Vergleich der Sicherheitseigenschaften

Die folgenden zwei Reduktionen sind im Standardmodell gültig.

Sei $h : X \to Z$ eine Hashfunktion.

Können wir zweite Urbilder berechnen, so können wir Kollisionen berechnen:

- Wähle $x \in X$ zufällig.
- Berechne ein zweites Urbild $x' \neq x$ mit $h(x') = h(x)$.
- Ausgabe von x, x'.

Resistenz gegen Kollisionen impliziert also Resistenz gegen schwache Kollisionen.

Vergleich der Sicherheitseigenschaften

Sei $h : X \to Y$ eine Kompressionsfunktion mit $\#X \geq 2\#Y$.

Können wir Urbilder berechnen, so können wir Kollisionen berechnen:

- Wähle $x \in X$ zufällig.
- Berechne ein Urbild x' von $h(x)$.
- Ausgabe von x, x', wenn $x \neq x'$. Sonst Fehler.

Sei $C = \{h^{-1}(\{h(x)\}) \mid x \in X\}$. Die Erfolgswahrscheinlichkeit ist

$$\begin{aligned} P &= (1/\#X) \sum_{x \in X} (\#h^{-1}(\{h(x)\}) - 1)/\#h^{-1}(\{h(x)\}) \\ &= (1/\#X) \sum_{c \in C} \sum_{x \in c} (\#c - 1)/\#c = (1/\#X) \sum_{c \in C} (\#c - 1) \\ &\geq (\#X - \#Y)/\#X \geq (\#X - \#X/2)/\#X \geq 1/2. \end{aligned}$$

Resistenz gegen Kollisionen impliziert also die Einweg-Eigenschaft.

Weitere Sicherheitseigenschaften

Hashfunktion $h : \{0,1\}^* \to \{0,1\}^n$.

Einwegeigenschaft auch bei partiell bekanntem Urbild:

- Aus $h(x)$ und einigen bekannten Bits von x sowie $|x|$ ganz x berechnen, z.B. aus $h(x_1||x_2)$ und x_1 den Wert von x_2 berechnen.
- Laufzeit eines Angreifers sollte nicht wesentlich schneller als $2^{\min(b,n)}$ sein, wobei b die Anzahl der unbekannten Bits in x ist.
- Ist Verschärfung der Einwegeigenschaft.
- Gilt im Zufallsorakelmodell.

Kollisionsresistenz bei partiell vorgegebenem Urbild:

- Folgt aus Kollisionsresistenz, für h kollisionsresistent ist $x_1 \mapsto h(x_1||x_2)$ ebenfalls kollisionsresistent, etc.
- Impliziert Einwegeigenschaft bei partiell bekanntem Urbild.

Konstruktion von Hashfunktionen

Neben Einwegeigenschaft und Kollisionsresistenz sollen Hashwerte von langen Nachrichten effizient ohne großen Speicheraufwand berechnet werden können (Magnetband, ...).

Allgemeines Prinzip: Iterierung.

Nachricht mit geeignetem Bitstring und Nachrichtenlänge padden (!), dann in geeignete Blöcke m_i der Blocklänge (Bitlänge) r aufteilen.

Bei der Berechnung des Hashwerts eine Zustandsvariable h_i der Bitlänge n mitführen. Erster Zustand ist h_0 = konstanter IV, letzter Zustand h_n ist Hashwert.

In jedem Schritt eine Kompressionsfunktion auf m_i und h_i anwenden, liefert h_{i+1}. Ähnlich wie im CBC Mode für Blockchiffren.

Bemerkung zum Padding

Verschiedene Varianten denkbar. Für nachfolgenden Satz/Beweis gewünschte Eigenschaft: Ist $M \in \{0,1\}^*$ eine Nachricht mit Padding, so soll es kein $T \in \{0,1\}^*$ geben, so daß $T||M$ eine Nachricht mit Padding ist.

