Diffie-Hellman-Schlüsselaustauschprotokoll

Problem: Austausch eines Schlüssels für ein Private-Key Verfahren über eine unsichere Leitung

Lösung: Diffie-Hellman Protokoll, basiert auf dem zahlentheoretischen Problem der diskreten Logarithmen

Definition: Sei p eine Primzahl und g eine Primitivwurzel $\mod p$. Dann gibt es für jede Zahl $A \in \{1, \ldots, p-1\}$ ein $a \in \{0, \ldots, p-2\}$ mit

$$A \equiv g^a \mod p.$$

a heißt *diskreter Logarithmus* von A zur Basis g.

Protokoll: Alice und Bob wollen Schlüssel K tauschen

- Einigung auf Primzahl p und Primitivwurzel $g \mod p$
- Alice wählt $a \in \{0, \ldots, p-2\}$ zufällig, berechnet
$$A = g^a \mod p$$
- Bob wählt $b \in \{0, \ldots, p-2\}$ zufällig, berechnet
$$B = g^b \mod p$$
- Austausch von A und B über unsichere Leitung
- gemeinsamer Schlüssel ist
$$K = A^b \mod p = B^a \mod p = g^{ab} \mod p$$

Sicherheit des Protokolls

Diffie-Hellman-Problem: Berechnung von

$$K = A^b \mod p = B^a \mod p = g^{ab} \mod p,$$

wenn nur p, g, A und B, jedoch nicht a und b bekannt sind

Es gilt

$$\text{Berechnung diskreter Logarithmen} \mod p$$
$$\Longrightarrow$$
$$\text{Berechnung Diffie-Hellman-Problem}$$

Bis jetzt ist kein effizientes Verfahren zur Berechnung diskreter Logarithmen bekannt.

Weitere Angriffsmöglichkeit: Man-In-The-Middle-Attack

- Angreifer gibt sich als Alice bzw. Bob aus
- tauscht Schlüssel K_A mit Alice, K_B mit Bob
- Alice schickt Bob eine Nachricht $c = E_{K_A}(m)$, Angreifer fängt diese ab, entschlüsselt sie mit K_A, verschlüsselt mit K_B und sendet $c' = E_{K_B}(m)$ an Bob

Digitale Signaturen

Problem: Nachricht m soll signiert werden, d.h. der Urheber von m ist zweifelsfrei zu bestimmen

Lösung: Elektronische Signatur durch Public-Key Verfahren

Vorgehensweise: (mit RSA)

- erzeuge öffentlichen Schlüssel (n, e), privaten Schlüssel d
- erzeuge Signatur s für Nachricht $m \in \{0, \ldots, n-1\}$

$$s = m^d \mod n$$

- Kommunikationspartner verifiziert $s^e \mod n = m$

Schwachstelle: Multiplikativität von RSA

- Angreifer möchte m von Alice signiert haben
- wähle m_1 mit $ggT(m, m_1) = 1$
- berechne m_2 mit $m_1 m_2 = m \mod n$
- Texte m_1 und m_2 von Alice signieren lassen

$$s_1 = m_1^d \mod n, \quad s_2 = m_2^d \mod n$$

- berechne $s = s_1 s_2 \mod n$, dann gilt

$$s = s_1 s_2 \mod n = (m_1 m_2)^d \mod n = m^d \mod n$$

- Angriff ist vom Typ Chosen-Message-Attack

Vorkehrungen

1. Lösung: Benutze Redundanzfunktion

Beispiel: Wähle als Redundanzfunktion $R(m) = m \circ m$

Signiert wird dann zum Text m der Wert $m \circ m$:

$$s = (m \circ m)^d \mod n$$

Vorteil: Selbst wenn m_1 und m_2 von der Form

$$m_1 = (x \circ x) \text{ und } m_2 = (y \circ y)$$

sind, so hat mit hoher Wahrscheinlichkeit $m = m_1 m_2 \mod n$ nicht diese Struktur

2. Lösung: Signatur mit Hashfunktion

Erinnerung: Hashfunktion h ist eine Abbildung

$$h : \Sigma^* \to \Sigma^n, n \in \mathbb{N}$$

- h *Einwegfunktion*, falls es für ein gegebenes $s \in \Sigma^n$ praktisch unmöglich ist, ein x mit $h(x) = s$ zu finden
- *Kollision* ist ein Paar (x, x') mit $x \neq x'$ und $h(x) = h(x')$
- h *schwach kollisionsresistent* falls es für ein gegebenes x praktisch unmöglich ist, eine Kollision (x, x') zu finden
- h *stark kollisionsresistent* falls es praktisch unmöglich ist, irgendeine Kollision (x, x') zu finden

Signatur mit Hashwert

Vorgehensweise:

