Schüler-Propädeutikum Mathematik 2020
Vortrag 3: Das RSA-Verfahren

Andreas Lochmann

16. Oktober 2020

1 RSA-Verschlüsselung

Nehmen wir an, Bob möchte an Alice eine Nachricht senden. Da sich jede Nachricht durch Zahlen ausdrücken lassen, nehme wir an, die Nachricht sei durch eine natürliche Zahl M gegeben. (Wenn die Nachricht länger ist, als das folgende Verfahren erlaubt, sendet man halt mehrere Teilnachrichten.) Bob bittet nun Alice, folgende Schritte auszuführen:

1. Alice sucht sich zwei große, voneinander verschiedene Primzahlen p und q aus, die sie für sich behält.
2. Alice berechnet $n = p \cdot q$ und $\phi := (p-1)(q-1)$.
   Man kann zeigen, dass $a^\phi \equiv 1\ (n)$ für alle zu n teilerfremden Zahlen a gilt (dies ist in Analogie zum kleinen Satz von Fermat, nur mit dem Unterschied, dass n hier nicht prim ist und daher $(p-1)(q-1)$ anstelle von $n-1$ eingesetzt werden muss).
3. Alice wählt zufällig eine Zahl e mit $1 < e < \phi$, die zu ϕ teilerfremd ist.
4. Alice berechnet eine Zahl d, so dass $e \cdot d \equiv 1\ (\phi)$ gilt.
5. Alice sendet n und e an Bob (dieses Paar bildet den **öffentlichen Schlüssel**) und behält d für sich, denn diese Zahl wird zum Entschlüsseln benötigt (dies ist der **private Schlüssel**).

Wir gehen davon aus, dass jeder auf der Welt n und e mitlesen kann. Jedoch ist das Zerlegen einer Zahl in Primfaktoren sehr aufwändig, und wenn Alice genügend große Primzahlen p und q gewählt hat, kann sie zwar n sehr schnell berechnen, aber die Zerlegung von n würde für jeden zu viel Zeit kosten, um praktisch realisierbar zu sein. Die Berechnung von d auf Grundlage von e und ϕ ist ebenfalls relativ schnell, wir müssen aber später noch über die Methode sprechen.

Bob muss als nächstes sicherstellen, dass $M < n$ ist, z.B. indem er die Nachricht im Zweifel in mehrere Teile zerlegt und einzeln sendet. Dann berechnet er

$$C \;=\; M^e \,\%\, n\,.$$

Dies ist die verschlüsselte Nachricht (das **Chiffrat**) und sendet C zurück an Alice. Zur Dechiffrierung berechnet Alice jetzt:

$$D \;=\; C^d \,\%\, n \;=\; (M^e)^d \,\%\, n \;=\; M^{e\cdot d} \,\%\, n \;=\; M\,.$$

Dass bei dieser Rechnung tatsächlich M heraus kommt, ist nicht selbstverständlich, und muss von uns noch bewiesen werden.

Wir müssen also noch folgende Punkte adressieren:

1. Warum ist $a^\phi \equiv 1\ (n)$ für alle zu n teilerfremden Zahlen a?
2. Wie findet man eine Zahl d mit $e \cdot d \equiv 1\ (\phi)$?
3. Warum ist $M^{e\cdot d}\,\%n \;=\; M$?
4. Wie viel Zeit benötigt ein Computer für die Erzeugung der Schlüssel, wie viel für die Verschlüsselung, und wie viel für die Entschlüsselung?
5. Wenn ein Angreifer C, e und n kennt, aber weder p, q, ϕ noch d, wie könnte er daraus M zurückgewinnen?

Wir arbeiten die Fragen in der obigen Reihenfolge ab. Dabei seien stets:

- p und q voneinander verschiedene Primzahlen,
- $n = pq$ und $\phi = (p-1)(q-1)$,
- e eine zu ϕ teilerfremde natürliche Zahl und
- d eine Zahl mit $ed \equiv 1\ (\phi)$.

Warum ist $a^\phi \equiv 1\ (n)$?

1.1 Satz *Sei $a \in \mathbb{N}$ eine zu n teilerfremde Zahl. Dann ist $a^\phi \equiv 1\ (n)$.*

Beweis Der Beweis dieses Satzes verläuft analog zum Beweis des kleinen Fermatschen Satzes. Dabei ist wichtig, dass ϕ die Zahl der zu n teilerfremden Zahlen zwischen 1 und n ist. □

Wie findet man eine Zahl d mit $e \cdot d \equiv 1\ (\phi)$?

