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 $\{0,1\}^* \to \{0,1\}^\ell$. H_s ist deterministisch.

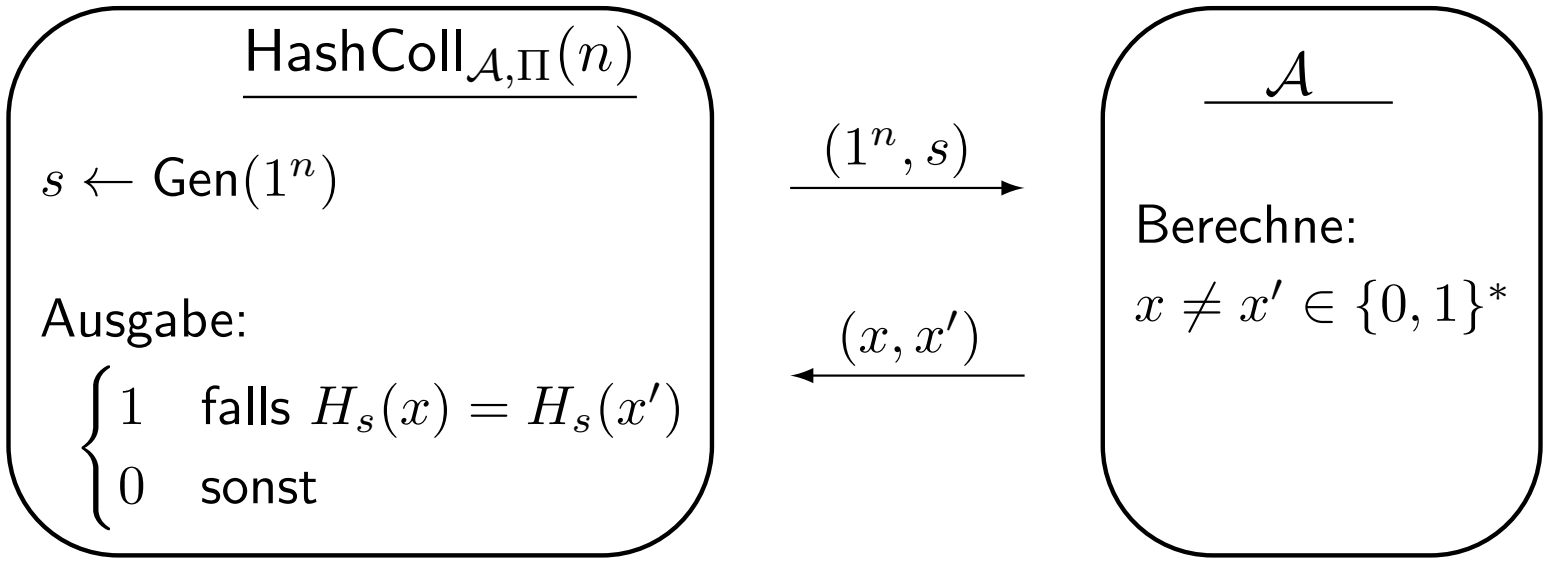
Spiel $HashColl_{\mathcal{A},\Pi}(n)$

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 $\mathrm{Ws}[HashColl_{\mathcal{A},\Pi}(n) = 1] \leq \mathrm{negl}(n)$.

Spiel HashColl



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 $\ell = 160$ Bit.

