DES – Data Encryption Standard

- Verschlüsselung von Klartext-Blöcken der Länge 64-Bit
- Umwandlung in einen gleichgroßen Chiffretext-Block
- Schlüssellänge 64 Bit (praktisch nur 56 Bit)
- Verwendung einfacher Grundfunktionen:
    - XOR-verknüpfung
    - Permutation
    - Substitution
- Zunächst geheimes Projekt für militärische Zwecke → später dann Standardisierung
- Heutzutage lassen sich die 2 hoch 64 bzw 2 hoch 56 Schlüssel durchaus prüfen und dadurch die Verschlüsselung knacken
- Erweiterung „Tripple-DES“ → Schlüssellänge 112 Bit
- Die 16-fache Kombi der Bestandteile ergibt ein sehr sicheres Verfahren
- Funktionsweise:
    - Eingangspermutation des 64-Bit-Blocks
    - Zerlegung des Ergebnisses in zwei 32-Bit-Blöcke
    - Block L wird mit einer Funktion F bearbeitet und anschließend mit R XOR-verknüpft
    - Das alte L wird zum neuen R (16-fache Wdh.)
    - Nach diesen 16 Runden wird der resultierende 64-Bit-Block mit der inversen Eingangspermutation bearbeitet
    - Der wichtigste Vorgang ist die Funktion F in die Runde
- Die Funktion F expandiert zunächst den 32-Bit Block in einen 48 Bit Block und XOR-verknüpft diesen dann mit dem Schlüssel → der Ergebnisblock wird nun in 8 Blöcke à 6 Bit unterteilt, welche dann auf die S-Boxen angewendet werden (S-Boxen sind Matrizen) → daraus ergibt sich pro Box ein 4-BitBlock → die 8 4-Bit-Blöcke werden anschließen wieder zu einem 32-Bit Block zusammengefasst
- Die S-Boxen sind das entscheidende Element → diese können nicht invertiert werden
- Die S1-Box:

$$S1 = \begin{bmatrix} 14 & 4 & 13 & 1 & 2 & 15 & 11 & 8 & \\ 0 & 15 & 7 & 4 & 14 & 2 & 13 & 1 & \\ 4 & 1 & 14 & 8 & 13 & 6 & 2 & 11 & \cdots \\ 15 & 12 & 8 & 2 & 4 & 9 & 1 & 7 & \end{bmatrix}$$

    - Die Matrix besteht aus insgesamt 16 Spalten
    - Die Bits jedes 6-er Blocks werde zur Auswahl eines 4-Bit-Wertes aus der S-Bix verwendet
    - Die Bits 1 und 6 werden zur Auswahl der Zeile verwendet, die Bits 2-5 zur Auswahl der Spalte
    - Beispiel:

$$(101101) \rightarrow Zeilenauswahl : 11(dez.3)$$
$$\rightarrow Spaltenauswahl : 0110\,(dez.6)$$
$$\Rightarrow Ergebnis : 0001\,(dez.1)$$

    - Der Wert 1 taucht in der Matrix häufiger auf → Ergebnis der S-Box gibt keinerlei Aussage, welche Zahlenfolge zuvor genutzt wurde

- Schlüsseltransformation:
    - jedes achte Bit des Schlüssels wird entfernt (Paritätsbits zur Fehlerüberprüfung)
    - die verbleibenden 56 Bit werden permutiert
    - Teilung in zwei 28-Bit-Hälften

    - von den 56 Bits werden nach festem Schema 48 ausgewählt
    - Verschiebung der 28-Bit-Hälften in jeder Runde um ein oder zwei Schritte, sodass insgesamt 28 Verschiebungen in den 16 Runden durchgeführt werden
- De-Chiffrierung von DES erfolgt in der umgekehrten Reihenfolge der Verschlüsselung, wobei allerdings zu berücksichtigen ist, dass die Funktion F nicht rückwärts durchlaufen werden kann:
    - Es wird der Weg der Verschlüsselung rückwärts gelaufen, damit die Funktion F vorwärts durchlaufen werden kann → der Schlüssel des letzten Durchgangs ist bekannt und kann entsprechend umgekehrt über die Durchläufe verschoben werden → XOR Verknüpfung lässt sich invertieren (2 Werte sind bekannt, der 3 wird ermittelt)
    - 2. Teil des Chiffretextes am Ausgang der vorletzen XOR-Verknüpfung bei der Verschlüsselung wird auf die Funktion F geführt und anschließend mit dem zweiten Teil des Chiffretextes (welcher am Ausgang der letzten XOR-Verknüpfung vorliegt) XOR-verknüpft, usw. → daraus wird in derselben Anzahl der Schritte der Klartext wieder hergestellt
    - Ablauf der Verschlüsselung:
      $(K1 \,\&\, L)\ XOR\ R = X, (K2 \,\&\, X)\ XOR\ L = Y, (K3 \,\&\, Y)\ XOR\ X = Z$
    - Ablauf der Entschlüsselung:
      $(K3 \,\&\, Y)\ XOR\ Z = X, (K2 \,\&\, X)\ XOR\ Y = L, (K1 \,\&\, L)\ XOR\ X = R$
- Sicherheit von DES:
    - trotz Offenlegung und prinzipieller Einfachheit des Verfahrens und der kurzen Schlüssel: sehr sicheres Verfahren
    - seit 1997 waren einige Known-Plaintext-Attacken erfolgreich
    - es gibt „schwache“ Schlüssel (z.B. nur Nullen oder nur Einsen)

