entwickler.press
shortcuts

Verschlüsselung im NSA-Zeitalter

Kryptografiestandards und Protokolle

Carsten Eilers

Carsten Eilers

Verschlüsselung im NSA-Zeitalter

Kryptografiestandards und Protokolle

ISBN: 978-3-86802-508-8

Ein Imprint der Software & Support Media GmbH

1 Sicherheit von symmetrischen Verfahren

Seit der Veröffentlichung der von Edward Snowden geleakten NSA-Daten ist die Verunsicherung groß: Wo hat die NSA überall die Finger im Spiel? Was ist noch sicher? Welchen Protokollen kann man noch vertrauen? Ich werde versuchen, diese Fragen in einem kleinen shortcut zu beantworten. Los geht es mit symmetrischer Verschlüsselung, bei der zum Ver- und Entschlüsseln der gleiche Schlüssel verwendet wird (**Abb. 1.1**). Ein symmetrisches Verschlüsselungssystem können Sie sich als undurchsichtigen Kasten mit einem Schloss vorstellen, für das es zwei gleiche Schlüssel gibt. Als erstes symmetrisches System stelle ich ein Verfahren vor, dass Sie keinesfalls mehr verwenden sollten: DES.

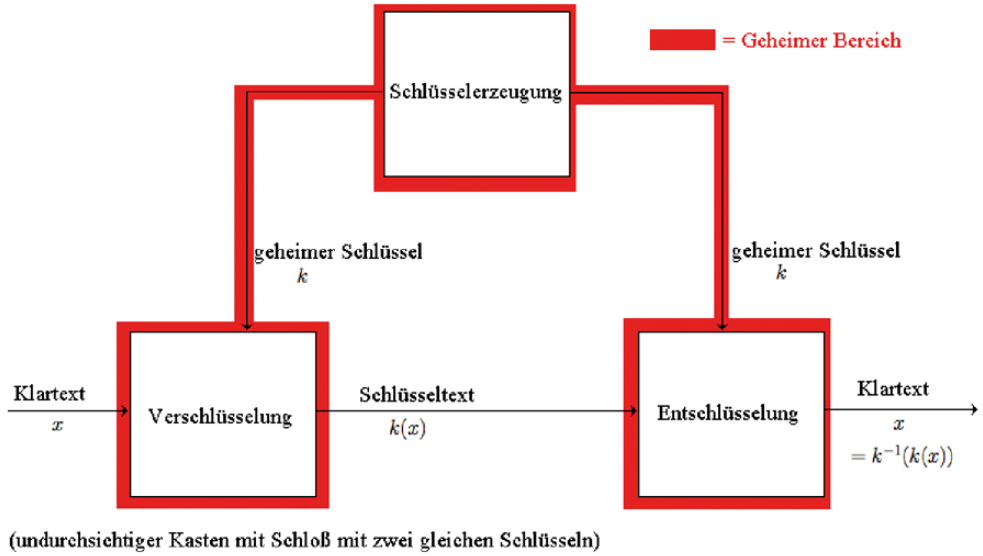

Abb. 1.1: Symmetrische Verschlüsselung

DES, der Data Encryption Standard

DES ist alt, aus IT-Sicht sogar uralt: Der Algorithmus des DES-Verfahrens wurde erstmals am 15. Januar 1977 in „Specification for the Data Encryption Standard; Federal Information

Processing Standards Publication 46“ (FIPS PUB 46) veröffentlicht. Die erste Version ist leider nicht mehr online verfügbar, sondern nur aktualisierte Fassungen: FIPS PUB 46-2 [1] und FIPS PUB 46-3 [3].

Das Alter ist aber nicht das Problem von DES. Das in der nächsten Folge vorgestellte asymmetrische RSA-Verfahren ist nur ein Jahr jünger und, einen ausreichend langen Schlüssel vorausgesetzt, immer noch sicher. Das Problem von DES ist der Schlüssel, genauer: seine Länge (oder aus heutiger Sicht: Kürze), die nicht geändert werden kann.

DES verwendet einen 56 Bit langen Schlüssel und verschlüsselt Blöcke von 64 Bit Länge. Der Schlüssel wird um acht Paritätsbits auf 64 Bit erweitert, die Paritätsbits werden für den Algorithmus jedoch nicht verwendet. Der DES-Algorithmus (**Abb. 1.2**) besteht aus:

- einer kryptographisch bedeutungslosen Eingangspermutation *IP* (Initial Permutation), die unter anderem den Klartextblock in die beiden 32-Bit-Blöcke L_0 und R_0 zerlegt
- 16 Iterationsrunden, in denen die eigentliche Verschlüsselung erfolgt
- einer zur Eingangspermutation inversen Ausgangspermutation IP^{-1}, vor deren Ausführung die Ergebnisse der 16. Iterationsrunde, L_{16} und R_{16}, nochmals vertauscht werden

Aus dem Schlüssel werden die 16 Teilschlüssel K_1 bis K_{16} erzeugt, einer für jede Iterationsrunde.

