Moderne Kryptografie

- Im gegensatz zur klassischen Kryptografie meint die moderne Kryptografie die Verfahren, die meist unter Einsatz eines Computers benutzt werden.
- Zeichen wie Buchstaben und Zahlen bilden nicht mehr die Grundlage, sondern die viel tiefere Ebene der Bits und Bytes. Dadurch wird eine ganz andere Dimension der möglichen Transformationen bereitgestellt.

Zwei Klassen werden unterschieden:

- **Symmetrische Verfahren**
- **Asymmetrische Verfahren**

Symmetrische Verfahren

- Bei den symmetrischen Verfahren wird von beiden Parteien derselbe Schlüssel verwendet.
- Der Absender verwendet diesen Schlüssel und einen Verschlüsselungsalgorithmus, um Daten zu verschlüsseln.
- Der Empfänger verwendet denselben Schlüssel und den entsprechenden Entschlüsselungsalgorithmus zum Entschlüsseln der Daten.
- Symmetrische Verschlüsselung (wird auch als Verschlüsselung mit privatem Schlüssel bezeichnet).

Symmetrische Verfahren

- Claude Shannon hat bewiesen, dass Private-Key-Systeme Schlüssel verwenden müssen, die mindestens so lang sind wie die zu verschlüsselnde Nachricht (*perfect secrecy*)
- Daher reicht für moderne Kryptographie die sogenannte rechnerische Sicherheit.
- Die symmetrische Verschlüsselung verwendet möglicherweise einen 256-Bit-Schlüssel, der keine perfekte Sicherheit nach Shannon bietet, wenn die Nachricht länger als 256 Bit ist.
- Für einen zufälligen 256-Bit-Schlüssel würden derzeit alle Rechenressourcen der Welt benötigt länger als das Universum bisher existiert. Dies wird allgemein als ausreichend angesehen.

Symmetrische Verfahren

- Für die symmetrische Verschlüsselung muss ein sicherer Kanal zum Austauschen des Schlüssels verwendet werden, was die Nützlichkeit dieser Art von Verschlüsselungssystem erheblich beeinträchtigt.
- Der Hauptnachteil eines Kryptosystems mit geheimen Schlüsseln hängt mit dem Austausch von Schlüsseln zusammen.
- Darüber hinaus muss ein Benutzer, der mit mehreren Personen kommunizieren und gleichzeitig eine separate Vertraulichkeitsstufe sicherstellen möchte, so viele private Schlüssel verwenden, wie es Personen gibt.
- von N Personen, die ein Kryptosystem mit geheimen Schlüsseln verwenden, ist es erforderlich, eine Anzahl von Schlüsseln zu verteilen, die $\frac{N*(N-1)}{2}$ entspricht.

Symmetrische Verfahren

Ein wichtiger Unterschied bei symmetrischen kryptografischen Algorithmen besteht in der Unterscheidung zwischen Stream- und Block-Chiffren.

Stream cipher: Stream-Chiffren wandeln ein Symbol von Klartext direkt in ein Symbol von Chiffretext um.

Vorteile:

- Transformationsgeschwindigkeit: Algorithmen sind zeitlich linear und räumlich konstant.
- Geringe Fehlerausbreitung: Ein Fehler beim Verschlüsseln eines Symbols hat wahrscheinlich keine Auswirkungen auf nachfolgende Symbole.

Symmetrische Verfahren

Nachteile:

- Geringe Verbreitung: Alle Informationen eines Klartextsymbols sind in einem einzelnen Chiffretextsymbol enthalten.

- Anfälligkeit für Einfügungen / Änderungen: Ein Attacker, der den Algorithmus bricht, fügt möglicherweise falschen Text ein, der authentisch aussieht.

Symmetrische Verfahren

Block ciphers: Es verschlüsselt eine Gruppe von Klartextsymbolen als einen Block.

Vorteile:

- Hohe Verbreitung: Informationen von einem Klartextsymbol werden in mehrere Chiffretextsymbole verteilt.
- Manipulationssicherheit: Es ist schwierig, Symbole einzufügen ohne erkannt zu werden.

Nachteile:

- Langsam: Ein ganzer Block muss angesammelt werden, bevor die Verschlüsselung / Entschlüsselung beginnen kann.
- Fehlerausbreitung: Ein Fehler in einem Symbol kann den gesamten Block beschädigen.

Feistel-Netzwerke

Produktalgorithmen: Bei einem Produktalgorithmus werden mehrere einfache, kryptographisch unsichere Schritte (genannt Runden) nacheinander ausgeführt. Diese Kombination der einfachen Funktionen kann die Sicherheit stark erhöhen.

- Feistel-Netzwerke wurden erstmals 1973 von Horst Feistel in seinem Artikel *"Cryptography and Computer Privacy"* beschrieben. Sie dienen der Beschreibung einer Runde in einem Produktalgorithmus.
- Viele moderne symmetrische Verschlüsselungsalgorithmen basieren auf Feistelnetzwerken.
- Blockverschlüsselungen, welche auf Feistelnetzwerken basieren sind garantiert umkehrbar (bijektiv). Damit ist die notwendige Grundbedingung für Blockchiffren erfüllt, dass es bei der Abbildung von Chiffreblöcken auf Klartextblöcke bei der Entschlüsselung zu keinen Mehrdeutigkeiten kommen darf.

Feistel-Netzwerke

- Ein Block wird zuerst in zwei (meist gleich große) Teile geteilt:

$$\boldsymbol{P = L_1 \,||\, R_1}$$

- Es wird in n aufeinanderfolgenden Runden verarbeitet.
- In jeder Runde wird einer dieser Teile mit der Ausgabe einer Rundenfunktion verknüpft.
- Die Rundenfunktion erhält den anderen Teilblock und einen Rundenschlüssel als Eingabe.
- Innerhalb der i-ten Runde (i läuft von 1 bis n) wird folgende Formel angewendet:

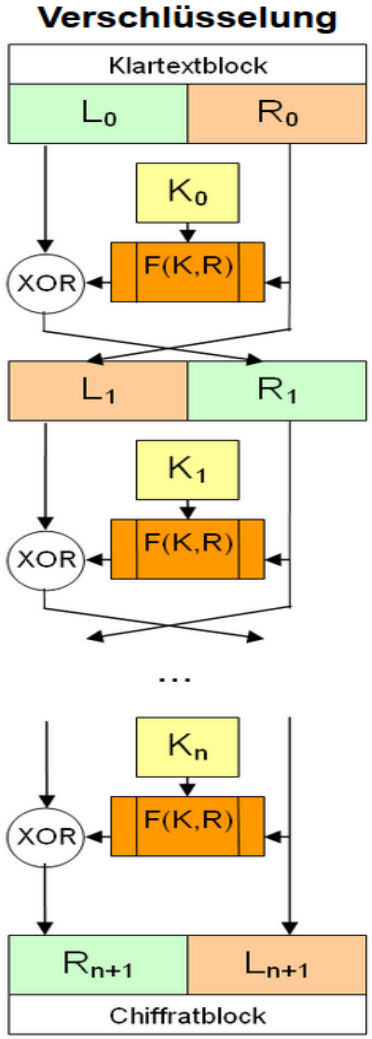
$$\boldsymbol{L_{i+1} = R_i}$$

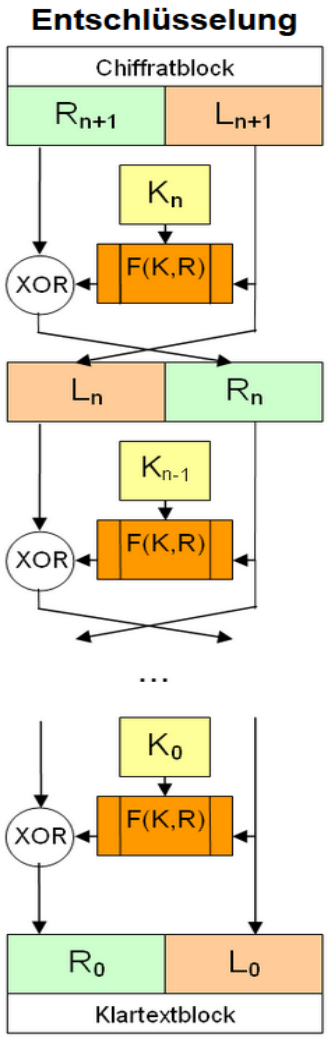
$$\boldsymbol{R_{i+1} = L_i \oplus F(R_i, K_i)}$$

