2 Kryptoanalyse des RSA-Verfahrens

„Kryptoanalyse des RSA-Verfahrens“ bedeutet *nicht*, dass das Verfahren gebrochen wird, sondern nur, dass seine Rahmenbedingungen soweit abgesteckt werden, dass man nach menschlichem Ermessen sicher in der Anwendung ist. Insbesondere gilt es einige Fallen zu vermeiden. Zu beantwortende Fragen sind insbesondere:

- Gibt es genügend Schlüssel, um einem Exhaustionsangriff zu entgehen?
- Auf welche mathematische Probleme führt das Brechen eines RSA-Geheimtexts? Oder das Bestimmen des geheimen Schlüssels aus dem öffentlichen?
- Wie muss man die Parameter wählen, um Schwachstellen zu vermeiden?

Einen guten Überblick über das Thema gibt der Artikel

D. Boneh: *Twenty years of attacks on the RSA cryptosystem.* Notices of the American Mathematical Society 46 (1999), 203–213.

Online unter

`http://crypto.stanford.edu/~dabo/abstracts/RSAattack-survey.html`

2.1 Der Primzahlsatz

Sei $\pi(x)$ die Anzahl aller Primzahlen $p \leq x$. Etwas allgemeiner sei $\pi_{a,b}(x)$ die Anzahl der Primzahlen $p \leq x$ der Form $p = ak + b$. Der Primzahlsatz ist die asymptotische Relation

$$\pi_{a,b}(x) \sim \frac{1}{\varphi(a)} \cdot \frac{x}{\ln(x)}$$

unter der Voraussetzung, dass a und b teilerfremd sind. Speziell für $a = 1$ und $b = 0$ wird asymptotisch geschätzt:

$$\pi(x) \sim \frac{x}{\ln(x)}.$$

Über die Qualität dieser Approximation gibt es viele theoretische und empirische Ergebnisse, zum Beispiel eine Formel von Rosser und Schoenfeld

$$\frac{x}{\ln(x)} \cdot \left(1 + \frac{1}{2\ln(x)}\right) < \pi(x) < \frac{x}{\ln(x)} \cdot \left(1 + \frac{3}{2\ln(x)}\right) \text{ für } x \geq 59.$$

Mit Hilfe des Primzahlsatzes kann man, wenn auch nicht völlig exakt, folgende Fragen beantworten:

Wieviele Primzahlen $< 2^k$ gibt es?

Antwort: $\pi(2^k)$, also ungefähr

$$\frac{2^k}{k \cdot \ln(2)}$$

Stück, mindestens (für $k \geq 6$)

$$\frac{2^k}{k \cdot \ln(2)} \cdot \left(1 + \frac{1}{2k\ln(2)}\right).$$

Für $k = 128$ sind das ungefähr $3.8 \cdot 10^{36}$, für $k = 256$ ungefähr $6.5 \cdot 10^{74}$.

Wieviele k-Bit-Primzahlen gibt es?

Antwort: $\pi(2^k) - \pi(2^{k-1})$, also ungefähr

$$\frac{2^k}{k \cdot \ln(2)} - \frac{2^{k-1}}{(k-1) \cdot \ln(2)} = \frac{2^{k-1}}{\ln(2)} \cdot \frac{k-2}{k(k-1)} \approx \frac{1}{2} \cdot \pi(2^k)$$

Stück. Für $k = 128$ sind das ungefähr $1.9 \cdot 10^{36}$, für $k = 256$ ungefähr $3.2 \cdot 10^{74}$. Anders ausgedrückt ist eine zufällig gewählte k-Bit-Zahl mit der Wahrscheinlichkeit

$$\frac{\pi(2^k) - \pi(2^{k-1})}{2^{k-1}} \approx \frac{\pi(2^k)}{2^k} \approx \frac{1}{k \cdot \ln(2)} \approx \frac{1.44}{k}$$

Primzahl; für $k = 256$ ist das ungefähr 0.0056.

Eine zuverlässige untere Schranke erhält man aus der Abschätzung

$$\pi(2^k) - \pi(2^{k-1}) > 0.71867 \cdot \frac{2^k}{k} \qquad \text{für } k \geq 21.$$

Auf jeden Fall gibt es in den für das RSA-Verfahren relevanten Größenbereichen so viele Primzahlen, dass ein Exhaustionsangriff völlig wirkungslos bleiben müsste.

Erweiterungen

Sei p_n die n-te Primzahl, also $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, ... Ferner sei $\vartheta(x)$ die Summe der Logarithmen der Primzahlen $\leq x$,

$$\vartheta(x) = \sum_{p \leq x,\, p \text{ prim}} \ln(p).$$

Dann gelten die asymyptotischen Formeln

$$\begin{aligned} p_n &\sim n \cdot \ln(n), \\ \vartheta(x) &\sim x, \end{aligned}$$

sowie die Fehlerschranken von Rosser/Schoenfeld:

$$\begin{aligned} n \cdot \left(\ln(n) + \ln\ln(n) - \frac{3}{2}\right) &< p_n < n \cdot \left(\ln(n) + \ln\ln(n) - \frac{1}{2}\right) \\ &\qquad \text{für } n \geq 20, \\ x \cdot \left(1 - \frac{1}{\ln(x)}\right) &< \vartheta(x) < x \cdot \left(1 - \frac{1}{2\ln(x)}\right) \quad \text{für } n \geq 41. \end{aligned}$$

2.2 Schlüsselbestimmung und Faktorisierung

Frage: *Wie kann man beim RSA-Verfahren den geheimen Exponenten d aus dem öffentlichen Exponenten e und dem Modul n bestimmen?*

Antwort: Jede der folgenden Aufgaben lässt sich effizient auf die anderen zurückführen:

(A) Finden eines passenden geheimen Schlüssels d.