Bei begrenzter Nachrichtenlänge von 2^n Bits:

- Nachricht mit beliebigen Bits zur vollen Blocklänge auffüllen.
- Länge der Nachricht als Bitstring fester Länge n anhängen (mit führenden Nullen, damit durch die Blocklänge teilbar).
- Liefert aber formal keine Funktion, die auf $\{0,1\}^*$ definiert ist.

Bei unbegrenzter Nachrichtenlänge:

- Padding z.B. von der Form $1||n_l||0||n_{l-1}||\cdots||0||n_0$, wobei die $n_i \in \{0,1\}^{r-1}$ die Nachrichtenlänge kodieren.

Davies-Meyer Kompressionsfunktion

Man benutzt einen Blockchiffre, um eine Kompressionsfunktion zu erhalten.

Blockchiffre $\mathcal{E} : \{0,1\}^k \times \{0,1\}^b \to \{0,1\}^b$ mit Schlüssellänge k und Blocklänge b.

Liefert Kompressionsfunktion $f : \{0,1\}^{k+b} \to \{0,1\}^b$ mit $f(K||m) = \mathcal{E}(K,m) \oplus m$.

Ist nicht sehr effizient ($\mathcal{E}$ bietet zuviel Funktionalität, z.B. $\mathcal{D}$). Man verwendet daher meist spezielle Kompressionsfunktionen.

Die Kompressionsfunktion $f(K||m) = \mathcal{E}(K,m)$ besitzt nicht die Einwegeigenschaft bei partiell bekanntem Urbild!

Merkle-Damgard Konstruktion

Hashfunktion $h : \{0,1\}^* \to \{0,1\}^n$ aus Kompressionsfunktion bauen.

Sei $f : \{0,1\}^m \to \{0,1\}^n$ eine Kompressionsfunktion und $r = m - n \geq 2$.

Der Hashwert von $x \in \{0,1\}^*$ wird dann wie folgt ausgerechnet:

- x hinten mit beliebigen Bits und der Nachrichtenlänge wie erwähnt padden und in s Blöcke $x_i \in \{0,1\}^r$ mit $0 \leq i \leq s-1$ aufteilen.
- $h_0 = IV$ für einen festen Initialwert (!).
- $h_{i+1} = f(h_i||x_i)$ für $0 \leq i \leq s-1$.
- Der Hashwert ist h_s.

Merkle-Damgard Konstruktion

Thm: Für eine kollisionsresistente Kompressionsfunktion f ist auch die durch die Merkle-Damgard erhaltene Hashfunktion h kollisionsresistent.

Bew: Sei $h(x) = h(x')$ mit $x \neq x'$. Definiere $x_i, x'_i, s, s', h_i, h'_i$ wie in der Konstruktion. Wir können $s \leq s'$ annehmen. Es gilt $h_s = h'_{s'}$. Gilt $(h_{s-j}, x_{s-j}) = (h'_{s'-j}, x'_{s'-j})$ für alle $1 \leq j \leq s$, so folgt $s = s'$ wegen des Paddings und dann $x = x'$ im Widerspruch zur Annahme. Sei also j minimal mit $1 \leq j \leq s$ und $(h_{s-j}, x_{s-j}) \neq (h'_{s'-j}, x'_{s'-j})$. Dann gilt $h_{s-(j-1)} = h'_{s'-(j-1)}$ und $h_{s-(j-1)} = f(h_{s-j}||x_{s-j})$ und $h'_{s'-(j-1)} = f(h'_{s'-j}||x'_{s'-j})$. Somit gibt es eine Kollision der Kompressionsfunktion. Diese kann durch einen Algorithmus „effizient“ gefunden werden, in dem die Merkle-Damgard Konstruktion für x und x' ausgeführt wird. $\square$

Effiziente Hashfunktionen

MD5 (Message Digest 5):

- Von Ron Rivest, 1992. Recht weit verbreitet.
- 128 Bit Hashwerte, ist für heute etwas knapp bemessen.
- Wurde kürzlich gebrochen, Kollisionen können gefunden werden, daher unsicher!