<u>Alternativen zu DES</u>

- Double-DES:
    - zweifache Anwendung von DES mit zwei verschiedenen Schlüsseln
    - Problem → Meet-in-the-Middle-Angriff möglich → Listenvergleich nach Codierung & Decodierung mit allen Schlüsseln möglich
    - Verbesserung nur um den Faktor 2, statt wie vielleicht zunächst angenommen um den Faktor 2^{56}
- Triple-DES:
    - Dreifache Anwendung des DES-Verfahrens, allerdings mit zwei Schlüsseln
    - Ablauf: Verschlüsselung mit S1 → Entschlüsselung mit S2 → Verschlüsselung mit S1
    - Praktisch an diesem Verfahren ist, dass eine entsprechende Software auch direkt das einfache DES realisiert → In diesem Fall wählt man einfach den Schlüssel S1 und S2 genau gleich → Nachteil: höhere Rechenzeit
- Weitere Alternativen zu DES:
    - IDEA (Intern. Data Encryption Standard) → 8 Runden à 64-Bit, Schlüssellänge 128 Bit → sehr hohe Sicherheit, aber noch nicht gut untersucht
    - RC2 (Rivest Cipher) → variable Schlüssellänge (bis 2^{1024}), Blockchiffre ohne S-Boxen, wurde bis vor kurzem geheim gehalten, Varianten RC4, RC5, RC6
    - Skipjack → Block-Chiffre, Schlüssellänge 80-Bit, 32 Runden
    - AES → Block mit 10 oder 12 Bit, Schlüssellängen 128, 192 oder 256 Bit
- Stromchiffre RC4:
    - eine Stromchiffre ist prinzipiell ein Pseudo-Zufallsgenerator, der mit einem Startwert (Seed) initialisiert wird

- der Seed wird aus dem Schlüssel erzeugt
- die erzeugte Zufallsfolge wird mit dem Klartext XOR-verknüpft, d.h. der „Schlüssel“ ist genauso lang wie die Nachricht
- der Empfänger entschlüsselt den Chiffretext, indem er dieselbe Zufallsfolge mithilfe des Seeds erzeugt und mit dem Chiffretext XOR-verknüpft
- Funktionsweise (Erzeugung des Schlüsselstroms):
  Startwert (Seed) $s_i \ (0 \le i \le 255)$ enthält jeweils einmal die Werte 0-255
  Schleife $(i = 0,\ j = 0)$
  $$\left.\begin{aligned} i &= (i+1) \bmod 256 \\ j &= (j + s_i) \bmod 256 \end{aligned}\right\} \textit{vertausche } s_i \textit{ mit } s_j$$
  $$t = (s_i + s_j) \bmod 256$$
  $$K = s_t$$
- Seed kann als Startwert für die Schleife in der Entschlüsselung verwendet werden

<u>Operationsmodi</u>

- ECB - Electronic Code Book:
  - einfachster Modus (entspricht der Grundfunktion von DES): ein Klartextblock wird in einen Chiffretextblock gewandelt → beide sind gleich groß
  - es ist theoretisch möglich, für jeden Schlüssel ein Codebuch anzulegen, in
  - Vorteil: eine lineare Bearbeitung des Klartextes ist nicht zwingend erforderlich
  - Nachteil: gleiche Klartextblöcke m_j ergeben denselben Chiffretext c_j
  - Beispiel für einen Angriff auf ECB:
    Eine Bank 1 überweist Geld zu einer Bank 2 → der Angreifer hat bei beiden Banken ein Konto und tätigt nach einander mehrere komplett gleiche Überweisungen → im Chiffretext tauchen bestimmte Muster wieder auf → dies sind die Kontoinformationen → der Angreifer kann nun selbst an Bank 2 Überweisungen schicken und sich somit Geld auf sein Konto überweisen
- CBC - Cipher Block Chaining:
  - Vermeidung der Probleme von ECB lassen durch die Verkettung von Chiffretextblöcken
  - Chiffretestblock A wird mit dem nachfolgenden Klartextblock B XOR-verknüpft und erst dieses Ergebnis zu Chiffretextblock B gewandelt
  - gleiche Klartextblöcke werden damit zu verschiedenen Chiffretextblöcken
  - komplett gleiche Nachrichten ergeben auch denselben Chiffretext → der erste Block mit zufälligen Werten gefüllt (Initialisierungsvektor IV)
  - der IV wird mit dem Chiffretext an den Empfänger übertragen
  - Bitfehler bei der Übertragung von c_j beeinflussen nur das zugehörige m_j und das nachfolgende m_{j+1}
  - fehlende oder ergänzte Bits zerstören jedoch den kompletten nachfolgenden Chiffretext
  - Verschlüsselung:
    $$c_0 = IV,\ c_j = E\left(c_{j-1}\ XOR\ m_j\right)$$
  - Entschlüsselung:
    $$m_j = c_{j-1}\ XOR\ D\left(c_j\right)$$
  - Beweis des Zusammenhangs:

$$m_j = c_{j-1} \; XOR \; D\left(E\left(c_{j-1} \; XOR \; m_j\right)\right)$$

$$m_j = \underbrace{\underbrace{c_{j-1} \; XOR \; c_{j-1}}_{=0} \; XOR \; m_j}_{=m_j}$$

- Ein Bitfehler (Bitdreher) in c_j beeinflusst nur den dazugehörigen Klartext m_j und den darauffolgenden Klartext m_{j+1}

$$m_j = c_{j-1} \; XOR \; D\left(c_j\right)$$

$$m_{j+1} = c_j \; XOR \; D\left(c_{j+1}\right)$$

$$m_{j+2} = c_{j+1} \; XOR \; D\left(c_{j+2}\right) \rightarrow kein \; c_j \; mehr \; in \; der \; Formel$$