# Merkle-Baum

**Idee:** Konstruktion von Merkle-Bäumen

- Ersetze Signaturkette durch Baum (sogenannter Merkle-Baum).
- Verwenden Baum der Tiefe $n$ für Nachrichten der Länge $n$.
- Die Wurzel erhält Label $\epsilon$.
- Die Kinder eines Knotens mit Label $w$ erhalten Label $w0$ und $w1$.
- Blätter besitzen Nachrichten-Label $m \in \{0,1\}^n$.
- Knoten mit Label $w$ speichern Schlüsselpaar $pk_w, sk_w$.
- Wurzelschlüssel $pk_\epsilon$ ist der öffentliche Schlüssel.

**Ziel:** Zertifiziere Pfad von Wurzel zu $m$ mittels Signaturen.

- Signieren Public-Keys auf Pfad inklusive der Nachbarknoten.

# Signieren und Verifizieren von $m = 001$

**Signieren von $m = 001$**

- Pfad von Wurzel zu $m$: $\epsilon, 0, 00, 001$ mit Nachbarknoten $1, 01, 000$.
- Signiere $(\mathbf{pk_0}||pk_1)$ mittels $sk_\epsilon$. Sei dies $\sigma_\epsilon$.
- Signiere $(\mathbf{pk_{00}}||pk_{01})$ mittels $sk_0$. Sei dies $\sigma_0$.
- Signiere $(pk_{000}||\mathbf{pk_{001}})$ mittels $sk_{00}$. Sei dies $\sigma_{00}$.
- Signiere $m$ mittels $sk_{001}$. Sei dies $\sigma_{001}$.
- Signatur ist $\sigma = (pk_0||pk_1, \sigma_\epsilon, pk_{00}||pk_{01}, \sigma_0, pk_{000}||pk_{001}, \sigma_{00}, \sigma_{001})$

**Verifizieren von $(m, \sigma)$:**

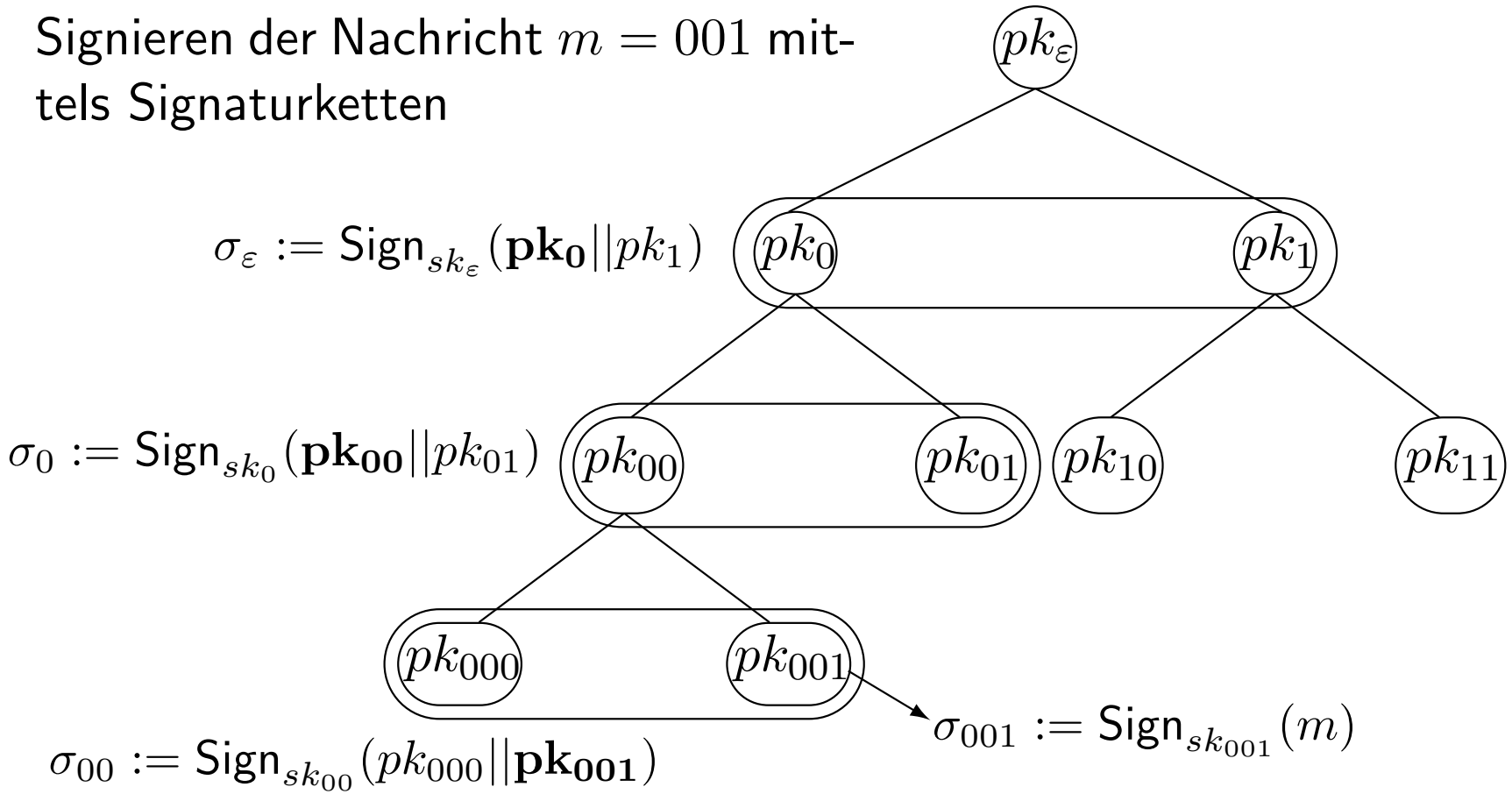
- Verifiziere $(pk_0||pk_1, \sigma_\epsilon)$ mittels $pk_\epsilon$.
- Verifiziere $(pk_{00}||pk_{01}, \sigma_0)$ mittels $pk_0$.
- Verifiziere $(pk_{000}||pk_{001}, \sigma_{00})$ mittels $pk_{00}$.
- Verifiziere $(001, \sigma_{001})$ mittels $pk_{001}$.