- Alice signiert beliebig langen Text x mittels bekannter kollisionsresistenter Hashfunktion
$$h : \{0,1\}^* \to \{0,\ldots,n-1\}$$
- Alice berechnet Signatur $s = h(x)^d \mod n$, versendet s zusammen mit Text x
- Bob erhält Paar (s, x), berechnet $m = s^e \mod n$
- Bob berechnet $h(x)$, falls dieser Wert mit m übereinstimmt, ist s eine gültige Signatur für x von Alice

Datenintegrität

Idee: Schütze Daten vor unbefugter Änderung mit Hilfe einer Hashfunktion h:

- berechne Hashwert der Daten x, erhalte $y = h(x)$
- speichere y in sicherer Umgebung (Chipkarte)
- beim Abrufen der Daten x bilde Hashwert $y' = h(x)$ und vergleiche diesen Wert mit y
- wenn h schwach kollisionsresistent ist, so kann kein x' mit $h(x') = h(x) = y$ erzeugt werden

Identifikation

Einfachste und verbreiteste Lösung: Passwörter

- wähle geheimes Passwort x, Einwegfunktion h
- in zentralem Verzeichnis wird $h(x)$ gespeichert
- bei Identifikation wird Passwort x versendet, Hashwert $h(x)$ gebildet und mit gespeichertem Wert verglichen

Vorteil: Passwortdatei muß nicht vor Lesezugriffen geschützt sein, weil h eine Einwegfunktion ist und somit praktisch kein Urbild von $h(x)$ berechnet werden kann.

Nachteil 1: Passwörter muß man sich merken. Einfach zu merkende Passwörter (z.B. Namen) können per brute-force Methode (Wörterbuch und entsprechende Hashwerte) ermittelt werden.

Nachteil 2: Passwortdatei muß vor Schreibzugriffen geschützt sein. Ansonsten kann Hashwert $h(x)$ durch $h(x')$ mit bekanntem Wert x' ersetzt werdem.

Nachteil 3: Passwort kann während der Übermittlung ausspioniert werden, falls es im Klartext übertragen wird.

Challenge-Response-Protokoll für Identifikation

Protokoll für Challenge-Response Identifikation:

- Bob schickt eine Aufgabe (Challenge) an Alice
- Alice löst die Aufgabe mittels eines nur ihr bekannten Geheimnisses
- Alice teilt Bob die Antwort (Response) mit
- Lösung korrekt $\rightarrow$ Identität von Alice verifiziert

Umsetzung:

Private-Key Verfahren:

- Geheimnis ist Schlüssel k
- Bob erzeugt Zufallszahl r, schickt diese an Alice
- Alice berechnet $r' = E_k(r)$, schickt r' an Bob
- Bob verifiziert $D_k(r') = r$

Public-Key Verfahren (RSA):

- Geheimnis ist privater Schlüssel d
- Bob erzeugt Zufallszahl r, schickt diese an Alice
- Alice signiert r durch $s = r^d \mod n$, schickt s an Bob
- Bob verifiziert $s^e \mod n = r$

Zero-Knowledge Protokolle

Idee: Alice überzeugt Bob von der Kenntnis des Geheimnis, ohne dass Bob etwas über das Geheimnis erfährt

Unterschied zu Public-Key Identifikation:
Mathematische Beweisbarkeit, dass Bob während des Protokolls keine Informationen über das Geheimnis erhält

Beispiel: Fiat-Shamir Protokoll

Vorbereitung:

- wähle Primzahlen p und q
- berechne $n = pq$
- wähle s aus Menge $\{1, \ldots, n-1\}$ mit $ggT(s, n) = 1$
- berechne $v = s^2 \mod n$
- öffentlicher Schlüssel ist (v, n)
- geheimer Schlüssel ist s

Bob will Alice beweisen, dass er die Quadratwurzel s von $v \mod n$ kennt, ohne diese preiszugeben

Mathematische Anlehnung an Rabin Chiffriersystem, bei dem man Wurzeln $\mod n$ berechnen muß

Fiat-Shamir Protokoll

Identifikationsprotokoll:

- Bob wählt $r \in \{1, \ldots, n-1\}$
- Bob berechnet $x = r^2 \mod n$, sendet x an Alice
- Alice wählt Bit $e \in \{0, 1\}$, sendet e an Bob (Challenge)
  - Fall 1: $e = 0$. Bob schickt r an Alice, Alice verifiziert $r^2 = x \mod n$ (Response)
  - Fall 2: $e = 1$. Bob schickt Zahl y mit $y = rs \mod n$ an Alice, Alice verifiziert $y^2 \equiv xv \mod n$ (Response)
- da Bob Quadratwurzel s von $v \mod n$ kennt, kann er beide Fragen korrekt beantworten

Frage: Welche Angriffsmöglichkeiten gibt es?

