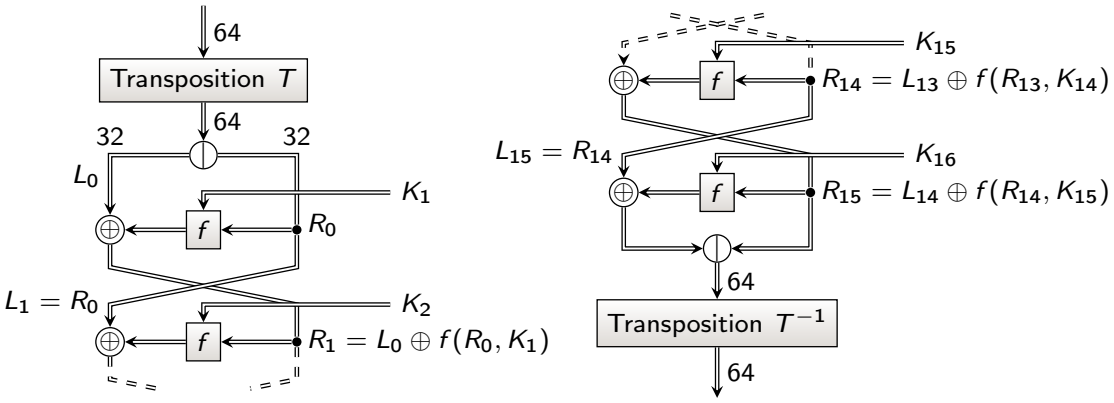
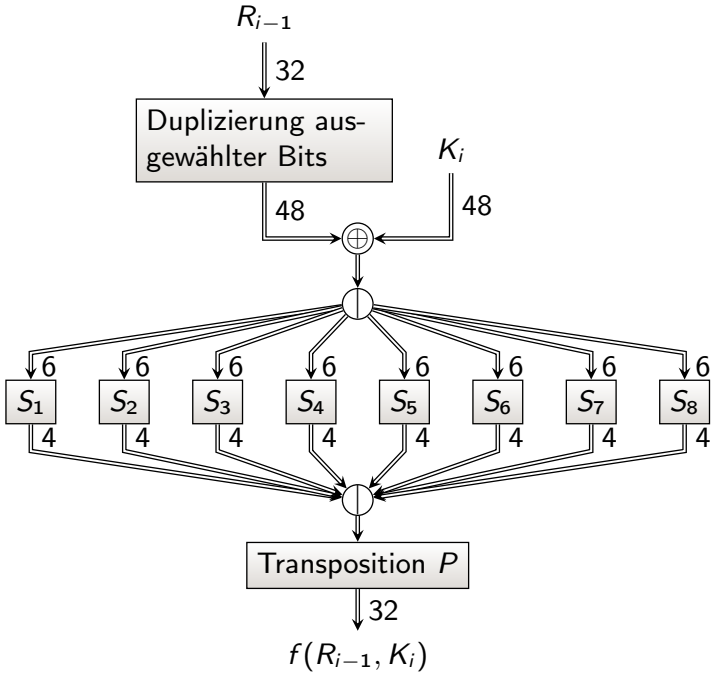
9 Rechnergestütze Blockchiffren

9.1 Data Encryption Standard (DES)

- 1977 von National Bureau of Standards (heute National Institute of Standards and Technology) genormt
- arbeitet auf 64 bit-Blöcken
- Schlüssel 56 bit + 8 bit Fehlerschutz



Aufbau der Funktion f:

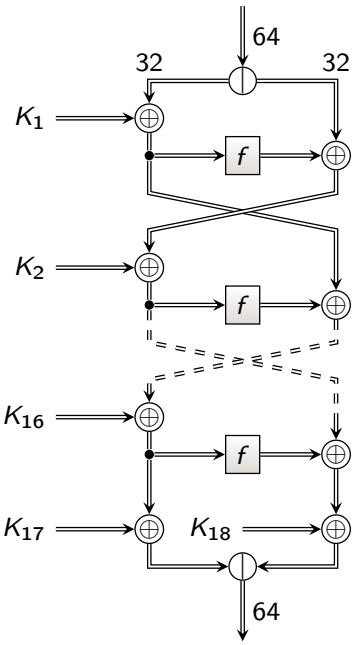


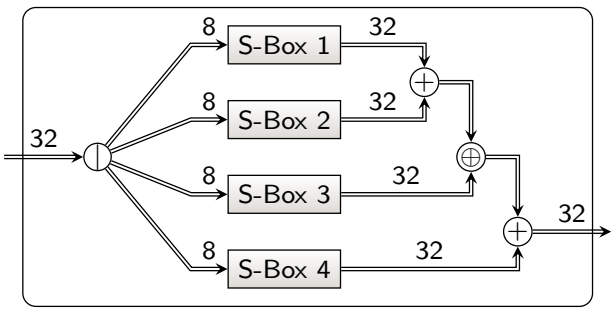
- K_i 48 (rundenabhängig) aus dem 56 bit Schlüssel extrahierte Bits
- S-Boxen $S_1, \ldots, S_8$ über Tabellen definierte Abbildungen

- effizient in Hardware realisierbar
- Entschlüsselung = Verschlüsselung mit Teilschlüsseln K_i in umgekehrter Reihenfolge
- Entwurf der S-Boxen zunächst nicht (öffentlich) begründet
- anhaltende Unterstellung, dass der Entwurf "Geheimtüren" enthält
- 56 bit Schlüssel heute zu kurz (angepasste Hardware erlaubt Brute-Force-Angriff innerhalb von Stunden)
- Erweiterung zu Triple-DES (3DES): nacheinander mit drei Schlüsseln verschlüsseln; immer noch relativ sicher (falls nicht doch Geheimtüren)

9.2 Blowfish

- auf den ersten Blick hohe Ähnlichkeit zu DES



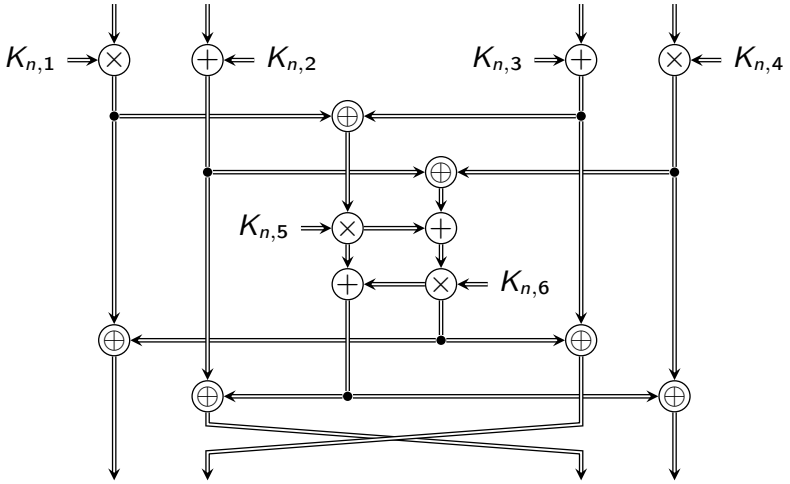


Funktion f

- S-Boxen über Tabellen mit je $2^8 = 256$ Einträgen von 32 bit Länge
- Besonderheit an Blowfisch: Ableitung der Teilschlüssel $K_1, \ldots, K_{18}$ und der S-Boxen aus Schlüssel von bis zu 448 bit
  1. initialisiere Teilschlüssel und S-Boxen-Einträge mit festen Werten (Hexadezimalstellen der Zahl π)
  2. XOR-verknüpfe alle Teilschlüssel mit (periodisch wiederholtem) Schlüssel
  3. verschlüssele 0-Block
  4. ersetze K_1 und K_2 mit Ergebnis aus 3
  5. verschlüssele Ergebnis aus 3 (mit neuen K_1 und K_2)
  6. ersetze K_3 und K_4 mit Ergebnis aus 5
  7. fahre entsprechend fort und ersetze alle weiteren Teilschlüssel und die S-Boxen-Einträge
- (einmaliger) Aufwand zur Ableitung der Teilschlüssel und S-Boxen relativ hoch, aber eigentliche Verschlüsselung auf 32 bit-Rechner effizient implementierbar
- keine kryptographischen Schwächen bekannt
- der Entwickler (B. Schneier) empfiehlt den Nachfolger Twofish