**Notation:**

- Sei $m \in \{0,1\}^n$. Wir definieren $m|_i := m_1 \dots m_i$ für $i = 0, \dots, n$.
- D.h. $m|_i$ ist der $i$-Zeichen Präfix von $m$ und $m|_0 = \epsilon$.

# Signaturpfad der Nachricht $m = 001$

Signieren der Nachricht $m = 001$ mittels Signaturketten



# Merkle Signaturen

## **Algorithmus** Merkle Signatur

Sei $\Pi' = (Gen', Vrfy', Sign')$ ein Einwegsignaturverfahren.

1. **Gen:** $(pk_\epsilon, sk_\epsilon) \leftarrow Gen'(1^n)$
2. **Sign:** Für Nachricht $m \in \{0,1\}^n$: FOR $i := 0$ to $n-1$
   - Falls $pk_{m|_i 0}, pk_{m|_i 1}$ noch nicht erzeugt, erzeuge und speichere $(pk_{m|_i 0}, sk_{m|_i 0}) \leftarrow Gen'(1^n)$, $(pk_{m|_i 1}, sk_{m|_i 1}) \leftarrow Gen'(1^n)$.
   - Erzeuge $\sigma_{m|_i} \leftarrow Sign'_{sk_{m|_i}}(pk_{m|_i 0}, pk_{m|_i 1})$.

   Berechne $\sigma_m \leftarrow Sign'_{sk_m}(m)$

   Ausgabe von $\sigma = ((pk_{m|_i 0} || pk_{m|_i 1}, \sigma_{m|_i})_{i=0}^{n-1}, \sigma_m)$.
3. **Vrfy':** Für $(m, \sigma)$ überprüfe

   $Vrfy'_{pk_{m|_i}}(pk_{m|_i 0} || pk_{m|_i 1}, \sigma_{m|_i}) \stackrel{?}{=} 1$ für $i = 0, \ldots, n-1$ und

   $Vrfy'_{pk_m}(m, \sigma_m) \stackrel{?}{=} 1$.

