Sicherheit der Merkle-Damgard Konstruktion

Satz Sicherheit der Merkle-Damgard Konstruktion

Sei (Gen, h) kollisionsresistent. Dann ist auch (Gen, H) kollisionsresistent.

Beweis:

- Sei $\mathcal{A}$ ein Angreifer für H_s mit Erfolgsws $\epsilon(n)$.
- Wir konstruieren einen Angreifer $\mathcal{A}'$ für h_s.

Algorithmus Angreifer $\mathcal{A}'$

EINGABE: s

1. $(x, x') \leftarrow \mathcal{A}(s)$. Sei $x \in \{0,1\}^L$ und $x' \in \{0,1\}^{L'}$. Seien $x_1 \ldots x_B$ und $x'_1 \ldots x'_{B'}$ die mit Nullen erweiterte Darstellung von x, x'.
2. Sei i maximal mit $z_{i-1}||x_i \neq z'_{i-1}||x'_i$ (existiert wegen $x \neq x'$).

   Setze $y := z_{i-1}||x_i$ und $y' := z'_{i-1}||x'_i$.

AUSGABE: (y, y') mit $h_s(y) = h_s(y')$

Sicherheit der Merkle-Damgard Konstruktion

Korrektheit: Zeigen $h_s(y) = h_s(y')$ für $y \neq y'$. Damit folgt

$$\mathrm{negl}(n) \geq \mathrm{Ws}[\mathit{HashColl}_{\mathcal{A}',\Pi_h}(n) = 1] = \mathrm{Ws}[\mathit{HashColl}_{\mathcal{A},\Pi_H}(n) = 1] = \epsilon(n).$$

Fall 1: $L \neq L'$

- Dann gilt $x_{B+1} \neq x'_{B+1}$ und
$$H(x) = z_{B+1} = h_s(\underbrace{z_B||x_{B+1}}_{y}) = h_s(\underbrace{z'_B||x'_{B+1}}_{y'}) = z'_{B+1} = H(x').$$

Fall 2: $L = L'$

- Sei i maximal mit $z_{i-1}||x_i \neq z'_{i-1}||x'_i$.
- Aus der Maximalität von i folgt $z_i = z'_i$.
- Damit gilt $z_i = h_s(\underbrace{z_{i-1}||x_i}_{y}) = h_s(\underbrace{z'_{i-1}||x'_i}_{y'}) = z'_i$.

Hashfunktionen in der Praxis

- Praktische Hashfunktionen verwenden gewöhnlich kein s.
- Damit sind sie im theoretischen Sinne nicht kollisionsresistent, da ein trivialer Angriff existiert, der eine Kollision ausgibt.
- Trotzdem können die besten *bekannten* Angriffe natürlich Komplexität $\Omega(2^{\frac{n}{2}})$ besitzen.
- Fast alle Hashfunktionen verwenden eine Kompressionsfunktion in Kombination mit der Merkle-Damgard Transformation.
- Als kollisionsresistent in der Praxis gelten derzeit z.B. SHA-2, TIGER, Whirlpool, FORK-256.
- Als nicht kollisionsresistent gelten: SHA-0, SHA-1, MD4, MD5, RIPEMD, Snefru, HAVAL, PANAMA, SMASH, etc.
- Kryptanalyse 2004 für SHA-0, SHA-1, MD4, MD5 von Wang et al.
- Derzeit Hash Algorithm Competition der NIST für neuen Standard.

Nested MAC für Nachrichten beliebiger Länge

Idee von NMAC:

- Hashe $m \in \{0,1\}^n$ auf einen Hashwert in $\{0,1\}^n$.
- Verwende MAC Π_h auf dem Hashwert.
- Wir konstruieren Π_h mittels schlüsselabhängiger Hashfunktion.

Algorithmus MAC Π_h fester Länge

Sei (Gen', h) eine kollisionsresistente Hashfkt $h : \{0,1\}^{2n} \to \{0,1\}^n$.

1. **Gen:** $s \leftarrow Gen'(1^n)$. Wähle $k_1 \in_R \{0,1\}^n$.
2. Bei Eingabe (s, k_1) und $m \in \{0,1\}^n$, berechne $t := h_s(k_1||m)$.
3. Bei Eingabe (s, k_1) und $m \in \{0,1\}^n$, verifiziere $t \stackrel{?}{=} h_s(k_1||m)$.

NMAC

Notation: Sei H_s^{IV} eine Merkle-Damgard Hashfunktion, bei der der Initialisierungsvektor auf den Wert IV gesetzt ist.

Algorithmus NMAC (Nested MAC)

Sei (Gen', h), $\Pi_h = (Gen^\Pi, Mac^\Pi, Vrfy^\Pi)$ wie zuvor. Sei (Gen', H) die Merkle-Damgard Transformation von (Gen', h).

