

**Alle Angaben ohne Gewähr. Keine Garantie auf Vollständigkeit oder Richtigkeit.**

<b>1</b>	<b>Einführung</b>	<b>2</b>
1.1	Ziel von Kryptographischen Verfahren	2
1.2	Informelle Definition von Signaturen	2
1.3	Digitale Signaturen	2
1.3.1	Definition	2
1.3.2	Correctness	2
1.4	Sicherheitsdefinitionen	2
1.4.1	Angreifermodelle	2
1.4.2	Angreiferziele	2
1.5	EUFCMA-Sicherheitsexperiment	3
1.5.1	Visualisierung: EUFCMA-Sicherheitsexperiment	3
1.5.2	Definition: Vernachlässigbarkeit	3
1.5.3	Definition: EUFCMA	3
1.6	EUFCMA-Sicherheitsexperiment	4
1.6.1	Visualisierung: EUFCMA-Sicherheitsexperiment	4
1.6.2	Definition: EUFCMA	4
1.7	Einmalsignaturen	4
1.7.1	Sicherheitsbegriffe für Einmalsignaturen	4
1.7.2	Beziehungen zwischen Sicherheitsdefinitionen	4
1.8	Perfekte Sicherheit	5
1.8.1	Warum müssen wir uns auf PPT-Angreifer beschränken?	5
1.8.2	Warum muss die Erfolgswahrscheinlichkeit des Angreifers nur vernachlässigbar sein?	5
1.9	Erweiterung des Nachrichtenraumes	5
1.9.1	Hashfunktionen	5
1.9.2	Kollisionsresistenz	5
1.9.3	Signatur mit unbeschränktem Nachrichtenraum ( <b>Hash-then-Sign</b> )	5
<b>2</b>	<b>q-mal Signaturen</b>	<b>6</b>
2.1	Von EUFCMA-Sicherheit zu EUFCMA-Sicherheit	6
2.1.1	Transformation	6
2.2	Mehrmal-Signaturverfahren aus Einmalsignaturverfahren	6
2.2.1	Naiver Ansatz: q Schlüsselpaare	6
2.2.2	Zwischenschritt: Hashfunktion verwenden	7
2.2.3	Merkle-Bäume	8
2.3	Komprimieren des geheimen Schlüssels	9
2.3.1	Pseudozufallsfunktion	9
2.3.2	Schlüsselgenerierung	9

## 1 Einführung

### 1.1 Ziel von Kryptographischen Verfahren

Kryptographische Verfahren sollen **Authentizität** (Dokument wurde von einer bestimmten Person signiert) und **Integrität** (Dokument wurde nicht verändert) sicherstellen.

### 1.2 Informelle Definition von Signaturen

- **asymmetrische** Verfahren
- Schlüsselpaar  $(pk, sk)$
- Nachricht  $m$  wird mit  $sk$  signiert und erzeugt Signatur  $\sigma$
- Mit  $pk$  kann überprüft werden, ob eine Signatur  $\sigma$  gültig für eine Nachricht  $m$  ist

### 1.3 Digitale Signaturen

#### 1.3.1 Definition

Ein digitales Signaturverfahren für einen Nachrichtenraum  $\mathcal{M}$  ist ein Tupel  $\Sigma = (Gen, Sign, Vfy)$  von probabilistischen Polyzeit (PPT) Algorithmen:

- $Gen(1^k) \rightarrow (pk, sk)$
- $Sign(sk, m) \rightarrow \sigma, m \in \mathcal{M}$
- $Vfy(pk, m, \sigma) \in \{0, 1\}$

#### 1.3.2 Correctness

**Correctness** ("Das Verfahren funktioniert"):  $\forall (pk, sk) \leftarrow Gen(1^k) \forall m \in \mathcal{M} : Vfy(pk, m, Sign(sk, m)) = 1$

### 1.4 Sicherheitsdefinitionen

Sicherheit besteht aus einem **Angreifermodell** (was kann der Angreifer tun, welche Angriffsmöglichkeiten stehen zur Verfügung) und einem **Angreiferziel** (was muss der Angreifer tun, um das Verfahren zu brechen).

