Stichworte:
Public-Key-Kryptographie, geheime und öffentliche Schlüssel,
RSA, Kodierung von Textnachrichten,
Diskreter Logarithmus-Problem (DL-Problem),
Diffie-Hellman-Schlüsselaustausch,
Diffie-Hellman-Problem (DH-Problem),
Bsp. für Man-in-the-Middle-Attacke

---

§1.2 Public-Key-Kryptographie

Public-Key-Kryptographie bezeichnet man auch als asymmetrische Kryptographie. Bei diesem Kommunikationsverfahren hat jeder Nutzer einen öffentlichen Schlüssel, den jeder einsehen kann, und einen privaten Schlüssel, den jeder Nutzer geheim hält: Jeder kann verschlüsseln, aber nur der rechtmäßige Empfänger entschlüsseln. Möchte Nutzer (B) eine Nachricht an Nutzer (A) sendet, benutzt er zur Verschlüsselung den öffentlichen Schlüssel von (A), die Entschlüsselung gelingt aber nur (A) mit dem privaten Schlüssel.
Kerkhoffs Prinzip: Die Sicherheit eines Kryptosystems darf nicht von der Geheimhaltung des Algorithmus abhängen, sondern von der Geheimhaltung der geheimen Schlüssel.

Ein solches Szenario (auch "Protokoll" genannt) ist das RSA-Verfahren, das wir in 1.2.1 behandeln. Die Verfahren in 1.2.2 und 1.2.3 sind Kryptographie-Verfahren, die mit allgemeinen Gruppen machbar sind.

1.2.1 RSA-Verfahren

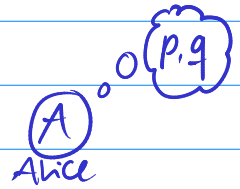
(RSA arbeitet mit $(\mathbb{Z}_m^*, \cdot)$)

Das RSA-Verfahren ist benannt nach einer Arbeit von R.L. Rivest, A. Shamir und L.M. Adleman aus dem Jahr 1978. Seine Sicherheit beruht auf der Schwierigkeit des Faktorisierungsproblems und wird bis heute zur sicheren Kommunikation benutzt.
Die Methode verlangt auch die Möglichkeit, große Primzahlen zu erzeugen, die möglichst zufällig gewählt sein sollen, ähnlich wie beim Münzwurfproblem (V4_16).
$n = p \cdot q$ muss so groß sein, dass alle bekannten Faktorisierungsverfahren zu langsam wären.

1.) Wir beschreiben den Verlauf des Verfahrens:

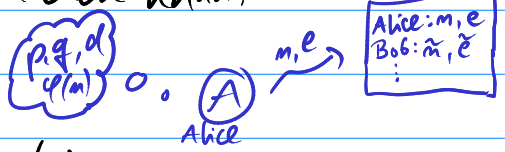
Die beiden Protagonisten heißen wieder Nutzer (A)lice und (B)ob.
Sie kommunizieren über einen unsicheren Kanal miteinander.

Schritt (1.) (Vorbereitung)

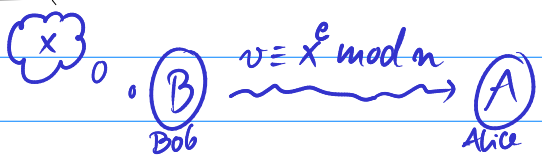


Jeder Nutzer, z.B. (A), wählt zwei große Primzahlen $p \neq q$, etwa gleichgroß mit ähnlicher Stellenanzahl, und berechnet $n = pq$ sowie $\varphi(n) = (p-1)(q-1)$.

Dann wählt (A) eine Zahl e mit $1 < e < \varphi(n)$ und berechnet $1 < d < \varphi(n)$ als Inverses von e mod $\varphi(n)$, d.h. $de \equiv 1 \bmod \varphi(n)$, unter Zuhilfenahme des euklidischen Algorithmus.
(A) hält $p, q, \varphi(n), d$ geheim und gibt n, e bekannt, z.B. durch Hinterlegung auf einen öffentlichen Schlüsselserver, wo jeder nachsehen kann.

