<u>Das RSA-Verfahren</u>

Schlüsselerzeugung: Wähle zufällig Primzahlen p und q.

Größenordnung von p und q: $\quad \mathbf{2^{512}}$

Berechne RSA-Modul $n = pq$.

Wähle Verschlüsselungsexponent e mit $1 < e < \varphi(n)$.

Erinnerung: $\varphi(n)$ ist Eulersche φ-Funktion.

- $\varphi(n)$ ist die Ordnung der Gruppe $(\mathbb{Z}/n\mathbb{Z})^*$
- $\varphi(n)$ Anzahl der Zahlen a mit $ggT(a,n) = 1$
- für Primzahlen p ist $\varphi(p) = p - 1$
- für m, n mit $ggT(m,n) = 1$ gilt: $\varphi(mn) = \varphi(m)\varphi(n)$
- allgemein gilt für $n = \prod_{p|n} p^{e_p}$ (p Primzahl):

$$\varphi(n) = \prod_{p|n} p^{e_p - 1}(p-1) = n \prod_{p|n}(1 - \frac{1}{p})$$

- für RSA-Modul n ist $\varphi(n) = (p-1)(q-1)$

Berechne Entschlüsselungsexponent d mit $1 < d < \varphi(n)$ und

$$de \equiv 1 \mod \varphi(n)$$

Ver- und Entschlüsselung

Klar- und Chiffretextraum	$m \in \mathbb{N}$ mit $0 \leq m < n$
Schlüsselraum	$k_{\text{public}} = (n, e)$, $k_{\text{private}} = d$
Verschlüsselungsfunktion	$E_{k_{\text{public}}}(m) = m^e \mod n = c$
Entschlüsselungsfunktion	$D_{k_{\text{private}}}(c) = c^d \mod n = m$

Beispiel:

- wähle $p = 5$ und $q = 13$, berechne $n = 65$
- $\varphi(65) = 4 * 12 = 48$; wähle $e = 5$, dann ist $d = 29$
- verschlüssele $m = 33$, erhalte $c = 33^5 \mod 65 = 63$
- entschlüssele $c = 63$, erhalte $m = 63^{29} \mod 65 = 33$

Satz: Es gilt $(m^e)^d \mod n = m$ für alle m mit $0 \leq m < n$.

Beweis:

- es gilt $ed \equiv 1 \mod \varphi(n)$
- $\exists l$ mit $ed = 1 + l(p-1)(q-1)$
- kleiner Satz von Fermat liefert
$$(m^e)^d = m(m^{(p-1)})^{(q-1)l} \equiv m \mod p$$
$$(m^e)^d = m(m^{(q-1)})^{(p-1)l} \equiv m \mod q$$
- $(m^e)^d \equiv m \mod n$

$\square$

<u>Effizienz von RSA</u>

Bei Verschlüsselung: $m^e \mod n$
Bei Entschlüsselung: $c^d \mod n$

Effizienz abhängig von Größenordnung von e und d.

In der Regel:

- e relativ klein
- d in der Größenordnung nahe bei n (2^{1024})

Verwende schnelle Exponentiation ($k = 1024$):

- Binärentwicklung des Exponenten $d = \sum\limits_{i=0}^{k} d_i 2^i, d_i \in \{0, 1\}$
- $c^d = c^{\sum\limits_{i=0}^{k} d_i 2^i} = \prod_{i=0}^{k} (c^{2^i})^{d_i} = \prod\limits_{0 \leq i \leq k, d_i = 1} c^{2^i}$
- berechne Quadrate c^{2^i} für $0 \leq i \leq k$
- es gilt $c^{2^{i+1}} = (c^{2^i})^2$
- berechne c^d als Produkt der Quadrate c^{2^i} mit $d_i = 1$

Beispiel: $c^d \mod 100$ mit $c = 8, d = 33 = 2^0 + 2^5$
$8^2 = 64$, $8^{2^2} = 4096 \equiv -4 \mod 100$, $8^{2^3} \equiv 16 \mod 100$
$8^{2^4} \equiv 56 \mod 100$, $8^{2^5} \equiv 36 \mod 100$
$\Rightarrow 8^{33} \equiv 8^{2^0} * 8^{2^5} \equiv 8 * 36 \equiv 88 \mod 100$

Laufzeit Entschlüsselung

Berechne $c^d \mod n$: sei k Bitlänge von RSA-Modul n:

$\Rightarrow k$ Quadrierungen, k Multiplikationen

Hier: RSA-Modul hat 1024 Bit, d.h. ≈ 2000 Operationen sind nötig, um nur einen Block zu entschlüsseln!

Kann durch Faktor 4 mit Hilfe des chinesischen Restsatzes beschleunigt werden!

Chinesischer Restsatz:

- $m_1, \dots, m_n \in \mathbb{N}$ paarweise teilerfremd
- $a_1, \dots, a_n \in \mathbb{Z}$
- gegeben: simultane Kongruenz

$$x \equiv a_1 \mod m_1, \dots, m \equiv a_n \mod m_n$$

- hat Lösung x, die eindeutig ist $\mod m = \prod_{i=1}^{n} m_i$
- setze dazu $M_i = \frac{m}{m_i}, 1 \leq i \leq n$
- löse $y_i M_i \equiv 1 \mod m_i, 1 \leq i \leq n$
- dann ist $x = (\sum_{i=1}^{n} a_i y_i M_i) \mod m$

Optimiere Entschlüsselung

Anwendung:

- berechne

$$m_p = c^d \mod p, \quad m_q = c^d \mod q$$

- löse simultane Kongruenz

$$m \equiv m_p \mod p, \quad m \equiv m_q \mod q$$

  durch Zahlen y_p und y_q mit

$$y_p p + y_q q = 1$$

- $m = (m_p y_q q + m_q y_p p) \mod n$
- y_p und y_q müssen nur einmal ausgerechnet werden und können dann gespeichert werden, da von m unabhängig

Beispiel: $p = 5, q = 13$, berechne 63^{29} :

- $m_p = 63^{29} \mod 5 = 3, \quad m_q = 63^{29} \mod 13 = 7$
- $y_p = -5, \quad y_q = 2$
- $m = (3 * 26 - 7 * (-25)) \mod 65 = -97 \mod 65 = 33$

Randomisierung

Problem: Bei public-key-Verfahren können beliebig viele `Klartext-Chiffretext` Paare erzeugt werden, da die Verschlüsselungsfunktion öffentlich bekannt ist.

