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 $\epsilon(n) := \mathrm{Ws}[Forge_{A',\Pi'}(n) = 1]$.
- $\mathcal{A}'$ 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.

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.

Digital Signature Standard – Schnorr Signaturen

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

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

Eigenschaften des DSS

Korrektheit:

$$\begin{aligned} g^{H(m)\cdot s^{-1}} y^{r\cdot s^{-1}} &= g^{H(m)(H(m)+xr)^{-1}k} g^{xr(H(m)+xr)^{-1}k} \bmod p \\ &= g^{(H(m)+xr)\cdot(H(m)+xr)^{-1}k} = g^k \bmod p. \end{aligned}$$

Parameterwahl:

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

Sicherheit:

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

Viele Public-Keys mittels eines Public Keys

Zertifierung

- Zertifizierungsstelle CA (Certificate Authority) veröffentlicht pk_{CA}.
- CA zertifiziert Schlüssel pk_A eines Nutzers Alice mit Zertifikat

$$cert_{CA \to A} \leftarrow Sign_{sk_{CA}}(\text{“alice@rub.de besitzt Schlüssel } pk_A\text{”}).$$

- Alice kann $(pk_A, cert_{CA \to A})$ über unsicheren Kanal verschicken.
- CMA-Sicherheit des Signaturverfahrens verhindert erfolgreiches Fälschen eines Zertifikat für einen anderen Schlüssel pk'_A.
- D.h. mit nur einem öffentlichen Schlüssel kann eine CA beliebig viele weitere öffentliche Schlüssel zertifizieren.
- Liefert sogenannte Public-Key Infrastruktur.

Random Oracle

Definition Random Oracle

Sei $\mathcal{F}^{n,\ell}$ die Menge aller Funktionen $\{0,1\}^n \to \{0,1\}^{\ell(n)}$. Ein *Random Oracle* ist eine zufällige Funktion $H \in_R \mathcal{F}^{n,\ell}$. Wir besitzen keine Beschreibung von H. Bei Anfrage x liefert das Random Oracle $H(x)$.

Anmerkung: Bildliche Darstellung

- Ein Random Oracle H ist eine Funktion in einer schwarzen Box.
- H ist beobachtbar über das Eingabe/Ausgabe-Verhalten der Box.

Alternative Beschreibung eines Random Oracles

- Oracle erhält Anfragen $x_1, \ldots, x_q$.
- Falls $x_i \neq x_j$ für alle $j < i$, gib $y_i \in_R \{0,1\}^{\ell(n)}$ aus.
- Falls $x_i = x_j$ für ein $j < i$, gib y_j aus.
- D.h. wir können uns vorstellen, dass das Orakel die Antworten auf Anfragen bei Bedarf erzeugt und konsistent beantwortet.

Random Oracles liefern Einwegfunktionen

Satz

Für polynomielles $\ell(n)$ sind Random Oracles Einwegfunktionen.

Beweis:

- Sei $x \in_R \{0,1\}^n$ und $y = H(x)$. Wollen ein Urbild von y ermitteln.
- Jeder Angreifer $\mathcal{A}$ stellt oBdA verschiedene Anfragen $x_1, \ldots, x_q$. (Warum sollte jeder Angreifer so verfahren?)
- $\mathcal{A}$ gewinnt offenbar falls $x_i = x$ für ein i, d.h. mit $\mathrm{Ws}[x_i = x] = \frac{q}{2^n}$.
- $\mathcal{A}$ gewinnt ebenfalls für $H(x_i) = y$, d.h. mit
$$\mathrm{Ws}[H(x_i) = y] = 1 - (1 - \tfrac{1}{2^{\ell(n)}})^q \leq 1 - (1 - \tfrac{q}{2^{\ell(n)}}) = \tfrac{q}{2^{\ell(n)}}.$$
- Damit gilt $Ws[Invert_{\mathcal{A},H}(n) = 1] \leq \frac{q}{2^n} + \frac{q}{2^{\ell(n)}}$.
- Für polynomielles q ist dies vernachlässigbar in n.
- Man beachte: $\mathcal{A}$ muss kein ppt-beschränkter Angreifer sein, der Beweis gilt für beliebige Angreifer (d.h. informationstheoretisch).

Random Oracle und kollisionsresistentes Hashen

Satz

Für polynomielles $\ell(n)$ sind Random Oracles kollisionsresistent.

Beweis:

- Jeder Angreifer $\mathcal{A}$ stellt oBdA verschiedene Anfragen $x_1, \ldots, x_q$.
- $\mathcal{A}$ gewinnt mit $Ws[H(x_i) = H(x_j)] \leq \frac{q^2}{2^{\ell(n)}}$. (Geburtstagsparadoxon)
- Dies ist vernachlässigbar für polynomielles q.

Random Oracle Methode

Definition Random Oracle Modell / Methode

Das *Random Oracle Modell (ROM)* nimmt die Existenz von Random Oracles an. Die *Random Oracle Methode* besteht aus 2 Schritten:

1. Konstruiere ein Verfahren Π mit Hilfe eines Random Oracles H und beweise die Sicherheit von Π im ROM.
2. Instantiiere Π mit einer kryptographischen Hashfunktion H' anstelle von H, z.B. mit SHA-1.

Negativ:

- Beschreibung von H' spezifiziert $H'(x)$ für alle x.
- Es existieren künstliche Kryptosysteme, die sicher im Random Oracle Modell aber unsicher für *jede* Instantiierung von H' sind.

Positiv:

- Ein Beweis im ROM ist besser als kein Beweis.
- Erfolgreicher Angriff muss die Instantiierung von H' attackieren.
- H' kann leicht durch eine andere Hashfunktion ersetzt werden.