Faire Public–Key Kryptosysteme

Olaf Bonorden

Dieser Text beschreibt das in [Mic92] vorgestellte Konzept von Fairen Public–Key Kryptosystemen. In Fairen Public–Key Kryptosystemen wird der geheime Schlüssel des Benutzers sicher an mehrere Trust Center verteilt, so daß die Regierung die Kommunikation des Benutzers im Falle eines richterlichen Beschlusses abhören kann.

1 Public–Key Kryptosysteme

Eigenschaften von Public–Key Kryptosystemen:

- jeder Benutzer hat öffentlichen Schlüssel S und privaten Schlüssel K
- Nachrichten werden mit dem öffentlichen Schlüssel des Empfängers verschlüsselt
- Key Center speichert alle offentlichen Schlüssel („Telefonbuch“)
- Beispiele: ElGamal, RSA

Public–Key Kryptographie führt zu sicheren und praktischen Kryptosystemen.

Problem: Wie kann man verhindern, daß kriminelle Organisationen durch Kryptographie ungestört kommunizieren können? Aktuell sind vor allem zwei Lösungsvorschläge in der Diskussion:

1. schwache Kryptosystem, die die Polizei (aber auch jeder andere) brechen kann
2. Vorabübermittlung eines geheimem Master–Keys an die Behörden

Beide Lösungen haben deutlich Nachteile, gewünscht sind folgende Eigenschaften:

- jeder kann mit jedem privat kommunizieren
- Überwachung mit Beschluß eines Richters einfach möglich

Dies führt zu Fairen Kryptosystemen.

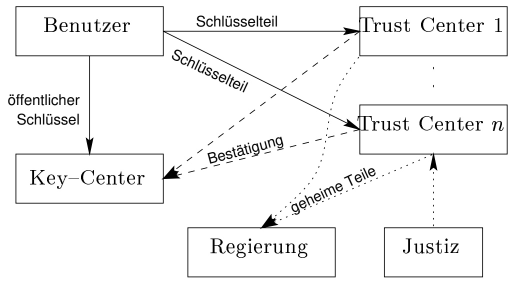

Abbildung 1: Übersicht Faire Public–Key Kryptosysteme

2 Faire Public–Key Kryptosysteme

2.1 Prinzip von Fairen Public–Key Kryptosystemen

Sei C ein verschlüsselter Text in einem Fairen Public–Key Kryptosystemen S. Dann kann ein Dritter (z.B. Behörde mit richterlicher Erlaubnis) entweder:

1. den Klartext von C im Kryptosystem S berechnen oder
2. beweisen, daß C nicht mit S verschlüsselt wurde.

2.2 Abstrakte Beschreibung

Im folgenden wird auf sehr abstrakte Art beschrieben, wie Faire Public–Key Kryptosystem konstruiert werden können.

Folgende Begriffe werden stellvertretend für entsprechende Einrichtungen benutzt:

Benutzer: Personen, die mit dem System verschlüsselte Nachrichen verschicken

Regierung: Darf nach einem richterlichen Beschluß (Justiz) die Kommunikation abhören

Trust Center: Einrichtungen, die Teile des geheimen Schlüssels speichert

Key Center: Sammelt öffentliche Schlüssels der Benutzer (vergl. Telefonbuch)

Im folgenden seien n Trust Center und beliebig viele Anwender gegeben. Jeder Anwender wählt sein Schlüsselpaar unabhängig und teilt seinen geheimen Schlüssel in n Teile mit folgenden Eigenschaften:

- der private Schlüssel kann mit dem Wissen aller n Teile berechnet werden
- mit $n-1$ oder weniger Teilen kann der Schlüssel nicht berechnet werden
- für jedes einzelne Teil kann unabhängig überprüft werden, ob es ein Teil des geheimen Schlüssels ist

Jeder Benutzer gibt jedem Trust Center einen Teil seines geheimen Schlüssels, im Falle eines richterlichen Beschlusses übergeben die Trust Center ihre geheimen Teile der Regierung, die daraus den geheimen Schlüssel berechnen kann (s.a. Abbildung 1).

2.3 Vergleich mit ähnlichen Konzepten

Das Problem, ein Geheimnis auf mehrere Personen zu verteilen, ist unter dem Namen *Secret Sharing* schon länger bekannt, dies kann zum Beispiel mittels eines Polynom–Interpolationsproblem gelöst werden:

- 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

$\Rightarrow$ 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.

Bei einer Erweiterung, dem *Verifiable Secret Sharing* (s. [CGMA86]), kann zwar überprüft werden, daß es ein Polynom durch die gegebenem Stützstellen gibt und somit $f(0)$ ein Geheimnis kodiert, allerdings fehlt auch hier der Beweis, daß es sich um $f(0)$ um den im Kryptosystem benutzten geheimen Schlüssel handelt. Hierzu werden *Nullwissen–Beweise* benötigt, in denen der Benutzer beweist, daß $f(0)$ sein geheimer Schlüssel ist, ohne den Schlüssel zu verraten.