9.3 International Data Encryption Algorithm (IDEA)

- 64 bit-Blöcke, Aufteilung in 4×16 bit
- Verschlüsselung in 8 Runden



- Multiplikationen erfolgen modulo $2^{16} + 1$ (prim!), Teilblock Null entspricht 2^{16}
- effizient auf 16 bit-Rechnern zu implementieren

- Schlüssellänge 128 bit
- Teilschlüssel: Aufteilung in 8×16 bit, Verschiebung um 25 bit, wiederholen bis genügend Teilschlüssel
- keine Schwächen (bis auf die Existenz schwacher Schlüssel) bekannt
- war bis zum 16. Mai 2011 patentgeschützt

Exkurs: Galois-Felder

Ring:

- Menge M
- Operation $+$ und $\cdot$ (Addition und Multiplikation)
- 0-Element: $a + 0 = 0 + a = a$ für alle $a \in M$
- additives Inverses $a + (-a) = -a + a = 0$ existiert für alle $a \in M$
- 1-Element: $a \cdot 1 = 1 \cdot a = a$ für alle $a \in M$
- multiplikatives Inverses $a \cdot a^{-1} = a^{-1} \cdot a = 1$ existiert nur für bestimmte $a \in M$ ("Einheiten")
- Beispiel: Restklassenring

Körper (engl. Field):

- Ringe, bei denen jedes Element außer 0 ein multiplikatives Inverses besitzt
- Beispiele: Restklassenring $\mathbb{Z}_p$, p prim; rationale Zahlen

- sei $\mathbb{K}$ ein Körper, $\mathbb{K}[\alpha]$ die Menge der Polynome in α mit Koeffizienten aus $\mathbb{K}$
- für geeignetes Polynom $g(\alpha) \in \mathbb{K}[\alpha]$ führt Rechnen modulo $g(\alpha)$ zu einem Körper
- Beispiel: $\mathbb{K} = \mathbb{R}, g(\alpha) = \alpha^2 + 1$
- Anforderung an $g(\alpha)$: nicht (mit Koeffizienten aus $\mathbb{K}$) faktorisierbar ("irreduzibel")

Galois-Feld (eigentlich "Field" oder "Körper"):
GF(p^n), p prim $\Leftrightarrow$ Körper $\mathbb{Z}_p[\alpha], g(\alpha)$ vom Grad n irreduzibel

Beispiel: $p = 2, g(\alpha) = \alpha^4 + \alpha + 1$ (4 Bit)

- $(\alpha^3 + 1) + (\alpha^2 + 1) = \alpha^3 + \alpha^2 + 2 \equiv \alpha^3 + \alpha^2$
- $(\alpha^3 + 1) \cdot (\alpha^2 + 1) = \alpha^5 + \alpha^3 + \alpha^2 + 1 \equiv \alpha^3 + \alpha + 1$
- $(\alpha^3 + 1)^{-1} = ? \rightarrow$ erweiterter euklidscher Algorithmus

