Security

- LV 4120 und 7240 -

Einwegfunktionen

- Idee und Konzept der Einwegfunktion
- Einfache Realisierungsmöglichkeit für eine Einwegfunktion
- Message Digest (MD) und Message Authentication Code (MAC)
- Hashfunktionen mit blockorientierten Verschlüsselungsalgorithmen
- Secure Hash Algorithm SHA-1

Definition:

Es seien **X** und **Y** zwei endliche, nichtleere Mengen. Eine Funktion $\mathbf{f : x \rightarrow y}$ heißt Einwegfunktion, wenn für alle $x \in \mathbf{X}$ der **Funktionswert** $f(x)$ leicht berechnet werden kann, es aber de facto unmöglich ist, für ein gegebenes $y \in \mathbf{Y}$ ein $x \in \mathbf{X}$ mit $f(x) = y$ zu finden. Der Wert x muss dabei nicht eindeutig bestimmt sein.

- X und Y müssen eine genügend große Kardinalität besitzen, damit die Attacke der vollständigen Suche ausgeschlossen werden kann.
- In der Praxis werden die Mengen **X** und **Y** durch Binärzeichen repräsentiert, deren Länge zwischen 64 und 2048 Bit liegen.
- Eine Verschlüsselungsfunktion $\mathbf{E_k : x \rightarrow y}$ kann als Einwegfunktion angesehen werden, wenn der Schlüssel k geheimgehalten wird.

Trapdoor one-way functions:

Eine Funktion f heißt Falltür-Einwegfunktion, wenn bei Kenntnis gewisser geheim zu haltender **Zusatzinformationen** für alle $y \in \mathbf{Y}$ ein $x \in \mathbf{X}$ mit $f(x) = y$ existiert und leicht ermittelt werden kann.

- In zahlreichen Anwendungen soll die **Inventierbarkeit** der Einwegfunktion aber gerade vermieden werden, z. B. Passwortablage f(**P**).
- Bei der Passwortablage in einem Verzeichnis reicht es aus, dass das Passwort **P** mittels der Einwegfunktion f in den Wert **w = f(P)** transformiert und anstelle des Passwortes gespeichert wird.
- Dadurch ist gewährleistet, dass aus dem gespeicherten Wert w das Passwort **P** auch dann nicht rekonstruiert werden kann, wenn f öffentlich bekannt ist – ein Vergleich jedoch damit ermöglicht wird.

Einfache Realisierungsmöglichkeiten:

Sind die Mengen $\mathbf{X} = \mathbf{Y} = \mathbf{Z}_n$ gegeben, wobei n eine große natürliche Zahl ist.

Dann kann die folgende **Funktion f(x)** für jede natürliche Zahl k als eine der bedeutesten Einwegfunktionen der Kryptologie angesehen werden:

$$y = f(x) = x^k \bmod n$$

- Der Funktionswert y = f(x) kann schnell berechnet werden.
- Praxisrelevante Größenordnungen von n und k sind:

$$2^{1024}$$

Für $x = f^{-1}(y)$ existiert kein effizienter Algorithmus

Kap. 5: Einwegfunktionen

Teil 1: Einwegfunktionen und Hashfunktionen

- Integritätsschutz
- Hashfunktion

Zielsetzung:

Das Ziel des Integritätsschutzes ist es, daß ein Empfänger einer Nachricht feststellen kann, ob er diese Nachricht unverfälscht erhalten hat. Das Grundprinzip des Integritätsschutzes besteht darin, die Nachricht unverschlüsselt und unverändert zu übersenden, gleichzeitig aber bestimmte Kontrollinformationen mitzuschicken, die die Kontrolle auf Unverfälschtheit der eigentlichen Nachricht ermöglicht. Für diese Kontrolldaten stellen sich damit folgende Bedingungen:

Bedingungen:

- Der Umfang der Kontrollinformationen muß möglichst gering sein, um die zusätzlich zu übertragenden Informationen zu minimieren.
- Praktisch jede Manipulation, auch nur eines einzelnen Bits der Nachricht muß anhand der Kontrollinformation feststellbar sein.
- Die Kontrollinformationen müssen unmanipulierbar übertragen werden können.

