10.4 Sichere Blockverschlüsselung

Verschlüsselung im Rechner:

Stromverschlüsselung *(stream cipher):*

kleine Klartexteinheiten (Bytes, Bits)
werden polyalphabetisch verschlüsselt

Blockverschlüsselung *(block cipher):*

große Klartexteinheiten (z.B. Blöcke von 64 Bits)
werden separat *sicher* verschlüsselt

Vorteile: schnell *und* sicher, Direktzugriff möglich
Nachteil: Block-Wiederholungen im Geheimtext sichtbar

10.4.1 Data Encryption Standard – DES

(IBM, NBS 1977)

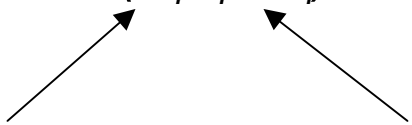
- Chiffriert/dechiffriert werden *64-Bit-Blöcke* mit *e(K,x)* bzw. *d(K,c)*
- unter Verwendung eines *56-Bit-Schlüssels K*
- in einem kombinierten Transpositions/Substitutionsverfahren

Ziel: Erzeugung von „Konfusion“ durch Substitution,
von „Diffusion“ durch Transposition und damit
Approximation einer „Zufallsverschlüsselung“, d.h.
Bitmuster im Geheimtext haben keinen erkennbaren
Bezug zu Bitmustern im Klartext,

z.B. selbst 000...00 oder 111...11 werden jeweils in
„zufällig aussehende“ Bitmuster chiffriert

In einem *16-fachen Zyklus* kommt eine Abbildung *f* wiederholt zur Anwendung:

$f(R_{i-1}, K_i)$, $i=1,..,16$



abgeleitet aus Klartext und Schlüssel K

abgeleitet aus Schlüssel K

$L_i = R_{i-1}$ $R_i = L_{i-1}$ $f(R_{i-1}, K_i)$ (= **xor**)

L_0 = *linke Hälfte der permutierten Klartext-Bits*

R_0 = *rechte Hälfte der permutierten Klartext-Bits*

Geheimtext = $R_{16}\, L_{16}$, *zurückpermutiert*

Beh.: Der Chiffrier-Algorithmus *dechiffriert*, wenn die Teilschlüssel K_i in *umgekehrter* Reihenfolge verwendet werden.

Bew.:

1. Die Permutation der Bits des Geheimtextes liefert $R_{16}\, L_{16}$.

2. Ein Schritt *i, i=16,..,1*, erzeugt aus $R_i\, L_i$ den Wert $l\, r$ mit

   $l = L_i = R_{i-1}$ und

   $r = R_i \quad f(L_i, K_i)$
   $= (L_{i-1} \quad f(R_{i-1}, K_i)) \quad f(L_i, K_i)$
   $= L_{i-1}$ wegen $L_i = R_{i-1}$

   also den Wert $R_{i-1}\, L_{i-1}$.

3. Die Vertauschung der $R_0\, L_0$ liefert $L_0\, R_0$, und die Rückpermutation liefert den Klartext.

Bemerkungen zum DES:

Schlüsselgröße 56 Bits ist heutzutage zu klein.

(Zur Erinnerung: 88 bei monoalphabetischer Substitution)

Dennoch viel sicherer als Substitutionsverfahren, weil Beziehung zwischen Klartext und Geheimtext ungleich komplizierter.

DES kann aber als Bestandteil komplexerer Verfahren verwendet werden, z.B. Dreifach-Verschlüsselung (s.u.) oder rückgekoppelte Verschlüsselung (→10.4.3);
auch als Zufallszahlengenerator einsetzbar.

DES ist gut in Hardware implementierbar – Chiffrierraten in der Größenordnung von 100 MB/sec werden erreicht.

Erhöhte Sicherheit durch 3-fach-Verschlüsselung mit *e* bzw. *d* :

$$e(K_1, d(K_2, e(K_3,x)))$$

Beachte:

① *2-fach*-Verschlüsselung bringt nicht den erhofften Gewinn:
Klartextangriff mit Ausprobieren von maximal
2^{57} Schlüsseln möglich.

② Auch bei 3-fach-Verschlüsselung ist die effektive
Schlüssellänge nur 112 (statt 168) Bits
(siehe *Schneier* Kap. 15).

Zu ①: Klartext x und zugehöriger Geheimtext $c = e(K_b, e(K_a,x))$ liegen vor – gesucht sind K_a und K_b .

