7 Asymmetrische Kryptosysteme

Diffie und Hellman kamen 1976 auf die Idee, dass die Geheimhaltung des Chiffrierschlüssels keine notwendige Voraussetzung für die Sicherheit eines Kryptosystems sein muss. Natürlich setzt dies voraus, dass die vom Sender und Empfänger benutzten Schlüssel k und k' voneinander verschieden sind und dass insbesondere der Dechiffrierschlüssel k' nur sehr schwer aus dem Chiffrierschlüssel k berechenbar ist. Ist dies gewährleistet, so kann jedem Kommunikationsteilnehmer X ein Paar von zusammengehörigen Schlüsseln k_X, k'_X zugeteilt werden. X kann nun den Chiffrierschlüssel k_X öffentlich bekannt geben, muss aber den Dechiffrierschlüssel k'_X unter Verschluss halten. Die Tatsache, dass sich mit k_X die Nachrichten nicht wieder entschlüsseln lassen, hat den entscheidenden Vorteil, dass k_X über einen authentisierten Kanal zum Sender gelangen kann (d.h. es ist zwar nicht notwendig, k_X geheim zu halten, aber die Herkunft von k_X muss verifizierbar sein).

- Von einem **symmetrischen Kryptosystem** spricht man, wenn die Kenntnis des Chiffrierschlüssels gleichbedeutend mit der Kenntnis des Dechiffrierschlüssels ist, der eine also leicht aus dem anderen berechnet werden kann.
- Dagegen sind bei einem **asymmetrischen Kryptosystem** nur die Dechiffrierschlüssel geheimzuhalten, während die Chiffrierschlüssel öffentlich bekanntgegeben werden können.

Für symmetrische Kryptosysteme sind auch die Bezeichnungen **konventionelles Kryptosystem**, **Kryptosystem mit geheimen Schlüsseln** oder ***Secret-Key*-Kryptosystem** üblich, wogegen asymmetrische Kryptosysteme auch häufig auch als **Kryptosysteme mit öffentlichen Schlüsseln** oder ***Public-Key*-Kryptosysteme** bezeichnet werden.

Bei einem symmetrischen Kryptosystem sind die Rollen von Sender und Empfänger untereinander austauschbar (beziehungsweise *symmetrisch*), da die Vertraulichkeit der Nachrichten auf einem *gemeinsamen Geheimnis* beruht, welches sich die beiden Kommunikationspartner in Form des zwischen ihnen vereinbarten Schlüssels verschaffen.

Der prinzipielle Unterschied zwischen symmetrischer und asymmetrischer Verschlüsselung kann sehr schön am Beispiel eines Tresors veranschaulicht werden, den Alice dazu verwendet, Bob geheime Dokumente zukommen zu lassen.

Symmetrische Verschlüsselung: Alice und Bob besitzen beide den gleichen Schlüssel k. Alice schließt mit ihrem Schlüssel die Nachricht in den Tresor ein und Bob öffnet ihn später wieder mit seinem Schlüssel. Das Tresorschloss lässt sich also mit k sowohl auf- als auch zuschließen.

Asymmetrische Verschlüsselung: Alice schließt die Nachricht mit dem Schlüssel k_B in den Tresor ein. Danach lässt sich das Tresorschloss mit diesem Schlüssel nicht mehr öffnen. Dies ist nur mit dem in Bobs Besitz befindlichen Schlüssel k'_B möglich. Obwohl also beide Schlüssel in das Schloss passen, können k_B und k'_B jeweils nur in eine von beiden Richtungen gedreht werden.

Da Alice nicht im Besitz von Bobs privatem Schlüssel k'_B ist, kann sie im Gegensatz zu Bob nicht alle mit k_B verschlüsselten Nachrichten entschlüsseln, insbesondere keine Kryptotexte, die Bob von anderen Teilnehmern zugeschickt werden. Zu beachten ist auch, dass mit den beiden Schlüsseln k_B und k'_B von Bob nur eine Nachrichtenübermittlung von Alice an Bob möglich ist. Für die umgekehrte Richtung müssen dagegen die beiden Schlüssel k_A und k'_A von Alice benutzt werden.

7.1 Das RSA-System

Das RSA-Kryptosystem basiert auf dem Faktorisierungsproblem und wurde 1978 von seinen Erfindern Rivest, Shamir und Adleman veröffentlicht. Während beim Primzahlproblem nur eine Ja-Nein-Antwort auf die Frage „Ist n prim?“ gesucht wird, muss ein Algorithmus für das Faktorisierungsproblem im Falle einer zusammengesetzten Zahl mindestens einen nicht-trivialen Faktor berechnen.

Für jeden Teilnehmer des RSA-Kryptosystems werden zwei große Primzahlen p, q und Exponenten e, d mit $ed \equiv_{\varphi(n)} 1$ bestimmt, wobei $n = pq$ und $\varphi(n) = (p-1)(q-1)$ ist.

Öffentlicher Schlüssel: $k = (e, n)$,
Geheimer Schlüssel: $k' = (d, n)$.