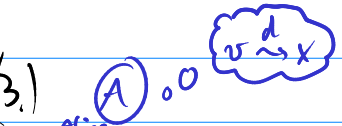


Schritt (2.)



Bob möchte Alice seinen Geheimtext (als eine Zahl x kodiert) schicken. Er besorgt sich die Daten n, e vom Server und verschlüsselt x zu $x^e \bmod n$.
Dann schickt er ihr das Ergebnis $v \equiv x^e (n)$ zwischen 1 und n.

Schritt (3.)



Alice entschlüsselt den geheimen Text v durch Berechnen von $v^d \bmod n$, sie erhält x, weil für ein $k \in \mathbb{Z}$ gilt: $ed = 1 + k \cdot \varphi(n)$, also folgt

$$v^d \equiv (x^e)^d \equiv x^{1 + k \cdot \varphi(n)} \equiv x \cdot \underbrace{\left(x^{\varphi(n)}\right)^k}_{\equiv 1 \,(n) \text{ nach Euler-Fermat, falls } ggT(x,n) = 1} \equiv x \quad (n).$$

2) **Bem.:** Die nötigen Berechnungen sind: schnelles modulares Potenzieren mod n, d.h. Berechnungen in der multiplikativen Gruppe $(\mathbb{Z}_n^*, \cdot)$,
Berechnen von d mit dem euklidischen Algorithmus, Erzeugen großer Pzen p, q.

3.) Bem.: Ein Unbefugter, der die Daten n, e, v dieser Kommunikation abfängt, ist nicht in der Lage, x ohne der Kenntnis von d, p, q, $\varphi(n)$ zu berechnen. Dazu müsste man n faktorisieren.

4.) Bem.: Wie sicher das Verfahren ist, hängt davon ab, wie groß die verwendeten Schlüssel sind. Aktuell ist eine Verschlüsselung, bei der p, q eine Bitlänge von mindestens 512 haben sollten; besonders sicher: 2048 Bit. Empfehlung der Bundesnetzagentur bis Ende 2020: mind. 1976 Bit. Gegen einen Angriff mit dem Quantencomputer hätte man allerdings keine Chance.

5.) Bem.: Auch in den seltenen Fällen p|x oder q|x, d.h. ggT(x, n) > 1, arbeitet das Verfahren korrekt (ohne Beweis).

6.) Bem.: Das Verfahren kann auch ohne Schlüsselserver benutzt werden. (B) kann (A) erst mitteilen, dass er ihr eine Nachricht schicken will. Dann erst erledigt (A) Schritt (1.) und teilt ihm die Daten n, e mit. Der Rest geht dann wie oben.

7.) Zur "Geschichte" von RSA: RSA wurde 1983 als Patent angemeldet, welches 2000 erlosch. Bis Ende der 90er Jahre verbot die US-Regierung Firmen, Software mit starker Verschlüsselung zu exportieren (z.B. T-Shirts mit aufgedruckter RSA-Anleitung...).

s. wikipedia "Crypto Wars"

Weiter sollten per Gesetzesvorlage Anbieter elektronischer Kommunikationsdienste dazu verpflichtet werden, Behörden die Möglichkeit zum Zugriff zu verschaffen; das Gesetz scheiterte am Widerstand von Industrie und Bürgerrechtlern. Es motivierte Phil Zimmermann dazu, den Standard PGP (= pretty good privacy) zu entwickeln, mit dem bis heute E-mails und anderes für jedermann sicher verschlüsselt werden können (speziell mit RSA; öffentliche Schlüsselserver dafür gibt es im Internet, z.B. auf pgp.mit.edu). Zimmermann stellte sein Programm 1991 kostenlos zur Verfügung. Es wurde ein Verfahren gegen ihn eröffnet, das sich über 3 Jahre lang hinzog. Vorwurf: er exportiere Verschlüsselungstechnologie, die wie Waffentechnologie einzustufen sei). Der Fall wurde fallengelassen, heute ist die Benutzung und Export in den USA straffrei. Bis heute zählt pgp als sicherste und empfehlenswerteste Verschlüsselung privater Kommunikation.