Hashfunktion:

Eine Hashfunktion ist eine Datentransformation mit folgenden Eigenschaften:

- **Kompressionseigenschaft:** Beliebig lange Bitfolgen werden auf Bitfolgen fester, im allgemeinen kürzerer Länge abgebildet (typischerweise 128 - 524 Bit)
- **Einwegeigenschaft:** Es muß praktisch unmöglich sein, zu einem vorgegebenen Hashwert eine Nachricht zu finden, deren Hashwert der vorgegebene Hashwert ist.
- **Kollisionsresistenz:** Es muß praktisch unmöglich sein, zwei Nachrichten zu finden, die zum gleichen Hashwert führen.

Prüfvorgang:

Mit Hilfe einer beiden Kommunikationspartnern bekannten Hashfunktion können A und B die Integrität einer Nachricht überprüfen: Alice hasht ihre Nachricht, und übermittelt diese und den Hashwert so an Bob, daß die Unverfälschtheit des Hashwertes gewährleistet ist. Bob hasht die empfangene Nachricht ebenfalls und vergleicht sein Ergebnis mit dem von Alice gelieferten Hashwert.

Kap. 5: Einwegfunktionen

Teil 2: Kryptographische Prüfsummen

- Message Digest (MD)
- Message Authentification Code (MAC)
- Kryptographische Hashfunktionen

Message Authentication Code (MAC):

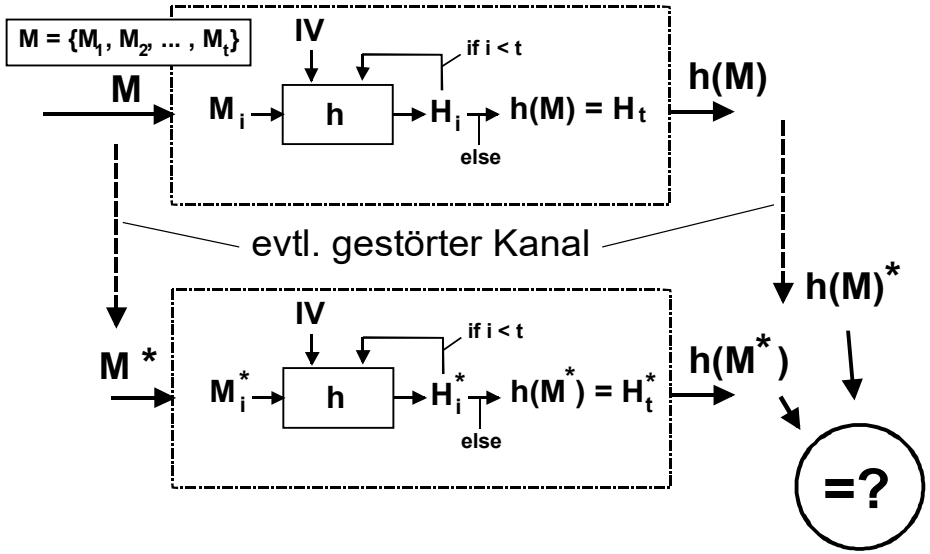
Ein Message Authentication Code ist eine kryptographische Checksumme zur Nachrichtensicherung, also eine Datentransformation, bei der zusätzlich ein geheimer Schlüssel in die Berechnung mit folgenden Eigenschaften eingeht:

- **Kompressionseigenschaft:** Beliebig lange Bitfolgen werden auf Bitfolgen fester im allgemeinen kürzerer Länge abgebildet.
- **Fälschungssicherheit:** Für jeden, der nicht im Besitz des Schlüssels ist, muß es praktisch unmöglich sein, den MAC-Wert einer neuen Nachricht zu berechnen, selbst wenn er im Besitz einiger alter Nachrichten mit den zugehörigen MAC-Werten gelangt ist.