Feistel-Netzwerke

- Dabei bildet F die sog. Runden- oder Transformationsfunktion und K_1 bis K_n sind die Rundenschlüssel.
- $\oplus$ steht für eine einfach umkehrbare Verknüpfung. Oft verwendet man das bitweise XOR, das mit seiner Umkehrung identisch, d. h. selbstinvers ist.
- Der verschlüsselte Text am Ende der Runden ist die Zusammenführung $\boldsymbol{C = L_{n+1} \,||\, R_{n+1}}$
- Feistelnetzwerke ermöglichen eine Entschlüsselung, ohne dass die Umkehrfunktion von F benötigt wird.
- Will man einen Geheimtext dechiffrieren, führt man die Schritte der obigen Formel in umgekehrter Reihenfolge aus, wobei man i von n bis 1 laufen lässt:

Feistel-Netzwerke





Feistel-Netzwerke

Beispiel mit 2 Runden:
$K_0 = 1000, K_1 = 0100, x = 10101110$

Als Vorbereitung **teilen wir die zu verschlüsselnde Nachricht** in einen **rechten** und **linken Teil** auf, also:

$$x = (\underbrace{1010}_{L} \mid \underbrace{1110}_{R})$$

Für die Verschlüsselung bei der **Feistel-Chiffre** gilt:

$$\boldsymbol{L_{i+1} = R_i}$$

$$\boldsymbol{R_{i+1} = L_i \oplus F(R_i, K_i)}$$

Feistel-Netzwerke

Beispiel mit 2 Runden:
$K_0 = 1000$, $K_1 = 0100$, $x = 10101110$

1. Runde:

$$\boldsymbol{L_1 = R_0 = 1110}$$
$$\boldsymbol{R_1 = L_0 \oplus F(R_0, K_0) = 1010 \oplus (1110 \oplus 1000) = 1010 \oplus 0110 = 1100}$$

2. Runde:

$$\boldsymbol{L_2 = R_1 = 1100}$$
$$\boldsymbol{R_2 = L_1 \oplus F(R_1, K_1) = 1110 \oplus (1100 \oplus 0100) = 1110 \oplus 1000 = 0110}$$

Die **verschlüsselte Nachricht** lautet also $\boldsymbol{11000110}$.

Data Encryption Standard (DES)

- Bei DES handelt es sich um einen symmetrischen Algorithmus, das heißt zur Ver- und Entschlüsselung wird derselbe Schlüssel verwendet.
- DES funktioniert als Blockchiffre, jeder Block wird also unter Verwendung des Schlüssels einzeln chiffriert, wobei die Daten in 16 Runden von Substitutionen und Transpositionen (Permutation) nach dem Schema von Feistel verwürfelt werden.
- Die Blockgröße beträgt 64 Bits, das heißt ein 64-Bit-Block Klartext wird in einen 64-Bit-Block Chiffretext transformiert.
- Auch der Schlüssel, der diese Transformation kontrolliert, besitzt 64 Bits. Jedoch stehen dem Benutzer von diesen 64 Bits nur 56 Bits zur Verfügung; die übrigen 8 Bits (jeweils ein Bit aus jedem Byte) werden zum Paritäts-Check benötigt.
- Die effektive Schlüssellänge beträgt daher nur 56 Bits.

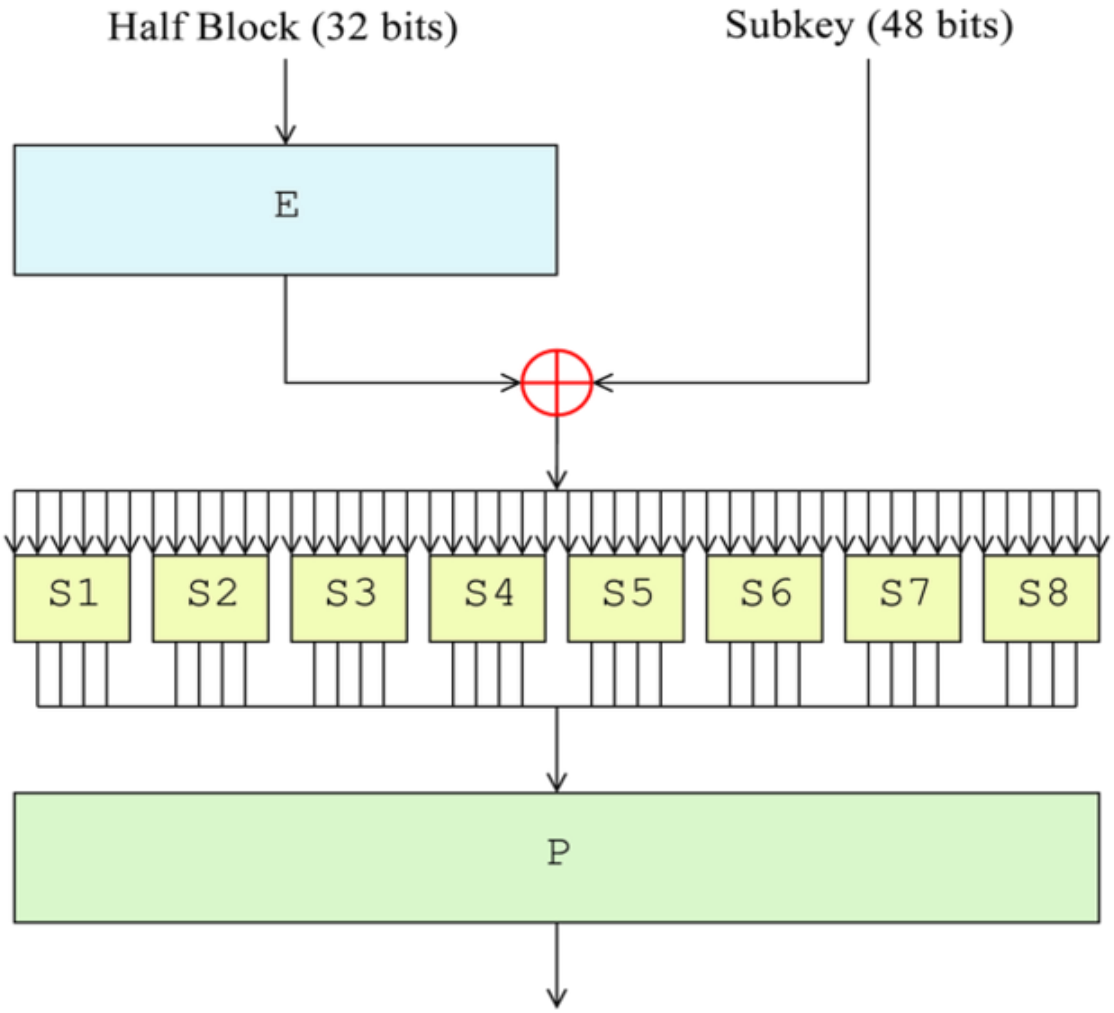
Data Encryption Standard (DES)

