ANWENDUNG UND HINTERGRÜNDE DES RSA-VERFAHRENS

Einführung

Schon seit Jahrtausenden versuchen die Menschen Geheimschriften zu entwickeln, um Nachrichten vor unbefugtem Zugriff zu schützen. Im heutigen Zeitalter der elektronischen Datenverarbeitung, speziell bei der Übermittlung im Internet, ist es wichtiger denn je geworden, sichere und komfortable Verschlüsselungsverfahren zu produzieren. Dabei hat sich in den letzten Jahren das 1977 von Ronald **R**ivest, Adi **S**hamir und Leonard **A**dleman entwickelte *RSA-Verfahren* durchgesetzt.
Heute wird es vor allem bei der Verschlüsselung im Internet benutzt und besitzt dort so etwas wie eine Monopolstellung. Folglich ist das *RSA-Verfahren* eines der berühmtesten Chiffrierverfahren überhaupt geworden.

Praktische Anwendung

Das *RSA-Verfahren* benutzt zwei öffentliche Schlüssel e, n und einen geheimen (privaten) Schlüssel d. Die öffentlichen Schlüssel werden für jeden frei zugänglich z.B. im Internet veröffentlicht. Der geheime Schlüssel wird jedem Teilnehmer einzeln von einer Schlüsselverteilungszentrale zugeteilt.
Die Anwendung des Verfahrens ist sehr einfach. Wir müssen dazu an dieser Stelle lediglich die Modulo-Division kennen lernen. Der Unterschied zur normalen Division besteht hierbei lediglich, dass das Ergebnis der Operation nicht der Quotient der beiden Zahlen ist, sondern der natürliche Rest.
Beispiel:

5 mod 2 = **1**; da 5 : 2 = 2 R **1**

Aufgaben:

9 mod 7 = da 9 : 7 = R

6 mod 3 = da 6 : 3 = R

0 mod 1 = da 0 : 1 = R

Stellen wir uns vor, dass jemand dem Teilnehmer Peter Müller eine Nachricht verschlüsselt schicken möchte.
Dazu haben die Entwickler des Algorithmus eine äußerst simple Formel zu Chiffrierung gefunden. Man erhält den Geheimtext c, indem man die Ausgangsnachricht m (in einer Zahlenfolge – liegt bei der EDV ja sowieso vor) mit dem öffentlichen Schlüssel e des Empfängers potenziert und modulo n (der zweite öffentliche Schlüssel) reduziert. Einzige Bedingung ist, dass $m \leq n$ ist, man die Nachricht also gegebenenfalls in Teile kleiner oder gleich n aufteilen muss.
Also:

$$c = m^e \bmod n$$

Genauso simpel kann Peter Müller den Geheimtext c wieder entschlüsseln, nämlich indem er ihn mit seinem geheimen (privaten) Schlüssel d potenziert und modulo n reduziert:

$$m' = c^d \bmod n$$

Dass der von Herrn Müller erhaltene Text m' wirklich gleich der Ausgangsnachricht m sein muss, wird im folgenden noch gezeigt, wir wollen uns das Verfahren jedoch zuerst an einem Beispiel deutlich machen und werden feststellen, dass hier m' = m ist.
Der Absender möchte die geheime Nachricht: „HALLO“ an Peter Müller übermitteln.
Er wandelt den Text als erstes in eine Zahl um. Dazu kann man z.B. jeden Buchstaben entsprechend der Reihenfolge im Alphabet durchnummerieren.
Also: A=01, B=02, C=03, etc.
Damit ist H=08, A=01, L=12, und O=15.
Also ist

m = 08 01 12 12 15

oder einfach m = 801.121.215 . Die öffentlichen Schlüssel seien n = 851 und e = 5 und der geheime Schlüssel von Peter Müller d = 317 (zur Wahl der Schlüssel später). Da $m \leq n$ gelten muss, teilen wir die Nachricht in drei Teile m_1, m_2 und m_3.