Fall 1: Wähle r, berechne $x = r^2 \mod n$. Dann kann man auf das Bit $e = 0$ richtig antworten. y ist jedoch nicht bekannt, ansonsten könnte $s = yr^{-1} \mod n$ berechnet werden.

Fall 2: Wähle y, berechne $x = y^2 v^{-1} \mod n$. Dann kann man auf das Bit $e = 1$ richtig antworten, jedoch nicht auf $e = 0$.

Analyse: Bei jedem Durchgang liegt die Betrugswahrscheinlichkeit bei $1/2$. Nach k Runden des Protokolls liegt die Betrugswahrscheinlichkeit bei $1/2^k$.

Sicherheit + Beispiel Fiat-Shamir Protokoll

Sicherheit: In beiden Fällen $e = 0$ und $e = 1$ muß eine Quadratwurzel $\mod n$ bestimmt werden.

$$\text{Bestimmung von Quadratwurzeln} \mod n$$
$$\Longleftrightarrow$$
$$\text{Faktorisierung von } n$$

Beispiel - Vorbereitung:

- wähle $p = 5$ und $q = 13$, dann ist $n = pq = 65$
- wähle $s = 23$, berechne $v = 23^2 \mod 65 = 9$
- öffentlicher Schlüssel ist $(9, 65)$
- geheimer Schlüssel ist 23

Identifikationsprotokoll:

- wähle $r = 19$
- berechne $x = 19^2 \mod 65 = 36$, sende x an Alice
- Alice wählt Bit $e \in \{0, 1\}$
  - Fall 1: $e = 0$. Schicke $r = 19$ an Alice, Alice verifiziert $19^2 = 36 \mod 65$
  - Fall 2: $e = 1$. Berechne Zahl y mit der Eigenschaft $y = 19 \cdot 23 \mod 65 = 47$, sende y an Alice, Alice verifiziert $47^2 \equiv 36 \cdot 9 \mod 65 = 64$

Public-Key-Infrastrukturen

Aufgabe einer Public-Key-Infrastruktur (**PKI**):

- Verteilung und Speicherung der öffentlichen und privaten Schlüssel aller Teilnehmer
- Schutz der öffentlichen Schlüssel vor Fälschung
- Geheimhaltung der privaten Schlüssel

Aufgabe einer persönlichen Sicherheitsumgebung (**PSE**):

- Speicherung (m.E. Erzeugung) des privaten Schlüssels
- Entschlüsselung von Chiffretexten
- Nachrichten signieren

Umgebung für eine PSE:

- geschützter Bereich auf der Festplatte (Software-PSE)
  Nachteil: hängt von Sicherheit des Betriebssystems ab
- Chipkarte (Hardware-PSE)
  Nachteil: Berechnungen ineffizient
  Lösung: verwende Hybrid-Verfahren

Hybrid-Verfahren:

- verschlüssele Text mit sym. Verfahren (z.B. DES)
- Sitzungsschlüssel wird per asym. Verfahren verschlüsselt

Zertifizierungsstellen

Aufgaben einer Certification-Authority (**CA**)

- Registrierung der Benutzer, Verifikation der persönlichen Daten (Prüfung der Identität) → Zertifikatsausstellung
- Erzeugung der Schlüssel (m.E.), Übertragung der privaten Schlüssel in die PSE des Benutzers
- Verwaltung der Schlüssel zum Signieren, Identifizieren und Verschlüsseln
- Rückruf von Zertifikaten
- Archivierung abgelaufener Schlüssel
- Verteilung der öffentlichen Schlüssel, CA bürgt mit ihrer digitalen Signatur für dessen Korrektheit

<u>Zusatz: Erweiterter Friedman-Test</u>

Erinnerung: Berechne Schlüssellänge h durch

$$h = \frac{0.0377n}{I * (n-1) - 0.0385n + 0.0762},$$

wobei n die Länge des Textes und I der Friedmansche Koinzidenzindex ist.

Verbesserung: Berechne die h Caesar-Verschiebungen der Subtexte $X_1, X_2, \ldots, X_h$ direkt mit Hilfe der Formel:

$$M_g = \sum_{i=0}^{25} \frac{p_i n_{i+g}}{n'} \quad , \quad 0 \leq g \leq 25.$$

Dabei ist

- p_i Standardhäufigkeit des i-ten Buchstabens im deutschen Alphabet
- n_{i+g} Häufigkeit des i-ten Buchstabens im entschlüsselten Subtext, falls Schlüssel g benutzt wurde
- n' Länge des Subtextes

Für den richtigen Schlüssel g sollte M_g nahe bei 0.0762 liegen, für alle anderen Schlüssel knapp über dem Wert für die Gleichverteilung 0.0385