9.5 Advanced Encryption Standard (AES)

- 2001 vom National Institute of Standards and Technology als DES-Nachfolger genormt
- von der NSA für Verschlüsslung von *Top Secret*-Material zugelassen
- aktuelle Prozessoren stellen spezielle AES-Instruktionen bereit
- arbeitet auf 128 bit-Blöcken
- Schlüssel wahlweise 128 bit, 192 bit oder 256 bit
- je nach Schlüssellänge Verschlüsselung in 10, 12 oder 14 Runden
- jede Runde umfasst die Funktionen
  - `SubBytes`
  - `ShiftRows`
  - `MixColumns` (außer letzte Runde)
  - `AddRoundKey` (mit rundenabhängigem Teilschlüssel)
- vor erster Runde zusätzliches `AddRoundKey` (mit Teilschlüssel 0)

- in AES werden 8 bit-Blöcke als Polynome 7. Grades mit Koeffizienten in $\mathbb{Z}_2$ interpretiert
- Beispiel: 00000011 $\equiv \alpha + 1$
- Rechnungen erfolgen modulo $\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1$ (und modulo 2 für die Koeffizenten) $\Rightarrow$ Galois-Feld GF(2^8)
- Beispiel: 00100100 + 11010110 $\equiv (\alpha^5 + \alpha^2) + (\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2 + \alpha) \equiv \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha \equiv$ 11110010 (XOR-Operation)
- Beispiel: 00100100 $\cdot$ 11010110 $\equiv (\alpha^5 + \alpha^2) \cdot (\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2 + \alpha) \equiv \alpha^{12} + \alpha^{11} + \alpha^8 + \alpha^7 + \alpha^4 + \alpha^3 \equiv \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 \equiv$ 11110000
- druch Wahl des Polynoms $\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1$ hat jedes Element (außer 0) ein multiplikatives Inverses, bestimmbar über erweiterten euklidschen Algorithmus

`SubBytes`

- Substitution auf 8 bit-Blöcken
- invertiere x in GF(2^8) zu x^{-1} (setze $x^{-1} = 0$ für $x = 0$)
- danach lineare Substitution (x^{-1} als Vektor in $\mathbb{Z}_2$ interpretiert)

$$x^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \mod 2$$

- gesamte Operation kann bei nur $2^8 = 256$ Einträgen leicht tabelliert werden

`ShiftRows`

- Daten in 4 Zeilen zu je 32 bit teilen; zweiten Zeile um 8, dritte um 16, vierte um 24 bit zyklisch nach links verschieben

`MixColumns`

- bilde aus der Darstellung als 4 Zeilen zu je 32 bit 4 Spalten von 4×8 bit als Vektor s_j (Spalte $j \in \{0, \ldots, 3\}$) in $\mathrm{GF}(2^8)^4$
- ersetze jeweils gemäß

$$s_j \leftarrow \begin{pmatrix} \alpha & \alpha+1 & 1 & 1 \\ 1 & \alpha & \alpha+1 & 1 \\ 1 & 1 & \alpha & \alpha+1 \\ \alpha+1 & 1 & 1 & \alpha \end{pmatrix} s_j$$

`AddRoundKey`

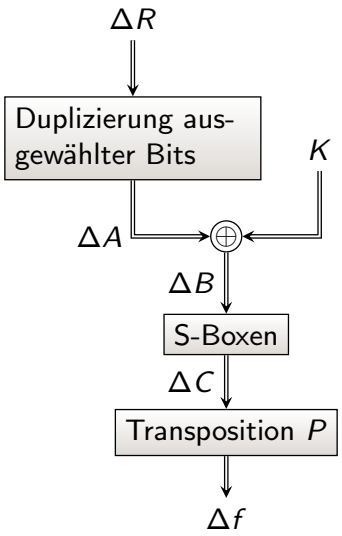
- XOR-Verknüpfung mit sukzessiven 128 bit-Blöcken aus mit `KeyExpansion` erweiterem Schlüssel

`KeyExpansion`

- die 128, 192 oder 256 bit des Schlüssels werden zu den insgesamt benötigten 1408, 1664 oder 1920 bit erweitert
- der erste Teil des erweiterten Schlüssels ist der Originalschlüssel, die zusätzlichen Bits werden 32 bit-blockweise unter Rückgriff auf einfache Transformationen, u.a. `SubBytes` gebildet