Da die allgemeine Konstruktion mittels Verifiable Secret Sharing zwar theoretisch möglich, aber in der Praxis nicht mit vertretbarem Aufwand realisierbar ist, werden in dem Paper nur einige spezielle Beispiele (Schlüsselaustausch nach Diffie–Hellmann und RSA) beschrieben.

3 Eigenschaften von Fairen Public–Key Kryptosystemen

Sicherheit: Unter den Voraussetzungen, daß

- nicht alle Trust Center korrupt sind und
- die Regierung alle Teile des privaten Schlüssels nur nach einer richterlichen Anweisung erhält,

sind Faire Public–Key Kryptosysteme genauso sicher wie Public–Key Systeme.

Effizienz: Die Schlüsselgenerierung und –aufteilung auf die Trust Center ist in polynomieller Zeit möglich und nur einmal pro Benutzer erforderlich. Die Kosten für die Kommunikation entsprechen denen des Public–Key Systems, da sich Faire Systeme nur bei der Schlüsselerzeugung und beim Abhören von normalen Public–Key Kryptosystemen unterscheiden. Auch das Berechnen des geheimen Schlüssels aus den n Einzelteilen ist in polynomieller Zeit möglich.

An zusätzlicher Kommunikation fallen nur $O(n)$ zusätzliche Nachrichten an.

4 Faires Diffie–Hellmann Schema

4.1 Schlüsselaustausch nach Diffie–Hellman

In [DH76] wurde das erste Public–Key Kryptoverfahren eingeführt. Dieses Protokoll dient dem Austausch eines geheimen Schlüssels über eine unsichere Leitung und funktioniert folgendermaßen:

Alle Berechnungen finden in einer zyklische Gruppe G mit einem erzeugendem Element $g \in <g> = G$ statt.

Alice	**Bob**
1. wählt zufällig $0 \leq s_A < \|G\|$	1. wählt zufällig $0 \leq s_B < \|G\|$
2. berechnet $a = g^{s_A}$	2. berechnet $b = g^{s_B}$
3. schickt a an Bob	3. schickt b an Alice
4. berechnet $K = b^{s_A}$	4. berechnet $K = a^{s_B}$

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

Die Sicherheit beruht auf der Vermutung, daß das Berechnen von diskreten Logarithmen in bestimmten Gruppen schwierig ist.

4.2 Aufteilen des geheimen Schlüssels

Das folgende Schema benutzt n Trust Center, die alle zur Rekonstruktion benötigt werden. Dieses Schema kann so erweitert werden, daß $t < n$ Trust Center zur Rekonstruktion ausreichen.

Benutzer:

- generiert geheime Schlüsselteile $S_{x_1}, \ldots, S_{x_n} \in \{1, \ldots, p-1\}$ zufällig und geheimen Schlüssel $S_x \equiv \sum_{i=1}^{n} S_{x_i} \mod (p-1)$
- berechnet die entsprechenden öffentlichen Teile: $T_i \equiv g^{S_{x_i}} \mod p$ sowie $P_x \equiv g^{S_x} \mod p$

- schickt (verschlüsselt) (S_{x_i}, T_i) an Trust Center i

Trust Center i:

- testet, ob $g^{S_{x_i}} \equiv T_i \mod p$, d.h. ob man mit allen Teilen $\log_g \prod_{i=1}^{n} T_i$ berechnen kann
- speichert (P_x, S_{x_i}) (wird benötigt, falls der Benutzer abgehört werden soll)
- schickt (P_x, T_i) signiert an Key Center

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

4.3 Analyse des Fairen Diffie–Hellman Schemas

Sicherheit:

- Das Wissen von $n-1$ Werten S_i hilft nicht zur Berechnung von
$$S_x = \log_g g^{S_x} = \log_g g^{S_{x1}+\cdots+S_{xn}},$$
da diese S_{x_i} einfach zufällig gewählte Werte sind.
- Key Center: Wissen von allen n Werten T_i hilft nicht

Abhören: Der geheime Schlüssel läßt sich einfach berechnen:

$$S_x = \sum_{i=1}^{n} S_{x_i} \mod (p-1), \text{ da gilt } g^{S_{x1}+\cdots+S_{xn}} \equiv \prod_{i=1}^{n} T_i \equiv P_x \equiv g^{S_x} \mod p$$

5 Faires RSA

5.1 RSA

RSA ist ein in [RSA78] vorgestelltes Public–Key Kryptosystem, welches auf dem Problem der Faktorisierung großer Zahlen beruht.

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 $ed \equiv 1 \mod \varphi(N)$ (mittels Erweitertem Euklidischen Algorithmus)
- $K = (N, e)$ öffentlicher, $S = (N, d)$ geheimer Schlüssel

Verschlüsseln: $\mathrm{enc}_K(x) = x^e \mod N$

Entschlüsseln: $\mathrm{dec}_S(y) = y^d \mod N$

5.2 Erweiterung Faires RSA

Um die Kommunikation abzuhören, soll N faktorisiert werden (dann kann $d \equiv e^{-1} \mod \varphi(N)$ leicht berechnet werden). Ein Teiler von N kann gefunden werden, wenn zwei modulo N gleiche Quadrate $x^2 \equiv y^2 \mod N$ bekannt sind, da damit folgt: $N|(x^2 - y^2) = (x-y)(x+y)$, so daß entweder $\mathrm{ggT}(x \pm y)$ einen nichttrivialen Teiler von N liefert oder $x \equiv -y \mod N$ gilt (vergl. Faktorisierungsalgorithmus Dixons Zufallsquadrate, z.B. [vzGG99])

Um auszuschließen, daß der Benutzer x und y mit $x \equiv -y \mod N$ wählt, sind einige Eigenschaften aus der Zahlentheorie notwendig.

