Digitale Unterschriften

Auch digitale Signaturen genannt.

Nachrichten aus Nachrichtenraum: $M \in \mathcal{M}$.
Signaturen aus Signaturenraum: $\sigma \in S$.
Schlüssel sind aus Schlüsselräumen: $d \in K_1$, $e \in K_2$.

Signierungsverfahren $\mathcal{S} : K_1 \times \mathcal{M} \rightsquigarrow S$.
Verifizierungsverfahren $\mathcal{V} : K_2 \times \mathcal{M} \times S \to \{0,1\}$.

Signatur von Nachricht m mit Schlüssel d: $\sigma \leftarrow \mathcal{S}(d,M)$.
Verifizierung von M,σ mit Schlüssel e: $f \leftarrow \mathcal{V}(e,M,\sigma)$.

e öffentlicher Schlüssel, d privater Schlüssel.

Bemerkungen

$\mathcal{S}$ und $\mathcal{V}$ sind effiziente Verfahren (z.B. Programme).
$\mathcal{S}$ und $\mathcal{V}$ dürfen probabilistisch sein (dürfen den Zufall verwenden).
$\mathcal{S}$ kann mehrdeutig sein ($\rightsquigarrow$).

Im allgemeinen wird nur ein Hashwert $H(M)$ und nicht M selbst signiert („Hash-then-Sign“):

- Effizienter, da $H(M)$ viel kürzer als M ist.
- Beweisbare Sicherheit von in der Praxis relevanten Verfahren (allerdings im Zufallsorakelmodell, RO).

Offenbar muß H kollisionsfrei sein, man kann keine zwei Nachrichten M_1, M_2 mit $H(M_1) = H(M_2)$ berechnen.

Angriffe und Sicherheitsmodelle

Ziele des Angreifers:

- Existentielle Fälschung. Der Angreifer berechnet eine Signatur für eine Nachricht.
- Universelle Fälschung: Der Angreifer kann Signaturen für jede beliebige Nachricht berechnen.
- Total break: Der Angreifer berechnet den geheimen Schlüssel des Signierers.

Angriffe und Sicherheitsmodelle

Informationen (Fähigkeiten) des Angreifers in aufsteigender Reihenfolge:

- Key-only Angriff: Der Angreifer kennt nur den öffentlichen Schlüssel des Signierers.
- Known-Signature Angriff: Der Angreifer erhält Nachrichten und die zugehörigen Signaturen.
- Chosen-Message Angriff: Der Angreifer kann sich die Nachrichten aussuchen und erhält die zugehörigen Signaturen.

Den letzte Variante gibt es auch in adaptiver Form.

Stärkstes Sicherheitsmodell: Sicherheit bezüglich existenzieller Fälschung unter adaptiven Chosen-Message Angriffen.

Signatur mit Einwegfunktion mit Falltür

Seien $f, h : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$ derart, daß es für jedes (zufällige) $e \in \{0,1\}^n$ ein $d \in \{0,1\}^n$ gibt, so daß die Funktion $f(e,\cdot)$ eine Einwegfunktion mit Falltür d und inverser Funktion $h(d,\cdot)$ ist.

Beispiel: RSA Funktionen $f(e,\cdot) : x \mapsto x^e$, $h(d,\cdot) : y \mapsto y^d$.

Dann für $M \in \{0,1\}^m$:

- Signieren: $\sigma = \mathcal{S}(d,M) = h(d,M)$, Signatur ist (M,σ).
- Verifizieren: $\mathcal{V}(e,M,\sigma) = 1$ genau dann, wenn $f(e,\sigma) = M$.

„Sicherheit“ basiert informell also darauf, daß niemand $\sigma = h(d,M)$ ausrechnen kann, ohne d zu kennen.

Signierer beweist, daß er die geheime Information kennt, ohne d preiszugeben.

RSA Signatur

Definiert wie auf voriger Folie mit RSA Funktion.

Prinzipielle Probleme (existenzielle Fälschungen):

- Homomorphie, Produkt von Signaturen ist Signatur des Produkts der Nachrichten.
- Elemente aus dem Signaturraum leicht hinschreibbar, Verschlüsseln liefert passende Nachricht.
- Abhilfe in beiden Fällen durch „Hash-then-Sign“. Hash Funktion soll auch Einwegfunktion sein ...

