RSA:

public key n, e

private key p, q, d

$n = p \cdot q$, $\varphi(n) = (p-1) \cdot (q-1)$

$d = \frac{1}{e} \bmod \varphi(n)$

$m^e \bmod n$

$m^d \bmod n$

Die Euler'sche φ-Funktion

$$\varphi(n) = \left| \{ r \mid 1 \le r \le n-1, ggT(r,n) = 1 \} \right|$$

$$\varphi(12) = |\{1, 5, 7, 11\}| = 4$$

$$\varphi(7) = |\{1,2,3,4,5,6\}| = 6$$

$$\varphi(9) = |\{1,2,4,5,7,8\}| = 6$$

allgemein:

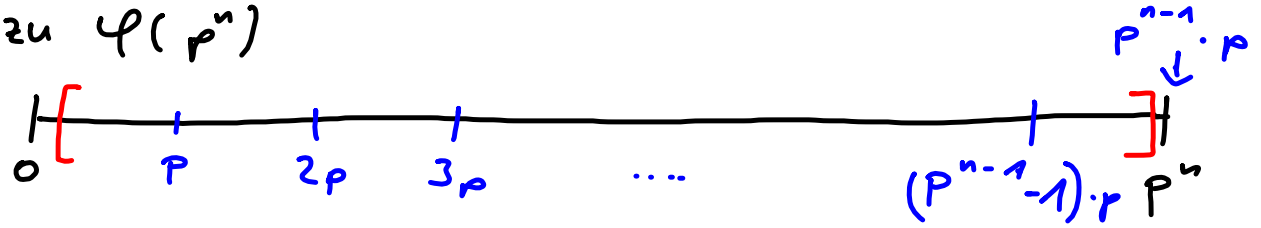
I) $\varphi(p) = p-1$ ist ein Spezialfall von

$$\varphi(p^n) = p^n - p^{n-1}$$

II) $\varphi(m \cdot m') = \varphi(m) \cdot \varphi(m')$ $ggT(m,m') = 1$

Spezialfall: $\varphi(p \cdot q) = (p-1) \cdot (q-1)$

zu $\varphi(p^n)$



$$\underbrace{p^n - 1}_{\text{alle}} - \underbrace{(p^{n-1} - 1)}_{\text{mit gem. Teiler}} = p^n - p^{n-1}$$

zu $\varphi(p \cdot q)$

wieviele teilerfremde Zahlen

$(p-1) \cdot q$

$$\underbrace{p \cdot q - 1}_{\text{alle}} - \underline{(q-1)} - \underline{(p-1)} = pq - p - q + 1$$

$$= (p-1) \cdot (q-1)$$

$$\varphi(p \cdot q)$$

Warum funktioniert RSA?

kleiner Fermat $a^{p-1} \equiv 1 \bmod p$

Verallgemeinerung $a^{\varphi(n)} \equiv 1 \bmod n$ $\quad ggT(a,n)=1$

$d \equiv \frac{1}{e} \bmod \varphi(n)$

$\Rightarrow e \cdot d \equiv 1 \bmod \varphi(n)$

$\Rightarrow \varphi(n)$ Teiler von $e \cdot d - 1$

$\Rightarrow e \cdot d - 1 = t \cdot \varphi(n)$ für einen Teiler t

Verschlüsselung und danach Entschlüsselung bewirkt

$$(m^e)^d \equiv m^{e \cdot d} \equiv m^{1+t \cdot \varphi(n)} \equiv m^1 \cdot (m^{\varphi(n)})^t \equiv m \bmod n$$

RSA Beschleunigungen

1) kleiner Exponent $e \geq 2^{16} + 1$

[nicht zu klein wie etwa $e = 3$

$n = 851$, $e = 3$

$m^e \equiv 343 \bmod 851$ $\quad m^3 = 343 \Rightarrow m = 7$

]

$\rightsquigarrow$ beschleunigt alle Sendevorgänge

2) Chinesischer Restsatz

$m'^d \equiv m \bmod n$ $\quad (d \bmod p-1)$

$m'^d \equiv m \bmod p$

$m'^d \equiv m \bmod q$

$d \bmod q-1$ betrachten

Effekt:

$\Theta((\log n)^3)$ vorher

$(\frac{1}{2}\log n)^3 = \frac{1}{8}\log^3 n$

Empfangsvorgänge um Faktor 4 beschleunigt

Attacken auf RSA

1) sehr kleine Exponenten $e = 3, 5$
2) wenn $d < n^{\frac{1}{4}}$
3) Faktorisierungsalgorithmen
   - exponentiell (Probedivision, Pollard-ϱ, ...) $n^{\frac{1}{2}}$, $n^{\frac{1}{4}}$ ungefährlich für RSA
   - subexponentiell s.u.
   - polynomiell noch keine bekannt

Subexponentielle Algorithmen

Laufzeit

$$\exp\left(b \cdot (\log n)^{a} \cdot (\log \log n)^{1-a}\right)$$

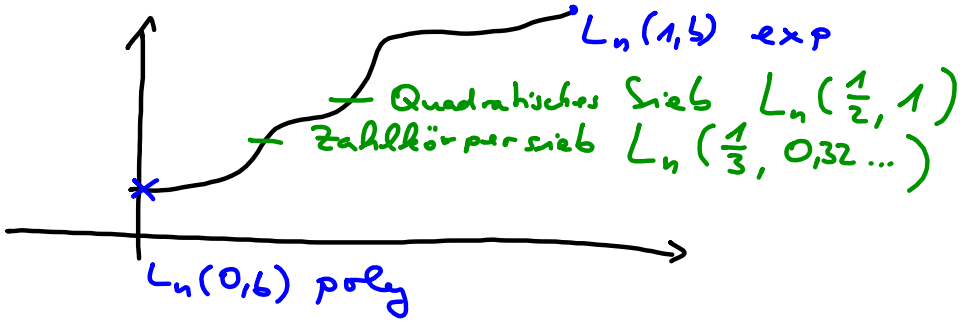
$\log = \ln$

$$= L_n(a, b)$$

$$L_n(0, b) = \exp(b \cdot (\log \log n)^{1}) = \exp(\log \log n)^{b} = \underbrace{(\log n)^{b}}_{\text{polynomiell}}$$

$$L_n(1, b) = \exp(b \cdot \log n) = n^{b}$$

exponentielle Laufzeit



Algorithmen	Schlüssellänge
polynomiell	kryptographisch nicht verwendbar
subexponentiell	$\geq$ 2048 Bit
exponentiell	$\geq$ 128 Bit, oft 256 Bit