

## Sicherheit von $\Pi_{MAC2}$

Wir definieren die folgenden Ereignisse

- *Forge*: Ein Block  $r || \ell || i || m_i$  in  $(m, t)$  wurde nicht angefragt.
- *Repeat*: Bei 2 MAC-Anfragen wird dasselbe  $r_i = r_j$  verwendet.

Aufgrund der Sicherheit von  $\Pi'$  gilt

$$\begin{aligned} \text{negl}(n) &\geq \text{Ws}[\text{Mac-forge}_{\mathcal{A}', \Pi'}(n) = 1] \\ &= \text{Ws}[\text{Mac-forge}_{\mathcal{A}, \Pi_{MAC2}}(n) = 1 \cap \text{Forge}] \\ &= \epsilon(n) - \text{Ws}[\text{Mac-forge}_{\mathcal{A}, \Pi_{MAC2}}(n) = 1 \cap \overline{\text{Forge}}] \\ &= \epsilon(n) - \underbrace{\text{Ws}[\text{Mac-forge}_{\mathcal{A}, \Pi_{MAC2}}(n) = 1 \cap \overline{\text{Forge}} \cap \overline{\text{Repeat}}]}_{\epsilon_1} \\ &\quad - \underbrace{\text{Ws}[\text{Mac-forge}_{\mathcal{A}, \Pi_{MAC2}}(n) = 1 \cap \overline{\text{Forge}} \cap \text{Repeat}]}_{\leq \text{Ws}[\text{Repeat}]} \end{aligned}$$

Zeigen nun  $\epsilon_1 = 0$  und  $\text{Ws}[\text{Repeat}] \leq \text{negl}(n)$ . Damit  $\epsilon(n) \leq \text{negl}(n)$ .

# Sicherheit von $\Pi_{MAC2}$

zu zeigen:  $Ws[Repeat] \leq \text{negl}(n)$

- Sei  $q(n)$  die Anzahl der MAC-Anfragen von  $\mathcal{A}$  an  $\mathcal{A}'$ .
- Bei der  $i$ -ten MAC-Anfrage wähle  $\mathcal{A}'$  den Identifikator  $r_i$ .
- *Repeat* tritt ein, falls  $r_i = r_j$  für ein  $i \neq j$ . Sei dies Ereignis  $E_{i,j}$ .
- Nach Geburtstagsparadoxon gilt:

$$\begin{aligned} Ws[Repeat] &= Ws\left[\bigcup_{1 \leq i < j \leq q(n)} E_{i,j}\right] \leq \sum_{1 \leq i < j \leq q(n)} Ws[E_{i,j}] \\ &\leq \frac{q(n)^2}{2^{\frac{n}{4}}} = \text{negl}(n). \end{aligned}$$

## Sicherheit von $\Pi_{MAC2}$

zu zeigen:  $Ws[\text{Mac-forge}_{\mathcal{A}, \Pi_{MAC2}}(n) = 1 \cap \overline{\text{Forge}} \cap \overline{\text{Repeat}}] = 0$

- **Idee:**  $\text{Mac-forge}_{\mathcal{A}, \Pi_{MAC2}}(n) = 1$  und  $\overline{\text{Repeat}}$  impliziert  $\text{Forge}$ .
- Sei  $(m, t) = (m_1 \dots m_\ell, r, t_1 \dots t_\ell)$  die Ausgabe von  $\mathcal{A}$ .

**Fall 1:** Identifikator  $r$  unterscheidet sich von allen  $r_j$ .

- Dann ist  $r||\ell||1||m_1$  nicht-angefragt mit gültigem Tag  $t_1$ .

**Fall 2:**  $r = r_i$  für genau ein  $i \in [q(n)]$ .

- Sei  $m^{(i)} = m'_1 \dots m'_{\ell'}$  die von  $\mathcal{A}$  angefragte Nachricht.
- Fall  $\ell \neq \ell'$ : Dann ist  $r||\ell||1||m_1$  nicht-angefragt mit gültigem Tag  $t_1$ .
- Fall  $\ell = \ell'$ : Wegen  $m \notin Q$  gilt  $m \neq m^{(i)}$ .
- D.h. es existiert ein  $j$ , so dass  $m_j \neq m'_j$ .
- Damit wurde  $r||\ell||j||m_j$  nicht angefragt. Tag  $t_j$  ist dafür gültig.

**Fall 3:**  $r = r_i$  für mehrere  $i \in [q(n)]$ .

- Fall ist ausgeschlossen, da wegen  $\overline{\text{Repeat}}$  gilt  $r_i \neq r_j$  für alle  $i \neq j$ .

# Sicherer MAC für Nachrichten beliebiger Länge

## **Korollar** Sicherer MAC für Nachrichten beliebiger Länge

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

### **Nachteile:**

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

# CBC-MAC für Nachrichten fester Längen

## Algorithmus CBC-MAC für Nachrichten fester Längen

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

- 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  $t_0 = 0^n$ ,  
 $t_i := F_k(t_{i-1} \oplus m_i)$  für  $i = 1, \dots, \ell$ .

