4 RSA und PGP

Im Juni 1991 wurde das Programm „PGP“ (für ‚pretty good privacy‘) von Phil Zimmermann ins Internet gestellt. Es ermöglichte jedermann, e-Mails derart gut zu verschlüsseln, dass nicht einmal die NSA (‚national security agency‘ der USA) die Botschaft entschlüsseln konnte. Zudem war auch das Problem des Schüssel-Austauschs gelöst, indem die Verschlüsselung mit einem **öffentlichen** Schlüssel des Empfängers erfolgt, aber nur der Empfänger mit seinem **privaten** Schlüsselteil die Botschaft entschlüsseln kann.

Phil Zimmermann wurde deswegen mit dem Vorwurf des „Waffenexports“ vor die US-Justiz gezogen. Das Verfahren wurde schliesslich eingestellt – das Programm war nun einmal freigesetzt und ermöglichte sowohl Oppositionellen in Diktaturen als auch kriminellen Gruppen den abhörsicheren Mailaustausch über öffentliche, unsichere Kanäle.

Inzwischen ist aus dem Programm PGP ein kommerzielles Paket geworden. Phil Zimmermann hat aber darauf bestanden, dass immer auch eine voll funktionsfähige Gratisversion angeboten werden muss. Diese müssen Sie in den folgenden Wochen auf ihrer Maschine installieren.

Das Programm „PGP“ benützt an zentraler Stelle die Verschlüsselung nach RSA. Wir werden zuerst anhand eines Beispiels mit kleinen Primzahlen sehen, wie RSA funktioniert. Dann werden wir die Korrektheit des Verfahrens in allen Details beweisen und brauchen dazu unsere ganzen Kenntnisse der Gruppe $(\mathbb{Z}_n^*, \cdot)$. Schliesslich werden Sie selber noch ein Beispiel mit etwas grösseren Primzahlen beisteuern.

Buchempfehlung: „Geheime Botschaften“ von Simon Singh, dtv 2011 (10. Auflage!)

Die Mathematik von RSA an einem Beispiel

Wir zeigen die einzeln Schritte, die für die Verschlüsselung nach RSA (Ron **R**ivest, Adi **S**hamir und Leonard **A**dleman 1977) durchgeführt werden müssen – mit sehr kleinen Zahlen.

1. Alice wählt zwei (riesige ...) Primzahlen – wir nehmen $p = 17$ und $q = 11$. Diese beiden Zahlen hält sie geheim.
2. Alice bildet die Produkte $n = p \cdot q = 17 \cdot 11 = 187$ und $m = (p-1) \cdot (q-1) = 16 \cdot 10 = 160$.
3. Alice wählt eine weitere Zahl e, welche teilerfremd ist mit m. Sie nimmt z. B. $e = 7$.
4. Alice veröffentlicht die beiden Zahlen n und e in einem Verzeichnis als ihren ‚public key‘. Das ist nur möglich, wenn n noch nicht belegt ist von jemand anderem. Aber es gibt ja reichlich Primzahlen ...
5. Bob will nun eine Botschaft an Alice verschlüsseln. Diese bestehe der Einfachheit halber aus einer einzigen Zahl b, z. B. $b = 88$.
6. Die Verschlüsselung besteht in der Berechnung von $c = (b^e) \mod n$, also $c = (88^7) \mod 187$.
7. Für derart kleine Zahlen ist die Berechnung von c kein Problem: Unser TR liefert uns sofort
$$\text{c} = \text{mod}(88\text{^}7,187) = 11.$$
8. Bob schickt nun die codierte Nachricht $c = 11$ über einen öffentlichen, unsicheren Kanal an Alice.

9. Weil e teilerfremd ist zu $m = (p-1)\cdot(q-1)$, existiert ein multiplikatives Inverses d mit $e \cdot d = 1 \mod m$. Unser Rechner liefert sofort

$$\text{d = euklidin(7,160) = 23.}$$

10. Diese Zahl d kann nur Alice leicht berechnen, denn nur sie kennt die beiden Primzahlen p und q. Die Zahl d ist ihr *privater Schlüssel*.