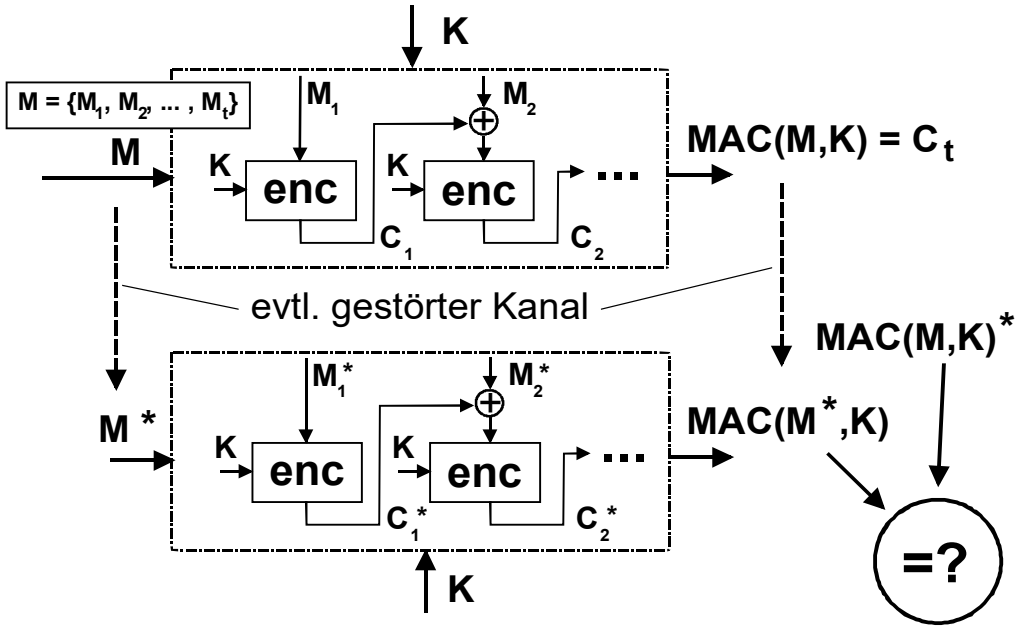
Prüfvorgang:

Besitzen Alice und Bob einen MAC und einen gemeinsamen, geheimen MAC-Schlüssel, so authentisiert Alice ihre Nachricht einfach dadurch, daß sie den MAC-Wert der Nachricht berechnet und zusammen mit der Nachricht an Bob schickt. Bob berechnet seinerseits den MAC-Wert der empfangenen Nachricht mit dem auch ihm bekannten MAC-Schlüssel.

Message Digest (MD)



Message Authentication Code (MAC)



„Schlüsselgesteuerte“ Hashfunktion:

Im Anhang D zum **Standard X.509** ist noch folgende Möglichkeit zur Realisierung einer Hashfunktion aufgezeigt:

- Zunächst werde die Nachricht **m** in geeignet lange Blöcke $m_1, m_2, ..., m_r$ unterteilt und ggf. mit Einsen aufgefüllt.
- Der Hashwert $h = H(\mathbf{m})$ wird dann für $i = 1, 2, ..., \mathbf{r}$ durch die nach-folgende Vorschrift ermittelt:

$$h_i = (h_{i-1} + m_i)^2 \bmod n,$$

  wobei als Startwert $h_0 = 0$ und als Hashwert $H(m) = h_r$ gesetzt werden.
- Durch die Modulofunktion wird das Vertauschen der Blöcke wirkungs-voll verhindert.

Kap. 5: Einwegfunktionen

Teil 3: Secure Hash Algorithm

- Beschreibung des Verfahrens
- Praktische Implementierung

SHA-1 ist heute eine sehr häufig benutzte Hashfunktion und wird beispielsweise im Digital Signature Standard (DSS) verwendet.

SHA-1:

- entwickelt vom U.S.-Government (NIST zusammen mit NSA)
- veröffentlicht 1993 unter der Bezeichnung FIPS PUB 180
- der Hashwert **h(M)** ist **160 Bit** lang (feste)
- die Original-Message **M** ist bis zu **$(2^{64} - 1)$ Bit** lang (variabel)
- Die Blockgröße $\mathbf{M_i}$ beträgt **512 Bit**
- Die Rundenzahl ist 80

Anmerkung:

Wir gehen im folgenden davon aus, dass uns die Original-Message M in Binärform $x \in \{0, 1\}^*$ vorliegt.

Anpassung der erforderlichen Länge:

Sei $\mathbf{x} \in \{0, 1\}^*$ die originäre Message und $|\mathbf{x}| < 2^{64}$. Dann wird x so ergänzt (expandiert), dass **x** ein Vielfaches von 512 Bit ist.

1. Zunächst wird an **x** eine 1 angehängt: $\mathbf{x} \leftarrow \mathbf{x} \circ 1$
2. Dann werden an **x** so viele Nullen angehängt, dass $\mathbf{x} = k \cdot 512 - 64$ bzw. $\mathbf{x} \equiv 448 \bmod 512$.
3. Schließlich wird die originäre **x**-Länge in Bits als 64-Bit-Integerzahl geschrieben und ebenfalls an **x** angehängt.