Jede Nachricht x wird durch eine Folge $x_1, x_2, \ldots$ von natürlichen Zahlen $x_i < n$ dargestellt, die einzeln wie folgt ver- und entschlüsselt werden.

$$\begin{aligned} E((e,n),x) &= x^e \bmod n, \\ D((d,n),y) &= y^d \bmod n. \end{aligned}$$

Die Chiffrierfunktionen E und D können durch „Wiederholtes Quadrieren und Multiplizieren“ effizient berechnet werden.

Der Schlüsselraum ist also

$$K = \{(c,n) \mid \text{ es ex. Primzahlen } p \text{ und } q \text{ mit } n = pq \text{ und } c \in \mathbb{Z}^*_{\varphi(n)}\}$$

und S enthält alle Schlüsselpaare $((e,n),(d,n)) \in K \times K$ mit $ed \equiv_{\varphi(n)} 1$.

Satz 153 *Für jedes Schlüsselpaar $((e,n),(d,n)) \in S$ und alle $x \in \mathbb{Z}_n$ gilt*

$$x^{ed} \equiv_n x.$$

Beweis: Sei $ed = z\varphi(n) + 1$ für eine natürliche Zahl z.

1. Fall: $\mathrm{ggT}(x,\, n) = 1$: Mit dem Satz von Euler-Fermat folgt unmittelbar

$$x^{ed} \equiv_n x^{z\varphi(n)+1} \equiv_n x.$$

2. Fall: $\mathrm{ggT}(x,\, n) = p$: Dann ist $x \in \mathbb{Z}_q^*$ und es gilt $x = lp \in \mathbb{Z}_q^*$ für eine natürliche Zahl l. Daher folgt

$$x^{ed} = x^{z(p-1)(q-1)}x = (\underbrace{x^{q-1}}_{\equiv_q 1})^{z(p-1)}x \equiv_q x$$

und

$$x^{ed} = (lp)^{ed} \equiv_p 0 \equiv_p x.$$

Somit erhalten wir $x^{ed} \equiv_{pq} x$.

3. Fall: $\mathrm{ggT}(x,\, n) = q$: analog zu Fall 2.
4. Fall: $\mathrm{ggT}(x,\, n) = n$: Dann gilt $x^{ed} \equiv_n 0 \equiv_n x$. ■

Zur praktischen Durchführung

1. Bestimmung von p und q:

   Man beginnt mit einer Zahl x der Form $30z$ (mit $z \in \mathbb{Z}$) und der verlangten Größenordnung (z. B. 10^{100}) und führt einen Primzahltest für $x+1$ durch. Ist die Antwort negativ, testet man der Reihe nach die Zahlen $x+7$, $x+11$, $x+13$, $x+17$, $x+19$, $x+23$, $x+29$, $x+30+1$, $x+30+7$, … bis eine Primzahl gefunden ist. Wegen $\frac{\pi(n)}{n} > \frac{2}{(3\ln n)}$ und da nur 8 von 30 Zahlen getestet werden, sind hierzu ungefähr $\frac{8\cdot 3\cdot \ln x}{30\cdot 2} = {}^2\!/_5 \ln x$ Primzahltests durchzuführen (bei 100-stelligen Dezimalzahlen sind das um die 92 Tests).

2. Bestimmung von d:

   d soll teilerfremd zu $\varphi(n) = (p-1)(q-1)$ sein. Diese Bedingung wird z. B. von jeder Primzahl größer als $\max\{p, q\}$ erfüllt.

3. Bestimmung von e:

   Da $\mathrm{ggT}(d, \varphi(n)) = 1$ ist, liefert der erweiterte Euklidsche Algorithmus das multiplikative Inverse e von d modulo $\varphi(n)$.

4. Ver- und Entschlüsselung:

   Modulares Exponentieren durch wiederholtes Quadrieren und Multiplizieren. Es gibt auch Hardware-Implementierungen, die (unter Verwendung des Chinesischen Restsatzes) mit Geschwindigkeiten von bis zu 225 Kbits/sec arbeiten und somit circa 1500 mal langsamer als der DES sind.

Kryptoanalytische Betrachtungen

1. Die Primfaktoren p und q sollten nicht zu nahe beieinander liegen, da sonst n durch das Verfahren der Differenz der Quadrate faktorisiert werden kann:

Sei $p > q$. Dann gilt mit $b = \frac{(p-q)}{2}$ und $a = p - b = q + b$

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

Um n zu faktorisieren, genügt es daher, eine Zahl $a > \sqrt{n}$ zu finden, so dass $a^2 - n = b^2$ eine Quadratzahl ist.

Für $n = 97\,343$ ist zum Beispiel $\lceil \sqrt{n} \rceil = 312$. Bereits für $a = 312$ ist $a^2 - n = 97\,344 - 97\,343 = 1 = 1^2$ eine Quadratzahl, woraus wir die beiden Faktoren $p = a + 1 = 313$ und $q = a - 1 = 311$ erhalten.

