Faire Public–Key Kryptosysteme

nach einem Paper von Silvio Micali[1]

Olaf Bonorden

24. Juni 1999

[1] Laboratory for Computer Science, Massachusetts Institute of Technologie

Überblick

- Einleitung
- Informale Beschreibung
- Diskussion der Eigenschaften
- konkrete Konstruktionen von Fairen Public–Key Kryptosystemen
  - Schlüsselaustausch nach Diffie–Hellmann
  - RSA
- Erweiterungen des Standard–Schemas

Kryptographie und Verbrechen

Problem: Verbrecherische Organisationen können Kryptographie für ungestörte Kommunikation nutzen

Lösungsideen: In der aktuellen Diskussion meist zwei Vorschläge:

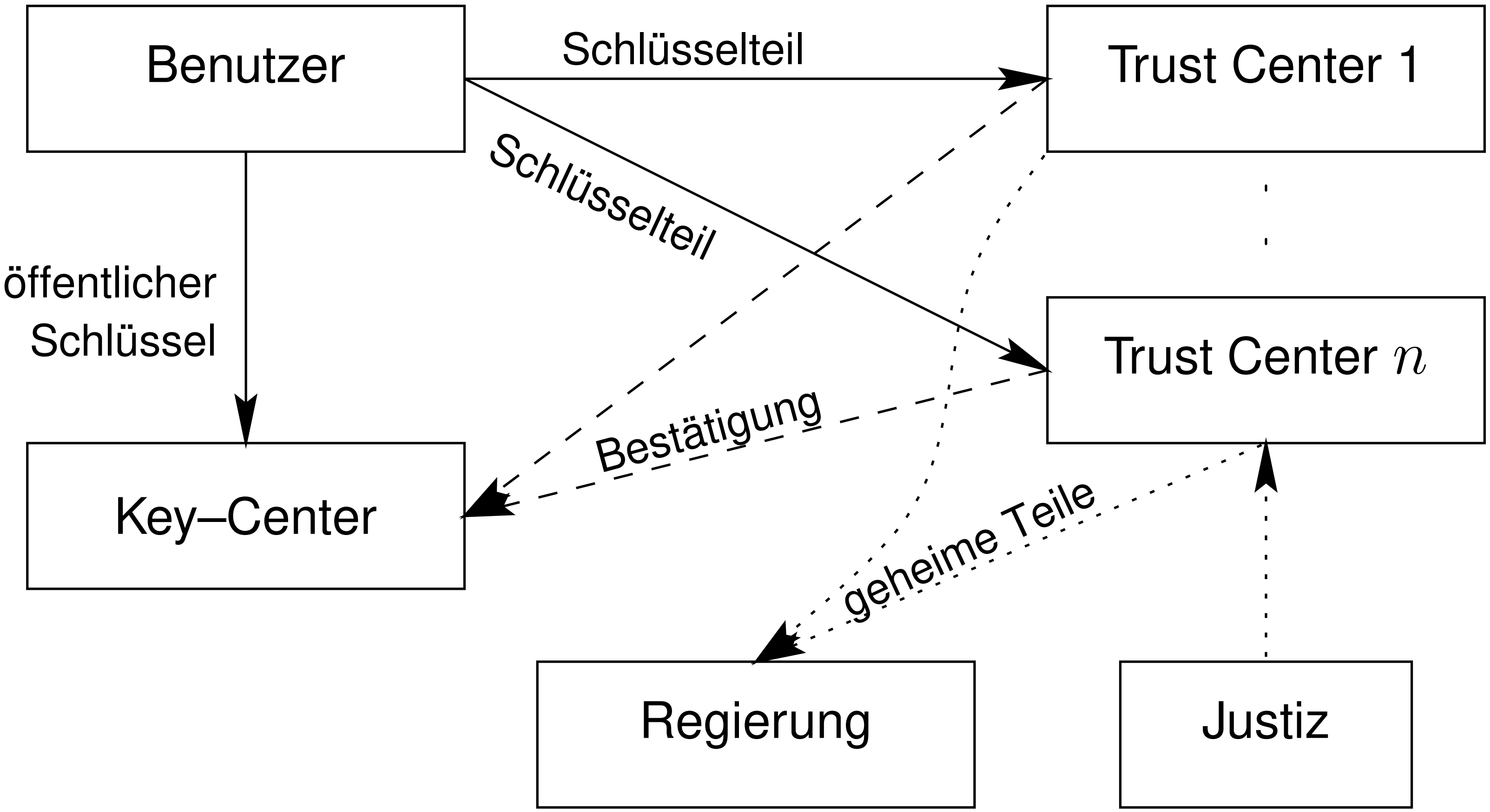
1. Benutzung *schwacher* Kryptosysteme
2. „Verraten“ des geheimen Schlüssels an den Staat

Beide Vorschläge haben Nachteile.