#### 1.4.1 Angreifermodelle

1. no-message attack (NMA)
  - Angreifer erhält nur  $pk$
2. **non-adaptive chosen-message attack (naCMA)**
  - Angreifer wählt  $m_1, \dots, m_q$
  - Angreifer erhält **danach**  $pk$  und Signaturen  $\sigma_1, \dots, \sigma_q$
3. **(adaptive) chosen-message attack (CMA)**
  - Angreifer erhält  $pk$
  - Angreifer wählt dann (adaptiv)  $m_1, \dots, m_q$  und erhält Signaturen  $\sigma_1, \dots, \sigma_q$
  - Adaptiv: Angreifer darf Wahl von  $m_i$  abhängig von vorherigen  $\sigma_j$  ( $j < i$ ) und  $pk$  machen

#### 1.4.2 Angreiferziele

1. Universal Unforgeability (UUF)
  - Nachricht  $m$  wird zufällig gewählt
  - Angreifer muss  $m$  signieren

## 2. Existential Unforgeability (EUF)

- Angreifer kann Nachricht  $m$  beliebig wählen und diese signieren

In den **Sicherheitsdefinitionen** werden **Angreiferziel** und **Angreifermodell** kombiniert, z.B.

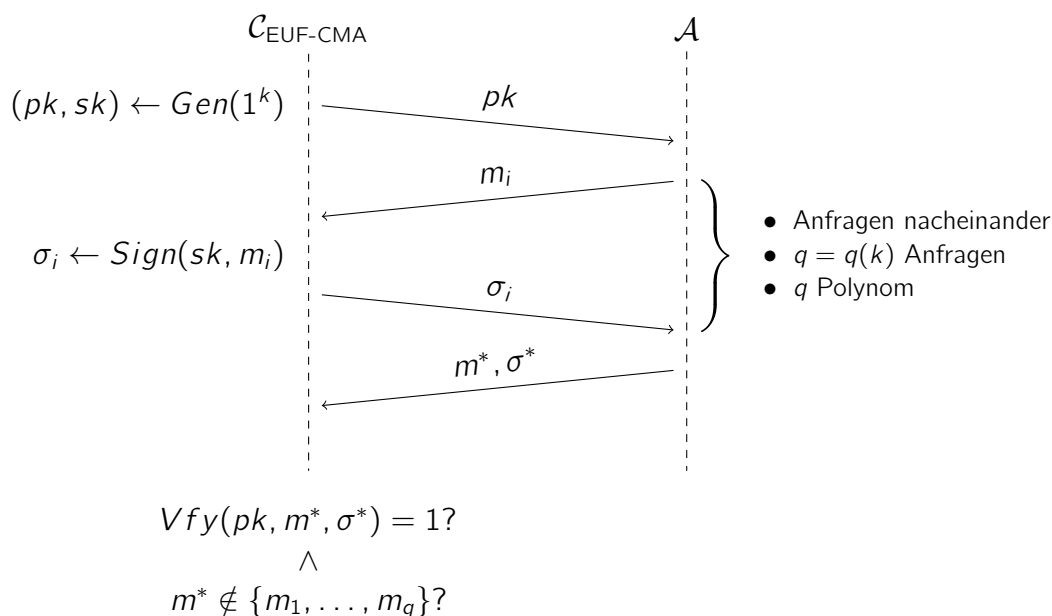
- EUF-CMA
- EUF-naCMA

## 1.5 EUF-CMA-Sicherheitsexperiment

Bei Sicherheitsexperimenten spielt ein Angreifer  $\mathcal{A}$  gegen einen Challenger  $\mathcal{C}$ .  $\mathcal{A}$  gewinnt, falls er die Sicherheit des Verfahrens bricht.

$\mathcal{A}$  muss dabei mit einer nicht vernachlässigbaren Wahrscheinlichkeit eine gültige Signatur erzeugen können, ohne den Schlüssel  $sk$  zu kennen.

