Digital Signature Algorithm

Luc Spachmann

FSU Jena

14.01.2021

Allgemeines

- Signaturalgorithmus
- Basiert auf Diskreten Logarithmen
- Definiert im 'Digital Signature Standard' (DSS)
- Benötigt globale Parameter p, q, g und Hashfunktion $hash$
- q ist Primzahl der Länge N
- p ist Primzahl der Länge L und darstellbar als $p = kq + 1$
- g ist ein Element aus $\mathbb{Z}_p^*$ mit Ordnung q
- Erlaubte Längen (L, N) sind:
  $(1024, 160), (2048, 224), (2048, 256), (3072, 256)$
- Zusätzlich muss gelten $N \leq |hash|$
- Hier: $(L, N) = (1024, 160), hash =$ sha-1

Generierung der Parameter

- Primzahl q der Länge L generieren
- Teste für passende k ob $p = qk + 1$ Primzahl ist
- Dabei k gewählt, sodass $qk + 1$ gewünschte Länge hat
- Falls p nicht Prim, anderes k probieren
- Ggf. auch anderes q wählen
- Wähle zufällig $1 < h < p - 1$
- Berechne $g = h^{\frac{p-1}{q}} \mod p$
- Falls $g = 1$, anderes h wählen
- Berechnetes g hat gewünschte Eigenschaft

Schlüssel

- Benötigt zusätzlich geheimen Schlüssel x und öffentlichen y
- x zufällig mit $1 < x < q$
- $y = g^x \mod p$
- Schlüssel werden nur von einer Person verwendet
- Globale Parameter können wiederverwendet werden

Signieren

- Benötigt Parameter p, q, g, x
- Signieren nur Hashwert $hash(m)$ der Nachricht m
- Benötigt für jede Nachricht unabhängiges $1 < j < q$
- Berechne $r = (g^j \mod p) \mod q$
- Falls $r = 0$ neues j wählen
- Berechne $s = j^{-1} \cdot (hash(m) + rx) \mod q$
- Falls $s = 0$ ebenfalls neues j wählen
- j muss geheim bleiben
- Tupel (r, s) ist Signatur
- j^{-1} ist modulo q
- $hash(m)$ wieder als Big-Endian Zahl interpretieren

Verifizieren

- Benötigt Parameter p, q, g, y
- Verifizieren, ob $0 < r < q$ und $0 < s < q$
- Berechne $w = s^{-1} \mod q$
- Berechne $u_1 = hash(m) \cdot w \mod q$
- Berechne $u_2 = rw \mod q$
- Berechne $v = (g^{u_1} y^{u_2} \mod p) \mod q$
- Verifiziere ob $v = r$

Aufgaben

- Implementiert Parametergenerierung
- Implementiert Signatur und Verifikation