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

Signaturschemata: Grundbestandteile

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

Sicherheitsanforderungen

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

RSA Signaturen

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: $\boldsymbol{sig_K(x)} = x^d \bmod n$
  - Für lange Texte und größere Sicherheit sind Optimierungen erforderlich
- **Verifikation**
  - Empfangene Signatur s wird mit öffentlichem Schlüssel potenziert

$$\boldsymbol{ver_K(x, s)} = x \equiv s^e \bmod n$$

Angriffe auf RSA Signaturen

- **Naives Protokoll ist zu einfach**
  - Verifikation berechnet unterschriebenes Dokument aus Signatur
  - Homomorphieeigenschaft erlaubt Zusammensetzen gültiger Signaturen
    $sig_K(x \cdot x') \;=\; x^d \cdot x'^d \bmod n \;=\; sig_K(x) \cdot sig_K(x')$
- **Existentielle Fälschung mit key only Attacke**
  - Angreifer erzeugt Zufallszahl $s \in \{0, .., n-1\}$
  - Angreifer erzeugt fiktive Nachricht $x = s^e \bmod n$
    s ist gültige Signatur von x (aber x ist meist keine sinnvolle Nachricht)
- **Universelle Fälschung mit chosen message Attacke**
  - Angreifer wählt Nachricht x und erzeugt Zufallszahl $s \in \{0, .., n-1\}$
  - 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 $sig \equiv x'^d \cdot s^{-1} \equiv (s^e)^d \cdot x^d \cdot s^{-1} \equiv x^d \bmod n$

Einfache RSA Signaturen sind schnell, aber unsicher

RSA Signatur mit Hashwerten

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

$$\boldsymbol{sig_K(x)} = h(x)^d \bmod n$$

  - Verifikation vergleicht mit Hashwert des empfangenen Dokuments

$$\boldsymbol{ver_K(x,s)} = h(x) \equiv s^e \bmod n$$

- **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 kollisionsresistentes 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

ElGamal Signaturen über $\mathbb{Z}_p$

Mehr als einfache Umkehrung des Kryptosystems

- **Schlüsselerzeugung**
  - Wähle große Primzahl p und ein erzeugendes Element g von $\mathbb{Z}_p^*$
  - Wähle ein zufälliges $a \in \{0, .., p-2\}$ 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 $r \in \{0, .., p-2\}$ mit $gcd(r, p-1)=1$ und berechne $b = g^r \bmod p$ sowie $s = r^{-1}(h(x)-a \cdot b) \bmod (p-1)$
  - Erzeugte Signatur ist $\boldsymbol{sig_K(x, r)} = (b, s)$
- **Verifikation**
  - Empfänger berechnet $\boldsymbol{ver_K(x, (b, s))} = A^b \cdot b^s \equiv g^{h(x)} \bmod p$
  - Korrektheit: $A^b \cdot b^s \equiv g^{a \cdot b} \cdot (g^r)^{r^{-1}(h(x)-a \cdot b)} \equiv g^{a \cdot b} \cdot g^{h(x)-a \cdot b} \equiv g^{h(x)} \bmod p$

ElGamal Signaturen am Beispiel

- **Schlüsselerzeugung**
  - Alice wählt $p = 23$, $g = 7 \in \mathbb{Z}_p^*$, $a = 6$
  - Alice berechnet $A = 7^6 \bmod 23 = 4$
  - Der Einfachheit halber wählt Alice die Identität als Hashfunktion
    id ist zwar formal kollisionsresistent aber keine Einwegfunktion !
  - Gesamtschlüssel ist $K := (23, 7, id, 6, 4)$, wobei $23, 7, id, 4$ öffentlich
- **Erzeugung der Signatur**
  - Alice will $x = 13$ signieren und wählt $r = 9$. Dann ist $gcd(r, 22)=1$.
  - Alice berechnet $b = 7^9 \bmod 23 = 15$
    sowie $s = 9^{-1}(id(13) - 6 \cdot 15) \bmod 22 = 5 * (-77) \bmod 22 = 11$
  - Erzeugte Signatur ist $(15, 11)$
- **Verifikation**
  - Empfänger prüft $4^{15} \cdot 15^{11} \equiv 7^{id(13)} \bmod 23$ d.h. $3 \cdot 22 \bmod 23 = 20$

