Mitschrift
“Kryptographische Algorithmen”
bei Prof. Schnorr (WS2000/2001)

Michael Jaeger (michael.jaeger@in-flux.de)

17. Oktober 2002($Revision: 1.1 $)

Diese Mitschrift entstand während der Vorbereitung für meine Diplomprüfung in „Kryptographische Algorithmen” und „Algorithmen in der Zahlentheorie”. Für Vollständigkeit oder Korrektheit kann ich nicht garantieren, bitte jedoch um aktive Mithilfe bei der Gestaltung dieses Skriptes. Sollten Fehler gefunden werden, so bitte ich darum, mir diese mitzuteilen, damit ich beheben kann.

Der Inhalt dieses Skriptes basiert auf den handschriftlichen Unterlagen der Vorlesung „Kryptographische Algorithmen” von Prof. Dr. C. P. Schnorr aus dem Wintersemester 2000/2001 und dem Skript „Algorithmen in der Zahlentheorie” von Dr. J. Blömer zu der Vorlesung im Wintersemester 1997/1998.

Mein Dank geht vor allem an das LyX-Team, durch deren Programm dieses Skript erst möglich wurde. Die Software existiert für diverse Plattformen und kann unter der Adresse `http://www.lyx.org` bezogen werden.

Inhaltsverzeichnis

KAPITEL 1

Elementare Zahlentheorie, Primzahltests

1.1. Die Gruppe $\mathbb{Z}_n$

Im Folgenden wird der zentrale Begriff der *Gruppe* eingeführt, der uns im gesamten Text begleiten wird. Ausserdem werden wir die Gruppen $\mathbb{Z}_n$, $\mathbb{Z}_p$ und $\mathbb{Z}_p^*$ kennen lernen, deren Kenntnis für alles Folgende unabdingbar ist.

Abelsche Gruppe

DEFINITION. Ein **abelsche oder kommutative Gruppe** $(G, e, \circ)$ besteht aus einer Menge G, dem neutralen Element $e \in G$ und der Verknüpfung

$$\begin{aligned} \circ : G \times G &\to G \\ (g_1, g_2) &\mapsto g_1 \circ g_2 \end{aligned}$$

mit folgenden Eigenschaften:

(1) Existenz des neutralen Elements: $g \circ e = e \circ g = g \, \forall g \in G$
(2) Kommutativität: $g_1 \circ g_2 = g_2 \circ g_1 \, \forall g_1, g_2 \in G$
(3) Existenz des Inversen: $\forall g \in G \exists h \in G : g \circ h = h \circ g = e$, $h = g^{-1}$
(4) Assoziativität: $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$

Die Gruppe heisst **endlich**, wenn nur endlich viele Elemente in G enthalten sind.

Ist eine Gruppe endlich, so wird die Ordnung eines Elementes wie folgt definiert:

Ordnung

DEFINITION. Sei G endl. Gruppe, $h \in G$. Dann ist die **Ordnung** von h definiert durch

$$\operatorname{ord}(h) = \min \left\{ i \in \mathbb{N} \mid h^i = 1_G \right\}.$$

Zwei spezielle Typen von Gruppen, die im Folgenden eine wichtige Rolle spielen werden, sind die Gruppen $\mathbb{Z}_n$ und $\mathbb{Z}_n^*$ mit $n \in \mathbb{N}$:

Restklassengruppe modulo n

DEFINITION. Sei $n \in \mathbb{N}$. Dann ist

$$\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z} = \{m + n\mathbb{Z} \mid m = 0, \ldots, n-1\} \cong [0, n[$$

die **Restklassengruppe modulo *n*.**

Die Gruppeneigenschaft muss natürlich bewiesen werden, da sie nicht auf den ersten Blick ersichtlich ist. Dazu verdeutliche man sich, dass zwei Äquivalenzklassen entweder identisch oder disjunkt sind. Daraus folgt, dass die Ordnung einer Untergruppe immer die Ordnung der Gruppe teilt (wenn diese endlich ist). Damit teilt also auch die Ordnung jedes Elements die Gruppenordnung.

Zyklische Untergruppen

DEFINITION. Die Untergruppen, die von einem Element erzeugt werden heissen **zyklische Untergruppen**.

LEMMA 1.1.1. *Sei G eine endliche Gruppe und $g \in G$ mit Ordnung m. Falls $d \in \mathbb{N}$ mit $g^d = e$, dann gilt $m \mid d$.*

BEWEIS. Angenommen $r \equiv d \mod m$ ungleich 0 und echt kleiner als m, und $r = d - k \cdot m$ mit $k \in \mathbb{N}$. Dann ist

$$g^r \equiv g^{d-k\cdot m} \equiv g^d \circ g^{-k\cdot m} \equiv e \circ (g^m)^{-k} \equiv e.$$

Das würde bedeuten, dass $r < m$ die Ordnung von g ist. Widerspruch. □

ggT

DEFINITION. Seien $m, n \in \mathbb{Z}$. Die Zahl $d \in \mathbb{N}$ heisst **grösster gemeinsame Teiler (ggT)** von m und n, falls für jede Zahl $k \in \mathbb{Z}$ gilt:

$$k \mid m \text{und} k \mid n \quad \Rightarrow \quad k \mid d$$

Nach dieser Definition ist die Ordnung einer Zahl g in $\mathbb{Z}_n$ mit der Addition gegeben durch

$$\operatorname{ord}(g) = |g| = \frac{n}{\operatorname{ggT}(n,g)},$$

da $\frac{n}{\operatorname{ggT}(n,g)} \cdot g \equiv 0 \mod n$ und jede Zahl, die diese Eigenschaft erfüllt kleiner als $|g|$ ist.

LEMMA 1.1.2. *Jede Untergruppe von $\mathbb{Z}_n$ wird von ihrem kleinsten Element zyklisch erzeugt.*

BEWEIS. Sei g das kleinste Element aus $U \subseteq \mathbb{Z}_n$. Angenommen, es existiert ein $u \in U$, das kein Vielfaches von g ist. Dann gilt

$$u = gq + r \quad \text{mit} \quad 0 < r < g \leq n - 1.$$

Da gq und r aber in der gleichen Untergruppe liegen, widerspricht das der Annahme, dass g das kleinste Element in U ist. □

1.2. Die Gruppe $\mathbb{Z}_n^*$

Eine Untergruppe von $\mathbb{Z}_n$ ist $\mathbb{Z}_n^*$. Diese Gruppe zählt zu den für uns wichtigsten Gruppen.

Multiplikative Gruppe, Einheitengruppe

DEFINITION. Die Untergruppe $\mathbb{Z}_n^* \subseteq \mathbb{Z}_n$ heisst die **multiplikative Gruppe** (**Einheitengruppe**) mit

$$[0,n[\cong \mathbb{Z}_n \supseteq \mathbb{Z}_n^* \quad \cong \quad \{a \in [0,n[\mid \operatorname{ggT}(a,n) = 1\}.$$

Eulerfunktion

Wir nennen $\varphi(n) = |\mathbb{Z}_n^*|$ die **Eulerfunktion**. Sie gibt an, wieviele Elemente in der multiplikativen Gruppe enthalten sind.

Zum Beweis der Gruppeneigenschaften wird folgendes Lemma benötigt:

LEMMA 1.2.1. *Seien $m, n \in \mathbb{Z}$ mit $m \neq n$, dann gilt*

$$ggT(n,m) = ggT(n \mod m, m) = ggT(m \mod n, n)$$

BEWEIS. Wir werden zeigen $\operatorname{ggT}(n,m) = \operatorname{ggT}(n \mod m, m)$, die andere Gleichheit wird genauso bewiesen.

1. Fall: $m > n$
Dann ist $n = n \mod m$ und damit $\operatorname{ggT}(n,m) = \operatorname{ggT}(n \mod m, m)$.

2. Fall: $m < n$
Sei nun $n = s \cdot d$ und $m = t \cdot d$ mit $d = \text{ggT}(n,m)$ und $\text{ggT}(s,t) = 1$ und $t < s$. Dann gilt

$$\begin{aligned} \text{ggT}(n,m) &= \text{ggT}(sd,td) \\ &= \text{ggT}(sd \mod td, td) \\ &= \text{ggT}((s-t)d, td) \\ &= d. \end{aligned}$$

□

Jetzt können wir die Gruppeneigenschaften von $\mathbb{Z}_n^*$ beweisen:

BEWEIS. *Abgeschlossenheit:* Wenn nun die Primzahl p das Produkt zweier Zahlen a und b teilt, so teilt sie auch a oder b. Das impliziert, dass aus $\text{ggT}(ab,n) \neq 1$ folgt, dass $\text{ggT}(a,n) \neq 1$ oder $\text{ggT}(b,n) \neq 1$ ist.

Neutrales Element: $e = 1$.

Existenz des Inversen: Sei $g \in \mathbb{Z}_n^*$, dann sind je zwei Elemente der Menge $\{ag \mod n \mid a \in \mathbb{Z}_n^*\}$ verschieden (ansonsten wäre $ag \equiv bg \mod n$ mit $b < a$ und damit $g(a-b) \equiv 0 \mod n$, dann $n \mid g(a-b)$ und weil $\text{ggT}(g,n) = 1$ gilt $n \mid (a-b)$, was aber ein Widerspruch zu $a - b < n$ ist). Also liegen in der Menge $\{ag \mod n \mid a \in \mathbb{Z}_n^*\}$ genauso viele Elemente wie in $\mathbb{Z}_n^*$. Gleichzeitig ist diese Menge aber auch eine Teilmenge von $\mathbb{Z}_n^*$, woraus die Gleichheit der Mengen folgt. Es muss also ein $a \in \mathbb{Z}_n^*$ geben mit $ag \equiv 1 \mod n$, was das gesuchte Inverse Element zu g ist. □

In diesem Zusammenhang möchten wir Aussagen über die Ordnung von $\mathbb{Z}_n^*$ treffen, was ja bei $\mathbb{Z}_n$ sehr einfach war.

PROPOSITION 1.2.2. *Sei $n = \Pi_{i=1}^r p_i^{e_i}$ mit $p_1, \ldots, p_r$ paarweise verschieden und prim, und $e_1, \ldots, e_r \geq 1$. Dann gilt*

$$\varphi(n) = \prod_{i=1}^{n} \varphi\left(p_i^{e_i}\right) \quad \textit{mit} \quad \varphi\left(p_i^{e_i}\right) = p_i^{e_i} - p_i^{e_i - 1}.$$

BEWEIS. Siehe Seite 8. □

Die Ordnung der Restklassengruppe einer zusammengesetzten Zahl ist also gleich dem Produkt der Ordnungen der Primfaktorpotenzen.