(B) Bestimmung von $\lambda(n)$ (CARMICHAEL-Funktion).

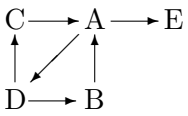
(C) Bestimmung von $\varphi(n)$ (EULER-Funktion).

(D) Faktorisierung von n.

Das Brechen von RSA ist die (möglicherweise echt) leichtere Aufgabe:

(E) Ziehen von e-ten Wurzeln in $\mathbb{Z}/n\mathbb{Z}$.

Der Beweis folgt dem Schema:



Sei dazu $n = p_1 \cdots p_r$ mit verschiedenen Primzahlen $p_1, \ldots, p_r$. Es wird stets angenommen, dass außer n auch der „öffentliche“ Exponent e bekannt ist.

Es ist klar, dass „A $\longrightarrow$ E“ gilt: Wenn d bekannt ist, zieht man die e-te Wurzel durch Potenzieren mit d. Die Umkehrung ist hier allerdings nicht bekannt: *Es könnte sein, dass das Brechen von RSA leichter als die Faktorisierung ist.*

„D $\longrightarrow$ C“: $\varphi(n) = (p_1 - 1) \cdots (p_r - 1)$.

„D $\longrightarrow$ B“: $\lambda(n) = \mathrm{kgV}(p_1 - 1, \ldots, p_r - 1)$.

„B $\longrightarrow$ A“: d wird durch Kongruenzdivision aus $de \equiv 1 \pmod{\lambda(n)}$ gewonnen.

„C $\longrightarrow$ A“: Da $\varphi(n)$ genau die gleichen Primfaktoren hat wie $\lambda(n)$, ist e auch zu $\varphi(n)$ teilerfremd. Durch Kongruenzdivision aus $de \equiv 1 \pmod{\varphi(n)}$ wird ein Exponent d gewonnen, der nicht der „echte“ ist, aber genauso als geheimer Schlüssel funktioniert, da auch $de \equiv 1 \pmod{\lambda(n)}$.

„A $\longrightarrow$ D“ ist wesentlich komplizierter zu zeigen; es wird auch nur ein probabilistischer Algorithmus konstruiert.

Vorbemerkungen

1. *Es reicht, n in zwei echte Faktoren zu zerlegen.*

   (a) Ist nämlich $n = n_1 n_2$ eine solche Zerlegung und o. B. d. A. $n_1 = p_1 \cdots p_s$ mit $1 < s < r$ so ist

   $$\lambda(n_1) = \mathrm{kgV}(p_1 - 1, \ldots, p_s - 1) \,|\, \mathrm{kgV}(p_1 - 1, \ldots, p_r - 1) = \lambda(n),$$

   also auch $de \equiv 1 \pmod{\lambda(n_1)}$. Also ist das Problem auf das analoge für n_1 und n_2 reduziert.

   (b) Da die Zahl der Primfaktoren von n höchstens ${}^2\log(n)$ ist, ist die dadurch gegebene rekursive Reduktion effizient.

2. *Wie kann eine zufällig gewählte Restklasse $w \in \mathbb{Z}/n\mathbb{Z}$ bei der Faktorisierung von n helfen?*

   (a) Findet man ein $w \in [1 \ldots n-1]$ mit $\mathrm{ggT}(w, n) > 1$, so ist n faktorisiert, da $\mathrm{ggT}(w, n)$ ein echter Teiler von n ist.

   (b) Findet man ein $w \in [2 \ldots n-2]$ mit $w^2 \equiv 1 \pmod{n}$ (also eine nichttriviale Quadratwurzel aus 1 in $\mathbb{Z}/n\mathbb{Z}$), so ist n ebenfalls faktorisiert:
   Da $n | w^2 - 1 = (w+1)(w-1)$ und $n \nmid w \pm 1$, ist $\mathrm{ggT}(n, w+1) > 1$, also n nach 1. faktorisiert.

Sei also jetzt ein Paar (d, e) von zusammengehörigen Exponenten bekannt. Dann ist auch $u := ed - 1 = k \cdot \lambda(n)$ bekannt (k und $\lambda(n)$ allerdings nicht).

Da $\lambda(n)$ gerade ist, ist

$$u = r \cdot 2^s \qquad \text{mit } s \geq 1 \text{ und } r \text{ ungerade.}$$

Wählt man irgendein $w \in [1 \ldots n-1]$, so gibt es zwei Möglichkeiten:

- $\mathrm{ggT}(w, n) > 1$ – dann ist n faktorisiert.
- $\mathrm{ggT}(w, n) = 1$ – dann ist $w^{r2^s} \equiv 1 \pmod{n}$.

Im zweiten Fall findet man effizient das minimale $t \geq 0$ mit

$$w^{r2^t} \equiv 1 \pmod{n}.$$

Es gibt wieder zwei Fälle:

- $t = 0$ – Pech gehabt.
- $t > 0$ – dann ist $w^{r2^{t-1}}$ eine Quadratwurzel $\neq 1$ aus 1 in $\mathbb{Z}/n\mathbb{Z}$.

Im zweiten Fall wird unterschieden:

- $w^{r2^{t-1}} \equiv -1 \pmod{n}$ – Pech gehabt.
- $w^{r2^{t-1}} \not\equiv -1 \pmod{n}$ – dann ist n nach Vorbemerkung 2 faktorisiert.

