Beliebige Anzahl von Signaturen

Algorithmus Signaturketten

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

1. **Gen':** $(pk_1, sk_1) \leftarrow Gen(1^n)$
2. **Sign':** Signieren der Nachricht m_i.
   - Verwende gemerkten Zustand $(m_{i-1}, \sigma_{i-1}, sk_i)$.
   - $(pk_{i+1}, sk_{i+1}) \leftarrow Gen(1^n)$. Berechne $\sigma'_i = Sign_{sk_i}(m_i||pk_{i+1})$.
   - Ausgabe $\sigma_i = (m_{i-1}, \sigma_{i-1}, pk_{i+1}, \sigma'_i)$. Merke $(m_i, \sigma_i, sk_{i+1})$.
3. **Vrfy':** Verifikation von $(m_i, \sigma_i) \overset{Sortieren}{=} (m_j, pk_{j+1}, \sigma'_j)_{j=1}^{i}$:

   Überprüfe $Vrfy_{pk_j}(m_j||pk_{j+1}, \sigma'_j) \overset{?}{=} 1$ für $j = 1, \ldots, i$.

- **Vorteile:** Ein öffentlicher Schlüssel pk_1, beliebige Signaturanzahl.
- **Nachteile:** Signaturlänge, Zustandsgröße und Verifikationszeit sind linear in der Anzahl der signierten Dokumente.
- Signaturen geben alle zuvor signierten Dokumente preis.

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 ϵ.
- 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_ϵ 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_ϵ. Sei dies σ_ϵ.
- Signiere $(\mathbf{pk_{00}}||pk_{01})$ mittels sk_0. Sei dies σ_0.
- Signiere $(pk_{000}||\mathbf{pk_{001}})$ mittels sk_{00}. Sei dies σ_{00}.
- Signiere m mittels sk_{001}. Sei dies σ_{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, σ):

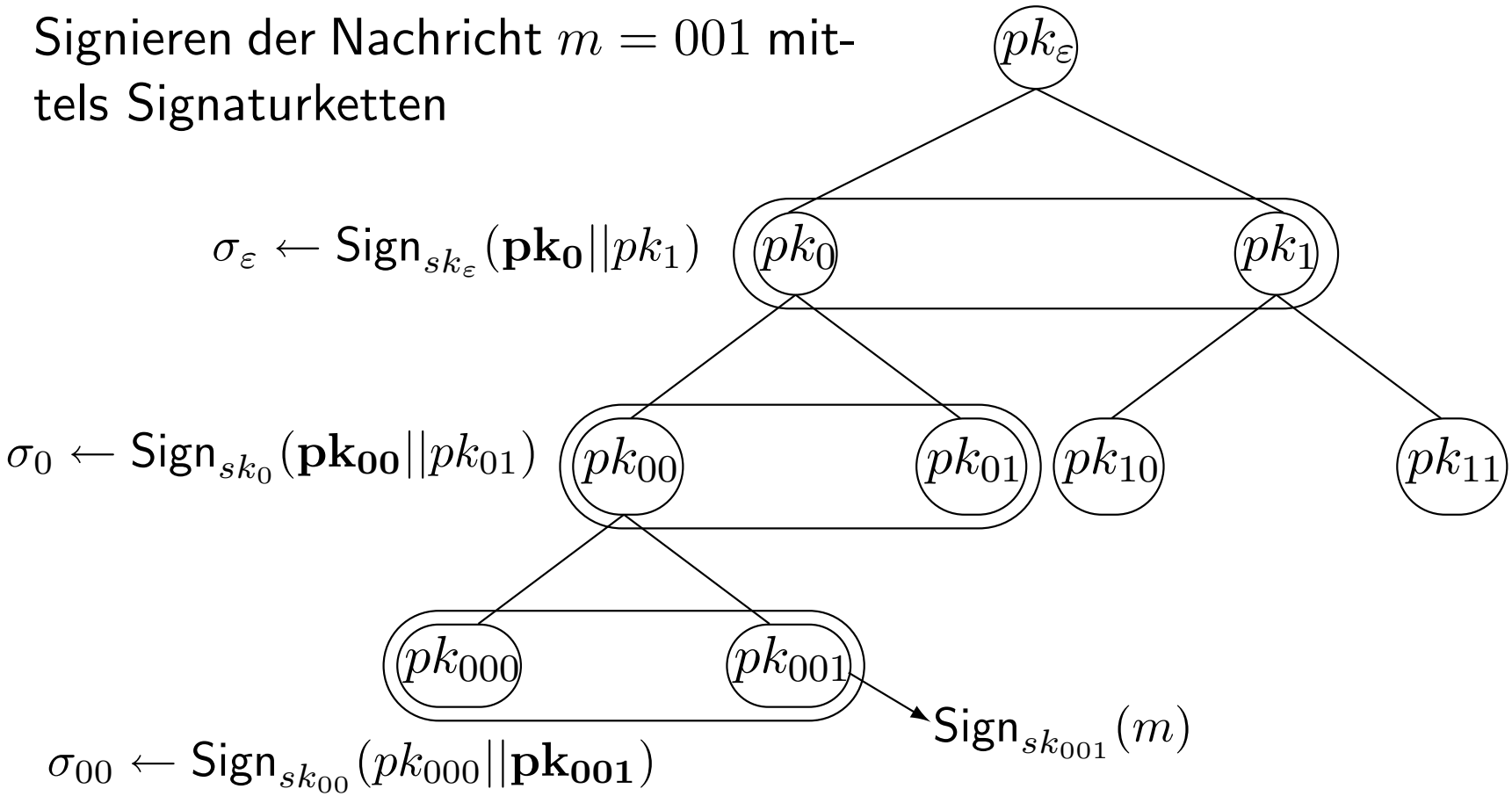
- Verifiziere $(pk_0||pk_1, \sigma_\epsilon)$ mittels pk_ϵ.
- Verifiziere $(pk_{00}||pk_{01}, \sigma_0)$ mittels pk_0.
- Verifiziere $(pk_{000}||pk_{001}, \sigma_{00})$ mittels pk_{00}.
- Verifiziere $(001, \sigma_{001})$ mittels pk_{001}.