$m_1 = 801$
$m_2 = 121$
$m_3 = 215$

Der Geheimtext c ist dann (in drei Teilen):

$c_1 = m_1^e \bmod n = 801^5 \bmod 851 = 816$
$c_2 = m_2^e \bmod n = 121^5 \bmod 851 = 692$
$c_3 = m_2^e \bmod n = 215^5 \bmod 851 = 361$

Diese Zahlen werden nun an Peter Müller öffentlich übermittelt.
Dieser versucht nun mit Hilfe seines geheimen Schlüssels d den Text zu entschlüsseln.

$m'_1 = c_1^d \bmod n = 816^{317} \bmod 851 = 801$
$m'_2 = c_2^d \bmod n = 692^{317} \bmod 851 = 121$
$m'_3 = c_3^d \bmod n = 361^{317} \bmod 851 = 215$

Er erhält also den Text m' = 801121215 oder einfach

m' = 08 01 12 12 15.

Da Peter Müller weiß, dass immer zwei Ziffern einem Buchstaben zugeordnet sind (entsprechend der Reihenfolge im Alphabet) kann er problemlos die entschlüsselte Zahl in die Zeichenfolge: „HALLO“ umwandeln. Die sichere Übermittlung hat funktioniert!

Aufgaben

1. Verschlüssele und entschlüssele zur Kontrolle wieder die Zeichenfolge „SOS“. Benutze zur Modulo-Division die Funktion MOD(m, n) von DERIVE. Wähle n = 851, e = 5 und d = 317.
2. Schreibe ein Java-Programm `rsa.java`, dass Zahlen bis 33 ver- und wieder entschlüsselt. Entwickle dazu eine rekursive Methode `long potenz(long bas, long exp)` welche die Potenz zweier Zahlen bildet. In Java lautet die Syntax für den Ausdruck m mod n: `m%n`. Wähle n = 33, e = 3 und d = 7.

Mathematischer Hintergrund des RSA-Verfahrens oder wieso ist m' = m?

Wieso ist aber der entschlüsselte Text immer gleich dem Ausgangstext? Dieser Frage wollen wir jetzt auf den Grund gehen und werden uns deshalb etwas in die höhere Mathematik hineinwagen. Dabei wollen wir auch klären, wie man die (öffentlichen und geheimen) Schlüssel zu wählen hat und warum.
Die Grundvoraussetzung für ein funktionierendes Verschlüsselungsverfahren ist, dass m' = m ist (d.h. der Empfänger muss den Ausgangstext nach der Entschlüsselung erhalten).
Also:

	$m' = m$	*einsetzen von $m' = c^d \bmod n$*
$\Leftrightarrow$	$c^d \bmod n = m$	*einsetzen von $c = m^e \bmod n$*
$\Leftrightarrow$	$(m^e \bmod n)^d \bmod n = m$	*nach der Potenzregel für die Modulo-Division*
$\Leftrightarrow$	$m^{e \cdot d} \bmod n = m$	(*)

Damit die Grundvoraussetzung m' = m für jede Nachricht gilt, müssen e, d und n so gewählt werden, dass obige Gleichung für jedes m erfüllt ist.
Um diese Zahlen zu finden ist ein Satz grundlegend, der auf den großen Schweizer Mathematiker Leonard Euler (1707-1783) zurückgeht. Er gilt (in unserer Fassung) für Zahlen n, die das Produkt zweier verschiedener Primzahlen p und q sind, also

$n = p \cdot q$ wobei $p \neq q$

Beispiele für solche Zahlen n sind 6 (= 2 · 3), 15 (= 3 · 5), oder wie oben 851 (= 23 · 37), nicht aber 17 (Primzahl), 105 (= 3 · 5 · 7), oder 49 (Quadratzahl, p = q).
Der Satz von Euler besagt:

Für jede natürliche Zahl m *mit* $m \leq n$ (= $p \cdot q$; $p \neq q$) *und jede natürliche Zahl* s *gilt*

$m^{s \cdot \varphi(n)+1} \bmod n = m$ mit $\varphi(n) = (p-1)(q-1)$ (auch Eulersche φ-Funktion genannt)

In Worten heißt das: Wenn man eine beliebige Zahl m mit der Zahl $s \cdot \varphi(n)+1$ (welche sich aus n ergibt) potenziert, dann erhält man bei der Division durch n als Rest wieder den Dividenden m.
Den Beweis des Satzes wollen wir hier außen vor lassen. Für interessierte Schüler ist er aber z.B. im Internet unter www.hh.schule.de/julius-leber-schule/melatob/SatzEuler.html nachzulesen.