Besser: Sichere Schlüsselhinterlegung bei mehreren Trust Centern

$\leadsto$ Faire Kryptosysteme

Faire Public–Key Kryptosysteme



Vergleich mit ähnlichen Konzepten

Teilen von Geheimnissen

- x_0 sei Geheimnis
- wähle zufälliges Polynom f mit Grad $< n$ und $f(0) = x_0$
- verteile Stützstellen $(x_i, f(x_i))$, $1 \leq i \leq n$ an n Trust Center

$\leadsto$ n Trust Center zusammen können das Polynom f durch Interpolation eindeutig bestimmen, weniger als n haben *keine* Information über $f(0)$

Problem: kein Trust Center kann überprüfen, ob $(x_i, f(x_i))$ Teil des Geheimnisses ist, d.h. ob $f(0)$ der Schlüssel ist.

Erzeugung von Schlüsseln

Jeder **Benutzer**

- erzeugt ein Schlüsselpaar aus geheimen und öffentlichem Schlüssel
- teilt den geheimen Schlüssel in n Teile und überträgt jeden Teil an ein Trust Center

Jedes **Trust Center**

- überprüft, ob es einen Teil des geheimen Schlüssel erhalten hat (**ohne** den geheimen Schlüssel zu kennen)
- signiert den öffentlichen Schlüssel und schickt ihn an das Key Center

Key Center

- empfängt signierte öffentliche Schlüssel und signiert den öffentlichen Schlüssel, wenn alle n Trust Center ihn signiert haben

Kommunikation

- Abholen des öffentlichen Schlüssels eines anderen Benutzers beim Key Center
- normale Anwendung des Public–Key Protokolls

Abhören

Nach Beschluß eines Richters:

- Anfordern des geheimen Teils des Schlüssels von jedem Trust Center
- Berechnung des geheimen Schlüssels
- Dekodieren des Textes

Eigenschaften

- Sicherheit des Public–Key Systems bleibt erhalten, falls nicht alle Trust Center korrupt sind
- Effizienz:
  - Schlüsselgenerierung hat polynomiellen Rechenaufwand
  - Benutzung analog dem Public–Key Kryptosystem
- beliebige, private Kommunikation bis ein gerichtlich bestätigter Verdacht vorliegt
- gleiche Rechte wie im Telefonnetz
- im Gegensatz zu anderen Methoden höhere Akzeptanz bei Benutzern (?)

Schlüsselaustausch nach Diffie–Hellman

Öffentlich: Zyklische Gruppe G mit erzeugendem Element $g \in <g> = G$

Alice

- wählt zufällig $0 \leq s_A < |G|$
- berechnet $a = g^{s_A}$
- schickt a an Bob
- berechnet $K = b^{s_A}$

Bob

- wählt zufällig $0 \leq s_B < |G|$
- berechnet $b = g^{s_B}$
- schickt b an Bob
- berechnet $K = a^{s_B}$

$\Rightarrow \; K = g^{s_A s_B}$ ist der gemeinsame, geheime Schlüssel

Bemerkung: Analog kann ein Public Key Kryptosystem konstruiert werden

($\rightsquigarrow$ ElGamal Kryptosystem)

Faires Diffie–Hellman Schema

(n Trust Center, alle benötigt)

Benutzer: • geheim: $S_{x_1}, \ldots, S_{x_n} \in \{1, \ldots, p-1\}$ zufällig,

$$S_x \equiv \sum_{i=1}^{n} S_{x_i} \bmod (p-1)$$

- öffentlich: $T_i \equiv g^{S_{x_i}} \bmod p$, $P_x \equiv g^{S_x} \bmod p$
- schickt (verschlüsselt) (S_{x_i}, T_i) an Trust Center i

Trust Center i: • testet, ob $g^{S_{x_i}} \equiv T_i \bmod p$

- speichert (P_x, S_{x_i})
- schickt (P_x, T_i) signiert an Key Center

Key Center: empfängt T_i und signiert P_x, falls $P_x \equiv \prod_{i=1}^{n} T_i \bmod p$