Der Nachweis, dass e und ϕ teilerfremd sind, und die Berechnung eines passenden d sind beide gleichzeitig mit Hilfe des **Euklidischen Algorithmus** möglich. Diesen möchte ich hier nur exemplarisch darstellen.

Beispiel: Sei $e = 17$ und $\phi = 40$. Wir nehmen wiederholt Division mit Rest vor:

$$\begin{aligned} 40 &= 2 \cdot 17 + 6 \\ 17 &= 2 \cdot 6 + 5 \\ 6 &= 1 \cdot 5 + 1 \end{aligned}$$

Die Zahl, die am Ende übrig bleibt (hier die 1) ist stets ein Teiler von e und ϕ, wie man leicht sieht: Ist x ein Teiler von e und ϕ, so ist notwendigerweise x auch ein Teiler des Rests aus der ersten Division (hier die 6). Dann muss auch der Rest der zweiten Division durch x teilbar sein, und so weiter. Das heißt, dass jeder Rest, auch der letzte, durch x teilbar sein muss. Andererseits werden die Reste immer kleiner, und so muss der letzte Rest der größte gemeinsame Teiler von e und ϕ sein. Im teilerfremden Fall ist dies 1.

Um nun d zu berechnen, stellt man die obigen Gleichungen so um, dass die Reste frei stehen:

$$\begin{aligned} 6 &= 4 - 2 \cdot 17 \\ 5 &= 17 - 2 \cdot 6 \\ 1 &= 6 - 1 \cdot 5 \end{aligned}$$

Setzt man diese Gleichungen nacheinander ineinander ein, kann man die 1 als Kombination aus $e = 17$ und $\phi = 40$ schreiben:

$$1 \;=\; 6 - 1 \cdot 5 \;=\; 6 - 1 \cdot (17 - 2 \cdot 6) \;=\; 3 \cdot 6 - 1 \cdot 17 \;=\; \cdots \;=\; 3 \cdot 40 - 7 \cdot 17$$

Wenn wir jetzt auf beiden Seiten modulo ϕ rechnen, fällt die 40 weg, und wir erhalten $1 \equiv -7 \cdot 17\ (\phi)$. Unser d ist also kongruent zu -7, und wir wählen z.B. $d = 33$, denn $33 \equiv -7\ (40)$.

Warum ist $M^{e \cdot d}\ \%n\ =\ M$?

Die Behauptung ist eine Folgerung aus folgendem, allgemeineren Satz:

1.2 Satz *Seien $r, s \in \mathbb{N}$ mit $r \equiv s\ (\phi)$ und $a \in \mathbb{N}$ teilerfremd zu n. Dann ist $a^r \equiv a^s\ (n)$.*

Beweis Nach Voraussetzung gibt es ein $k \in \mathbb{Z}$ mit $r = s + k \cdot \phi$. Damit ist

$$a^r \;=\; a^{s+k\cdot\phi} \;=\; a^s \cdot (a^k)^\phi \;\equiv\; a^s \cdot 1 \;=\; a^s \pmod{n}\,.$$

□

In unserem Fall ist jetzt $r = e \cdot d$ und $s = 1$. Damit ist $M^{e \cdot d} \,\%\, n = M$ zumindest für solche M bewiesen, die zu n teilerfremd sind. Wenn M und n nicht teilerfremd sind, muss $M = p$ oder $M = q$ sein, und wir können durch direktes Nachrechnen die Behauptung nachweisen.

2 Was ist der Zeitbedarf?

Schlüsselberechnung: Zunächst müssen zwei genügend große Primzahlen berechnet werden – über diesen Zeitbedarf sprechen wir im fünften Vortrag. Die Berechnung von n und ϕ besteht nur aus zwei Multiplikationen und ist entsprechend schnell, selbst für sehr große Primzahlen. Die Auswahl einer zufälligen Zahl e geht leicht – dann muss allerdings geprüft werden, dass e und ϕ teilerfremd sind. Der euklidische Algorithmus kann den ggT berechnen, und benötigt dafür (unter Verwendung normaler Division mit Rest) etwa $c_1 \cdot (\ln(e \cdot \phi))^3$ Schritte, mit einer festen Konstante c_1; oder anders gesagt: Wenn man Zahlen mit doppelt so vielen Ziffern verwendet, wird die Laufzeit des Algorithmus auch nur etwa verachtfacht. Bei der Anwendung des euklidischen Algorithmus erhält man auch d als Nebenprodukt.

