RSA Kryptosystem

Wurde 1977 von Rivest, Shamir und Adleman erfunden.

- Genaue Beschreibung im PKCS #1.
- RSA US-Patent 2000 ausgelaufen (aber nicht für spezielle Techniken - „multiprime RSA“).
- De-facto Standard für asymmetrische Kryptosysteme.
- 1973 im geheimen vom CESG (UK) erfunden.

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 $(p, q, \phi(n), d)$.

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.

Bew: Wir müssen zeigen: $x^{ed} = x$ für $x \in \mathbb{Z}/n\mathbb{Z}$. Es gilt $ed = 1 + m(p-1)$ für ein $m \in \mathbb{Z}$ und $x^{p-1} = 1$ für $x \in (\mathbb{Z}/p\mathbb{Z})^\times$, daher auch $x^{ed} = xx^{m(p-1)} = x$ für alle $x \in \mathbb{Z}/p\mathbb{Z}$. Analog für q. Also gilt $x^{eq} = x$ für $x \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ und wegen $\mathbb{Z}/n\mathbb{Z} \cong \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ auch für $x \in \mathbb{Z}/n\mathbb{Z}$. $\square$

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

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

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.

Primzahlerzeugung

Thm (Primzahlsatz): Sei $\pi(x) = \#\{p \mid p \text{ Primzahl}, 1 \leq p \leq x\}$.
Für $x > 17$ gilt $x/\log(x) < \pi(x) < 1.25506x/\log(x)$.
Für $x \to \infty$ gilt $\pi(x)/(x/\log(x)) \to 1$.

Strategie:

- Zufällige ganze Zahlen x der gewünschten Bitlänge b erzeugen.
- Testen, ob x prim. Erfolgswahrscheinlichkeit $1/(b\log(2))$.

Primzahltests:

- Miller-Rabin: Eine Antwort „ist prim“ ist mit Wahrscheinlichkeit $\leq (1/4)^r$ falsch, bei r Iterationen. Kann man unter 2^{-128} drücken. Eine Antwort „ist nicht prim“ ist immer richtig.
- Einige andere probabilistische Tests ...
- Agrawal-Kayal-Saxena (2002): Deterministischer, polynomieller Primzahltest („primes in P“).

RSA Sicherheit

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.