

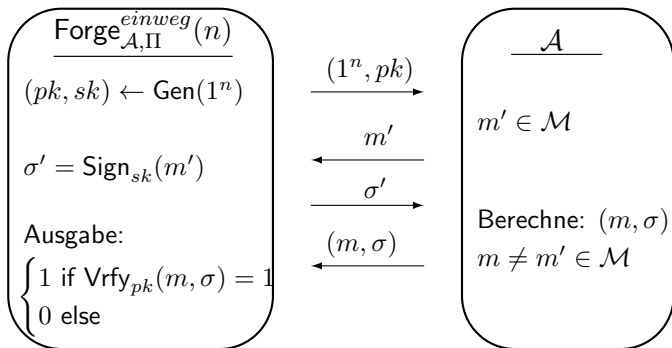
Einwegsignaturen

Ziel: Einwegsignaturen

- Konstruieren Verfahren zum sicheren Signieren *einer* Nachricht.
- Konstruktion mittels kollisionsresistenter Hashfunktionen.

Spiel $\text{Forge}_{\mathcal{A}, \Pi}^{\text{einweg}}(n)$

- 1 $(pk, sk) \leftarrow \text{Gen}(1^n)$
- 2 $(m, \sigma) \leftarrow \mathcal{A}^{\text{Sign}_{sk}(\cdot)}(pk)$, wobei $\mathcal{A}$ **eine** Nachricht $m' \neq m$ an $\text{Sign}_{sk}(\cdot)$ anfragen darf.
- 3 $\text{Forge}_{\mathcal{A}, \Pi}^{\text{einweg}}(n) = \begin{cases} 1 & \text{falls } \text{Vrfy}_{pk}(m, \sigma) = 1 \\ 0 & \text{sonst} \end{cases}$.



Definition CMA-sichere Einwegsignaturen

Ein Signaturverfahren Π heit *CMA-sichere Einwegsignatur*, falls fr alle ppt $\mathcal{A}$ gilt $\Pr[\text{Forge}_{\mathcal{A}, \Pi}^{\text{einweg}}(n) = 1] \leq \text{negl}(n)$.

Beispiel von Lamports Einwegsicherungen

Illustration: Signieren einer 3-Bit Nachricht

- Verwende Einwegfunktion $f : D \rightarrow R$.
- Wähle als geheimen Schlüssel 6 Element $x_{i,j}$ zufällig aus D . Setze

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix}.$$

- Für alle $x_{i,j}$ berechne $y_{i,j} = f(x_{i,j})$. Dies liefert

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{pmatrix}.$$

- Unterschreibe $m = m_1 m_2 m_3 \in \{0, 1\}^3$ mit $\sigma = x_{1,m_1} x_{2,m_2} x_{3,m_3}$.
- Verifikation von (m, σ) : Überprüfe $f(\sigma_i) = y_{i,m_i}$ für $i = 1, 2, 3$.

Lamports Einwegsignaturen

Definition Lamport Einwegsignaturen

Sei f eine Einwegfunktion. Konstruieren Signaturen für $m \in \{0, 1\}^\ell$.

- **Gen:** Bei Eingabe 1^n :

Wähle $x_{i,j} \in_R \{0, 1\}^n$, berechne $y_{i,j} := f(x_{i,j})$ für $i \in [\ell], j \in \{0, 1\}$.

Setze $sk = \begin{pmatrix} x_{1,0} & \dots & x_{\ell,0} \\ x_{1,1} & \dots & x_{\ell,1} \end{pmatrix}$ und $pk = \begin{pmatrix} y_{1,0} & \dots & y_{\ell,0} \\ y_{1,1} & \dots & y_{\ell,1} \end{pmatrix}$.

- **Sign:** Für $m_1 \dots m_\ell \in \{0, 1\}^\ell$, Ausgabe (m, σ) mit

$$\sigma = (x_{1,m_1}, \dots, x_{\ell,m_\ell}).$$

- **Vrfy:** Für (m, σ) überprüfe $f(\sigma_i) \stackrel{?}{=} y_{i,m_i}$ für $i \in [\ell]$.

Sicherheit von Lamport Einwegsignaturen

Satz CMA-Sicherheit von Lamport

Unter der Annahme, dass f eine Einwegfunktion ist sind Lamport Einwegsignaturen CMA-sicher für Nachrichtenlänge ℓ polynomiell in n .

Beweis: Sei $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ das Lamport Signaturverfahren.

- Sei $\mathcal{A}$ ein Angreifer mit $\epsilon(n) := \text{Ws}[\text{Forge}_{\mathcal{A}, \Pi}^{\text{einweg}}(n) = 1]$.
- Wir konstruieren einen Invertierer $\mathcal{A}'$ für f mittels $\mathcal{A}$.

Algorithmus Invertierer $\mathcal{A}'$

EINGABE: y

- 1 Wähle $i \in_R \{0, 1\}^\ell$ und $b \in_R \{0, 1\}$.
- 2 Berechne $(pk, sk) \leftarrow \text{Gen}(1^n)$. Setze $y_{i,b} \leftarrow y$ in pk .
- 3 $(m, \sigma) \leftarrow \mathcal{A}$. Bei Signaturanfrage für m' antworte mit
 $\sigma' = (\sigma_{1,m'_1}, \dots, \sigma_{\ell,m'_\ell})$ falls $m'_i \neq b$. Sonst Abbruch.