Der Aufwand für die Suche ist proportional zur Differenz $a - \sqrt{n}$, die sich wegen $\sqrt{a+x} \leq \sqrt{a} + \frac{x}{2\sqrt{a}}$ wie folgt nach oben und unten abschätzen lässt:

$$\frac{b^2}{2a} \leq a - \sqrt{a^2 - b^2} = a - \sqrt{n} = \sqrt{n + b^2} - \sqrt{n} \leq \frac{b^2}{2\sqrt{n}}.$$

Im Fall $c = p/q \geq 2$ gilt jedoch

$$\frac{b^2}{2a} = \frac{(c-1)^2 p}{4(c+1)} \geq p/12,$$

so dass dieser Angriff keinen nennenswerten Vorteil mehr bringt.

2. Das kleinste gemeinsame Vielfache $k = \mathrm{kgV}(p-1,\ q-1)$ sollte möglichst groß sein, damit nicht ein zu d äquivalentes $\widetilde{d}$ durch Raten oder durch eine systematische Suche gefunden werden kann. Wie folgende Äquivalenzumformungen zeigen, ist nämlich ein Exponent $\widetilde{d}$ äquivalent zu d, falls $\widetilde{d} \equiv_k d$ ist.

$$\begin{aligned}
& \forall x : x^{e\widetilde{d}} \equiv_n x \\
\Leftrightarrow\ & \forall x : x^{e\widetilde{d}} \equiv_p x \quad \wedge \quad \forall x : x^{e\widetilde{d}} \equiv_q x \\
\Leftrightarrow\ & (p-1) | (e\widetilde{d} - 1) \quad \wedge \quad (q-1) | (e\widetilde{d} - 1) \qquad \text{(Satz von Gauß)} \\
\Leftrightarrow\ & e\widetilde{d} \equiv_{p-1} 1 \wedge \quad e\widetilde{d} \equiv_{q-1} 1 \\
\Leftrightarrow\ & \widetilde{d} \equiv_{p-1} d \wedge \quad \widetilde{d} \equiv_{q-1} d \\
\Leftrightarrow\ & \widetilde{d} \equiv_k d
\end{aligned}$$

Wegen

$$(p-1)(q-1) = \mathrm{kgV}(p-1,\ q-1)\,\mathrm{ggT}(p-1,\ q-1)$$

sollte $\mathrm{ggT}(p-1,\ q-1)$ also relativ klein sein, d. h. $(p-1)$ und $(q-1)$ müssen große, nicht gemeinsam vorkommende Primfaktoren enthalten. Um dies zu erreichen, wählt man zur Bestimmung von p und q jeweils eine große Primzahl $\widehat{p}$ bzw. $\widehat{q}$ und testet die Zahlen $l\widehat{p}+1$, (für $l = 2, 4, 6, \ldots$) mit einem Primzahltest. Hierzu sind gewöhnlich $l/2 \leq \ln \widehat{p}$ Tests auszuführen (nach Elliot und Halberstam).

3. Für unterschiedliche Teilnehmer sollten verschiedene Module $n = pq$ gewählt werden. Wie wir später sehen werden, erlaubt nämlich die Kenntnis eines Schlüsselpaares $(e, n), (d, n)$ mit $ed \equiv_{\varphi(n)} 1$ die effiziente Faktorisierung von n (siehe Satz 157). Zudem lässt sich eine Nachricht x, die ein Benutzer an zwei Empfänger mit demselben Modul n schickt, leicht bestimmen. Gilt nämlich für die öffentlichen Schlüssel (e_1, n) und (e_2, n), dass $\mathrm{ggT}(e_1, e_2) = 1$ ist (die Wahrscheinlichkeit hierfür ist ungefähr $0{,}81$), so kann x leicht aus e_1, e_2, $y_1 = x^{e_1} \bmod n$ und $y_2 = x^{e_2} \bmod n$ berechnet werden:

   1. Fall: $\mathrm{ggT}(y_1, n) > 1$. Dann ist $\mathrm{ggT}(y_1, n) \in \{p, q\}$, d. h. das System ist gebrochen.

   2. Fall: $\mathrm{ggT}(y_1, n) = 1$. Berechne mit dem erweiterten Euklidschen Algorithmus Zahlen a, b, y_1^{-1} mit $ae_1 + be_2 = 1$ und $y_1 y_1^{-1} \equiv_n 1$, wobei o. B. d. A. $a < 0$ sei. Dann ist $y_1^{-1} \equiv_n x^{\varphi(n)-e_1}$, also

$$(y_1^{-1})^{-a} y_2^b \equiv_n (x^{\varphi(n)-e_1})^{-a} x^{e_2 b} \equiv_n x^{e_1 a + e_2 b} \equiv_n x.$$

4. Sei $n = pq$ (mit $p, q \in \mathbb{P}$; $p > q$). Dann können $p,\ q$ leicht aus n und $\varphi(n)$ berechnet werden. Wegen

$$\varphi(n) = (p-1)(q-1) = (p-1)(n/p - 1) = -p + n + 1 - n/p$$

   erhalten wir die Gleichung

$$p - \underbrace{(n + 1 - \varphi(n))}_{c} + n/p = 0,$$

   die auf die quadratische Gleichung $p^2 - cp + n = 0$ führt, aus der sich p zu $\frac{c+\sqrt{c^2-n}}{2}$ bestimmen lässt.