- Auf den 64 Bit Block wird eine initiale Permutation angewandt. Danach wird der Block in zwei Teile aufgeteilt und jeder Teil in ein 32 Bit Register gespeichert.
- Die beiden Blockhälften werden in Folge als linke und rechte Hälfte bezeichnet. Auf die rechte Blockhälfte wird die Feistel-Funktion angewandt.
- Danach wird die rechte Hälfte mit der linken Hälfte XOR verknüpft und das Ergebnis im Register der nächsten Runde für die rechte Hälfte gespeichert.
- In das linke Register der nächsten Runde wird die ursprüngliche rechte Blockhälfte kopiert.
- Nach Ende der letzten Runde werden die beiden Hälften wieder zusammengeführt und eine finale Permutation durchgeführt.

Data Encryption Standard (DES)

- Die *F*-Funktion von DES arbeitet auf Halbblöcken zu je 32 Bits und besteht aus vier Phasen:
  1. Die R-Blöcke werden mittels einer geeigneten Permutation *E* (*Expansion*) auf 48 Bits Länge expandiert, indem einzelne Bits mehrfach verwendet werden.
  2. Das Ergebnis wird mit einem Teilschlüssel XOR-verknüpft. Für jede Runde wird hierzu nach einer festen Vorschrift ein anderer 48-Bit-Teilschlüssel aus dem Hauptschlüssel generiert.
  3. Die resultierenden Blöcke werden in acht 6-Bit-Stücke zerteilt und diese mittels Substitution durch S-Boxen auf eine Länge von 4 Bits komprimiert. Diese nicht-lineare Transformierung in den S-Boxen stellt das Herzstück der Sicherheit von DES dar, ohne sie wäre DES linear und trivial zu brechen.
  4. Die 32 Bits Ausgabe der S-Boxen werden mittels einer festen Permutation *P* rearrangiert.

Data Encryption Standard (DES)

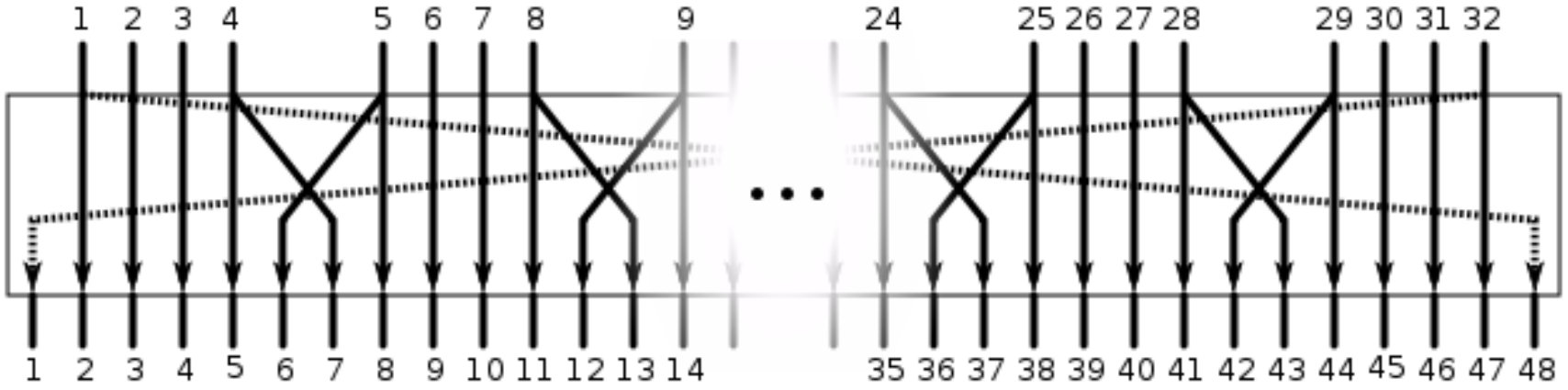


Eine Feistel-Runde (F-Funktion)

Data Encryption Standard (DES)

Die Expansion (1):

- Um den Halbblock in der Feistel-Funktion von 32 Bits auf 48 Bits zu erweitern, wird der Halbblock in 4-Bit-Gruppen aufgeteilt.
- Die Bits am Rand jeder 4-Bit-Gruppe werden vorn, beziehungsweise hinten an die benachbarte 4-Bit-Gruppe angehängt.



Data Encryption Standard (DES)

Die Substitution (3):

- Die Substitutionsboxen (S-Boxen) beim DES sind standardisiert.
- Ausgangwert wird aus Tabellen ermittelt.
- Dafür wird der Eingabewert gesplittet. So bildet das erste und letzte Bit zusammen die Zeile, und die Spalte ergibt sich aus den übrigen Bits.
- Eine Änderung dieser Boxen reduziert die Sicherheit drastisch!

S_1		Mittlere 4 Bits des Eingabewertes															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Äußere Bits	**00**	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
	01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
	10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
	11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

Betriebsmodi (DES)

- Der DES-Algorithmus beschreibt zunächst nur, wie ein Datenblock mit 64 Bits verarbeitet wird.
- Zur Verarbeitung einer Nachricht beliebiger Länge lässt sich der DES wie auch jede andere Blockchiffre in verschiedenen Betriebsmodi verwenden.
- Erst die Kombination von Blockchiffre und Betriebsmodus erlaubt es, Nachrichten zu verschlüsseln, die länger sind als die Blocklänge.
  - **Electronic Code Book Mode (ECB)**
  - **Cipher Block Chaining Mode (CBC)**
  - **Cipher Feedback Mode (CFB)**
  - **Output Feedback Mode (OFB)**
  - **Counter Mode (CTR)**

Betriebsmodi (DES)

Electronic Code Book Mode (ECB):

- Die einfachste Betriebsart, jeder Block wird unabhängig von den anderen verschlüsselt.
- Gleiche Nachrichtenblöcke werden auch gleich verschlüsselt.
- Eine Vertauschung von Blöcken im Chiffrat führt zur gleichen Vertauschung der Blöcke in der entschlüsselten Nachricht.
- Ein Fehler in einem Block beeinflusst nur die Entschlüsselung dieses Blocks.
- Von der Verwendung des ECB-Modus wird daher abgeraten, es sei denn, es soll nur einmal ein einziger Nachrichtenblock verschlüsselt werden.

Betriebsmodi (DES)

Cipher Block Chaining Mode (CBC):

- Ein Nachrichtenblock wird vor dem Verschlüsseln mit dem vorhergehenden Chiffratblock verknüpft.
- Für den ersten Block nehmen wir hierzu einen Initialisierungsvektor.
- Eine Umordnung der Blöcke ohne die Entschlüsselung zu beeinträchtigen ist nicht mehr möglich, da ein Chiffratblock nun von allen vorhergehenden Blöcken abhängt.
- Da die Verschlüsselung auch vom Initialisierungsvektor abhängt, werden zwei gleiche Nachrichten mit unterschiedlichen IVs auch unterschiedlich verschlüsselt.
- Ein Fehler in einem Block, z. B. bei der Übertragung des verschlüsselten Textes, wirkt sich nur auf den entsprechenden und den darauffolgenden Klartextblock aus.