Sicherheitsaspekte von ElGamal Signaturen

- **Die Einweg-Hashfunktion ist notwendig**

  Ansonsten ist eine existentielle Fälschung durchführbar

  – Hierzu wähle $u, v \in \mathbb{Z}$ mit $gcd(v, p-1) = 1$ und setze

  $b = g^u \cdot A^v \bmod p, \quad s = (-b) \cdot v^{-1} \bmod (p-1)$

  und $x = h^{-1}(s \cdot u) \bmod (p-1)$

  – Es folgt $A^b \cdot b^s \equiv A^b \cdot g^{u \cdot s} \cdot A^{v \cdot s} \equiv A^b \cdot g^{h(x)} \cdot A^{-b} \equiv g^{h(x)} \bmod p$

  – Damit ist $(x, (b, s))$ ein gültig signiertes Dokument

- **Der Zufallswert r darf nur einmal verwendet werden**

  – Sonst hat Angreifer zwei signierte Dokumente $((x_1, (b, s_1)), \ (x_2, (b, s_2))$

  und da b sich nicht ändert, gilt $s_1 - s_2 = r^{-1}(h(x_1) - h(x_2)) \bmod (p-1)$

  – 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 $a \equiv b^{-1}(h(x) - s_1 \cdot r) \bmod (p-1)$

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·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 $a \in \{1, .., p-2\}$ 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** $\boldsymbol{sig_K(x, r)} = (b, s)$
  - Wähle zufälliges $r \in \{0, .., p-2\}$ und berechne $b = (g^r \bmod p) \bmod q$ sowie $s = r^{-1}(h(x)+a \cdot b) \bmod q$
- **Verifikation**
  - Berechne $\boldsymbol{ver_K(b, s)}$: $b \equiv ((g^{s^{-1} \cdot h(x) \bmod q} \cdot A^{b \cdot s^{-1} \bmod q}) \bmod p) \bmod q$
  - Korrektheit: $g^{s^{-1} \cdot h(x) \bmod q} \cdot A^{b \cdot s^{-1} \bmod q} \equiv g^{s^{-1} \cdot (h(x)+a \cdot b)} \equiv g^r \equiv b \bmod p$

Digital Signature Algorithm am Beispiel

- **Schlüsselerzeugung**
  - Alice wählt $q = 739$ und $p = 2957 = 4{\cdot}q{+}1$
  - Alice wählt $g_0 = 2003$ und berechnet $g = g_0^4 \bmod 2957 = 1534$
  - Alice wählt $a = 512$ und berechnet $A = 1534^{512} \bmod 2957 = 99$
  - Alice verzichtet auf die Hashfunktion h
  - Alice veröffentlicht als Schlüssel $(2957, 739, 1534, id, 99)$
- **Erzeugung der Signatur** $\boldsymbol{sig_K(x,r)} = (b, s)$
  - Alice will $x = 333$ signieren und wählt $r = 444$.
  - Alice berechnet $b = (1534^{444} \bmod 2957) \bmod 739 = 440$

    sowie $s = 444^{-1}(id(333){+}512{\cdot}440) \bmod 739 = 248{\cdot}218 \bmod 739 = 117$
- **Verifikation**
  - Empfänger überprüft die Gleichung

    $440 \;=\; ((1534^{117^{-1}\cdot 333 \bmod 739}{\cdot}99^{440\cdot 117^{-1} \bmod 739}) \bmod 2957) \bmod 739$

Weitere Signaturschemata

- **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