### 1.5.1 Visualisierung: EUF-CMA-Sicherheitsexperiment



$\mathcal{A}$  gewinnt, falls  $Vfy(pk, m^*, \sigma^*) = 1$  **und**  $m^* \notin \{m_1, \dots, m_q\}$

### 1.5.2 Definition: Vernachlässigbarkeit

Eine Funktion  $\text{negl} : \mathbb{N} \rightarrow [0, 1]$  ist *vernachlässigbar*, wenn

$$\forall c \in \mathbb{N} \exists k_0 \in \mathbb{N} \forall k \geq k_0 : \text{negl}(k) < \frac{1}{k^c}$$

### 1.5.3 Definition: EUF-CMA

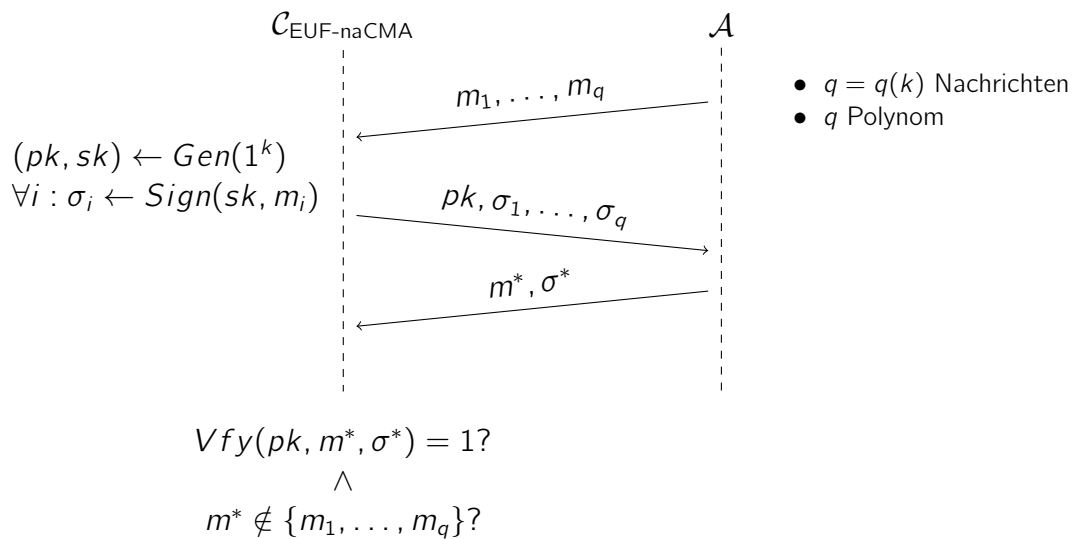
Ein digitales Signaturverfahren  $\Sigma = (\text{Gen}, \text{Sign}, \text{Vfy})$  ist *EUF-CMA-sicher*, wenn für alle PPT  $\mathcal{A}$  gilt, dass

$$\begin{aligned}
 & \Pr[\mathcal{A} \text{ gewinnt EUF-CMA-Experiment}] \\
 &= \Pr[\mathcal{A}^{\mathcal{C}_{\text{EUF-CMA}}}(pk) = (m^*, \sigma^*) : Vfy(pk, m^*, \sigma^*) = 1 \wedge m^* \notin \{m_1, \dots, m_q\}] \\
 &\leq \text{negl}(k)
 \end{aligned}$$

für eine im Sicherheitsparameter  $k$  vernachlässigbare Funktion  $\text{negl}$ .