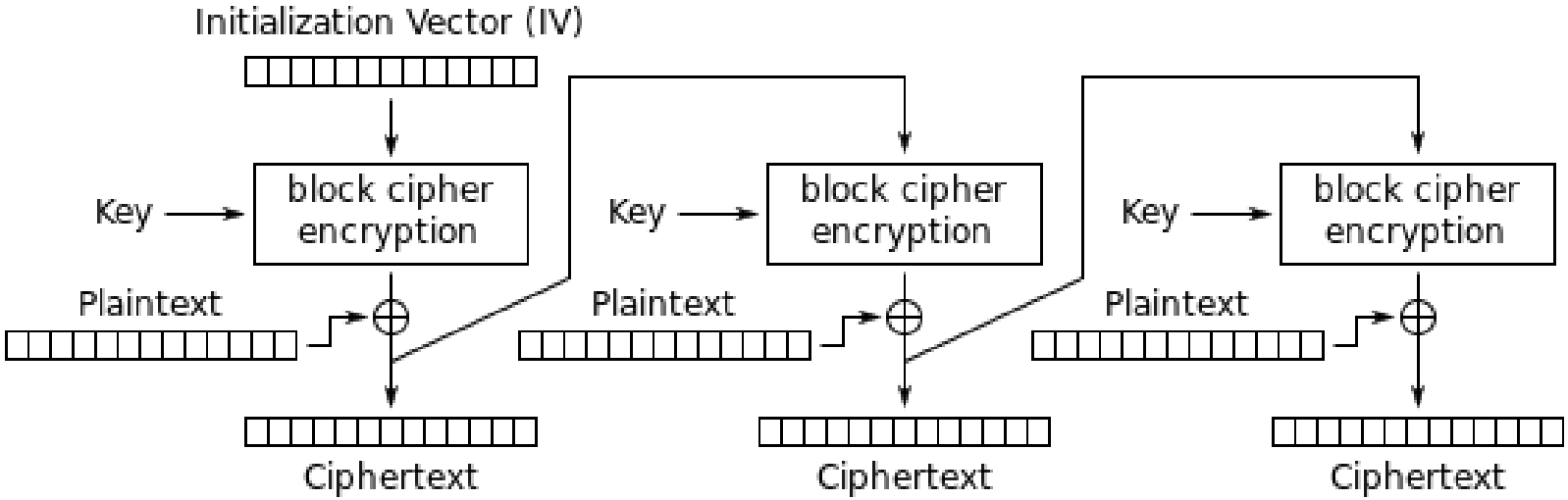
Betriebsmodi (DES)

Cipher Feedback Mode (CFB):

- Bei Nachrichten deren Länge kein Vielfaches der Blocklänge ist, kann der CFB verwendet werden.
- Ist eine Betriebsart (Modus), in der Blockchiffren als Stromchiffren betrieben werden.
- In diesem Modus wird, die Ausgabe der Blockchiffre mit dem Klartext bitweise XOR (exklusives ODER) verknüpft um daraus den Geheimtext zu bilden.
- Ist der Wert des Initialisierungsvektors nicht bekannt, so kann der Datenstrom ab dem zweiten Block trotzdem entschlüsselt werden.
- Der Initialisierungsvektor (IV) dient ähnlich wie bei dem Cipher Block Chaining (CBC) als Startwert.

Betriebsmodi (DES)

Cipher Feedback Mode (CFB):



Cipher Feedback (CFB) mode encryption

$$C_0 = P_0 \oplus E_k(IV)$$
$$C_i = P_i \oplus E_k(C_{i\_1})$$

Betriebsmodi (DES)

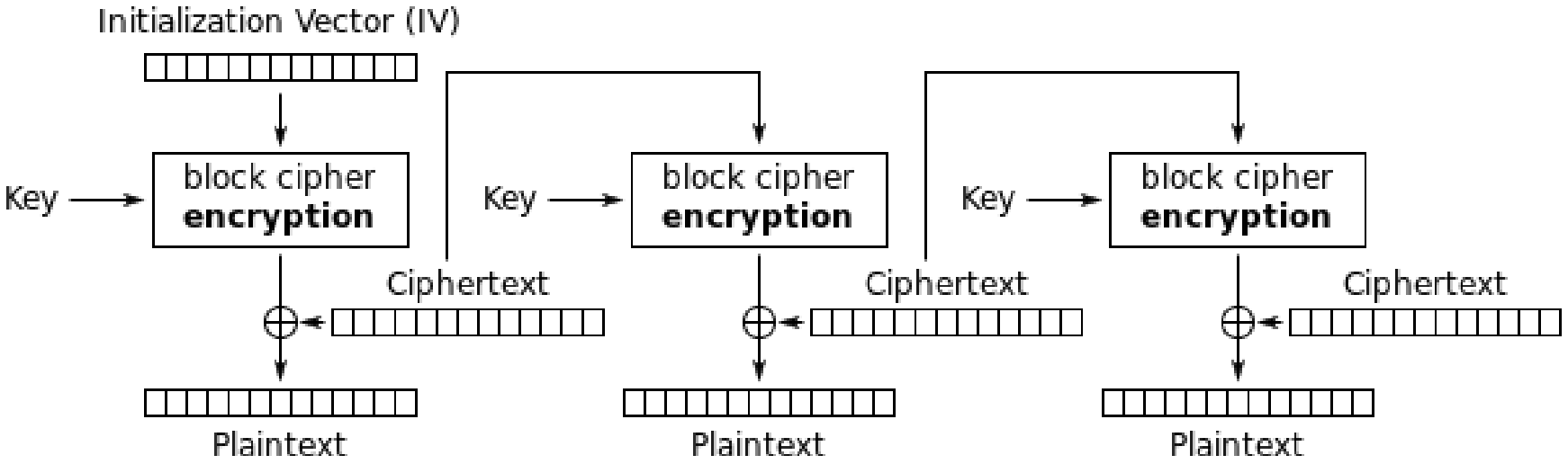
Cipher Feedback Mode (CFB):

- Die Entschlüsselung beim Empfänger, funktioniert wie Verschlüsselung, erzeugt also bei gleichem Initialisierungsvektor und gleichem Schlüssel die gleiche binäre Datenfolge mit der die XOR-Operation des Sender rückgängig gemacht werden kann.

- Der wesentliche Nachteil dieser Stromchiffre: Durch nur einen einzigen Bitfehler, der bei der Übertragung auftreten kann, wird im aktuellen Klartextdatenblock genau ein Bitfehler erzeugt und zusätzlich im nachfolgenden Datenblock im Mittel 50 % der Datenbits zerstört.