Gemeinsame Verwendung mit RSA Verschlüsselung:

- Es müssen unterschiedliche Paare (d,e) verwendet werden.
- Sonst Angriff auf Verschlüsselung: Erhalte Chiffretext $c = m^e \bmod n$, berechne $cr^e \bmod n$ für r zufällig, erfrage Signatur $\sigma = (cr^e)^d \bmod n$ von $cr^e \bmod n$, berechne $m = \sigma/r$.

RSA Signatur

In PKCS # 1 Version 2.1 werden zwei RSA Signaturverfahren definiert:

- PSS (probabilistisch, keine eindeutigen Signaturen)
- PKCS #1 Version 1.5

Hier geht es im wesentlichen um die Codierung der Nachricht, auf die dann die inverse RSA Funktion angewendet wird.

PSS und die Variante zwei Folien zuvor (mit surjektiver Hash Funktion, heißt dann Full Domain Hash) sind sicher bezüglich existenzieller Fälschung unter einem adaptiven chosen-message Angriff (im Zufallsorakelmodell).

PSS hat eine bessere Sicherheitsreduktion als FDH.

Signatur mit Einwegfkt ohne Falltür

One-time Signatur Verfahren von Lamport (man kann mit einem Schlüsselpaar nur einmal unterschreiben):

Sei $f : \{0,1\}^n \to \{0,1\}^n$ eine Einwegfunktion.

Schlüsselerzeugung:

- $x_{i,j} \in \{0,1\}^n$ zufällig, für $1 \le i \le m$ und $j \in \{0,1\}$. Setze $y_{i,j} = f(x_{i,j})$. Der öffentliche Schlüssel sind die $y_{i,j}$, der geheime die $x_{i,j}$.

Signieren von $M \in \{0,1\}^m$:

- Ist $M = M_1 || \ldots || M_m$ mit $M_i \in \{0,1\}$, so setze $\sigma = x_{1,M_1} || \ldots || x_{m,M_m}$. Die Signatur ist (M,σ).

Verifizieren von (M,σ):

- Unterschrift ok $\Leftrightarrow$ $f(\sigma_i) = y_{i,M_i}$ für $\sigma = \sigma_1 || \ldots || \sigma_m$ mit $\sigma_i \in \{0,1\}^n$ und alle i.

Signatur mit Einwegfkt ohne Falltür

Dieses Signaturverfahren ist für eine bijektive Funktion f sicher bezüglich existenzieller Fälschung unter einem key-only Angriff.

Allerdings nicht besonders praktikabel, da Signaturen n-mal so lang sind wie die Nachrichten.

Will man mehrmals signieren, benötigt man für jede Signatur ein eigenes Schlüsselpaar. In dieser Situation stellt sich besonders die Frage nach der Authentizität der öffentlichen Schlüssel.

Dies kann mit einem Authentifizierungsbaum gelöst werden.

Relevanz des Verfahrens dann:

- Beliebige Einwegfunktion verwendbar, muß nicht auf Faktorisieren oder zyklischen Gruppen beruhen.
- Daher quantencomputersichere Unterschriften möglich ... (?)

Signatur mit Einwegfkt ohne Falltür

Vereinfachter Ansatz: Authentifizierungsgerade.

- Für die i-te Signatur hat man zwei Schlüsselpaare. Das erste Paar dient zum Signieren der i-ten Nachricht, das zweite Paar zum Signieren der neuen, $(i+1)$-ten Schlüsselpaare.
- Die nullten Schlüsselpaare werden „geglaubt“ oder durch ein Trustcenter authentifiziert.
- Der Verifizierer überprüft die Authentizität der Schlüssel für $i, i-1, \ldots, 0$.

Probleme:

- Specheraufwand linear in i, Verifizierungsaufwand linear in i.
- Abhilfe (teilweise): Balancierten Binärbaum (oder mehrere) verwenden, so daß Strecke bis zum nullten Schlüssel nur $O(\log(i))$ lang ist.

Schnorr Signatur (Variation)

Benutzt Einwegmonomorphismus (beispielsweise Exponieren in zyklischer Gruppe) als Einwegfunktion ohne Falltür.

Sei $p : (G,+) \to (V,\cdot)$ ein Einwegmonomorphismus zyklischer Gruppen (z.B. $\mathbb{Z}/\ell\mathbb{Z} \to V$, $x \mapsto g^x$) und H eine Hashfunktion nach $\mathbb{Z}/\ell\mathbb{Z}$, $\#G = \ell$.

Schlüsselerzeugung:

- Wähle $x \in G$ zufällig und setze $y = p(x)$. Der öffentliche Schlüssel ist y, der geheime x.