# Eigenschaften von Merkle Signaturen

**Eigenschaften:**

- Erlaubt das Signieren aller möglichen $2^n$ Nachrichten.
- Schlüsselpaare werden nur bei Bedarf erzeugt.

**Vorteile:** gegenüber Signaturketten

- Signaturlänge/Verifikationszeit sind linear in der Nachrichtenlänge aber unabhängig von der Anzahl Signaturen.
- Keine Preisgabe der zuvor signierten Nachrichten.

**Satz** Sicherheit von Merkle Signaturen

Sei $\Pi'$ eine CMA-sichere Einwegsignatur. Dann sind Merkle Signaturen $\Pi$ ein CMA-sicheres Signaturverfahren.

**Beweis:**

- Sei $\mathcal{A}$ ein Angreifer mit $\epsilon(n) := \mathrm{Ws}[Forge_{A,\Pi}(n) = 1]$.
- $\mathcal{A}'$ frage höchstens $q$ Signaturen an. Setze $\ell := 2nq + 1$ als obere Schranke für die Anzahl der benötigten Schlüssel in $\Pi$.
- Wir konstruieren mittels $\mathcal{A}$ einen Angreifer $\mathcal{A}'$ für $\Pi'$.

# Sicherheit von Merkle Signaturen

## **Algorithmus** Angreifer $\mathcal{A}$ für die Einwegsignatur

EINGABE: $pk$, Zugriff auf eine Anfrage an Orakel $Sign_{sk}(\cdot)$

1. Berechne $(pk^{(k)}, sk^{(k)}) \leftarrow Gen(1^n)$ für $k = 1, \ldots, \ell$.
   Wähle $k^* \in_R [\ell]$. Ersetze $pk^{(k^*)}$ durch $pk$. $j \leftarrow 2$
2. $(m, \sigma') \leftarrow \mathcal{A}'(pk^{(1)})$. Signaturanfragen für $m$: For $i \leftarrow 1$ to $n-1$
   - Falls $pk_{m|_i 0}, pk_{m|_i 1}$ undefiniert, $(pk_{m|_i 0}, pk_{m|_i 1}) \leftarrow (pk^{(j)}, pk^{(j+1)})$.
     $j \leftarrow j + 2$
   - Berechne $\sigma_{m|_i}$, $\sigma_m$ und $\sigma$ analog zu Merkle-Signaturen.
     Falls $sk = sk^{(k^*)}$ benötigt, verwende das Signierorakel $Sign_{sk}(\cdot)$.
3. Sei $\sigma' = ((pk'_{m|_i 0} || pk'_{m|_i 1}, \sigma'_{m|_i})_{i=0}^{n-1}, \sigma'_m)$
   - **Fall 1:** $\exists 0 \leq i < n$ mit $(pk'_{m|_i 0} || pk'_{m|_i 1}) \neq (pk_{m|_i 0} || pk_{m|_i 1})$.
     Sei $i$ minimal. Dann gilt $pk'_{m|_i} = pk_{m|_i} = pk^{(k)}$ für ein $k \in [\ell]$.
     Falls $k = k^*$, Ausgabe $(pk'_{m|_i 0} || pk'_{m|_i 1}, \sigma'_{m|_j})$.
   - **Fall 2:** Es gilt $pk'_m = pk_m = pk^{(k)}$ für ein $k \in [\ell]$.
     Falls $k = k^*$, Ausgabe $(m, \sigma'_m)$.