Insgesamt haben wir bei diesem Verfahren nach beliebiger Wahl von $w \in [1 \ldots n-1]$ vier Ausgänge, zwei, bei denen n faktorisiert wird, und zwei, bei denen dies nicht der Fall ist. Die letzteren werden mit

$$\begin{array}{lll} (\mathrm{E}_{n,u}(w)/\mathrm{I}) & w^r \equiv 1 \pmod{n} & \text{und} \\ (\mathrm{E}_{n,u}(w)/\mathrm{II}) & w^{r2^t} \equiv -1 \pmod{n} & \text{für ein } t \text{ mit } 0 \leq t < s \end{array}$$

bezeichnet. Insgesamt ergibt sich die Baumstruktur:

$w \in [1 \ldots n-1] \longrightarrow$
  $\mathrm{ggT}(w,n) > 1 \longrightarrow n$ faktorisiert.$\diamond$
  $w \in \mathbb{M}_n \longrightarrow$
    $w^r \equiv 1 \pmod{n} \longrightarrow (\mathrm{E}_{n,u}(w)/\mathrm{I}.)\diamond$
    $w^r \not\equiv 1 \pmod{n} \longrightarrow$
      $w^{r2^t} \equiv -1 \pmod{n} \longrightarrow (\mathrm{E}_{n,u}(w)/\mathrm{II}.)\diamond$
      $w^{r2^t} \not\equiv -1 \pmod{n} \longrightarrow n$ faktorisiert.$\diamond$

Wir können also n „mit hoher Wahrscheinlichkeit“ faktorisieren, wenn es nur „wenige“ solcher „schlechten“ Zahlen mit $(\mathrm{E}_{n,u}(w)/\mathrm{I,II})$ gibt. Wie viele es sind, wird im nächsten Abschnitt untersucht.

2.3 Die Wahrscheinlichkeit der Faktorisierung

Sei $n \in \mathbb{N}_3$. Ferner sei zunächst $u \in \mathbb{N}_2$ beliebig gerade, $u = r \cdot 2^s$ mit ungeradem r und $s \geq 1$. Dazu werden folgende Mengen eingeführt:

$$\begin{aligned}
A_u^{(0)} &= B_u^{(0)} := \{w \in \mathbb{M}_n \mid w^r = 1\} \quad [\text{Fall } (\mathrm{E}_{n,u}/\mathrm{I})],\\
A_u^{(t)} &:= \{w \in \mathbb{M}_n \mid w^{r\cdot 2^t} = 1, w^{r\cdot 2^{t-1}} \neq 1\} \quad \text{für } 1 \leq t \leq s,\\
B_u^{(t)} &:= \{w \in A_u^{(t)} \mid w^{r\cdot 2^{t-1}} = -1\} \quad [\text{Fall } (\mathrm{E}_{n,u}/\mathrm{II})],\\
A_u &:= \bigcup_{t=0}^{s} A_u^{(t)} = \{w \in \mathbb{M}_n \mid w^u = 1\},\\
B_u &:= \bigcup_{t=0}^{s} B_u^{(t)} \quad [\text{Fall } (\mathrm{E}_{n,u}) \text{ (I oder II)}].\\
C_0 &:= \{w \in \mathbb{M}_n \mid \operatorname{Ord} w \text{ ungerade}\},\\
C_1 &:= \{w \in \mathbb{M}_n \mid -1 \in \langle w \rangle\},\\
C &:= C_0 \cup C_1.
\end{aligned}$$

Bemerkungen

1. $A_u^0 \leq A_u \leq \mathbb{M}_n$ sind Untergruppen, ebenso $A_u^0 \leq C_0 \leq \mathbb{M}_n$.
2. $B_u^{(t)} = A_u^{(t)} \cap C$ für $t = 0, \ldots, s$, denn in einer zyklischen Gruppe $\langle w \rangle$ kann es außer 1 nur eine weitere Quadratwurzel aus 1 geben.
3. Also gilt auch $B_u = A_u \cap C$.
4. B_u ist im Fall von Abschnitt 2.2 genau die Ausnahmemenge mit $(\mathrm{E}_{n,u})$, die nicht zur Faktorisierung von n führt. Der folgende Satz sagt aus, dass die Wahrscheinlichkeit, zufällig ein Element dieser Ausnahmemenge zu erwischen, $< \frac{1}{2}$ ist; probiert man der Reihe nach k zufällige Elemente, ist die Wahrscheinlichkeit, n nicht faktorisiert zu haben, $< 1/2^k$, wird also sehr schnell *extrem* klein.

Satz 1 *Sei n ungerade und keine Primpotenz. Sei $u = r \cdot 2^s$ ein Vielfaches von $\lambda(n)$ mit ungeradem r. Dann ist*

$$\#B_u \leq \frac{1}{2} \cdot \varphi(n).$$

Beweis. Nach dem *folgenden Hilfssatz* ist C und damit erst recht B_u in einer echten Untergruppe von $\mathbb{M}_n$ enthalten. $\diamond$

Hilfssatz 1 (Dixon, AMM 1984) *Sei $n \in \mathbb{N}_3$. Ferner sei $\langle C \rangle = \mathbb{M}_n$. Dann ist n eine Primpotenz oder gerade.*

Beweis. Sei in diesem Beweis $\lambda(n) = r \cdot 2^s$ mit ungeradem r. (Da $n \geq 3$, ist $s \geq 1$. In der „alten“ Bedeutung werden r und s hier nicht benötigt.) Sei

$$h : \mathbb{M}_n \longrightarrow \mathbb{M}_n, \qquad w \mapsto w^{r \cdot 2^{s-1}}.$$

Dann ist h Gruppen-Homomorphismus mit $h(\mathbb{M}_n) \subseteq \{v \in \mathbb{M}_n \mid v^2 = 1\}$ (Gruppe der zweiten Einheitswurzeln $\bmod n$). Da die $w \in C_0$ ungerade Ordnung haben, ist $h(C_0) \subseteq \{1\}$. Für $w \in C_1$ ist $h(w) \in \langle w \rangle$ und $h(w)^2 = 1$, also $h(w)$ eine der beiden Einheitswurzeln $\pm 1 \in \langle w \rangle$.