Wie wir gesehen haben, ist das RSA-System gebrochen, falls die Faktorisierung des Moduls n gelingt. Das Brechen von RSA ist daher höchstens so schwer wie das Faktorisieren von n. Dagegen ist nicht bekannt, ob auch umgekehrt ein effizienter Algorithmus, der bei Eingabe von e, n, y ein x mit $x^e \equiv_n y$ berechnet, in einen effizienten Faktorisierungsalgorithmus für n umgewandelt werden kann. D.h., es ist offen, ob das Brechen von RSA mindestens so schwer ist wie das Faktorisieren von n. Wie der folgende Satz zeigt, ist die Bestimmung des geheimen Sclüssels nicht leichter als das Faktorisieren von n, da bei Kenntnis von d leicht ein Vielfaches $v = ed-1$ von $k = \mathrm{kgV}(p-1,\ q-1)$ bestimmt werden kann. Im Beweis benutzen wir folgendes Lemma.

Lemma 154 *Seien y, z zwei Lösungen der Kongruenz $x^2 \equiv_n a$ mit $y \not\equiv_n \pm z$. Dann ist $\mathrm{ggT}(y + z,\ n)$ ein nicht-trivialer Faktor von n.*

Beweis: Wegen $y^2 \equiv_n z^2$ existiert ein $t \in \mathbb{Z}$ mit

$$y^2 - z^2 = (y + z)(y - z) = tn.$$

Da jedoch weder $y + z$ noch $y - z$ durch n teilbar ist, folgt $\mathrm{ggT}(y + z,\ n) \notin \{1,\ n\}$. ■

Betrachte folgenden probabilistischen Algorithmus RSA-FACTORIZE, der durch eine leichte Modifikation des Miller-Rabin Primzahltests entsteht.

Algorithmus 155 MILLERRABIN(n)

```
    sei n − 1 = 2^m u, u ungerade
 1  choose a ∈ {1, 2, ..., n − 1}
 2  if ggT(a, n) > 1 then
 3    output „zusammengesetzt“
 4  end
 5  b ← a^u mod n
 6  s ← 0
 7  while b^2 ≢ 1(mod n) and (s ≤ m) do
 8    b ← b^2 mod n
 9    s ← s + 1
10  end
11  if b ≢ ±1(mod n) then
12    output „zusammengesetzt“
13  else
14    output „prim“
15  end
```

Algorithmus 156 RSA-FACTORIZE(n, v)

```
    sei v = 2^m u, u ungerade
 1  choose a ∈ {1, 2, ..., n − 1}
 2  if ggT(a, n) > 1 then
 3    output ggT(a, n)
 4  end
 5  b ← a^u mod n

 6  while b^2 ≢ 1(mod n) do
 7    b ← b^2 mod n
 8  end

 9  if b ≢ ±1(mod n) then
10    output ggT(b + 1, n)
11  else
12    output „?“
13  end
```

Satz 157 FACTORIZE(n, v) *ist ein Las-Vegas Algorithmus, der bei Eingabe von n, v einen Primfaktor von n mit Wahrscheinlichkeit $> 1/2$ ausgibt, falls $n = pq$ (p, q prim) und $v > 0$ ein Vielfaches von $k = \mathrm{kgV}(p-1, q-1)$ ist.*

Beweis: Mit Lemma 154 folgt

$$b \not\equiv_n \pm 1, b^2 \equiv_n 1 \quad \Rightarrow \quad \mathrm{ggT}(b+1,\ n) \in \{p,\ q\},$$

womit die Korrektheit der Ausgabe von RSA-FACTORIZE in Zeile 10 gezeigt ist. In den folgenden Behauptungen schätzen die Wahrscheinlichkeit ab, mit der dem Algorithmus eine Faktorisierung von n gelingt.

Sei $p - 1 = 2^i u_1$ und $q - 1 = 2^j u_2$ mit u_1, u_2 ungerade und sei o. B. d. A. $i \leq j$.

Behauptung 158 $\mathrm{ggT}(u,\ p-1) = u_1$ *und* $\mathrm{ggT}(u,\ q-1) = u_2$.

Beweis: Wegen $\mathrm{kgV}(p-1, q-1) = 2^{\max(i,j)}\,\mathrm{kgV}(u_1, u_2) | v = 2^m u$ folgt $u_1 | u$, $u_2 | u$ und $i, j \leq m$. Da nun u ungerade ist und $p - 1 = 2^i u_1$ ist, folgt $\mathrm{ggT}(u, p-1) = u_1$ ($\mathrm{ggT}(u, q-1) = u_2$ folgt analog). □

Behauptung 159 $\underbrace{\|\{a \in \mathbb{Z}_n^* \mid a^u \equiv_n 1\}\|}_{=:\alpha} = u_1 u_2$.