8.) Kodierung von Textnachrichten: Wir beschreiben hier ein Verfahren, das die Machbarkeit der Kodierung Text → Zahl demonstrieren soll. Wenn man es so anwenden möchte, sind aber größere Blöcke erforderlich, damit nicht durch Häufigkeitsanalysen der Blöcke Rückschlüsse auf die Geheimnachricht möglich werden.
Die Buchstaben A,...,Z des Alphabets werden mit 0,...,25 identifiziert, das Leerzeichen mit 26. Klartexte werden zu Blöcken aus je drei Zahlen zusammengefasst, also z.B. KLARTEXT_ → 10,11,0 / 17,19,4 / 23,19,26
Jedem Block x_1, x_2, x_3 ordnen wir die Zahl $x = x_1 \cdot 27^2 + x_2 \cdot 27 + x_3$ (im 27er System) zu, also: KLARTEXT_ → 7587 / 12910 / 17306,
welche beim RSA-Verfahren gemäß $x^e \equiv v \ (n)$ verschlüsselt wird.
Jeder Wert v wird im 29er-System umgewandelt gemäß
$v = v_1 \cdot 29^2 + v_2 \cdot 29 + v_3$ zu einem Block $v_1, v_2, v_3 \in \{0,\dots,28\}$,
der wieder als Text geschrieben werden kann (mit zusätzlichen Zeichen für 27 und 28, z.B. "." = 27, "," = 28).
Ist n zwischen 27^3 und 29^3, werden Ver- und Entschlüsselung eindeutig (ohne Beweis) → für größere n werden größere Blöcke nötig!

1.2.2 Diffie-Hellman-Verfahren

9.) Das Problem des diskreten Logarithmus (DL-Problem):
Geg. sei eine abelsche Gruppe, wir beschreiben das Problem multiplikativ und additiv:

In $(G,\cdot,1)$:	In $(G,+,0)$:
Sei $x \in G$, $n = \mathrm{ord}(x)$, $y \in \langle x\rangle = \{x^l ; l \in \mathbb{Z}\}$.	Sei $x \in G$, $n = \mathrm{ord}(x)$, $y \in \langle x\rangle = \{l \cdot x ; l \in \mathbb{Z}\}$.
Bestimme $k \bmod n$ mit $y = x^k$.	Bestimme $k \bmod n$ mit $y = k \cdot x$.
("diskreter Logarithmus")	("diskreter Logarithmus")

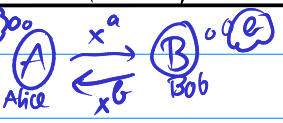
10.) Ist eine Gruppe G gegeben, in der das DL-Problem schwer ist, kann dies für ein Kryptoverfahren genutzt werden.

- Im Fall $G = (\mathbb{Z}_m^*, \cdot, 1)$ ist das DL-Problem ähnlich schwer wie das Faktorisierungsproblem. Auch dafür konnte P. Shor 1994 zeigen, dass es auf einem Quantencomputer schnell lösbar ist.
- Im Fall, dass $G = (E(\mathbb{K}), +, \mathcal{O})$ die Gruppe einer (kryptographisch) geeigneten elliptischen Kurve ist, ist das DL-Verfahren quasi unlösbar. Die besten bekannten Algorithmen sind langsamer als die für das DL-Problem für $\mathbb{Z}_m^*$. Darauf beruht die als höher angesehene Sicherheit bei der Kryptographie mit elliptischen Kurven. Algorithmen auf Quantencomputern, die das DL-Problem für elliptische Kurven schnell lösen könnten, sind derzeit unbekannt.