Im allgemeinen hat $\mathbb{Z}_n^*$ nicht die gleiche, einfache Struktur wie die Gruppe $\mathbb{Z}_n$, die ja immer eine zyklische Gruppe ist. Es gibt jedoch Spezialfälle, in denen die beiden Gruppen eine ähnliche Struktur aufweisen:

LEMMA 1.2.3. *Sei $n \in \mathbb{N}$. Die Gruppe $\mathbb{Z}_n^*$ ist genau dann zyklisch, wenn $n = 2, 4$ oder aber $n = p^e$ oder $n = 2p^e$ ist, mit $p \in \mathbb{P}$ und $p > 2$ und $e \in \mathbb{N}$.*

In diesen Spezialfällen ist $\mathbb{Z}_n^*$ also zyklisch und es gilt für ein beliebiges $a = g^d \in \mathbb{Z}_n^*$, wobei g das erzeugende Element von $\mathbb{Z}_n^*$ ist nach der Definition des ggT auf Seite 6

$$ord_n(a) = \frac{\varphi(n)}{ggT(\varphi(n), d)}.$$

Für $\mathbb{Z}_p^*$ gibt es deshalb genau $\varphi(p-1)$ erzeugende Elemente, da wenn wir ein erzeugendes Element g besitzen, sich alle anderen erzeugenden Elemente durch g^d bilden lassen mit

$0 < d < p-1$ *und* $ggT(d, p-1) = 1$*, da nur dann gewährleistet ist, dass erst nach dem* $p-1$*-ten Potenzieren ein Vielfaches von p vorliegt:*

$$g^{p-1} \equiv 0 \mod p \quad \Rightarrow \quad \left(g^d\right)^{p-1} \equiv 0 \mod p.$$

Chinesischer Restsatz

PROPOSITION 1.2.4. ***(Chinesischer Restsatz)*** *Sei* $m = m_1 \cdot \ldots \cdot m_r$*, mit* $ggT\left(m_i, m_j\right) = 1 \forall i \neq j$ *dann gelten folgende Isomorphismen:*

$$\mathbb{Z}_m \overset{\textbf{Ring-Iso}}{\cong} \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \times \mathbb{Z}_{m_r}$$

und

$$\begin{array}{rccccccc}
\psi: & \mathbb{Z}_m & \to & \mathbb{Z}_{m_1} \times & \ldots & \times \mathbb{Z}_{m_r} \\
& \cong & & \cong & & \cong \\
& [0,m[& & [0,m_1[& & [0,m_r[\\
& \ni & & \ni & & \ni \\
& a & \mapsto & (a \mod m_1, & \ldots & ,a \mod m_r) \\
& & & = & & = \\
& & & (a_1, & \ldots & a_r)
\end{array}$$

Es gilt also[1]

$$\begin{array}{ccccccc}
\mathbb{Z}_m & \overset{\text{Ring-Iso.}}{\cong} & \mathbb{Z}_{m_1} \times & \mathbb{Z}_{m_2} \times & \ldots & \times \mathbb{Z}_{m_r} \\
\cup & & & & & \\
\mathbb{Z}_m^* & \underset{\text{Gruppen-Iso.}}{\cong} & \mathbb{Z}_{m_1}^* \times & \mathbb{Z}_{m_2}^* \times & \ldots & \times \mathbb{Z}_{m_r}^*
\end{array}$$

Für beliebige Zahlen $a_1, \ldots, a_k$ bedeutet das, dass das folgende System von Kongruenzen eine Lösung besitzt, die modulo $\prod_{i=1}^k m_i$ eindeutig ist:

$$\begin{array}{rcl}
x & \equiv & a_1 \mod m_1 \\
& \vdots & \\
x & \equiv & a_k \mod m_k
\end{array}$$

Damit läßt sich Satz 1.2.2 beweisen:

BEWEIS. Zunächst soll gezeigt werden, dass $\varphi(p^e) = p^e - p^{e-1}$ gilt. Hierzu überlege man sich, dass nur die Zahlen kleiner p^e einen ggT ungleich 1 besitzen, die ein Vielfaches von p als Primfaktor besitzen (und implizit kleiner als p^e sind). Davon gibt es genau p^{e-1} Zahlen, nämlich genau die Zahlen aus $\left\{i \cdot p \mid 0 \leq i < p^{e-1}\right\}$.

Für ein beliebiges $n \in \mathbb{N}$ sei die Primfaktorzerlegung $n = \prod_{i=1}^k p_i^{e_i}$ gegeben. Sei $a \in \mathbb{Z}_n^*$, dann ist für jedes $i = 1, \ldots, k$ die Zahl $a_i \equiv a \mod p_i^{e_i}$ ein Element aus $\mathbb{Z}_{p_i^{e_i}}^*$. Jedem $a \in \mathbb{Z}_n^*$ kann also ein k-Tupel $\left(a_1, \ldots, a_k\right) \in \mathbb{Z}_{p_1^{e_1}}^* \times \ldots \times \mathbb{Z}_{p_k^{e_k}}^*$ zugeordnet werden.

Nach dem Chinesischen Restsatz entspricht aber jedes dieser Tupel genau einem $a \in \mathbb{Z}_n$, wenn man als Module $m_i = p_i^{e_i}$ wählt. Dieses a muss zwangsläufig in $\mathbb{Z}_n^*$ liegen, da ansonsten a mit mindestens einem $p_i^{e_i}$ nicht teilerfremd wäre. Widerspruch zur Voraussetzung $a \in \mathbb{Z}_n^*$.

[1] Zur Erinnerung: Ein **Ring** ist eine Menge mit 2 Verknüpfungen „+" und „·", wobei $(R, +)$ eine abelsche Gruppe ist und das Distributiv- und das Assoziativgesetz gelten.

Für jeden Modul $m_i = p_i^{e_i}$ gibt es also $\varphi(m_i) = p_i^{e_i} - p_i^{e_i-1}$ verschiedene Möglichkeiten, um eine Zahl aus $\mathbb{Z}_n^*$ zu erzeugen. Damit gilt $\varphi(n) = \prod_{i=1}^{k} \left(p_i^{e_i} - p_i^{e_i-1}\right)$. □

Desweiteren kann mit Hilfe des CRT gezeigt werden, dass $\mathbb{Z}_n^*$ nicht zyklisch sein kann, falls $n = pq$ ist, wobei $p \neq 2$ und $q \neq 2$ verschiedene Primzahlen sind. Wäre das nämlich der Fall, so gäbe es ein Element in $\mathbb{Z}_n^*$ der Ordnung $\varphi(n)$ und damit der Ordnung $\varphi(n) = (p-1)(q-1)$. Im Folgenden werden wir aber zeigen, dass bereits $a^{(p-1)(q-1)/2} \equiv 1 \mod n$ ist für alle $a \in \mathbb{Z}_n^*$:

Sei $a_p \in \mathbb{Z}_p^*$ mit $a_p \equiv a \mod p$ und $a_q \in \mathbb{Z}_q^*$ mit $a_q \equiv a \mod q$. Damit gilt

$$\begin{aligned} a_p^{(p-1)(q-1)/2} &\equiv a^{(p-1)(q-1)/2} \mod p \\ a_q^{(p-1)(q-1)/2} &\equiv a^{(p-1)(q-1)/2} \mod q. \end{aligned}$$

Nach dem CRT ist $a^{(p-1)(q-1)/2} \equiv 1 \mod n$ genau dann, wenn

$$\begin{aligned} a_p^{(p-1)(q-1)/2} &\equiv 1 \mod p \\ a_q^{(p-1)(q-1)/2} &\equiv 1 \mod q. \end{aligned}$$

Da $\varphi(p) = p-1$ ist (für $p \in \mathbb{P}$) und $q-1$ gerade und ≥ 2 ist, gilt nach Fermat (Satz 1.2.6)

$$a_p^{(p-1)(q-1)/2} \equiv \left(a_p^{p-1}\right)^{(q-1)/2} \equiv 1^{(q-1)/2} \equiv 1 \mod p.$$

Für q wird der Beweis analog geführt. $\mathbb{Z}_n^*$ ist also nicht unbedingt zyklisch.

Die Tatsache, dass die Untergruppen einer Gruppe immer entweder disjunkt oder gleich sind, wird in dem Beweis des folgenden Satzes verwendet:

PROPOSITION 1.2.5. ***(Lagrange)*** $\forall h \in G : ord(h) \mid |G|$.

Das führt zum Satz von Fermat-Legendre:

PROPOSITION 1.2.6. ***(Fermat, Legendre)*** *Für $n \in \mathbb{N}$ gilt:*

$$a^{\varphi(n)} \equiv 1 \mod n \left(= 1_{\mathbb{Z}_n^*}\right) \quad \forall a : ggT(a,n) = 1$$

*Nach **Fermat** gilt dann*

$$p \in \mathbb{P} : \quad a^{p-1} \equiv 1 \mod p \quad \forall a : p \nmid a$$

Nach Definition der Ordnung gilt: $h^{ord(h)} = 1_G$ $(h \in G) \Rightarrow h^{|G|} = 1_G$ (Dieser Satz ist eine Folgerung aus dem Satz von Lagrange, dafür aber älter als dieser).

Der Satz besagt also, dass ein Element potenziert mit der Gruppenordnung immer das Einselement ergibt. Für prime p gilt damit für jedes Element $a \in \mathbb{Z}_p$: $a^{p-1} \equiv 1 \mod p$. Es gibt allerdings auch Zahlen n, die nicht prim sind und trotzdem diese Äquivalenz erfüllen, die sogenannten Carmichaelzahlen:

Charmichaelzahl

DEFINITION. Eine zusammengesetzte Zahl $n \in \mathbb{N}$ heisst **Carmichaelzahl**, wenn für alle $a \in \mathbb{Z}_n^*$ gilt $a^{n-1} \equiv 1 \mod n$.

Dazu wird eine spezielle Funktion eingeführt:

Carmichael Funktion

DEFINITION. Die **Carmichael Funktion** $\lambda(n)$ ist definiert durch

$$\lambda(n) = \max\{\text{ord}(a) \mid a \in \mathbb{Z}_n^*\}.$$

Sie beschreibt also die Anzahl der Elemente der größten zyklischen Untergruppe von $\mathbb{Z}_n^*$. Ist $\mathbb{Z}_n^*$ selbst schon eine zyklische Gruppe, so gilt $\lambda(n) = \varphi(n)$. Nach Lagrange gilt: $\lambda(n) \mid \varphi(n)$, da die Ordnung einer Untergruppe immer die Ordnung der Gruppe selbst teilt. Ist $\lambda(n) = \varphi(n)$, so handelt es sich bei $\mathbb{Z}_n^*$ um eine zyklische Gruppe, da es offensichtlich ein Element gibt, das die Gruppe erzeugt.

PROPOSITION 1.2.7. *Sei G endliche Gruppe, dann gilt*

$$kgV(ord(a) \mid a \in G) = \max\{ord(a) \mid a \in G\}$$

BEWEIS. Siehe [**Schnorr**, Kor. 90]. □

PROPOSITION 1.2.8. *Es gelten folgende Gleichungen:*

$$\begin{array}{lcll} \lambda(p^e) & = & \varphi(p^e) = p^e - p^{e-1} = p^{e-1}(p-1) & \textit{fuer } p \neq 2 \\ \lambda(2^e) & = & 2^{e-2} & \textit{fuer } e \geq 3 \ (\textit{im Fall } e = 2 \textit{ s.o.}) \end{array}$$

BEWEIS. Zum Beweis muss man wissen, dass $\mathbb{Z}_{p^e}^*$ zyklisch ist (siehe Korollar 1.2.9). In diesem Fall ist die größte zyklische Untergruppe von $\mathbb{Z}_{p^e}^*$ die Gruppe selber und die obere Gleichung ist erfüllt. Siehe auch [**Wohlfahrt**, Seite 93]. □

COROLLARY 1.2.9. *Sei p prim, $p \neq 2$, dann ist $\mathbb{Z}_{p^e}^*$ **zyklisch**, d.h.*

$$\exists g: \quad \mathbb{Z}_{p^e}^* = \left\{g, g^2, \ldots, g^{\varphi(p^e)} \equiv 1 \mod p^e\right\} = \langle g \rangle \quad \textit{mit}: ord(g) = \varphi(p^e)$$

(g ist also der "Generator" der Gruppe).

