

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.

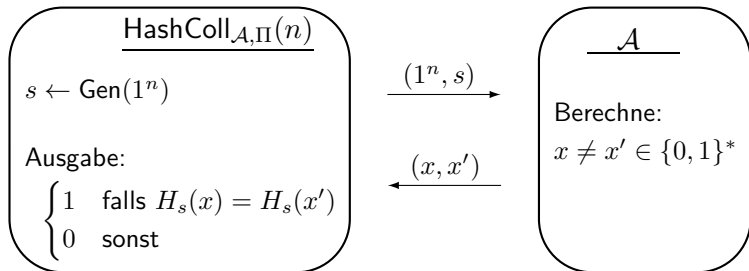
Spiel $HashColl_{\mathcal{A}, \Pi}(n)$

- 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 $\mathbb{W}_s[HashColl_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n)$.

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 Π kollisionsresistent. Dann ist Π 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 $\text{Ws}[\text{HashColl}_{\mathcal{A}', \Pi}(n) = 1] = \epsilon(n)$

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

- 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 $\ell = 160$ Bit.

Merkle-Damgard Transformation

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

Algorithmus Merkle-Damgard Konstruktion

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

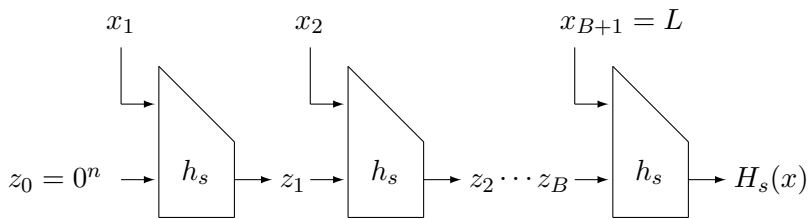
1 **Gen:** $s \leftarrow Gen(1^n)$.

2 **H:** Bei Eingabe s und $x \in \{0, 1\}^L$:

- ▶ Erweitere x mit Nullen, bis die Länge ein Vielfaches von ℓ ist.
- ▶ Schreibe $x = x_1 \dots x_B$ 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
$$z_i := h_s(z_{i-1} || x_i) \text{ f\"ur } i = 1, \dots, B + 1.$$

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

Merkle Damgard Konstruktion



Sicherheit der Merkle-Damgard Konstruktion

Satz Sicherheit der Merkle-Damgard Konstruktion

Sei $\Pi_h = (\text{Gen}, h)$ kollisionsresistent. Dann ist auch $\Pi_H = (\text{Gen}, H)$ 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 $x_1 \dots x_B$ und $x'_1 \dots x'_C$ die mit Nullen erweiterte Darstellung von x, x' .
- 2 Falls $L \neq L'$, setze $y := z_B || x_{B+1}$ und $y' := z'_C || x'_{C+1}$.
- 3 Sonst 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: Wir zeigen $h_s(y) = h_s(y')$ für $y \neq y'$. Damit folgt

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

Fall 1: $L \neq L'$

- Dann gilt $x_{B+1} \neq x'_{B+1}$ und

$$H_s(x) = z_{B+1} = h_s(\underbrace{z_B || x_{B+1}}_y) = h_s(\underbrace{z'_C || x'_{C+1}}_{y'}) = z'_{C+1} = H_s(x').$$

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 *bekanntesten* 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.
- Seit 2008 Hash Algorithm Competition für neuen NIST-Standard.
- Finalisten (Dez 2010): BLAKE, Grøstl, JH, Keccak, Skein.
- Sieger (Okt 2012): Keccak, genannt SHA-3.