Merkle-Damgard Transformation

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

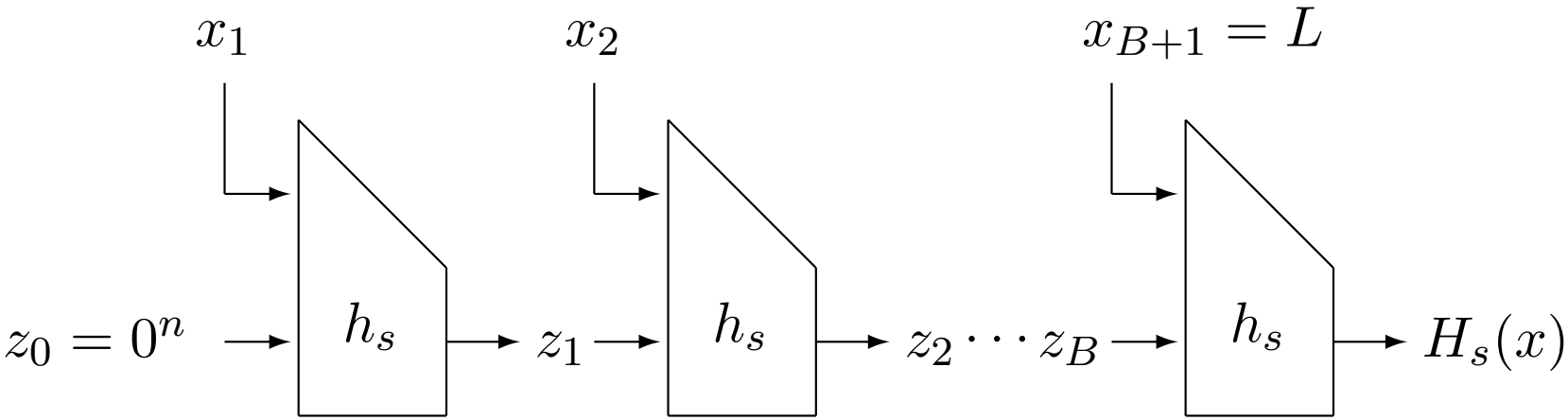
Algorithmus Merkle-Damgard Konstruktion

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

Merkle Damgard Konstruktion



Sicherheit der Merkle-Damgard Konstruktion

Satz Sicherheit der Merkle-Damgard Konstruktion

Sei $\Pi_h = (Gen, h)$ kollisionsresistent. Dann ist auch $\Pi_H = (Gen, H)$ 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 $x'_1 \dots x'_{B'}$ die mit Nullen erweiterte Darstellung von x, x'.
2. Falls $L \neq L'$, setze $y := z_B||x_{B+1} = z_b||L$, $y' := z'_{B'}||x'_{B'+1} = z'_{B'}||L'$
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

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

Fall 1: $L \neq L'$

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

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 *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.
- Seit 2008 Hash Algorithm Competition für neuen NIST-Standard.
- Finalisten (Dez 2010): BLAKE, Grøstl, JH, Keccak, Skein.

Effizienten MAC-Konstruktion mittels Hashfunktionen

Idee von NMAC:

- Hashe $m \in \{0,1\}^*$ auf einen Hashwert in $\{0,1\}^n$.
- Verwende Π_{MAC3} für Nachrichten fixer Länge auf dem Hashwert.
- Wir konstruieren Π_{MAC3} mittels schlüsselabhängiger Hashfunktion, bei der ein Teil des Hasharguments aus dem Schlüssel besteht.

Algorithmus MAC Π_{MAC3} für Nachrichten fester Länge n

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

1. **Gen:** $s \leftarrow Gen_h(1^n)$, s kann öffentlich sein. Wähle $k_1 \in_R \{0,1\}^n$.
2. **Mac:** Bei Eingabe (s, k_1) und $m \in \{0,1\}^n$, berechne
$$t := h_s(k_1||m).$$
3. **Vrfy:** Bei Eingabe (s, k_1) und $(m, t) \in \{0,1\}^n \times \{0,1\}^n$, 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.