BEWEIS. Da $\mathbb{Z}_p$ zyklisch ist, kann man ein erzeugendes Element g für $\mathbb{Z}_p$ wählen. Wir werden jetzt zeigen, dass man das g sogar so wählen kann, dass $\mathrm{ord}_{p^e}(g) = (p-1)p^{e-1} = \left|\mathbb{Z}_{p^e}^*\right|$ ist: Angenommen, für g gilt $g^{p-1} \equiv 1 \mod p^2$, dann ist $g+p$ immer noch erzeugendes Element von $\mathbb{Z}_p$, aber nun mit

$$(g+p)^{p-1} \equiv g^{p-1} + (p-1)g^{p-2}p \equiv 1 + ap \mod p^2, \quad a \not\equiv 0 \mod p,$$

denn es gilt $g^{p-2} \not\equiv 0 \mod p$. Sei o.B.d.A. g eine Primitivwurzel von $\mod p$ mit $g^{p-1} \equiv 1 + ap \mod p^2$ und $a \not\equiv 0 \mod p$. Dann gilt nach [**Wohlfahrt**, Hilfssatz 4.4.2]

$$g^{(p-1)p^{e-2}} \equiv 1 + ap^{e-1} \not\equiv 1 \mod p^e,$$

und für alle anderen Teiler d mit $d \mid (p-1)p^{e-2}$ gilt erst recht $g^d \not\equiv 1 \mod p^e$ und deshalb $\mathrm{ord}_{p^e}(g) = (p-1)p^{e-1}$. □

Für zusammengesetzte Zahlen gilt:

COROLLARY 1.2.10. *Mit $p_i \neq 2$ für $i = 1, \ldots, r$ gilt:*

$$\lambda\left(\prod_{i=1}^r p_i^{e_i}\right) = kgV_{i=1,\ldots,r}\lambda\left(p_i^{e_i}\right) = \prod_{i=1}^r p_i^{e_i-1} kgV\left(p_1 - 1, \ldots, p_r - 1\right).$$

BEWEIS. Nach dem CRT ist

$$a^{\mathrm{kgV}} = \left(a_1^{\mathrm{kgV}}, \ldots, a_r^{\mathrm{kgV}}\right) = (1, \ldots, 1) = 1,$$

also ist $\mathrm{kgV}_n(a) \leq \mathrm{kgV}$. Umgekehrt ist aber $a_i^{\mathrm{ord}_n(a)} = 1 \mod p_i^{e_i}$, also $\mathrm{ord}_{p_i^{e_i}}(a) \mid \mathrm{ord}_n(a)$. Daraus folgt, dass $\mathrm{ord}_n(a) \geq \mathrm{kgV}$. Damit ist die Gleicheit bewiesen. □

1.3. Der Euklidische Algorithmus

Der Euklidische Algorithmus beschreibt ein elementares Verfahren für die Berechnung des ggT zweier Zahlen.

Algorithm 1 Euklidischer Algorithmus

EINGABE: $m,n \in \mathbb{N}$
AUSGABE: $\mathrm{ggT}(m,n)$

(1) $a := m, b := n, r := 0$
(2) **if** $a < 0$ **then**
  (a) $a := -a$
(3) **if** $b < 0$ **then**
  (a) $b := -b$
(4) **while** $b \neq 0$ **then**
  (a) $r := b$
  (b) $b := a \mod r$
  (c) $a := r$
(5) **else**
  (a) **return** a

Die Korrektheit beruht auf der Aussage von Lemma 1.2.1 auf Seite 6:

$$\mathrm{ggT}(m,n) = \mathrm{ggT}(n, m \mod n)$$

Die Laufzeit besitzt unter der Annahme, dass $|n| > |m|$ gilt $O\left(\log^2(|n|)\right)$-viele Bitoperationen.

Durch eine kleine Erweiterung kann der euklidische Algorithmus dazu benutzt werden, um den ggT als Linearkombination von m und n darstellen zu können:

Algorithm 2 Erweiterter Euklidischer Algoritmus

EINGABE: $m,n \in \mathbb{Z}$
AUSGABE: $\mathrm{ggT}(m,n)$ sowie $s,t \in \mathbb{Z}$ mit $\mathrm{ggT}(m,n) = sm + tn$

(1) $a := m, b := n, s_0 := 1, s_1 := 0, t_0 := 0, t_1 := 1, r := 0, q := 0, u := 0, v := 0$
(2) **if** $a < 0$ **then**
  (a) $a := -a, s_0 := -1$
(3) **if** $b < 0$ **then**
  (a) $b := -b, t_1 := -1$
(4) **while** $b \neq 0$ **then**
  (a) $r := b, b := a \mod r, q := a/r, a := r$
  (b) $u := s_1, v := t_1, s_1 := s_0 - qs_1, t_1 := t_0 - qt_1, s_0 := u, t_0 := v$
(5) **else**
  (a) **return** $\mathrm{ggT}(m,n) = a, s = s_0, t = t_0$

Zu Überlegungen zur Korrektheit mache man sich klar, dass zu Anfang gilt

$$\begin{aligned} s_0 m + t_0 n &= a \\ s_1 m + t_1 n &= b. \end{aligned}$$

Im weiteren Verlauf des Algorithmus werden nun alle Änderungen, die an a und b vorgenommen werden auch auf $s_0, s_1, t_0 t_1$ angewendet. Dadurch ist die Laufzeit - abgesehen von einem konstanten Faktor - identisch zu der des einfachen Euklidischen Algorithmus.

Besonders interessant ist dieser Algorithmus, da er für zwei Zahlen, deren ggT gleich 1 ist das Inverse der einen Zahl modulo der anderen berechnet:

$$\begin{aligned} sa + tn &= 1 \\ \Rightarrow \quad sa &\equiv 1 \mod n \end{aligned}$$

s ist also das Inverse zu a modulo n. Es gilt also:

COROLLARY. *Sei $a \in \mathbb{Z}_n^*$. Das Inverse von a in $\mathbb{Z}_n^*$ kann in Zeit $O\left(\log^2(n)\right)$ berechnet werden.*

1.4. Kryptographie

In diesem Abschnitt sollen einige kryptographische Verfahren zusammen mit den ihnen zugrunde liegenden Sicherheitsmechanismen vorgestellt werden und somit eine Motivation für das Kapitel „Faktorisierung" gegeben werden.

Kryptographisches System

DEFINITION. Ein **kryptographisches System** besteht aus einer Menge K von Schlüsseln. Zu jedem Schlüssel $k \in K$ gibt es

(1) Die Menge P_k aller Nachrichten, die mit Hilfe des Schlüssels k verschlüsselt werden können.
(2) Die Menge C_k aller möglichen Verschlüsselungen.
(3) Eine Verschlüsselungsregel e_k und eine Entschlüsselungsregel d_k

$$\begin{aligned} e_k : P_k &\longrightarrow C_k \\ d_k : C_k &\longrightarrow P_k, \end{aligned}$$

so dass für alle $x \in P$ gilt: $d_k\left(e_k(x)\right) = x$.

Ein solches kryptographisches System wird **sicher** genannt, wenn ein Angreifer aus einer verschlüsselten Nachricht weder den geheimen Schlüssel noch den Klartext effizient ermitteln kann – auch wenn ihm das verwendete kryptographische System bekannt ist.

EXAMPLE. **(DES – Data Encryption Standard)**
Bei diesem System gilt

$$\begin{aligned} K &= \{0,1\}^{56} \\ P_k = C_k &= \{0,1\}^{64} \end{aligned}$$

Dabei werden die Nachrichten in 64-Bit-Folgen (x_i) aufgeteilt, die mit der bekannten Permutation IP erzeugt werden. Diese werden dann wie folgt bearbeitet:

(1) Aufspaltung der 64-Bit-Folge in zwei 32-Bit-Folgen $x_0 = L_0 R_0$
(2) $L_i, R_i \in \{0,1\}^{32}$ mit $i = 1, \ldots, 16$ werden berechnet:

$$L_i = R_{i-1},\ R_i = L_{i-1} \oplus f\left(R_{i-1}, K_{i-1}\right)$$

wobei f eine allgemein bekannte Funktion ist und die $K_i \in \{0,1\}^{46}$ nach einem öffentlichen Verfahren aus k berechnet werden.

(3) Zuletzt wird das Inverse der Permutation IP auf $R_{16}L_{16}$ angewandt, so dass die Verschlüsselung y von x entsteht

$$y = IP^{-1}\left(R_{16}L_{16}\right)$$

Bei diesen sogenannten **symmetrischen Verfahren** stellt der Austausch eines geheimen Schlüssels über einen unsicheren Kanal das Hauptproblem dar. Ein Verfahren, um dieses Problem zu lösen wird im Folgenden vorgestellt:

Algorithm 3 Diffie-Hellman Verfahren

ANWENDUNG: Sicherer Schlüsseltausch über einen unsicheren Kanal.

(1) A und B wählen eine Primzahl p und ein erzeugendes Element g der Gruppe $\mathbb{Z}_p^*$. *[öffentlich]*
(2) A wählt $r_A \in_R [1, p-2]$ *[geheim]* und sendet g^{r_A} an B *[öffentlich]*
(3) B wählt $r_B \in_R [1, p-2]$ *[geheim]* und sendet g^{r_B} an A *[öffentlich]*
(4) Der geheime Schlüssel ist nun $g^{r_A r_B}$.

Die Sicherheit dieses Verfahrens wurde noch nicht bewiesen, jedoch ist die Berechnung von $g^{r_A r_B}$ nach Kenntnis von g^{r_A} und g^{r_B} für einen Angreifer ähnlich schwer wie die Berechnung des diskreten Logarithmus:

Diskreter Logarithmus

DEFINITION. **(Diskreter Logarithmus)** Sei p prim, g ein erzeugendes Element von $\mathbb{Z}_p^*$ und $a \in \mathbb{Z}_p^*$. Gesucht ist $e \in \mathbb{Z}_{p-1}$ mit $g^e \equiv a \mod p$.

Der Zusammenhang zwischen dem Diskreten Logarithmus und dem Diffie-Hellman-Verfahren erklärt folgender Korollar:

COROLLARY 1.4.1. *Falls der diskrete Logarithmus in polynomieller Zeit berechnet werden kann, so kann aus Kenntnis von g^{r_A} und g^{r_B} der geheime Schlüssel $g^{r_A r_B}$ berechnet werden.*

Neben den symmetrischen Verschlüsselungsverfahren, wie dem DES gibt es noch die Klasse der asymmetrischen Verfahren, deren prominentester Vertreter das RSA-Verfahren ist.

Die Sicherheit dieses Verfahrens basiert darauf, dass es nicht effizient möglich ist, aus b und n die Zahl a mit $ab \equiv 1 \mod \varphi(n)$ zu berechnen. Dieses Problem hängt eng mit dem der Faktorisierung zusammen. Sind nämlich die Primfaktoren p, q von n bekannt, so kann mit Hilfe des Erweiterten Euklidischen Algorithmus auch a berechnet werden und dann ist auch $\varphi(n) = (p-1)(q-1)$ bekannt. Damit kann a aus $ab \equiv 1 \mod \varphi(n)$ wie gehabt berechnet werden.

Genauso kann bei Kenntnis von $\varphi(n)$ die Faktorisierung $n = p \cdot q$ berechnet werden:

$$\begin{aligned} \varphi(n) &= (p-1)(q-1) \\ n &= pq. \end{aligned}$$

Bleibt das Lösen der quadratischen Gleichung

$$p^2 - (n - \varphi(n) + 1)\, p + n = 0,$$

was in polynomieller Zeit machbar ist.

Ein weiteres asymmetrisches Verschlüsselungsverfahren ist das ElGamal-Verfahren.

Algorithm 4 Rivest-Shamir-Adleman Schema (RSA)

ANWENDUNG: Verwendung von zwei Schlüsseln pro Benutzer: ein öffentlicher zum Verschlüsseln und ein geheimer zum Entschlüsseln.
Ein Schlüssel k hat die Form

$$k = \left\{ (n,p,q,a,b) \in \mathbb{N}^5 \mid p,q \text{sind zwei Primzahlen } n = pq, ab \equiv 1 \mod \varphi(n) \right\}$$

Die Schlüsselerzeugung des *geheimen* Schlüssels a und des *öffentlichen* Schlüssels b funktioniert folgendermassen:

(1) Berechne $n = p \cdot q$ mit $p,q \in \mathbb{P}$ und wähle ein $a \in \mathbb{N}$ mit $\mathrm{ggT}(a, \varphi(n)) = 1$
(2) Berechne ein b mit $ab \equiv 1 \mod \varphi(n)$

Die Verschlüsselung ist dann

$$e_k(x) \equiv x^b \mod n$$

und die Entschlüsselung

$$d_k(y) \equiv y^a \equiv \left(x^b\right)^a \equiv x^{ab} \equiv x^{k\varphi(n)+1} \equiv x^{k\varphi(n)} x \equiv x \mod n$$

Es werden also n und b veröffentlicht, während $p,q,a,\varphi(n)$ geheim gehalten werden.