Wenn also $e \cdot d = s \cdot \varphi(n) + 1$ ist, dann gilt nach dem Satz von Euler die Grundvoraussetzung m' = m. (siehe auch (*))
Also müssen e und d so gewählt werden, dass

$e \cdot d = s \cdot (p-1)(q-1) + 1$ für irgendeine natürliche Zahl s gilt.

Wenn man nun eine von $\varphi(n) = (p-1)(q-1)$ teilerfremde (d.h. der ggT ist 1) Zahl e hat, ist es leicht möglich eine Zahl d zu finden, sodass e und d die obige Bedingung erfüllen (p und q sind zunächst frei wählbare Primzahlen). Dazu benutzt man den so genannten *„erweiterten euklidischen Algorithmus"*.

Erweiterter euklidischer Algorithmus
Wir wollen uns diesen Algorithmus an einem Beispiel klarmachen.
Wir wählen p = 13 und q = 17. Dann ist $\varphi(n) = (p-1)(q-1) = (13-1)(17-1) = 192$.
Dazu suchen wir als erstes eine von $\varphi(13 \cdot 17) = 192$ teilerfremde Zahl e. Dieses kann durch ausprobieren geschehen. Möglich sind z.B. die Zahlen 5, 7, 11, 13, …, 31, 37, 41, 43, 47, … Wir wollen e = 41 wählen.
Der Algorithmus funktioniert nun so:

1.) Teile φ(n) ganzzahlig durch e (ganzzahlige Division ist die Division mit Rest – nicht die Modulo-Division!).
Also: $192 = 4 \cdot 41 + 28$ (I)

2.) Teile ganzzahlig den vorhergehenden Divisor durch den vorhergehenden Rest.
Also: $41 = 1 \cdot 28 + 13$ (II)

3.) Wiederhole Schritt 2.) solange bis der Rest 1 ist.
Also: $28 = 2 \cdot 13 + 2$ (III)
$13 = 6 \cdot 2 + 1$ (IV)

4.) Löse die letzte Gleichung (IV) durch Äquivalenzumformung nach der 1 auf.
Also: $1 = 13 - 6 \cdot 2$ (IV*)

5.) Der 2. Faktor des 2. Summanden entspricht dem Rest (2. Summanden) der vorhergehenden Gleichung (hier (III)). Löse die vorhergehende Gleichung nach dem Rest auf und setze ein.
Also: $1 = 13 - 6 \cdot (28 - 2 \cdot 13)$ (III) ⇒ (IV*)

6.) Multipliziere die Klammer aus und klammer den 1. Summanden aus (hier die 13). Fasse zusammen.
Also: $1 = 13 \cdot (1 + 6 \cdot 2) - 6 \cdot 28$
$= 13 \cdot 13 - 6 \cdot 28$

7.) Wiederhole Schritt 5.) und 6.) sooft, bis du die Gleichung (I) eingesetzt hast.
Also: $1 = 13 \cdot (41 - 1 \cdot 28) - 6 \cdot 28$ *(ausmultiplizieren der Klammer und ausklammern der 28)*
$= 13 \cdot 41 - (13 + 6) \cdot 28$ *(zusammenfassen)*
$= 13 \cdot 41 - 19 \cdot 28$ *(die 28 aus (I) einsetzen)*
$= 13 \cdot 41 - 19 \cdot (192 - 4 \cdot 41)$ *(ausmultiplizieren, ausklammern und zusammenfassen)*
$= 89 \cdot 41 - 19 \cdot 192$

Die gesuchte Zahl d ist der Faktor vor e, denn:

$$1 = 89 \cdot 41 - 19 \cdot 192$$
$$\Leftrightarrow \quad 89 \cdot 41 = 19 \cdot 192 + 1$$
$$\Leftrightarrow \quad 89 \cdot e = 19 \cdot \varphi(n) + 1$$
$$\Leftrightarrow \quad d \cdot e = s \cdot \varphi(n) + 1$$

Also ist für p = 13, q = 17 und e = 41 der geheime Schlüssel nach dem *erweiterten euklidischen Algorithmus* d = 89.