Ausgabe des Tags  $t := t_\ell$ .

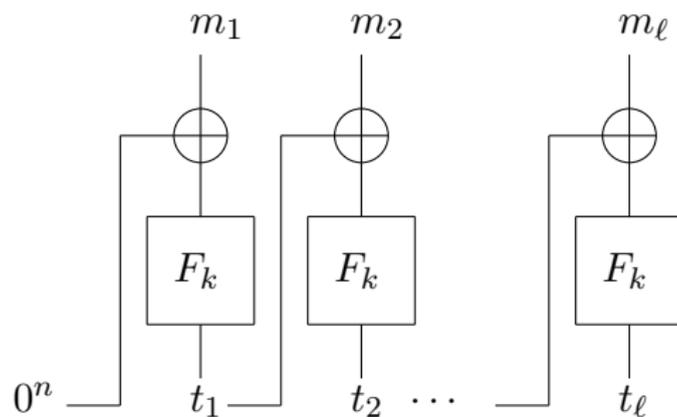
- 3 **Vrfy:** Für  $k \in \{0, 1\}^n$  und  $(m, t) \in (\{0, 1\}^n)^\ell \times \{0, 1\}^n$ ,

$$\text{Ausgabe} = \begin{cases} 1 & \text{falls } \text{Mac}_k(m) = t \\ 0 & \text{sonst} \end{cases}.$$

### Anmerkungen:

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

# CBC Mac



# 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  $(t_0, t_\ell)$  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  $t_\ell$  ausgegeben. Die Werte  $t_1, \dots, t_{\ell-1}$  kann der Verifizierer selbst bestimmen, die MAC-Länge ist nur  $n$ .
- Werden  $t_1, \dots, t_{\ell-1}$  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  $\ell$ .
- Verwende  $k_\ell$  als Schlüssel in CBC, d.h.  $t_i := F_{k_\ell}(t_{i-1} \oplus m_i)$ .
- Wir erhalten einen eigenen Schlüssel  $k_\ell$  für jede feste Länge  $\ell$ .

## 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  $\ell$  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.

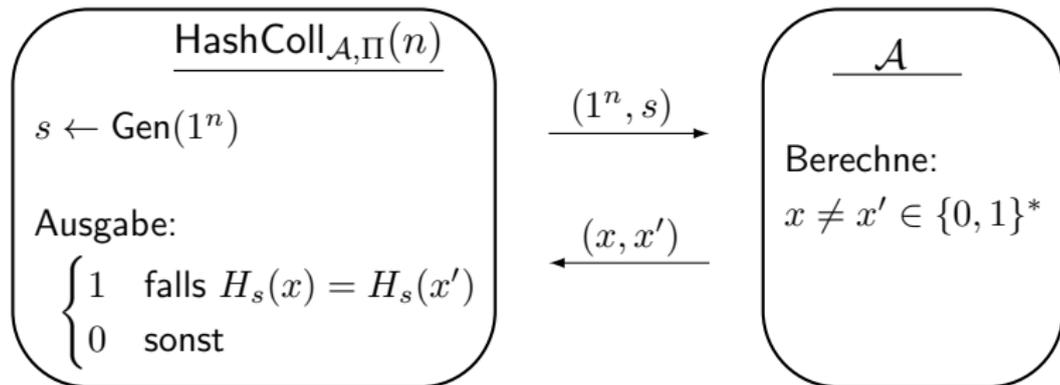
## Spiel $HashColl_{\mathcal{A}, \Pi}(n)$

- 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  $\Pi$  heißt *kollisionsresistent*, falls für alle ppt  $\mathcal{A}$  gilt  $\mathbb{W}_s[HashColl_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n)$ .

# Spiel HashColl



# Schwächere Sicherheitskonzepte

## 2.Urbild Resistenz

Gegeben:  $s, x$

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

### Satz Kollisionsresistenz impliziert 2.Urbild Resistenz

Sei  $\Pi$  kollisionsresistent. Dann ist  $\Pi$  2.Urbild resistent.

#### Beweis:

- Sei  $\mathcal{A}$  ein 2.Urbild Angreifer auf  $\Pi = (\text{Gen}, H)$  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  $\text{Ws}[\text{HashColl}_{\mathcal{A}', \Pi}] = \epsilon(n)$

# 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  $\Pi$  2.Urbild resistent und komprimierend. Dann ist  $\Pi$  Urbild resistent

**Beweisskizze:** Sei  $\mathcal{A}$  ein Urbild Angreifer auf  $\Pi$  mit Erfolgsws  $\epsilon$ .

## 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  $H_s : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$

- 1 Wähle verschiedene  $x_1, \dots, x_q \in \{0, 1\}^*$  für geeignetes  $q$ .
- 2 Berechne  $y_i = H_s(x_i)$  für  $i = 1, \dots, q$  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.