Algorithm 5 ElGamal-Verfahren

ANWENDUNG: Asymmetrische Verschlüsselung. Dabei hat ein Schlüssel k die Form

$$k = \left\{ (p,g,h,e) \in \mathbb{N} \times \mathbb{Z}_p^* \times \mathbb{Z}_p^* \times \mathbb{Z}_{p-1} \mid p \text{ ist prim, } g \text{ erzeugt } \mathbb{Z}_p^*, h \equiv g^e \mod p \right\}$$

Ein Schlüssel wird durch Wahl einer Primzahl p und eines erzeugenden Elements g für $\mathbb{Z}_p^*$ erzeugt. Dazu wird aus einem $e \in \mathbb{N}$ die Zahl $h \equiv g^e \mod p$ berechnet. Die Zahlen p,g,h sind öffentlich, der Exponent e bleibt geheim. Es gilt $P_k = \mathbb{Z}_p^*, C_k = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$.
Zum Verschlüsseln wird ein $k \in_R \mathbb{Z}_{p-1}$ gewählt und

$$e_k(x,k) = \left(g^k \mod p, xh^k \mod p\right) = (y_1,y_2)$$

berechnet.
Die Entschlüsselungsfunktion ist dann

$$d_k(y_1,y_2) \equiv y_2 (y_1^e)^{-1} \mod p$$

Wie das Verfahren von Diffie-Hellman hängt auch dieses eng mit dem Problem des Diskreten Logarithmus zusammen. Genaugenommen sind beide Verfahren gleich schwer/sicher.

Für all diese Verfahren müssen effizient Primzahlen und erzeugende Elemente bestimmt werden, womit sich der nächste Abschnitt beschäftigt.

KAPITEL 2

Primzahltests

2.1. Der φ-Test

Zu Beginn ein sehr intuitiver Primzahltest:

Algorithm 6 Der φ-Test

EINGABE: $n \in \mathbb{N}$ ungerade.
AUSGABE: n „zusammengesetzt" oder „prim".

(1) Wähle $a \in_R \mathbb{Z}_n$ mit $a \not\equiv 0 \mod n$.
(2) **if** $\mathrm{ggT}(a,n) \neq 1$ **then**
  (a) **return** „zusammengesetzt".
(3) **else**
  (a) **return** „prim".

Bei Primzahlen gibt der φ-Test immer „prim" aus. Ist die Eingabe eine zusammengesetzte Zahl, so ist die Antwort mit einer Wahrscheinlichkeit von $1 - \varphi(n) / (n-1)$ korrekt, da es $\varphi(n)$-viele Zahlen $< n$ gibt, die keinen ggT mit n besitzen.

Wendet man den Test mehrfach mit unabhängig voneinander gewählten a's an, so multiplizieren sich die Wahrscheinlichkeiten und die Wahrscheinlichkeit, dass der Algorithmus bei einer zusammengesetzten Zahl ein korrektes Ergebnis liefert ist dann $1 - (\varphi(n) / (n-1))^k$ bei k Versuchen.

2.2. Der Fermat-Test

Eine Erweiterung des φ-Tests ist der Fermat-Test:

Algorithm 7 Fermat-Test

EINGABE: Die Zahl n, die auf Primheit getestet werden soll.
AUSGABE: n prim oder n nicht prim.

(1) Wähle zufällig eine Zahl $a \in_R \mathbb{Z}_n$.
(2) *** *φ-Test* ***
  if $\mathrm{ggT}(a,n) \neq 1$ **then**
  (a) **return** n ist nicht prim.
(3) **else**
  (a) **if** $a^{n-1} \not\equiv 1 \mod n$ **then**
    (i) **return** n ist nicht prim.
  (b) **else**
    (i) **return** n ist prim

$F(n)$ „Zeugen im Fermat-Test"

Die **Menge der Zeugen (der Nicht-Primheit) im Fermat-Test** ist

$$F(n) := \left\{a \in \mathbb{Z}_n^* \mid a^{n-1} \not\equiv 1 \mod n\right\}.$$

Wenn $\mathbb{Z}_n^* \setminus F(n) = \left\{a \in \mathbb{Z}_n^* \mid a^{n-1} \equiv 1 \mod n\right\} \neq \emptyset$ und $\neq \mathbb{Z}_n^*$, dann ist $\mathbb{Z}_n^* \setminus F(n)$ eine echte Untergruppe von $\mathbb{Z}_n^*$. Dann gilt aber auch $\operatorname{ord}(\mathbb{Z}_n^* \setminus F(n)) \mid \operatorname{ord}(\mathbb{Z}_n^*)$, also $\operatorname{ord}(\mathbb{Z}_n^* \setminus F(n)) \mid \varphi(n)$ und ist ein echter Teiler. Ist n also keine Primzahl, so sind mindestens die Hälfte aller Zahlen in $\mathbb{Z}_n^*$ Zeugen im Fermat-Test, da 2 der kleinste Teiler der Ordnung der Gruppe sein kann. Ebenfalls gilt dann $\lambda(n) \nmid (n-1)$, da andernfalls folgen würde

$$\lambda(n) \mid (n-1) \overset{n-1=k\cdot\lambda(n)}{\Rightarrow} a^{\lambda(n)} \equiv 1 \mod n \quad \Leftrightarrow \quad a^{\lambda(n)\cdot k} \equiv a^{n-1} \equiv 1 \mod n,$$

was ein Widerspruch zu der Annahme ist, dass n nicht prim ist ($\rightarrow$Fermat).

Aus dieser Erkenntnis können wir die Menge der Carmichaelzahlen auch wir folgt definieren::

Carmichael Zahl

DEFINITION. Eine **Carmichaelzahl** ist eine nicht-prime Zahl n mit $\lambda(n) \mid n-1$.

Diese Zahlen bestehen also den φ- und den Fermat-Test, obwohl sie nicht prim sind – und das mit gleicher Wahrscheinlichkeit!

COROLLARY 2.2.1. *Ist n ungerade, nicht-prim, $\lambda(n) \nmid n-1$, dann liefert der Fermat-Test mit Ws $\geq \frac{1}{2}$ einen Zeugen $a \in F(n)$.*

BEWEIS. Die Wahrscheinlichkeit ergibt sich aus dem Quotienten der Anzahl Zeugen im Fermat-Test und der Anzahl der Zahlen in $\mathbb{Z}_n^*$: $Ws_a = \frac{\#F(n)}{\varphi(n)} \geq \frac{1}{2}$. □

PROPOSITION 2.2.2. ***(Satz von Pollard)*** *Eine Zahl $n \in \mathbb{N}$ ist genau dann eine Carmichaelzahl, wenn folgende Bedingungen erfüllt sind:*

(1) Es gibt keine Primzahl p mit $p^2 \mid n$.
(2) n ist ein Produkt von $r \geq 3$ verschiedenen ungeraden Primzahlen.
(3) Für jede Primzahl p gilt: $p \mid n \quad \Rightarrow \quad (p-1) \mid (n-1)$.

BEWEIS. „⇒": Wenn $n = \Pi_{i=1}^r p_i^{e_i}$ ungerade ist, dann folgt aus $\lambda(n) \mid \varphi(n) = \Pi_{i=1}^r \varphi\left(p_i^{e_i}\right) \mid n-1$, dass auch gilt $\varphi\left(p_i^{e_i}\right) = p_i^{e_i-1}(p_i-1) \mid n-1$. Da aber gilt $p_i \nmid n-1$ (trivial, da p_i ja n teilt), müssen alle $e_i = 1$ sein. Also ist n quadratfrei.

Angenommen, $r = 2$, also $n = p_1 \cdot p_2$ mit $p_1 < p_2$ o.B.d.A., dann gilt

$$\lambda(n) \mid \varphi(n) = (p_1-1)(p_2-1) \mid n-1 \Rightarrow p_2 - 1 \mid \underbrace{p_1 p_2}_{=n} - 1 = p_1(p_2-1) + (p_1-1).$$

Dann würde aber auch gelten, dass $p_2 - 1 \mid p_1 - 1$, was ein Widerspruch wäre zu $p_1 < p_2$. Es ist also $r \geq 3$.

Wir können also annehmen, dass n das Produkt von mindestens 3 verschiedenen Primzahlen ist. Sei p ein Primteiler von n und $n = pr$ mit $\operatorname{ggT}(r,p) = 1$. Sei g das erzeugende Element für $\mathbb{Z}_p^*$ und sei $a \in \mathbb{Z}_n$ definiert durch

$$\begin{aligned} a &\equiv q \mod p \\ a &\equiv 1 \mod r \end{aligned}$$

Dieses a exisitert wegen dem CRT und es gilt $\operatorname{ord}_n(a) \mid p-1$. Aus $a^{n-1} \equiv 1 \mod n$ folgt dann $(p-1) \mid (n-1)$.

„⇐": Sind nun diese drei Bedingungen für n erfüllt, so ist n ein Produkt von mindestens 3 Primzahlen (keine Primzahlpotenzen!), und für jeden Primfaktor gilt $(p_i - 1) \mid (n-1)$. Für ein beliebiges $a \in_R \mathbb{Z}_n^*$ gilt dann also $a^{n-1} \equiv 1 \mod p_i$ für alle Primteiler p_i von n. Damit ist aber auch $a^{n-1} \equiv 1 \mod n$, die Ordnung von a teilt also $n-1$. □

2.3. Legendre-/Jacobi-Symbol

Für nun folgende Primzahltests benötigen wir das Legendre-/Jacobi-Symbol. Beide zeigen an, ob eine Zahl x bzgl. n ein Quadrat ist.

Legendresymbol

Quadratischer Rest

DEFINITION. Sei p eine ungerade Primzahl. Das **Legendre-Symbol** $\left(\begin{array}{c} x \\ p \end{array}\right) \in \{0, \pm 1\}$ gibt für $x \in \mathbb{Z}$ an, ob es ein **quadratischer Rest** bezüglich p ist oder nicht. Dabei ist

$$\begin{aligned} \mathrm{QR}_p &= \left\{x^2 \mod p \mid x \in \mathbb{Z}_p^*\right\} \\ \mathrm{QNR}_p &= \mathbb{Z}_p^* \setminus \mathrm{QR}_p \\ \mathrm{QR}_p &\subseteq \mathbb{Z}_p^* \quad \text{(Untergruppe)} \end{aligned}$$

Es gilt:

$$\begin{aligned} \left(\begin{array}{c} x \\ p \end{array}\right) &= \left(\begin{array}{c} x \mod p \\ p \end{array}\right) \\ \left(\begin{array}{c} x \\ p \end{array}\right) &= x^{\frac{p-1}{2}} \mod p \quad \text{fuer } p \nmid x \\ &\Rightarrow \begin{cases} \left(\begin{array}{c} x \\ p \end{array}\right) = 0 & \text{fuer } p \text{ prim}, p > 2, p \mid x \\ \left(\begin{array}{c} 0 \\ p \end{array}\right) = 0 & \text{fuer } p \text{ prim}, p \neq 2 \end{cases} \\ \left(\begin{array}{c} x \\ p_1 \cdot \ldots \cdot p_r \end{array}\right) &= \left(\begin{array}{c} x \\ p_1 \end{array}\right) \cdot \ldots \cdot \left(\begin{array}{c} x \\ p_r \end{array}\right) \end{aligned}$$

Weiterhin gilt

$$\begin{aligned} \left(\begin{array}{c} x \\ p \end{array}\right) \equiv x^{\frac{p-1}{2}} \mod p &= 1 \quad \Leftrightarrow \quad x \in \mathrm{QR}_p \\ &= -1 \quad \Leftrightarrow \quad x \in \mathrm{QNR}_p \\ &= 0 \quad \Leftrightarrow \quad \mathrm{ggT}(x,p) \neq 1 \end{aligned}$$