Für alle Schlüssel K_i den einfach verschlüsselten Klartext tabellieren:

$$c_i = e(K_i,x) \ .$$

Für alle Schlüssel K_j Geheimtext einfach entschlüsseln,

$$y_j = d(K_j,c) \ ,$$

und jeweils prüfen, ob das y_j unter den c_i vorkommt.

Falls $y_j = c_i$ für ein bestimmtes j und i,

„ K_i und K_j sind die gesuchten Schlüssel“ vermuten und mit weiterem Klartext/Geheimtext-Paar überprüfen.

→ Höchstens $2*2^{56}$ Schlüssel werden durchprobiert.

DES in Unix:

❶ *Zur Erinnerung:* Paßwort-Verschlüsselung bei Unix (4.1.1)
arbeitet mit **`crypt`**`(3)` (nicht **`crypt`**`(1)`!)
Die **`crypt`** -Routine verschlüsselt 0 mit dem Paßwort als Schlüssel (!) unter Verwendung einer DES-Variante, in die das *salt* eingeht.

❷ Der Befehl **`des`** verschlüsselt mit DES, z.B.

```
jefe: des -E >temp
Enter key:
Verifying password - Enter key:
1234567890
abc
jefe: des -D <temp
Enter key:
1234567890
abc
```

10.4.2 IDEA

(International Data Encryption Algorithm, Lai/Massey 1990)

- verschlüsselt 64-Bit-Blöcke
- bei einer Schlüssellänge von 128 Bits,
- betreibt wie DES Konfusion und Diffusion,
- arbeitet mit 8 (statt 16) Runden,
- verwendet nur *xor*, Addition und Multiplikation ,
- ist doppelt so schnell wie DES,
- gilt als erheblich **sicherer** als DES.

10.4.3 Rückgekoppelter Geheimtext

Elementare Blockverschlüsselung *(ECB mode, electronic code book)* hat zwei Nachteile:

- *Wiederholungen* von Klartextblöcken im Geheimtext erkennbar
- *Wiedereinspielen* zuvor abgefangener Blöcke verletzt Authentizität

Abhilfe: Schlüssel laufend durch *den Text selbst* modifizieren, z.B. so:

Cipher Feedback (CFB) benutzt die Blockverschlüsselung für eine Stromverschlüsselung mit Rückkoppelung

Cipher Block Chaining (CBC) benutzt bei der Verschlüsselung eines Klartextblocks den vorangegangenen Geheimtextblock

10.4.3.1 DES als Zufallszahlengenerator

für Strom/Blockverschlüsselung mit Abreißblock (10.3.2.2):

die mit K verschlüsselten Werte eines Zählers $i = 0,1,2,...$
– oder auch nur deren letzte k Bits –
dienen als *one-time pad* mit ***xor***-Verknüpfung;
Periode ist 2^{64} !

Beispiel $k=1$ (aufwendig):

Verschlüsselung des i-ten Klartext-Bits: $c_i = x_i \quad [e(K,i)]_0$

Entschlüsselung des i-ten Geheimtext-Bits: $x_i = c_i \quad [e(K,i)]_0$

(DES-Entschlüsselung d kommt *nicht* zum Einsatz)

10.4.3.2 Cipher Feedback (CFB)

benutzt die Blockverschlüsselung für eine
Stromverschlüsselung mit Rückkoppelung:

Beim Sender wird statt *i* der Inhalt eines Schieberegisters verwendet, in das bei jedem Schritt das zuletzt generierte Geheimtext-Bit eingeschoben wird. Statt $[e(K,i)]_0$ kommt also zum Einsatz

$$[e(K, c_{i-1}\, c_{i-2} \ldots c_{i-64})]_0$$

Empfängerseitig wird ein eintreffendes Geheimtext-Bit nicht nur entschlüsselt, sondern auch – für die nächste Entschlüsselung – in das dortige Schieberegister eingebracht.

10.4.3.3 Cipher Block Chaining (CBC)

benutzt bei der Verschlüsselung eines

Klartext*blocks* den vorangegangenen Geheimtext*block*

(effizienter als CFB)

Verschlüsselung eines Klartextblocks x_i : $c_i = e(K, x_i \quad c_{i-1})$

Entschlüsselung eines Geheimtextblocks c_i : $x_i = d(K, c_i) \quad c_{i-1}$

$(= d(K, e(K, x_i \quad c_{i-1})) \quad c_{i-1} = x_i \quad c_{i-1} \quad c_{i-1} = x_i)$

10.5 Asymmetrische Verschlüsselung

(asymmetric encryption, public-key encryption)

Prinzip (Diffie, Hellman, Merkle 1976-78):

- Statt eines Schlüssels K gibt es ein *Schlüsselpaar* K_E, K_D zum Verschlüsseln bzw. Entschlüsseln;

- der **öffentliche Schlüssel** K_E *(public key)* muß nicht geheimgehalten werden, da er nicht aus dem **privaten Schlüssel** K_D *(private key)* bestimmt werden kann.

