

Angriffe auf das RSA Verfahren

Dennis Knorr

26.08.2008

Inhaltsverzeichnis

Was ist RSA?

- notwendige Mathematik
- Das RSA-Verfahren

Angriffsarten auf RSA

- Bruteforce und Optimierungen
- Seitenkanalattacken
- Faktorisierungsverfahren
- Unsichere RSA-Parameter

Die Geschichte von RSA

- ▶ inoffiziell scheinen Mitarbeiter des britischen Geheimdienstes schon in den ersten 1970 Jahren eine hnliche Version von RSA entwickelt zu haben
- ▶ Offiziell wurde das Verfahren von Ron **R**ivest, Adi **S**hamir und Leonard **A**dleman entwickelt und auch patentiert.
- ▶ Das Patent lief bis September 2000, jetzt darf man es legal einsetzen.
- ▶ RSA ist vermutlich das am meisten eingesetzte asymmetrische Kryptoverfahren.

verwendete Mathematik

- ▶ Modulorechnung
- ▶ (erweiterter) Euklidischer Algorithmus
- ▶ Chinesischer Restsatz
- ▶ Eulerscher Satz

Modulorechnung

Folgende Aussagen sind äquivalent:

- ▶ $a \equiv b \text{ mod } m$
- ▶ $a = b + k * m, k \in \mathbb{Z}$
- ▶ a und b lassen bei der Division durch m den selben Rest
- ▶ rechnen modulo m ist wie rechnen in $\mathbb{Z}_m = \{0, 1, \dots, m\}$
- ▶ $17 + 7 \equiv 6 + 7 \text{ mod } 11 \equiv 13 \text{ mod } 11 \equiv 2$

(erweiterter) Euklidischer Algorithmus

- ▶ $n \in \mathbb{N}$ ist der grösste gemeinsame Teiler zwei ganzer Zahlen a und b .
- ▶ $n = \text{ggT}(a, b)$
- ▶ Dann gibt es zwei Zahlen x und $y \in \mathbb{Z}$, für die folgende Gleichung gilt:
- ▶ $n = x * a + y * b$
- ▶ Anwendung ist die Berechnung des Inversen mod n
- ▶ $a^{-1} \equiv x \pmod{n}$ mit $1 = \text{ggT}(a, n) = a * x + y * n$

Der chinesische Restsatz

- ▶ Der Chinesische Restsatz macht eine Aussage über die Äquivalenz von Zahlenstrukturen
- ▶ $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$ mit $n = p * q$
- ▶ $x \cong (y, z)$ mit $x \in \mathbb{Z}_n, y \in \mathbb{Z}_p, z \in \mathbb{Z}_q$
- ▶ Dies funktioniert in beide Richtungen und wird eingesetzt um Berechnungen für RSA schneller ablaufen zu lassen.

Der eulersche Satz

- ▶ Der eulersche Satz sagt etwas über Rechenregeln beim modularen Potenzieren aus. Interesting Background omitted..
- ▶ $n = p * q, \Phi(n) = (p - 1) * (q - 1)$
- ▶ $a \in \mathbb{Z}, n \in \mathbb{N} : a^{\Phi(n)} \equiv 1 \pmod n$ mit $\text{ggT}(a, n) = 1$

Vorraussetzungen

- ▶ Man wählt 2 Primzahlen p und q , die ungefaehr gleich gross sind, da sonst die Faktorisierung schneller von statten gehen kann und noch speziellere Attacken möglich sind.
- ▶ $\log_2(p) \sim \log_2(q)$
- ▶ Man waehlt den privaten Schlüssel d und den öffentlichen Schlüssel e : $d * e \equiv 1 \bmod \Phi(n)$

Ver- und Entschlüsselung

- ▶ Verschlüsselung $c = m^e \bmod n$
- ▶ Entschlüsselung $m = c^d \bmod n$
- ▶ Dies geht auch mit dem CRT, was die Grösse der Variablen halbiert, natürlich hat man dann 2 davon.

Kurze Notationsnotiz

- ▶ e ist der öffentliche Schlüssel
- ▶ d ist der geheime Schlüssel
- ▶ n ist der Modulus, der faktorisiert $p \cdot q$ ist
- ▶ c ist eine Chiffretextnachricht
- ▶ m ist eine Klartextnachricht

Bruteforceangriff auf m

1. Wähle Modulus sowie den öffentlichen Schlüssel und die Chiffrenachricht
2. for i gleich 0 to n
3. berechne $c = i^e \bmod n$
 - 3.1 if($c = \text{chiffre} \bmod n$) then return m

Bruteforceangriff auf den privaten Schluessel

1. Waehle den Modulus sowie $m \in_R \mathbb{Z}_n$
2. berechne $c = m^e \bmod n$
3. for i gleich 0 to n
 - 3.1 if($c^i = m \bmod n$) then return i

Meet in the Middle Attacken

Es gibt diverse Optimierungen fuer Bruteforce, auch ist Meet-in-the-Middle oft anwendbar.