BEWEIS. Sei $\langle g \rangle = \mathbb{Z}_p^* = \left\{g, g^2, \ldots, g^{p-1} \equiv 1 \mod p\right\}$. Es gilt

$$\mathrm{QR}_p = \left\{g^2, \ldots, g^{p-1}\right\}.$$

Ist also x ein quadratischen Rest, so gilt mit $x = g^{2k}$, $k \in \left\{1, \ldots, \frac{p-1}{2}\right\}$

$$x^{\frac{p-1}{2}} \equiv \left(g^{2k}\right)^{\frac{p-1}{2}} \equiv \left(\underbrace{g^{p-1}}_{\equiv 1 \mod p}\right)^k \equiv 1^k \equiv 1 \mod p.$$

Ist x ein quadratischer Nichtrest, so gilt mit $x = g^{2k-1}$, $k \in \left\{1, \ldots, \frac{p-1}{2}\right\}$

$$x^{\frac{p-1}{2}} \equiv \left(g^{2k-1}\right)^{\frac{p-1}{2}} \equiv g^{(p-1)k-\frac{p-1}{2}} \equiv \frac{\left(g^{p-1}\right)^k}{g^{\frac{p-1}{2}}} \equiv (*)$$

von $\left(g^{p-1}\right)^k$ wissen wir bereits, dass es gleich 1 ist modulo p. Desweiteren wissen wir, dass ord $\left(g^{\frac{p-1}{2}}\right) = 2$, da die Untergruppe, die von diesem Element erzeugt wird genau zwei Elemente besitzt (nämlich die 1 und die das erzeugende Element selbst). Dann kann aber das erzeugende Element kein quadratischer Rest sein, da es mit sich selbst multipliziert 1 ergibt. Also ist $g^{\frac{p-1}{2}} \equiv -1 \mod p$. Es ergibt sich

$$(*) \equiv x^{\frac{p-1}{2}} \equiv \frac{1^k}{-1} \equiv -1.$$

□

Zur effizienten Berechnung des Legendre-Symbols sollen nun weitere Regeln eingeführt werden:

PROPOSITION 2.3.1. *Für $x, y \in \mathbb{Z}_n$, n ungerade und prim gilt*

(1) $\left(\begin{array}{c} xy \\ p \end{array}\right) = \left(\begin{array}{c} x \\ p \end{array}\right) \cdot \left(\begin{array}{c} y \\ p \end{array}\right)$ (n quadratfrei und ungerade)

(2) $\left(\begin{array}{c} -1 \\ p \end{array}\right) = (-1)^{\frac{p-1}{2}}$

(3) $\left(\begin{array}{c} 2 \\ p \end{array}\right) = (-1)^{\frac{p^2-1}{8}}$

(4) $\left(\begin{array}{c} p \\ q \end{array}\right) = \left(\begin{array}{c} q \\ p \end{array}\right) \cdot (-1)^{\frac{(p-1)(q-1)}{4}}$, $p, q > 2$ und prim „**Gauß'sches Reziprozitätsgesetz**"

Gauß'sches Reziprozitätsgesetz

BEWEIS. Zum Beweis siehe [**Wohlfahrt**]. □

Nach Fermat gilt

$$a^{p-1} \equiv 1 \mod p \quad , \quad \text{fuer alle } a \neq 0 \mod p$$
$$a^{\frac{p-1}{2}} \equiv \left(\begin{array}{c} a \\ p \end{array}\right) \mod p \quad , \quad \text{fuer } a \in \mathbb{Z}.$$

Eulerkriterium

LEMMA 2.3.2. ***(Eulersches Kriterium)*** *Sei $a \in \mathbb{Z}_p^*$. Dann gilt*

$$\left(\frac{a}{p}\right) = 1 \quad \Leftrightarrow \quad a^{\frac{p-1}{2}} \equiv 1 \mod p.$$

BEWEIS. Weil $\mathbb{Z}_p^*$ zyklisch ist, exisitert ein erzeugendes Element g für $\mathbb{Z}_p^*$.

„$\Rightarrow$": Ist $\left(\frac{a}{p}\right) = 1$, so exisitert ein $b \in \mathbb{Z}_p^*$ mit $a \equiv b^2 \mod p$ und $b = g^i$. Dann ist also $a \equiv g^{2i} \mod p$ und somit

$$a^{\frac{p-1}{2}} \equiv g^{(p-1)i} \equiv 1 \mod p.$$

„⇐": Sei $a^{\frac{p-1}{2}} \equiv 1 \mod p$ und $a = g^i$. Da gilt $a^{\frac{p-1}{2}} \equiv \left(g^i\right)^{(p-1)/2} \equiv g^{i(p-1)/2} \equiv 1 \mod p$, gilt $i(p-1)/2 \mid p-1$ und damit $i = 1$ oder $i = 2$. Im Fall $i = 1$ ist $a \in QR_p$, da $a = q = \left(g^{(p-1)/2}\right)^2$, im Fall $i = 2$ ist $a = g^2 \in QR_p$. □

Eine allgemeinere Form des Legendre-Symbols ist das Jacobisymbol.

Jacobisymbol

DEFINITION. **(Jacobisymbol)** Sei $n \in \mathbb{N}$ mit $n = \prod_{i=1}^{k} p_i^{e_i}$, dann gilt für $a \in \mathbb{Z}$

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \cdot \ldots \cdot \left(\frac{a}{p_k}\right)^{e_k},$$

wobei $\left(\frac{a}{p_i}\right)$ das Legendresymbol ist. $\left(\frac{a}{n}\right)$ heisst das **Jacobisymbol modulo *n***. Um es etwas klarer zu machen: Das Legendresymbol ist nur über Primzahlen definiert, während beim Jacobisymbol die Zahl unten auch zusammengesetzt sein kann.

Ist das Jacobisymbol modulo n von a gleich -1, so ist die Gleichung $x^2 \equiv a \mod n$ prinzipiell nicht lösbar. Im Fall $\left(\frac{a}{n}\right) = 1$ kann nicht allgemein gesagt werden, dass dann die Gleichung lösbar ist (Gegenbeispiel: $n = p^2$ und $a \in_R \mathbb{Z}_p^*$ mit $p \in \mathbb{P}$ – dann ist $\left(\frac{a}{n}\right) = 1$, aber $x^2 \equiv a \mod n$ nicht lösbar). Jedoch können viele Aussagen über das Legendresymbol auf das Jacobisymbol verallgemeinert werden (z.B. Satz 2.3.1 auf der vorherigen Seite).

Insbesondere gelten für das Jacobisymbol folgende Regeln:

PROPOSITION 2.3.3. *1) Sei $g \in \mathbb{Z}_p^*$ mit $\langle g \rangle = \mathbb{Z}_p^*$ und $a \in \mathbb{Z}_p^*$ mit $a = g^i$, dann gilt*

$$\left(\frac{a}{p}\right) = 1 \Leftrightarrow i \text{ ist ungerade.}$$

2) $a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \mod p$

3) Es gibt in $\mathbb{Z}_p^*$ genauso viele quadratische Reste wie quadratische Nichtreste, nämlich jeweils $(p-1)/2$ viele.

4) $\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right)$

5) Sei $a > 0$, $ggT(a,n) = 1$, dann gilt $\left(\frac{a}{n}\right) = \left(\frac{n}{a}\right)^{(n-1)(a-1)/4}$.

Zur schnellen Berechnung des Legendre- oder Jacobi-Symbols kann folgender Algorithmus benutzt werden:

Laufzeit: Dieser Algorithmus entspricht im wesentlichen dem euklidischen Algorithmus und berechnet deshalb den Wert des Legendre-Symbols in $O\left((\log n)^2\right)$-vielen Bitoperationen.

In diesem Zusammenhang sei ein Algorithmus angegeben, mit dem unter Zuhilfenahme des Jacobi-Symbols die Quadratwurzel einer Zahl berechnet werden kann. In diesem Algorithmus wird zu einem gegebenen Qaudratischen Rest a ein Quadratischer Nichtrest u zufällig gesucht. Diesen verwendet wir, um a nach $\left(\mathbb{Z}_p^*\right)^{2^r}$ zu transponieren: Ist die Wurzel von $a^{p-1} \equiv -1 \mod n$, so existiert diese nicht – in diesem Fall wird der quadratische Nichtrest anmultipliziert, was dann wieder einen quadratischen Rest ergibt. Das machen wir so lange, bis wir einen quadratischen Nichtrest in $\left(\mathbb{Z}_p^*\right)^{2^r}$ haben. Dort ist das Wurzelziehen dann sehr einfach und wir können das Ergebnis dann wieder nach $\mathbb{Z}_p^*$ heruntertransponieren, indem wir die anmultiplizierten quadratischen Nichtreste wieder herausdividieren

Algorithm 8 Berechnung des Legendre-/Jacobi-Symbols

EINGABE: n ungerade, $a \in \mathbb{Z}_n^*$
AUSGABE: Legendre-Symbol $\left(\frac{a}{n}\right)$

(1) $\sigma := 1$, $x \equiv a \mod n$.
(2) **while** $x \neq 1$ **do**
  (a) **** x halbieren, bis es ungerade ist ****
    while x gerade **do**
      (i) $\left(\frac{x}{n}\right) := \left(\frac{\frac{x}{2}}{n}\right)$.
      (ii) $\sigma := \sigma \cdot (-1)^{(n^2-1)/8}$.
  (b) $\left(\frac{x}{n}\right) := \left(\frac{n \mod x}{x}\right)$.
  (c) $\sigma := \sigma \cdot (-1)^{(n-1)(x-1)/4}$
(3) **if** $n = 1$ **then**
  (a) **return** σ.
(4) **else**
  (a) **return** 0.

(allerdings nur mit dem halben Exponenten, da ja auch von dem quadratischen Nichtrest die Wurzel gezogen wurde).

Algorithm 9 Berechnung der Quadratwurzel modulo p

EINGABE: $p \in \mathbb{N}$ prim, $a \in \mathbb{Z}_p^*$ mit $\left(\frac{a}{p}\right) = 1$
AUSGABE: $b \in \mathbb{Z}_p^*$ mit $b^2 \equiv a \mod p$

(1) Berechne $p - 1 = 2^r d$ mit d ungerade.
(2) $u \in_R \mathbb{Z}_p^*$ mit $\left(\frac{u}{p}\right) = -1$.
(3) $e := 0$.
(4) **for** $i = 2, \ldots, r$ **do**
  (a) **if** $(au^{-e})^{\frac{p-1}{2^i}} \not\equiv 1 \mod p$ **then**
    (i) $e := 2^{i-1} + e$.
(5) $h := au^{-e} \mod p$.
(6) $b := u^{e/2} h^{\frac{d+1}{2}} \mod p$.
(7) **return** b.

Laufzeit: Für Schritt 1 werden $O(\log p)$ Schritte benötigt. In Schritt 2 finden wird nach einer erwarteten Laufzeit von 2 Schritten einen quadratischen Nichtrest, da genau die Hälfte aller Zahlen aus $\mathbb{Z}_p^*$ quadratische Nichtreste sind. Die Berechnung von $\left(\frac{u}{p}\right)$ für jedes gefundene u benötigt eine Anzahl von $O\left((\log p)^2\right)$ vielen Schritten. Die Schleife in Schritt 4 wird $O(\log p)$-mal durchlaufen, wobei bei jedem Schritt $O\left(\log^3 p\right)$ viele Operationen durchgeführt werden müssen (sukkzessives Quadrieren), da $r \leq \log p$ ist – bleibt eine Laufzeit von $O\left((\log p)^4\right)$. Das b in Schritt 6 kann dann in Zeit $O\left((\log b)^3\right)$ vielen Schritten berechnet werden.
Insgesamt benötigt der Algorithmus also eine Laufzeit von $O\left((\log p)^4\right)$ vielen Schritten.