$\Rightarrow$ Das **erweiterte x** hat somit eine Gesamtlänge von $\mathbf{x} = k \cdot 512$ Bit.
vgl. Pseudocode S. 17

Anpassung der erforderlichen Länge:
(Pseudocode)

```
// Vorbereitung der Nachricht 'message':

var int message_laenge := bit_length(message)

erweitere message um bit "1"

erweitere message um bits "0" bis Länge von message
in bits ≡ 448 (mod 512)

erweitere message um message_laenge als 64-Bit big-
endian Integer
```

Berechnung des Hashwertes:

Bei der Berechnung von **SHA-1(x)** werden

- die Funktionen

  $f_t : \{0, 1\}^{32} \times \{0, 1\}^{32} \times \{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$

sowie

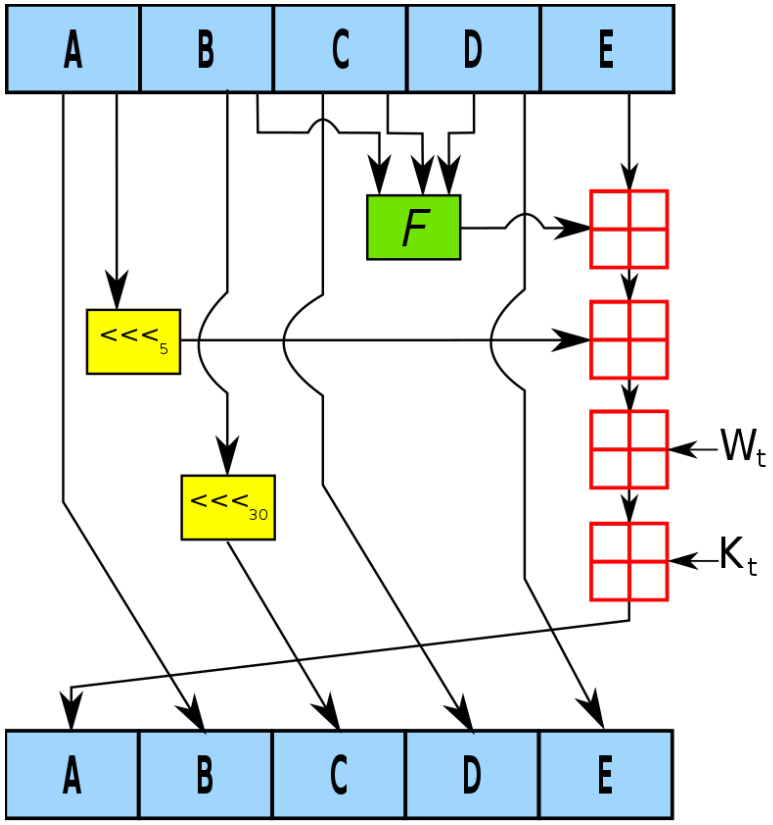
- die Konstanten

  K_t

gemäß nebenstehender Abbildung benutzt.

W_t = 32 Bit Wort des **erweiterten x**

t = Rundenzähler [0; 79]



Definitionen:

$$f_t(B, C, D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D) & 0 \le t \le 19 \\ B \oplus C \oplus D & 20 \le t \le 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \le t \le 59 \\ B \oplus C \oplus D & 60 \le t \le 79 \end{cases} \quad \text{für}$$

$$K_t = \begin{cases} 5A827999 & 0 \le t \le 19 \\ 6ED9EBA1 & 20 \le t \le 39 \\ 8F1BBCDC & 40 \le t \le 59 \\ CA62C1D6 & 60 \le t \le 79 \end{cases} \quad \text{für}$$

Hashwertprozedur:

A. Zerlege den **erweiterten Bitstring x** im Anschluss an den vorangegangenen Schritt 3 in 512 Bit Blöcke, d. h. $x = M_1 M_2 M_3 \ldots M_k$.

B. Initialisiere $H_0 = 67452301$, $H_1 = EFCDAB89$, $H_2 = 98BADCFE$, $H_3 = 10325476$, $H_4 = C3D2E1F0$.

C. **for** i = 1 **to** k **do** // Wiederhole für alle Blöcke M_i

1. Schreibe jedes M_i als Folge von **sechzehn** 32-Bit-Wörtern W_n, d. h. $M_i = W_0 W_1 \ldots W_{15}$**;**
2. Erweitere die **sechzehn** 32-Bit-Wörter um 64 weitere 32-Bit-Wörter, so dass sich insgesamt 80 Wörter je 32 Bit ergeben, d. h. Für t = 16, 17, ..., 79 setze $W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$**;**

