Digitale Signaturen

Prof. Dr. Rüdiger Weis

TFH Berlin

Sommersemester 2008

Digitale Unterschriften

Unterschrift „von Hand“:

- Physikalische Verbindung mit dem unterschriebenen Dokument (beides steht auf dem gleichen Blatt).
- Fälschen erfordert einiges Geschick (wenn die Fälschung bei einer gründlichen Prüfung nicht auffallen soll . . .).
- Kopieren einer Unterschrift ist nicht möglich (Fotokopien und Faxe haben einen geringeren Beweiswert als Originale).

Grundlagen der digitalen Unterschriften

RSA Signaturen

RSA-Modulus $n = pq$ mit öffentlichem Exponenten e und geheimem Exponenten e.
Aus dem Grundlagenkapitel wissen wir, dass für jede Nachricht $m \in \mathbb{Z}_n^*$ gilt: $(m^d)^e \equiv m \mod n$.
Wenn man zuerst die geheime („Signier“-)Operation anwendet, und dann die öffentliche Operation (die „Verifikation“), dann erhalte ich die ursprüngliche Nachricht.

Definition

Ein System für Digitale Signatureb besteht aus drei effizienten Algorithmen:

- Schlüsselerzeugung:
  $(\mathrm{PK}, \mathrm{SK}) := G(< \text{Sicherheitsparam.} >)$.
- Unterschreiben einer Nachricht m:
  $U := S(\mathrm{SK}, m)$.
- Verifizieren einer Unterschrift U' für die Nachricht m':
  $V(PK', m', U') \in \{\text{„OK“}, \text{„falsch“}\}$.

Korrektheit

Dieses System ist korrekt, wenn für jedes mit G erzeugte Schlüsselpaar $(\mathrm{PK}, \mathrm{SK})$ und jede Nachricht m gilt:

$$\textit{Wenn } U := S(SK, m), \textit{ dann ist } V(PK, m, U) = \text{„}OK\text{“}.$$

Angriffsmodell

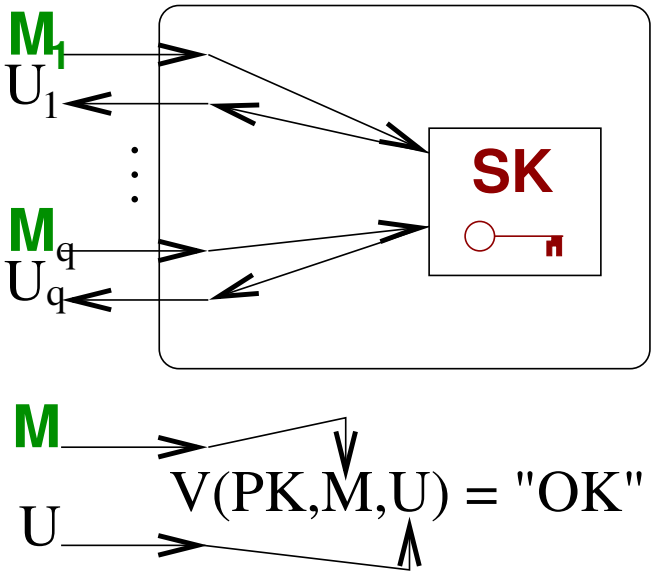
Gegner kennt öffentlichen Schlüssel und versucht, eine Unterschrift zu fälschen ("Existential Forgery"):

Fragephase: Für $i := 1$ bis q:
Nachricht M_i an Orakel,
Unterschrift
$U_i = S(\mathrm{SK}, M_i)$.

Antwortphase:
(M, U) mit
$M \notin \{M_1, \ldots, M_q\}$.

Wertung: Gegner *gewinnt*
$\Leftrightarrow V(\mathrm{PK}, M, U) =$„OK“.

Angriffsmodell (Grafik)



Sicherheit

Ein Digitale Unterschriften System ist sicher (gegen Chosen Message Angriffe), wenn für jeden effizienten Angreifer gilt, dass dessen Wahrscheinlichkeit, das Spiel zu gewinnen, vernachlässigbar klein ist.

Beispiel

(„naiver“ Einsatz von RSA):
Schlüsselerzeugung: SK=(n, d), PK=(n, e).
Unterschrift unter die Nachricht $m \in \mathbb{Z}_n^*$: $U = m^d \mod n$.
Verifizieren von (m, U): V(PK,m,U)=OK $\Leftrightarrow$ $U^e \equiv m \mod n$.

Sogar wenn $q = 0$ gilt, kann man dieses Digitale Unterschriften System „knacken“. (Wie?)

Hash-Then-Sign Unterschriften

Um Nachrichten beliebiger Länge zu unterschreiben, verwendet man meistens eine kryptographische Hashfunktion
$H : \{0,1\}^* \longrightarrow \{0,1\}^L$.
Statt der Nachricht M unterschreibt man dann tatsächlich den Hashwert $H(M)$.

Sicherheit von H

Eine *notwendige* Sicherheitsbedingung ist, dass die Hashfunktion kollisionsresistent ist, d.h., dass es praktisch unmöglich ist, zwei Inputs $M \neq M'$ zu finden mit $H(M) = H(M')$.
(Warum ist diese Bedingung notwendig?)

Für jede Hashfunktion H mit L Ausgabebits kann man mit einem Rechenaufwand von etwa $2^{L/2}$ Rechenoperationen Kollisionen finden.
(Wie und Warum?)

