Sicherer MAC für Nachrichten beliebiger Länge

Korollar Sicherer MAC für Nachrichten beliebiger Länge

Sei F eine Pseudozufallsfunktion. Dann ist Π_{MAC2} für $\Pi' = \Pi_{MAC}$ sicher.

Nachteile:

- Für $m \in (\{0,1\}^{\frac{n}{4}})^{\ell}$ sind ℓ Anwendungen von F_k erforderlich.
- Der MAC-Tag $t = (r, t_1 \ldots t_\ell)$ besitzt Länge $(\ell + 1)n$ Bits.

CBC-MAC für Nachrichten fester Länger

Algorithmus CBC-MAC für Nachrichten fester Länger

Sei F eine Pseudozufallsfunktion und $m \in (\{0,1\}^n)^\ell$ für festes ℓ.

1. **Gen:** Wähle $k \in_R \{0,1\}^n$.
2. **Mac:** Für $k \in \{0,1\}^n$ und $m = m_1 \dots m_\ell \in (\{0,1\}^n)^\ell$ setze $t_0 = 0^n$,
$$t_i := F_k(t_{i-1} \oplus m_i) \text{ für } i = 1, \dots, \ell.$$
Ausgabe des Tags $t := t_\ell$.
3. **Vrfy:** Für $k \in \{0,1\}^n$ und $(m,t) \in (\{0,1\}^n)^\ell \times \{0,1\}^n$,
$$\text{Ausgabe} = \begin{cases} 1 & \text{falls } Mac_k(m) = t \\ 0 & \text{sonst} \end{cases}.$$

Anmerkungen:

- Für den CBC-MAC kann Sicherheit für festes ℓ gezeigt werden.
- Tags besitzen nur Länge n, unabhängig von ℓ.
- Konstruktion ist *unsicher* für variables ℓ (Übung).

Vergleich mit CBC Modus bei Verschlüsselung

Rolle des Initialisierungsvektors

- Benötigen zur Sicherheit beim CBC Modus $IV \in_R \{0,1\}^n$.
- Der CBC-MAC verwendet einen festen Wert $IV = t_0 = 0^n$.
- Verwendet man ein zufälliges $t_0 \in_R \{0,1\}^n$ und gibt (t_0, t_ℓ) als Tag aus, so ist dies eine unsichere MAC-Konstruktion! (Übung)

Ausgabe

- CBC Modus: Ausgabe aller c_i, um entschlüsseln zu können.
- Beim CBC-MAC wird nur t_ℓ ausgegeben. Die Werte $t_1, \ldots, t_{\ell-1}$ kann der Verifizierer selbst bestimmen, die MAC-Länge ist nur n.
- Werden $t_1, \ldots, t_{\ell-1}$ ausgegeben, so ist der MAC unsicher! (Übung)

Man beachte:
Scheinbar harmlose Änderungen können drastische Effekte haben.

Sichere CBC-MACs für Nachrichten variabler Länge

Erzeugen längenabhängiger Schlüssel:

- Berechne $k_\ell := F_k(\ell)$ als Schlüssel für Nachrichten der Länge ℓ.
- Verwende k_ℓ als Schlüssel in CBC, d.h. $t_i := F_{k_\ell}(t_{i-1} \oplus m_i)$.
- Wir erhalten einen eigenen Schlüssel k_ℓ für jede feste Länge ℓ.

Anhängen der Länge:

- Verwende $t_0 := F_k(\ell)$ als Initialisierungsvektor.
- Dies ist äquivalent zum Berechnen des Tags von $\ell || m_1 \dots m_\ell$.
- Man beachte: Berechnen des Tags $m_1 \dots m_\ell || \ell$ ist unsicher.

Verwenden zweier Schlüssel:

- Wähle $k_1, k_2 \in_R \{0,1\}^n$. Berechne den Tag $t = F_{k_2}(Mac_{k_1}(m))$.
- Alternativ: Wähle $k_1 := F_k(1)$ und $k_2 := F_k(2)$.
- Vorteil: Mac-Berechnung möglich, ohne ℓ a priori zu kennen.

Hashfunktionen und Kollisionen

Definition Hashfunktion

Eine *Hashfunktion* ist ein Paar (Gen, H) von pt Algorithmen mit

1. **Gen:** $s \leftarrow Gen(1^n)$. *Gen* ist probabilistisch.
2. **H:** H_s berechnet Funktion $\{0,1\}^* \rightarrow \{0,1\}^\ell$. H_s ist deterministisch.

Spiel $HashColl_{\mathcal{A},\Pi}(n)$

1. $s \leftarrow Gen(1^n)$
2. $(x, x') \leftarrow \mathcal{A}(s)$
3. $$HashColl_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & \text{falls } H_s(x) = H_s(x') \text{ und } x \neq x' \\ 0 & \text{sonst} \end{cases}.$$

Definition Kollisionsresistenz

Eine Hashfunktion Π heißt *kollisionsresistent*, falls für alle ppt $\mathcal{A}$ gilt $\mathrm{Ws}[HashColl_{\mathcal{A},\Pi}(n) = 1] \leq \mathrm{negl}(n)$.