# Sicherheit von Merkle Signaturen

**Beweis:** Fortsetzung

- Verteilung der Nachrichten für $\mathcal{A}'$ ist identisch zum Forge-Spiel.
- D.h. $\mathcal{A}$ liefert eine gültige Signatur $(m', \sigma')$ mit Ws $\epsilon(n)$.
- Sowohl für Fall 1 als auch für Fall 2 gilt $\mathrm{Ws}[k = k^*] = \frac{1}{\ell}$.
- Wir nehmen im folgenden an, dass $k = k^*$.
- **Fall 1:** $\exists$ neuer Public-Key in Geschwisterknotenpaar.
- $\mathcal{A}$ stellte eventuell Orakelanfrage für Nachricht $(pk_{m|_i 0}||pk_{m|_i 1})$.
- Wegen $(pk'_{m|_i 0}||pk'_{m|_i 1}) \neq (pk_{m|_i 0}||pk_{m|_i 1})$ ist $\sigma'_{m|_i}$ bezüglich $pk$ eine gültige Signatur für eine neue Nachricht.
- **Fall 2:** $pk'_m$ existiert bereits.
- $\mathcal{A}'$ kann nicht Orakelanfrage $m$ gestellt haben, da er $m$ ausgibt.
- Damit ist $\sigma'_m$ eine gültige neue Signatur für $m$ bezüglich $pk$.
- **Insgesamt:** $\mathrm{negl}(n) \geq \mathrm{Ws}[Forge^{einweg}_{\mathcal{A},\Pi}(n) = 1] = \frac{\epsilon(n)}{\ell}$.
- Da $\ell$ polynomiell ist, folgt $\epsilon(n) \leq \mathrm{negl}(n)$.

# Existenz CMA-sicherer Signatur

**Korollar** Signatursatz

Falls kollisionsresistente Hashfunktionen existieren, so existiert ein CMA-sicheres Signaturverfahren.

**Beweis:**

- Unter der Annahme kollisionsresistenter Hashfunktionen existieren CMA-sichere Einwegsignaturen.
- Aus CMA-sicheren Einwegsignaturen können CMA-sichere Signaturen konstruiert werden.

**Anmerkung:**

- Man kann sogar zeigen, dass ein CMA-sicheres Signaturverfahren existiert unter der Annahme der Existenz von Einwegfunktionen.

# Digital Signature Standard (DSA)

**Systemparameter:**

- Primzahlen $p, q$, wobei $q$ Bitlänge $n$ besitzt und $q|p-1$, $q^2 \nmid p-1$.
- Generator $g$ einer Untergruppe von $\mathbb{Z}_p^*$ mit Ordnung $q$.

## **Algorithmus** Digital Signature Standard

1. **Gen:** $(p, q, g, H) \leftarrow Gen(1^n)$ mit Hashfunktion $H : \{0,1\}^* \to \mathbb{Z}_q$.
   Wähle $x \in_R \mathbb{Z}_q$, berechne $X := g^x \bmod p$.
   Setze $pk = (p, q, g, H, X)$, $sk = (p, q, g, H, x)$.
2. **Sign:** Für $m \in \{0,1\}^*$, wähle $r \in_R \mathbb{Z}_q^*$ und berechne
   $$R = g^r \bmod p, \quad t := R \bmod q. \quad s := (H(m) + xt) \cdot r^{-1} \bmod q.$$
   Signatur $\sigma = (s, t)$.
3. **Vrfy:** Für $(m, \sigma = (s, t))$ berechne $\hat{R} = (g^{H(m)} X^t)^{1/s}$ und überprüfe
   $$\hat{R} \stackrel{?}{=} t \bmod q.$$

# Eigenschaften des DSS

**Korrektheit:**

$$\begin{aligned} \hat{R} &= (g^{H(m)} X^t)^{1/s} \bmod p \\ &= g^{(H(m)+xt)\cdot(H(m)+xt)^{-1} r} = g^r \bmod p. \end{aligned}$$

**Parameterwahl:**

- Bitlänge von $p$: 1024, Bitlänge $n$ von $q$: 160.
- Die Signaturlänge von $(s, t) \in \mathbb{Z}_q^2$ ist damit nur 320 Bit.
- Dlog in $\mathbb{Z}_p^*$: subexponentieller Index-Calculus Algorithmus
- Dlog in $\langle g \rangle$: Pollard-Rho mit Komplexität $\mathcal{O}(2^{\frac{n}{2}})$.

**Sicherheit:**

- Keine größeren Schwächen bekannt.
- Aber: DSS besitzt **keinen** Sicherheitsbeweis.

# Schnorr Signaturen

**Systemparameter:**

- Primzahl $q$, wobei $q$ Bitlänge $n$ besitzt.
- Generator $g$ einer Gruppe mit Ordnung $q$.