Beweis: Mit dem Chinesischen Restsatz folgt

$$\alpha = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^u \equiv_p 1\}\|}_{=:\beta} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^u \equiv_q 1\}\|}_{=:\gamma}.$$

Sei g ein Erzeuger von $\mathbb{Z}_p^*$. Dann gilt

$$g^{ku} \equiv_p 1 \quad \Leftrightarrow \quad ku \equiv_{p-1} 0.$$

Dies zeigt $\beta = \mathrm{ggT}(u,\ p-1) = u_1$. Analog folgt $\gamma = u_2$. □

Behauptung 160 $t \geq i \quad \Rightarrow \quad \forall a : a^{2^t u} \not\equiv_n -1.$

Beweis:

$$\begin{array}{rcccl} t \geq i & \Rightarrow & 2^t u & \equiv_{p-1} & 0 \\ & \Rightarrow & a^{2^t u} & \equiv_p & 1 \\ & \Rightarrow & a^{2^t u} & \not\equiv_p & -1 \\ & \Rightarrow & a^{2^t u} & \not\equiv_n & -1. \end{array}$$

□

Behauptung 161 $0 \leq t < i \quad \Rightarrow \quad \underbrace{\|\{a \in \mathbb{Z}_n^* \mid a^{2^t u} \equiv_n -1\}\|}_{=:\alpha_t} = 2^{2t} u_1 u_2.$

Beweis: Mit dem Chinesischen Restsatz folgt

$$\alpha_t = \underbrace{\|\{a \in \mathbb{Z}_p^* \mid a^{2^t u} \equiv_p -1\}\|}_{=:\beta_t} \cdot \underbrace{\|\{a \in \mathbb{Z}_q^* \mid a^{2^t u} \equiv_q -1\}\|}_{=:\gamma_t}.$$

Sei g ein Erzeuger von $\mathbb{Z}_p^*$. Dann gilt

$$g^{k2^t u} \equiv_p -1 \quad \Leftrightarrow \quad k2^t u \equiv_{p-1} \frac{p-1}{2}.$$

Wegen $t < i$ ist $\mathrm{ggT}(2^t u,\ p-1) = 2^t u_1$ ein Teiler von $\frac{p-1}{2} = 2^{i-1} u_1$ und daher ist $\beta_t = \mathrm{ggT}(2^t u,\ p-1) = 2^t u_1$ ($\gamma_t = 2^t u_2$ folgt analog). □

Behauptung 162 $\|\{a \in \mathbb{Z}_n^* \mid a^u \equiv_n 1 \ \vee \ \exists t, 0 \leq t < i : a^{2^t u} \equiv_n -1\}\| \leq \varphi(n)/2.$

Beweis:

$$\begin{array}{rl} & \|\{a \in \mathbb{Z}_n^* \mid a^u = 1 \ \vee \ \exists t, 0 \leq t < i : a^{2^t u} = -1\}\| \\ \leq & u_1 u_2 + \sum_{t=0}^{i-1} 2^{2t} u_1 u_2 \\ = & u_1 u_2 (1 + \sum_{t=0}^{i-1} 2^{2t}) = u_1 u_2 (1 + \frac{2^{2i}-1}{3}) \\ = & u_1 u_2 (2/3 + 2^{2i}/3) \\ = & 2/3 u_1 u_2 + 1/3 2^{2i} u_1 u_2 \\ \leq & \varphi(n)(1/6 + 1/3), \text{ da } u_1 u_2 \leq \varphi(n)/4 \text{ und } 2^{2i} u_1 u_2 \leq 2^{i+j} u_1 u_2 = \varphi(n) \\ = & \varphi(n)/2. \end{array}$$

■

Bemerkung 163 *Falls es möglich wäre, aus n, e, $y = x^e \bmod n$ die Parität von x in Polynomialzeit zu bestimmen, so könnte auch der gesamte Klartext x in Polynomialzeit aus n, e und y berechnet werden. Das letzte Bit des Klartextes ist also genau so sicher wie der gesamte Klartext: Falls RSA nicht total gebrochen werden kann, kann auch nicht das letzte Bit des Klartextes ermittelt werden.*

Sei nämlich

$$\textit{klartext-half}(y) = \textit{half}(x) = \begin{cases} 0 & \textit{falls } 0 \le x \le (n-1)/2 \\ 1 & \textit{falls } (n+1)/2 \le x \le n-1 \end{cases}$$

und

$$\textit{klartext-parity}(y) = \textit{parity}(x) = \begin{cases} 1 & \textit{falls } x \textit{ ungerade} \\ 0 & \textit{falls } x \textit{ gerade} \end{cases}$$

dann kann wegen $\textit{half}(x) = \textit{parity}(2x \bmod n)$ *die Berechnung von* $\textit{klartext-half}(y)$ *auf die Berechnung von* $\textit{klartext-parity}(y)$ *reduziert werden:*

$$\textit{klartext-half}(y) = \textit{half}(x) = \textit{parity}(2x \bmod n) = \textit{klartext-parity}(2^e y \bmod n).$$