Signieren von M:

- Wähle $k \in G$ zufällig und setze $r = p(k)$, $h = H(M||r)$, $u = hx + k$, $\sigma = (u, r)$. Die Unterschrift ist (M, σ).

Verifizieren von (M, σ):

- Unterschrift ok $\Leftrightarrow p(u) = y^{H(M||r)} r$.

Schnorr Signatur

Bei der „echten“ Schnorr Signatur ersetzt man r durch h in σ:

Signieren von M:

- Wähle $k \in G$ zufällig und setze $r = p(k)$, $h = H(M||r)$, $u = hx + k$, $\sigma = (u, h)$. Die Unterschrift ist (M, σ).

Verifizieren von (M, σ):

- Unterschrift ok $\Leftrightarrow h = H(M||p(u)y^{-h})$.

Da man h aus r, aber auch r aus h und den anderen Werten ausrechnen kann, sind beide Verfahren äquivalent.

Schnorr Signatur

Die echte Variante hat im allgemeinen kürzere Signaturen, da h aus $\log_2(\ell)$, aber r aus beispielsweise $\log_2(q)$ Bits besteht, wenn $V \subseteq \mathbb{F}_q^\times$.

Für $G = \mathbb{Z}/\ell\mathbb{Z}$ werden bei der echten Variante insgesamt $2\log_2(\ell)$ Bits Signatur produziert.

Fall $V \subseteq \mathbb{F}_q^\times$:

- V kann in $\mathbb{F}_q^\times$ beliebiger Größe eingebettet sein.
- Daher kann man die Gruppengröße ℓ und Körpergröße q so aufeinander abstimmen, dass Indexcalculus in $\mathbb{F}_q^\times$ genauso lange (oder länger) braucht wie Pollard rho in V.

Schnorr Signatur

Das Schnorr Signaturverfahren ist im Zufallsorakelmodell sicher bezüglich existenzieller Fälschung unter adaptivem chosen-message Angriff. Ein erfolgreicher Angreifer könnte hierbei benutzt werden, um p zu invertieren bzw. ein DLP zu lösen.

Die Idee des Sicherheitsbeweises im Zufallsorakelmodell ist wie folgt:

Gegeben $y \in V$. Zu finden ist $x \in G$ mit $y = p(x)$.

Die Signierungsanfragen des Angreifers werden simuliert: Sei $h \in \mathbb{Z}/\ell\mathbb{Z}$ und $u \in G$ zufällig und setze $r = p(u)y^{-h}$, $H(M||r) = h$. Dann ist $\sigma = (u, r)$ eine gültige Unterschrift.

Der Angreifer liefert uns eine Fälschung (M, σ_1) mit $\sigma_1 = (u_1, r)$, für die er nicht die Signatur, aber den Hashwert $H(M||r)$ erfragt hat.

Schnorr Signatur

Wir lassen den Angreifer noch einmal rechnen mit genau denselben Eingaben, denselben Hashwerten und demselben Zufallsband. Bei der Anfrage $H(M||r)$ liefern wir jetzt einen anderen Zufallswert zurück.

Man kann zeigen („Forking Lemma“), daß der Angreifer nach Voraussetzung auch in diesem Fall mit signifikanter Wahrscheinlichkeit eine Fälschung (M, σ_2) mit $\sigma_2 = (u_2, r)$ produziert.

Dann gilt $p(u_i) = y^{h_i} r$ mit h_i dem jeweiligen Hashwert $H(M||r)$.

Es folgt $p(u_1 - u_2) = y^{h_1 - h_2}$. Da die h_i zufällig sind, gilt $h_1 - h_2 = 0 \bmod \ell$ mit insignifikanter Wahrscheinlichkeit.

Folglich $p((u_1 - u_2)/(h_1 - h_2)) = y$ und mit $x = (u_1 - u_2)/(h_1 - h_2)$ haben wir ein Urbild von y unter p berechnet. Dies zeigt im übrigen auch, daß k nicht doppelt verwendet werden oder vorhersehbar sein sollte.

DSA

DSA = Digital Signature Algorithm aus DSS.

DSS = Digital Signature Standard, NIST 1994.

Schlüsselerzeugung:

- Wähle zufällige 160-Bit Primzahl ℓ und n-Bit Primzahl p, wobei $n = 512 + 64t$, $0 \le t \le 8$, und $\ell \mid (p-1)$. Empfohlen ist $n = 1024$.
- Wähle zufälliges $h \in \mathbb{F}_p$ mit $g := h^{(p-1)/\ell} \neq 1$.
- Wähle $x \in \mathbb{Z}/\ell\mathbb{Z}$ zufällig und setze $y = g^x$.
- Der öffentliche Schlüssel ist y, der private Schlüssel ist x.