Es werden dabei Gleichungen erstellt, die aehnlich wie bei BabyStepGiantStep beschaffen sind.

MitM auf den privaten Schlüssel

Mittels der Formel $c^d = c^{A*d_0+d_1} = (c^A)^{d_0} * c^{d_1} = m \bmod n$ finden wir in einer Tabelle Schlüssel derart, sodass wir eine Lookuptabelle durchsuchen, die wir 2 mal benutzen können.

Es wird dabei auf die Äquivalenz $(c^A)^{d_0} = (m^{-1}) * c^{d_1}$ abgezielt.

Eingabe: n , e , obere Grenze B

1. Wähle m zufällig und setze $c = m^e \bmod n$
2. setze $A = \sqrt{\lceil B \rceil}$
3. for $d_0 = 0$ to A
 - 3.1 speichere $(d_0, (c^A)^{d_0} \bmod n)$ nach der zweiten Komponente sortiert in einer Tabelle
4. for $d_1 = 0$ to A
 - 4.1 suche mit binärer Suche, ob in der Tabelle ein Wert $(d_1, m * (c^{-1})^{d_1} \bmod n)$ vorkommt
 - 4.2 falls ja, ist die Suche beendet, weil man hat Werte für d_0 und d_1 gefunden.
5. Die Ausgabe ist $d = d_0 * A + d_1$

MitM-Angriffe

- ▶ Dieser Angriff hat einen Time-Memory-TradeOff mit je Platz- wie auch Zeitkomplexitaet von $O(\sqrt{d})$
- ▶ Dies laesst sich auch auf Versionen, die CRT-RSA verwenden ausweiten oder auch auf Angriffe für m.

MitM-Angriff auf den Klartext m

- ▶ Falls $m < N^{1/e}$ gilt, lässt sich mit einer MitM-Attacke leicht der Klartext berechnen
- ▶ Der Algorithmus läuft analog wie beim Angriff auf d
- ▶ Die verwendete Gleichung, auf der die Lookuptabelle aufbaut, heisst $m_0^e = c * m_1^{-e} \bmod n$
- ▶ Dies funktioniert allerdings nur mit einer bestimmten Wahrscheinlichkeit, da sich m als Produkt zweier Zahlen m_0 und m_1 darstellen lassen muss

Seitenkanalattacken

Seitenkanalattacken sind Angriffe in einer real existierenden und angewandten Implementierung verwenden um Aufschluss über das Geheimnis zu bekommen.

Hierbei wird Dauer einer Chiffrierung oder Stromverbrauch verwendet um Rueckschluesse zu ziehen.

Kochers TimingAngriff

Bei Implementationen auf Chipkarten wird zum Beispiel der Zeitverbrauch gemessen, da mittels Repeated Squaring (alias SQM) Nachrichten verschlüsselt oder signiert werden. Die Eingabe ist n , m und d (oder analog e). Der Exponent ist hierbei als Bitvektor zu lesen.

1. $z = 1$
2. for $i = 0$ to $m - 1$
 - 2.1 if $d_i = 1$ then $z = z * x \bmod n$
 - 2.2 if $(i \mid m - 1)$ then $x = x^2 \bmod n$

der Bellcore Angriff

Auf Chipkarten wird oft nicht in \mathbb{Z}_n gerechnet sondern in \mathbb{Z}_p und \mathbb{Z}_q , da dies weniger rechenaufwendig ist.

Statt $y = x^d \bmod n$ wird $y_p = x^{d_p} \bmod p$ und $y_q = x^{d_q} \bmod q$ gerechnet. Mit dem CRT wird dies dann zu y_n "vermixt".

Wenn durch Fault Injection y_p oder y_q veraendert wird, kann man ueber die Gleichung $(y^e - x, N)$ erst p und dann q ermitteln.

Fehlerangriff auf den oeffentlichen Schluessel e

Angenommen, jemand will mit einem Schlüssel (N, e) eine Nachricht fuer A verschluesseln. Dem Angreifer gelingt es aber, dem Sender den Schluessel (N, e_f) unterzuschieben, wobei $\text{ggT}(e, e_f) = 1$ ist. Dann wird der Empfaenger dem Sender sagen, er konnte die Nachricht natuerlich nicht dechiffrieren. Der Sender schickt dann die Nachricht evtl. nochmal mit dem korrekten Schluessel an den Empfaenger.

Dann kann der Angreifer ohne Kenntniss des privaten Schluessels des Empfaengers die Nachricht m mit dem erweiterten Euklid berechnen.

$$(m^e)^v * (m^{e_f})^u = m^1 \text{ mod } n$$

Übersicht ueber ein paar Verfahren(-sklassen)

Die wichtigste Angriffsvariante auf RSA sind wohl die Angriffe auf das Geheimnis $n = p * q$. Daher gibt es natuerlich eine Klasse an Angriffen, die sich darauf spezialisiert.