Korrektheit: Die Korrektheit des Algorithmus läßt sich einsehen, wenn man überlegt, dass gilt

$$\begin{aligned} & b^2 \equiv a \mod p \\ \Leftrightarrow \quad & u^e h^{d+1} \equiv a \mod p \\ \Leftrightarrow \quad & u^e \left(au^{-e}\right)^{d+1} \equiv a \mod p \\ \Leftrightarrow \quad & a^{d+1} u^{e(1-d-1)} \equiv a \mod p \\ \Leftrightarrow \quad & a^d u^{-ed} \equiv 1 \mod p \\ \Leftrightarrow \quad & \left(au^{-e}\right)^d \equiv 1 \mod p. \end{aligned}$$

Die letzte Kongruenz lässt sich induktiv beweisen, da in Schritt 4, sobald $(au^{-e})^{\frac{p-1}{2}} \equiv -1 \mod p$ ist, auf $e\ 2^{i-1}$ aufaddiert wird, was einer Multiplikation der linken Seite mit $u^{\frac{p-1}{2}}$ entspricht ($\equiv -1 \mod p$, da $u \in QNR_p$).

Der Fermat-Test lässt sich nun verallgemeinern zum Solovay-Strassen-Test.

2.4. Der Solovay-Strassen-Test

Algorithm 10 Solovay-Strassen-Test

EINGABE: $n \in \mathbb{N}$ ungerade (weil das Jacobi-Symbol für gerade Zahlen nicht definiert ist)
AUSGABE: n prim oder n nicht prim

(1) Wähle $a \in_R \mathbb{Z}_m^*$ zufällig.
(2) **if** $\mathrm{ggT}(a,n) \neq 1$ **then**
  (a) **return** n zusammengesetzt.
(3) **if** $a^{\frac{n-1}{2}} \not\equiv \left(\begin{array}{c} a \\ n \end{array}\right) \mod n$ **then**
  (a) **return** n ist nicht prim.
(4) **else**
  (a) **return** n ist prim.

Laufzeit: Die Berechnung von $\mathrm{ggT}(a,n)$ und die Berechnung von $\binom{a}{n}$ benötigen jeweils Zeit $O\left(\log^2 n\right)$. Der Wert von $a^{\frac{n-1}{2}} \mod n$ kann durch sukzessives Quadrieren mit $O(\log n)$ Operationen in $\mathbb{Z}_n$ berechnet werden,wobei jede Operation $O\left(\log^2 n\right)$-viel Zeit benötigt. Insgesamt wird für Schritt 3 also ein Zeit von $O\left(\log^3 n\right)$ benötigt. Bleibt also eine Laufzeit von $O\left((\log n)^3\right)$ vielen Bitoperationen.

Der Algorithmus ist ein sog. *Monte-Carlo-Algorithmus*, d.h. die Ausgabe „n ist zusammengesetzt" ist immer korrekt, während die Aussage „n ist prim" mit einem Fehler $\leq 1/2$ falsch ist. Durch mehrfache unabhängige Anwendung des Tests kann so mit großer Wahrscheinlichkeit eine korrekte Aussage über die Primheit von n gemacht werden.

Die **Menge der Zeugen (der Nicht-Primheit) im Solovay-Strassen-Test** ist

$$SS(n) := \left\{ a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \not\equiv \left(\begin{array}{c} a \\ n \end{array}\right) \mod n \right\}.$$

$SS(n)$ „Zeugen im Solovay-Strassen-Test"

Es gilt $F(n) \subseteq SS(n)$, da

$$a \in F(n) \Rightarrow a \in SS(n) \quad \Leftrightarrow \quad a \in \overline{SS}(n) \Rightarrow a \in \overline{F}(n),$$

wobei $\overline{F}(n)$ und $\overline{SS}(n)$ die Mengen der nicht-Zeugen im Fermat- bzw. Solovay-Strassen-Test sind. Damit folgt:

$$\begin{aligned} a \in \overline{SS}(n) &\Rightarrow a^{\frac{n-1}{2}} \equiv \left(\begin{array}{c} a \\ n \end{array}\right) \equiv \pm 1 \mod n \Rightarrow a^{n-1} \equiv (\pm 1)^2 \equiv 1 \mod n \\ &\Rightarrow a \in F(n). \end{aligned}$$

Weiterhin gilt, dass wenn es Zeugen im SS-Test gibt, diese mindestens die Hälfte aller Zahlen in $\mathbb{Z}_n^*$ ausmachen

$$SS(n) \neq \emptyset \Rightarrow \#\mathbb{Z}_n^* \setminus SS(n) \leq \frac{\varphi(n)}{2} \Rightarrow \frac{\#SS(n)}{\varphi(n)} \geq \frac{1}{2}.$$

da wie im Fermat-Test $\mathbb{Z}_n^* \setminus SS(n) \subseteq \mathbb{Z}_n^*$ eine Untergruppe ist und $a \mapsto \left(\begin{array}{c} a \\ n \end{array}\right) \mod n$ multiplikativ ist.

LEMMA 2.4.1. *Sei $n \in \mathbb{N}$ ungerade, dann gilt*

$$n\,prim \quad \Leftrightarrow \quad SS(n) = \emptyset$$

BEWEIS. „⇒" gilt nach Definition von $\left(\begin{array}{c} a \\ n \end{array}\right)$.

„⇐" siehe auch [**Blömer**, Lemma 6.1]

Für $n = 2^k q + 1$, k maximal, sei

$$\begin{aligned} F(n) &= \left\{a \in \mathbb{Z}_n^* \mid a^{n-1} \not\equiv 1 \mod n\right\} \\ SS(n) &= \left\{a \in \mathbb{Z}_n^* \mid a^{\frac{n-1}{2}} \not\equiv \left(\begin{array}{c} a \\ n \end{array}\right) \mod n\right\} \\ MR(n) &= \left\{a \in \mathbb{Z}_n^* \mid a^q \not\equiv 1 \mod n \text{ und } a^{q2^i} \not\equiv -1 \mod n \forall i = 0, \ldots, k-1\right\}. \end{aligned}$$

Dass gilt $SS(n) = \emptyset \Rightarrow F(n) = \emptyset$ ist klar. n muss also prim oder Carmichael sein. Wir müssen nun zeigen, dass gilt

$$\begin{aligned} SS(n) = \emptyset &\Rightarrow n \text{ ist nicht Carmichael} \\ \Leftrightarrow \quad n \text{Carmichael} &\Rightarrow SS(n) \neq \emptyset. \end{aligned}$$

Sei also $a \in \mathbb{Z}_n^*$, $n = p_1 \cdot \ldots \cdot p_r$, $r \geq 3$ Carmichael, mit $\left(\begin{array}{c} a \\ n \end{array}\right) = -1$ und $\left(\begin{array}{c} a \\ p_1 \end{array}\right) = +1$.

Wenn $a^{\frac{n-1}{2}} \equiv 1 \mod n$, dann ist $SS(n) \neq \emptyset$, wir sind also fertig.

Sei also $a^{\frac{n-1}{2}} \not\equiv +1 (\equiv -1) \mod n$. Nach dem CRT gilt

$$\begin{array}{rcccc} a & = & (& a_1, \ \ldots \ , a_r &). \\ & & & \in \quad\quad \in & \\ & & & \mathbb{Z}_{p_1}^* \quad\quad \mathbb{Z}_{p_r}^* & \end{array}$$

Ersetzte nun a_1 durch $\tilde{a}_1$, so dass $\tilde{a}_1$ Generator von $\mathbb{Z}_{p_1}^*$, also auch $\left(\begin{array}{c} \tilde{a}_1 \\ p_1 \end{array}\right) = -1$. Sei $\tilde{a} = (\tilde{a}_1, a_2, \ldots, a_r)$ und

$$\left(\begin{array}{c} \tilde{a} \\ n \end{array}\right) = \Pi_{i=1}^r \left(\begin{array}{c} a \\ p_i \end{array}\right) = \left(\begin{array}{c} \tilde{a}_1 \\ p_1 \end{array}\right) \cdot \left(\begin{array}{c} a_2 \\ p_2 \end{array}\right) \cdot \ldots \cdot \left(\begin{array}{c} a_r \\ p_r \end{array}\right) = - \left(\begin{array}{c} a \\ n \end{array}\right) = 1$$

Da $\mathrm{ord}_n\tilde{a}$ mindestens genauso groß ist wie $\mathrm{ord}_n a$, da $\mathrm{ord}_{p_1}\tilde{a}_1 = \varphi\left(p_1\right)$ (also die maximale Ordnung) ist, teilt $\mathrm{ord}_n(a) = \mathrm{kgV}\left(\mathrm{ord}_{p_i}(a_i)\right)$ also $\mathrm{ord}_n\tilde{a} = \mathrm{kgV}\left(\mathrm{ord}_{p_1}\left(\tilde{a}_1\right), \mathrm{ord}_n(a)\right)$, da die Ordnung der Gruppe, die durch a erzeugt wird die Gruppenordnung $\mathrm{ord}_{p_1}\tilde{a}$ teilt.

Wenn nun $\tilde{a}^{\frac{n-1}{2}} \equiv 1 \mod n$ wäre, dann wäre auch $a^{\frac{n-1}{2}} \equiv 1 \mod n$ – ein Widerspruch zur Anahme. Folglich ist

$$\left(\begin{array}{c} \tilde{a} \\ n \end{array}\right) = 1 \not\equiv \tilde{a}^{\frac{n-1}{2}} \mod n \quad , \text{ d.h. } \tilde{a} \in SS(n).$$

□

Um es nochmal zusammenzufassen: Die Zeugen der Nichtprimheit im Solovay-Strassen-Test ist eine Gruppe – genauso bilden die „Nichtzeugen" eine Untergruppe von $\mathbb{Z}_n^*$, und haben damit eine Ordnung von höchstens $n/2$. Dass es überhaupt Elemente in $\mathbb{Z}_n^*$ gibt, die die Nichtzeugen-Eigenschaft besitzen, zeigt as vorangehende Lemma.

PROPOSITION 2.4.2. *Ist n ungerade, nicht prim, dann ist* $\#SS(n) \geq \frac{\varphi(n)}{2}$.

COROLLARY 2.4.3. *Findet der Solovay-Strassen-Test in k statistisch unabhängigen Iterationen keinen Zeugen* $a \in SS(n)$ *der Zusammengesetztheit, dann ist*

$$Ws_{a^{(1)},\ldots,a^{(k)}}(n\,nicht-prim) \leq 2^{-k}.$$

LEMMA 2.4.4. *Sei p prim,* $p-1 = 2^k q$, *q ungerade. Dann gilt für alle* $a \in \mathbb{Z}_p^*$:

$$a^q \equiv 1 \mod p \quad \vee \quad \exists i, 0 \leq i < k : a^{q2^i} \equiv -1 \mod p.$$

BEWEIS. Die 1 hat im Körper $\mathbb{Z}_p$ nur zwei Quadratwurzeln, nämlich $+1$ und -1. Genauso gilt: $x^2 - 1 \in \mathbb{Z}_p[x]$ hat genau zwei Lösungen (nämlich $+1$ und -1).

Nach Fermat gilt nun

$$\begin{array}{lllll} a^{q2^k} & \equiv & a^{p-1} & \equiv & 1 \mod p \\ a^{q2^{k-1}} & \equiv & & & \pm 1 \mod p \,(\text{da die Wurzel gezogen wird}). \end{array}$$

Es ist also entweder schon $a^q \equiv 1 \mod p$ und man quadriert bis $a^{q2^k} \equiv 1 \mod p$ oder es tritt irgendwann eine -1 auf. □

Nachfolgendes „Sätzchen" wird in den Beweisen zu den Miller-Rabin-Tests verwendet:

FACT. *Sei* $b \equiv 1$ *oder* $-1 \mod n$. *Dann ist n nicht prim.*

BEWEIS. Aus der Voraussetzung folgt, dass gilt $b^2 \equiv 1 \mod n$. Somit ist $n \mid (b^2-1) = (b-1)(b+1)$, jedoch ist wegen $b \not\equiv 1 \mod n$ oder $b \not\equiv -1 \mod n$ n weder ein Teiler von $b-1$ noch von $b+1$. Somit ist n zusammengesetzt aus mindestens zwei Primfaktoren, die Teiler von $b-1$ und $b+1$ sind. Also ist n nicht prim. □

2.5. Der Miller-Rabin-Test

Wir definieren die **Menge der Zeugen (der Nicht-Primheit) im Miller-Rabin-Test** als

$$MR(n) := \left\{ a \in \mathbb{Z}_n^* \mid a^q \not\equiv 1 \mod n \wedge \forall i, 0 \leq i < k : a^{q2^i} \not\equiv -1 \mod n \right\}.$$

$MR(n)$ „Zeugen im Miller-Rabin-Test"

Offenbar gilt $F(n) \subseteq MR(n)$, da aus

$$a \in F(n) \Rightarrow a^{n-1} \not\equiv 1 \mod n \Rightarrow a^{q2^k} = (a^q)^{2^k} \not\equiv 1 \mod n$$

folgt, dass $a^q \not\equiv \pm 1 \mod n$ sein muss, da ansonsten nach k-facher Quadrierung eine 1 herauskommen würde.

