

# Die Mathematik von RSA

Markus Arndt

(Anmerkung: Text ist als Zahl kodierbar, z.B. ASCII, ISO-8859-15, UTF-8, ...)

Seien  $p, q$  prim,  $n := p * q$  und  $m \in \mathbb{Z}/n\mathbb{Z}$  die zu chiffrierende Zahl,  $e \in (\mathbb{Z}/\varphi(n)\mathbb{Z})^*$ ,  $d = e^{-1}$  (existiert, da  $e$  teilerfremd  $\varphi(n)$ ).

(Anmerkung:  $p$  und  $q$  groß wählen, wegen Sicherheit u. Klartextrraum muß abgedeckt sein)

Verschlüsselungsfunktion:  $E(x) : (\mathbb{Z}/n\mathbb{Z}) \rightarrow (\mathbb{Z}/n\mathbb{Z}), x \mapsto x^e$

Entschlüsselungsfunktion:  $D(x) : (\mathbb{Z}/n\mathbb{Z}) \rightarrow (\mathbb{Z}/n\mathbb{Z}), x \mapsto x^d$

Nun ist  $(e,n)$  der öffentliche und  $(d,n)$  der private Schlüssel.

(Erinnerung:  $\varphi(n) = \varphi(pq) = (p-1)(q-1)$  für  $p, q$  Primzahlen.)

## Bestimmung von $e$ (2 Beispiele)

$e$  soll Einheit in  $\mathbb{Z}/\varphi(n)\mathbb{Z}$  sein.

1) Wähle zufälliges (gerades)  $e$ , überprüfe durch eukl. Alg. ob  $\text{ggT}(e, \varphi(n)) = 1$ . Ansonsten wähle  $e$  neu.

2) Wähle Primzahl  $e$  mit  $\varphi(n)/2 < e < \varphi(n)$ , denn diese kann dann unmöglich ein Primfaktor von  $\varphi(n)$  sein  $\Rightarrow \text{ggT}(e, \varphi(n)) = 1$ .

## Berechnung von $d$ ?

$e \in (\mathbb{Z}/\varphi(n)\mathbb{Z})^*$

$\Rightarrow \exists d : e * d = \varphi(n) * k + 1$  für ein  $k \in \mathbb{N}$ .

$\Leftrightarrow e * d - \varphi(n) * k = 1$  (und  $e$  und  $\varphi(n)$  nach Voraussetzung teilerfremd)

$\Leftrightarrow e * d - \varphi(n) * k = \text{ggT}(e, \varphi(n))$  (= Lin.komb. zweier Zahlen zu ihrem ggT)

Hier wird nun der erweiterte euklidische Alg. angewendet, um die Koeffizienten ( $d$  und  $k$ ) dieser Linearkombination zu ermitteln.  $\Rightarrow$  Man erhält  $d$ .

## Korrektheit von RSA

Es ist zu zeigen, dass in jedem Fall gilt  $(m^e)^d \equiv m$ , für  $p, q, m$  und  $n$  definiert wie bisher.  
Zur Erinnerung:  $e * d \equiv 1 \pmod{\varphi(n)} \Leftrightarrow e * d = 1 + k * \varphi(n)$  für ein  $k \in \mathbb{N}$

1. Fall ( $m$  teilerfremd zu  $n$ )

(Anmerkung: kleiner Euler)

Es gilt  $(m^e)^d = m^{1+k*\varphi(n)} = (m * (m^{\varphi(n)}))^k \equiv m * 1^k = m \pmod{n}$

2. Fall ( $\text{ggT}(m, n) > 1$ )

2.1 Angenommen  $p \mid m \wedge q \mid m \Rightarrow m = n \Leftrightarrow m \equiv 0 \pmod{n} \Rightarrow (m^e)^d \equiv m$

2.2 Angenommen es gilt  $p \mid m \Rightarrow p \mid (m^e)^d - m \wedge q \nmid m$ . (mit kl. Fermat)  
 $\Rightarrow (m^e)^d = m^{1+k*\varphi(n)} = m * m^{k*(p-1)(q-1)} = m * m^{(q-1)^{p-1}} \equiv m \pmod{q}$   
 $\Rightarrow q \mid (m^e)^d - m$

2.3 Angenommen  $q \mid m \Rightarrow q \mid (m^e)^d - m \wedge p \nmid m$  (funktioniert analog)

$\Rightarrow p * q \mid (m^e)^d - m \Leftrightarrow (m^e)^d \equiv m \pmod{n}$ .

## Sicherheit von RSA

Öffentlichkeit hat  $(e, n)$  (public key), kann Nachrichten sicher an Besitzer von  $(d, n)$  übertragen.

$\Rightarrow$  Es darf nicht möglich sein  $d$  in annehmbarer Zeit aus  $(e, n)$  zu errechnen.

