Zur Sicherheit von RSA

Sebastian Petersen

19. Dezember 2011

RSA

Schlüsselerzeugung

- Der Empfänger (E) wählt große Primzahlen p und q.
- E berechnet $N := pq$ und $\varphi := (p-1)(q-1)$.
- E wählt e teilerfremd zu φ.
- E berechnet d mit $\boxed{ed \equiv 1 \ (\varphi)}$.

Öffentlicher Schlüsselteil von E: (N, e) - kann auf die Homepage gesetzt werden.

Geheimer Schlüssel von E: (p, q, φ, d).

RSA

Öffentlicher Schlüsselteil von E: (N, e)
Geheimer Schlüssel von E: (p, q, φ, d).

Verschlüsselung der Nachricht $m \in \mathbb{Z}/N$ **an** E**:** $c := m^e$ in $\mathbb{Z}/N$.
(Berechnung benutzt nur öffentliche Schlüsselteile von E.)

Entschlüsselung einer Nachricht c **durch** E**:** Via der Formel $m = c^d$ in $\mathbb{Z}/N$. (Benutzt den geheimen Exponenten von E.)

Sicherheit von RSA

Öffentlicher Schlüsselteil von E: (N, e)
Geheimer Schlüssel von E: (p, q, φ, d).

Angreifer O kennt (N, e) und c.

Nimm an, es gelingt O die Zahl N zu faktorisieren:

- Er kennt dann p und q.
- Er kann dann $\varphi = (p-1)(q-1)$ berechnen.
- Er kennt auch e.
- Er berechnet dann d mit $ed \equiv 1 \ (\varphi)$, so wie E selbst das anfangs auch getan hat!

Wenn O den geheimen Exponenten d kennt, dann kann er nach Belieben mitlesen.

Sicherheit von RSA

Die Sicherheit von RSA beruht also auf dem folgenden

Dogma: Wenn $p \neq q$ große (sagen wir: pq hat ≥ 1000 Bit) Primzahlen Zahlen sind, bei deren Erzeugung gewisse Vorsichtsmaßnahmen (siehe später) beachtet werden, dann ist es aus Laufzeitgründen **technisch unmöglich** gegeben $N := pq$ die beiden Faktoren p und q zu berechnen.

- Der schnellste bekannte Algorithmus (das Zahlkörpersieb) ist viel zu langsam.
- Kein Polynomzeit-Algorithmus für FACTORING(N) bekannt. Experten vermuten, daß es keinen gibt.
- RSA-Contest: Details nächste Seite.

Sicherheit von RSA

Zum RSA-Contest:

- 1991 und 2001 hat die Firma RSA Listen von RSA-Moduln veröffentlicht und Preisgelder ausgelobt, z.B. 200.000 USD für die 2024-Bit-Zahl RSA-2024.
  <u>Nur "kleine" Zahlen der Liste bisher faktorisiert.</u>
- RSA-576 wurde 2003 faktorisiert (verteiltes Rechnen an Uni Bonn, MPI Bonn, IEM Essen).
- RSA-640 und RSA-768 sind ebenfalls faktorisiert (2005 und 2009, ebenfalls verteiltes Rechnen und hohe Laufzeit).
- Der Wettbewerb lief 2007 aus.
- RSA-1024 und RSA-2048 sind (soweit ich weiß) nach 10 Jahren von "Angriffen" durch Experten-Teams immer noch nicht faktorisiert!

Dies spricht für die Richtigkeit des Dogmas, wenn N eine 1000- oder 2000-Bit-Zahl ist.

RSA-Vorsichtsmaßnahmen

Es ist dringend zu empfehlen, bei der Wahl von p und q die folgenden **Vorsichtsmaßnahmen** zu beachten:

- p, q sollen groß genug sein, sodaß $N \approx 2^{1000}$ oder sogar $N \approx 2^{2000}$.
- $|p - q|$ soll nicht weder zu klein noch zu groß sein. Ideal ist, wenn p ein paar Bit länger ist als q.
- $p - 1$ und $q - 1$ sollen jeweils ihrerseits einen großen Primfaktor haben, etwa mit 300 Bit.

Die nächste Folie erklärt, warum man die 2. Vorsichtsmaßnahme braucht.

Fermat-Faktorisierung

Nimm an, $N = pq$ mit $|p - q|$ klein.

