Einwegsignaturen für Nachrichten beliebiger Länge

Satz Einwegsignaturen für Nachrichten beliebiger Länge

Unter der Annahme kollisionsresistenter Hashfunktionen existiert ein CMA-sicheres Einwegsignatur-Verfahren für Nachrichten beliebiger Länge.

Beweisskizze:

- Aus der Existenz von kollisionsresistenten Hashfunktionen folgt die Existenz von Einwegfunktionen. (Übung)
- Nutzen Einwegfunktion f zur Konstruktion von CMA-sicheren Lamport-Signaturen der Nachrichtenlänge ℓ.
- Nutzen Hash-and-Sign Paradigma für beliebige Nachrichtenlänge. Hier verwenden wir erneut kollisionsresistente Hashfunktionen.

Einfache k-wegsignaturen

k-wegsignaturen

- Definiere mittels k-maliger Anwendung von $Gen_{\text{Lamport}}(1^n)$
$$pk = (pk_1, \ldots, pk_k) \text{ und } sk = (sk_1, \ldots, sk_k).$$
- Unterzeichnen die i-te Nachricht m mittels sk_i als (m, σ).
- Man bezeichnet i auch als *Zustand* im Signaturverfahren.
- (m, σ) gilt als gültig, falls (m, σ) für ein pk_i gültig ist.

Nachteile:

- Anzahl k muss bei Schlüsselgenerierung feststehen.
- Länge von pk und sk hängen beide von k ab.

Idee:

- Konstruiere neues Schlüsselpaar nur bei Bedarf.
- Validiere (pk_{i+1}, sk_{i+1}) mittels (pk_i, sk_i).

Zwei Signaturen mit einem öffentlichen Schlüssel

- Sei $\Pi = (Gen, Sign, Vrfy)$ ein Einwegsignaturverfahren.
- Konstruieren $\Pi' = (Gen', Sign', Vrfy)$ für zwei Signaturen.

Gen':

- Erzeuge $(pk_1, sk_1) \leftarrow Gen(1^n)$.

Sign' und Vrfy' von m_1:

- Erzeuge (pk_2, sk_2). Berechne $\sigma_1' \leftarrow Sign_{sk_1}(m_1||pk_2)$.
- Ausgabe der Signatur $\sigma_1 = (pk_2, \sigma_1')$. Merke (m_1, σ_1, sk_2).
- Verifikation von $(m_1, \sigma_1) = (m_1, pk_2, Sign_{sk_1}(m_1||pk_2))$:

$$\text{Überprüfe } Vrfy_{pk_1}(m_1||pk_2, \sigma_1') \stackrel{?}{=} 1.$$

Sign' und Vrfy' von m_2:

- Erzeuge (pk_3, sk_3). Berechne $\sigma_2' \leftarrow Sign_{sk_2}(m_2||pk_3)$.
- Ausgabe $\sigma_2 = (m_1, \sigma_1, pk_3, \sigma_2')$. Merke (m_2, σ_2, sk_3)
- Verifikation von $(m_2, \sigma_2) = (m_2, m_1, pk_2, \sigma_1', pk_3, \sigma_2')$:

$$\text{Überprüfe } Vrfy_{pk_1}(m_1||pk_2, \sigma_1') \stackrel{?}{=} 1 \textbf{ und } Vrfy_{pk_2}(m_2||pk_3, \sigma_2') \stackrel{?}{=} 1.$$

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 $m = 101$

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
   $Vrfy_{pk_m}(m, \sigma_m) \stackrel{?}{=} 1$.