Betriebsmodi (DES)

Cipher Feedback Mode (CFB):



Cipher Feedback (CFB) mode decryption

$$P_0 = C_0 \oplus Ek(IV)$$
$$P_i = C_i \oplus E_k(C_{i\_1})$$

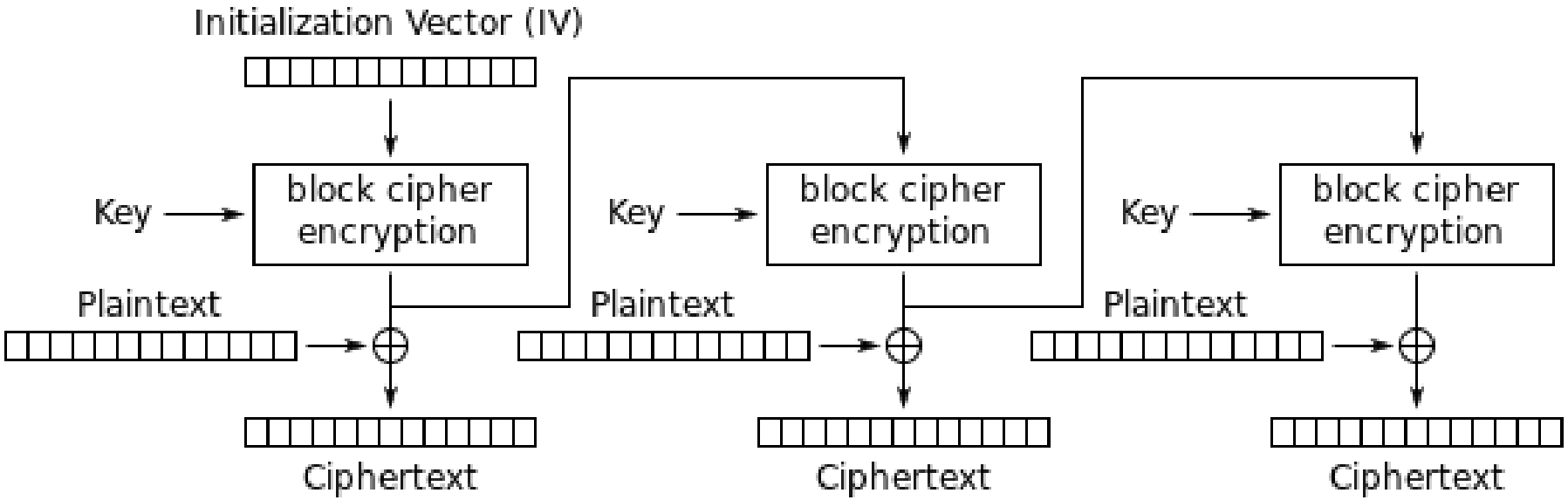
Betriebsmodi (DES)

Output Feedback Mode (OFB):

- Beim Output Feedback Mode wird im Unterschied zum CFB Mode nicht ein Chiffratblock, sondern die Ausgabe der Verschlüsselungsfunktion als Feedback genutzt.
- Dadurch wird jede Fehlerfortpflanzung vermieden.
- Der Schlüsselstrom ist unabhängig von der Nachricht. Das bedeutet auch, dass für jede Nachricht ein anderer Initialisierungsvektor benutzt werden muss, da der gesamte Schlüsselstrom nur von ihm abhängt.

Betriebsmodi (DES)

Output Feedback Mode (OFB):



Output Feedback (OFB) mode encryption

Betriebsmodi (DES)

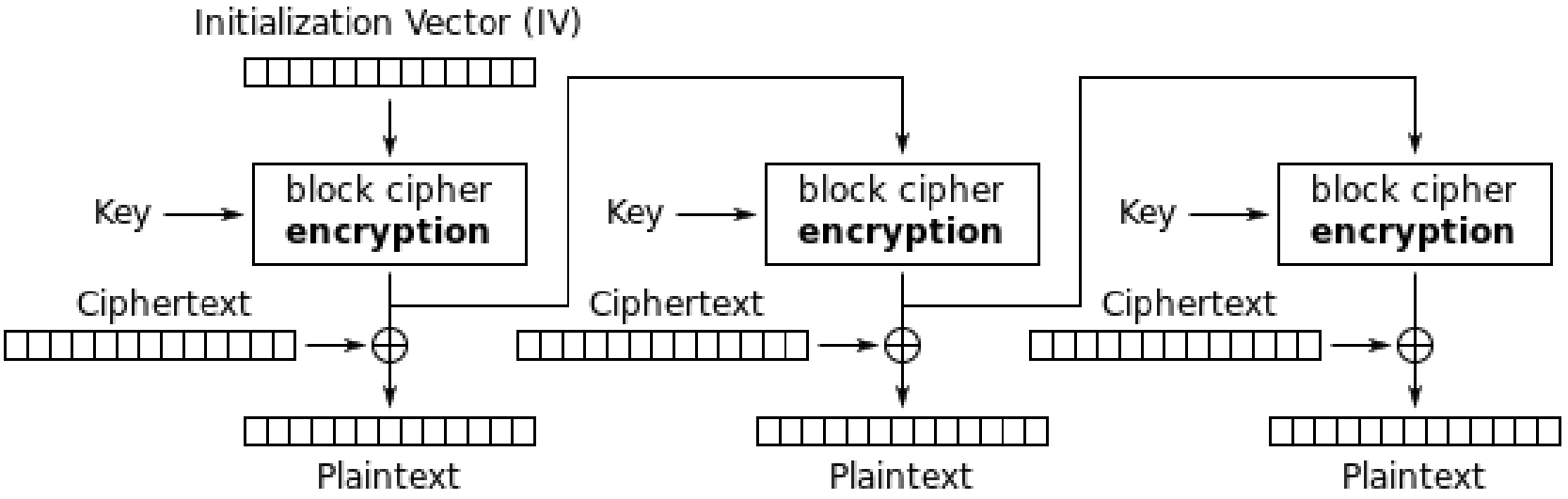
Output Feedback Mode (OFB):

- Ein weiterer Vorteil des Verfahrens ist, dass keine separate Entschlüsselungsfunktion notwendig ist, denn Ver- und Entschlüsselung sind identisch.

- Als Blockalgorithmus eignen sich auch Verfahren, die nicht bijektiv sind. Beispielsweise könnte als Blockalgorithmus auch eine sichere Hashfunktion wie SHA-256 verwendet werden. Die Blockgröße wäre dann die Ausgabegröße der Hashfunktion, bei SHA-256 also 256 Bits.

Betriebsmodi (DES)

Output Feedback Mode (OFB):



Output Feedback (OFB) mode decryption

Betriebsmodi (DES)

Output Feedback Mode (OFB):

- Wenn mehrere Nachrichten mit dem gleichen Schlüssel verschlüsselt werden, muss für jede Nachricht ein anderer Initialisierungsvektor verwendet werden, da ansonsten der gleiche Schlüsselstrom erzeugt wird.
- In diesem Fall ist das Verfahren gegen einen einfachen Angriff anfällig, bei dem zwei Chiffrate XOR-verknüpft werden.
- Durch die Verknüpfung löschen sich die zur Verschlüsselung verwendeten (gleichen) Schlüsselströme aus, und als Ergebnis bleibt das XOR der beiden Klartexte, aus dem die Klartexte leicht ermittelt werden können.
- Ist sogar der Klartext einer Nachricht bekannt, kann ein Klartextangriff durchgeführt werden: Durch XOR-Verknüpfung von Klar- und Geheimtext kann man den verwendeten Schlüsselstrom bestimmen und die entsprechenden Abschnitte der anderen Nachrichten entschlüsseln.