Die Schlüsselberechnung selbst für sehr große Primzahlen ist damit relativ schnell möglich. Aber selbst wenn die Schlüsselberechnung viel mehr Zeit benötigen *würde*, müsste diese Berechnung von Alice nur einmal ausgeführt werden, um mehrere sichere Kommunikationen auszuführen.

Verschüsselung: Man könnte meinen, dass die Berechnung der Potenz $M^e \% n$ für die sehr große Zahlen M, e und n sehr lange benötigt; immerhin müsste man $e - 1$ Multiplikationen ausführen, oder? Jedoch gibt es sehr einfache Möglichkeit, diesen Schritt zu beschleunigen. Hier ein Beispiel:

$$13^{11} \;=\; 13^{10} \cdot 13 \;=\; (13^5)^2 \cdot 13 \;=\; ((13^2)^2 \cdot 13)^2 \cdot 13$$

benötigt nur 5 Multiplikationen, nicht 10:

$$\begin{aligned} 13^2 &= 169 \\ 169^2 &= 28.561 \\ 28.561 \cdot 13 &= 371.293 \\ 371.293^2 &= 137.858.491.849 \\ 13 \cdot 137.858.491.849 &= 1.792.160.394.037 \end{aligned}$$

Die Reihenfolge von Quadratbildungen und Multiplikationen mit 13 sind übrigens nicht zufällig, sondern folgen aus der Binärzerlegung von $11 = (1011)_2$; betrachtet man nämlich die obige Rechnung genauer findet man:

$$((13^2)^2 \cdot 13)^2 \cdot 13 \;=\; 13^{2 \cdot 2 \cdot 2} \cdot 13^2 \cdot 13 \;=\; 13^{8+2+1} \;=\; 13^{11}$$

Da die Zahlen im Computer sowieso im Binärsystem realisiert sind, lassen sich so sehr schnell auch große Potenzen berechnen: Die Zahl der benötigten Multiplikationen ist also die Zahl der binären Ziffern des Exponenten minus 1 (für die Quadrate), plus die Zahl der Einsen in der Binärdarstellung des Exponenten (für Multiplikationen mit 13). Wenn also die Zahl der Ziffern sich verdoppelt, verdoppelt sich auch die Laufzeit der Berechnung. Das ist auch dann noch der Fall, wenn nach jeder Multiplikation eine Modulo-Rechnung angehängt wird. Insgesamt sagen wir: Die Verschlüsselung hat eine Laufzeit von etwa $c_2 \cdot \ln(e)$ für eine feste Konstante c_2.

Entschlüsselung: Die Entschlüsselung läuft im Prinzip genau so ab wie die Verschlüsselung, nur mit einem anderen Exponenten. Daher ist der Zeitbedarf vergleichbar.

Aufgabe: Berechnen Sie $3^{18} \,\%\, 100$ mit obiger Methode.

3 Angriffe durch Faktorisierung von n

Überlegen wir zunächst, wie ein konventioneller Angriff aussehen könnte: Eve hat die Kommunikation belauscht und kennt n, e und C. Kann Eve daraus M rekonstruieren?

Eine Möglichkeit besteht natürlich darin, n zu faktorisieren. Dies ist jedoch das Herz des ganzes Algorithmus: Wir wählen so große Primzahlen, dass es nach aktuellem technischen Stand nicht möglich ist, n in sinnvoller Zeit zu faktorisieren. Dazu würde ein Angreifer versuchen, alle Zahlen von 1 bis $\sqrt{n}$ durchzugehen und als Faktor von n zu identifizieren, würde also etwa $\sqrt{n}$ als Laufzeit benötigen. Selbst wenn Eve nur die Primzahlen in diesem Bereich durchgeht, wären im Schlimmstfall noch immer etwa

$$\frac{\sqrt{n}}{\ln \sqrt{n}} \;=\; \frac{2\sqrt{n}}{\ln n}$$

Divisionen nötig – bei genügend großem n ist dies viel zu viel.