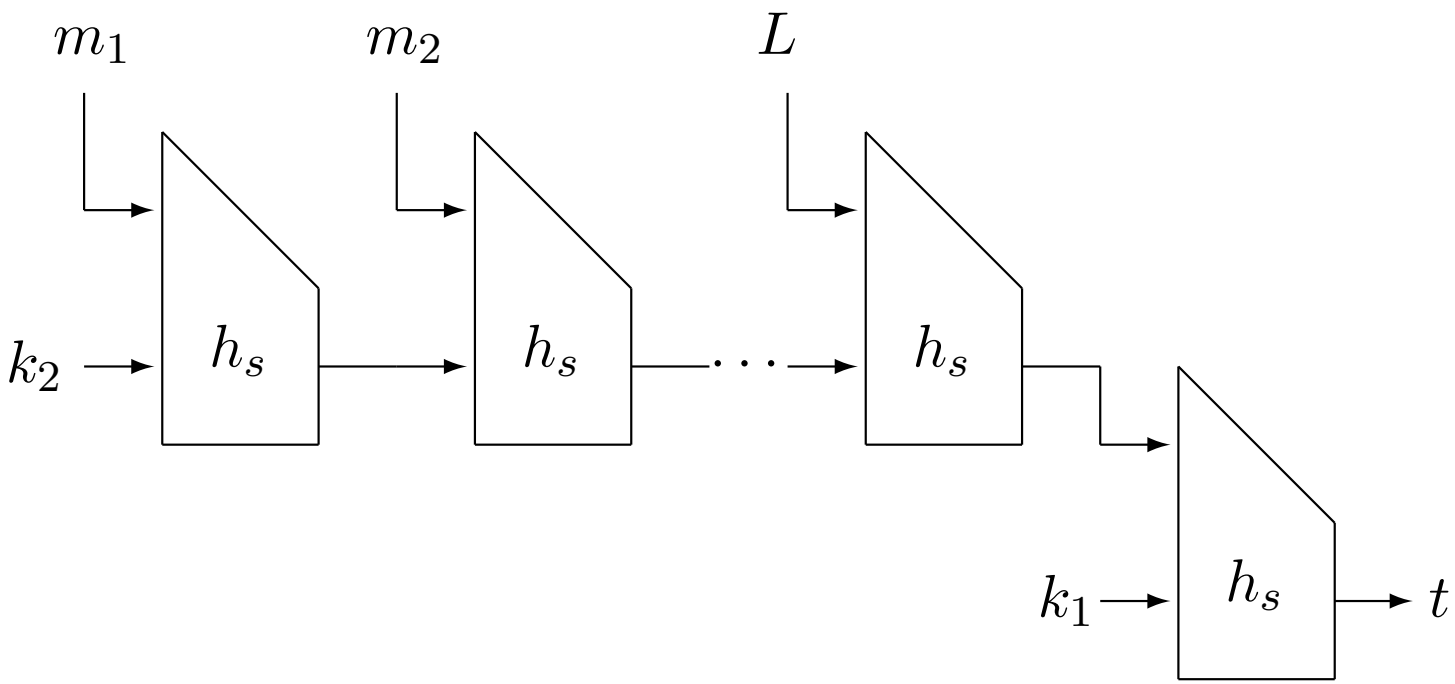
Algorithmus NMAC (Nested MAC)

Sei $\Pi_h = (Gen_h, h)$, $\Pi_{MAC3} = (Gen', Mac', Vrfy')$ wie zuvor. Sei (Gen_h, H) die Merkle-Damgard Transformation von (Gen_h, h).

1. **Gen:** $s \leftarrow Gen_h(1^n)$. 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
$$t := Mac'_{s,k_1}(H_s^{k_2}(m)) = h_s(k_1 || H_s^{k_2}(m)).$$
3. **Vrfy:** Bei Eingabe (s, k_1, k_2) und $(m, t) \in \{0,1\}^* \times \{0,1\}^n$,
$$\text{Ausgabe} = \begin{cases} 1 & \text{falls } t = Mac_{s,k_1,k_2}(m) \\ 0 & \text{sonst} \end{cases}.$$

Praxis-Variante: Fixiere s, d.h. einzelne Hash-Funktion (z.B. SHA-1).
Anmerkung: Wir können auch $k_2 = 0^n$ setzen. Vorteil von Schlüssel k_2: Sicherheit kann auch unter schwächerer Annahme gezeigt werden.

NMAC



Sicherheit von NMAC

Satz Sicherheit von NMAC

Sei $\Pi_h = (Gen_h, h)$ kollisionsresistent und sei Π_{MAC3} sicher. Dann ist auch 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 $j \in [q]$ mit $H_s^{k_2}(m) = H_s^{k_2}(m_j)$.

- Wegen $m \neq m_j$ ist (m, m_j) 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_{MAC3}.

