RSA-Verschlüsselung

Inhaltsverzeichnis

1.) Einleitung
(Kryptographie, Arithmetik)
2.) Eulersche φ-Funktion
3.) Miller-Rabin-Test
4.) Fermat-Satz
5.) Euklidischer Algorithmus
4.1 Der einfache euklidische Algorithmus
4.2 Der moderne euklidische Algorithmus
4.3 Der erweiterte euklidische Algorithmus
6.) RSA
7.) ((Programmtechnische Umsetzung))
8.) Schlusswort
9.) Quellen

Einleitung

In diesem Text werde ich die RSA-Methode als eine der modernen Verschlüsselungsverfahren behandeln. Kryptographische Verfahren gab es bereits im dritten Jahrtausend v.Chr., sie wurden zum Schutz des Diplomatischen Briefverkehrs genutzt. Diese Verfahren waren allerdings sehr leicht zu knacken, und deshalb wurden diese immer weiter verbessert. Das Problem dabei ist jedoch, dass um eine unknackbares Verfahren zu entwickeln ein Verschlüsselungsschlüssel von der gleichen Länge wie der zu verschlüsselnde Text benötigt werden würde. Dieser müsste aber geheim bleiben – daher könnte man auch direkt die Nachricht geheim übertragen. Deshalb hat man auf einen immer längeren Schlüssel und eine kompliziertere Verschlüsselung gesetzt (wie die Enigma im 2. Weltkrieg). Allerdings blieben diese Verfahren von dritten immer mit viel Aufwand decodierbar. Besonders in de Zeit der Hochleistungscomputer können sehr schnell sehr viele Kombinationen durchprobiert werden und die klassischen Methoden sind von daher vollkommen unsicher. Aus diesem Grunde wurde die RSA-Verschlüsselungstechnik entwickelt.

Nachfolgend werden wir viel mit *modularer Arithmetik* umgehen, daher werde ich eben eine kleine Einführung geben.
Die modulare Arithmetik (oder auch „Rechnen mit Rest“) ist im Prinzip ganz einfach. Wir haben das schon alle in der Grundschule gemacht.

Früher (in der Grundschule) hat man eine Gleichung wie etwa *25 / 7* so gelöst:
*25 / 7 = 3 Rest **4***

Die Rechnung könnten wir auch als Modulus (mod) aufschreiben:
$25 = \mathbf{4} \bmod 7$

Anschließend möchte sollten wir einige Rechentricks uns ansehen.
Wie zum Bespiel löst man die Rechnung $17^{28} \bmod 5$ im Kopf?

Als erstes nimmt man die Zahl 28 auseinander: 28 = 16 + 8 + 4

Daraus ergibt sich:
$17^{28} \equiv 17^{16} * 17^{8} * 17^{4}$
Nun Rechnet man das Modul von 17 aus.
$17 \equiv \mathbf{2} \bmod 5$
Das bedeutet:
$17^{2} \equiv \mathbf{2}^{2} \bmod 5 \equiv 4 \bmod 5$
$17^{4} \equiv 4^{2} \bmod 5 \equiv 1 \bmod 5$
$17^{8} \equiv 1^{2} \bmod 5 \equiv 1 \bmod 5$
$17^{16} \equiv 1^{2} \bmod 5 \equiv 1 \bmod 5$

$17^{28} \equiv 17^{16} * 17^{8} * 17^{4} \equiv 1 * 1 * 1 \equiv 1 \bmod 5$

Eulersche φ-Funktion

Die eulersche φ-Funktion(auch eulersche Funktion genannt) ist eine zahlentheoretische Funktion. Sie gibt für jede Zahl *n* an, wie viele positive ganze Zahlen $a < n$ zu ihr teilerfremd sind.

Beispiele

- Die Zahl 6 ist zu zwei der Zahlen von 1 bis 6 teilerfremd (1 und 5), also ist $\varphi(6) = 2$.
- Die Zahl 13 ist als Primzahl zu den zwölf Zahlen 1 bis 12 teilerfremd, also ist $\varphi(13) = 12$.
- Die ersten zehn Werte der φ-Funktion lauten:

n	1	2	3	4	5	6	7	8	9	10
teilerfremde Reste	1	1	1, 2	1, 3	1, 2, 3, 4	1, 5	1, 2, 3, 4, 5, 6	1, 3, 5, 7	1, 2, 4, 5, 7, 8	1, 3, 7, 9
φ(n)	1	1	2	2	4	2	6	4	6	4

Eigenschaften

Die φ-Funktion ist eine multiplikative zahlentheoretische Funktion. Das heißt, dass für teilerfremde Zahlen *m* und *n* die Gleichung

$$\varphi(m * n) = \varphi(m) * \varphi(n)$$

gilt.

