, sobald das fehlende oder zusätzliche Bit nicht mehr den Inputblock der Empfängerseite verfälscht.

# 4 Zusammenfassung

Für lange Jahre stellte DES das dominierende symmetrische Verschlüsselungsverfahren dar, trotz kontroverser Diskussionen über seine Sicherheit, vor allem im Hinblick auf die geringe Schlüssellänge. Als Zwischenlösung wurde auf Triple-DES zurückgegriffen, das jedoch für viele Anwendungen zu langsam ist. Andere Algorithmen, wie z.B. IDEA, sind patentiert und müssen lizensiert werden.

Als Antwort auf das wachsende Bedürfnis, DES zu ersetzen, hat das NIST im Jahr 1997 ein Programm für einen *Advanced Encryption Standard* ins Leben gerufen und dazu aufgerufen, Vorschläge für eine neue symmetrische Blockchiffre einzubringen, die über einen größeren Schlüsselraum, eine größere Blockgröße, höhere Geschwindigkeit und größere Flexibilität verfügt als DES. Einige vielversprechende Algorithmen wurden vorgeschlagen – einer von ihnen soll nach dem Willen des NIST in Zukunft DES als Standard ersetzen, so daß wieder ein sicherer Algorithmus allgemein vefügbar ist.

# 5 Weitere Informationen

- ◇ `http://www.cryptography.com/des/`
  Informationen über den Spezialcomputer, der DES in 56 Stunden knackte,

- ◇ `http://csrc.nist.gov/encryption/aes/aes_home.htm`
  Einzelheiten zum *Advanced Encryption Standard*,

- ◇ `ftp://rtfm.mit.edu/pub/usenet-by-group/sci.crypt`
  FAQ der Newsgroup *sci.crypt*,
- ◇ `ftp://ftp.funet.fi/pub/crypt/cryptography/symmetric/des/`
  Verschiedene Implementationen von DES und Triple-DES in C,
- ◇ `ftp://ftp.funet.fi/pub/crypt/cryptography/papers/idea.chap.3.ps.gz`
  `ftp://ftp.funet.fi/pub/crypt/cryptography/symmetric/idea/`
  Detaillierte Beschreibung von IDEA und C-Sourcecode,
- ◇ `http://www.counterpane.com/`
  Bruce Schneiers Firma, mit vielen Informationen auch über die eigenene Algorithmen.

## Literatur


[Schneier] Bruce Schneier, *Angewandte Kryptographie*, Addison-Wesley 1996

[FumyRiesz] W. Fumy, H. P. Riesz, *Kryptographie*, Oldenbourg-Verlag