## 1.6 EUF-naCMA-Sicherheitsexperiment

### 1.6.1 Visualisierung: EUF-naCMA-Sicherheitsexperiment



$\mathcal{A}$  gewinnt, falls  $Vfy(pk, m^*, \sigma^*) = 1$  **und**  $m^* \notin \{m_1, \dots, m_q\}$

### 1.6.2 Definition: EUF-naCMA

Ein digitales Signaturverfahren  $\Sigma = (\text{Gen}, \text{Sign}, \text{Vfy})$  ist *EUF-naCMA-sicher*, wenn für alle PPT  $\mathcal{A}$  gilt, dass

$$\begin{aligned}
 & \Pr[\mathcal{A} \text{ gewinnt EUF-naCMA-Experiment}] \\
 &= \Pr[\mathcal{A}^{\mathcal{C}_{\text{EUF-naCMA}}} = (m^*, \sigma^*) : Vfy(pk, m^*, \sigma^*) = 1 \wedge m^* \notin \{m_1, \dots, m_q\}] \\
 &\leq \text{negl}(k)
 \end{aligned}$$

für eine im Sicherheitsparameter  $k$  vernachlässigbare Funktion *negl*.

## 1.7 Einmalsignaturen

- Ziel: Signaturen, die viele Nachrichten signieren können
- Vorstufe: Signaturen, die nur **eine** Nachricht **sicher** signieren können (**Einmalsignaturen**)
- für jeden *public key* sollte nur eine einzige Signatur ausgestellt werden, sonst evtl. unsicher

### 1.7.1 Sicherheitsbegriffe für Einmalsignaturen

Analog zum vorherigen Kapitel definieren wir **EUF-1-CMA** und **EUF-1-naCMA** für Einmalsignaturen.

### 1.7.2 Beziehungen zwischen Sicherheitsdefinitionen

EUF-naCMA  $\Leftarrow$  EUF-CMA

$\Downarrow$

$\Downarrow$

EUF-1-naCMA  $\Leftarrow$  EUF-1-CMA

Beweis im Skript.

## 1.8 Perfekte Sicherheit

In den Definitionen, z.B. bei EUF-CMA finden sich zwei Einschränkungen, die im folgenden erläutert werden:

### 1.8.1 Warum müssen wir uns auf PPT-Angreifer beschränken?

Durch Brute-Force könnte ein unbeschränkter Angreifer alle Signaturen durchprobieren und so valide Signaturen für beliebige Nachrichten finden, wodurch er beim Sicherheitsexperiment immer gewinnen würde.

### 1.8.2 Warum muss die Erfolgswahrscheinlichkeit des Angreifers nur vernachlässigbar sein?

Die Erfolgswahrscheinlichkeit kann nicht 0 sein, da der Angreifer durch zufälliges Raten eine gültige Signatur für eine beliebige Nachricht finden könnte, wodurch er das Sicherheitsexperiment gewinnt.

## 1.9 Erweiterung des Nachrichtenraumes

Wir konstruieren fast immer Signaturen mit "kleinem" Nachrichtenraum, z.B.

- $\mathbb{Z}_p = \{0, \dots, p-1\}$ ,  $p$  prim
- $\{0, 1\}^{q(k)}$ ,  $q$  Polynom,  $k$  Sicherheitsparameter

Unser Ziel ist es jedoch, beliebige Nachrichten, z.B.  $\{0, 1\}^*$ , zu signieren.

### 1.9.1 Hashfunktionen

Eine kryptographische Hashfunktion  $H = (Gen_H, Eval_H)$  ist ein Tupel aus zwei PPT-Algorithmen:

- $Gen_H(1^k)$  berechnet  $t$ , sodass  $t$  eine Funktion

$$H_t : \{0, 1\}^* \rightarrow \mathcal{M}_t$$

spezifiziert

- $Eval_H(1^k, t, x)$  berechnet  $H_t(x)$

### 1.9.2 Kollisionsresistenz

Eine Hashfunktion  $H = (Gen_H, Eval_H)$  ist **kollisionsresistent**, falls für alle  $t \leftarrow Gen_H(1^k)$  und für alle PPT  $\mathcal{A}$  gilt, dass

$$\Pr[\mathcal{A}(1^k, t) = (x, x') : H_t(x) = H_t(x') \wedge x \neq x'] \leq \text{negl}(k)$$

für eine im Sicherheitsparameter  $k$  vernachlässigbare Funktion  $\text{negl}$ .