11. Alice empfängt die codierte Botschaft $c = 11$ von Bob und entschlüsselt sie mit $b = (c^d) \mod n$, also

$$\text{b = mod(11\^{}23,187) = 88.}$$

Nun können Sie einwenden, dass ja jeder die Zahl $n = 187$ sofort in die zwei ‚geheimen' Primzahlen p und q zerlegen kann und damit ebenfalls den ‚privaten' Schlüssel d berechnen kann, der zum Dechiffrieren benötigt wird. Sie haben bei $n = 187$ natürlich recht – die Sache sieht aber sofort anders aus, wenn die bewussten Primzahlen p und q einige hundert Stellen haben: Es exstiert bis heute kein effizienter Algorithmus, um riesige Zahlen in riesige Faktoren zu zerlegen. Jeder PC kann heute das Produkt von zwei tausendstelligen Zahlen in einer Sekunde berechnen, die Zerlegung dieses Produkts in die beiden Faktoren dauert aber immer noch Millionen von Jahren! Genau darauf beruht die Sicherheit der Verschlüsselung nach RSA. Findet jemand einen schnellen Faktorisierungsalgorithmus für grosse Zahlen, so ist RSA als Verschlüsselungsmethode sofort unbrauchbar.

Wir beweisen nun allgemein, dass die Verschlüsselung und Entschlüsselung nach RSA für alle Botschaften funktioniert.

1. Wähle (sehr grosse) Primzahlen p und q mit $p \neq q$. Es gibt ja unendlich viele davon.

2. Setze $n := p \cdot q$ und rechne fortan in $\mathbb{Z}_n$. Dies ist **kein** Körper, p und q sind ja ein Paar von Nullteilern.

3. Wir betrachten nun die multiplikative Gruppe $\mathbb{Z}_n^*$, also die Menge aller Elemente in $\mathbb{Z}_n$, die ein multiplikatives Inverses haben.

4. $n = p \cdot q$ hat nur die Teiler 1, p, q und n. Also sind nur die Vielfachen von p und q nicht teilerfremd mit n:

$$p,\ 2\cdot p,\ 3\cdot p,\ \ldots,\ (q-1)\cdot p \quad \text{und} \quad q,\ 2\cdot q,\ 3\cdot q,\ \ldots,\ (p-1)\cdot q$$

Nur diese Zahlen sowie die 0 haben kein multiplikatives Inverses in $\mathbb{Z}_n$. Es ist daher

$$\text{ord}(\mathbb{Z}_n^*) = n-(p-1)-(q-1)-1, \text{ also } \text{ord}(\mathbb{Z}_n^*) = p\cdot q - p - q + 1 = (p-1)\cdot(q-1).$$

5. Es ist also $\text{ord}(\mathbb{Z}_n^*) = (p-1)\cdot(q-1)$, was immer noch eine sehr grosse Zahl ist. Wir setzen $m := (p-1)\cdot(q-1)$. Nach Korollar 3.14 gilt für alle $a \in \mathbb{Z}_n^*$ $a^m = 1$ modulo n.

6. Wähle noch $e \in \mathbb{Z}_n^*$ so, dass e und m teilerfremd sind ($m-1$ ist zum Beispiel eine solche Zahl). e hat dann in $\mathbb{Z}_m^*$ ein Inverses d, welches wir z. B. mit dem erweiterten Euklidischen Algorithmus berechnen können. Es ist also $e \cdot d = 1$ modulo m.

7. Wegen $\text{ord}(\mathbb{Z}_n^*) = m$ folgt für alle Elemente von $\mathbb{Z}_n^*$,

$$a^{e\cdot d} = a^{k\cdot m+1} = (a^m)^k \cdot a^1 = 1^k \cdot a = a \text{ modulo } n.$$

Erstaunlicherweise gilt diese Gleichung aber für **alle** Elemente in $\mathbb{Z}_n$ (und nicht nur für diejenigen in $\mathbb{Z}_n^*$), was wir in **8.** und **9.** beweisen werden.

