11.2012, Jakob Schmusch, K1d, Gymnasium Balingen

Thesenpapier: asymmetrische Verschlüsselung RSA mit Hilfe von Primzahlen

Anders als bei symmetrischen Verschlüsselungsverfahren gibt es bei der asymmetrischen Verschlüsselung zwei Schlüssel. Nach der Verschlüsselung der Informationen mit dem „öffentlichen“ Schlüssel können die Daten nur mit dem geheimen „privaten“ Schlüssel wieder entschlüsselt werden. Es wird also mit einem anderen Schlüssel aufgeschlossen, als abgeschlossen wurde. Das hat den Vorteil, dass der Schlüssel zum entschlüsseln nicht transportiert werden muss, und dadurch nicht abgefangen werden kann.

Ronald **R**ivest, Adi **S**hamir, und Leonard **A**dleman erfanden 1977 ein Verfahren, mit dem die asymmetrische Verschlüsselung mathematisch realisiert wurde. Dieses nach ihnen benannte Verfahren wird RSA-Algorithmus genannt, welcher im Folgenden dargestellt wird.

Für jedes Ziel der Datenübertragung wird ein Schlüsselpaar bestehend aus öffentlichem Schlüssel (engl. public key) (**N**, **e**) und privatem Schlüssel (**d**) (engl. private key) generiert. Dabei soll man von dem öffentlichen Schlüssel nicht auf den privaten Schlüssel schließen können, um sicherzustellen, dass nur der rechtmäßige Empfänger die Nachricht entschlüsseln kann. Dies nennt man eine Einwegfunktion. Beispiele für eine solche ist das Drücken von Zahnpasta aus einer Tube oder die Multiplikation zweier großer Primzahlen. Je größer das Produkt zweier Primzahlen, desto schwieriger ist es, die zwei Primfaktoren zu erraten.

(p und q sind im Folgenden immer Primzahlen)

$p \cdot q = 15$ => p = 3; q = 5

$p \cdot q = 22$ => p = 2; q = 11

$p \cdot q = 204091$ => p = ?; q = ? Hier ist es nur mit großem Aufwand möglich, p und q zu finden.

Ist das Produkt also groß genug, kann man es bedenkenlos als öffentlichen Schlüssel veröffentlichen.

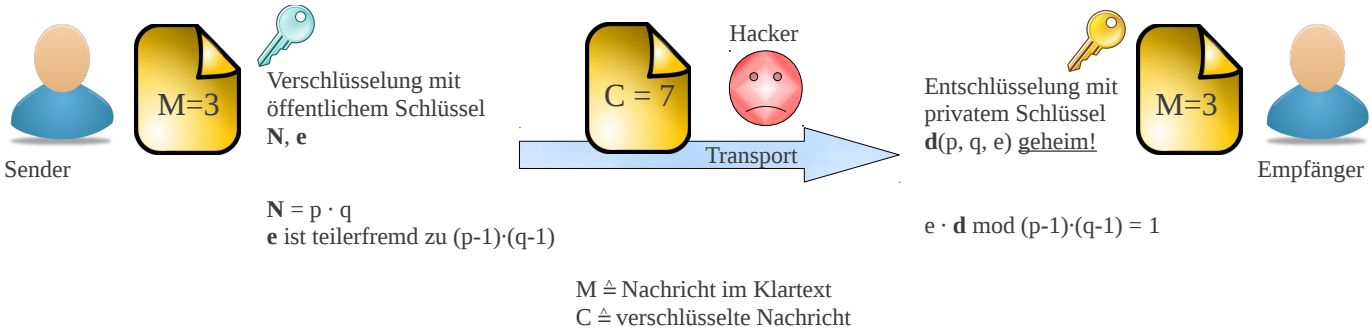
Es ist also notwendig, möglichst effizient große Primzahlen generieren zu können.

Zusätzlich generiert man einen weiteren öffentlichen Wert **e**, der zu (p-1)·(q-1) teilerfremd ist.

Der private Schlüssel wird nun in Abhängigkeit von **p**, **q** und **e** generiert. Es gilt:

e · d mod (p-1)·(q-1) = 1 (d wird mit dem euklidischen Algorithmus berechnet – wird hier nicht erläutert)

Hier nochmal zur Veranschaulichung eine schematische Darstellung:



Nach der Generierung des Schlüsselpaars können jetzt mit Hilfe des RSA-Algorithmus Zahlen verschlüsselt werden.

Zum Verschlüsseln:

$$C = M^e \bmod N \qquad M<N$$

Zum Entschlüsseln:

$$M = C^d \bmod N$$

Beispiel:

(Mit (zu) kleinen Zahlen)

Wir wollen uns auf die Online-Banking Webseite unserer Bank einloggen und müssen zur Authentifizierung unser Passwort übermitteln. Da dies aber nicht im Klartext abgefangen können werden soll, wird RSA-Verschlüsselung eingesetzt.

Unsere Bank hat zu diesem Zweck ihren öffentlichen Schlüssel mit **N=77** und **e=13** veröffentlicht.

Das Passwort, welches wir sicher übermitteln wollen lautet **M=6**. (M<N => Verschlüsselung möglich)

Verschlüsselung:

$C = M^e \bmod N$ **M<N**

$C = 6^{13} \bmod 77$

$C = 13060694016 \bmod 77$ 13060694016 : 77 = 169619402 Rest 62

C = 62

Unser Passwort „6“ ist jetzt verschlüsselt und kann nun „sicher“ (N zu klein) als „62“ versendet werden.

Entschlüsselung:

Nachdem die „62“ bei der Bank angekommen ist, muss die Zahl von ihr wieder entschlüsselt werden, um sie mit ihrer Datenbank abzugleichen. Dazu nutzt sie ihren geheimen privaten Schlüssel **d=37**.

$M = C^d \bmod N$

$M = 62^{37} \bmod 77$

M = 6

Sicherheit:

Doch wodurch ist jetzt die Sicherheit gewährleistet?

Würde die Zahl 62 beim Transport abgefangen, könnte sie nicht entschlüsselt werden, da der Angreifer dazu den privaten Schlüssel **d** bräuchte. Berechnen kann er ihn in absehbarer Zeit auch nicht, da er dazu N=77 in die zwei Primfaktoren p und q zerlegen müsste, was bei großen Zahlen schwer ist.

((e · **d**) mod (**p**-1)·(**q**-1) = 1)

Generierung der Schlüssel:

Doch wie kommen die Schlüssel zustande?

Zuerst werden zwei große Primzahlen **p** und **q,** sowie ein Exponent **e** generiert, welcher zu **(p-1)·(q-1)** teilerfremd ist.

p = 7; q = 11;

e = 13 da teilerfremd zu (7-1)·(11-1) = 60

Das Produkt von p und q ergibt das extrem große Produkt **N**.

N = 7·11 = 77

Der private Schlüssel **d** wird aus **(e · d) mod (p-1)·(q-1) = 1** errechnet. (Umstellung: euklidischen Algorithmus)

Setzt man für d den privaten Schlüssel (s.o.) 37 ein, geht die Gleichung auf.

Quelle: http://www.matheprisma.uni-wuppertal.de/Module/RSA/index.htm