Jedoch haben wir im Laufe der Zeit weitere Faktorisierungsmethoden kennengelernt. Hier ist eine einfache von Fermat aus dem Jahr 1643:[1]

Fermats Faktorisierungsmethode: Nehmen wir $p < q$ an. p und q sind (als große Primzahlen) beide ungerade und ihre Differenz daher gerade. Sei $r = (q-p)/2$ die halbe Differenz und $s = (p+q)/2$ der Mittelwert. Dann ist

$$n \;=\; pq \;=\; (s-r)(s+r) \;=\; s^2 - r^2\,.$$

Wenn p und q etwa gleich groß sind, ist r klein und daher r^2 deutlich kleiner als s^2. Wir können nun die Wurzel aus n ziehen (dies geht sehr schnell, mit Laufzeit wie $\ln(n)$), aufrunden und dann nach geeigneten Differenzen suchen.

[1] Dieser Teil wird im fünften Video gezeigt.

Beispiel: Wir faktorisieren $n = 527$. Die Wurzel aus dieser Zahl ist $\sqrt{n} \approx 22{,}956\ldots$, aufgerundet also 23. Dies ist unsere erste Vermutung für s. Wenn $s = 23$ wäre, wäre $r^2 = s^2 - n = 2$. Dies ist keine Quadratzahl, also kann s nicht 23 sein. Versuchen wir als nächstes $s = 24$. Dann ist $r^2 = s^2 - n = 49 = 7^2$. Also finden wir $r = 7$ und damit $n = (s - r) \cdot (s + r) = 17 \cdot 31$.

Aufgabe: Faktorisieren Sie $n = 345.383$ mit Hilfe der Fermatschen Faktorisierungsmethode.

Lösung: Es ist $\sqrt{n} \approx 587{,}69\ldots$, aufgerundet also 588. Wenn $s = 588$ wäre, wäre $r^2 = s^2 - n = 361$. Dies ist eine Quadratzahl: $r = 19$. Daraus folgt $n = (588-19)\cdot(588+19) = 569\cdot 607$; wir haben die Zerlegung von n bereits im ersten Schritt gefunden.

Daraus lernen wir, dass p und q nicht zu nahe beieinander liegen dürfen, sondern sich um mehrere Größenordnungen unterscheiden sollten. Nur dann müssen in der Fermatschen Faktorisierungsmethode so viele Möglichkeiten für s ausprobiert werden, dass sie sich nicht mehr lohnt.

Wir haben unsere Methoden im Laufe der Jahrhunderte weiter verfeinert, und die aktuell beste bekannte Faktorisierungsmethode für beliebige Zahlen bis ca. 100 Dezimalstellen ist das **Quadratische Sieb** aus dem Jahr 1981 mit einer Laufzeit in der Größenordnung von $\exp(\sqrt{\ln n \cdot \ln\ln n})$; vergleichen Sie dies bitte mit der anfangs benannten Methode mit einer Laufzeit von etwa

$$\frac{2\sqrt{n}}{\ln n} \;=\; \exp\left(\ln \frac{2\sqrt{n}}{\ln n}\right) \;=\; \exp\left(\tfrac{1}{2}\ln n + \ln 2 - \ln\ln n\right) .$$

Es gibt wesentlich schnellere Methoden für bestimmte Primzahlen. Wenn beispielsweise $n = pq$ zu faktorisieren ist, und $p - 1$ viele kleine Primfaktoren besitzt, gibt es darauf aufbauend eine Methode, auch die Faktorisierung von n zu beschleunigen (die Pollard-$p-1$-Methode von 1974). Deshalb verlangen wir aktuell von unseren Primzahlen, dass $p - 1$ und $q - 1$ möglichst große Primfaktoren enthalten, im Idealfall z.B. sichere Primzahlen sind (im Sinn einer Sophie-Germain-Primzahl).

Wir wissen nicht, ob und wann wir nicht eine deutlich bessere allgemeine Methode zur Faktorisierung finden werden. Bis dahin gehen wir davon aus, dass die Faktorisierung von n kein plausibler Angriffsvektor ist, solange p und q geeignet gewählt werden.

Eins ist jedoch bekannt: Nehmen wir an, Eve kennt eine Methode, aus n, e und C effizient die Nachricht M zu rekonstruieren. Das ist äquivalent dazu, aus n und e den Dechiffrier-Schlüssel d zu rekonstruieren. Dann kann man zeigen, dass Eve auch eine effiziente Methode zur Berechnung der Faktoren p und q kennen würde. Das wäre sensationell, denn dann würde Eve über eine effiziente Faktorisierungsmethode verfügen! Daher können wir davon ausgehen, dass Eve auch nicht in der Lage ist, d zu berechnen.