Berechnung

Da jede Primzahl *p* nur durch 1 und sich selbst teilbar ist, ist sie zu den Zahlen 1 bis $p - 1$ teilerfremd. Es gilt daher:

$$\varphi(p) = p-1$$

Miller-Rabin-Test

Das Rabin-Miller-Verfahren ist ein statistisches Verfahren um hoch wahrscheinliche Primzahlen zu finden. Dabei liefert dieses Verfahren keine 100%igen Primzahlen sondern filtert nur eindeutig nicht prime Zahlen heraus. Die Wahrscheinlichkeit bei mehreren Durchläufen dieses Testes eine nicht prime Zahl zu erhalten ist jedoch geringer als das Risiko, dass Empfänger und Sender einer Nachricht gleichzeitig vom Blitz erschlagen werden.
Hier das Verfahren:

1. Wähle *n* als zu testende Zahl (ungerade)
2. Berechne *d*: n - *1* = 2^s * *d*
3. Wähle zufällig eine Zahl *a,* für die gilt *1* < *a* < *n*
4. Wenn $a^d \equiv 1 \pmod n$ oder $a^{d2r} \equiv -1 \pmod n$ ist *n* eine starke Pseudoprimzahl oder eine Primzahl

Man kann diesen Test noch mit anderen Verfahren, wie z.B. mit dem Lucas-Lehmer-Test, kombinieren, in der Regel reicht dieser Test jedoch bei mehrfacher Anwendendung auf ein *n*.

Fermat-Satz

Der **kleine fermatsche Satz**, kurz „der **kleine Fermat**“, ist ein Lehrsatz der Zahlentheorie. Er macht eine Aussage über die Eigenschaften von Primzahlen und wurde im 17. Jahrhundert von Pierre de Fermat aufgestellt. Der Satz beschreibt die allgemein gültige Kongruenz:

$a^p \equiv a \pmod{p}$

wobei a eine ganze Zahl und p eine Primzahl ist.

Verallgemeinerung

Man kann den kleinen Fermatschen Satz zum Satz von Euler verallgemeinern.

Für zwei teilerfremde Zahlen n und a gilt:

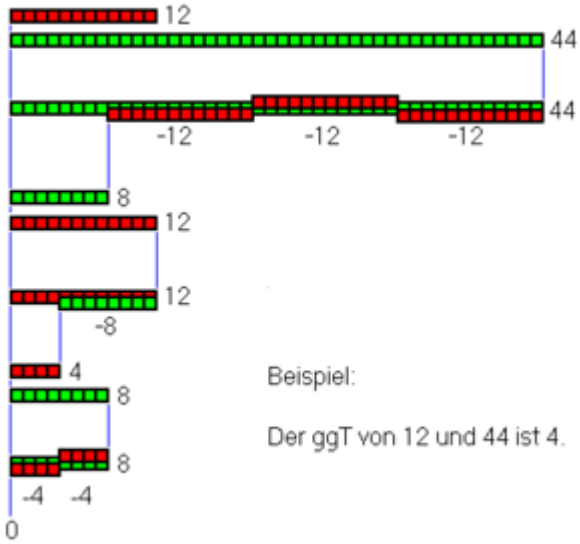
$a^{\varphi(n)} \equiv 1 \pmod{n}$

wobei $\varphi(n)$ die eulersche φ-Funktion bezeichnet. Diese liefert die Anzahl der Zahlen zwischen 1 und n-1 welche teilerfremd zu n sind. Ist n eine Primzahl, so ist $\varphi(n) = n - 1$, so dass man Fermats kleinen Satz als Spezialfall erhält.

Euklidischer Algorithmus

Der einfache Euklidische Algorithmus

Der euklidische Algorithmus ist ein Algorithmus aus dem mathematischen Teilgebiet der Zahlentheorie. Mit ihm lässt sich der größte gemeinsame Teiler(ggT) Zweier natürlicher Zahlen berechnen. Der euklidische Algorithmus ist der älteste nicht-triviale Algorithmus und wird heute noch eingesetzt. Er wurde von dem griechischen Mathematiker Euklid um 300 v.Chr. niedergeschrieben.
Um den ggT zu finden zog Euklid wiederholt die kleinere der beiden Längen von der größeren ab.