Signieren von M:

- Wähle k mit $1 \le k \le \ell - 1$ zufällig.
- Setze $h = (g^k \bmod p) \bmod \ell$ und $u = k^{-1}(\text{SHA-1}(M) + hx) \bmod \ell$. Wenn $h = 0$ oder $u = 0$, dann neues k.
- Die Signatur ist (M, σ) mit $\sigma = (u, h)$.

DSA

Verifizieren von (M, σ):

- Überprüfe $1 \leq h \leq \ell - 1$ und $1 \leq u \leq \ell - 1$. Wenn nein, Signatur zurückweisen.
- Berechne $v = u^{-1} \bmod \ell$ und $w = ((g^{\text{SHA-1}(M)} y^h)^v \bmod p) \bmod \ell$.
- Signatur ok genau dann, wenn $w = h$.

DSA ist Variante des älteren ElGamal Signaturverfahrens.

ECDSA: Elliptic Curve DSA, basiert auf elliptischen Kurven.

DSA Anmerkungen:

- k soll pseudozufällig sein, nicht doppelt verwenden.
- Ohne Verwendung von SHA-1 nicht sicher bezüglich existenzieller Fälschung unter key-only Angriff.
- Ohne Größencheck von h und u nicht sicher bezüglich universeller Fälschung, wenn nur eine Signatur gegeben ist.

Blinde Signaturen

Blinde Signaturen stellen ein Zwei-Pateien Protokoll dar.

- B schickt A Information. A erstellt eine Vorunterschrift.
- B modifiziert diese Vorunterschrift in eine reguläre Unterschrift (M, σ) von A der Nachricht M.
- A kennt weder M noch σ und kann keine Verbindung von (M, σ) zu B herstellen. B kann keine weiteren Unterschriften produzieren.

Realisierung zum Beispiel mit RSA durch Verwendung einer „blinding“ Funktion.

- B berechnet $M' = Mk^e \bmod n$ für zufälliges k und schickt M' an A.
- A signiert M', berechnet also $\sigma' = M'^d \bmod n$ und schickt σ' an B.
- B berechnet $\sigma = \sigma' k^{-1} \bmod n$. Dies ist eine Unterschrift von M.

Blinde Unterschriften finden bei elektronischem Geld Anwendung.

Undeniable Signature Schemes

Verifikation benötigt Mitarbeit des Signierers. Dieser darf aber nicht schummeln können (echte Signatur als unecht darstellen).

Verhindert Duplizierung und Verbreitung signierter Dokumente.

Schlüsselgenerierung:

- Wie üblich $y = g^x$.

Signieren:

- $\sigma = M^x$.

Verifizieren (durch Beweis des Wissens eines diskreten Logarithmus):

- Wähle x_1, x_2 zufällig, schicke $\sigma^{x_1} y^{x_2}$ zum Signierer.
- Signierer schickt $w = (\sigma^{x_1} y^{x_2})^{1/x}$ zurück.
- Signatur ok genau dann, wenn $w = M^{x_1} g^{x_2}$.

Undeniable Signature Schemes

Wenn der Signierer nicht kommuniziert, geht man davon aus, daß die Signatur ok ist.

Volle Verifikation (Überprüfung, ob die Signatur (M, σ) ok oder eine Fälschung ist oder ob der Signierer schummelt):

- Führe Verifikation mit x_1, x_2 aus, liefert w. Ist Signatur ok, dann fertig.
- Führe Verifikation mit x_1', x_2' aus, liefert w'. Ist Signatur ok, dann fertig.
- Berechne $c = (wg^{-x_2})^{x_1'}$ und $c' = (w' g^{-x_2'})^{x_1}$.
- Wenn $c = c'$, dann ist σ Fälschung. Ansonsten schummelt der Signierer.

Beweis: Übungsaufgabe.

Bemerkungen

Verwendung von Verschlüsselung und Signaturen:

- Besser erst Signieren, dann Verschlüsseln.
- Sonst ist ja nur der Chiffretext unterschrieben ...

Versteckte Informationsübertragung:

- Man kann die zufälligen Exponenten k mit Information versehen und so Signaturverfahren auch für die versteckte Informationsübertragung nutzen.
- Der private Schlüssel muß hierfür aber beiden Parteien bekannt sein ...