Insgesamt ist $h(C) \subseteq \{\pm 1\}$.

Ist n keine Primpotenz, so $n = pq$ mit teilerfremden $p, q \in \mathbb{N}_2$. Da $2^s | \lambda(n) = \mathrm{kgV}(\lambda(p), \lambda(q))$, können wir o. B. d. A. $2^s | \lambda(p)$ annehmen. Nach dem chinesischen Restsatz gibt es ein $w \in \mathbb{M}_n$ mit $w \equiv 1 \pmod{q}$, so dass $w \bmod p$ die Ordnung 2^s hat. Dann ist $h(w) \not\equiv 1 \pmod{p}$, also erst recht $h(w) \neq 1$. Da $h(w) \equiv 1 \pmod{q}$, ist auch $h(w) \neq -1$ – außer wenn $q = 2$.

Also ist $h(\mathbb{M}_n) \not\subseteq \{\pm 1\}$, es sei denn, n erfüllt die Behauptung des Hilfssatzes. $\diamond$

2.4 Faktorisierungsalgorithmen

Wie schnell kann man große Zahlen faktorisieren?

- Es gibt „schnelle“ Faktorisierungverfahren für Zahlen der Gestalt $a^b \pm c$ mit „kleinen“ Werten a und c, z. B. für die MERSENNE- und FERMAT-Zahlen $2^b \pm 1$. Die Wahrscheinlichkeit, bei der Erzeugung von RSA-Schlüsseln aus zufällig gewählten Primzahlen einen solchen Modul zu konstruieren, ist verschwindend gering und wird gewöhnlich vernachlässigt.

- Die FERMAT-Faktorisierung von n: Man sucht eine Zahl $a \geq \sqrt{n}$, so dass $a^2 - n$ eine Quadratzahl $= b^2$ ist. Dann ist

$$n = a^2 - b^2 = (a + b)(a - b)$$

  faktorisiert. [Beispiel: $n = 97343$, $\sqrt{n} \approx 311.998$, $312^2 - n = 1$, $n = 313 \cdot 311$.] Diese Methode ist effizient, wenn a nahe bei $\sqrt{n}$ gefunden wird, also $a^2 \approx n$ ist, also im Fall $n = pq$, wenn die Differenz $|p - q|$ der Faktoren klein ist.

- Die schnellsten allgemein anwendbaren Faktorisierungverfahren –
  - Zahlkörpersieb (SILVERMAN 1987, POMERANCE 1988, A. K. LENSTRA/ H. W. LENSTRA/ MANASSE/ POLLARD 1990),
  - Elliptische-Kurven-Faktorisierung (H. W. LENSTRA 1987, ATKIN/ MORAIN 1993), –

  haben einen Zeitaufwand der Größenordnung

$$L_n := e^{\sqrt[3]{\ln n \cdot (\ln \ln n)^2}},$$

  sind also „subexponenziell“, aber noch „superpolynomiell“. *Insbesondere ist die Faktoriserung als Angriff auf das RSA-Verfahren* wesentlich *effizienter als die vollständige Suche.*

Daraus ergeben sich folgende Schätzungen:

Zahl	Bits	Dezimal-stellen	Aufwand (MIPS-Jahre)	Status
rsa120	399	120	100	auf PC $<$ 1 Woche
rsa154	512	154	100 000	TE RIELE 1999
rsa200	665	200	$(*)$	FRANKE 2005
	1024	308	10^{11}	nicht mehr sicher
	2048	616	10^{15}	kurzfristig sicher

$(*)$ mit 80 Rechnern à 2.2 GHz in 4.5 Monaten

Bei der Extrapolation der Aufwandsschätzungen ist zu beachten:

- Sie sind mit großer Unsicherheit behaftet,
- sie gelten nur, solange keine wesentlich schnelleren Faktorisierungsalgorithmen gefunden werden.

Insbesondere ist *bisher nicht bewiesen*, dass es nicht vielleicht doch einen polynomiellen Faktorisierungsalgorithmus gibt.

Neuere Entwicklungen (in der obigen Tabelle schon berücksichtigt) sind:

- Ein Artikel von A. K. Lenstra/ E. Verheul, *Selecting cryptographic key sizes*, der den Stand der Technik im Jahr 2000 zusammenfasst und extrapoliert. Die Abschätzungen wurden als zu pessimistisch bewertet, da der Speicherbedarf der Verfahren nicht berücksichtigt wurde.
- Ein Vorschlag von Bernstein, *Circuits for integer factorization*, der die Stellenzahl (!) verdreifacht, die bei festem Aufwand mit dem schnellsten Faktorisierungsverfahren erreichbar ist.
- Spezielle Hardware-Designs von Shamir und Mitarbeitern:
  - TWINKLE (The Weizmann Institute Key Locating Machine) – die Hardware-Umsetzung einer Idee von Lehmer aus den dreißiger Jahren, die das Faktorisieren um den Faktor 100 – 1000 beschleunigt (1999),
  - TWIRL (The Weizmann Institute Relation Locator) – der das Faktorisieren um einen weiteren Faktor 1000 – 10000 beschleunigt (2003) unter Berücksichtigung des Vorschlags von Bernstein,

  also insgesamt ein Faktor etwa 10^6 (oder 2^{20}) für das Zahlkörpersieb, ohne allerdings die Größenordnung L_n der Komplexität zu ändern.