Euklid Graphisch (Quelle: Wikipedia.org)

Der moderne Euklidische Algorithmus

Wenn jedoch die Differenz der beiden Zahlen sehr groß ist, braucht man sehr viele Schritte. Deshalb benutzt man heute hauptsächlich den Division-Algorithmus, der anstatt in vielen Schritten die kleinere Zahl abzuziehen, den Rest der Division der größeren Zahl durch die kleinere berechnet.

a und r_0 sind die Ausgangszahlen, wobei $a < r_o$.

$a = q_1 * r_0 + r_1 \quad a \equiv r_1 \ (mod\ r_0)$
$r_0 = q_2 * r_1 + r_2 \quad r_0 \equiv r_2 \ (mod\ r_1)$
$r_1 = q_3 * r_2 + r_3 \quad usw.$
.
.
.
$r_{n-1} = q_{n+1} * r_n + 0$

Der Divisor der letzten Division ist dann der ggT.
$ggT(a, b) = r_n$

Erweiterter Euklidischer Algorithmus

Der erweiterte Euklidische Algorithmus ermittelt neben dem ggT auch noch zwei ganze Zahlen *s* und *t*, die folgende Gleichung erfüllen

$ggT(a, b) = s * a + t * b$

Hier der einfache Euklid mit den Zahlenpaar 99, 78:

$99 = 1 * 78 + 21$
$78 = 3 * 21 + 15$
$21 = 1 * 15 + 6$
$15 = 2 * 6 + 3$
$6 = 2 * 3 + 0$

Für den erweiterten Euklid wird das ganze rückwärts aufgewickelt:

$3 = 15 - 2 * 6$
$= 15 - 2 * (21 - 1 * 15)$ $= 3 * 15 - 2 * 21$
$= 3 * (78 - 3 * 21) - 2 * 21$ $= 3 * 78 - 11 * 21$
$= 3 * 78 - 11 * (99 - 1 * 78)$ $= \mathbf{14} * 78 \ \mathbf{- 11} * 99$

Im ersten Schritt wird die vorletzte Gleichung des einfachen Euklids nach r_{n-1} umgestellt. Anschließend wird aus dem jeweils darüber liegenden Schritt der Rest rekursiv eingesetzt.
Ergebnis: *s = 14 und t = -11*.

RSA

Bei der Nachrichtenübermittelung gibt es immer zwei Seiten und den *Spion*, in meinem Fall Empfänger *E* und Sender *S* .
Wenn S Nachricht verschlüsseln will, tut er dies mit dem *öffentlichen Schlüssel* (*public key*) der von E bereitgestellt wird. Das Entschlüsseln mit dem *öffentlichen Schlüssel* ist unmöglich, sondern ist nur mit einem *geheimen Schlüssel* (*privater Schlüssel*, p*rivate key*) von *E* möglich. Da der *private Schlüssel* sehr lang ist, gibt es viele Kombinationen, die ein *Spion* ausprobieren müsste. Das bedeutet jedoch auch, das nicht einmal der Sender die Nachricht wieder entschlüsseln könnte.
Der *öffentliche Schlüssel* setzt sich dabei aus dem p*rivate Schlüssel* und einer anderen Primzahl zusammen

Erzeugung des öffentlichen und privaten Schlüssels

Der *öffentliche Schlüssel* ist ein Zahlenpaar (*e*,*N*) und der *private Schlüssel* ein Zahlenpaar (*d*,*N*), wobei *N* bei beiden Schlüsseln gleich ist. Man nennt *N* den *RSA-Modul*, *e* den *Verschlüsselungsexponenten* und *d* den *Entschlüsselungsexponenten*. Diese Zahlen werden durch das folgende Verfahren erzeugt:

1. Wähle zufällig und stochastisch unabhängig zwei Primzahlen p ungleich q mit ca. 150 Stellen. In der Praxis rät man dazu jeweils eine Zahl und führt mit dieser anschließend einen Primzahltest (*siehe Miller-Rabin*) durch.
2. Berechne das RSA-Modul *N*
   $N = p * q$
3. *Berechne die* Eulersche φ-Funktion *von N*
   $\varphi(N) = \varphi(p,q) = (p-1)(q-1)$
4. *Wähle eine zu φ(N)* teilerfremde Zahl *e,* für die gilt $1 < e < \varphi(N)$
   Teilerfremd bedeutet: $ggT(N, e) = 1$ *(siehe „Euklidische Algorithmus“)*
   Aus Effizienzgründen wird e klein gewählt, üblich ist $2^{16} + 1 = 65537$
5. Berechne den Entschlüsselungsexponenten *d*:
   $e*d \equiv 1 \pmod{\varphi(N)}$
   (mit Zuhilfenahme des erw. Euklids)

Damit haben wir alle benötigten Schlüssel (*e, d, N*). Alle anderen Variablen sollten auf sichere Weise gelöscht werden.

