Betriebssysteme und Sicherheit – Sicherheit

Public-Key Kryptographie

Modulringe

Wir betrachten den endlichen Körper

$$Z_p \ (p > 2)$$

Modulringe sind abgeschlossen bzgl. Addition und Multiplikation (falls p prim)

- Neutrales Element
  - $a + 0 = a, \ a \cdot 1 = a$
- Inverses
  - $a + (-a) = 0, \ a \cdot a^{-1} = 1$

Equivalenzrelation

$a, a+n, a+2n, a+3n, \ldots a+cn \equiv a \pmod{n}$

Beispiel:

$2, 9, 16, 23, \ldots -5, -12 \equiv 2 \pmod{7}$

Restklasse a (mod n):

Alle Zahlen, deren Rest bei Division durch n a ist

Repräsentant: a ($0 \leq a < n$)

Rechnen mit Restklassen

$a + b = c \Leftrightarrow (a \bmod n) + (b \bmod n) \equiv (c \bmod n)$

Beispiel: $5 + 7 = 12$

(mod 9) $5 + 7 \equiv 3$

(mod 11) $5 + 7 \equiv 1$

Gilt auch für Inverse:

(mod 9) $5 \equiv 3 - 7$

Gilt auch für Multiplikation

$a \cdot b = c \Leftrightarrow (a \bmod n) \cdot (b \bmod n) \equiv (c \bmod n)$

Beispiel: $3 \cdot 4 = 12$

(mod 7) $3 \cdot 4 \equiv 5$

(mod 11) $3 \cdot 4 \equiv 1$

Gilt auch für Inverse:

(mod 7) $3 \equiv 5 \cdot 4^{-1}$

Beachte Inverse sind auch Elemente in Z_p !

$3 \equiv 5 \cdot 2 \qquad (4 \cdot 2 \equiv 1)$

Exponentation

$a^b \equiv c \pmod{n}$

$\underbrace{a \cdot a \cdot a \cdot \ldots \cdot a}_{\text{b mal}} \equiv c \pmod{n}$

Schwere Probleme

Es sei f eine Funktion mit Eingabe x:

$y = f(x)$ leicht zu berechnen

aber

$x = f^{-1}(y)$ sehr schwer zu berechnen

Beispiel – Faktorisierung

Gegeben

- Primzahlen p und q
- $f(p, q) = pq = n$
- $f^{-1}(n)$ sehr schwer zu berechnen

Falls n 1024 Bit hat, braucht ein 1 Mio EUR Computer ca. 1 Jahr

Beispiel – Diskreter Logarithmus

Gegeben

- Restklassen g, x in Z_p
- $f(g, x) = g^x \pmod p = y$
- $f^{-1}(g, y)$ sehr schwer zu berechnen

Falls p 1024 Bit und x 160 Bit braucht ein 1 Mio EUR Computer ca. 1 Jahr

Schlüsselverteilungsproblem

Symmetrischer Schlüssel erlaubt

- Verschlüsseln, Überprüfen von MAC

Aber auch

- Entschlüsseln, Berechnen von MAC

Public-Key Kryptographie

Öffentlicher Schlüssel:

pubKey

Privater Schüssel:

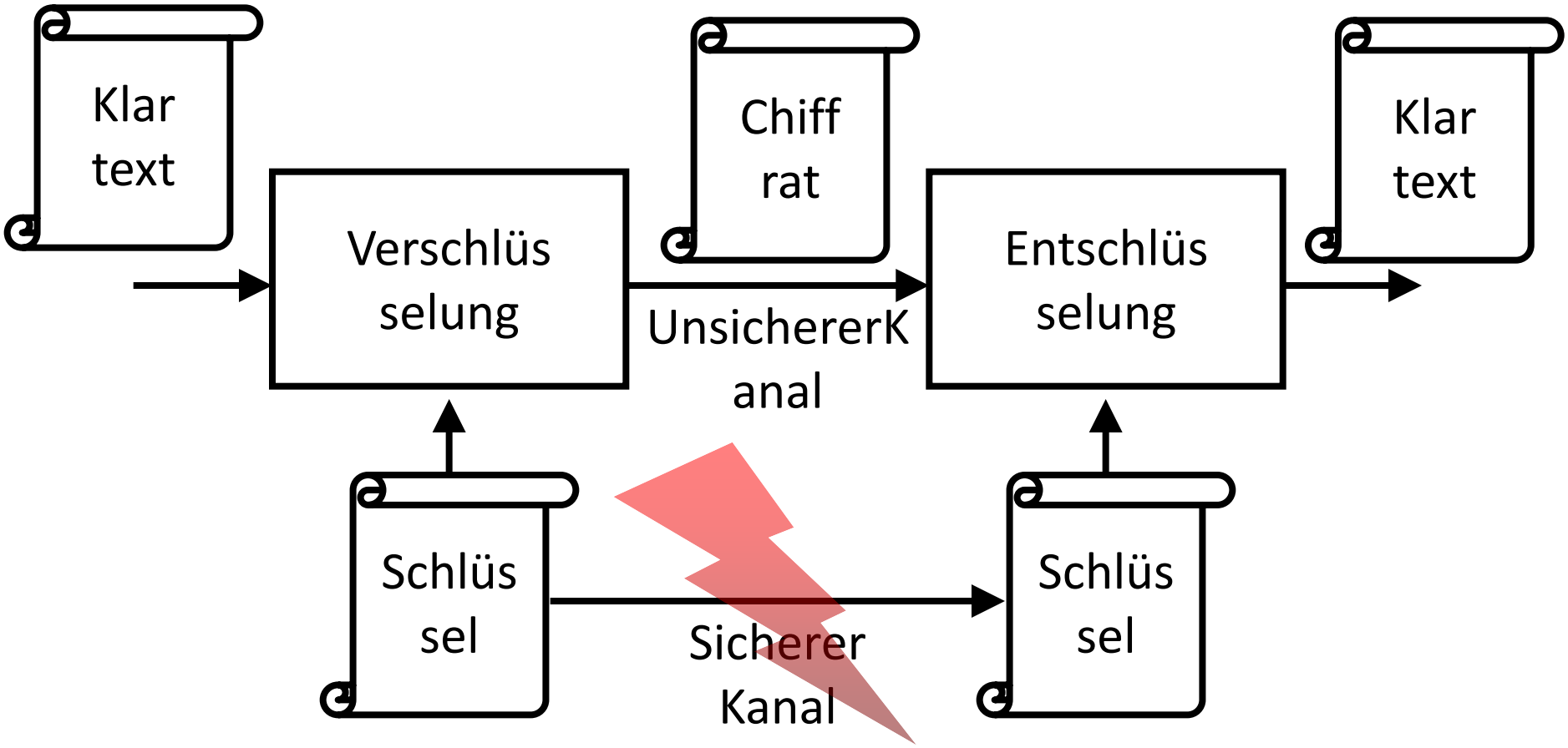
privKey

Es gilt:

pubKey = f(privKey) – leicht

privKey = f^{-1}(pubKey) – schwer

Schlüsselaustausch



Modulare Exponentiation

- $a^b = c \pmod n$
- a^b hat $b \cdot \log a$ Bits ($> \log n$)
- Daher ungünstig
  - Erst $a^b = c$ berechnen
  - Dann c mod n berechnen
- Frage: Kann man den Modulus ausnutzen?