4 Andere Angriffsmethoden?

Eve kennt also n, e und C, und weiß, dass $C \equiv M^e \% n$ ist, kann aber p, q und d nicht rekonstruieren. Wenn Eve Glück hat, sind M und e so klein, dass $M^e < n$ ist, dann ist M einfach die e-te Wurzel aus C und leicht zu berechnen, ganz ohne Kenntnis von p und q. Also müssen Alice und Bob sicherstellen, dass e und M nicht zu klein ist. Wenn umgekehrt d zu klein ist, kann Eve die Nachricht schlicht durch Potenzieren berechnen. Eve kann auch die Verschlüsselung mehrfach auf C anwenden, und $C^e \% n$, $(C^e)^e \% n = C^{e^2} \% n$, $C^{e^3} \% n$ etc. berechnen. Eine dieser Zahlen wird notwendigerweise wieder M sein, doch Alice kann durch eine gute Wahl von p und q dafür sorgen, dass dieser Angriff zu viele Schritte erfordert, um brauchbar zu sein.

Wenn Alice dieselbe Nachricht an mehrere Nutzer unter Verwendung desselben e schickt, besteht ebenfalls die Möglichkeit für Eve, die Nachricht M relativ leicht zu berechnen, ohne n faktorisieren zu müssen.

Man muss sich darüber im Klaren sein, dass Alice und Bob keine Menschen sind! Es sind Computerprogramme, die sich nicht davon irritieren lassen, wenn sie innerhalb kurzer Zeit Millionen von Anfragen vom selben Computer erhalten, und weiterhin brav ihre Algorithmen ausführen. Deshalb basieren viele weitere Angriffsmethoden darauf, diese Schwachstelle auszunutzen. Beispielsweise könnte Eve versuchen, den Klartext M von Bob zu raten (denken Sie daran, dass es sich zumeist um standardisierte Protokolle handelt, die zwischen Computer ausgetauscht werden!), und aus der Kenntnis von M und C auf den geheimen Schlüssel von Alice zu schließen. Damit wäre die gesamte weitere Kommunikation von Alice korrumpiert, bis Alice einen neuen Schlüssel wählt.

Aus Gründen wie diesen wird RSA nicht in Reinform verwendet, sondern mit weiteren Verschlüsselungen und Modifikationen angereichert, und immer weiter ergänzt.

Es sollte an dieser Stelle ergänzt werden, dass RSA nicht die einzige Methode zur asymmetrischen Verschlüsselung ist. Es gibt Alternativen, die aber in Bezug auf die Sicherheit oft sehr ähnliche Eigenschaften wie RSA besitzen. Ohne Vollständigkeit zu beanspruchen, liste ich hier drei Alternativen zu RSA auf:

- **ElGamal-Public-Key-Verschlüsselung** basiert darauf, dass es leicht ist, zu potenzieren, aber schwierig, Logarithmen modulo einer Primzahl zu berechnen.
- **Rabin-Public-Key-Verschlüsselung** basiert darauf, dass es leicht ist, zu quadrieren, aber schwierig, Wurzeln modulo n zu berechnen.
- **Verschlüsselung mit elliptischen Kurven** verwendet anstelle von Modulo-Rechnung eine alternative Definition von Addition und Multiplikation, basierend auf Nullstellen bestimmter Polynome in zwei Variablen. Diese Polynome werden aufgrund ihres Ursprungs „elliptische Kurven“ genannt, haben

aber in der tatsächlichen Anwendung eigentlich nichts mehr mit Kurven im traditionellen Sinn zu tun.

Knobelaufgabe: Kann man ein Verschlüsselungsverfahren wie RSA auch verwenden, um sich zu identifizieren? Nehmen wir an, Alice hat, wie zuvor, Primzahlen p und q gewählt und $n = p \cdot q$ zusammen mit einem passenden e veröffentlicht. Bob nimmt jetzt Kontakt auf, kann sich aber nicht sicher sein, ob er tatsächlich mit Alice kommuniziert, oder mit jemandem, der sich nur als Alice ausgibt (denn Alice's öffentlicher Schlüssel (n und e) ist ja jedem bekannt). Wie kann Alice gegenüber Bob beweisen, tatsächlich Alice zu sein, ohne dabei ihren privaten Schlüssel (die Zahl d, oder alternativ die Primzahlen p und q) preiszugeben?