AUSGABE: $= \begin{cases} x_i & \text{falls } m_i = b \\ \text{Abbruch} & \text{sonst} \end{cases}$.

Sicherheit von Lamport Einwegsignaturen

Beweis: Fortsetzung

- Sei (m, σ) eine gültige Signatur mit $m_i = b$.
- Dann ist σ_i ein Urbild von y , d.h. $f(\sigma_i) = y$.
- Wahl von y im Invertier-Spiel erfolgt durch $x \in_R D$ und $y := f(x)$.
- D.h. pk ist identisch verteilt zum Lamport Signaturverfahren.
- Benötigen $m'_i \neq b$ und $m_i = b \neq m_i$. Es gilt $\text{Ws}[m'_i \neq b] = \frac{1}{2}$.
- Wegen $m \neq m'$ folgt $\text{Ws}[m'_i \neq m_i] \geq \frac{1}{\ell}$.
- Wir erhalten insgesamt $\text{Ws}[\text{Invert}_{\mathcal{A},f}(n) = 1]$
 - $= \text{Ws}[\text{Forge}_{\mathcal{A},\Pi}^{\text{einweg}}(n) \wedge (m'_i \neq b) \wedge (m'_i \neq m_i)]$
 - $= \text{Ws}[\text{Forge}_{\mathcal{A},\Pi}^{\text{einweg}}(n)] \cdot \text{Ws}[(m'_i \neq b)] \cdot \text{Ws}[(m'_i \neq m_i)] \geq \epsilon(n) \cdot \frac{1}{2\ell}$
- Aufgrund der Einweg-Eigenschaft von f gilt
$$\text{negl}(n) \geq \text{Ws}[\text{Invert}_{\mathcal{A},f}(n) = 1].$$
- Daraus folgt $\epsilon(n) \leq 2\ell \cdot \text{negl}(n)$.
- Dies ist vernachlässigbar für polynomielles ℓ .

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.

Beweis:

- Aus der Existenz von kollisionsresistenten Hashfunktionen folgt die Existenz von Einwegfunktionen. (Siehe auch Krypto I.)
- 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 -weg Signaturen

k -weg Signaturen

- Definiere mittels k -maliger Anwendung von $Gen_{\text{Lamport}}(1^n)$
 $pk = (pk_1, \dots, pk_k)$ und $sk = (sk_1, \dots, sk_k)$.
- 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 $\Pi = (\text{Gen}, \text{Sign}, \text{Vrfy})$ ein Einwegsignaturverfahren.
- Konstruieren $\Pi' = (\text{Gen}', \text{Sign}', \text{Vrfy}')$ für zwei Signaturen.

Gen':

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

Sign' und Vrfy' von m_1 :

- Erzeuge $(pk_2, sk_2) \leftarrow \text{Gen}(1^n)$. Berechne $\sigma'_1 \leftarrow \text{Sign}_{sk_1}(m_1 || pk_2)$.
- Ausgabe der Signatur $\sigma_1 = (pk_2, \sigma'_1)$. Merke (m_1, σ_1, sk_2) .
- Verifikation von $(m_1, \sigma_1) = (m_1, pk_2, \text{Sign}_{sk_1}(m_1 || pk_2))$:

Überprüfe $\text{Vrfy}_{pk_1}(m_1 || pk_2, \sigma'_1) \stackrel{?}{=} 1$.

Sign' und Vrfy' von m_2 :

- Erzeuge $(pk_3, sk_3) \leftarrow \text{Gen}(1^n)$. Berechne $\sigma'_2 \leftarrow \text{Sign}_{sk_2}(m_2 || pk_3)$.
- Ausgabe $\sigma_2 = (m_1, \sigma_1, pk_3, \sigma'_2)$. Merke (m_2, σ_2, sk_3)
- Verifikation von $(m_2, \sigma_2) = (m_2, m_1, pk_2, \sigma'_1, pk_3, \sigma'_2)$:

Überprüfe $\text{Vrfy}_{pk_1}(m_1 || pk_2, \sigma'_1) \stackrel{?}{=} 1$ und $\text{Vrfy}_{pk_3}(m_2 || pk_3, \sigma'_2) \stackrel{?}{=} 1$.

Beliebige Anzahl von Signaturen

Algorithmus Signaturketten

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

- ➊ **Gen'**: $(pk_1, sk_1) \leftarrow Gen(1^n)$
- ➋ **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 $(m_i, \sigma_i, sk_{i+1})$.
- ➌ **Vrfy'**: Verifikation von $(m_i, \sigma_i) \stackrel{\text{Sortieren}}{=} (m_j, pk_{j+1}, \sigma'_j)_{j=1}^i$:
Überprüfe $Vrfy_{pk_j}(m_j || pk_{j+1}, \sigma'_j) \stackrel{?}{=} 1$ für $j = 1, \dots, i$.

Vorteil: 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.