Insbesondere sieht die Abschätzung von Lenstra/ Verheul jetzt eher zu optimistisch aus. 1024-*Bit-Schlüssel sollten schnellstens aus dem Verkehr gezogen werden.* 2048-Bit-Schlüssel sind gerade noch für ein paar Jahre als sicher zu betrachten.

Empfehlung: Die Primzahlen p und q, aus denen der RSA-Modul $n = pq$ konstruiert wird, sollen mit mindestens 1024 Bit Länge gewählt werden, und zwar so, dass auch ihre Differenz $|p-q|$ eine Länge in der Größenordnung 1024 Bit hat.

2.5 Iterationsangriff

Sei allgemein $E\colon M \longrightarrow M$ eine bijektive Abbildung der endlichen Menge M und $D = E^{-1}$ die Umkehrabbildung; wir stellen uns E als Verschlüsselungsfunktion vor.

Dann ist E in der Permutationsgruppe $\mathfrak{S}(M)$ enthalten, die die (riesige) Ordnung $\#\mathfrak{S}(M) = (\#M)!$ hat. Immerhin ist sie endlich, und somit gibt es ein $s \in \mathbb{N}_1$ mit $E^s = \mathbf{1}_M$, also

$$D = E^{s-1}.$$

Also ist D aus E bestimmbar durch hinreichend häufige Iteration – ein Angriff, der natürlich nur für asymmetrische Verfahren relevant ist. Um sich davor zu schützen, muss man *die Ordnung von E – das kleinste $s \geq 1$ mit $E^s = \mathbf{1}_M$ – möglichst groß wählen.*

Das Beispiel RSA

Hier ist $M = C = \mathbb{Z}/n\mathbb{Z}$, also $\#\mathfrak{S}(M) = n!$, wobei n schon eine sehr große Zahl ist, so dass noch nicht unmittelbar eine Gefahr zu sehen ist – der Angreifer könnte zwar $E^{n!-1}$ bilden, aber das wird er selbst mit dem effizientesten Potenzalgorithmus in diesem Universum nicht schaffen.

Allerdings sind die RSA-Verschlüsselungsfunktionen in einer wesentlich kleineren Untergruppe von $\mathfrak{S}(M)$ enthalten – deren Ordnung der Angreifer glücklicherweise nicht kennt:

Um das herzuleiten, betrachten wir die Abbildung

$$\Phi : \mathbb{N} \longrightarrow \mathrm{Abb}(M, M), \quad e \mapsto E_e \quad \text{mit } E_e(a) = a^e \bmod n.$$

Sie hat die folgenden Eigenschaften:

Bemerkungen

1. Für $e, f \in \mathbb{N}$ ist $E_{ef} = E_e \circ E_f$, weil $a^{ef} \equiv (a^f)^e \pmod n$ für alle $a \in M$. Also ist Φ Homomorphismus der multiplikativen Halbgruppe $\mathbb{N}$.

2. Ist $e \equiv f \pmod{\lambda(n)}$, so ist $E_e = E_f$: Wenn $f = e + k\lambda(n)$, folgt $a^f = a^{e+k\lambda(n)} \equiv a^e \pmod n$ für alle $a \in M$.

3. Ist e mod $\lambda(n)$ invertierbar, so ist E_e bijektiv: Ist $de \equiv 1 \pmod{\lambda(n)}$, so $E_d \circ E_e = E_1 = \mathbf{1}_M$. Also ist die von Φ induzierte Abbildung,

   $$\bar{\Phi} : \mathbb{M}_{\lambda(n)} \longrightarrow \mathfrak{S}(M),$$

   ein Gruppen-Homomorphismus.

4. $\bar{\Phi}$ ist injektiv: Ist nämlich $\Phi(e) = E_e = \mathbf{1}_M$, so ist $a^e \equiv a \pmod{n}$ für alle $a \in M$, also $a^{e-1} \equiv 1 \pmod{n}$ für alle $a \in \mathbb{M}_n$, also $\lambda(n)|e-1$, also $e \equiv 1 \pmod{\lambda(n)}$.

Damit ist bewiesen:

Satz 2 *Die RSA-Verschlüsselungsfunktionen E_e bilden eine zu $\mathbb{M}_{\lambda(n)}$ isomorphe Untergruppe $H \leq \mathfrak{S}(M)$ von der Ordnung $\varphi(\lambda(n))$ und vom Exponenten $\lambda(\lambda(n))$.*

Die Ordnung einer einzelnen Verschlüsselungsfunktion E_e kann natürlich noch viel kleiner sein; die zyklische Untergruppe $\langle e \rangle \leq \mathbb{M}_{\lambda(n)}$ hat die Ordnung $s := \mathrm{Ord}(e)|\lambda(\lambda(n))$.

Damit sind wir auf folgende Probleme gestossen:

1. Wie groß ist $\lambda(\lambda(n))$?

Antwort (ohne Beweis): „Im allgemeinen“ ist $\lambda(\lambda(n)) \approx \frac{n}{8}$.

