# RSA Kryptosystem

Wurde 1977 von Rivest, Shamir und Adleman erfunden.

- Genaue Beschreibung im PKCS #1.
- De-facto Standard für asymmetrische Kryptosysteme.

Schlüsselerzeugung:

Seien $p, q$ zwei verschiedene, ungerade Primzahlen und $n = pq$.
Sei $1 < e < \phi(n)$ teilerfremd zu $\phi(n) = (p-1)(q-1)$ und $1 < d < \phi(n)$ mit $ed = 1 \bmod \phi(n)$.

Der öffentliche Schlüssel ist $(n, e)$.
Der geheimen Schlüssel ist $(p, q, \phi(n), d)$.

$n$ heißt RSA Modul.



# RSA Kryptosystem

Die RSA Funktion wird durch $f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$, $x \mapsto x^e \bmod n$ definiert.

Thm: Die zu $f$ inverse Funktion ist durch $g : \mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$, $y \mapsto y^d \bmod n$ gegeben.

Verschlüsselung: Nachricht als $m \in \mathbb{Z}/n\mathbb{Z}$ darstellen, dann Chiffretext $c = f(m)$ mit öffentlichem Schlüssel berechnen.

Entschlüsselung: $m = g(c)$ mit dem geheimen Schlüssel berechnen.

Zur Beschleunigung der Entschlüsselung kann man nach dem chinesischen Restsatz und mit der Formel von Garner statt in $\mathbb{Z}/n\mathbb{Z}$ auch in $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ rechnen. Dies ist 3-4 mal schneller.



# RSA Parameter

Die Primzahlen $p$, $q$ sollen eine ungefähr gleiche Länge von mindestens $512$ Bit haben, also $n$ eine Bitlänge von $1024$.

Man erwartet, daß $n$ mit $2048$ Bit bis $2022$ und $n$ mit $4096$ Bit bis $2050$ sicher sind. Ein $n$ mit $6800$ Bit entspricht der Sicherheit von AES-128.

Übliche Möglichkeiten für $e$ sind $e \in \{3, 5, 17, 65537\}$.

Der Wert von $d$ hat dann im allgemeinen eine Bitlänge ähnlich wie $n$. Verschlüsseln ist daher viel schneller als entschlüsseln.



# RSA Äquivalenz der Parameter

Thm: Aus einem Teil des geheimen Schlüssels $(p, q, \phi(n), d)$ und dem öffentlichen Schlüssel $(n, e)$ kann man die anderen Teile des geheimen Schlüssels effizient ausrechnen.

- Gegeben $p$ oder $q$ ist das klar.
- Gegeben $\phi(n)$ haben wir zwei Gleichungen $pq = n$, $(p-1)(q-1) = \phi(n)$. Man kann leicht nach $p$ und $q$ auflösen.
- Gegeben $d$ ist $ed - 1$ bei kleinem $e$ ein kleines Vielfaches von $\phi(n)$. Durch Ausprobieren kann man $\phi(n)$ herausbekommen.
- Auch bei beliebigem $e$ kann man aus $d$ die Werte $p, q, \phi(n)$ berechnen (nächste Folie).

Ein Algorithmus zum Berechnen von $\phi(n)$ oder $d$ ist also polynomiell äquivalent zu einem Algorithmus zum Faktorisieren von $n$.

# RSA Äquivalenz der Parameter

Bei beliebigem $e$ berechnen wir $p, q, \phi(n)$ aus $d$ wie folgt:

Sei $s = v_2(ed-1)$ und $k = (ed-1)/2^s$. Für $a \in \mathbb{Z}$ mit $\gcd\{a,n\} = 1$ gilt $a + n\mathbb{Z} \in (\mathbb{Z}/n\mathbb{Z})^\times$ und $(a^k)^{2^s} = 1 \bmod n$, daher hat $a^k$ eine Ordnung $\bmod p$, welche $2^s$ teilt. Gilt nun $(a^k)^{2^i} = 1 \bmod p$ und $(a^k)^{2^i} \neq 1 \bmod q$, so folgt $p = \gcd\{(a^k)^{2^i} - 1, n\}$. Analoges gilt für $q$. Wir wollen die Anzahl solcher $a$ bestimmen.

Sei $\phi : \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ der CRT Isomorphismus. OBdA gelte $v_2(p-1) \leq v_2(q-1)$. Seien $g, h \in \mathbb{Z}$ Erzeuger von $(\mathbb{Z}/p\mathbb{Z})^\times$ bzw. $(\mathbb{Z}/q\mathbb{Z})^\times$. Wenn $v_2(p-1) < v_2(q-1)$ so hat $(g^x, h^y)^k$ mit $x, y \in \mathbb{Z}$ und $y \neq 0 \bmod 2$ an der ersten Koordinate Ordnung $\leq 2^{v_2(p-1)}$ und an der zweiten Koordinate Ordnung $= 2^{v_2(q-1)}$. Dies liefert $(p-1)(q-1)/2$ Werte $a = \phi((g^x, h^y))$ mit $(a^k)^{2^{v_2(p-1)}} = 1 \bmod p$, $(a^k)^{2^{v_2(p-1)}} \neq 1 \bmod q$.