PROPOSITION 2.5.1. *Sei $n \in \mathbb{N}$ ungerade. Dann gilt*

(1) n prim $\Rightarrow MR(n) = \emptyset$
(2) n nicht-prim $\Rightarrow \#MR(n) \geq \frac{3}{4}\varphi(n)$

BEWEIS. 1. gilt nach Lemma 2.4.4.

2. Siehe Bew. von Satz 2.5.2. □

Der nachfolgende Beweis aus [**Childs**, S. 373ff] zeigt eine kleinere Schranke ($\varphi(n)/2$) und wird zur Ergänzung aufgeführt.

BEWEIS. Sei n nicht-Carmichael und nicht-Prim und

$$PS_n = \{a \in \mathbb{Z}_n | a^{n-1} \equiv 1 \mod n\}$$

die Untergruppe der Nicht-Zeugen von $\mathbb{Z}_n$. Da n nicht-Carmichael ist, muss es mindestens ein $[a] \in \mathbb{Z}_n$ geben, das ein Zeuge der Nicht-Primheit ist (sonst wäre n ja prim). Also ist PS_n eine echte Untergruppe, und es gilt $\text{ord}(PS_n) | \text{ord}(\mathbb{Z}_n)$. Somit kann eine Untergruppe maximaler Größe höchstens die Hälfte der Elemente der Obergruppe enthalten. Also ist die Anzahl der Nicht-Zeugen $\geq \frac{\varphi(n)}{2}$.

Sei n also Carmichael. Dann ist $n = p_1 p_2 \cdots p_r$ mit $r \geq 3$ und $p_i - 1 \mid n-1$ für $1 \leq i \leq r$. Sei $n - 1 = 2^e q$ mit q ungerade und $p_i - 1 = 2^{s_i} t_i$ mit t_i ungerade. Wegen $p_i - 1 \mid n-1$ gilt $t_i \mid q$ und $s_i \leq e$ für alle i. Somit gilt auch $p_i - 1 \mid 2^s q$ für i beliebig. Wir benennen die Primfaktoren jetzt so um, dass sie sortiert nach s_i vorliegen:

$$s_1 = s_2 = \ldots = s_d > s_{d+1} \geq \ldots \geq s_r.$$

Mit $s = s_1$ beizeichnen wir den Exponenten der größten Zweierpotenz, die in den Primfaktoren enthalten ist.

Wir betrachten für ein $a \in \mathbb{Z}_n^*$ die Sequenz

$$\left\{ a^q, a^{2q}, \ldots, a^{2^{s-1}q}, a^{2^s q}, \ldots, a^{2^e q} \right\}.$$

Für $a^{2^s q} \mod n$ ist unter Anwendung des CRT

$$a^{2^s q} \mod n \underset{CRT}{\cong} \left(\left[a^{2^s q}\right]_{p_1}, \ldots, \left[a^{2^s q}\right]_{p_r} \right) \underset{p_i - 1 | 2^s q}{\equiv} \left([1]_{p_1}, \ldots, [1]_{p_r} \right) \underset{CRT}{\cong} [1]_n.$$

Jetzt betrachten wir $a^{2^{s-1}q} \mod n$:
Falls $s_i < s$, dann gilt $p_i - 1 \mid 2^{s-1}q$, wegen $p_i - 1 = 2^{s_i} t_i = ord(p_i)$, und somit ist

$$a^{2^{s-1}q} \equiv 1 \mod p_i.$$

Falls jedoch $s_i = s$ ist, dann ist $a^{2^{s-1}q} \equiv 1 \text{ oder } -1 \mod p_i$. Wir können jetzt zeigen, dass beide Möglichkeiten gleich wahrscheinlich sind:

Sei $\mathbb{Z}_{p_i} = \langle \beta \rangle$. Dann ist $\text{ord}(\beta) = p_i - 1 = 2^s t_i$, und wegen $t_i \mid q$ ist $\beta^{2^s q} \equiv 1 \mod p_i$. Die Wurzel von $\beta^{2^s q}$ ist $\beta^{2^{s-1}q} \equiv 1 \mod p_i$ oder $\equiv -1 \mod p_i$ und da p_i prim ist, hat 1 modulo p_i nur zwei Wurzeln. Es gilt $2^s t_i \nmid 2^{s-1}q$, also ist $\beta^{2^{s-1}q} \not\equiv 1 \mod p_i$ sondern $\equiv -1 \mod p_i$. Somit ist

$$\begin{aligned}
\beta^{(2^{s-1}q)^1} &\equiv -1 \mod p_i \\
\beta^{(2^{s-1}q)^2} &\equiv 1 \mod p_i \\
\beta^{(2^{s-1}q)^3} &\equiv -1 \mod p_i \\
&\vdots \\
(\beta^c)^{2^{s-1}q} &\equiv (-1)^c \mod p_i.
\end{aligned}$$

Wie man sieht kommen $[1]_{p_i}$ und $[-1]_{p_i}$ gleich oft vor. Dies gilt für beliebige p_i. Wenn wir ein Element

$$\left([a_1]_{p_1}, [a_2]_{p_2}, \ldots, [a_r]_{p_r} \right) \in U_{p_1} \times U_{p_2} \times \cdots \times U_{p_r}$$

in die $2^{s-1}q$-te Potenz erheben, erhalten wir

$$\left([a_1^{2^{s-1}q}]_{p_1}, [a_2^{2^{s-1}q}]_{p_2}, \ldots, [a_r^{2^{s-1}q}]_{p_r} \right) \equiv \left(\overbrace{[\pm 1], \ldots, [\pm 1]}^{s_i = s}, \overbrace{[1], \ldots, [1]}^{s_i < s} \right).$$

Dies bedeutet, dass – unter Anwendung des CRT – gilt

$$[a_r^{2^{s-1}q}]_n \equiv [\pm 1] \quad \Rightarrow \quad [a_r^{2^s q}]_n \equiv [1]$$

und a ist dann ein Nicht-Zeuge. Andernfalls wäre a, wenn eine Kompontente $\not\equiv [\pm 1]$ ist, mit $[a_r^{2^s q}]_n \not\equiv [1]$ ein Zeuge der Nicht-Primheit. Es können zwei Fälle eintreten:

Fall 1: $r - d > 0$.
Also gilt mindestens für diese $r - d$ Kompontenten $s_i < s$ und deshalb $[a_r^{2^{s-1}q}]_{p_r} \equiv [1]$. Da $[-1]_n \equiv ([-1]_{p_1}, [-1]_{p_2}, \ldots, [-1]_{p_r})$ ist auf jeden Fall $[a]_n \not\equiv [-1]$. Genauso muss für den Fall $[a]_n \equiv [1]$ gelten, dass sämtliche $[a_i^{2^{s-1}q}]_{p_i} \equiv [1]_{p_i}$ sind. Jetzt sind $r - d$ Komponenten $\equiv [1]$ und die restlichen Komponenten sind dies jeweils mit Wahrscheinlichkeit $1/2$. Somit gilt mit Wahrscheinlichkeit $(1/2)^d$ für die d $[a] \in \mathbb{Z}_n$, dass $[a^{2^{s-1}q}] \equiv [1]$. Weil – wie wir gesehen haben – für sämtliche $a \in Z_n$ gilt $[a_i^{2^s q}]_{p_i} \equiv [1]$, sind mindestens $1 - (1/2)^d$ Zeugen. Wegen $d \geq 1$ ist die Wahrscheinlichkeit $\geq 1/2$.

Fall 2: $r = d$.
Hier gilt für sämtliche Komponenten $[a_i^{2^{s-1}q}]_{p_i} \equiv [1]_{p_i}$nur mit der jeweiligen Wahrscheinlichkeit $1/2$. Also ist $[a]_n \equiv [1]$ mit der Wahrscheinlichkeit $(1/2)^d$. Mit derselben Wahrscheinlichkeit ist $[a]_n \equiv [-1]$. Also ist die Wahrscheinlichkeit, einen Nicht-Zeugen zu finden, $(1/2)^d + (1/2)^d$. Wegen $d \geq 3$ ist die Wahrscheinlichkeit $\leq 1/4$ und die Wahrscheinlichkeit, einen Zeugen zu finden, ist somit $\geq 3/4$. □

PROPOSITION 2.5.2. *Sei $n = \prod_{i=1}^{r} p_i^{e_i}$ ungerade und nicht-prim, dann gilt*

$$\begin{aligned} \frac{\varphi(n) - \#MR(n)}{\varphi(n)} &= \begin{cases} \frac{1}{2^{r-1}} & , \quad n \text{ Carmichael} \\ \frac{1}{2^r} & , \quad \text{sonst} \end{cases} \\ &\leq \frac{1}{4} \quad \text{fuer } r \geq 2 \end{aligned}$$

BEWEIS. Zur Analyse des Miller-Rabin-Tests siehe auch [**Knuth2**, 4.5.4., Aufgabe (22)].

In der Vorlesung wurde folgender Beweis behandelt:

Falls n nicht-Carmichael dann gilt $F(n) \subseteq MR(n)$.
Sei also $n = p_1 \cdot \ldots \cdot p_r$ mit $r \geq 3$ Carmichaelzahl, d.h. $\lambda(n) \mid n-1$ und $\forall i : p_i - 1 \mid n-1$. Wir setzen wieder $n - 1 = q2^k$, wobei q ungerade ist, und $a_i \equiv a \mod p_i \forall i$. Ist $a \notin MR(n)$, dann gilt

$$\exists i, 0 \leq i < k : a^{q2^i} \equiv -(\pm 1)^{\delta_{i,0}} \mod n = \begin{cases} \pm 1 & , \quad i = 0 \\ -1 & , \quad i > 0 \end{cases}$$

für $i = 0$ ist das Vorzeichen der ± 1 festgelegt durch die letzte Komponente a_r. Sei $\sigma := -(\pm 1)^{\delta_{i,0}}$.
Fakt 1: *„Die Wahrscheinlichkeit, dass ein zufälliges $a \in_R \mathbb{Z}_{p_\nu^{e_\nu}}$ kein Zeuge der Nichtprimheit, also $\notin MR(n)$ ist, ist $\leq 1/2$."*

Für $\nu = 1, \ldots, r-1$ und $a_\nu \in_R \mathbb{Z}_{p_\nu^{e_\nu}}$ soll also gelten

$$\mathrm{Ws}_{a_\nu}\left[a_\nu^{q2^i} \equiv \sigma \mod p_\nu^{e_\nu}\right] \leq \frac{1}{2}.$$

Zum Beweis führen wir den Gruppenhomomorphismus $H_{i,\nu}$ ein mit

$$\begin{aligned} H_{i,\nu} : \quad \mathbb{Z}^*_{p_\nu^{e_\nu}} &\to \mathbb{Z}^*_{p_\nu^{e_\nu}} \\ a_\nu &\mapsto a_\nu^{q2^i} \mod p_\nu^{e_\nu}. \end{aligned}$$

Für $\sigma = \pm 1$ gilt

$$\sigma = -1 : \qquad \#H_{i,\nu}^{-1}(\sigma) = \left|\left\{a \in \mathbb{Z}^*_{p_\nu^{e_\nu}} \mid a^{q2^i} \equiv -1 \mod p_\nu^{e_\nu}\right\}\right| \leq \#H_{i,\nu}^{-1}(1)$$

$$\sigma = 1, i = 0 : \qquad \#H_{i,\nu}^{-1}(\sigma) = \left|\left\{a \in \mathbb{Z}^*_{p_\nu^{e_\nu}} \mid a^{q} \equiv 1 \mod p_\nu^{e_\nu}\right\}\right| \leq \frac{p_\nu^{e_\nu - 1}(p_\nu - 1)}{2}.$$

Letzteres, weil es in $\mathbb{Z}^*_{p_\nu^{e_\nu}}$ ein Element a_ν mit Ordnung

$$\underbrace{\lambda\left(p_\nu^{e_\nu}\right)}_{\text{Maximalordnung}} = p_\nu^{e_\nu - 1}(p_\nu - 1).$$

gibt und wegen

$$\underbrace{(p_\nu - 1)}_{\text{gerade}} \nmid \underbrace{q}_{\text{ungerade}}$$

folgt, dass es $\forall q_\nu < \varphi\left(p_\nu^{e_\nu}\right) : a_\nu^{q_\nu} \not\equiv 1 \mod p_\nu^{e_\nu}$. Also ist $H_{i,\nu}^{-1}(\sigma)$ echte Untergruppe von $\mathbb{Z}^*_{p_\nu^{e_\nu}}$ und somit höchstens halb so groß.