9.6 Differentielle Kryptanalyse

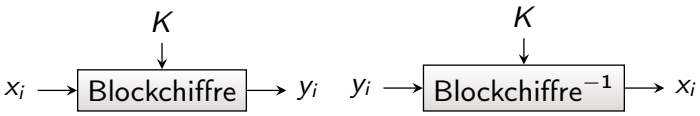
- es sei zunächst eine einzelne Rundenfunktion von DES betrachtet
- wir gehen von einem Chosen-Plaintext-Angriff mit zwei verschiedenen Klartexten aus
- die Differenz der beiden rechten Blöcke am Eingang der Runde sei ΔR
- Differenzen $\Delta A = \Delta B$ leicht zu bestimmen
- bei bekannten ΔB bestimmte Differenzen ΔC wahrscheinlicher als andere
- damit auch statistische Aussage über die Differenz der beiden Blöcke am Ausgang der Rundenfunktion Δf möglich
- offensichtlich: $\Delta R = 0 \Rightarrow \Delta f = 0$ mit Wahrscheinlichkeit 1
- weniger offensichtlich: $\Delta R = 0x60000000 \Rightarrow \Delta f = 0x00808200$ mit Wahrscheinlichkeit $\frac{14}{64} \gg \frac{1}{2^{32}}$



- durch Kaskadierung durch mehrere Runden hindurch wird es möglich, die Eingangsdifferenzen der letzten Runde mit gewisser Wahrscheinlichkeit vorherzusagen
- daraus kann der letzte Rundenschlüssel, und damit 48 von 56 bit des Schlüssels vorhergesagt werden
- nach vollen 16 DES-Runden nicht mehr praktikabel (Wahrscheinlichkeiten ausreichend nivelliert)
- S-Boxen von DES sind speziell gegen differentielle Kryptanalyse ausgelegt, obwohl diese erst seit 1990 öffentlich bekannt ist
- da alle gängigen Verfahren gegen differentielle Kryptanalyse resistent sind, ist sie eher für den Entwurf von Verfahren, denn für erfolgreiche Angriffe wichtig
- eine ähliche Idee, über Wahrscheinlichkeiten zu operieren, ist die lineare Kryptanalyse, bei der die Verschlüsselungsfunktion über eine lineare Funktion approximiert wird (die nur mit einer gewissen Wahrscheinlichkeit das richtige Ergebnis liefert)

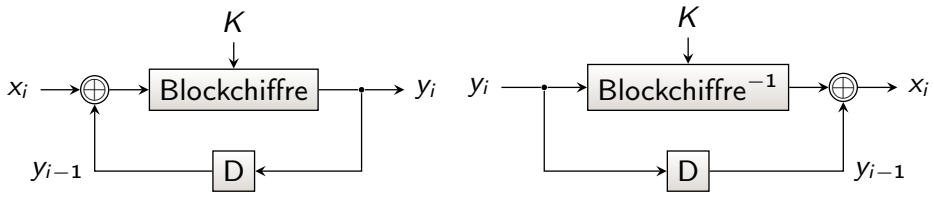
9.7 Betriebsmodi

Electronic Codebook (ECB)



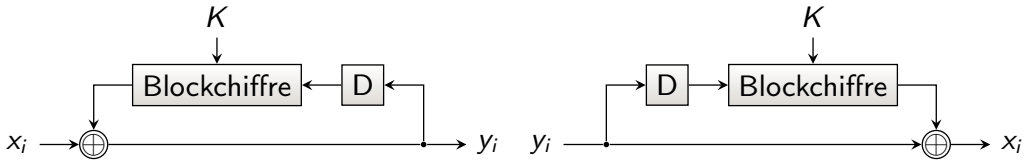
- jeder Block des Klartextes wird unabhängig verschlüsselt
- polygraphische, monoalphabeitsche Subsitution
- Ver- und Entschlüsselung parallelisierbar
- Ver- und Entschlüsselung von Textteilen möglich
- Muster bleiben erhalten
- Einbruch über häufige Geheimtextblöcke möglich
- Block-Replay möglich