# RSA Äquivalenz der Parameter

Wenn $v_2(p-1) = v_2(q-1)$ so betrachten wir Paare $(x, y)$, wo ein Wert gerade und einer ungerade ist. Es gibt $2(p-1)(q-1)/4$ solche Paare. Die Elemente $a = \phi((g^x, h^y))$ erfüllen $(a^k)^{2^{v_2(p-1)}} = 1 \bmod p$ und $(a^k)^{2^{v_2(p-1)}} \neq 1 \bmod q$ oder umgekehrt.

Zusammenfassend gehen wir wie folgt vor:

- $a \bmod n$ zufällig wählen. $s = v_2(ed-1)$, $k = (ed-1)/2^s$.
- Check $\gcd\{a,n\} \neq 1$.
- Check $\gcd\{(a^k)^{2^i} - 1, n\} \neq 1$ für $0 \leq i \leq s-1$.
- Wenn ein ggT ungleich eins ist, haben wir $p$ bzw. $q$ gefunden.

Die Erfolgswahrscheinlichkeit ist $\geq 1/2$, da es $(p-1)(q-1)$ Elemente $a$ mit $\gcd\{a,n\} = 1$ gibt, und mindestens die Hälfte davon zum Erfolg führt. $\square$



# RSA Äquivalenz der Parameter

Wichtige Folgerung:

- Wird aus irgendeinem Grund $d$ öffentlich, genügt es nicht, für das alte $n$ ein neues, geheimes $d$ zu bestimmen.
- Es macht keinen Sinn, daß mehrere Benutzer sich das gleiche $n$ bei unterschiedlichen, geheimen Schlüsseln teilen.



# RSA Problem

Das RSA Problem ist, zu $(n, e)$ und $c \in \mathbb{Z}/n\mathbb{Z}$ ein $m \in \mathbb{Z}/n\mathbb{Z}$ mit $m^e = c$ zu berechnen (also $e$-te Wurzeln in $\mathbb{Z}/n\mathbb{Z}$ zu ziehen, oder die RSA Funktion zu invertieren).

Dies ist das eigentliche Problem, auf dem die Chosen-Plaintext Sicherheit von RSA beruht.

Wenn man $n$ faktorisieren kann, dann kann man das RSA Problem lösen (klar, $d$ wie in der Schlüsselerzeugung berechnen).

Es ist ein offenes Problem, ob das RSA Problem polynomiell äquivalent zum Faktorisieren von $n$ ist.

# Asymptotische Notationen

Für $x \in \mathbb{R}^n$ und $x_0 \in \mathbb{R}$ schreiben wir $x \geq x_0$, wenn dies koordinatenweise gilt.

Sei $U \subseteq \mathbb{R}^n$ und $g : U \to \mathbb{R}^{>0}$. Wir definieren:

- $O(g) = \{ f \mid (\exists V \subseteq \mathbb{R}^n : f : V \to \mathbb{R})$ und $(\exists x_0, c \in \mathbb{R} : \forall x \in \mathbb{R}^n$ mit $x \geq x_0 : x \in U \cap V$ und $|f(x)| \leq cg(x))\}$.
- Omega$(g)$ definiert wie $O(g)$, aber mit $|f(x)| \geq cg(x)$.
- Theta$(g) = O(g) \cap$ Omega$(g)$.

- $O^{\sim}(g) = \cup_{i \in \mathbb{Z}} O(\log(g)^i g)$.
- $o(g) = \{ f \mid (\exists V \subseteq \mathbb{R}^n : f : V \to \mathbb{R})$ und $\forall \varepsilon > 0 : \exists x_0 \in \mathbb{R} : \forall x \in \mathbb{R}^n$ mit $x \geq x_0 : x \in U \cap V$ und $|f(x)| \leq \varepsilon g(x)\}$



# Faktorisieren ganzer Zahlen

Dies ist ein zentrales Gebiet in der algorithmischen Zahlentheorie.

$L_n(u,v) = \exp((v + o(1)) \log(n)^u \log(\log(n))^{1-u})$

Namen und Laufzeiten einiger Methoden:

- Probedivision bis $n^{1/2}$: $O^{\sim}(n^{1/2})$.
- Pollard $p-1$ Methode: Im allgemeinen ähnliche Laufzeit wie Probedivision.
- Pollard $\rho$ Methode: $O^{\sim}(n^{1/4})$.
- Elliptische Kurven Methode: $L_p(1/2, c)$ mit $p$ kleinster Primteiler von $n$ und $c$ kleine Konstante. Verallgemeinert $p-1$ Methode.
- Klassengruppen Methode: $L_n(1/2, 1)$ (bewiesen, aber ineffizient).
- Quadratisches Sieb: $L_n(1/2, 1)$ (effizient).
- Zahlkörpersieb: $L_n(1/3, (64/9)^{1/3})$.