Lösung: private-key-Verfahren: CBC–Modus; RSA:

- b binäre Länge RSA-Modul ($= 1024$)
- k mit 2^k ist obere Schranke für Angreiferalgorithmus
- definiere $l = b - k - 1$ und Funktionen

$$G : \{0,1\}^k \longrightarrow \{0,1\}^l$$
$$H : \{0,1\}^l \longrightarrow \{0,1\}^k$$

- verschlüssele $m \in \{0,1\}^l$ mit Zufallswert $r \in \{0,1\}^k$:

$$c = ((m \oplus G(r)) \circ (r \oplus H(m \oplus G(r)))^e \mod n$$

- Entschlüsselung:

$$c^d \mod n = (m \oplus G(r)) \circ (r \oplus H(m \oplus G(r)))$$

- erhalte r durch Rechnung

$$r = H(m \oplus G(r)) \oplus (r \oplus H(m \oplus G(r)))$$

- und berechne zuletzt

$$m = (m \oplus G(r)) \oplus G(r)$$

Sicherheit von RSA

Es gilt die Äquivalenz:

$$\text{Berechnung von } d \text{ mittels } (n, e) \\ \Longleftrightarrow \\ \text{Faktorisierung von } n$$

Zum heutigen Zeitpunkt sind keine effizienten Faktorisierungsmethoden bekannt, d.h. die ein Angriff auf RSA durch Berechnung der Primfaktoren ist nicht praktikabel (solange p und q groß genug gewählt sind).

Aber: Es ist nicht bekannt, ob man auch ohne Faktorisierung von n einen abgefangenen Chiffretext c entschlüsseln kann!

Weiterer Unsicherheitsfaktor: e zu klein gewählt

$\longrightarrow$ Low-Exponent-Attacke

kleinstmöglichstes e mit $ggT(e, \varphi(n)) = 1$ ist $e = 3$

Satz:

- $e \in \mathbb{N}, n_1, \ldots, n_e \in \mathbb{N}$ paarweise teilerfremd
- $m \in N$ mit $0 \leq m \leq n_i$ für $1 \leq i \leq e$
- $c \in \mathbb{N}$ mit $c \equiv m^e \mod n_i$ für alle i und $0 \leq c < \prod_{i=1}^{e} n_i$
- $\Longrightarrow c = m^e$!

Chosen Ciphertext Attacke

Wurde von Daniel Bleichenbacher entwickelt [**?**]

Ist vom Typ *Adaptive Chosen-Ciphertext Attacke*:

- basiert auf dem Standard PKCS #1
- Format: | 00 | 02 | padding string | 00 | data block |
- Zugang zu Orakel, dass PKCS-Konformität überprüft
- gesucht ist $m \equiv c^d \mod n$ für gegebenen Chiffretext c
- wähle Zahl s und berechne

$$c' \equiv cs^e \mod n$$

- sende c' an das Orakel, falls PKCS-konform sind die ersten Bits von ms genau 00 bzw. 02
- dabei wird die Multiplikativität von RSA ausgenutzt

$$c_1 = m^e \mod n, \quad c_2 = s^e \mod n$$
$$\Longrightarrow$$
$$c' = c_1 c_2 \mod n = (ms)^e \mod n$$

- durch geeignete Auswahl von Zahlen s_i erhalte mehr und mehr Informationen über Klartext m
- $2^{20} \approx 1000000$ chosen ciphertexts c' reichen im Durchschnitt, um Klartext m herzuleiten

Das Rabin-Verfahren

Für das Rabin-Verfahren gilt die Äquivalenz:

$$\text{Berechnung von } m = c^2 \mod n$$
$$\Longleftrightarrow$$
$$\text{Faktorisierung von } n$$

Vorgehensweise:

- wähle Primzahlen p, q mit $p \equiv q \equiv 3 \mod 4$
- berechne $n = pq$. public-key: n; private-key: (p, q)
- verschlüsseln von $m \in \{0, \ldots, n-1\}$: $c = m^2 \mod n$
- Entschlüsselung durch Wurzelziehen:

  $$m_p = c^{(p+1)/4} \mod p, \quad m_q = c^{(q+1)/4} \mod q$$

- wie bei RSA: chinesischen Restsatz benutzen
- bestimme Koeffizienten y_p, y_q mit $y_p p + y_q q = 1$
- berechne

  $r = (y_p p m_q + y_q q m_p)$ mod n, $s = (y_p p m_q - y_q q m_p)$ mod n

- $\pm r$ und $\pm s$ sind die vier Quadratwurzeln von c in der Menge $\{0, \ldots, n-1\}$, eine dieser Zahlen entspricht Klartext m
- Problem: Eindeutigkeit von Klartext m nicht gegeben

Literatur

[1] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. *Lecture Notes in Computer Science*, 1462:1–??, 1998.