Stellen wir x in der Form

$$\begin{aligned} x &= b_0(n/2) + b_1(n/4) + b_2(n/8) + \cdots \\ &= \sum_{i=0}^{\infty} b_i(n/2^{i+1}) \end{aligned}$$

dar, so berechnet sich die Bitfolge b_i, $i = 0, 1, \ldots$ zu

$$b_i = \textit{half}(2^i x \bmod n) = \textit{parity}(2^{i+1} x \bmod n) = \textit{klartext-parity}(2^{(i+1)e} y \bmod n).$$

Daher lässt sich x mit Orakelfragen an klartext-parity leicht durch eine Intervallschachtelung unter Berechnung der Bits b_i für $i = 0, 1, \ldots, \lfloor \log_2 n \rfloor$ bestimmen (da $\sum_{i=\lfloor \log_2 n \rfloor + 1}^{\infty} (n/2^{i+1}) < n/2^{\log_2 n} = 1$):

```
1   low ← 0
2   high ← n
3   mid ← (high+low)/2
4   while ⌈low⌉ < ⌊high⌋ do
5     y ← 2^e y mod n
6     if klartext-parity(y) then
7       low ← mid
8     else
9       high ← mid
10    end
11    mid ← (low+high)/2
12  end
13  output ⌊high⌋
```

Beispiel 164 *Sei $n = 1\,457$, $e = 779$ und $y = 722$. Angenommen, das Orakel klartext-parity liefert die folgenden Werte $b_i = $ klartext-parity$(2^{(i+1)e} y \bmod n)$:*

i	0	1	2	3	4	5	6	7	8	9	10
$2^{(i+1)e}y$	1136	847	1369	1258	1156	826	444	408	1320	71	144
b_i	1	0	1	0	1	1	1	1	1	0	0

Dann führt die Intervallschachtelung

i	b_i	*low*	*high*	*mid*
-	-	$0,00$	$1\,457,00$	$728,50$
0	1	$728,50$	$1\,457,00$	$1\,092,75$
1	0	$728,50$	$1\,092,75$	$910,63$
2	1	$910,63$	$1\,092,75$	$1\,001,69$
3	0	$910,63$	$1\,001,69$	$956,16$
4	1	$956,16$	$1\,001,69$	$978,93$
5	1	$978,93$	$1\,001,69$	$990,31$
6	1	$990,31$	$1\,001,69$	$996,00$
7	1	$996,00$	$1\,001,69$	$998,85$
8	1	$998,85$	$1\,001,69$	$1\,000,27$
9	0	$998,85$	$1\,000,27$	$999,56$
10	0	$998,85$	$999,56$	$999,21$

auf den Klartext $x = 999$, der tatsächlich die vorgegebene Paritätsbitfolge generiert:

i	0	1	2	3	4	5	6	7	8	9	10
$2^{i+1}x$	541	1082	707	1414	1371	1285	1113	769	81	162	324
b_i	1	0	1	0	1	1	1	1	1	0	0

◁

7.2 Das Rabin-System

Wie das RSA-Verfahren beruht das Rabin-System darauf, dass es zwar effiziente Algorithmen für das Testen der Primzahleigenschaft gibt, effiziente Faktorisierungsalgorithmen aber nicht bekannt sind. Im Gegensatz zum RSA-Verfahren, von dem nicht bekannt ist, dass es mindestens so schwer zu brechen ist wie das Faktorisierungsproblem, ist genau dies beim Rabin-System der Fall. Die Sicherheit des Rabin-Systems ist also äquivalent zur Schwierigkeit des Faktorisierungsproblems. Es verwendet als Falltürfunktion eine quadratische Polynomfunktion modulo dem Produkt $n = pq$ zweier großer Primzahlen. Jeder Teilnehmer wählt ein Paar p, q von Primzahlen mit der Eigenschaft $p \equiv_4 q \equiv_4 3$ und eine Zahl $e \in \{0, \ldots, n-1\}$. Die Zahlen n und e werden öffentlich bekannt gegeben, die Faktorisierung von n wird jedoch geheimgehalten.

Öffentlicher Schlüssel: $e,\ n$
Geheimer Schlüssel: $p,\ q$.

Jede Nachricht m wird in eine Folge von natürlichen Zahlen $m_i \in \{0, \ldots, n-1\}$ aufgeteilt, die der Reihe nach wie folgt verschlüsselt werden.

$$E(x,\ (e,n)) \quad = \quad x(x+e) \bmod n = y.$$

Zur Entschlüsselung eines Kryptotextes $y \in \{0, \ldots, n-1\}$ muss der legale Empfänger B die quadratische Kongruenzgleichung $x(x+e) \equiv_n y$ lösen, die äquivalent zu der Kongruenz

$$(\underbrace{x + 2^{-1}e}_{x'})^2 \equiv_n (2^{-1}e)^2 + y,$$

(*quadratische Ergänzung*) ist, wobei $2^{-1} = (n+1)/2$ das multiplikative Inverse zu 2 modulo n ist. Unter Verwendung des folgenden Satzes können die vier Lösungen x_i' (für $1 \leq i \leq 4$) dieser Kongruenx effizient von B bestimmt werden.

