Die Berechnung der Primfaktoren aus den Exponenten

Ernst G. Giessmann, Annie Yousar*

2023-11-23

Zusammenfassung

Wie man bei RSA den geheimen Schlüssel d aus dem Produkt $p \cdot q$ und dem öffentlichen Exponenten e bestimmt, weiß jeder. Wir zeigen hier den umgekehrten Weg, nämlich wie man aus den beiden Exponenten e und d und dem Modul m die Primfaktoren p und q berechnet und geben dafür *Einzeiler* in der Sprache dc an. Dass unter den üblichen RSA-Annahmen über die Primfaktoren die Kenntnis von e und d äquivalent zur Faktorisierung des Moduls ist, war wohl Folklore, und es gab auch einige randomisierte Algorithmen. Hier stellen wir den deterministischen Algorithmus des Annex C der NIST SP800-52B vor ([NIST]).

Wenn $p-1$ und $q-1$ teilerfremd wären, wäre alles so einfach.

Weil dann das kleinste gemeinsame Vielfache KGV von $p-1$ und $q-1$ gleich $(p-1)(q-1)$ wäre, und folglich $e \cdot d - 1$ durch $(p-1)(q-1)$ teilbar ist

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

Andererseits hat man die Division mit Rest für $m = p \cdot q$:

$$e \cdot d - 1 = k \cdot m + r,$$

mit $0 \le r < m$.

Vergleicht man diese beiden Darstellungen von $e \cdot d - 1$

$$h \cdot m - h \cdot (p+q-1) = k \cdot m + r,$$

ergeben sich $h = k+1$ und $r = m - h \cdot ((p+q)-1)$, woraus man leicht die halbe Summe S von p und q

$$S = \frac{p+q}{2} = \frac{1 + (m-r)/(k+1)}{2}$$

erhält. Die halbe Differenz

$$D = \frac{p-q}{2} = \sqrt{S^2 - m}$$

ist nach den binomischen Formeln die Quadratwurzel aus $S^2 - m$. Die beiden Primfaktoren sind dann $p = S + D$ und $q = S - D$.

Man benötigt daher für die Berechnung von p und q nur die Werte k und r aus der Division mit Rest von $e \cdot d - 1$ durch m.

mailto:{giessmann,a.yousar}@informatik.hu-berlin.de*

Obwohl natürlich $p-1$ und $q-1$ nicht teilerfremd sind, löst man damit schon zahlreiche Fälle. Einige Schlüsselerzeugungsroutinen berechnen nämlich d als e^{-1} modulo $(p-1)(q-1)$. Damit ist die Voraussetzung für den obigen Beweis, dass nämlich $e \cdot d - 1$ durch $(p-1)(q-1)$ teilbar ist, automatisch erfüllt[1].

In der Sprache dc geht das in einer Zeile (38 Zeichen), angewendet auf die Eingabe m, e, d (in dieser Reihenfolge)

```
dc << EOF
3604265963 23 2507231783
*1-rdsm~lmr-r1+/1+2/d2^Lm-vsddld-rld+f
EOF
```

Trotzdem können wir die Grundidee auch auf den allgemeinen Fall anwenden: Wir erhalten zwei vergleichbare Darstellungen, wenn wir $e \cdot d - 1$, das durch das kleinste gemeinsame Vielfache KGV von $p-1$ und $q-1$ teilbar ist, mit einem kleinen Vielfachen VGT des größten gemeinsamen Teilers GGT der beiden Zahlen multiplizieren

$$\begin{aligned} e \cdot d - 1 &= h_1 \cdot \mathrm{KGV} \\ \mathrm{VGT} \cdot (e \cdot d - 1) &= h_2 \cdot \mathrm{GGT} \cdot h_1 \cdot \mathrm{KGV} \\ \mathrm{VGT} \cdot (e \cdot d - 1) &= h_2 \cdot h_1 \cdot (p-1)(q-1) \quad = \quad h \cdot m - h \cdot (p+q-1), \end{aligned}$$

und durch die Division mit Rest

$$\mathrm{VGT} \cdot (e \cdot d - 1) = k \cdot m + r, \text{ mit } 0 \le r < m.$$

erhalten wir wieder die zweite Darstellung. Nun setzt man wie vorher $k = h-1$ und $r = m - h \cdot (p+q-1)$ und so weiter. Die weitere Rechnung ändert sich nicht, vorausgesetzt, dass der Wert $h \cdot (p+q-1)$ klein genug, also kleiner als m, ist.