Analyse des Fairen Diffie–Hellman Schemas

Sicherheit

- Das Wissen von $n-1$ Werten S_i hilft nicht zur Berechnung von

$$S_x = \log_g g^{S_x} = \log_g g^{S_{x1}+\ldots+S_{xn}}$$

- Key Center: Wissen von n Werten T_i hilft nicht

Abhören

- geheimer Schlüssel $S_x = \sum_{i=1}^{n} S_{x_i} \bmod (p-1)$, da gilt

$$g^{S_{x1}+\ldots+S_{xn}} \equiv \prod_{i=1}^{n} T_i \equiv P_x \equiv g^{S_x} \bmod p$$

RSA

Schlüsselgenerierung

- p, q prim, $N = pq$, $\varphi(N) = (p-1)(q-1)$ (Eulers φ–Funktion)
- $e \in \{2, \ldots, \varphi(N) - 2\}$ zufällig gleichverteilt mit $\mathrm{ggT}(e, \varphi(N)) = 1$
- Berechnung von $d \in \mathbb{Z}_{\varphi(N)}$ mit $ed \equiv 1 \bmod \varphi(N)$ (mittels EEA)
- $K = (N, e)$ öffentlicher, $S = (N, d)$ geheimer Schlüssel

Verschlüsseln

$$\mathrm{enc}_K(x) = x^e \bmod N$$

Entschlüsseln

$$\mathrm{dec}_S(y) = y^d \bmod N$$

Aufteilen des Schlüssels

Idee: Faktorisieren von N ermöglicht Berechnung des geheimen Schlüssels e

Dixons Zufallquadrate: $x^2 \equiv y^2 \bmod N \Rightarrow (x-y)(x+y) \equiv 0 \bmod N$

$\Rightarrow \mathrm{ggT}((x \pm y), N)$ nichttrivaler Teiler von N oder $x \equiv -y \bmod N$

$\leadsto$ Benutzer verteilt als Geheimnis zwei Wurzeln einer Zahl $\bmod N$

Problem: Benutzer könnte x und x so wählen, daß $x \equiv -y \bmod N$ gilt

Lösung: $x \equiv -y \bmod N$ kann mittels des Jacobi–Symbols überprüft werden

Exkurs Zahlentheorie

Lemma 1 *Seien p, q prim, $N = pq$. Dann gilt:*

1.
$$\left.\begin{array}{l} s \in \mathbb{Z}_N^\times \text{ ist ein} \\ \text{Quadrat mod } N \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} \exists x, y \in \mathbb{Z}_N : x, -x, y, -y \text{ paarweise} \\ \text{verschieden und } x^2 \equiv y^2 \equiv s \bmod N \end{array}\right.$$

2. *Ein Viertel der Elemente von $\mathbb{Z}_N^\times$ sind Quadrate modulo N*

Definition 1 *Seien $a, b \in \mathbb{N}$. Dann ist das* Jacobisymbol $\left(\frac{a}{b}\right)$ *definiert als*

1. *Falls $b = p$ prim:* $\left(\frac{a}{p}\right) := \begin{cases} 0, & \mathrm{ggT}(a,p) \neq 1 \\ +1, & a \text{ ist ein Quadrat modulo } p \\ -1, & \text{sonst} \end{cases}$

2. *Falls $b = \prod_{i=1}^{r} p_i^{e_i}$ gilt, so ist* $\left(\frac{a}{b}\right) := \prod_{i=1}^{r} \left(\frac{a}{p_i}\right)^{e_i}$

Lemma 2 *Seien p, q prim mit $p \equiv q \equiv 3 \bmod 4$, $N = pq$, $s \in \mathbb{Z}_N^\times$ ein Quadrat mit Wurzeln x, y. Dann gilt o.B.d.A.:*

$$\left(\frac{x}{N}\right) = \left(\frac{-x}{N}\right) = 1 \text{ und } \left(\frac{y}{N}\right) = \left(\frac{-y}{N}\right) = -1$$

$\Rightarrow$ mit $s \equiv x^2 \equiv y^2 \bmod N$ und $\left(\frac{x}{N}\right) \neq \left(\frac{y}{N}\right)$ kann N faktorisiert werden

Faires RSA

Benutzer

- wählt Primzahlen p, q mit $p \equiv q \equiv 3 \bmod 4$, $N = pq$
- wählt zufällig n Zahlen x_i mit $\forall 1 \leq i \leq n : \left(\frac{x_i}{N}\right) = 1$
- berechnet $X = \prod_{i=1}^{n} x_i$, für $1 \leq i \leq n$ die Quadrate $x_i^2 \bmod N$ und $Z = X^2 = \prod_{i=1}^{n} (x_i^2)$
- berechnet Y mit $Y^2 \equiv Z \bmod N$ und $\left(\frac{Y}{N}\right) = -1$
- für $1 \leq i \leq n$: veröffentlicht x_i^2 und schickt x_i verschlüsselt an Trust Center i
- schickt Y an Key Center

