Das RSA-Verfahren

Michael Gendreyzig

18. März 1999

(Originalfassung 17. Juli 1995)

1. Einführung

Wir haben in vorangegangenen Vorträgen Theorie und Anwendung von Kryptoverfahren auf der Grundlage der projektiven Geometrie kennengelernt. Die folgende Stunde werden wir uns mit genau einem Kryptosystem - *dem RSA-Verfahren* - befassen, das sich hauptsächlich auf Ergebnisse der Zahlentheorie abstützt.

Dieses 1978 nach den Entdeckern
Ronald Rivest, Adi Shamir und Leonard Adleman benannte Verfahren hat folgende Klassifikationseigenschaften:
Es handelt sich um ein

- (asymmetrisches) Public-Key Kryptosystem

- mit trapdoor oneway-function (Einwegfunktion mit Falltür).

Dazu sei bemerkt:

In einem Public-Key Kryptosystem wird jeweils ein Schlüsselpaar pro Kommunikationspartner benötigt. Jedes dieser Schlüsselpaare besteht aus einem geheimen und einem öffentlichen Schlüssel. Eine Nachricht wird von dem Sender mit dem öffentlichen Schlüssel des Empfängers verschlüsselt und der Empfänger entschlüsselt die Nachricht mit seinem geheimen Schlüssel.

Basis der Public-Key Kryptosysteme sind sogenannte Einweg-Funktionen (engl. oneway-functions). Dies sind Funktionen, die ohne größeren Aufwand berechenbar sind, deren Inverse man aber, obwohl sie existieren, nicht ohne weiteres ermitteln kann. Die Problematik liegt dabei nicht im Finden eines Algorithmus, sondern im Aufwand zur Berechnung des Algorithmus. Einweg-Funktionen können zur Verschlüsselung eingesetzt werden, wenn die verschlüsselte Nachricht nicht mehr entschlüsselt werden muß, z.B. UNIX-Passworte.

Soll das Inverse einer Einweg-Funktion berechenbar sein, führt dies zu den sogenannten Einweg-Funktionen mit Falltür (engl. trapdoor oneway-functions). Bei diesen Funktionen ist die Berechnung des Inversen mit Hilfe einer (vor der Öffentlichkeit) geheimzuhaltenden Zusatzinformation effizient berechenbar.

Satz 1 (Fermat-Euler) *Sei Primzahl p und ganze Zahl a gegeben. Dann ist*

$$a^p \equiv a\ (mod\ p) \tag{1}$$

Ist zusätzlich p Nichtteiler von a, dann gilt:

$$a^{p-1} \equiv 1\ (mod\ p) \tag{2}$$

<u>Beweis:</u>

Mit $p \not| a$, also $(p, a) = 1$ folgt (2) aus (1) mit der Kürzungsregel $ac \equiv bc\ (mod\ m) \Leftrightarrow a \equiv b\ (mod\ \frac{m}{(c,m)})$:

$a^p = a^{p-1} \cdot a \equiv 1 \cdot a\ (mod\ \frac{p}{\underbrace{(a,p)}_{=1}}) \Leftrightarrow (2)$

Für (IV) $a = 0$ ist (1) trivial richtig.

(IS): $(a+1)^p = \sum\limits_{j=0}^{p} \binom{p}{j} a^j = a^p + \underbrace{\ldots}_{\text{p enthalten}} + 1 \equiv \underbrace{a^p}_{\equiv a \text{ nach IB}} + 1 \equiv a + 1\ (mod\ p)$

Definition 1 (Euler'sche φ-Funktion)

$$\varphi(m) = |\{a \in \mathbb{Z}_m : (a,m) = 1\}|$$

Für paarweise verschiedene Primzahlen p_i gilt:
$\varphi\left(\prod_{i=1}^{k} p_i\right) = (p_1 - 1)(p_2 - 1) \cdot \ldots \cdot (p_k - 1)$

Satz 2 *Sei n eine quadratfreie natürliche Zahl, d.h. $n = p_1 \cdot \ldots \cdot p_s$ mit paarweise verschiedenen Primzahlen p_i. Dann gilt:*

$$x^{l\varphi(n)+1} \equiv x \ (mod\ n) \quad \text{für alle} \quad x, l \in \mathbb{Z} \quad (3)$$

Beweis:

- $n = p$ mit p prim $\Rightarrow \varphi(n) = p - 1$
  - $x \equiv 0\ (mod\ p)$:
    $x^{l\varphi(n)+1} = (\underbrace{kp}_{\equiv 0\ (mod\ p)})^{l\varphi(n)+1} \equiv 0 mod\ p$
  - $x \not\equiv 0\ (mod\ p)$: Mit Satz 1(2) ist:
    $x^{l\varphi(n)+1} = (\underbrace{x^{\overbrace{\varphi(n)}^{=p-1}}}_{\equiv 1 mod\ p})^l \cdot x \equiv 1^l \cdot x \equiv x\ (mod\ p)$

- $n = \prod^{s} p_i \Rightarrow$
  $\varphi(n) = (p_1 - 1)(p_2 - 1) \ldots (p_s - 1)$
  $x^{l\varphi(n)+1} = x^{l(p_1-1)(p_2-1)\ldots(p_s-1)} \cdot x =$

$$
\begin{aligned}
p_1: \quad & (\underbrace{x^{p_1-1}}_{\equiv 1})^{l(p_2-1)\ldots(p_s-1)} \cdot x \\
& \equiv \;\; x \; (mod\; p_1) \\
& \vdots \\
p_i: \quad & (\underbrace{x^{p_i-1}}_{\equiv 1})^{l(p_1-1)\ldots(p_{i-1}-1)(p_{i+1}-1)\ldots(p_s-1)} \cdot x \\
& \equiv \;\; x \; (mod\; p_i) \\
& \vdots
\end{aligned}
$$

Aus $p_i \mid x^{l\varphi(n)+1} - x, i = 1, \ldots s$ folgt:

$p_1 \cdot \ldots \cdot p_s \mid x^{l\varphi(n)+1} - x$

Also: $x^{l\varphi(n)+1} \equiv x \; (mod \; \underbrace{\prod^{s} p_j}_{=n})$

3. Das RSA-Verfahren

Ein schwer zu lösendes Problem ist die Primfaktorzerlegung großer natürlicher Zahlen (wohingegen das Finden und Multiplizieren zweier großer Primzahlen relativ einfach ist). Der effizienteste (bisher noch nicht veröffentlichte) bekannte Algorithmus zur Faktorisierung einer Zahl n stammt von Richard Schroeppel und benötigt näherungsweise

$$\exp(\sqrt{\ln(n) \cdot \ln(\ln(n))}) \approx \exp(\sqrt{2,3d \cdot \ln(2,3d)})$$

Rivest, Shamir und Adleman nutzten diesen Sachverhalt und formulierten das RSA-Verfahren:

Das RSA-Verfahren:

- Wähle zwei große Primzahlen p,q und berechne $n = p \cdot q$

- Wähle $d \in [0, n-1]$ mit $(\varphi(n), d) = 1$, z.B. d Primzahl und $\max(p, q) < d < \varphi(n) - 1$

- Bestimme $e \in [0, n-1]$ mit $e \cdot d \equiv 1 \ (mod\ \varphi(n))$, d.h. e ist multiplikative Inverse modulo $\varphi(n)$ zu d.

- verschlüssele eine Nachricht $x \in [0, n-1]$ durch $E(x) = x^e \ (mod\ n)$

- entschlüssele einen Kryptotext $y \in [0, n-1]$ durch $D(y) = y^d \ (mod\ n)$

Hierbei bestimmt das Paar

- (e, n) den öffentlichen Schlüssel und
- (d, n) den geheimen Schlüssel.

Geheimzuhalten sind gleichermaßen d, p, q und $\varphi(n)$.

Zu beweisen ist:

$$D(E(x)) \equiv E(D(x)) \equiv x \tag{4}$$

<u>Beweis:</u> Aus $de \equiv 1 \ (mod\ \varphi(n))$ und Satz 2 folgt: $D(E(x)) \equiv x^{de} \equiv x^{k\varphi(n)+1} \equiv x \equiv x^{ed} \equiv E(D(x))$
Die Existenz von e ist gesichert, da mit $(d, \varphi(n)) = 1 \quad a, b \in \mathbb{Z}$ existieren, sodaß $da - b\varphi(n) = 1 \Rightarrow e = a$ und $de = b\varphi(n) + 1 \Leftrightarrow ed \equiv 1 \ (mod\ \varphi(n))$

4. Realisierungen

- Bestimmung von p,q und d durch Primzahltests (z.B. Rabin-Miller)

- Bestimmung von e durch erweiterten Euklidschen Algorithmus:

```
function invmod(phi,d:integer):integer;
var x0:=ph, x1:=d, b0=0, b1=1, y : integer;
begin
  while x1#0 do
    y:=x0 div x1;
    (x0,x1):=(x1,x0-y*x1);
    (b0,b1):=(b1,b0-y*b1);
  od;
  if b0>=0 then invmod:=b0 else invmod:=b0+phi fi
end;
```