8. Aus $e \cdot d = k \cdot m + 1 = k \cdot (p-1) \cdot (q-1) + 1$ folgt sofort $e \cdot d = 1$ modulo $p-1$ und $e \cdot d = 1$ modulo $q-1$. Nun rechnen wir modulo p für den Rest von **8.**:

- Ist a mod $p = 0$, so gilt sowieso $a^{e \cdot d} = 0^{e \cdot d} = 0$
- Ist a mod $p \neq 0$, so ist a ein Element von $\mathbb{Z}_p^*$. Es ist aber $\text{ord}(\mathbb{Z}_p^*) = p-1$, und daher (wieder nach Korollar 3.14) $a^{e \cdot d} = a^{h \cdot (p-1)+1} = (a^{p-1})^h \cdot a = 1^h \cdot a = a$.

In allen Fällen gilt somit $a^{e \cdot d} = a$ modulo p und genauso $a^{e \cdot d} = a$ modulo q.

9. Wir sollten zeigen, dass für **alle** $a \in \mathbb{Z}_n$ gilt $a^{e \cdot d} = a$ modulo n. Aus **8.** wissen wir: Es gibt Zahlen r und s in $\mathbb{N}$, sodass gilt $a^{e \cdot d} = r \cdot p + a$ sowie $a^{e \cdot d} = s \cdot q + a$. Für r und s muss also gelten $r \cdot p = s \cdot q$. Das sind jetzt alles Rechnungen für natürliche Zahlen, wir rechnen im Moment nicht modulo einer bestimmten Zahl!

- Ist $r \cdot p = 0 = s \cdot q$, dann gilt $a^{e \cdot d} = a$ und damit gilt auch $a^{e \cdot d} = a$ modulo n. Dieser triviale Fall gilt nur für $a = 0$ und $a = 1 \ldots$
- Sei nun $r \cdot p = s \cdot q$, mit $r \neq 0$ und $s \neq 0$. Dann muss p ein Primfaktor von s sein, und wir haben $a^{e \cdot d} = t \cdot p \cdot q + a = t \cdot n + a$, und es folgt $a^{e \cdot d} = a$ modulo n.

10. Mit unseren Werten von p, q, n, e und d gilt somit für alle $a \in \mathbb{Z}_n$ $a^{e \cdot d} = a$ modulo n. Nun publizieren wir n und e als **öffentlichen Schlüssel** und hüten d als unseren **privaten Schlüssel**.

11. Wir lassen Botschaften b, die an uns gerichtet sind, verschlüsseln mit

$$c := (b^e \text{ modulo } n).$$

12. Erhalten wir eine codierte Botschaft c, so entschlüsseln wir mit

$$b := (c^d \text{ modulo } n) = b^{e \cdot d} \text{ modulo } n.$$

Für alle Werte von b gilt ja $b^{e \cdot d} = b$. □

Wir betonen nochmals, dass die ganze Sicherheit des Verfahrens darauf beruht, dass die Zerlegung der riesigen Zahl n in ihre beiden Primfaktoren p und q auch mit geballter Rechenleistung Millionen von Jahren dauert. Daher kann ein Feind den privaten Schlüssel d nicht innert nützlicher Frist aus n und e berechnen.

Aufgaben:

1. Zeigen Sie, wie man z. B. 69^{89} modulo 551 effizient berechnen kann, indem man 69^2, 69^4, 69^8, 69^{16}, 69^{32} und 69^{64} berechnet (immer modulo 551).

2. Rechnen Sie ein eigenes Beispiel wie dasjenige von Alice und Bob durch, aber mit Primzahlen zwischen 50 und 99.

3. Wie **2.**, aber mit Primzahlen zwischen 200 und 400.

4. Schreiben Sie ein Programm potmod(a, b, n), welches fast sofort a^b modulo n berechnet auch für grosse Zahlen a, b und n .