Cipher Block Chaining (CBC)



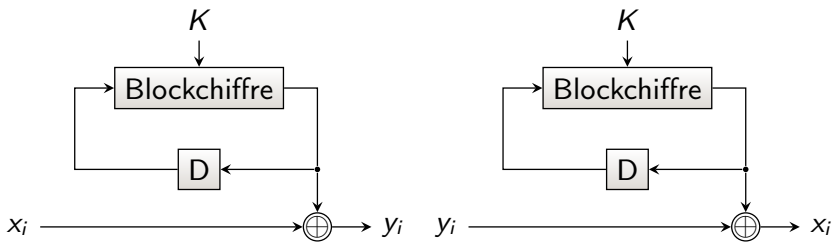
- jeder Block des Klartextes wird vor der eigentlichen Verschlüsselung mit dem vorhergehenden Geheimtextblock XOR-verknüpft
- erster Block wird mit (ebenfalls zu übertragendem) beliebigem Initialisierungsvektor (IV) XOR-verknüpft
- selbst derselbe Klartext mit demselben Schlüssel ergibt unterschiedliche Geheimtexte
- keinerlei Replay möglich, wenn gleiche IV verboten (z.B. Zeitstempel)
- Fehler in einem Geheimtextblock wirkt sich (nur) auf zwei entschlüsselte Klartextblöcke aus

Cipher Feedback (CFB)



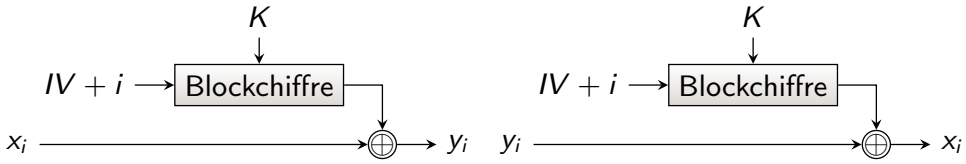
- der letzte Block des Geheimtextes wird mit dem Blockchiffrierverfahren verschlüsselt, Ergebnis wird mit dem Klartext XOR-verknüpft, um Geheimtext zu erhalten
- erster Block mit IV
- IV muss immer gewechselt werden
- Text kann in kleineren Blöcken verarbeitet werden, als Blockgröße des Chiffrierverfahrens (Bsp: Text wird Byte für Byte verarbeitet, die jeweils letzten 16 Byte werden mit AES verschlüsselt, vom Ergebnis wird nur 1 Byte benutzt und mit dem nächsten Klartextbyte XOR-verknüpft, um Geheimtextbyte zu erzeugen)
- Rechenaufwand vervielfacht sich
- Fehlerfortpflanzung begrenzt (im Bsp. auf 16 weitere Byte)
- automatische Resynchronistation nach Verschiebung um (im Bsp.) 1 Byte

Output Feedback (OFB)



- Iteration des Blockchiffrierverfahrens (ausgehend von immer anderem IV) erzeugt quasi-nichtperiodischen Schlüssel zur XOR-Verknüpfung mit dem Klartext
- keinerlei Fehlerfortpflanzung
- Synchronisationsfehler katastrophal

Counter (CTR)



- Anwendung des Blockchiffrierverfahrens auf Zähler (beginnend bei immer anderem IV) erzeugt quasi-nichtperiodischen Schlüssel zur XOR-Verknüpfung mit dem Klartext
- Ver- und Entschlüsselung parallelisierbar
- Ver- und Entschlüsselung von Textteilen möglich
- keinerlei Fehlerfortpflanzung
- Synchronisationsfehler katastrophal
- Erweiterung zum Galois/Counter Mode, bei dem die Ausgabe um ein Authentication Tag ergänzt wird, mit dem sich Manipulationen erkennen lassen