### 1.9.3 Signatur mit unbeschränktem Nachrichtenraum (Hash-then-Sign)

Wir wollen nun Signaturen mit unbeschränktem Nachrichtenraum konstruieren. Gegeben:

- $\Sigma' = (Gen', Sign', Vfy')$  mit Nachrichtenraum  $\mathcal{M}$
- kollisionsresistente Hashfunktion  $H : \{0, 1\}^* \rightarrow \mathcal{M}$

Konstruiere  $\Sigma = (Gen, Sign, Vfy)$  mit Nachrichtenraum  $\{0, 1\}^*$ :

- $Gen(1^k)$  berechnet  $(pk, sk) \leftarrow Gen'(1^k)$
- $Sign(sk, m)$  berechnet  $\sigma \leftarrow Sign'(sk, H(m))$
- $Vfy(pk, m, \sigma)$  gibt  $Vfy'(pk, H(m), \sigma)$  aus

## 2 q-mal Signaturen

### 2.1 Von EUF-naCMA-Sicherheit zu EUF-CMA-Sicherheit

Gegeben

- ein EUF-naCMA-sicheres Signaturverfahren  $\Sigma'$  und
- ein EUF-1-naCMA-sicheres Einmalsignaturverfahren  $\Sigma^{(1)}$

können wir mittels **Transformation** ein **EUF-CMA**-sicheres Signaturverfahren  $\Sigma$  konstruieren.

#### 2.1.1 Transformation

Gegeben:

- EUF-naCMA-sicheres Signaturverfahren  $\Sigma' = (Gen', Sign', Vfy')$
- EUF-1-naCMA-sicheres Signaturverfahren  $\Sigma^{(1)} = (Gen^{(1)}, Sign^{(1)}, Vfy^{(1)})$

Konstruiere nun  $\Sigma = (Gen, Sign, Vfy)$  wie folgt:

- $Gen(1^k)$ :

$$(pk, sk) := (pk', sk') \leftarrow Gen'(1^k)$$

- $Sign(sk, m)$ :

$$\begin{aligned} (pk^{(1)}, sk^{(1)}) &\leftarrow Gen^{(1)}(1^k) \\ \sigma' &\leftarrow Sign'(sk, pk^{(1)}) \\ \sigma^{(1)} &\leftarrow Sign^{(1)}(sk^{(1)}, m) \\ \sigma &:= (pk^{(1)}, \sigma^{(1)}, \sigma') \end{aligned}$$

- $Vfy(pk, m, \sigma)$  gibt 1 aus, wenn

$$Vfy'(pk, pk^{(1)}, \sigma') = 1 \wedge Vfy^{(1)}(pk^{(1)}, m, \sigma^{(1)}) = 1$$

sonst 0

Es wird also für jede Signatur ein neues Einmalschlüsselpaar erzeugt.

### 2.2 Mehrmal-Signaturverfahren aus Einmalsignaturverfahren

Einmalsignaturverfahren sind effizient und einfach zu konstruieren, daher würden wir gerne eine Variation dieser verwenden, um mehrfach signieren zu können (q-mal-Signaturverfahren).

#### 2.2.1 Naiver Ansatz: q Schlüsselpaare

Der naive Ansatz ist,  $q$  Schlüsselpaare zu verwenden und einen Zähler  $st \in \{1, \dots, q\}$  als Zustand zu verwenden, der auch im Secret Key und in der Signatur vorkommt:

- $Gen(1^k)$ :

$$\begin{aligned} (pk_i, sk_i) &\leftarrow Gen^{(1)}(1^k) \text{ für alle } i \in \{1, \dots, q\} \\ pk &:= (pk_1, \dots, pk_q) \\ sk &:= (sk_1, \dots, sk_q, st = 1) \end{aligned}$$