Gewinn: Problem der Übermittlung eines geheimen Schlüssels zwischen Kommunikationspartnern entfällt;

weitere Vorteile s.u.

Kryptoanalyse wird potentiell dadurch erleichtert, daß der öffentliche Schlüssel bekannt ist.

➔ Sichere asymmetrische Verfahren sind so konzipiert, daß die Ermittlung des geheimen Schlüssels aus dem öffentlichen Schlüssel *am Aufwand scheitert* (10.3.2.4).

Als Grundlage dienen vorzugsweise Probleme, die *NP-vollständig* sind und damit *als nicht effizient lösbar gelten.*

(Lösung erfordert *exponentiellen Aufwand*.)

10.5.1 Knapsack-Verschlüsselung

(Merkle/Hellman 1978)

Knapsack-Problem ist NP-vollständig:

Gegeben: Folge $A = a_1,...a_n$ natürlicher Zahlen und natürliche Zahl C

Gesucht: Teilfolge von A derart, daß die Summe der Elemente gleich C ist

(M.a.W.: gesucht ist Folge/"Vektor" M von Nullen und Einsen mit $A\,M = C$)

Eigenschaften: Weder Existenz noch Eindeutigkeit sind gesichert.

Komplexität der besten bekannten Algorithmen ist von der Ordnung $O(2^{n/2})$.

Beispiel: $A = 1\ \ 10\ \ 5\ \ 22\ \ 3$ $C = 14$

wird gelöst durch $M = 1\ 1\ 0\ 0\ 1$

Spezieller Knapsack ist

Einfacher Knapsack *(superincreasing knapsack):* $a_i > \sum_{j=1,..,i-1} a_j$

Beispiel: $A = 1\ \ 3\ \ 5\ \ 10\ \ 22$ $C = 14$ $M = 1\ 1\ 0\ 1\ 0$

Lösung – falls existent – wird gefunden durch den Versuch,
„den Rucksack von rechts her zu füllen“

Idee für asymmetrische Verschlüsselung:

Aus einfachem Knapsack A‘ wird
ein schwerer Knapsack A gemacht,
(genannt *trapdoor knapsack*)

1. Wähle $u \geq 2\, a'_n$; damit ist $u > \sum_{i=1,n} a'_i$

2. Wähle zu u teilerfremdes w und berechne w^{-1} mit
$$w\, w^{-1} \bmod u = 1$$

3. Bilde A mit $a_i = w\, a'_i \bmod u$

Beh.: Wenn M Lösung von $C = A\,M$ ist, dann auch von $C' = A'\,M$ mit $C' = w^{-1}\,C \mod u$ (und umgekehrt).

Bew.:

$$\begin{aligned} C' &= w^{-1}\,C \mod u \\ &= w^{-1}\,A\,M \mod u \\ &= w^{-1}\,(\sum(w\,a'_i \mod u)\,m_i) \mod u \\ &= w^{-1}\,w\,A'\,M \mod u \\ &= A'\,M \mod u \\ &= A'\,M \end{aligned}$$

- *Chiffrieren* eines Klartextes M (als Dualzahl) mit öffentlichem Schlüssel A: $C = A\,M$

- *Dechiffrieren* eines Geheimtextes C (als Dualzahl) mit privatem Schlüssel (A', w^{-1}, u) durch Lösen von $C' = A'\,M$ d.h. von $w^{-1}\,C \mod u = A'\,M$

Beispiel:

$A' = 1\ 3\ 5\ 10 \qquad u = 20 \qquad\qquad w = 7 \qquad w^{-1} = 3$

→ $A = 7\ 1\ 15\ 10$

Klartext: $M = 1\ 1\ 0\ 1$ chiffriert: $C = 18$

Dechiffrieren: $C' = w^{-1}\ C \mod u = 3 * 18 \mod 20 = 14$

Lösen von $14 = A'\ M$ ist simpel,
weil A‘ einfacher Knapsack: $14 = 1+3+10$, also

$M = 1\ 1\ 0\ 1$