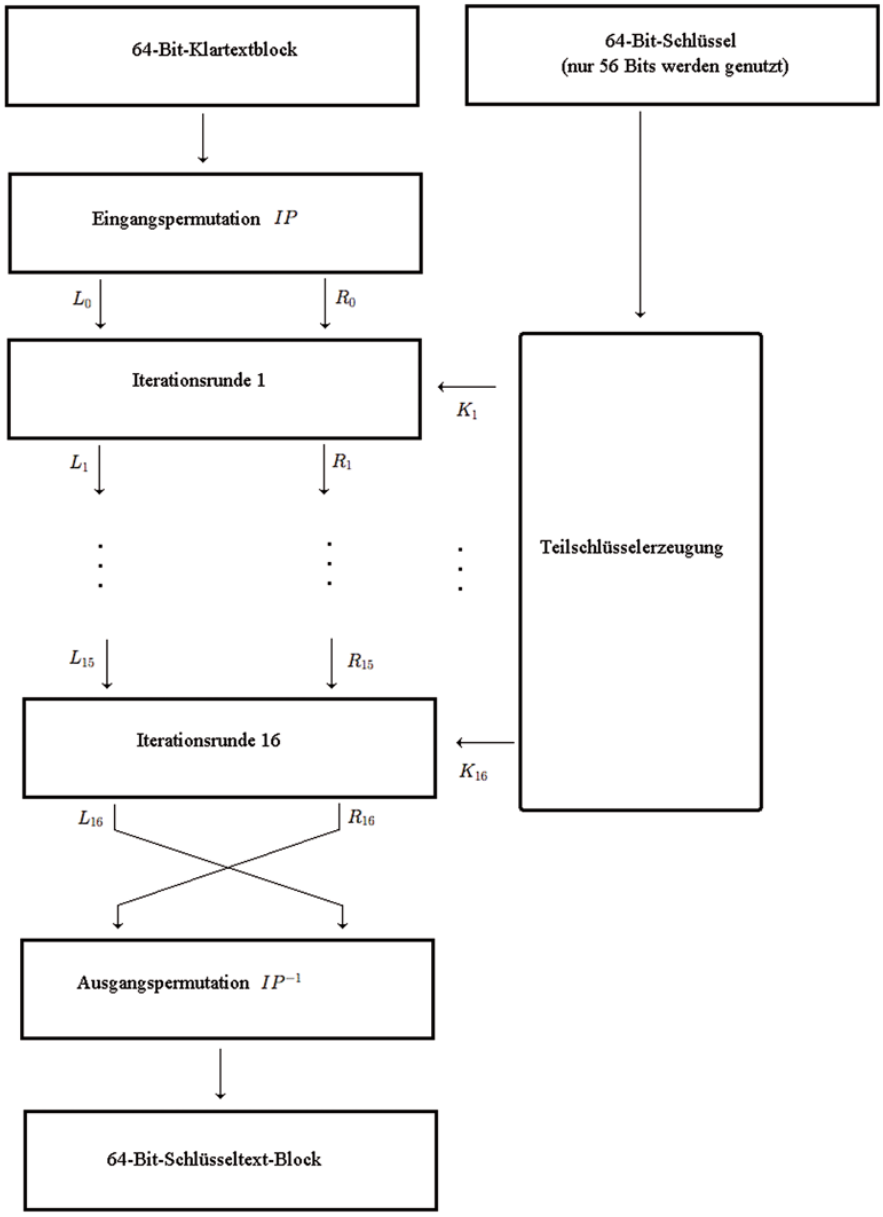

Abb. 1.2: Der DES-Algorithmus im Überblick

Die Iterationsrunden

Die Iterationsrunden entsprechen den Runden eines so genannten *Feistel-Netzwerks*. Die für die Entschlüsselung notwendige Vertauschung erfolgt nach der 16. Iterationsrunde, der DES-Algorithmus kann also unverändert sowohl zur Ver- als auch Entschlüsselung genutzt werden. Die zum Entschlüsseln

notwendige Umkehrung der Reihenfolge der Iterationen erfolgt durch die Umkehrung der Reihenfolge der Teilschlüssel K_i.

Jede Iterationsrunde *i* erhält als Eingabe die beiden Blöcke L_{i-1} und R_{i-1}. Die Verschlüsselungsfunktion *f* verwendet den geheimen Schlüssel K_i, um aus dem gegebenen Block R_{i-1} einen (Geheimtext-)block $f(R_{i-1},K_i)$ zu erzeugen. Die eigentliche Verschlüsselung erfolgt dann, indem die beiden Halbblöcke vertauscht und L_{i-1} mit $f(R_{i-1},K_i)$ XOR-verknüpft wird:

$L_i := R_{i-1}$

$R_i := L_{i-1} \oplus f(R_{i-1},K_i)$

Grafisch lässt sich das Ganze wie in der linken Hälfte von **Abbildung 1.3** darstellen. Für die Entschlüsselung muss dieser Prozess umgekehrt werden. Das Ergebnis von Runde *i* ist wie oben zu sehen:

$L_i := R_{i-1}$

$R_i := L_{i-1} \oplus f(R_{i-1},K_i)$

Zum Entschlüsseln werden L_i und R_i getauscht, außerdem wird der Rundenindex *i* rückwärts statt vorwärts gezählt. Führt man die Runde erneut durch, so ergibt sich:

$L_i =: R_{i-1}$

$L_{i-1} \oplus f(R_{i-1},K_i) \oplus f(L_i,K_i) =$

$L_{i-1} \oplus f(L_i,K_i) \oplus f(L_i,K_i) =: L_{i-1}$

Grafisch lässt sich das wie in der rechten Hälfte von **Abbildung 1.3** darstellen.