# Faktorisieren ganzer Zahlen

Die ersten drei Methoden haben also in $\log(n)$ exponentielle Laufzeit, während die anderen sogenannte subexponentielle Laufzeit haben.

Die elliptische Kurven Methode (ECM) ist besonders geeignet, wenn $n$ einen relativ kleinen Primfaktor hat. Ansonsten ist das quadratische Sieb (QS) im allgemeinen schneller, und am schnellsten ist das Zahlkörpersieb (GNFS), aber erst für sehr große Zahlen.

Um grobe Größenangaben zu machen: Man verwendet ECM zum Beispiel bis $40$ Dezimalstellen oder für einen gewissen Bruchteil der erwarteten Laufzeit des QS, danach QS bis 100 Dezimalstellen, und dann das GNFS.

Faktorisierungsprojekte im Netz: NFSNET und ECMNET.



# RSA Sicherheit

RSA wie bisher beschrieben wird häufig „plain RSA“ oder „textbook RSA“ genannt. Im Sinn strenger Sicherheitsmodelle (IND-CCA2) ist es vollkommen unsicher.

Im folgenden werden ein paar Fehler bei der Benutzung von „plain RSA“ diskutiert. Wir beschränken uns auf solche Fehler, welche das Lösen des RSA Problems ermöglichen.

Kritische Punkte sind:

- die Wahl der Parameter $n, e, d$.
- mögliche Struktur in den Nachrichten $m$.
- die Verwendung des Systems.

# RSA: Kleines $d$

Wenn $d$ sehr klein ist, kann es durch Ausprobieren erraten werden.
Für kleines $d$ gibt es den Angriff von Wiener:

Thm (Wiener): Gilt $q < p < 2q$ und $d < n^{1/4}/3$, so kann $d$ aus $n, e$ (mit $e < \phi(n)$) effizient berechnet werden.

Der Beweis verwendet Kettenbrüche.

Im Satz genügt $d < n^{0.292}$ (Boneh und Durfee).
Man vermutet, daß $d < n^{1/2}$ bereits ausreicht.



# RSA: Kleines $e$

Hastad Broadcast Angriff.

Vereinfachte Version: Annahme, daß Nachricht $m$ mit Exponent $e$ bezüglich $k \geq e$ vielen RSA moduli $n_i$ verschlüsselt ist.

- Also $c_i = m^e \bmod n_i$ für $1 \leq i \leq k$.
- Sei $n = \prod_i n_i$. OBdA $\gcd\{n_i, n_j\} = 1$.
- Chinesischer Restsatz liefert $0 \leq c < n$ mit $c = c_i \bmod n_i$.
- Es gilt $c = m^e \bmod n$.
- Wegen $m < n_i$ und $e \leq k$ folgt $m^e < n$ und $c = m^e$ in $\mathbb{Z}$, also $m = c^{1/e}$.

Allgemeine Version:

Thm (Coppersmith): Sei $f \in \mathbb{Z}[t]$ normiert und $d = \deg(f)$. Dann können die Lösungen $f(x) = 0 \bmod n$ mit $x \in \mathbb{Z}$ und $|x| < n^{1/d}$ effizient berechnet werden.
Der Beweis verwendet Gitter und LLL Reduktion.



# RSA: Kleines $e$

Thm (Hastad): Seien $n_i$ teilerfremd und $g_i \in \mathbb{Z}[x]$ normiert mit $d = \max_i \deg(g_i)$, für $1 \leq i \leq k$. Es gebe ein eindeutiges $m < \min_i n_i$ mit $g_i(m) = 0 \bmod n_i$. Ist $k > d$, so kann $m$ effizient aus den $n_i$ und $g_i$ berechnet werden.

Bew: Sei $n = \prod_i n_i$ und $g = \prod_i \lambda_i x^{d-\deg(g_i)} g_i$, wobei $\lambda_i = \delta_{i,j} \bmod n_j$, und $\sum_i \lambda_i = 1$ (chinesischer Restsatz). Dann ist $g$ normiert, $\deg(g) = d$ und $g(m) = 0 \bmod n$. Wegen $m < \min_i n_i < n^{1/k} < n^{1/d}$ kann $m$ mit dem Satz von Coppersmith effizient bestimmt werden. □

Anwenden mit $g_i(m) = f_i(m)^{e_i} - c_i \bmod n_i$.

- Speziell $e_i = e$ und lineare $f_i$ (z.B. festes Padding, $f_i(m) = m + b_i$).
- Im allgemeinen kann auf die Bedingung $g_i$ normiert verzichtet werden.

Gegen Broadcast Angriff hilft randomisiertes Padding von $m$.