- Beschleunigung durch Chin. Restsatz:
  Aus $de' \equiv 1 \ (mod\ p-1)$ und
  $de'' \equiv 1 \ (mod\ q-1)$ bestimme e' und e''
  Bestimme sodann x',x'' aus
  $x' \equiv x^{de'} \ (mod\ p)$ und $x'' \equiv x^{de''} \ (mod\ q)$
  Das gesuchte x ist $z \cdot q \cdot x' + z' \cdot p \cdot x'' \ (mod\ n)$
  mit $zp \equiv 1 \ (mod\ q)$ und $z'q \equiv 1 \ (mod\ p)$

- Schnelles Potenzieren durch
  'repeat squaring & multiplication':

```
function fastexp(b,e,n:integer):integer;
var x:=1 : integer;
begin
  while e#0 do
    while e mod 2 = 0 do
      e:=e div 2;
      b:=(b*b) mod n;
    od;
    e:=e-1;
    x:=(x*b) mod n;
  od;
  fastexp:=x;
end;
```

5. Angriffsmöglichkeiten

- 'brute force'-Methode der erschöpfenden Suche über den Klartextraum.
  Nachteile: Anzahl der zu untersuchenden möglichen Klartexte und Ergebnis kein Schlüssel.

- Faktorisierung von n:
  Nach dem Algorithmus von Schroeppel ist:

Stellen von n	Operationen	Zeit (1μs/Op)	Zeit (1νs/Op)
50	$1,4 \cdot 10^{10}$	$3,9$ Stunden	14 Sekunden
75	$8,8 \cdot 10^{12}$	102 Tage	$2,4$ Stunden
100	$2,3 \cdot 10^{15}$	72 Jahre	26 Tage
200	$1,2 \cdot 10^{23}$	$3,7 \cdot 10^{9}$ Jahre	$3,7 \cdot 10^{6}$ Jahre
300	$1,5 \cdot 10^{29}$	$4,7 \cdot 10^{15}$ Jahre	$4,7 \cdot 10^{12}$ Jahre

  C101=P66*P36 brauchte 66 Stunden auf 1024 Prozessoren.
  Wenn p,q nahe bei $\sqrt{n}$, kann Fermat'sche Faktorisierung angewandt werden. Konsequenz: p,q nicht nahe bei $\sqrt{n}$!

- Problem der Faktorisierung von $\varphi(n)$ ist genauso schwer wie Faktorisierung von n.

- Iteration:
  Es gilt: für alle $x \in \mathbb{Z}_n \quad \exists_1 k$ mit
  $c_{k+1} = c_1 (= c = E(m)) \Rightarrow c_k = m$.

  <u>Beispiel:</u>
  Sei $(e, d, n) = (17, 157, 2773 = 47 \cdot 59), m = 2209$,
  c_1=2209^{17}=1504 (mod 2773)= $c = m^e$
  Dann gilt:
  c_2=1504^{17}=2444 (mod 2773)
  c_3=2444^{17}=0470 (mod 2773)
  c_4=0470^{17}=2209 (mod 2773)= m
  c_5=2209^{17}=1504 (mod 2773)= c

Abhilfe: Mit $n = p_1 \cdot p_2$ sollen

- $p_1 - 1$ und $p_2 - 1$ große Primzahlen enthalten

- $(p_1 - 1, p_2 - 1)$ klein sein

Außerdem für $p_i^{'} = \frac{p_i - 1}{2}, i \in \{1, 2\}$ sollen

- $p_1^{'} - 1$ und $p_2^{'} - 1$ große Primzahlen enthalten

- $(p_1^{'} - 1, p_2^{'} - 1)$ klein sein

6. Anwendungen

Schlüsselverteilungsverfahren:
RSA bietet einen sicheren Kanal zur Übermittlung von (symmetrischen) Private-Keys.

Message Verification / Authentikation:
Person A: $((d, n)), (e', n')$
Person B: $((d', n')), (e, n)$

$$x' \equiv x^d \ (mod\ n) \quad \text{und} \quad y' \equiv (x')^{e'} \ (mod\ n')$$
$$\Downarrow$$
$$(y')^{d'} \equiv (x')^{e'd'} \equiv x' \ (mod\ n') \quad \text{und}$$
$$(x')^e \equiv (x)^{de} \equiv x \ (mod\ n)$$