3. Setze $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$;
4. **Hauptschleife**: Für t = 0, 1, ..., 79 setze nunmehr

$$T = S^5(A) + f_t(B, C, D) + E + W_t + K_t$$

$$E = D,\ D = C,\ C = S^{30}(B),\ B = A,\ A = T;$$

5. Setze im aktuellen Schritt i

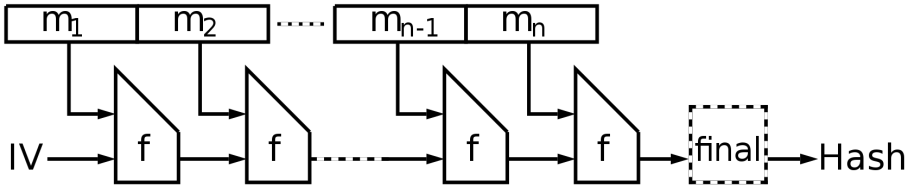
$$H_0 = H_0 + A,\ H_1 = H_1 + B,\ H_2 = H_2 + C,\ H_3 = H_3 + D,\ H_4 = H_4 + E;$$

next i // Ende des i-ten Durchlaufs

Im Anschluss an den **k-ten** Durchlauf obiger Prozedur ist der Hashwert von x dann gegeben durch

$$\text{SHA-1}(x) = H_0\, H_1\, H_2\, H_3\, H_4$$

Der so berechnete Hashwert **SHA-1(x)** wird nun mit dem nächsten Nachrichtenblock gehasht bzw. nach Abarbeitung des letzten Nachrichtenblocks als **Hashwert** der **Gesamt-Nachricht** ausgegeben.



Anmerkung:

- In der vorangestellten Prozedur bedeutet $S^k(w)$ einen zirkulären Links-Shift eines 32-Bit-Wortes w um k Bits.
- Außerdem bedeutet + die Addition zweier durch 32-Bit-Wörter dargestellter Zahlen mod 2^{32}.
- Mit $\oplus$ ist die bitweise XOR-Verknüpfung bezeichnet.

SHA-1:

- Ein kritisches Angriffsszenario setzt voraus, dass der Angreifer eine (zumindest in Teilen) sinnvolle Variante eines Dokuments erzeugen kann, die den gleichen SHA-1-Wert besitzt.
- 2017 wurde die erste Kollision von SHA-1 anhand zweier unterschiedlicher PDF-Dateien veröffentlicht.
- Der Rechenaufwand war dabei enorm. Eine einzelne CPU hätte etwa 6500 Jahre dafür benötigt.
- Derzeit werden Hashfunktionen der SHA-2-Familie (SHA-224, SHA-256, SHA-384 und SHA-512) empfohlen.
- Langfristig sollen diese durch den neuen Standard **SHA-3** ersetzt werden.

Berechnungen:

Nachricht	Hashwert der Nachricht
4711	e8fed7c5621fcc32f5db606fefee7c98f36cc2fa
4712	5ee217943f0d94ebbbdc7825adfd41fea2268f05
“”	da39a3ee5e6b4b0d3255bfef95601890afd80709
“Franz“	cb7ec4b22a9ba1e588e7f76247c201792d82e262
“Ganz“	e24176bf5cce5c6630792c6f2eb63144678f3ed5

Kap. 5: Einwegfunktionen

Zusammenfassung:

- In diesem Kapitel wurde das Konzept der Einweg- und Falltür-Einwegfunktionen erläutert.
- Was deren Klassifikation anbelangt, so wird zwischen Manipulation Detection Codes und Message Authentication Codes unterschieden.
- Daneben wurden Realisierungen mit Hilfe der modularen Potenz-funktion sowie eines geeigneten Blockverschlüsslers vorgestellt.
- Des Weiteren wurden Einwegfunktionen auf der Basis des Faktori-sierungsproblems und diskreten Logarithmusproblems besprochen.

Kap. 5: Einwegfunktionen

Zusammenfassung (Fortsetzung):

- Der für den Einsatz mit dem Digital Signature Standard (DSS) vom NIST zusammen mit der NSA entwickelte SHA-1 produziert einen 160 Bit langen Hashwert.
- Dadurch bietet SHA-1 einen besseren Schutz vor einem Brute-Force-Angriff (einschließlich Geburtstagsangriff) als die Vorgängeralgorithmen MD4 und MD5.
- SHA-1 kann heutzutage nicht mehr als ein sehr sicherer Hashalgorithmus bezeichnet werden.