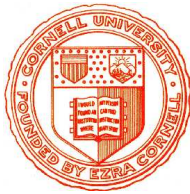


Kryptographie und Komplexität



Einheit 6.2

Digitale Signaturen



1. Sicherheitsanforderungen
2. RSA Signaturen
3. ElGamal Signaturen

WOZU UNTERSCHRIFTEN?

- **Verbindliche Urheberschaft von Dokumenten**

- Unterschrift zeigt an, daß Unterzeichnender das Dokument verfaßt oder zur Kenntnis genommen und zugestimmt hat
- Unterschrift muß auf dem Originaldokument geleistet werden um Bindung zwischen Dokument und Unterzeichnendem herzustellen

- **Signatur digitaler Dokumente**

- Wichtige elektronische Dokumente müssen Unterschriften tragen
 - Internet-Kaufverträge, elektronische Finanztransaktionen
 - Gerichtsunterlagen, verbindliche email-Zusagen
- Digitale Signatur soll gleiche Sicherheit wie Unterschrift anbieten
 - Fälschungssicher, nicht zu kopieren, nicht zu leugnen

- **Protokoll ähnlich zu Public-Key Kryptographie**

- Urheber erzeugt privaten, geheimen Signaturschlüssel
- Urheber hinterlegt Schlüssel zur Überprüfung seiner Signatur
- Urheber benutzt privaten Schlüssel um Dokumente zu unterzeichnen
- Empfänger verifiziert Signatur mit öffentlichem Schlüssel

- **Klartexte**

- Endliche Menge \mathcal{P} von Wörtern eines Alphabets
- Dokumente, die unterzeichnet werden sollen

- **Signaturen**

- Endliche Menge \mathcal{A} von Wörtern eines Alphabets
- Unterschriften, die an ein Dokument angehängt werden

- **Schlüssel**

- Endliche Menge \mathcal{K} möglicher Schlüssel
- Daten, die essentiell für Codierung und Decodierung sind

- **Signatur- und Verifikationsfunktionen**

- Funktionen $sig_K: \mathcal{P} \rightarrow \mathcal{A}$ und $ver_K: \mathcal{P} \times \mathcal{A} \rightarrow \{t, f\}$ für jeden Schlüssel $K \in \mathcal{K}$ mit der Eigenschaft $ver_K(x, s) = t \Leftrightarrow s = sig_K(x)$ für jeden Klartext $x \in \mathcal{P}$

Möglichst wenig Erfolg für die mächtigste Attacke

● Mögliche Attacken

- **Key only attack**: Angreifer kennt nur den öffentlichen Schlüssel
- **Known signature attack**: Angreifer kennt mehrere Paare von Dokumenten und zugehörigen Signaturen eines Signierers
- **Chosen message attack**: Angreifer kann im Voraus für beliebig viele selbstgewählte Dokumente eine gültige Signatur bekommen
- **Adaptive chosen message attack**: Angreifer kann auch während der Attacke Signaturen auf alle Dokumente seiner Wahl bekommen
- Auch möglich: **Angriff auf Schlüsselverwaltung**

● Mögliche Ergebnisse eines Angriffs

- **Existential forgery**: ein neues gültiges Dokument/Signatur-Paar
- **Selective forgery**: gültige Signatur zu einzelnen neuen Dokumenten
- **Universal forgery**: gültige Signatur zu jedem beliebigen Dokument
- **Total break**: Angreifer bestimmt geheimen Schlüssel des Signierers

Einfache “Umkehrung” des RSA Kryptosystems

● Schlüsselerzeugung

- Generiere n als Produkt zweier großer Primzahlen p und q
- Erzeuge zufälliges e mit $\gcd(e, \varphi(n))=1$ und berechne $d = e^{-1} \bmod \varphi(n)$
- Gesamtschlüssel ist $K := (n, p, q, d, e)$, wobei nur e, n öffentlich

