RSA Kryptosystem

In Zeiten von Datenspionage, Hausdurchsuchungen und Datenklau ist es immer wichtiger seine Daten gut zu verschlüsseln. Ob es nun die Passwörter sind die man auf dem PC hat oder in Dantebanken oder ob es sensible Datensätze sind.

Hierzu gibt es viele Möglichkeiten und Algorhytmen einen dieser Algorhytmen möchte ich in diesem Paper näher behandeln.

RSA ist ein asymmetrischer Algorhytmus d.h. er ist langsamer als z.B AES (Advanced Encryption Standard) oder DES (Data Encryption Standard). Doch wenn er langsamer ist wozu kann man ihn dann benutzen? Er ist bei kleineren Daten durchaus gut verwendbar. Im Klartext heißt dies, das man mit ihm den „Schlüssel" von einem symmetrischen Verfahren Krypten kann um eine höhere Sicherheit zu erhalten oder eine Andere Möglichkeit wäre ihn als digitale Signatur zu verwenden.

Doch wie funktioniert er?

Da es eine Einwegverschlüsselung ist muss eine „tdowf" (*trap door one-way function)* eingebaut werden damit auch eine Entschlüsselung möglich ist. Hierzu werden der Public und Privat Key verwendet.

Public-Key= RSA Modul und Verschlüsselungsexponent
Private-Key=RSA Modul und Entschlüsselungsexponent.

In der Theorie sieht das ganze dann so aus:

1. Zwei zufällige Primzahlen werden ausgewählt(Hierbei sollte drauf geachtet werden das die Primzahlen nicht größer als 100 sind da es sonnst ziemlich lange dauern könnte bei der Verschlüsselung)
2. Primzahl1*Primzahl2 = RSA Modul
3. Berechnung der Eulerischen Funktion vom RSA Modul
4. Eine zufällige zahl zwischen 1 und dem Ergebnis von 3
5. Berechnung des Entschlüsselungsexponent indem die Multiplikativ Inverse von 4 genommen wird im Bezug auf 3

In der Praxis sieht das dann wie folgt aus (Beispiel aus WIkipedia):

1. Wir wählen *p* = 11 und *q* = 13 für die beiden Primzahlen.
2. Der RSA-Modul ist $N = p \cdot q = 143$.
3. Die eulersche φ-Funktion nimmt damit den Wert $\varphi(N) = \varphi(143) = (p-1)(q-1) = 120$ an.
4. Die Zahl *e* muss zu 120 teilerfremd sein. Wir wählen *e* = 23. Damit bilden *e* = 23 und *N* = 143 den öffentlichen Schlüssel.
5. Berechnung der Inversen zu *e*:
   Es gilt: $e \cdot d + k \cdot \varphi(N) = 1 = ggT(e, \varphi(N))$
   bzw. im konkreten Beispiel: $23 \cdot d + k \cdot 120 = 1 = ggT(23, 120)$
   Mit dem erweiterten euklidischen Algorithmus berechnet man nun die Faktoren *d* = 47 und *k* = − 9, so dass die Gleichung aus dem Beispiel wie folgt aussieht: $23 \cdot 47 + (-9) \cdot 120 = 1$
   d ist der private Schlüssel, während k nicht weiter benötigt wird.

Somit wird deutlich das durch ein recht „smiples" Verfahren eine hohe Sicherheit gewährleistest werden kann.
Doch nun wurde immer noch kein Text verschlüsselt. Dies funktioniert mit dem Modulo zum RSA Modul und der Potenz von *e* zum Klartext.
Als Beispiel:

RSA Modul = 155
E = 32
Klartext = 20

Dann wäre die Formel:
$20^{32} \bmod 155 = 90$
K^e mod N = C

Somit wäre das gecryptete Ergebnis 90.

Nun da wir etwas mit RSA crypten können muss es schließlich auch möglich sein das ganze wieder zu Entschlüsseln. Dies geht mit dem Privat Key. Die Formel dafür ist fast die selbe wie beim Verschlüsseln nur das wir hier den Privat Key als Exponenten zum Hash-Wert haben.
K = C^d mod N

MFG
n0x *„Only god can judge me"*