- $Sign(sk, m)$ :

$$i := st$$

$$\sigma_i \leftarrow Sign^{(1)}(sk_i, m)$$

$$\sigma \leftarrow (\sigma_i, i)$$

$$st := st + 1$$

- $Vfy(pk, m, \sigma = (\sigma_i, i))$ :

$$Vfy^{(1)}(pk_i, m, \sigma_i) \stackrel{?}{=} 1$$

### Eigenschaften bezogen auf Signaturanzahl (q):

- $|pk| \in \Theta(q)$
- $|sk| \in \Theta(q)$
- $|\sigma| \in \Theta(1)$

### 2.2.2 Zwischenschritt: Hashfunktion verwenden

Ein weiterer möglicher Ansatz ist die verwendung einer Hashfunktion

- $H$  Hashfunktion
- $Gen(1^k)$ :

$$(pk_i, sk_i) \leftarrow Gen^{(1)}(1^k) \text{ für alle } i \in \{1, \dots, q\}$$

$$pk := H(pk_1, \dots, pk_q)$$

$$sk := (sk_1, \dots, sk_q, pk_1, \dots, pk_q, st = 1)$$

- $Sign(sk, m)$ :

$$i := st$$

$$\sigma_i \leftarrow Sign^{(1)}(sk_i, m)$$

$$\sigma \leftarrow (\sigma_i, i, pk_1, \dots, pk_q)$$

$$st := st + 1$$

- $Vfy(pk, m, \sigma = (\sigma_i, i))$ :

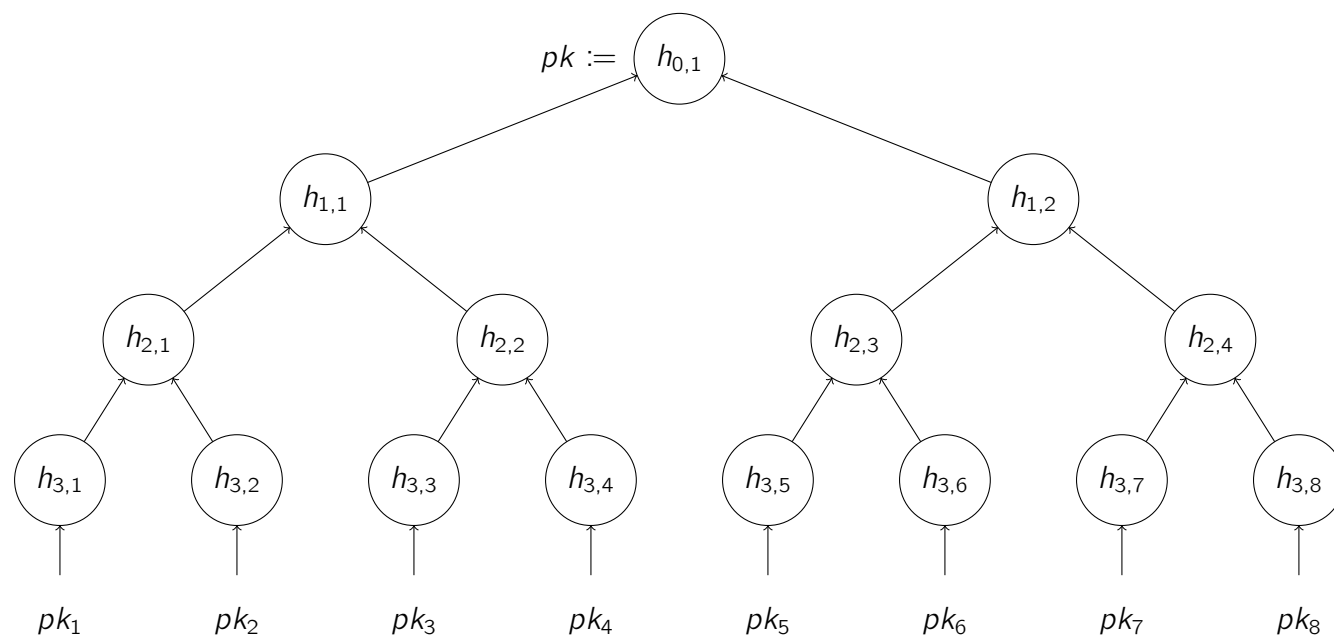
$$Vfy^{(1)}(pk_i, m, \sigma_i) \stackrel{?}{=} 1 \text{ und } H(pk_1, \dots, pk_q) \stackrel{?}{=} pk$$