● Erzeugung der Signatur

- Textblock wird als Binärdarstellung einer Zahl x interpretiert
- Signiere durch Potenzieren mit Geheimschlüssel: $\mathit{sig}_K(x) = x^d \bmod n$
- Für lange Texte und größere Sicherheit sind Optimierungen erforderlich

● Verifikation

- Empfangene Signatur s wird mit öffentlichem Schlüssel potenziert

$$\mathit{ver}_K(x, s) = x \equiv s^e \bmod n$$

- **Naives Protokoll ist zu einfach**

- Verifikation berechnet unterschriebenes Dokument aus Signatur
- Homomorphieeigenschaft erlaubt Zusammensetzen gültiger Signaturen

- **Existentielle Fälschung mit key only Attacke**

- Angreifer erzeugt Zufallszahl $s \in \{0, \dots, n-1\}$
- Angreifer erzeugt fiktive Nachricht $x = s^e \bmod n$
- s ist gültige Signatur von x

Verhinderbar durch Einfügen von Redundanz vor Unterzeichnung

- **Universelle Fälschung mit chosen message Attacke**

- Angreifer wählt Nachricht x und erzeugt Zufallszahl $s \in \{0, \dots, n-1\}$
- Angreifer berechnet $x' = s^e \cdot x \bmod n$
- Angreifer läßt x' signieren und erhält $s' = x'^d \bmod n$
- Angreifer berechnet $sig = s' \cdot s^{-1} \bmod n$
- Es folgt $sig \equiv x'^d \cdot s^{-1} \equiv (s^e)^d \cdot x^d \cdot s^{-1} \equiv x^d \bmod n$

● Signatur großer Dokumente

- Signatur aller Textblöcke des Dokuments x ist zu aufwendig
- Signiere Hashwert $h(x)$ für eine kollisionsresistente Hashfunktion h

$$\mathit{sig}_K(x) = h(x)^d \bmod n$$

- Verifikation vergleicht mit Hashwert des empfangenen Dokuments

$$\mathit{ver}_K(x, s) = h(x) \equiv s^e \bmod n$$

● Sicher gegen existentielle Fälschung

- Verifikation ermöglicht keine Rekonstruktion des gesamten Dokuments da die Hashfunktion auch eine Einwegfunktion ist
- Angreifer kann gültiges Paar (x, s) nicht zufällig erzeugen

● Homomorphieeigenschaft geht verloren

- Für kollisionsrestientes h darf $h(x_1 \cdot x_2) \equiv h(x_1) \cdot h(x_2) \bmod n$ nicht gelten, da man sonst für viele Hashwerte ein Urbild bestimmen könnte
- Universelle Fälschung mit chosen message Attacke nicht mehr möglich

Mehr als einfache Umkehrung des Kryptosystems

● Schlüsselerzeugung

- Wähle große Primzahl p und ein erzeugendes Element g von \mathbb{Z}_p^*
- Wähle ein zufälliges $a \in \{0, \dots, p-2\}$ und berechne $A = g^a \bmod p$
- Wähle eine kollisionsresistente Hashfunktion h
- Gesamtschlüssel ist $K := (p, g, h, a, A)$, wobei p, g, h, A öffentlich

● Erzeugung der Signatur

- Wähle zufälliges $r \in \{0, \dots, p-2\}$ mit $\gcd(r, p-1)=1$ und berechne $b = g^r \bmod (p-1)$ sowie $s = r^{-1}(h(x) - a \cdot b) \bmod (p-1)$
- Erzeugte Signatur ist $\mathit{sig}_K(x, r) = (b, s)$

● Verifikation

- Empfänger berechnet $\mathit{ver}_K(b, s) = A^b \cdot b^s \equiv g^{h(x)} \bmod p$
- Korrektheit: $A^b \cdot b^s \equiv g^{a \cdot b} \cdot (g^b)^{b^{-1}(h(x) - a \cdot b)} \equiv g^{a \cdot b} \cdot g^{h(x) - a \cdot b} \equiv g^{h(x)} \bmod p$