Um  $d$  aus  $e$  und  $n$  zu errechnen muss folgende Gleichung gelöst werden (mit EEA):

$$e * d - k * \varphi(n) = 1$$

Da aber gilt  $\varphi(n) = (p - 1) * (q - 1)$  müsste zunächst  $n$  faktorisiert werden um  $q$  und  $p$  zu finden.

### Weitere Möglichkeiten

Da gilt " $(m^e)^d = m^{e*d} = (m^d)^e$ " kann natürlich auch der Besitzer des privaten Schlüssels Nachrichten signieren, so dass Herkunft und Integrität gesichert sind.

### Alternative Methode zur Berechnung von $d$

Aus Eulers Version des "Satz von Fermat" geht das  $d$  auch direkt hervor.

$$\begin{aligned} e^{\varphi(\varphi(n))} &\equiv 1 \pmod{\varphi(n)} \Leftrightarrow e * e^{\varphi(\varphi(n))-1} \equiv 1 \pmod{\varphi(n)} \\ \Rightarrow d &= e^{\varphi(\varphi(n))-1} \pmod{\varphi(n)} \end{aligned}$$

### Schlüsselerzeugung ohne Euklidischen Algorithmus

Wir wissen dass gelten soll  $e * d \equiv 1 \equiv \varphi(n) + 1 \pmod{\varphi(n)} \Rightarrow$  Faktorisiere  $\varphi(n) + 1$  und erhalte  $m$  Primfaktoren  $x_i \Rightarrow 1 = \prod_{i=1}^m x_i = \prod_{i=1}^o x_i * \prod_{j=o+1}^m x_j = e * d$ .

## Schnelles Potenzieren

Um eine effiziente Implementierung von RSA zu ermöglichen verwendet man die Methodik des "schnellen Potenzierens". Dies ist ein Verfahren um mit sehr wenigen Rechenschritten potenzieren zu können.

Dabei wird der Exponent in seine Binärentwicklung zerlegt und die Potenz dann anhand der Summanden im Exponenten zu Faktoren auseinandergezogen. Es gilt natürlich, dass auf einem Rechner für gewöhnlich Zahlen bereits als Tupel der Koeffizienten der Binärentwicklung gespeichert werden. Ein weiterer Trick ist nötig, doch dieser wird erst gleich im praktischen Beispiel erläutert. Anschaulich:

Seien  $a, c \in \mathbb{R}$ , die  $b_i$  Koeffizienten aus  $\{0, 1\}$ .

$$a^c \wedge c = \sum_{i=0}^n b_i * 2^i$$
$$a^c = \prod_{i=0}^n a^{b_i * 2^i}$$

Natürlich muss man nun nur noch die Faktoren berücksichtigen die ein  $b_i \neq 0$  im Exponenten haben. An einem Beispiel soll nun gezeigt werden wieso dies schneller funktioniert als gewöhnlich.

Wollen wir naiv  $8^{35}$  rechnen (z.B. in  $\mathbb{Z}/43\mathbb{Z}$ ), so müssten wir 34 Multiplikationen durchführen. Doch arbeiten wir mit schnellem Potenzieren, dann müssen wir zuerst die Binärentwicklung von 35 niederschreiben (bedenke: auf einem Rechner ist die Zahl bereits in Binärdarstellung repräsentiert).

$$\begin{aligned} \Rightarrow 8^{35} &= 8^{(1*2^0+1*2^1+0*2^2+0*2^3+0*2^4+1*2^5+0*2^6+0*2^7+0*2^8)} \\ &= 8^{(2^0+2^1+2^5)} \text{ (Summanden mit Koeffizienten 0 werden ignoriert; nicht mitkalkuliert)} \\ &= 8^{(2^0)} * 8^{(2^1)} * 8^{(2^5)} \end{aligned}$$

Es gilt nun:  $c^{(2^{i+1})} = c^{(2^i * 2)} = c^{(2^i) * 2} = (c^{2^i})^2$  (besagter Trick).

$$\Rightarrow 8^{2^5} = (8^{2^4})^2 = ((8^{2^3})^2)^2 = (((8^{2^2})^2)^2)^2 = (((8^2)^2)^2)^2$$

$\Rightarrow$  Wir müssen lediglich die 8 fünf mal quadrieren um alle  $8^{2^1}, 8^{2^2}, \dots, 8^{2^5}$  zu errechnen (denn die Zwischenergebnisse merken wir uns).

Also rechnet man nun (in mod 43):

$$8^1 = 8, 8^2 = 64 = 21, 21^2 = 441 = 11, 11^2 = 121 = 35, 35^2 = 1125 = 21, 21^2 = 441 = 11 \pmod{43}$$

Mit 5 Quadrierungen haben wir nun alle benötigten Faktoren bestimmt, es folgen die 3 verbleibenden Multiplikationen.

$$8^{2^0} * 8^{2^1} * 8^{2^5} = 8 * 21 * 11 = 1848 = 42 (= 8^{35}) \pmod{43}$$

Anstelle der 34 Multiplikationen haben wir lediglich 7 Multiplikationen gebraucht (5 mal potenzieren, 2 Produkte).