Welchen Beitrag hat denn jetzt Phil Zimmermann gemacht?

Er hat all diese Ideen so in ein Programm gepackt, dass sie auch von Otto Normalverbraucher verwendet werden können! Die Erzeugung der grossen Primzahlen p und q, die Berechnung von n, e und d, das Verschlüsseln und Entschlüsseln geschieht alles im Hintergrund, schnell, effizient und ohne grosses Zutun des Benutzers.

Wir beschreiben die einzelnen Schritte, welche Sie selber später auch durchführen sollen!

1. Alice und Bob installieren beide die Trial- oder die Freeware-Version des Programms „PGP Desktop“ auf ihrem PC. Es existieren Versionen für Linux, Mac und Windows.
2. Sie erzeugen beide mit einer Serie von zufälligen Mausbewegungen zwei riesige Primzahlen: p und q bei Bob, p' und q' bei Alice.
3. Das Programm erzeugt automatisch die öffentlichen und die privaten Schlüssel von Alice und Bob:

   Bob's public key: n und e, Bob's private key: d
   Alice's public key: n' und e', Alice's private key: d'

   Es gilt

   $$n = p \cdot q \text{ und } e \cdot d = 1 \text{ modulo } (p-1)\cdot(q-1),$$
   $$n' = p' \cdot q' \text{ und } e' \cdot d' = 1 \text{ modulo } (p'-1)\cdot(q'-1).$$
4. Das Programm erlaubt es, die öffentlichen Schlüssel auf einem **keyserver** (z. B. bei ‚keyserver.pgp.com‘) zu deponieren. Diese öffentlichen Schlüssel gehören zu einem Benutzernamen und einer bestimmten Mailadresse.
5. Bob will nun Alice eine Botschaft b senden (bestehend z. B. aus einer eMail, also einer Reihe von Zeichencodes, die in Gruppen von 512 Zeichen zu **einer** riesigen Zahl zusammengefasst werden). Das Programm PGP erzeugt spontan einen Schlüssel s, der nur dieses Mal verwendet wird, um die Botschaft b in einem konventionellen, einfachen Verfahren (z. B. DES oder IDEA) in den Code c umzuformen.
6. Nun verschlüsselt Bob's Programm diesen Einmal-Schlüssel s per RSA zuerst mit dem privaten Schlüssel von Bob und dann noch mit dem öffentlichen Schlüssel von Alice:

   $$s \longrightarrow t := s^d \text{ modulo } n$$
   $$t \longrightarrow u := t^{e'} \text{ modulo } n'$$
7. Bob schickt nun c und u über einen beliebigen unsicheren Kanal (meist per eMail) an Alice.
8. Alice entschlüsselt mit ihrem PGP-Programm den doppelt verschlüsselten Einmalschlüssel u zuerst mit ihrem privaten Schlüssel und dann mit dem öffentliche Schlüssel von Bob:

   $$u \longrightarrow u^{d'} \text{ modulo } n' \text{ liefert } t, \text{ da } t^{e'\cdot d'} = t \text{ modulo } n'$$
   $$t \longrightarrow t^{e} \text{ modulo } n \text{ liefert } s, \text{ da } s^{d\cdot e} = s \text{ modulo } n$$

   Damit ist zusätzlich sichergestellt, dass die Botschaft auch wirklich von Bob stammt, denn nur Bob kann s mit seinem privaten Schlüssel verschlüsseln!
9. Dann entschlüsselt das Programm von Alice die Botschaft c mithilfe des Schlüssels s wieder nach einem schnellen, einfachen Verfahren wie DES oder IDEA und gibt die Originalbotschaft b aus.

Alle Einzelschritte laufen auf beiden Seiten vollautomatisch im Hintergrund, der Benutzer von „PGP“ braucht sich nicht darum zu kümmern. Es nützt niemanden etwas, wenn er die Botschaft (c, u) abfangen kann, da der private Schlüssel von Alice gebraucht wird, um aus u den Einmalschlüssel s herzustellen, der für die Dechiffrierung von c benötigt wird. Es kann sich auch niemand als Bob ausgeben und Alice eine falsche Mitteilung unterjubeln, da ja Bob's privater Schlüssel benötigt wird für den Schritt von s zu t.