- **Zufallswert r muß sich ändern**

- Sonst hat Angreifer zwei signierte Dokumente $((x_1, (b, s_1)), (x_2, (b, s_2)))$
- Wegen $b=g^r$ ist $s_1 - s_2 = r^{-1}(h(x_1) - h(x_2)) \bmod (p-1)$
- Sind $s_1 - s_2$ oder $h(x_1) - h(x_2)$ invertierbar modulo $p-1$
dann bestimmt der Angreifer r und berechnet anschließend
den geheimen Schlüssel mit der Formel $a \equiv b^{-1}(h(x) - s_1 \cdot r) \bmod (p-1)$

- **Kollisionsresistente Hashfunktion ist notwendig**

- Ansonsten ist eine existentielle Fälschung durchführbar
- Hierzu wähle $u, v \in \mathbb{Z}$ mit $\gcd(v, p-1) = 1$ und setze
 $b = g^u \cdot A^v \bmod p$, $s = (-b) \cdot v^{-1} \bmod (p-1)$ und $x = s \cdot u \bmod (p-1)$
- Es folgt $A^b \cdot b^s \equiv A^b \cdot g^{u \cdot s} \cdot A^{v \cdot s} \equiv A^b \cdot g^x \cdot A^{-b} \equiv g^x \bmod p$
- Damit ist $(x, (b, s))$ ein gültig signiertes Dokument

DER DIGITAL SIGNATURE ALGORITHM (DSA)

- **NIST Variante der ElGamal Signaturen**

- Feste Blockgröße und enge Grenzen für Wahl der Parameter

- **Schlüsselerzeugung**

- Wähle 160-Bit Primzahl q und weitere Primzahl p mit $512+t \cdot 64$ Bit für ein $t \leq 8$ so daß q Teiler von $p-1$

- Wähle erzeugendes Element g_0 von \mathbb{Z}_p^* und setze $g = g_0^{(p-1)/q} \bmod p$

- Wähle ein zufälliges $a \in \{1, \dots, p-2\}$ und berechne $A = g^a \bmod p$

- Wähle eine kollisionsresistente Hashfunktion h

- Gesamtschlüssel ist $K := (p, q, g, h, a, A)$, wobei nur a geheim bleibt

- **Erzeugung der Signatur**

$$\mathit{sig}_K(x, r) = (b, s)$$

- Wähle zufälliges $r \in \{0, \dots, p-2\}$ und berechne

$$b = (g^r \bmod (p-1)) \bmod q \text{ sowie } s = r^{-1}(h(x) + a \cdot b) \bmod q$$

- **Verifikation**

- Berechne $\mathit{ver}_K(b, s) = b \equiv ((g^{s^{-1} \cdot h(x)} \bmod q \cdot A^{b \cdot s^{-1} \bmod q}) \bmod p) \bmod q$

- Korrektheit: $g^{s^{-1} \cdot h(x)} \bmod q \cdot A^{b \cdot s^{-1} \bmod q} \equiv g^{s^{-1} \cdot (h(x) + a \cdot b)} \equiv g^k \equiv r \bmod p$

- **Weitere Varianten von ElGamal Signaturen**
 - **Schnorr Signaturen**: ElGamal Schema mit 1024-Bit Schlüsseln und kurzen (160-Bit) Signaturen
 - **ECDSA**: Digitaler Signatur Algorithmus auf Basis elliptischer Kurven
- **Unabstreitbare Signaturen**
 - Schutz gegen Kopieren von Signaturen
 - Absender wird an Verifikation der Signatur beteiligt
 - “Disavowal”-Protokoll ermöglicht falsche Signaturen abzustreiten
- **Fail-Stop Signaturen**
 - Schutz gegen Unterschriftsfälschung durch extrem mächtige Angreifer
 - Protokoll enthält neben Signature- und Verifikationsalgorithmus ein Verfahren zum Nachweis von Fälschungen
- **Gruppensignatur**
 - Unterschrift identifiziert Gruppe von Personen statt Einzelpersonen
 - Nur die Gruppe kann Identität des Unterzeichners feststellen