Praktikum: "Mathematik am Computer" HS 2023
Prof. H. Harbrecht
M. Fallahpour, E. Gajendran, G. Ohlmann, M. Ramm Universität Basel

---

Serie 9

Matlab - RSA

zur 47. KW (20.11. – 24.11.2023)

Heute beschäftigen wir uns mit dem

RSA-Public-Key-Kryptosystem,
Rivest, **S**hamir und **A**dleman (MIT, 1977).

RSA ist ein so genanntes asymmetrisches Verschlüsselungsverfahren. Es werden Funktionen ausgenutzt, bei denen eine Richtung leicht, aber die andere Richtung sehr schwer zu berechnen ist. Diese werden auch Einwegfunktionen genannt. In unserem Fall beschreibt die Einwegfunktion die Faktorisierung einer grossen Zahl, also die Zerlegung dieser Zahl in ihre Primfaktoren.
Die Zahlen e, d (Verschlüsselungs- und Entschlüsselungsexponent) und M (RSA-Modul) werden erzeugt. Dabei bildet das Paar (e, M) den öffentlichen Schlüssel (*public key*) und (d, M) den privaten Schlüssel (*private/secret key*). Es wird folgendermassen vorgegangen:

I. Generiere den öffentlichen und den privaten Schlüssel.

1. Wähle drei beliebige Zahlen a, b und c.
2. Definiere p bzw. q als die auf a bzw. b folgenden Primzahlen.
3. Berechne den RSA-Modul $M = p \cdot q$.
4. Berechne die Eulersche ϕ-Funktion von M, also $\phi(M) = (p-1) \cdot (q-1)$.
5. Sind $\phi(M)$ und c nicht teilerfremd (also $\mathrm{ggT}(c, \phi(M)) \neq 1$), vergrössere c *solange* um 1, *bis* $\mathrm{ggT}(c, \phi(M)) = 1$. Setze dann den *public key* $e = c$.
6. Finde einen *private key* d. Dabei muss d erfüllen, dass $e \cdot d \equiv 1 \bmod \phi(M)$.

II. Verschlüsselung einer Nachricht.

1. Wenn jemand eine Nachricht x verschlüsseln will, die nur du entschlüsseln können sollst, übergib ihm deine *public keys* M und e (den *private key* d darf nur diejenige Person kennen, die die Schlüssel generiert hat).
2. Die verschlüsselte Nachricht y ist dann $y = x^e \bmod M$.

III. Entschlüsselung der Nachricht.

1. Erhältst du eine mit deinen *public keys* verschlüsselte Nachricht y, benötigst du nur noch deinen *private key* d.
2. Die entschlüsselte Nachricht ist dann $x = y^d \bmod M$.

Bemerkung: Dass die Entschlüsselung dann tatsächlich wieder den Klartext liefert, lässt sich mit der folgenden Rechnung einsehen:

$$\begin{aligned} y^d \bmod M &= (x^e)^d \bmod M \quad = \quad x^{ed} \bmod M \\ &= x^{\phi(M)k+1} \bmod M \quad (ed \equiv 1 \bmod \phi(M),\ k \text{ passend gewählt}) \\ &= (x^{\phi(M)})^k \cdot x \bmod M \\ &= 1^k \cdot x \bmod M \quad (x^{\phi(M)} \equiv 1 \bmod M \text{ (Satz von Euler)}) \\ &= x \bmod M \\ &= x, \qquad \text{wenn } x \in \{1, \ldots, M-1\}. \end{aligned}$$

Aufgabe 9.1 (1+3+3+3 Punkte):

a) Finde heraus was die Matlab-Befehle `nextprimeAlt` (auf der Praktikumswebseite verfügbar), `mod` und `gcd` tun, wie sie richtig benutzt werden und teste sie dann anhand eines Beispiels. Siehe dazu z.B. `doc mod` und vor allem `doc gcd`. Was liefert der Befehl `[G,U,V] = gcd(A,B)`?

b) Schreibe eine Funktion `[M,e,d] = generate_key(a,b,c)`, die (gemäss Teil I. oben) drei beliebige Zahlen a, b und c als Input nimmt und daraus die Schlüssel M, e und d des RSA-Verfahrens generiert. Teste deine Funktion für $a = 18$, $b = 12$, $c = 2020$ (Ergebnis: $M = 247$, $e = 2021$, $d = 101$).
Hinweis: Um im letzten Schritt d zu bestimmen, verwende, dass $e \cdot d + \phi(M) \cdot k = \text{ggT}(e, \phi(M))$, für ein passendes $k \in \mathbb{Z}$, ist (siehe `doc gcd`). Beachte ausserdem, dass $d \geq 0$ sein soll. Falls $d < 0$ ist, setze $d = (\phi(M) + d) \bmod \phi(M)$.

c) Um den Ausdruck $x^y \bmod n$ zu berechnen, wird Matlab zuerst x^y auswerten, was bei sehr grossen Zahlen, wie wir sie hier verwenden werden, zu Ungenauigkeiten führt. Um dies zu vermeiden, verwenden wir, dass

$$ab \bmod n = (a \bmod n)(b \bmod n) \bmod n$$

gilt. Daraus folgt, dass

$$x^y \bmod n = \underbrace{(\ldots(((x \bmod n)x \bmod n)x \bmod n)\ldots)x \bmod n}_{y\text{-mal mod anwenden}}.$$

Schreibe eine Funktion `z = my_modulo(x,y,n)`, die $x^y \bmod n$ für grosse Zahlen exakt berechnet. Benutze dazu eine `for`-Schleife. Beachte, dass das Eingabeargument `x` auch ein Vektor sein kann! Teste deine Funktion für $x = [10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8]$, $y = 5$, $n = 19$ (Ergebnis: $z = (3, 9, 8, 5, 15, 7, 2, 6)$) und vergleiche das mit dem Ergebnis vom Befehl `mod((10.^(1:8)).^5,19)`, das Rundungsfehler beinhaltet.

d) Schreibe zwei Funktionen `y = encrypt(t,e,M)` und `t = decrypt(y, d, M)`, die eine Nachricht x wie oben angegeben (Teile II. und III.) ver- bzw. entschlüsseln können. Dabei musst du `double` verwenden, um einen Text t in einen Vektor x umzuformen und `char`, um einen Vektor x in einen Text t umzuformen. Zum Beispiel ist `double('Mathe') = [77,97,116,104, 101]` und `char([80,114,97,107,116,105,107,117,109]) = 'Praktikum'`. Verwende ausserdem deine Funktion `my_modulo`, um Ausdrücke der Form $x^y \bmod M$ zu berechnen. Verschlüssle und entschlüssle selber ein Beispiel.

Um dein Programm abschliessend zu testen, erzeuge mit `generate_key` eigene Schlüssel, lass deinen Studikollegen/Studikollegin eine Nachricht mit deinen *public keys* M und e verschlüsseln und versuche, sie mit deinem *private key* zu entschlüsseln!

Allgemeine Informationen zum Praktikum befinden sich auf der Webseite
`http://cm.dmi.unibas.ch/teaching/praktikumI/praktikumI.html`