1. **Gen:** $s \leftarrow Gen'(1^n)$. Wähle Schlüssel $k_1, k_2 \in_R \{0,1\}^n$.
2. **Mac:** Bei Eingabe (s, k_1, k_2) und $m \in \{0,1\}^n$, berechne
$$t := Mac^\Pi_{s,k_1}(H^{k_2}_s(m)) = h_s(k_1 || H^{k_2}_s(m)).$$
3. **Vrfy:** Bei Eingabe (s, k_1, k_2) und $(m, t) \in \{0,1\}^n \times \{0,1\}^n$,
$$\text{Ausgabe} = \begin{cases} 1 & \text{falls } t = Mac_{s,k_1,k_2}(m) \\ 0 & \text{sonst} \end{cases}.$$

Praxis-Variante: Fixiere s, d.h. einzelne Hash-Funktion (z.B. SHA-1).

Sicherheit von NMAC

Satz Sicherheit von NMAC

Sei (Gen', h) kollisionsresistent und sei Π sicher. Dann ist NMAC sicher.

Beweisskizze:

- Sei $\mathcal{A}$ ein Angreifer für NMAC.
- $\mathcal{A}$ stelle $Mac(\cdot)$ Orakelanfragen aus $Q = \{m_1, \ldots, m_q\}$.
- Anschließend gebe $\mathcal{A}$ gültiges (m, t) aus mit $m \notin Q$.

Fall 1: Es existiert ein $i \in [q]$ mit $H_s^{k_2}(m) = H_s^{k_2}(m_i)$.

- Wegen $m \neq m_i$ ist (m, m_i) eine Kollision für $H_s^{k_2}$.
- Nach Merkle-Damgard Konstruktion liefert dies Kollision für h_s.

Fall 2: Es gilt $H_s^{k_2}(m) \neq H_s^{k_2}(m_i)$ für alle $i \in [q]$.

- Sei $Q' = \{H_s^{k_2}(m) \mid m \in Q\}$. Es gilt $H_s^{k_2}(m) \notin Q'$.
- Damit ist $(H_s^{k_2}(m), t)$ eine gültige Fälschung für Π.

HMAC – Hash-Based MAC

Nachteil von NMAC: Benötigen das Setzen von IV in H.

Idee von HMAC:

- Erzeuge k_1, k_2 durch Vorschalten einer Anwendung von h_s.
- Definieren Konstanten $opad$, $ipad$ und berechnen

$$k_1 = h_s(IV||k \oplus opad) \text{ und } k_2 = h_s(IV||k \oplus ipad).$$

Algorithmus HMAC

Sei (Gen', H) wie zuvor. Seien $opad, ipad \in \{0,1\}^n$ konstant.

1. **Gen:** $s \in Gen'(1^n)$. Wähle $k \in_R \{0,1\}^n$.
2. **Mac:** Für (s,k) und $m \in \{0,1\}^n$ berechne
$$H_s(k \oplus opad||H_s(k \oplus ipad||m)).$$
3. **Vrfy:** Für (s,k) und $(m,t) \in \{0,1\}^{2n}$, verifiziere $t \stackrel{?}{=} Mac_{s,k}(m)$.

Anmerkung:

$$Mac_{s,k}(m) = H_s(k \oplus opad|| \underbrace{H_s(k \oplus ipad||m)}_{H_s^{k_2}(m)}) = H_s^{k_1}(H_s^{k_2}(m)).$$

HMAC ist eine Variante von NMAC

- Wir berechnen beim HMAC den MAC-Wert $H_s^{k_1}(H_s^{k_2}(m))$.
- D.h. die äußere Hashfunktion $H_s^{k_1}$ wird stets auf einen Nachrichtenblock $H_s^{k_2}(m) \in \{0,1\}^n$ fester Länge angewendet.
- Daher ist das Anhängen der Nachrichtenlänge bei $H_s^{k_1}$ unnötig.
- Entspricht der Berechnung von $h_s^{k_1}(H_s^{k_2}(m))$, analog zu NMAC.
- D.h. HMAC ist ein Spezialfall von NMAC, wobei k_1 und k_2 aus k mittels Anwendung von h_s abgeleitet werden.
- Wir definieren den folgenden Pseudozufallsgenerator

$$G(k) = \underbrace{h_s(IV||k \oplus opad)}_{k_1} || \underbrace{h_s(IV||k \oplus ipad)}_{k_2}.$$

Korollar Sicherheit von HMAC mittels Sicherheit von NMAC

Sei G ein Pseudozufallsgenerator, (Gen', h) kollisionsresistent und Π_h sicher. Dann ist die HMAC-Konstruktion sicher.

Praktische Bedeutung von HMAC

Anwendung von HMAC:

- Vorgestellt 1996 von Bellare, Canetti und Krawczyk.
- HMAC wird in der Praxis oft in Kombination mit SHA-1 verwendet.
- HMAC findet Anwendung z.B. in den Protokollen Internet Protocol Security (IPSec) und Transport Layer Security (TLS).
- Wurde 1998 standardisiert und ist weitverbreitet in der Praxis.
- HMAC ist im Vergleich zum CBC-MAC deutlich schneller.