### Eigenschaften bezogen auf Signaturanzahl (q):

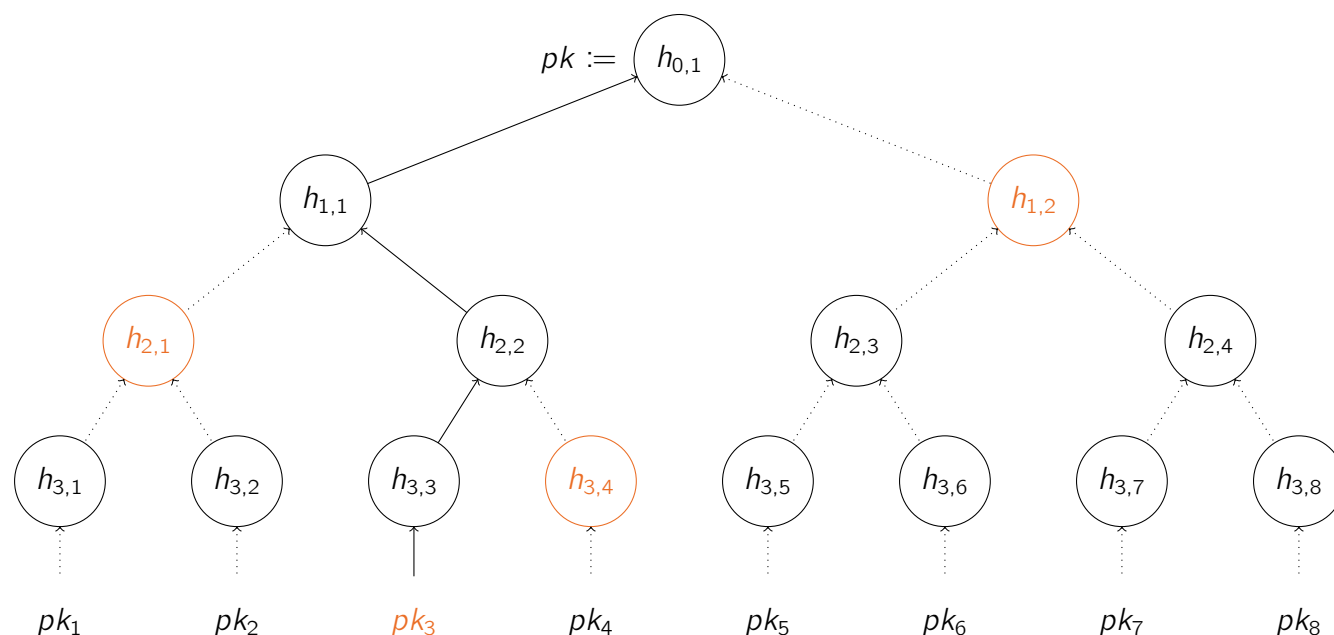
- $|pk| \in \Theta(1)$
- $|sk| \in \Theta(q)$
- $|\sigma| \in \Theta(q)$

## 2.2.3 Merkle-Bäume

**Merkle-Bäume** (auch **Hash-Bäume** genannt) sind (meist binäre) Bäume, bei denen die Blätter Hashwerte der Daten sind und jeder Knoten darüber aus Hashwerten seiner Kinder besteht:



Der **Co-Pfad** eines Knotens  $v$  in einem Binärbaum mit Wurzel  $r$  ist die Folge aller Knoten  $u_1, \dots, u_n$  wobei  $u_i$  der Geschwisterknoten des  $i$ -ten Knotens auf dem Pfad von  $v$  zu  $r$  ist:



Der **Co-Pfad** wird nun in die Signatur hinzugefügt, wodurch der  $pk$  von  $pk_3$  ausgehend (in diesem Beispiel) in  $Vfy$  berechnet werden kann.