Betriebsmodi (DES)

Output Feedback Mode (OFB):

Beispiel: P1 = 01100100 (Dezimal 100), P2 = 01101110 (Dezimal 110).
Key = 01001011 (Dezimal 75).
C1 = 00101111, C2 = 00100101

Klartext (P1)	0	1	1	0	0	1	0	0
Klartext (P2)	0	1	1	0	1	1	1	0
Xor	0	0	0	0	1	0	1	0

C1	0	0	1	0	1	1	1	1
C2	0	0	1	0	0	1	0	1
Xor	0	0	0	0	1	0	1	0

Betriebsmodi (DES)

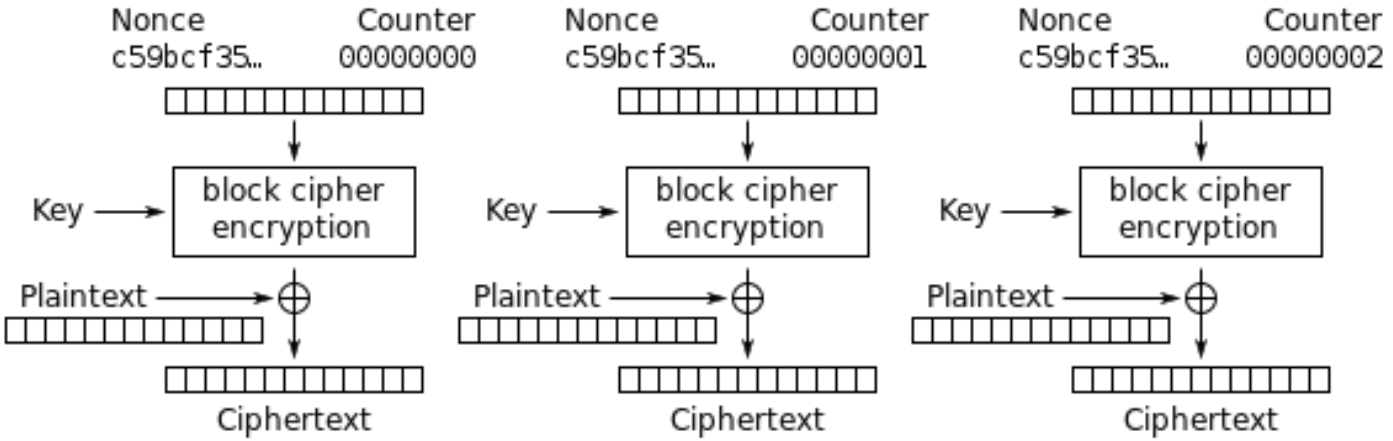
Counter Mode (CTR):

- Eine Vereinfachung des Output Feedback Modes.
- Bei dem anstelle eines Feedbacks eine Folge natürlicher Zahlen verschlüsselt wird. Damit ist es möglich, einen Block zu entschlüsseln, ohne vorher die anderen Blöcke entschlüsseln zu müssen.
- der Initialisierungsvektor hier besteht aus einer für jedes Chiffrat neu zu wählenden Zufallszahl (Nonce) verknüpft mit einem Zähler, der mit jedem weiteren Block hochgezählt wird. Die Verknüpfung kann z. B. durch Konkatenation (Anhängen), Addition oder XOR erfolgen.

Betriebsmodi (DES)

Counter Mode (CTR):

- Zur Verschlüsselung wird ein Initialisierungsvektor ctr_i mit dem Schlüssel K verschlüsselt und so ein Zwischenschlüssel produziert. Dieser wird im Anschluss mittels einer XOR-Operation mit dem Klartext kombiniert. Daraus entsteht der Geheimtext.



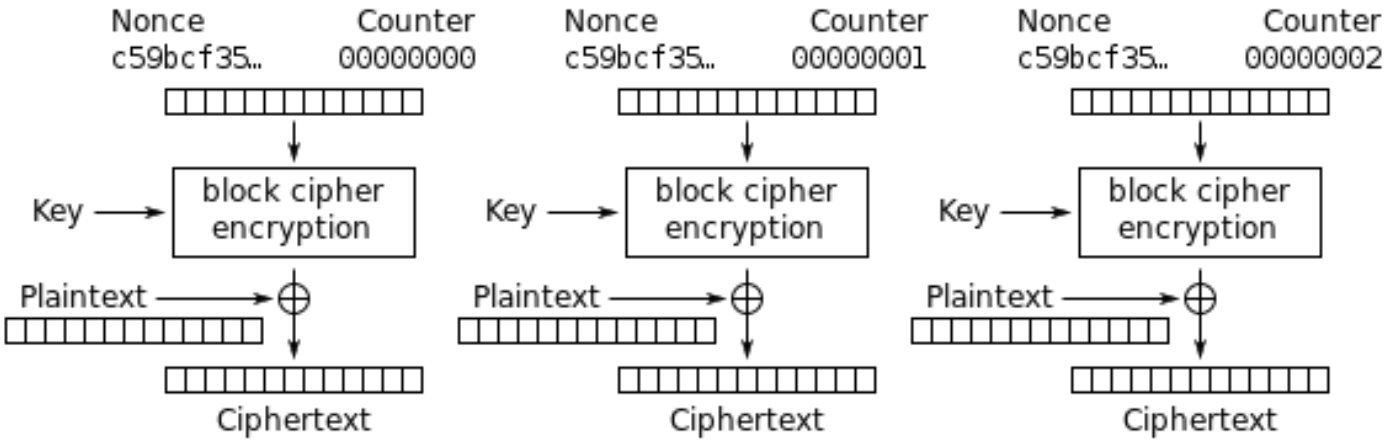
Counter (CTR) mode encryption

Betriebsmodi (DES)

Counter Mode (CTR):

- Zur Verschlüsselung wird ein Initialisierungsvektor ctr_i mit dem Schlüssel K verschlüsselt und so ein Zwischenschlüssel produziert. Dieser wird im Anschluss mittels einer XOR-Operation mit dem Klartext kombiniert. Daraus entsteht der Geheimtext.

$$C_i = P_i \oplus E_k(ctr_i)$$



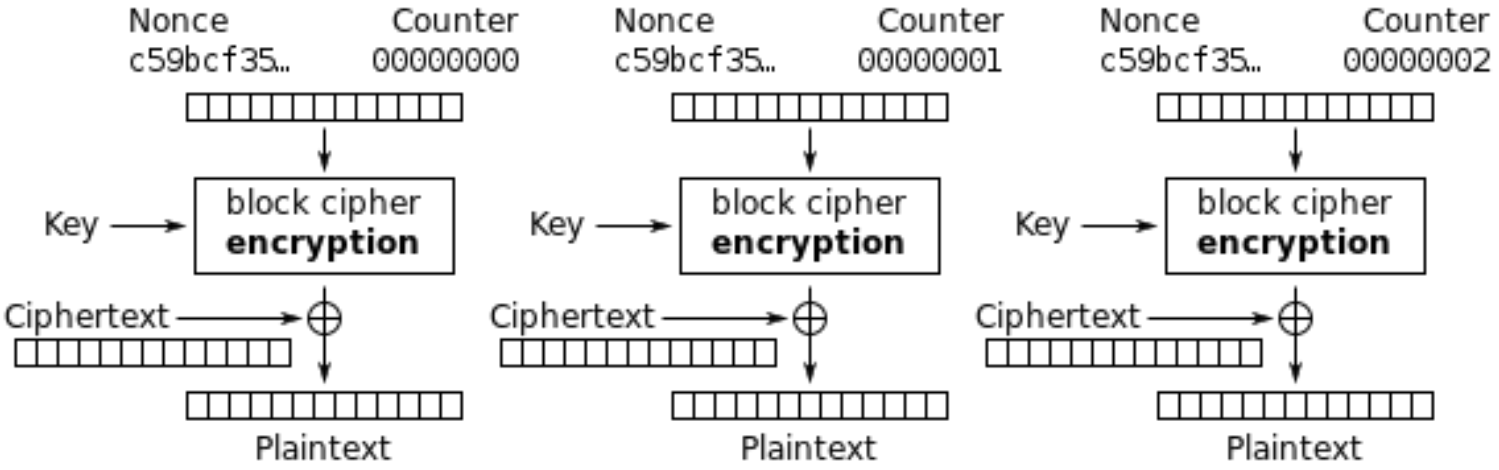
Counter (CTR) mode encryption

Betriebsmodi (DES)