Beispiel: RSA PKCS #1, Vers. 1.5

Seien $k = \lceil \log_2(n)/8 \rceil$ und $\lambda = \lceil L/8 \rceil$. Um eine Nachricht M zu unterschreiben, wird der λ-Byte Wert $H = H(M)$ berechnet und zu einem k-Tupel m von Bytes ergänzt.

$$m \;=\; (\; 00 \;||\; 01 \;||\; FF \;\; ||\; FF \;||\; \cdots \;||\; FF \;||\; 00 \;||\; H \;).$$

Es besteht m (in dieser Reihenfolge) aus

1. zwei konstanten Bytes 00 und 01,
2. $k - \lambda - 3$ konstanten Bytes FF,
3. einem konstanten Byte 00 und
4. dem λ-Byte Hashwert $H = H(M)$.

Beispiel (Fortsetzung)

Man beachte bei PKCS #1:

- Bei Digitalen Unterschriften beginnt der Einbettungsstring mit „$00 \,||\, 01$“.
- Bei Public-Key Verschlüsselung beginnt er mit „$00 \,||\, 02$“.

RSA-FDH "Full Domain Hash"

- Schlüsselerzeugung: PK=(n, e), SK=(n, d).
  Hashfunktion $H : \{0,1\}^* \to \mathbb{Z}_n^*$.
- $S((n, d), M) = H(M)^d \mod n$.
- $V((n, e), M', U) = 1 \Leftrightarrow U^e \equiv H(M') \mod n$.

Theorem

Wenn das Berechnen e-ter Wurzeln mod n hart ist, dann ist RSA-FDH im Zufallsorakel-Modell sicher.

Public-Key Infrastrukturen (PKI)

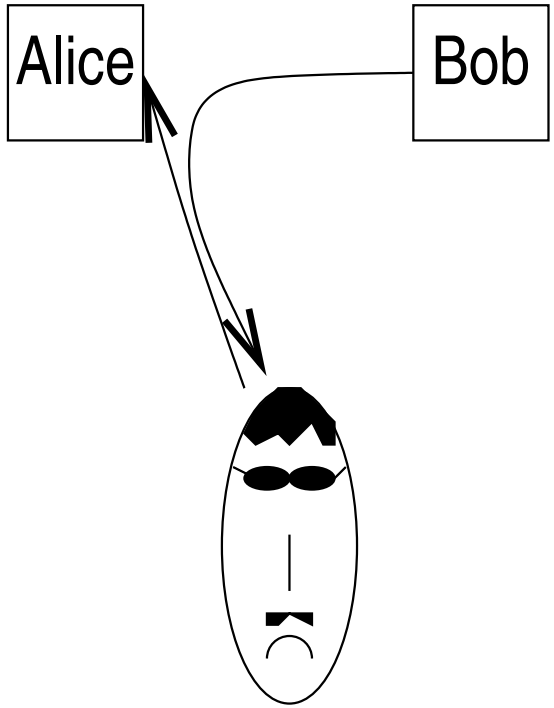
Man stelle sich vor, Alice und Bob haben sich nie getroffen und sind in einer der folgenden Situationen:

- Alice will Bob eine verschlüsselte EMail schicken.
- Alice und Bob wollen einen Vertrag, den sie über das Internet ausgehandelt haben, unterschreiben.

Wie gelangt Alice an den öffentlichen Schlüssel von Bob?

Bedrohung

- Da (noch) kein authentischer öffentlicher oder geheimer Schlüssel bekannt ist, muss Bob PK_{Bob} über einen unsicheren Kanal verschicken.
- Der Finsterling ersetzt PK_{Bob} durch seinen eigenen öffentlichen Schlüssel PK_F.



Eine Public-Key Infrastruktur regelt die Speicherung und Verteilung von öffentlichen Schlüsseln. Sie soll vor derartigen "Man-in-the-Middle" Angriffen schützen.

Zertifikate

Im Allgemeinen beruhen Public-Key Infrastrukturen auf dem Ausstellen von Zertifikaten.
Ein Zertifikat für Bob, ist ein Dokument, das (mindestens) die folgenden Informationen enthält:

- Bobs Benutzernamen (ggf. sein Pseudonym) und
- den öffentlichen Schlüssel PK_{Bob} von Bob, einschließlich einer Angabe des von Bob verwendeten Kryptosystems.

Eine dritte Person (z.B. Zacharias) kann Bob ein Zertifikat ausstellen, d.h., ein solches Dokument unterschreiben.

Zertifikate (2)

Bob kann das Zertifikat weitergeben, z.B. an Alice (über den unsicheren Kanal).

- Wenn Alice ihrerseits Zacharias *vertraut* und den öffentlichen Schlüssel von Zacharias kennt, dann wird sie $\mathrm{PK}_{\mathrm{Bob}}$ verwenden.
- Wenn Alices *Vertrauen* berechtigt ist, dann ist der Schlüssel $\mathrm{PK}_{\mathrm{Bob}}$ authentisch.

Verteilen von Zertifikaten

Die Grasswurzelmethode (z.B. PGP "Web of Trust"):

- Jeder Teilnehmer kann Zertifikate ausstellen.
- Jeder Teilnehmer legt für sich fest, bis zu welchem Maß er anderen Teilnehmern, deren öffentlichen Schlüssel er kennt, *vertraut*.

Die Zentralstruktur (z.B. Signaturgesetz):

- Nur die *Wurzelinstanz* stellt Zertifikate aus.
- Alle Teilnehmer müssen der Wurzelinstanz vertrauen.

Danksagungen

Danksagungen

- Nach einer Vorlesung von Stefan Lucks
- Erstellt mit Freier Software