Satz 165 *Sei $n = pq$ für Primzahlen p, q mit $p \equiv_4 q \equiv 3$. Dann besitzt die quadratische Kongruenz $x^2 \equiv_n a$ für jedes $a \in \mathsf{QR}_n$ genau vier Lösungen, wovon nur eine ein quadratischer Rest ist.*

Beweis: Mit $x^2 \equiv_n a$ besitzen wegen $n = pq$ auch die beiden quadratischen Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ Lösungen, und zwar jeweils genau zwei (siehe Korollar 136): $y_{1,2} = a^{(p+1)/4} \bmod p$ und $z_{1,2} = a^{(q+1)/4} \bmod q$. Mit dem Chinesischen Restsatz können wir vier verschiedene Lösungen $x_{i,j}$, $1 \leq 1, 2 \leq 2$ mit

$$\begin{aligned} x &\equiv_p y_i \\ x &\equiv_q z_j \end{aligned}$$

bestimmen. Es können aber auch nicht mehr als diese vier Lösungen existieren, da sich daraus für mindestens eine der beiden Kongruenzen $x^2 \equiv_p a$ und $x^2 \equiv_q a$ mehr als zwei Lösungen ableiten ließen.

Um schließlich zu zeigen, dass genau eine der vier Lösungen ein quadratischer Rest ist, nehmen wir o. B. d. A. an, dass $y_1 \in \mathsf{QR}_p$, $y_2 \in \mathsf{QNR}_p$ und $z_1 \in \mathsf{QR}_q$, $z_2 \in \mathsf{QNR}_q$ sind. Betrachten wir folgendes Lemma als Zwischenschritt.

Lemma 166 *Jeder quadratische Rest a modulo n ist auch quadratischer Rest modulo k, falls k Teiler von n ist.*

$$a \in \mathsf{QR}_n \wedge k|n \quad \Rightarrow \quad a \in \mathsf{QR}_k$$

Beweis:

$$\begin{aligned} a \in \mathsf{QR}_n \quad &\Rightarrow \quad \exists b \in \mathbb{Z}_n^* : b^2 \equiv_n a \\ &\Rightarrow \quad \exists b \in \mathbb{Z}_k^* : b^2 \equiv_k a \\ &\Rightarrow \quad a \in \mathsf{QR}_k. \end{aligned}$$

□

Mit diesem Lemma folgt, daß x_2, x_3, x_4 keine quadratischen Reste modulo n sein können. Weiterhin gibt es Zahlen $l \in \mathbb{Z}_p^*$, $k \in \mathbb{Z}_q^*$ mit $l^2 \equiv_p y_1$ und $k^2 \equiv_q z_1$. Sei m eine Lösung für das System

$$\begin{aligned} x &\equiv_p l, \\ x &\equiv_q k, \end{aligned}$$

dann folgt

$$x_{1,1} \equiv_p y_1 \equiv_p l^2 \equiv_p m^2$$
$$x_{1,1} \equiv_q z_1 \equiv_q k^2 \equiv_q m^2$$

folgt $x_{1,1} \equiv_n m^2$. Also ist $x_{1,1}$ ein quadratischer Rest modulo n. ■

Als weitere zahlentheoretische Funktion mit für die Kryptographie wichtigen Eigenschaften erhalten wir somit die Quadratfunktion $x^2 : \mathsf{QR}_n \to \mathsf{QR}_n$, die nach vorigem Satz bijektiv ist (falls $n = pq$ für Primzahlen p, q mit $p \equiv_4 q \equiv 3$). Ihre Umkehrfunktion $x \mapsto \sqrt{x}$ heißt **diskrete Wurzelfunktion**, und kann (nur) bei Kenntnis der Primfaktoren p und q von n effizient berechnet werden. Wir werden später sehen, dass die Berechnung dieser Funktion äquivalent zur Faktorisierung von n ist. Es ist nicht einmal ein effizientes Verfahren bekannt, mit dem man ohne Kenntnis der Faktorisierung von n für ein gegebenes $a \in \mathbb{Z}_n^*$ entscheiden kann, ob $a \in \mathsf{QR}_n$ ist oder nicht.

Das Problem, aus den vier Kandidaten den richtigen Klartext auzuwählen, kann entweder unter Ausnutzung der Redundanz der Klartextsprache (es muss sich „sinnvoller" Klartext ergeben) oder durch Senden folgender Zusatzinformation gelöst werden. Wegen

$$\left(\frac{-b}{n}\right) = \underbrace{\left(\frac{-b}{p}\right)}_{-\left(\frac{b}{p}\right)} \underbrace{\left(\frac{-b}{q}\right)}_{-\left(\frac{b}{q}\right)} = \left(\frac{b}{n}\right)$$

lässt sich $x' = x + 2^{-1}e$ (und damit x) eindeutig durch Angabe der beiden Bits

$$b_0 = \begin{cases} 0 & \text{falls } \left(\frac{x'}{n}\right) = -1, \\ 1 & \text{sonst} \end{cases} \quad \text{und} \quad b_1 = \text{half}(x') = \begin{cases} 0 & \text{falls } x' \leq (n-1)/2 \\ 1 & \text{sonst} \end{cases}$$

unter den Lösungen von $(x')^2 \equiv_n (2^{-1}e)^2 + y$ kennzeichnen.