Counter Mode (CTR):

- Zur Entschlüsselung wird wieder derselbe Zähler ctr_i mit dem Schlüssel K verschlüsselt, um den gleichen Zwischenschlüssel zu erhalten. Dieser wird nun mittels einer XOR-Operation auf den Geheimtext angewendet, so dass der Klartext wiedergewonnen wird.

$$P_i = C_i \oplus E_k(ctr_i)$$



Counter (CTR) mode decryption

DES (Schwächen)

Bei Schlüssellänge von nur 56 Bit, konnte DES bereits durch Brute-Force-Angriffe gebrochen werden, indem systematisch alle möglichen Schlüssel (2^{56} = ca. 72 Billiarden) getestet wurden.

- Die EFF baute 1998 eine etwa 250.000 Dollar teure Maschine mit dem Namen „Deep Crack“.
- Dieser Superrechner enthielt 1536 spezielle Krypto-Chips und konnte pro Sekunde etwa 88 Milliarden Schlüssel testen.
- Im Juli 1998 gelang es mit dieser Maschine, einen DES-Code in 56 Stunden zu knacken.
- Im 2006 wurde COPACOBANA von zwei Arbeitsgruppen an den Universitäten Bochum und Kiel gebaut.
- Im Gegensatz zu Deep Crack besteht eine COPACOBANA aus rekonfigurierbaren Hardware-Bausteinen, sog. FPGAs.
- COPACOBANA kann 65 Milliarden DES-Schlüssel pro Sekunde testen, woraus sich eine durchschnittliche Suchzeit von 6,4 Tagen für eine DES-Attacke ergibt.

Advanced Encryption Standard (AES)

- ist eine Blockchiffre, die als Nachfolger für DES im Oktober 2000 vom National Institute of Standards and Technology (NIST) als Standard bekanntgegeben wurde.
- Nach seinen Entwicklern Joan Daemen und Vincent Rijmen wird AES auch **Rijndael-**Algorithmus genannt.
- Es handelt sich um ein symmetrisches Verschlüsselungsverfahren, d. h. der Schlüssel zum Ver- und Entschlüsseln ist identisch.
- AES schränkt die Blocklänge auf 128 Bit und die Wahl der Schlüssellänge auf 128, 192 oder 256 Bit ein.
- Die Bezeichnungen der drei AES-Varianten AES-128, AES-192 und AES-256 beziehen sich jeweils auf die gewählte Schlüssellänge.
- Der Algorithmus ist frei verfügbar und darf ohne Lizenzgebühren eingesetzt sowie in Soft- und Hardware implementiert werden.
- AES-192 und AES-256 sind in den USA für staatliche Dokumente mit höchstem Geheimhaltungsgrad zugelassen.

Advanced Encryption Standard (AES)

Substitutions-Permutations-Netzwerk (SPN)

- Ein Substitutions-Permutations-Netzwerk besteht aus einer Anzahl von Runden gleichen Aufbaus.
- In jeder Runde wird zuerst ein Rundenschlüssel auf die Eingabe addiert. Dann wird das Ergebnis in mehrere Blöcke aufgeteilt, und jeder Block mittels der Substitutionsbox (S-Box) durch einen anderen Block ersetzt.
- Diese Blöcke werden wiederum durch eine Permutationsbox (P-Box) vermischt.

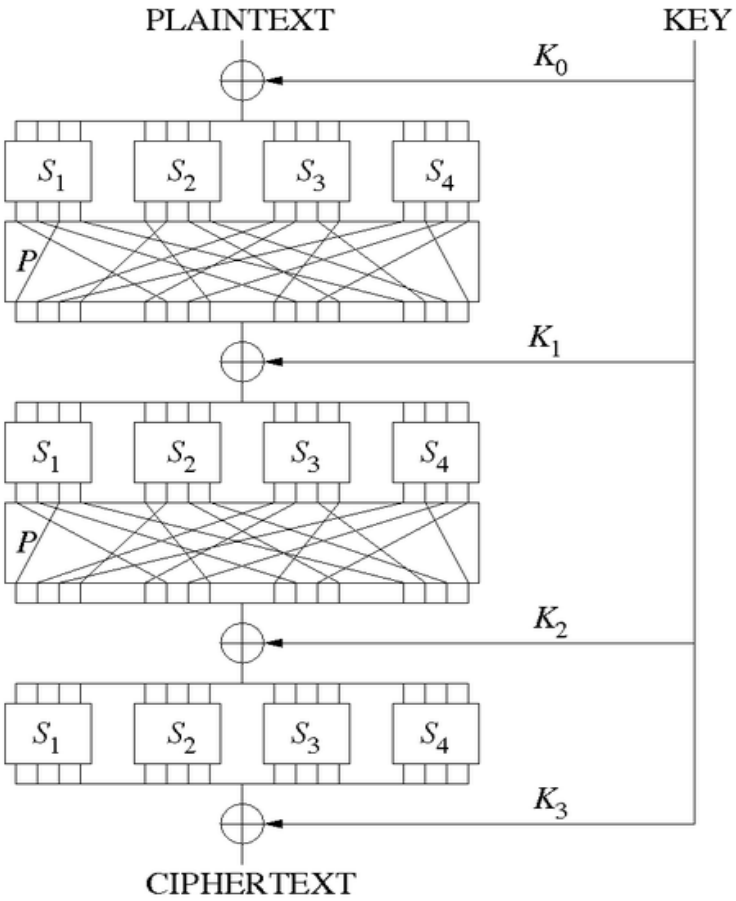
Advanced Encryption Standard (AES)

Substitutions-Permutations-Netzwerk (SPN)

- Bei der letzten Runde kann die P-Box weggelassen werden, weil sie von jedem aus dem Ergebnis trivial herausgerechnet werden kann.
- Wenn sich ein Bit des Klartextes oder Schlüssels ändert, ändern sich mehrere Bit der Ausgabe der S-Box, die dann durch die P-Box auf verschiedene S-Boxen der nächsten Runde verteilt werden.
- Im Gegensatz zu Feistelchiffren sind Substitutions-Permutations-Netzwerke im Allgemeinen nicht durch einfaches Umordnen des Schlüssels umkehrbar, da die S-Box nicht selbstinvers ist.

Advanced Encryption Standard (AES)

Substitutions-Permutations-Netzwerk (SPN)



Advanced Encryption Standard (AES)

- Jeder Block wird zunächst in eine zweidimensionale Tabelle mit vier Zeilen geschrieben, deren Zellen ein Byte groß sind.