RIPEMD-160:

- Europäisches Design (K. U. Leuven und BSI), 1996.
- 160 Bit Hashwerte (interne Blockgröße 512 Bit).
- Im ISO/IEC 10118-3 standardisiert.
- Ebenfalls unsicher.

Effiziente Hashfunktionen

SHA-1 (Secure Hash Algorithm):

- Von der NSA.
- 160 Bit Hashwerte (interne Blockgröße 512 Bit).
- Am weitesten verbreitete Hashfunktion.
- Im ISO/IEC 10118-3 und FIPS180-1 standardisiert.
- Sicherheit angeknackst, liegt bei ca. 2^{63} anstelle von 2^{80}.

SHA-256, SHA-384, SHA-512:

- Im FIPS180-2.
- 256, 384 und 512 Bit Hashwerte.
- Recht neu (2001).
- Scheinen zur Zeit die einzig sicheren (effizienten) Hashfunktionen zu sein ...

SHA-1

Eingabe x muß Bitlänge $|x| \leq 2^{64} - 1$ haben.

SHA-1 Padding:

- Eingabe x. Setze $d \leftarrow (447 - |x|) \bmod 512$.
- $l \leftarrow$ Binärdarstellung von $|x|$, wobei $|l| = 64$.
- $y \leftarrow x||1||0^d||l$. Ausgabe y.

$\vee$ logisches Oder, $\wedge$ logisches Und, $\bar{\cdot}$ Negation.

80 Funktionen:

$$f_i(B,C,D) = \begin{cases} (B \wedge C) \vee (\bar{B} \wedge D) & \text{für } 0 \leq i \leq 19 \\ B \oplus C \oplus D & \text{für } 20 \leq i \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{für } 40 \leq i \leq 59 \\ B \oplus C \oplus D & \text{für } 60 \leq i \leq 79. \end{cases}$$

SHA-1

80 Konstanten:

$$K_i = \begin{cases} \texttt{5A827999} & \text{für } 0 \leq i \leq 19 \\ \texttt{6ED9EBA1} & \text{für } 20 \leq i \leq 39 \\ \texttt{8F1BBCDC} & \text{für } 40 \leq i \leq 59 \\ \texttt{CA62C1D6} & \text{für } 60 \leq i \leq 79. \end{cases}$$

ROTL^x = zyklischer Shift um x Bits nach Links. + Addition modulo 2^{32}.

Kompressionsfunktion $f : \{0,1\}^{512} \times \{0,1\}^{160} \to \{0,1\}^{160}$:

- Eingabe $M \in \{0,1\}^{512}$, $H \in \{0,1\}^{160}$.
- Schreibe $M = W_0|| \ldots ||W_{15}$ mit $W_i \in \{0,1\}^{32}$.
- Schreibe $H = H_0|| \ldots ||H_4$ mit $H_i \in \{0,1\}^{32}$.
- Für $t \leftarrow 16, \ldots, 79$: $W_t \leftarrow \mathrm{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$.
- $A \leftarrow H_0$, ..., $E \leftarrow H_4$.

SHA-1

- Für $t \leftarrow 0, \ldots, 79$:

  $y \leftarrow \text{ROTL}^5(A) + f_t(B,C,D) + E + W_t + K_t$.

  $E \leftarrow D$, $D \leftarrow C$, $C \leftarrow \text{ROTL}^{30}(B)$.

  $B \leftarrow A$, $A \leftarrow y$.
- $H_0 \leftarrow H_0 + A$, $\ldots$, $H_4 \leftarrow H_4 + E$.
- Ausgabe $H_0 || \ldots || H_4$.

SHA-1:

- Eingabe $x \in \{0,1\}^*$.
- $x \leftarrow$ SHA-1 Padding(x).
- Schreibe $x = M_1 || \ldots || M_n$ mit $M_i \in \{0,1\}^{512}$.
- $H \leftarrow$ `67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0`.
- Für $i \leftarrow 1, \ldots, n$: $H \leftarrow f(M_i, H)$.
- Ausgabe von H.