Will man sich dessen sicher sein, wählt man p, q speziell, d. h., $p = 2p'+1$, $q = 2q'+1$ mit verschiedenen Primzahlen $p', q' \geq 3$. (Solche Primzahlen p', q' heißen Germain-**Primzahlen** nach Sophie Germain, die durch Betrachtung dieser Zahlen einen wesentlichen Fortschritt für das Fermat-Problem erzielt hatte.) Für $n = pq$ ist dann

$$\lambda(n) = \mathrm{kgV}(2p', 2q') = 2p'q' \approx \frac{n}{2}.$$

Sind weiterhin auch $p' = 2p'' + 1$ und $q' = 2q'' + 1$ speziell, so ist

$$\lambda(\lambda(n)) = 2p''q'' \approx \frac{n}{8}.$$

Der Primzahlsatz lässt erwarten, dass es auch solche Zahlen noch in astronomischen Mengen gibt.

2. Wann ist $\mathrm{Ord}(e) = \lambda(\lambda(n))$? Oder nicht wesentlich kleiner?

Antwort: meistens. (Auch hier existiert eine mathematische Analyse.)

Als Folgerung kann man festhalten: *Bis auf vernachlässigbar unwahrscheinliche Ausnahmen ist s von der gleichen Größenordnung wie n.* Die besten bekannten Ergebnisse dazu finden sich im Kapitel 23 des Buchs von Shparlinski, siehe das Literaturverzeichnis der Vorlesung.

Ergänzend zu Abschnitt 2.2 lässt sich jetzt die Aufgabe

(F) Finden der Ordnung s der Verschlüsselungsfunktion

hinzufügen. Es gilt die komplexitätstheoretische Implikation

$$(\mathrm{F}) \longrightarrow (\mathrm{A})$$

(wenn die Ordnung s bekannt ist, ist $D = E^{s-1}$ und daher auch $d = e^{s-1}$ bekannt), aber vielleicht nicht die Umkehrung. *Die Ordnung der Verschlüsselungsfunktion zu finden, ist mindestens so schwierig das Faktorisieren des Moduls.*

2.6 Brechen *eines* Geheimtextes

Das Brechen von Geheimtexten könnte aber noch leichter sein: Für einen gegebenen Geheimtext c könnte nämlich $E_e^r(c) = c$ sein, obwohl $E_e^r \neq \mathbf{1}_M$ ist. Ist dann a der Klartext, also $c = E_e(a)$, so kann der Kryptoanalytiker berechnen:

$$E_e^{r-1}(c) = D_e(E_e^r(c)) = D_e(c) = a.$$

Die mathematische Beschreibung dieser Situation sieht so aus:

- $\mathbb{M}_{\lambda(n)}$ operiert auf der Menge $M = \mathbb{Z}/n\mathbb{Z}$, ebenso die zyklische Untergruppe $G := \langle e \rangle \leq \mathbb{M}_{\lambda(n)}$.
- Für $a \in M$ ist $G \cdot a = \{a^{e^k} \mid 0 \leq k < s\}$ die Bahn.
- Der Stabilisator $G_a = \{f \in G \mid a^f \equiv a \pmod{n}\}$ ist Untergruppe von G; zwischen den Mengen $G \cdot a$ und G/G_a gibt es eine natürliche Bijektion.
- Für die Bahnlänge $t = \#G \cdot a$ gilt

$$t = \frac{s}{\#G_a}, \qquad t|s|\lambda(\lambda(n))$$
$$E_e^r(c) = c \iff E_e^r(a) = a \iff t|r.$$

- $G \cdot c = G \cdot a$ und $\#G_c = \#G_a$. (Die beiden Stabilisatoren sind zueinander konjugiert.)

Damit sind wir auf ein weiteres Problem gestoßen:

3. Wann ist $t = s$, d. h., der Stabilisator G_a trivial? Oder zumindest sehr klein?

   Antwort auch hier: meistens.

Das Finden der Bahnlänge t von a und c ist also mindestens so schwierig wie das Brechen des Geheimtextes c.

Zwei neuere Artikel zeigen, wie gering das Risiko ist, versehentlich eine kleine Bahnlänge zu erwischen und somit den Iterationsangriff zu ermöglichen:

- J. J. Brennan/ Bruce Geist, Analysis of iterated modular exponentiation: The orbits of x^α mod N. **Designs, Codes and Cryptography** 13 (1998), 229–245.
- John B. Friedlander/ Carl Pomerance/ Igor E. Shparlinski, Period of the power generator and small values of Carmichael's function. **Mathematics of Computation** 70 (2001), 1591–1606, + 71 (2002), 1803–1806.

2.7 Mehrfach-Verwendung eines Moduls

Frage: Was passiert, wenn beim RSA-Verfahren zwei verschiedene Teilnehmer denselben Modul n verwenden?

D. h., A und B haben die öffentlichen Schlüssel (n, e_A) und (n, e_B).

Zunächst sind offensichtlich A und B voreinander nicht sicher, da beide n faktorisieren, also auch den geheimen Schlüssel des jeweils anderen bestimmen können. Ein gemeinsamer Modul ist also höchstens in einer Kooperationssituation sinnvoll, wo A und B einander uneingeschränkt vertrauen.