Schwächere Sicherheitskonzepte

2.Urbild Resistenz

Gegeben: s, x
Gesucht: $x' \neq x$ mit $H_s(x') = H_s(x)$

Satz Kollisionresistenz impliziert 2.Urbild Resistenz

Sei Π kollisionsresistent. Dann ist Π 2.Urbild resistent.

Beweis:

- Sei $\mathcal{A}$ ein 2.Urbild Angreifer auf $\Pi = (Gen, H)$ mit Erfolgsws $\epsilon(n)$.

Algorithmus Angreifer $\mathcal{A}'$ auf Kollisionsresistenz

EINGABE: s

1. Wähle $x \in \{0,1\}^*$.
2. $x' \leftarrow \mathcal{A}(s, x)$

AUSGABE: x, x'

- Offenbar gilt $\mathrm{Ws}[HashColl_{\mathcal{A}',\Pi}] = \epsilon(n)$

Schwächere Sicherheitskonzepte

Urbild Resistenz

Gegeben: $s, y = H_s(x)$
Gesucht: x' mit $H_s(x') = y$

Satz 2.Urbild Resistenz impliziert Urbild Resistenz

Sei Π 2.Urbild resistent und komprimierend. Dann ist Π Urbild resistent

Beweisskizze: Sei $\mathcal{A}$ ein Urbild Angreifer auf Π mit Erfolgsws ϵ.

Algorithmus Angreifer $\mathcal{A}'$ auf 2.Urbild

EINGABE: s, x

1. Berechne $y = H_s(x)$.
2. $x' \leftarrow \mathcal{A}(s, y)$

AUSGABE: x, x' falls $x \neq x'$

- Es gilt $x \neq x'$ mit signifikanter Ws, falls H seine Eingabe komprimiert, d.h. der Urbildraum ist größer als der Bildraum.

Damit ist Kollisionsresistenz der *stärkste Sicherheitsbegriff*.

Geburtstagsangriff auf Hashfunktionen

Algorithmus Geburtstagsangriff

EINGABE: s mit $H_s : \{0,1\}^* \to \{0,1\}^\ell$

1. Wähle verschiedene $x_1, \ldots, x_q \in \{0,1\}^*$ für geeignetes q.
2. Berechne $y_i = H_s(x_i)$ für $i = 1, \ldots, q$ und sortiere die y_i.
3. Finde in der sortierten Liste x_i, x_j mit $y_i = y_j$.

AUSGABE: x_i, x_j mit $H_s(x_i) = H_s(x_j)$

Anmerkungen:

- Annahme: y_i sind zufällig gleichverteilt in $\{0,1\}^\ell$.
- Geburtstagsproblem: Für $q = 2^{\frac{\ell}{2}} + 1$ erhalten wir mit Ws mind $1 - e^{-\frac{1}{2}}$ eine Kollision $y_i = y_j$ in Schritt 3. (Übung)
- Die Auswertung von H_s koste konstante Laufzeit $\mathcal{O}(1)$.
- Dann besitzt der Algorithmus Laufzeit $\mathcal{O}(q \log q) = \mathcal{O}(\ell \cdot 2^{\frac{\ell}{2}})$.

Konsequenz für Hashfunktionen:

- Wir benötigen mindestens Ausgabelänge $\ell = 128$ Bit.

Merkle-Damgard Transformation

Ziel: Konstruiere $H : \{0,1\}^* \to \{0,1\}^\ell$ aus $h : \{0,1\}^{2\ell} \to \{0,1\}^\ell$.

Algorithmus Merkle-Damgard Konstruktion

Sei (Gen, h) eine kollisionsresistente Hashfunktion mit $h : \{0,1\}^{2\ell} \to \{0,1\}^\ell$. Wir konstruieren (Gen, H) wie folgt.

1. **Gen:** $s \leftarrow Gen(1^n)$.
2. **H:** Bei Eingabe s und $x \in \{0,1\}^L$:
   - Erweitere x mit Nullen bis die Länge ein Vielfaches von ℓ ist.
   - Schreibe $x = x_1 \ldots x_B$ mit $x_i \in \{0,1\}^\ell$ und $B = \lceil \frac{L}{\ell} \rceil$.
   - Setze $x_{B+1} = L$ (binär kodiert). Initialisiere $z_0 := 0^\ell$, berechne
     $$z_i := h_s(z_{i-1} || x_i) \text{ für } i = 1, \ldots, B+1.$$

   Ausgabe des Hashwerts $H_s(x) := z_{B+1}$.