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

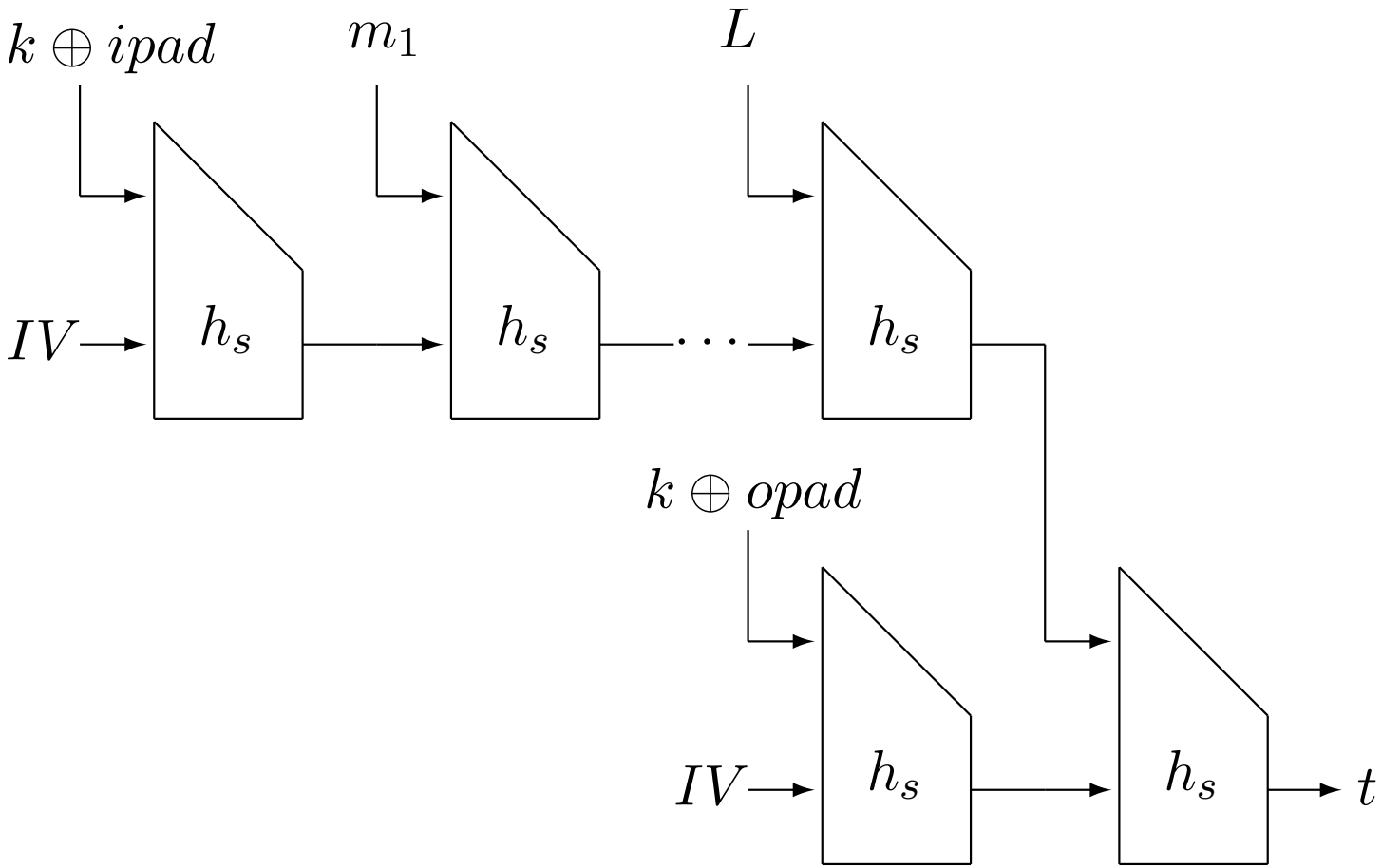
Sei (Gen_h, H) wie zuvor. Seien $opad, ipad \in \{0,1\}^n$ konstant.

1. **Gen:** $s \in Gen_h(1^n)$. Wähle $k \in_R \{0,1\}^n$.
2. **Mac:** Für (s,k) und $m \in \{0,1\}^*$ berechne
$$Mac_{s,k}(m) = H_s(k \oplus opad||H_s(k \oplus ipad||m)).$$
3. **Vrfy:** Für (s,k) und $(m,t) \in \{0,1\}^* \times \{0,1\}^n$, verifiziere
$$t \stackrel{?}{=} Mac_{s,k}(m).$$

Anmerkung:

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

HMAC



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_{MAC3} 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.