Es kommt aber noch schlimmer: Falls jemand die gleiche Nachricht a an A und B schickt, kann *jeder* diese entschlüsseln. Die Geheimtexte sind:

$$c_\mathrm{A} = a^{e_\mathrm{A}} \bmod n, \quad c_\mathrm{B} = a^{e_\mathrm{B}} \bmod n.$$

Unter der nicht wesentlich einschränkenden Annahme, dass e_A und e_B teilerfremd sind, findet eine Angreiferin mit dem erweiterten EUKLIDischen Algorithmus Koeffizienten x und y mit

$$xe_\mathrm{A} + ye_\mathrm{B} = 1;$$

dabei haben x und y notwendigerweise verschiedenes Vorzeichen, o. B. d. A. $x < 0$. Ist $\mathrm{ggT}(c_\mathrm{A}, n) > 1$, so kann die Angreiferin n faktorisieren und ist fertig. Andernfalls bestimmt sie

$$g := c_\mathrm{A}^{-1} \bmod n$$

durch Kongruenzdivision (also wieder mit dem EUKLIDischen Algorithmus) und hat dann

$$g^{-x} \cdot c_\mathrm{B}^{y} \equiv (a^{e_\mathrm{A}})^x \cdot (a^{e_\mathrm{B}})^y \equiv a \pmod{n}.$$

Die geheimen Schlüssel d_A und d_B hat sie damit allerdings nicht bestimmt.

Der gemeinsame Modul n ist also nur sicher, wenn A und B sich voll vertrauen und diesen Modul außerdem geheim halten. Dann können sie nur miteinander kommunizieren – und da ist es viel effizienter, gleich eine symmetrische Chiffre zu verwenden.

2.8 Kleine Exponenten

Frage: Ist es beim RSA-Verfahren gefährlich, einen kleinen öffentlichen Exponenten e zu wählen?

Der Exponent $e = 1$ ist völlig unbrauchbar, da dann nicht verschlüsselt wird.

Der Exponent $e = 2$ ist für RSA nicht verwendbar, da er gerade und somit nicht teilerfremd zu $\lambda(n)$ ist. Es gibt allerdings eine verwandte Chiffre, das RABIN-Verfahren, das gerade mit $e = 2$ so arbeitet. Bei diesem muss der Empfänger Quadratwurzeln $\bmod n$ ziehen können, und das kann er, wenn er n faktorisieren kann (siehe später). (Er muss auch noch eine Möglichkeit haben, unter den verschiedenen Quadratwurzeln den richtigen Klartext zu erkennen.)

Der kleinste für das RSA-Verfahren prinzipiell geeignete Exponent ist $e = 3$. Hier entsteht eine Schwäche, sobald jemand die gleiche Nachricht a an *drei* verschiedene Teilnehmer A, B und C schickt. Deren öffentliche Schlüssel seien $(n_\mathrm{A}, 3)$, $(n_\mathrm{B}, 3)$ und $(n_\mathrm{C}, 3)$. O. B. d. A. sind die Moduln n_A, n_B und n_C paarweise teilerfremd, sonst kann die Angreiferin mindestens zwei davon faktorisieren, also erst recht a lesen. Ansonsten hat sie (mit etwas Glück) drei Geheimtexte

$$c_\mathrm{A} = a^3 \bmod n_\mathrm{A}, \quad c_\mathrm{B} = a^3 \bmod n_\mathrm{B}, \quad c_\mathrm{C} = a^3 \bmod n_\mathrm{C}$$

mit $0 \le a < n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}$, also $a^3 < n_\mathrm{A} n_\mathrm{B} n_\mathrm{C}$. Mit dem chinesischen Restalgorithmus konstruiert sie daraus eine Zahl $\tilde{c} \in \mathbb{Z}$ mit

$$0 \le \tilde{c} < n_\mathrm{A} n_\mathrm{B} n_\mathrm{C}$$

und

$$\tilde{c} \equiv c_\mathrm{X} \quad \text{für X} = \mathrm{A}, \mathrm{B}, \mathrm{C}.$$

Wegen der Eindeutigkeit ist $\tilde{c} = a^3$ in $\mathbb{Z}$. Damit bestimmt sie $a = \sqrt[3]{\tilde{c}}$ durch Wurzelziehen in $\mathbb{Z}$. (Auch hier gelingt es ihr nicht, die geheimen Exponenten zu bestimmen.)

Die Verallgemeinerung dieses Angriffs auf einen beliebigen „kleinen“ gemeinsamen öffentlichen Exponenten ist offensichtlich: Wenn eine identische Nachricht an e verschiedene Teilnehmer geschickt wird, kann diese von jedem gelesen werden. So abwegig ist dieser Angriffspunkt nicht, wenn man sich etwa eine konstante „Protokollinformation“ am Beginn längerer Nachrichten vorstellt. In der Praxis wird der Exponent $e = 2^{16} + 1 = 65537$ meist als hinreichend sicher für „normale“ Anwendungen angesehen.

2.9 Die Signaturfalle

Ein Problem, das nicht die Sicherheit der Chiffre RSA, sondern die Rahmenbedingungen ihrer Anwendung betrifft, ist die Signaturfalle: Da RSA in umgekehrter Reihenfolge zur digitalen Signatur verwendet wird, muss man stets darauf achten, dass man nicht im Glauben, einen Text digital zu signieren, in Wirklichkeit einen vorgelegten Geheimtext entschlüsselt. Würden wirklich normlerweise Klartexte signiert, würde das sofort ins Auge springen. Es gibt aber (mindestens) drei Gründe, warum das nicht so ist:

1. Bei der digitalen Signatur wird aus Performanzgründen so gut wie immer ein (kryptographischer) Hash-Wert der Nachricht signiert. Dieser ist von einer Zufalls-Bitkette nicht zu unterscheiden.

2. Bei der starken Authentisierung wird statt einer Passworteingabe als Identitätsnachweis eine zufällige Bitkette digital signiert. Selbst wenn das Ergebnis ein entschlüsselter Klartext wäre – man sieht es in der Regel gar nicht, es wird direkt dem Kommunikationspartner (z. B. Zielrechner) übersandt.