Der Beitrag von Diffie und Hellmann

Schon 1976 haben Whitfield Diffie und Martin Hellmann gezeigt, dass es möglich ist, ganz öffentlich einen geheimen Schlüssel zu vereinbaren. Damit haben sie den Weg für das RSA-Trio vorbereitet, welches dann 1977 den Durchbruch geschafft hat.

Diffie und Hellmann haben den folgenden Vorschlag gemacht:

1. Man legt eine (grosse) Primzahl p fest und zudem eine weitere Zahl a mit $1 < a < p$. Sowohl a als auch p werden publiziert.
2. Alice und Bob wollen nun eine Schlüsselzahl s vereinbaren. Alice wählt eine beliebige Zahl e, behält diese geheim und sendet Bob a^e modulo p. Bob wählt ebenfalls eine Zahl d und sendet Alice a^d modulo p.
3. Alice berechnet nun $(a^d)^e = a^{d \cdot e}$ modulo p und Bob berechnet $(a^e)^d = a^{e \cdot d}$ modulo p. Beide erhalten dieselbe Zahl s, welche sie als (Einmal-)Schlüssel verwenden können.
4. Aus den Werten von p, a, a^e und a^d lassen sich die Werte von e und d und damit von s nur sehr schwer ermitteln, man muss praktisch alle Möglichkeiten durchprobieren. Das dauert bei grossen Primzahlen p **sehr** lange.

Wieder beruht die ‚Sicherheit‘ nur darauf, dass der Rechenaufwand für die Entschlüsselung sehr gross ist. Die Sicherheit ist aber recht komfortabel, wenn der Rechenaufwand in die Millionen von Jahren geht ...

Zum Schluss sei noch erwähnt, dass in England bereits 1975 und ganz unabhängig von den Entwicklungen in den USA die drei Geheimdienstler James Ellis, Clifford Cock und Malcolm Williams das RSA-Verfahren und die Methode des Austauschs eines Einmal-Schlüssels entdeckt haben. Ihre Arbeiten standen aber unter strenger militärischer Geheimhaltung. Erst nach der Publikation von PGP wurden sie freigegeben, und so gelangten die drei Männer ab 1977 noch zu einer bescheidenen Anerkennung in der wissenschaftlichen Gemeinde.

Aufgaben:

1. Installieren Sie eine Trial- oder Freeware-Version von „PGP Desktop“ auf Ihrem PC.
2. Erzeugen Sie Ihre privaten und öffentlichen Schlüssel, und deponieren Sie ihre öffentlichen Schlüssel auf keyserver.pgp.com.
3. Suchen Sie dort meinen öffentlichen Schlüssel: Suchen Sie nach dem Namen ‚Martin Gubler‘. Meine Mail-Adresse ist gub@stafag.ch.
4. Suchen Sie sich eine nette Primzahl aus (oder zwei). Tippen Sie diese Primzahlen in eine Text-Datei und verschlüsseln Sie diese als PGP-Botschaft an mich.
5. Senden Sie mir eine Mail mit der PGP-Botschaft als Attachement.
6. Ich werde versuchen, Ihre Botschaft zu entschlüsseln; und ich werde Ihnen ebenfalls eine PGP-verschlüsselte Primzahl zusenden.
7. Entschlüsseln Sie meine PGP-Botschaft an Sie und drucken Sie die entschlüsselte Botschaft aus. Bringen Sie den Ausdruck mit in die Schule als Beleg dafür, dass alles geklappt hat.
8. Lästern Sie nun abhörsicher untereinander über mich oder die Schulleitung ...

Version 2.2, vom Juli 2016

Ausgearbeitet von Martin Gubler, Kantonsschule Frauenfeld, anno 1999

Mit LaTeX in eine lesbare Form gebracht von Alfred Hepp im August 2011