- ▶ Die k-Angriffe
- ▶ Sieb des Eratosthenes
- ▶ Quadratische Reste
- ▶ Faktorisieren bei bekannten Schluesselbits

k-Angriffe

Die Schluessel e und d berechnen sich ja aus $e * d = 1 \bmod \Phi(n)$, was auch folgender Gleichung entspricht:

$$\begin{aligned} e * d &= 1 + k * (p - 1)(q - 1) \text{ oder} \\ e * d - 1 &= k * (n - (p + q) + 1) \end{aligned}$$

Jetzt gibt es Ansätze k moeglichst geschickt zu waehlen, so dass man das Geheimnis p oder q erraten kann.

Unter anderem existiert der Ansatz, $p + q$ zu schaetzen, da man dadurch eine quadratische Gleichung ueber den ganzen Zahlen (effizient) loesen kann. Einen Algorithmus dafuer werde ich nur auf Wunsch vorfuehren, da er Kenntnis des Produktes von e UND d voraussetzt...

Sieb des Erathostenes

Das Sieb ist eine verbesserte Form der Probedivision mit der Eingabe von n und der Ausgabe eines Faktors oder der Aussage, dass n prim ist.

1. Erstelle ein Feld $A[2 \dots \sqrt{n}] = 0$
2. fuer $t = 2$ bis \sqrt{n}
 - ▶ if ($A[t] = 0$) dann:
 - ▶ falls dieses t das n teilt, gebe t aus und stoppe
 - ▶ sonst: fuer $s = t$ bis \sqrt{n}/t setze $E[s] = 1$

Quadratische Reste

Fermat hatte folgende Idee:

- ▶ Finde $x, y \in \mathbb{Z}_n$ so dass
- ▶ $x^2 = y^2 \pmod n$ und $\text{ggT}(x * y, n) = 1$ ist
- ▶ Daraus folgt $x^2 - y^2 = (x - y)(x + y) = 0 \pmod n$
- ▶ Dann liefert $\text{ggT}(x \pm y, n)$ mit grosser Wahrscheinlichkeit einen nichttrivialen Faktor von n

Quadratische Reste II

- ▶ Auf diesen Verfahren bauen die aktuellen Faktorisierungsverfahren fuer RSA auf.
- ▶ Quadratisches Sieb und GNFS
- ▶ Dabei werden geeignete Versuchszahlenklassen, sogenannte Faktorbasen gewaehlt, die das ganze sehr viel schneller machen

Faktorisieren bei bekannten Schluesselbits

- ▶ Coppersmith Methode
- ▶ Polynome ueber Gitter
- ▶ Wenn mehr als $\log_2(N^{1/4})$ Bits bekannt sind von p so kann, n in $O(\log(n))$ berechnet werden
- ▶ k kann dann wieder gut geschaetzt werden :)

Ünsichere RSA-Parameter

- ▶ kleiner Verschlüsselungsexponent
- ▶ kleiner Entschlüsselungsexponent
- ▶ Paddings und stereotype Nachrichten

Kleiner Verschlüsselungsexponent

- ▶ Wenn mehrere Nachrichten mit dem gleichen Verschlüsselungsexponent verschlüsselt wurden..
- ▶ und der Klartext kleiner ist als alle Moduli
- ▶ so kann der Klartext effizient ueber den reellen Zahlen als n -te Wurzel geloest werden
- ▶ Dies funktioniert sogar auch mit 2 verschiedenen Verschlüsselungsexponenten, dann wirds allerdings aufwendiger

Kleiner Verschlüsselungsexponent II

- ▶ m wurde mit $(n_1, 3)$, $(n_2, 3)$ und $(n_3, 3)$ verschlüsselt
- ▶ m ist kleiner als alle n .
- ▶ man führe den Chinesischen Restsatz durch, so dass man $c \bmod (n_1 * n_2 * n_3)$ erhält
- ▶ Man löse die reelle Wurzel $c^{1/3}$ und erhalte m .

Kleiner Entschluesselungsexponent - Wiener Angriff

- ▶ schon 1990 hat Wiener einen Angriff vorgeschlagen, wenn folgende Formel gilt:
- ▶ $d \leq (1/3) * N^{1/4}$
- ▶ dann ist die Gleichung $e * d + k * (p - 1)(q - 1) - 1 = k * N$ in einem Gitter in
- ▶ $O(\log^2(n))$ loesbar

Paddings und stereotype Nachrichten

- ▶ Das univariate RSA-Polynom $0 = x^e - c \bmod n$
- ▶ kann mit der Coppersmith-methode geloest werden, wenn ein Teil der Nachricht bekannt ist
- ▶ $c = (x + s)^e \bmod n$
- ▶ Dies funktioniert wenn die unbekannte Nachricht $x \leq N^{1/e}$ ist.

- ▶ Was man draus lernt (Nicht-determinismus, OAEP)
- ▶ Fragen, Wiederholungen
- ▶ ENDE.