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

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

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

Aus diesem Lemma folgt insbesondere, daß jedes Quadrat zwei Wurzeln x, y mit $x \not\equiv -y \mod N$ hat, und daß es genügend viele Quadrate in $\mathbb{Z}_N^\times$ gibt.

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

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

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

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

Mit diesem Lemma folgt, daß N mit $s \equiv x^2 \equiv y^2 \mod N$ und $\left(\frac{x}{N}\right) \neq \left(\frac{y}{N}\right)$ faktorisiert werden kann. Also muß der Benutzer zwei Wurzeln eines Quadrates mit verschiedenen Jacobi–Symbolen berechnen. Eine Wurzel Y wird veröffentlicht, die andere wird geteilt an die Trust Center geschickt. Der genaue Ablauf ist wie folgt:

Benutzer:

- wählt Primzahlen p, q mit $p \equiv q \equiv 3 \mod 4$, $N = pq$
- wählt zufällig n Zahlen x_i mit $\forall 1 \leq i \leq n : \left(\frac{x_i}{N}\right) = 1$
- berechnet $X = \prod_{i=1}^{n} x_i$ und für $1 \leq i \leq n$ die Quadrate $x_i^2 \mod N$, weiterhin $Z = X^2 = \prod_{i=1}^{n} (x_i^2)$
  (X ist die private Wurzel von Z, die in x_i geteilt an die Trust Center verteilt wird)
- berechnet Y mit $Y^2 \equiv Z \mod N$ und $\left(\frac{Y}{N}\right) = -1$
  (dies ist die öffentliche Wurzel von Z)
- 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 $\left(\frac{x_i}{N}\right) = 1$, d.h. ob das zusammengefügte Geheimnis eine anderen Wurzel als Y und $-Y$ ist
- 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 $Y^2 = \prod_{i=1}^{n} (x_i^2)$, d.h. ob das Geheimnis richtig geteilt wurde
- testet, ob N ein Produkt zweier Primzahlen ist (Details s. [Mic92])
- signiert öffentlichen Schlüssel des Benutzers

5.3 Beispiel in MuPAD

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;
       209
>> phi_N := (p-1)*(q-1);
       180
>> e := 103;
>> d := e^(-1) mod phi_N;
        7
```

- Aufteilen des Geheimnisses

```
>> numlib::jacobi(113,N);   # Raten von x1 #
       -1
>> numlib::jacobi(81,N);
        1
>> x[1] := 81;
>> numlib::jacobi(199,N);   # Raten von x2 #
        1
>> x[2] := 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
```

- 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
```

Die Funktion `msqrts` berechnet zwar alle Wurzeln, dies ist jedoch nur effizient, wenn die Faktorisierung von N bekannt ist (ein Blick in den Quelltext zeigt das enthaltenen `ifactor`, d.h. die Funktion kann so nicht direkt benutzt werden, sondern muß so angepaßt werden, daß N nicht neu faktorisiert wird).

Trust Center:

- Überprüfung, ob ein Teil der geheimen Wurzel empfangen wurde

```
>> 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 $\mathbb{Z}_N^\times$, s. [Mic92])

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
```

6 Varianten des Standard–Schemas

Folgende Erweiterungen der Möglichkeiten von Fairen Public–Key Kryptosystemen werden in [Mic92] außerdem diskutiert:

- nicht alle Trust Center werden benötigt
- „nicht vertrauenswürdige Trust Center“, d.h. Trust Center erfährt nicht, zu welchem Benutzer der Teil gehört, den die Regierung anfordert
- zeitbeschränkte Abhörung

Literatur

[CGMA86] Chor, Benny, Shafir Goldwasser, Silvio Micali and Baruch Awerbuch: *Verifiable Secret Sharing and Archieving Simultaneity in the Presence of Faults.* In *Proceedings of the* 26^{th} *Annual IEEE Symposium of Foundations of Computer Science*, pages 383–395. IEEE, New York, 1986.

[DH76] Diffie, W. and M. Hellmann: *New Directions in Cryprographie.* IEEE Trans. Inform. Theory, IT-22, 6:644–654, November 1976.

[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.

[RSA78] Rivest, A., A. Shamir and L. Adleman: *A Method for Obtaining Digital Signatures and Public–Key Cryptosystems.* Communications of the ACM, 22(2):120–126, February 1978.

[vzGG99] Gathen, Joachim von zur and Jürgen Gerhard: *Modern Computer Algebra.* Cambridge University Press, 1999.