11.) **Der Diffie-Hellman-Schlüsselaustausch**

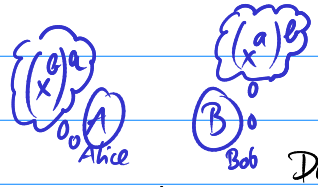
Hier vereinbaren Alice und Bob durch einen öffentlichen Kanal einen gemeinsamen geheimen Schlüssel, die sie dann für ein symmetrisches Kryptoverfahren nutzen können. Geg. sei eine Gruppe G und $x \in G$, sowie $n \in \mathbb{N}$. Diese Daten seien öffentlich bekannt.

In $(G, \cdot, 1)$:

(1.) 

Alice denkt sich eine Zahl $a \in \{1, \dots, n-1\}$ und schickt $x^a \in G$ an Bob.

Bob denkt sich eine Zahl $b \in \{1, \dots, n-1\}$ und schickt $x^b \in G$ an Alice.

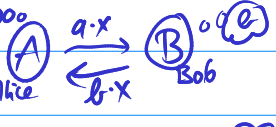
(2.) 

Alice berechnet mit a das Gruppenelement $(x^b)^a$

Bob berechnet mit b das Gruppenelement $(x^a)^b$

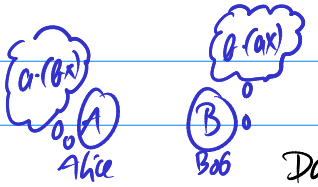
Danach besitzen beide den gemeinsamen geheimen Schlüssel $(x^b)^a = x^{ab} = x^{ba}$.

In $(G, +, 0)$:

(1.) 

Alice denkt sich eine Zahl $a \in \{1, \dots, n-1\}$ und schickt $a \cdot x \in G$ an Bob.

Bob denkt sich eine Zahl $b \in \{1, \dots, n-1\}$ und schickt $b \cdot x \in G$ an Alice.

(2.) 

Alice berechnet mit a das Gruppenelement $a \cdot (bx)$

Bob berechnet mit b das Gruppenelement $b \cdot (ax)$

Danach besitzen beide den gemeinsamen geheimen Schlüssel $a \cdot (bx) = abx = b \cdot (ax)$

12.) Ein Unbefugter, der die Daten x^a, x^b bzw. ax, bx abhört, kann die geheimen Schlüssel berechnen, wenn er das DL-Problem lösen kann. Es genügt aber schon, dafür das folgende, ev. leichtere Problem zu lösen:

Diffie-Hellman-Problem (DH-Problem):
Berechne zu $x^a, x^b \in \langle x \rangle \subseteq G$ in $(G, \cdot, 1)$ das Element $x^{ab} \in \langle x \rangle$.
Es ist aber davon auszugehen, dass auch DH ein schweres Problem ist.
(Bem.: DL lösbar $\Rightarrow$ DH lösbar ist klar, „$\in^a$" ist unbekannt.)

13.) Weiter ist beim Schlüsselaustausch entscheidend, dass sich Alice und Bob sicher sein können, wirklich mit dem angegebenen Absender zu kommunizieren:
Ein Unbefugter könnte versuchen, sich erst als Alice auszugeben, und so mit Bob einen Schlüssel x^{eb} auszutauschen und dies ebenso mit Alice tun. $\vec{x^{ea}}$
Gelingt dies, braucht der Unbefugte die verschlüsselten Nachrichten zwischen Alice und Bob abzufangen:
Die Nachrichten von Alice an Bob dekodiert er mit dem Alice-Schlüssel x^{ea} schickt sie mit dem Bob-Schlüssel x^{eb} kodiert an Bob weiter, und umgekehrt.
Er kann so die gesamte geheime Kommunikation abhören.
Man nennt dies eine "Man-in-the-middle-Attacke."