Beispiel 167 *Wählen wir $p = 7$, $q = 11$ und $e = 2$, so ergeben sich*

Öffentlicher Schlüssel: $e = 2,\ n = 77$
Geheimer Schlüssel: $p = 7,\ q = 11.$

Um den Klartext $x = 12$ zu verschlüsseln, wird der Kryptotext

$$y = E(12; 2, 77) = 12(12 + 2) \bmod 77 = 14$$

erzeugt. Da $2^{-1}e = 2^{-1} \cdot 2 = 1$ ist, kann dieser durch Lösen der quadratischen Kongruenz

$$(x + 1)^2 \equiv_{77} y + 1 = 15,$$

entschlüsselt werden. Hierzu löst der legale Empfänger zunächst die beiden Kongruenzen

$$y^2 \equiv_7 15 \equiv 1 \quad \textit{und} \quad z^2 \equiv_{11} 15 \equiv 4$$

zu $y_{1,2} = \pm 15^2 \bmod 7 = \pm 1$ (wegen $\frac{p+1}{4} = 2$) und $z_{1,2} = \pm 4^3 \bmod 11 = \pm 2$ (wegen $\frac{q+1}{4} = 3$). Mit dem Chinesischen Restsatz lassen sich $y_{1,2}$ und $z_{1,2}$ zu den vier Lösungen $x'_{i,j} = 57, 64, 13$ und 20 zusammensetzen, die auf die vier Klartextkandidaten $12, 19, 56$ und 63 führen. ◁

Es ist klar, dass das System gebrochen ist, sobald n in seine Primfaktoren p, q zerlegt werden kann. Wie wir gleich sehen werden, sind für Zahlen n von dieser Bauart das Faktorisierungsproblem und das Problem, eine Lösung der quadratischen Kongruenz $x^2 \equiv_n a$ für ein gegebenes $a \in \mathsf{QR}_n$ zu finden, äquivalent. Kann also das Rabin-System gebrochen werden, so muss ein effizienter Algorithmus A existieren, der bei Eingabe (c, n), $c \in \mathsf{QR}_n$, eine Zahl $a = A(c, n)$ mit $a^2 \equiv_n d$ berechnet. Unter Verwendung von A lässt sich folgender effizienter probabilistischer Algorithmus RABIN-FACTORIZE angeben, der n faktorisiert.

Algorithmus 168 RABIN-FACTORIZE(n)

```
1  repeat forever
2    choose a ∈ {1, ..., (n−1)/2}
3    if ggT(a, n) > 1 then output ggT(a, n)
4    c ← a^2 mod n
5    b ← A(c, n)
6    if b ≢_n ±a then output ggT(a + b, n)
```

Satz 169 *Falls* RABIN-FACTORIZE(n) *hält, gibt er einen Primfaktor von $n = pq$, p, q prim, aus. Die Wahrscheinlichkeit, dass hierfür mehr als t Schleifendurchläufe benötigt werden, ist kleiner als 2^{-t}.*

Beweis: Es ist klar, dass $\mathrm{ggT}(a, n)$ im Fall $\mathrm{ggT}(a, n) > 1$ ein Primfaktor von n ist. Lemma 154 zeigt, dass dies im Fall $b \not\equiv_n \pm a$ auch für $\mathrm{ggT}(a + b, n)$ gilt.

Sei nun X die ZV, die die Wahl von a beschreibt, und sei α die Wahrscheinlichkeit, dass der Algorithmus in einem Schleifendurchlauf einen Primfaktor findet. Wir nehmen o.B.d.A. an, dass $A(c, n)$ im Fall $c \in \mathsf{QR}_n$ eine Zahl b in der Menge $U = \mathbb{Z}_n^* \cap \{1, \ldots, \frac{n-1}{2}\}$ zurückliefert. Für $a \in U$ sei a' die in $U - \{a\}$ eindeutig bestimmte Lösung x von $x^2 \equiv_n a^2$. Dann gilt

$$\alpha \;=\; \underbrace{Prob[\mathrm{ggT}(X, n) > 1]}_{Prob[X \notin U] =: \beta} + \underbrace{Prob[X \in U \text{ und } X \not\equiv_n \pm A(X^2, n)]}_{\gamma}.$$

mit

$$\begin{aligned}\gamma \;&=\; \sum_{a \in U} Prob[X \in U \wedge A(X^2, n) = a] \underbrace{Prob[X = a' \mid X \in U \wedge A(X^2, n) = a]}_{1/2} \\ &=\; \frac{1}{2} \sum_{a \in U} Prob[X \in U \wedge A(X^2, n) = a] = \frac{1}{2} Prob[X \in U] = (1 - \beta)/2.\end{aligned}$$

Somit ist $\alpha = \beta + \gamma = (\beta + 1)/2 > 1/2$, d.h. die Wahrscheinlichkeit, dass RABIN-FACTORIZE mehr als t Schleifendurchläufe ausführt, ist $(1 - \alpha)^t < 2^{-t}$. ■