Trust Center i

- testet, ob $\left(\frac{x_i}{N}\right) = 1$
- berechnet x_i^2
- speichert (x_i, x_i^2) und schickt x_i^2 signiert an Key Center

Key Center

- testet, ob $\left(\frac{-1}{N}\right) = 1$ ($\Rightarrow \forall x : \left(\frac{x}{N}\right) = \left(\frac{-x}{N}\right)$)
- empfängt Y und alle x_i^2
- testet, ob $Y^2 = \prod_{i=1}^{n} (x_i^2)$
- testet, ob N ein Produkt zweier Primzahlen ist
- signiert öffentlichen Schlüssel des Benutzers

Beispiel

Benutzer

- Generierung des Schlüssels

```
>> q := 11; p := 19;
>> isprime(q), q mod 4, isprime(p), p mod 4;
         TRUE, 3, TRUE, 3

>> N := p*q;

>> phi_N := (p-1)*(q-1);
>> e := 103; d := e^(-1) mod phi_N;
```

Benutzer (Fortsetzung)

- Aufteilen des Geheimnisses

```
>> x[1] := 9^2 mod N;
         81
>> x[2] := 54^2 mod N;
        199
>> K[1] := x[1]^2 mod N;
         82
>> K[2] := x[2]^2 mod N;
        100
>> X := x[1]*x[2] mod N;
         26
>> Z := X^2 mod N;
         49
```

Benutzer (Fortsetzung)

- Berechnung einer Wurzel Y von Z mit $\left(\frac{Y}{N}\right) = 1$

```
>> wurzeln := numlib::msqrts( Z, N );
       [7, 26, 183, 202]
>> Y := 0;
>> i:=1;
>> while numlib::jacobi(wurzeln[i], N)<> -1 do
>>    i := i+1;
>> end_while;
>> Y := wurzeln[i];
        7
```

Trust Center

```
>> for i from 1 to 2 do
>>   if ( numlib::jacobi(x[i],N)=1 ) then
>>     print( "Trust Center ", i, "ok" );
>>   end_if;
>> end_for;

     "Trust Center ", 1, "ok"

     "Trust Center ", 2, "ok"
```

Key Center

- Test, ob $\left(\frac{-1}{N}\right) = 1$

```
>> numlib::jacobi( -1, N );
      1
```

- Test, ob Schlüssel korrekt zerlegt

```
>> Y^2 = K[1]*K[2] mod N;
       49 = 49
```

- Test, ob N in genau zwei Primfaktoren zerfällt
  (Primzahltests, Test auf Primpotenz, Anteil der Quadrate in $\mathbb{Z}_N^\times$)

Rekonstruktion des geheimen Schlüssels

- Zusammensetzen der geheimen Teile

```
>> X := x[1]*x[2] mod N;
         26
```

- Berechnung der Faktorisierung von N aus den Wurzeln von Z

```
>> p := igcd( X-Y, N );
         19
>> q:= N div p;
         11
```

Zusammenfassung

Faire Kryptosysteme

- sicher und effizient
- Einsatz kein mathematisches, sondern ein gesellschaftliches Problem

Erweiterungen

- weniger Trust Center zur Rekonstruktion benötigt
- Trust Center erkennen nicht, wessen Schlüssel sie verraten
- zeitbeschränktes Überwachen

Und wenn Kriminelle andere Kryptosysteme benutzen?

1. Szenario: andere Kryptosysteme verboten

Jeder Benutzer nicht–fairer Kryptosysteme kann zumindest für den Gebrauch illegaler Kryptographie vor Gericht gestellt werden

2. Szenario: alle Kryptosysteme erlaubt

- Verteilung öffentlicher Schlüssel schwieriger
- kein einfacher Zugriff auf alternative Produkte
- wenn alle faire Kryptgraphie benutzen: hoher Aufwand für Kriminelle, eigene Systeme zu entwickeln

Literatur

[Mic92] MICALI, SILVIO: *Fair Public-Key Cryptosystems*. In BRICKELL, ERNEST F. (editor): *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 113–138. Springer-Verlag, 1993, 16–20 August 1992.