Wenn man e und d bei gegebenem p und q so wählt, dann ist die Grundvoraussetzung erfüllt, da jetzt gilt:

$$e \cdot d = s \cdot (p - 1)(q - 1) + 1$$

denn in der Tat gilt:

$$41 \cdot 89 = 19 \cdot (13 - 1)(17 - 1) + 1$$

also nach dem Satz von Euler für jedes m

$$m' = m^{ed} \bmod n = m,$$

Also stimmt der Ausgangstext immer mit der vom Adressaten dechiffrierten Nachricht überein.

Sicherheit des RSA-Verfahrens

Die Sicherheit des *RSA-Verfahrens* begründet sich in der Schwierigkeit große Zahlen zu faktorisieren, d.h. in ihre Primfaktoren zu zerlegen. Bislang ist die Faktorisierung der Zahl n (das würde die Kenntnis von p und q bedeuten, d.h. man kann den geheimen Schlüssel d, der für die Dechiffrierung notwendig ist bestimmen) die einzige bekannte Möglichkeit den *RSA-Algorithmus* zu knacken (der Beweis dafür ist nicht erbracht!). Diese Möglichkeiten heißen auch *trapdoor* („Geheimtür“).
Wenn man jedoch eine genügend große Zahl n wählt ist es praktisch nicht möglich die geheime Nachricht zu knacken, da eine Faktorisierung dieser Zahl (Zerlegung in p und q) viel zu zeit- und arbeitsintensiv wäre. Um eine Zahl mit 155 Dezimalstellen (512 Bits) zu faktorisieren brauchte man beispielsweise 1999 noch 7 Monate, wobei man 300 PCs parallel laufen ließ. Man spricht deswegen bei so einem Verschlüsselungsverfahren von einer *„praktischen Sicherheit“*. Die 100%ige Sicherheit ist hierbei nicht bewiesen, aber der Algorithmus ist mehrfach gut untersucht worden und ist dennoch nicht geknackt.
Da die Rechenleistung heutiger Computer rasant ansteigt, müssen immer größere Zahlen n gewählt werden, um optimale Sicherheit zu gewährleisten. Heutzutage werden schon *RSA-Verschlüsselungen* mit einer Länge von n von mindestens 512 Bits empfohlen (das entspricht einer Zahl mit 155 Dezimalstellen!). Um weiterhin mit Hilfe des *RSA-Algorithmus* sicher chiffrieren zu können, wird also immer mehr Rechenleistung und Arbeitsaufwand des Computers gefordert. Man darf also nicht davon ausgehen, dass wir eine Beschleunigung bei der Verschlüsselung mit *RSA* in Zukunft erfahren werden.

Aufgaben

1. Ermittle nach dem *erweiterten euklidischen Algorithmus* den geheimen Schlüssel d für p = 11, q = 13 und die teilerfremde Zahl e = 31. Kontrolliere dein Ergebnis mit geeigneter Rechnung.
2. Bestimme geeignete Schlüssel n, e und d ohne Verwendung des *erweiterten euklidischen Algorithmus* für p = 5 und q = 7.
3. Damit ein Angreifer eine geheime Nachricht entschlüsseln kann, muss dieser lediglich den geheimen Schlüssel d kennen. Es werden nacheinander die *RSA-verschlüsselten* Zahlen 45, 54, 45 gesendet. Entschlüssele durch faktorisieren unter Kenntnis der öffentlichen Schlüssel n = 65 und e = 5 mit Hilfe von DERIVE die verschlüsselte Zeichenfolge (Buchstaben wurden durchnummeriert). Beachte, dass bei der Erzeugung der Schlüssel der *erweiterte euklidische Algorithmus* benutzt wurde. Kontrolliere durch erneute Verschlüsselung.
   Bewerte unter Einbeziehung deines Ergebnisses die Sicherheit der *RSA-Verschlüsselung* bei relativ kleinem n.