- Suche ab $floor(\sqrt{N}) + 1$ nach einer Zahl x derart, daß $x^2 - N$ eine Quadratzahl ist.
- Berechne y mit $y^2 = x^2 - N$.
- Nun gilt $N = x^2 - y^2 = (x + y)(x - y)$.
- Nun ist $p = x + y$ und $q = x - y$. - Faktoren von N gefunden.

Wenn $|p - q|$ zu klein, dann findet man "schnell" ein x mit $x^2 - N$ Quadratzahl.

Wenn p ein paar Bit länger ist als q (z.B. 16 mal so groß), dann dauert das Auffinden von x zu lange.

$-- >$ PARI-Experiment

Padding und Salting

Man wird üblicherweise Texte über ASCII versenden wollen, nicht Texte über dem Restklassenring $\mathbb{Z}/N$.
Umrechnungen werden erforderlich.

ASCII-Texte kann man in Bitvektoren umrechnen. ASCII-Tabelle:

ASCII	DEZ	BIN
A	65	1000001
B	66	1000010
C	67	1000011
?	63	1111111
(	40	0101000

Bsp.: HALLO ist 1001000 1000001 1001100 1001100 1001111.

Es genügt also, Bitvektoren verschicken zu können.

Padding und Salting

Will Bitvektoren verschicken können.
RSA verschlüsselt Elemente von $\mathbb{Z}/N$.
Idee: Verwende dazu Abbildungen:

$$pad : \{0,1\}^* \to (\mathbb{Z}/N)^* \quad \text{und} \quad repad : (\mathbb{Z}/N)^* \to \{0,1\}^*$$

mit $repad \circ pad = Id$.

Konstruktionsidee:
$[x_0, \cdots, x_s] := \sum_{i=0}^{s} x_{s-i} 2^i$ für $x \in \{0,1\}^s$.
Sei s maximal mit $2^s \leq N$.
Codiere Bitstrom $x \in \{0,1\}^*$ zu

$$pad(x) = (\overline{[x_0, \cdots, x_s]}, \overline{[x_{s+1}, \cdots, x_{2s}]}, \cdots)$$

wobei $\overline{a} =$ Restklasse von a in $\mathbb{Z}/N$.
repad macht das durch 2-adisches Entwickeln rückgängig.

Padding und Salting

Die Routinen *pad* und *repad* müssen öffentlich gemacht werden. Statt dem Bitvektor x verschickt man nun die RSA-Verschlüsselung von $pad(x)$.

Wörterbuch-Angriff: Ein Angreifer O kann bei diesem Verfahren prüfen, ob ein gegebener Text $T \in \{0,1\}^*$ unter den Nachrichten an eine Person E vorkommt!

Dazu muß O nur den *öffentlichen* Schlüssel (N, e) von E nachschlagen, $pad(T)$ damit verschlüsseln und prüfen, ob der entstehende "Zahlensalat" unter den Nachrichten an E vorkommt.

Ausweg: Salting!

Padding und Salting

Salting. Man verwendet als Paddingroutine eine *randomisierte* Abbildung von folgendem Typ:
Sei t ein Parameter (etwa $t = 100$) und s maximal mit $2^{t+s} \leq N$.

$$pad^r(x) = (\overline{[b_1, \cdots, b_t, x_0, \cdots, x_s]}, \overline{[b_{t+1}, \cdots, b_{2t}, x_{s+1}, \cdots, x_{2s}]}, \cdots)$$

wobei die b_i Zufallsbits sind. Die b_i werden auch Salts genannt. Dies kann durch eine geeignete Routine *repad*^{r} rückgängig gemacht werden.

Beachte: Ein gegebener Bitvektor der Länge rs kann nun auf 2^{rt} verschiedene Arten verschlüsselt werden. Wörterbuchangriff wird dadurch aus Laufzeitgründen unmöglich!

Padding und Salting

Der Name *pad*^{r} steht dabei für randomisiertes Padding.

Fazit: (Adleman) The RSA-kernel without suitable randomized padding routines does not satisfy basic security notions.

Man sollte also immer den RSA-Kern mit randomisiertem Padding verbinden. Der Standard-Algorithmus für das randomisierte Padding heißt PKCS. Wir können nicht auf alle technischen Details von PKCS eingehen.