Verschlüsseln von Nachrichten

Um eine Nachricht *K* zu verschlüsseln, verwendet der Absender die Formel

$C \equiv K^e \pmod{N}$

und erhält so aus dem Klartext *K* den Geheimtext *C*.

Entschlüsseln von Nachrichten (Decodieren)

Der Geheimtext *C* kann durch modulare Exponentiation wieder zum Klartext *K* entschlüsselt werden. Der Empfänger benutzt die Formel

$K \equiv C^d \pmod{N}$

mit dem nur ihm bekannten Wert *d* sowie *N*.

$c^d \equiv (K^e)^d \bmod N \equiv K^{e*d} \bmod N \equiv K^{\varphi(N)+1} \bmod N \equiv K^{\varphi(N)} * K^1 \bmod N \equiv K \bmod N$

Praktisches Beispiel

Vorarbeiten

Um einen Text zu verschlüsseln, müssen zunächst Buchstaben in Zahlen umgewandelt werden. Dazu verwendet man in der Praxis zum Beispiel den ASCII-Code. Hier sei willkürlich die folgende Zuordnung gewählt:

```
A=01 B=02 C=03 usw. (00 = Leerzeichen)
```

Darüber hinaus sei angenommen, dass jeweils drei Zeichen zu einer Zahl zusammengefasst werden. Die Buchstabenfolge „AXT“ wird also zu „012420“. Die kleinste zu verschlüsselnde Zahl ist dann 000000 (drei Leerzeichen), die größte 262626 (ZZZ). Der Modulus $N = p * q$ muss also größer 262626 sein.

```
Klartext:  W I K  I P E  D I A
Kodierung: 230911 091605 040901
```

Schlüsselerzeugung

Zunächst werden geheim zwei Primzahlen gewählt, beispielsweise $p = 307$ und $q = 859$. Damit ergibt sich:

$N = p * q = 263713$

$\varphi(N) = (p - 1) * (q - 1) = 262548$

$e = 1721$ (zufällig, teilerfremd zu $\varphi(N)$)

$d = 1373$ (das multiplikative Inverse zu $e \bmod \varphi(N)$ mit Hilfe des Erweiterten euklidischen Algorithmus)

Öffentlicher Schlüssel: $e = 1721$ und $N = 263713$

Geheimer Schlüssel: $d = 1373$ und $N = 263713$

Verschlüsselung

$C_n = K_n^e \bmod N \quad \text{für } n=1,2,3(,...)$

$C_1 = 230911^{1721} \bmod 263713 = 001715$

$C_2 = 091605^{1721} \bmod 263713 = 184304$

$C_3 = 040901^{1721} \bmod 263713 = 219983$

Entschlüsselung

$K_n = C_n^d \bmod N \quad \text{für } n=1,2,3(,...)$

$K_1 = 001715^{1373} \bmod 263713 = 230911$

$K_2 = 184304^{1373} \bmod 263713 = 091605$

$K_3 = 219983^{1373} \bmod 263713 = 040901$

Schlusswort

RSA ist eine sehr sichere und viel benutzte Verschlüsselung. Sie ist jedoch im Vergleich zu 3DES und AES mindestens um einen Faktor 1.000 langsamer. Deshalb benutzt man für die Verschlüsselung größerer Datenmengen hybride Verfahren. Diese setzen symmetrische sowie die asymmetrische Verfahren ein. Es genügt jedoch, RSA zum Austausch eines Schlüssels für die symmetrische Verschlüsselung zu benutzen.

Quellen

- Wikipedia.org
  - → http://de.wikipedia.org/wiki/Phi-Funktion
  - → http://de.wikipedia.org/wiki/Kleiner_fermatscher_Satz
  - → http://de.wikipedia.org/wiki/RSA-Kryptosystem
  - → http://de.wikipedia.org/wiki/Miller-Rabin-Test
  - → http://de.wikipedia.org/wiki/Euklidscher_Algorithmus
  - → http://de.wikipedia.org/wiki/Erweiterter_Euklid

- Bg-stJohann.tsn.at
  - → http://www.bg-stjohann.tsn.at/homepage/faecher/informatik/fischer/node29.html
  - → http://www.bg-stjohann.tsn.at/homepage/faecher/informatik/fischer/node31.html

- uni-erlangen.de
  - → http://www2.informatik.uni-erlangen.de/Lehre/SS2006/HalloWelt/zahlentheorie.pdf?language=de

- Und viele weitere

Hinweis:

Als Multiplikationszeichen wurde ein Stern (*) und als Dividierungszeichen ein Schrägstrich (/) benutzt.