- Die Anzahl der Spalten variiert somit je nach Blockgröße von 4 (128 Bits) bis 8 (256 Bits).

- Jeder Block wird nun nacheinander bestimmten Transformationen unterzogen. Aber anstatt jeden Block einmal mit dem Schlüssel zu verschlüsseln, wendet Rijndael verschiedene Teile des erweiterten Originalschlüssels nacheinander auf den Klartext-Block an.

Einwegfunktion

- Ist eine mathematische Funktion, die komplexitätstheoretisch „leicht“ berechenbar, aber „schwer“ umzukehren ist.
- Im allgemein werden auch Funktionen so bezeichnet, zu denen bisher keine in angemessener Zeit praktisch ausführbare Umkehrung bekannt ist.

Eine Einwegfunktion f muss folgende Eigenschaften aufweisen:

- Die Berechnung des Funktionswerts $y = f(x)$ zu gegebenem x ist „einfach“, d. h., es existiert ein Algorithmus, der ihn in Polynomialzeit berechnet.
- Die Berechnung der Umkehrung der Funktion, d. h. eines Urbildes x zu einem gegebenen y so dass $f(x) = y$, ist allerdings „schwer“, d. h., es existiert kein Algorithmus F, der in Polynomialzeit läuft und mit nicht vernachlässigbarer Wahrscheinlichkeit zu dem eingegebenen Bild ein Urbild ausgibt.

Trapdoor-Einwegfunktion

- Eine Variante der Einwegfunktionen, diese lassen sich nur dann effizient umkehren, wenn man eine gewisse Zusatzinformation besitzt.

- Falltürfunktionen werden in asymmetrischen Verschlüsselungsverfahren verwendet.

- Bekannte Einwegfunktionen:
  - die kryptographischen Hashfunktionen wie SHA-2
  - die Multiplikation zweier Primzahlen ist einfach, während die Umkehrung, die Primfaktorzerlegung, schwer ist

Asymmetrische Verfahren

- Asymmetrisches Kryptosystem ist ein Oberbegriff für Public-Key-Verschlüsselungsverfahren, Public-Key-Authentifizierung und digitale Signaturen.

- Jeder Benutzer erzeugt sein eigenes Schlüsselpaar, das aus einem geheimen Teil (privater Schlüssel) und einem nicht geheimen Teil (öffentlicher Schlüssel) besteht.

- Der öffentliche Schlüssel ermöglicht es jedem, Daten für den Besitzer des privaten Schlüssels zu verschlüsseln, dessen digitale Signaturen zu prüfen oder ihn zu authentifizieren.

Asymmetrische Verfahren

- Der private Schlüssel ermöglicht es seinem Besitzer, mit dem öffentlichen Schlüssel verschlüsselte Daten zu entschlüsseln, digitale Signaturen zu erzeugen oder sich zu authentisieren.

- Ein **Public-Key-Verschlüsselungsverfahren** ist ein Verfahren, um mit einem öffentlichen Schlüssel einen Klartext in einen Geheimtext umzuwandeln, aus dem der Klartext mit einem privaten Schlüssel wieder gewonnen werden kann.

- Der private Schlüssel muss geheim gehalten werden und es muss praktisch unmöglich sein, ihn aus dem öffentlichen Schlüssel zu berechnen.

Asymmetrische Verfahren

- Die theoretische Grundlage für asymmetrische Kryptosysteme sind Falltürfunktionen.
- Diese sind Funktionen, die leicht zu berechnen, aber ohne ein Geheimnis (die „Falltür“) praktisch unmöglich zu invertieren sind. Der öffentliche Schlüssel ist dann eine Beschreibung der Funktion, der private Schlüssel ist die Falltür.
- Eine Voraussetzung ist, dass der private Schlüssel aus dem öffentlichen nicht berechnet werden kann.
- Der öffentliche Schlüssel muss dem Kommunikationspartner bekannt sein
- Der entscheidende Vorteil von asymmetrischen Verfahren ist, dass sie das Schlüsselverteilungsproblem vermindern.

RSA Verfahren

- Ein asymmetrisches kryptografisches Verfahren, das sowohl zum Verschlüsseln als auch zum digitalen Signieren verwendet werden kann.

- Es verwendet ein Schlüsselpaar, bestehend aus einem privaten Schlüssel, der zum Entschlüsseln oder Signieren von Daten verwendet wird, und einem öffentlichen Schlüssel, mit dem man verschlüsselt oder Signaturen prüft.

- Der private Schlüssel wird geheim gehalten und kann nicht aus dem öffentlichen Schlüssel berechnet werden.

RSA Verfahren

- Der öffentliche Schlüssel (public key) ist ein Zahlenpaar (e, N).
- der private Schlüssel (private key) ist ebenfalls ein Zahlenpaar (d, N).
- N ist bei beiden Schlüsseln gleich. Man nennt N den RSA-Modul, e den Verschlüsselungsexponenten und d den Entschlüsselungsexponenten.

RSA Verfahren

Diese Zahlen werden durch das folgende Verfahren erzeugt:

- Wähle zufällig und stochastisch unabhängig zwei Primzahlen $p \neq q$.
- Berechne den RSA-Modul $N = p.q$.
- Berechne die Eulersche Phi-Funktion von N: $\varphi(N) = (p-1).(q-1)$
- Wähle eine zu $\varphi(N)$ teilerfremde Zahl e, für die gilt $1 < \mathrm{e} < \varphi(N)$
- Berechne den Entschlüsselungsexponenten d als multiplikativ Inverses von e bezüglich des Moduls $\varphi(N)$

$$\mathrm{e.d} + \mathrm{k}.\varphi(N) = 1 = ggT(e, \varphi(N))$$

RSA Verfahren

Beispiel:

1. Wir wählen $p = 11$ und $q = 13$ für die beiden Primzahlen.
2. Der RSA-Modul ist $N = p.q = 143$.
3. Die eulersche φ-Funktion hat damit den Wert $\varphi(N) = \varphi(143) = (p-1).(q-1) = 120$.
4. Die Zahl e muss zu 120. Wir wählen $e = 23$. Damit bilden $e = 23$ und $N = 143$ den öffentlichen Schlüssel.
5. $23.d + k.120 = 1 = ggT(23, 120)$. Mit dem erweiterten euklidischen Algorithmus berechnet man nun die Faktoren $d = 47$ und $k = -9$. $d = 47$ und $N = 143$ bilden den privaten Schlüssel.

RSA Verfahren

Verschlüsseln von Nachrichten:

$$c \equiv m^e \ (mod\ N)$$

die Zahl 7 verschlüsselt werden: $2 \equiv 7^{23}\ (mod\ 143)$

$c = 2$

Entschlüsseln von Nachrichten:

$$m \equiv c^d \ (mod\ N)$$

Für $c = 2$ wird berechnet $7 \equiv 2^{47}\ (mod\ 143)$

$m = 7$

RSA Verfahren

- In der Praxis sucht man zwei gigantisch große Primzahlen.
- Die multipliziert man nun und bekommt eine noch viel gigantisch größere Zahl
- wenn man 4096-Bit große Primzahlen benutzt (also bis ca. $1{,}0444 \cdot 10^{1233}$) erhält man nach der Multiplikation eine bis ca. $1{,}0908 \cdot 10^{2466}$ große Zahl.
- Das errechnen von solch großen Zahlen aus dem *public-key* (also das erstellen einer Primfaktorzerlegung) ist nach heutigen Rechenleistungen nicht möglich.