Modulare Exponentiation

- Einfacher Algorithmus

  c = 1

  for i = 1 to b

    c = c*a (mod n) (max. 2 log n Bits)

Schnelle Modulare Exponentiation

Nutze Bit-Repräsentation von b: $b_{n-1}...b_1b_0$

$$c = a^{2^0 b_0 + 2^1 b_1 + \ldots + 2^{n-1} b_{n-1}}$$

$$c = a^{2^0 b_0} \cdot a^{2^1 b_1} \cdot \ldots \cdot a^{2^{n-1} b_{n-1}}$$

$$c = \prod_{i=0}^{n-1} a_i^{b_i}$$

$$a_0 = a$$

$$a_i = a_{i-1}^2$$

Algorithmus

```
c = 1
while b > 0
    if (b & 1)
        c = c * a (mod n)
    a = a * a (mod n)
    b = b >> 1
```

Euklidischer Algorithmus

Der Euklidische Algorithmus dient dazu den größten gemeinsamen Teiler zu berechnen

$a, b \rightarrow ggT(a, b)$

Der Erweiterte Euklidische Algorithmus kann zu dem das Inverse $b^{-1} \pmod a$ berechnen, falls $ggT(a, b) = 1$

EEA

Gegeben: a, b (wlog a > b)

1. Berechene $\lfloor a/b \rfloor$ und c = a (mod b)
   Schreibe
   $a = \lfloor a/b \rfloor \cdot b + c$
2. Falls c = 0, dann ggT(a,b) = b und der Algorithmus bricht ab
   Falls c = 1, dann zum 2. Teil
3. Setze a = b und b=c und wiederhole 1.

EEA – 2.Teil

1. Stelle Gleichungen um, so dass
   $c = a - \lfloor a/b \rfloor b$
   (Im ersten Schritt c = 1)
2. Falls es keine vorherige Gleichung gibt, ist der Algorithmus fertig
   Ersetze b durch vorherige Gleichung
3. Löse Faktoren nach a und b auf
4. Gehe zu 2.

Beispiel – 1.Teil

99 und 26

$99 = 3 \cdot 26 + 21$

$26 = 1 \cdot 21 + 5$

$21 = 4 \cdot 5 + \mathbf{1}$

Beispiel – 2.Teil

$$\mathbf{1} = 21 - 4 \cdot 5 \qquad (21 = 4 \cdot 5 + \mathbf{1})$$

$$5 = 26 - 1 \cdot 21 \qquad (26 = 1 \cdot 21 + 5)$$

$$21 = 99 - 3 \cdot 26 \qquad (99 = 3 \cdot 26 + 21)$$

$$\mathbf{1} = 21 - 4 \cdot (26 - 1 \cdot 21)$$

$$\mathbf{1} = -4 \cdot 26 + 5 \cdot 21$$

$$\mathbf{1} = -4 \cdot 26 + 5 \cdot (99 - 3 \cdot 26)$$

$$\mathbf{1} = 5 \cdot 99 \boxed{- 19} \cdot 26$$

Fermat und Euler

Fermat: $a^p = a \pmod p$

Euler:

Let $n = p^a q^b \ldots$

$\phi(n) = p^{a-1}(p-1)q^{b-1}(q-1)\ldots$

$a^{\phi(n)} = 1 \pmod n$

Konsequenz

$b \equiv c \pmod{\phi(n)}$

$a^b \equiv a^c \pmod{n}$

oder

$a^b = a^{b \bmod \phi(n)}$

Trickfrage

Was ist $14213^{69122} \pmod 7$?

Faktorisierung

$n = pq$

$\phi(n) \rightarrow n$	leicht zu berechnen
$n \rightarrow \phi(n)$	wahrscheinlich sehr schwer zu berechnen

RSA

Privater Schlüssel:

$\phi(n)$

Öffentlicher Schlüssel:

n, e

RSA – Verschlüsselung

Verschlüsselung von Nachricht m

$c = m^e \pmod{n}$

Entschlüsselung

$ed = 1 \pmod{\phi(n)}$ - EEA

$m = c^d \pmod{n}$

Beispiel

$p = 7, q = 11 \quad \Rightarrow n = 77$

$\Rightarrow \phi(n) = 60$

$e = 7, m = 3$

Verschlüsselung

$c = m^e \pmod n = 3^7 \pmod{77} = 31$

Entschlüsselung

EEA(60, 7) $\Rightarrow d = 43$

$m = c^d \pmod n = 31^{43} \pmod{77} = 3$