3. Außerdem kann ein beliebiger Text durch „Camouflage“ vorverschlüsselt zum Signieren bzw. Entschlüsseln vorgelegt werden. Selbst wenn man sich das Ergebnis dieses Vorgangs ansieht, erkannt man nicht, dass man in Wirklichkeit entschlüsselt hat – siehe unten. Diese Eigenschaft des RSA-Verfahrens ist sogar sehr nützlich: Sie ist Grundlage der blinden Signatur und damit der Erzeugung von digitalen Pseudonymen und anonymen Berechtigungsnachweisen. Siehe
   `http://www.uni-mainz.de/~pommeren/DSVorlesung/KryptoProt/`

Es handelt sich hierbei übrigens um einen *Angriff mit gewähltem Geheimtext*. Dieses Problem wird in der Praxis dadurch umgangen, dass man für die drei – evtl. vier – Funktionen

- Chiffrierung,
- digitale Signatur,
- starke Authentisierung,
- evtl. blinde Signatur

jeweils separate Schlüsselpaare erzeugt und verwendet.

Nun also zu der „Camouflage“, die zur Verschleierung des Angriffs mit gewähltem Geheimtext dient. Der Ablauf ist:

1. Der Angreifer M hat einen Geheimtext $x = E_{\mathrm{A}}(a)$ aufgefangen. Er verschlüsselt ihn mit einer nur ihm bekannten Funktion C zu $y = C(x)$.

2. Er schiebt y dem Opfer A zur digitalen Signatur unter; dieses erzeugt $z = D_{\mathrm{A}}(y)$.

3. Der Angreifer entfernt die „Camouflage“ durch eine geeignete Rücktransformation C'. Hat er ein Paar (C, C') zur Verfügung, so dass $C' \circ D_{\mathrm{A}} \circ C = D_{\mathrm{A}}$, so ist $a = D_{\mathrm{A}}(x) = C'(z)$.

Eine Besonderheit des RSA-Verfahrens ist, dass ein solches Paar (C, C') von Transformationen existiert; ist $E_{\mathrm{A}}(a) = a^e \bmod n$, so ist C die Verschiebechiffre auf $\mathbb{M}_n = (\mathbb{Z}/n\mathbb{Z})^\times$ mit u^e und C' die Multiplikation mit $u^{-1} \bmod n$, wobei $u \in \mathbb{M}_n$ zufällig gewählt wird. Der Ablauf des Angriffs ist also:

1. M wählt u und bildet $y = C(x) = u^e x \bmod n$.

2. A erzeugt $z = y^d \bmod n$.

3. M berechnet $C'(z) = zu^{-1} = y^d u^{-1} = u^{ed} x^d u^{-1} = x^d = a$ in $\mathbb{Z}/n\mathbb{Z}$.

2.10 Weitere Angriffe

Einige weitere Angriffsansätze werden hier nur überblicksartig erwähnt; für eine ausführlichere Behandlung sei auf den in der Einleitung zitierten Artikel von D. Boneh verwiesen:

1. **Kleine private Exponenten.** M. Wiener hat eine Möglichkeit entdeckt, mit Hilfe eines Kettenbruch-Algorithmus den privaten Schlüssel d aus dem öffentlichen Schlüssel (n, e) effizient zu bestimmen, wenn $d < \frac{1}{3} \cdot \sqrt[4]{n}$.

2. **Abhängige Klartexte** nach Franklin/ Reiter. Gibt es einen affinen Zusammenhang $a_2 = sa_1 + t$ zwischen zwei verschiedenen Klartexten a_1 und a_2 mit bekannten Koeffizienten s und $t \neq 0$, so sind die Klartexte effizient aus dem öffentlichen Schlüssel (n, e), den Koeffizienten s und t sowie den zugehörigen Geheimtexten bestimmbar. Coppersmith hat gezeigt, wie man einen solchen affinen Zusammenhang antreffen kann, wenn a_1 und a_2 aus demselben Klartext durch unterschiedliches „Padding“ entstehen.

3. **Partielles Leck** nach Boneh/ Durfee/ Frankel/ Coppersmith. Falls das letzte Viertel der Bits von einer der Zahlen d (dem privaten Schlüssel) p oder q (den Primfaktoren des Moduls) bekannt sind, ist n effizient faktorisierbar.

4. **Timing- und Power-Attacken** nach Kocher. Hier wird der Prozessor beim Verschlüsseln beobachtet und aus seinem Zeitbedarf oder Stromverbrauch auf die Bits des geheimen Schlüssels geschlossen; dieser Angriff beruht darauf, dass diese Werte in den Ressourcenverbrauch des binären Potenzalgorithmus eingehen.

5. **Differenzielle Fehleranalyse** nach Shamir u. a. Hier wird der Prozessor (etwa auf einer Chipkarte) ein wenig über den spezifizierten Bereich seines einwandfreien Funktionierens gebracht – etwa durch Verbiegen, Erwärmen, Bestrahlen. Aus einzelnen Bitfehlern werden statistische Aussagen über die internen Parameter gewonnen.

Eine Reihe weiterer Angriffe sind bekannt, die sich nicht gegen das RSA-Verfahren direkt richten, sondern gegen Fehler bei der Implementation, Verwendung in kryptographischen Protokollen, Einbindung in eine Systemumgebung u. ä.

Dass ein RSA-Verfahren mit mehr als zwei Primfaktoren nicht schwächer, gegen manche Angriffe sogar resistenter ist, wird plausibel gemacht in:

- M. Jason Hinek, Mo King Low, Edlyn Teske: On some attacks on multi-prime RSA. SAC 2002, 385–404.