- $Gen(1^k)$ :

$$(pk_i, sk_i) \leftarrow Gen^{(1)}(1^k) \text{ für alle } i \in \{1, \dots, q\}$$

$$pk := \text{Baum-Hash}(pk_1, \dots, pk_q)$$

$$sk := (sk_1, \dots, sk_q, pk_1, \dots, pk_q, st = 1)$$



- $Sign(sk, m)$ :

$$i := st$$

$$\sigma_i \leftarrow Sign^{(1)}(sk_i, m)$$

$$\sigma \leftarrow (\sigma_i, i, pk_i, \text{Co-Pfad})$$

$$st := st + 1$$

- $Vfy(pk, m, \sigma)$ :

Berechne Wurzel  $h'$

$$Vfy^{(1)}(pk_i, m, \sigma_i) \stackrel{?}{=} 1 \text{ und } h' \stackrel{?}{=} pk$$

### Eigenschaften bezogen auf Signaturanzahl (q):

- $|pk| \in \Theta(1)$
- $|sk| \in \Theta(q)$
- $|\sigma| \in \Theta(\log q)$

## 2.3 Komprimieren des geheimen Schlüssels

Um den geheimen Schlüssel zu komprimieren, verwenden wir statt echtem Zufall *Pseudozufall* zur Generierung.

### 2.3.1 Pseudozufallsfunktion

Eine Pseudozufallsfunktion oder Pseudorandom function (**PRF**) ist eine Funktion, die ununterscheidbar von einer zufälligen Funktion ist. Sie erhält dafür zusätzlich einen *Seed*  $s$  mit Länge  $k$  als Eingabe:

$$\begin{array}{ccccc} \text{PRF: } \{0, 1\}^k \times \{0, 1\}^n & \rightarrow & \{0, 1\}^l \\ \uparrow & & \uparrow & & \uparrow \\ \text{Seed } s & & \alpha & & PRF(s, \alpha) \end{array}$$

### 2.3.2 Schlüsselgenerierung

Bisher wird unser Schlüssel durch einen *probabilistischen* Algorithmus erzeugt:

$$(pk_i, sk_i) \leftarrow Gen^{(1)}(1^k) \quad \forall i \in \{1, \dots, q\}$$

Probabilistische Algorithmen können auch als **deterministische Algorithmen mit Zufall  $r$  als Eingabe** gesehen werden:

$$Gen^{(1)}(1^k) \quad \hat{=} \quad Gen_{\text{det}}^{(1)}(1^k, r)$$

Damit ist die bishere Schlüsselberechnung äquivalent zu folgender:

$$(pk_i, sk_i) := Gen_{\text{det}}^{(1)}(1^k, r_i) \quad \forall i \in \{1, \dots, q\} \quad \text{für echt zufällige } r_i$$

Mit **echt zufälliger** Funktion  $F$  also:

$$(pk_i, sk_i) := Gen_{\text{det}}^{(1)}(1^k, F(i)) \quad \forall i \in \{1, \dots, q\}$$

Mit einem zufälligen Seed  $s$  können wir den echten Zufall durch Pseudozufall ersetzen:

$$(pk_i, sk_i) := Gen_{\text{det}}^{(1)}(1^k, PRF(s, i)) \quad \forall i \in \{1, \dots, q\} \quad \text{für } s \xleftarrow{\$} \{0, 1\}^k$$

Dadurch müssen nur noch der Seed  $s$  und der Zähler  $st$  im Secret Key gespeichert werden, bei Bedarf können die Schlüsselpaare neu berechnet werden:

$$sk = (s, st)$$

## Eigenschaften bezogen auf Signaturanzahl ( $q$ ):

- $|pk| \in \Theta(1)$
- $|sk| \in \Theta(1)$
- $|\sigma| \in \Theta(\log q)$