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 $m|_i = \epsilon$.

Signaturkette der Nachricht $m = 001$

Signieren der Nachricht $m = 001$ mittels Signaturketten



Signieren und Verifizieren von $m = 101$

Signieren von $m = 101$

- Pfad von Wurzel zu m: $\epsilon, 1, 10, 101$ mit Nachbarknoten $0, 11, 100$.
- Signiere $(pk_0 || \mathbf{pk_1})$ mittels sk_ϵ. Sei dies σ_ϵ.
- Signiere $(\mathbf{pk_{10}} || pk_{11})$ mittels sk_1. Sei dies σ_1.
- Signiere $(pk_{100} || \mathbf{pk_{101}})$ mittels sk_{10}. Sei dies σ_{10}.
- Signiere m mittels sk_{101}. Sei dies σ_{101}.
- Signatur ist $\sigma = (pk_0 || pk_1, \sigma_\epsilon, pk_{10} || pk_{11}, \sigma_1, pk_{100} || pk_{101}, \sigma_{10}, \sigma_{101})$

Verifizieren von (m, σ):

- Verifiziere $(pk_0 || pk_1, \sigma_\epsilon)$ mittels pk_ϵ.
- Verifiziere $(pk_{10} || pk_{11}, \sigma_1)$ mittels pk_1.
- Verifiziere $(pk_{100} || pk_{101}, \sigma_{10})$ mittels pk_{10}.
- Verifiziere $(101, \sigma_{101})$ mittels pk_{101}.

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 $m|_i = \epsilon$.

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 $i \leftarrow 0$ 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, σ) 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 Π eine CMA-sichere Einwegsignatur. Dann sind Merkle Signaturen Π' ein CMA-sicheres Signaturverfahren.

Beweis:

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

S

Sicherheit von Merkle Signaturen

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

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

1. Berechne $(pk^{(i)}, sk^{(i)}) \leftarrow Gen(1^n)$ für $i = 1, \ldots, \ell$.
   Wähle $i' \in_R [\ell]$. Ersetze $pk^{(i')}$ 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 $\sigma_{m|_i}$, σ_m und σ analog zu Merkle-Signaturen.
     Falls $sk = sk^{(i')}$ 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 $k = i'$, 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 $k = i'$, Ausgabe (m, σ'_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', σ') mit Ws $\epsilon(n)$.
- Sowohl für Fall 1 als auch für Fall 2 gilt $\mathrm{Ws}[k = i'] = \frac{1}{\ell}$.
- Wir nehmen im folgenden an, dass $k = i'$.
- **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 σ'_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 ℓ 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.

Anmerkung:

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