Nun müssen wir aber noch ein kleines Vielfaches VGT des größten gemeinsamen Teilers GGT der beiden Zahlen $p-1$ und $q-1$ finden.

Wie man leicht sieht, teilt *jeder* gemeinsame Teiler von $p-1$ und $q-1$ nicht nur $e \cdot d - 1$, ein Vielfaches der beiden Zahlen, sondern auch $m-1$

$$m - 1 = (p-1) \cdot (q-1) + p + q - 2 = (p-1) \cdot (q-1) + (p-1) + (q-1)$$

und damit auch den größten gemeinsamen Teiler GCD von $e \cdot d - 1$ und $m-1$.

Insbesondere teilt dann der größte gemeinsame Teiler GGT von $p-1$ und $q-1$ auch diesen Teiler GCD von $e \cdot d - 1$ und $m-1$. GCD ist damit ein geeigneter Kandidat für das gesuchte Vielfache VGT des größten gemeinsamen Teilers GGT von $p-1$ und $q-1$.

Man berechnet also VGT als den größten gemeinsamen Teiler GCD von $e \cdot d - 1$ und $m-1$ und bestimmt dann die Werte k und r aus der Division mit Rest von $\mathrm{GCD} \cdot (e \cdot d - 1)$ durch m. Die weitere Berechnung von p und q ändert sich nicht.

Bestimmt man in dem oben angegebenen Beispiel den geheimen Exponenten als Inverse zu e modulo $\mathrm{LCM}(p-1, q-1)$ und nicht im Restklassenring modulo $(p-1)(q-1)$, so scheitert die Rechnung von dc, das Ergebnis wird falsch.

[1] Die notwendige PKCS#1-Bedingung für d ist, dass $ed \equiv 1 \bmod \mathrm{LCM}(p-1, q-1)$ gilt, das bedeutet aber nicht, dass d das Inverse zu e im Restklassenring modulo $\mathrm{LCM}(p-1, q-1)$ sein muss.

Erst mit der zusätzlichen Berechnung des größten gemeinsamen Teilers GCD von $e \cdot d - 1$ und $m - 1$ ist man erfolgreich (braucht in dc dann allerdings 66 Zeichen, die aber noch als Einzeiler durchgehen)[2]

```
dc << EOF
3604265963 23 5180231
*1-dSErdsm1-[dSarLa%d0<a]dsax+LE*lm~lmr-r1+/1+2/d2^Lm-vsddld-rld+f
EOF
```

Da das Programm dc mit beliebiger Genauigkeit rechnet[3], kann man die beiden Programme auch für beliebig große RSA-Schlüssel verwenden, eine Eingabe von Hexadezimalzahlen ist natürlich ebenfalls möglich

```
dc << EOF
16doi
D6D4BBEB 17 95715227
*1-rdsm~lmr-r1+/1+2/d2^Lm-vsddld-rld+f
EOF
```

und entsprechend mit der Berechnung des größten gemeinsamen Teilers von $ed - 1$ und $m - 1$:

```
dc << EOF
16doi
D6D4BBEB 17 4F0B47
*1-dSErdsm1-[dSarLa%d0<a]dsax+LE*lm~lmr-r1+/1+2/d2^Lm-vsddld-rld+f
EOF
```

Zur sicheren Speicherung eines geheimen RSA-Schlüssels reicht folglich eine Schlüsselreferenz (ein Fingerprint) des öffentlichen Schlüssels zusammen mit dem geheimen Exponenten allein aus. Er ist dann auch nicht so einfach als solcher zu erkennen, ganz im Gegensatz zu einem wie üblich gespeicherten RSA-Schlüssel, in dem zwei aufeinanderfolgende Zufallszahlen gleicher Größe als Primzahlen sofort auffallen und deren Produkt die dritte Zufallszahl ergibt.

Literatur

[NIST] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, Scott Simon: Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography, NIST Special Publication 800-56B, Revision 2, 2019-03, https:/doi.org

[2] `[dSarLa%d0<a]dsax+` berechnet den größten gemeinsamen Teiler der beiden oberen Zahlen vom Stapel

[3] In bc wäre das Programm etwas verständlicher, aber länger: `m=3604265963; e=23; d=5180231; a=x=e*d-1; b=m-1; while(a){c=b;b=a;a=c%a}; r=b*x%m; k=(b*x-r)/m; s=(1+(m-r)/(k+1))/2; t=sqrt(s^2-m); s+t; s-t.`