## **Algorithmus** Schnorr Signaturen

1. **Gen:** $(q, g, H) \leftarrow Gen(1^n)$ mit Hashfunktion $H : \{0,1\}^* \rightarrow \{0,1\}^k$.
   Wähle $x \in_R \mathbb{Z}_q$, berechne $X := g^x \bmod p$.
   Setze $pk = (q, g, H, X)$, $sk = (q, g, H, x)$.
2. **Sign:** Für $m \in \{0,1\}^*$, wähle $r \in_R \mathbb{Z}_q$ und berechne
   $$R := g^r \bmod p, \quad t := H(R||m). \quad s := r + xt \bmod q.$$
   Signatur $\sigma = (s, t) \in \mathbb{Z}_q \times \{0,1\}^k$.
3. **Vrfy:** Für $(m, \sigma = (s, t))$ berechne $\hat{R} = g^s X^{-t}$ und überprüfe
   $$H(\hat{R}||m) \stackrel{?}{=} t.$$

# Eigenschaften von Schnorr Signaturen

**Korrektheit:**

$$\begin{aligned} \hat{R} &= g^s X^{-t} \\ &= g^{r+xt} g^{-xt} = g^r = R \bmod p. \end{aligned}$$

**Parameterwahl:**

- Bitlänge $n$ von $q$: 160, Bitlänge von $k$: 80
- Die Signaturlänge von $(r, s) \in \mathbb{Z}_q \times \{0,1\}^k$ ist damit nur 240 Bit.

**Sicherheit:**

- Sicherheitsbeweis unter der Diskreten Logarithmus Annahme wenn $H$ als Random Oracle modelliert wird.

# Übersicht Krypto I

Abkürzungen:

- PRNG = Pseudozufallsgenerator
- PRF = Pseudozufallsfunktion.

| **Funktionalität** | **Sicherh.** | **Konstrukt** | **Annahme** |
|---|---|---|---|
| One-Time Pad Verschlüsselung | perfekt | $m \oplus k$ | keine |
| Stromchiffre | KPA | $m \oplus G(k)$ | PRNG |
| Blockchiffre (CBC, OFB, CTR) | CPA | $(r, m \oplus F_k(r))$ | PRF |
| MAC | unfälsch-bar | $F_k(m)$ | PRF |
| Encrypt-then-Authenticate | CCA | $(c, t) = (Enc_{k_1}(m), Mac_{k_2}(c))$ | PRF |

# Übersicht Krypto II - Verschlüsselung

Abkürzung: TD-OWP = Trapdoor-Einwegpermutation

| **Funktionalität** | **Konstrukt** | **Annahme** |
|---|---|---|
| Schlüsselaustausch | sicher | Unterscheidungsannahme |
| *Diffie-Hellman* | $g^{xy}$ | *DDH:* $g^{xy} \leftrightarrow g^z$ |
| PK Verschlüsselung | CPA | TD-OWP |
| *RSA* | $(r^e, hc(r) \oplus m)$ | *RSA-Annahme* |
| *Rabin* | $(x^2, lsb(x) \oplus m)$ | *Faktorisierungsannahme* |
| PK Verschlüsselung | CPA (ROM) | TD-OWP |
| *ROM-RSA* | $(r^e, H(r) \oplus m)$ | *RSA-Annahme* |
| PK Verschlüsselung | CCA (ROM) | TD-OWP + PRF |
| *ROM-RSA2* | $(r^e, Enc'_{H(r)}(m))$ | *RSA-Annahme* |
| PK Verschlüsselung | CPA | Unterscheidungsannahme |
| *ElGamal* | $(g^y, g^{xy} \cdot m)$ | *DDH:* $g^{xy} \leftrightarrow g^z$ |
| *Goldwasser-Micali* | $z^m \cdot x^2$ | *QR:* $QR_N \leftrightarrow QNR_N^{+1}$ |
| *Paillier* | $(1+N)^m \cdot r^N$ | *DCR:* $r^N \bmod N^2 \leftrightarrow r$ |