Fakt 2: *„Die Wahrscheinlichkeit, dass ein zufälliges $a \in_R \mathbb{Z}_n$ ein Zeuge der Nichtprimheit ist, ist $\geq 1 - 1/2^{r-1}$"*
Es gilt:

$$\mathrm{Ws}_{a_1,\ldots,a_{r-1}}\left[a_\nu^{q2^i} \equiv \sigma \mod p_\nu^{e_\nu} \text{ fuer } \nu = 1,\ldots,r-1\right] \leq \frac{1}{2^{r-1}}.$$

Zum Beweis überlege man sich, dass die Ereignisse für $\nu = 1,\ldots,r-1$ unabhängig sind. □

Algorithm 11 Miller-Rabin-Test

EINGABE: $n \in \mathbb{N}$ ungerade.
AUSGABE: n prim oder n zusammengesetzt.

(1) Berechne $n-1 = q2^r$ mit q ungerade
(2) Wähle $a \in_R \mathbb{Z}_n^*$ zufällig
(3) **do**
  (a) Berechne sukzessive $a_0 \equiv a^q \mod n$, $a_1 \equiv a_0^2 \mod n$, …, $a_k \equiv a_{k-1}^2 \mod n$
(4) **until** $a_k \equiv 1 \mod n$ or $k = r$
(5) **if** $k = 0$ **then**
  (a) **return** n prim
(6) **elseif** $k = r$ and $a_k \not\equiv 1 \mod n$ **then**
  (a) **return** n zusammengesetzt
(7) **elseif** $a_{k-1} \not\equiv -1 \mod n$ **then**
  (a) **return** n zusammengesetzt
(8) **else**
  (a) **return** n prim

Korrektheit: Ist n prim, so ist die Ausgabe des Algorithmus nach Satz 2.5.1 in jedem Fall „n prim". In dem Fall, dass n zusammengesetzt ist, ist die Ausgabe nach dem gleichen Satz mit einer Wahrscheinlichkeit $\geq 3/4$ korrekt.

Laufzeit: Die Laufzeit wird von der Schleife in Schritt 3 dominiert, in der die Werte a_0 bis a_k berechnet werden. Da $k \leq r \leq \log n$ gilt und jede Quadratur Zeit $O\left(\log^2 n\right)$ benötigt, ist die Laufzeit des Algorithmus $O\left(\log^3 n\right)$.

BEWEIS. Wie auch der Primzahltest SS ist MR ein Monte-Carlo Test mit einseitigem Fehler. Die Fehlerwahrscheinlichkeit kann durch k-maliges Wiederholen des Tests mit unabhängiger Wahl der $a \in_R \mathbb{Z}_n^*$ beliebig klein gemacht werden (nämlich auf $1 - 4^{-k}$ gedrückt werden - für die Aussage, dass n prim ist). □

2.6. Vergleich von MR und SS

Wie bisher gezeigt wurde sind die erwarteten Laufzeiten von MR und SS beide $O\left(\log^3 n\right)$, trotzdem ist der Algorithmus von MR schneller als der von SS. Ausserdem ist die Fehlerwahrscheinlichkeit von MR geringer als die von SS. Zudem ist die Menge $\overline{MR(n)}$ kleiner als $\overline{SS(n)}$ und es gilt sogar $\overline{MR(n)} \subseteq \overline{SS(n)} \Leftrightarrow SS(n) \subseteq MR(n)$ (in dem Fall, dass n zusammengesetzt ist und MR eine falsche Antwort gibt, falls also $a \in \overline{MR(n)}$, dann gibt SS genauso eine falsche Antwort, da $a \in \overline{MR(n)} \Rightarrow a \in \overline{SS(n)}$). Für den Beweis wird aber noch ein Lemma benötigt:

LEMMA. *Sei $p \neq 2$ eine Primzal und $e \in \mathbb{N}$. Für jede Zahl $a \in \mathbb{Z}$ gilt*

$$\exists i \in \mathbb{N}: \quad ord_{p^e}(a) = p^i ord_p(a).$$

BEWEIS. Sei $d = \mathrm{ord}_p(a)$, also $a^d = kp+1$, $k \in \mathbb{N}$. Dann gilt für jedes $r \in \mathbb{N}$ nach dem Binomischen Lehrsatz

$$a^{dr} = (kp+1)^r = \sum_{i=0}^{r} \binom{r}{i} (kp)^i.$$

In dieser Summe sind die Terme $i \geq e$ gleich 0 modulo p^e. Sei nun

$$r = p^e \max\left\{ p^j \mid p^j \text{ teilt } e! \right\}.$$

Dann gilt $\binom{r}{i} \equiv 0 \mod p^e$, da

$$\binom{r}{i} = \frac{r(r-1)\cdots(r-i+1)}{i!}$$

im Zähler r, und somit eine Potenz der Form p^{e+s} steht. Durch die Wahl von r enthält der Nenner aber nur die Potenz p^s, so dass sich die p^s im Zähler herauskürzen. Es bleibt also

$$a^{dr} = (kp+1)^r \equiv \sum_{i=0}^{r} \binom{r}{i} (kp)^i \equiv 1 \mod p^e$$

denn alle Summanden sind 0, ausser für $i = 0$.

Wie man sieht gilt also

$$\mathrm{ord}_{p^e}(a) \mid dr \mid \underbrace{\mathrm{ord}_p(a)}_{=d} p^r$$

und ausserdem auch

$$\mathrm{ord}_p(a) = d \mid \mathrm{ord}_{p^e}(a).$$

Aus diesen beiden Relationen folgt nun

$$\exists i \in \mathbb{N}: \quad \mathrm{ord}_{p^e}(a) = p^i \mathrm{ord}_p(a).$$

□

Kommen wir nun zu dem Beweis der Aussage $a \in \overline{MR(n)} \Rightarrow a \in \overline{SS(n)}$ (wenn MR die falsche Antwort gibt, dann auch SS):

PROPOSITION 2.6.1. *Sei $n \in \mathbb{N}$ ungerade. Dann gilt $SS(n) \subseteq MR(n)$.*

BEWEIS. Wenn n prim ist, so gilt $\mathbb{Z}_n^* = SS(n) = MR(n)$.

Ist n zusammengesetzt mit $n = p_1^{e_1} \cdot \ldots \cdot p_s^{e_s}$ und $n-1 = 2^r d$ mit d ungerade. Dann ist zu zeigen

$$a \in \overline{MR(n)} \quad \Rightarrow \quad a \in \overline{SS(n)}.$$

Ist $a \in MR(n)$, so gilt $a^d \equiv 1 \mod n$ oder $a^{2^{k-1}d} \equiv -1 \mod n$ für $1 \leq k \leq r$. Dann gilt auch $2^k \mid \mathrm{ord}_n(a)$ und $2^{k+1} \nmid \mathrm{ord}_n(a)$ (falls $a^d \equiv 1 \mod n$, so setzen wir $k = 0$). Dafür schreiben wir $2^k \| \mathrm{ord}_n(a)$. Dann gilt auch $2^k \| \mathrm{ord}_{p_j^{e_j}}(a)$ für $j = 1, \ldots, s$: Da gilt

$$\exists i \in \mathbb{N}: \quad p_j^i \mathrm{ord}_{p_j}(a) = \mathrm{ord}_{p_j^{e_j}}(a)$$

(Beweis siehe oben) folgt aus $2^k \| \mathrm{ord}_n(a)$, dass gilt $2^k \mid \left(p_j - 1\right)$ für alle j.

Sei jetzt k_j so gewählt, dass $2^{k_j} \| \left(p_j - 1\right)$ gilt. Ausserdem sei $J = \left\{ j \mid k = k_j \right\}$. Für alle $j \notin J$ gilt $k_j \geq k+1$. Schliesslich sei

$$m = \sum_{j \in J} e_j.$$

Damit gilt

$$p_j \quad \equiv \quad \begin{cases} 2^k + 1 \mod 2^{k+1} & , j \in J \\ 1 \mod 2^{k+1} & , j \notin J \end{cases}$$

also

$$n \equiv \left(2^k + 1\right)^m \equiv \sum_{i=0}^{m} \binom{m}{i} 2^{ik} \equiv m2^k + 1 \mod 2^{k+1}.$$

Daraus schliessen wir

$$\begin{aligned} 2^k \| n-1 \quad \text{fuer} \quad m \text{ ungerade} \\ 2^{k+1} \mid n-1 \quad \text{fuer} \quad m \text{ gerade} \end{aligned}$$

Da ja wie oben schon gezeigt gilt $2^k \| \text{ord}_{p_j^{e_j}}(a)$ für alle j folgt

$$\begin{aligned} m \text{ gerade} \quad &\Rightarrow \quad a^{\frac{n-1}{2}} \equiv a^{2^k \cdot d} \equiv 1 \mod p_j^{e_j} \quad \forall j \\ m \text{ ungerade} \quad &\Rightarrow \quad a^{\frac{n-1}{2}} \equiv a^{2^{k-1} d} \equiv -1 \mod p_j^{e_j} \quad \forall j. \end{aligned}$$

Mit dem CRT gilt dann

$$\begin{aligned} m \text{ gerade} \quad &\Rightarrow \quad a^{\frac{n-1}{2}} \equiv 1 \mod n \\ m \text{ ungerade} \quad &\Rightarrow \quad a^{\frac{n-1}{2}} \equiv -1 \mod n \end{aligned}$$

oder einfach

$$a^{\frac{n-1}{2}} \quad \equiv \quad (-1)^m \mod n.$$

Jetzt ist noch zu zeigen, dass $\binom{a}{n} = (-1)^m$ ist: Sei g_j erzeugendes Element für $\mathbb{Z}_{p_j}^*$. Dann ist $a \equiv g_j^i \mod p_j$ für ein ungerades i genau dann, wenn $2^{k_j} \| \text{ord}_{p_j}(a)$ und damit $2^{k_j} \| p_j - 1$:

$$\begin{aligned} a &\equiv g_j^i \mod p_j \\ \Rightarrow \quad \text{ord}_{p_j}(a) &= \text{ord}_{p_j}\left(g_j^i\right) \\ &= \frac{p_j - 1}{i} \\ \Rightarrow \quad i &= \frac{p_j - 1}{\text{ord}_{p_j}(a)}. \end{aligned}$$

Da oben schon gezeigt wurde, dass gilt $2^k \mid \text{ord}_{p_j}(a)$ und $k \leq k_j$ ist, folgt, dass $a = g^i$ mit ungeradem i nur dann der Fall sein kann, wenn gilt $k_j = k$, was gleichbedeutend ist mit $\left(\frac{a}{p_j}\right) = -1$. Damit erhalten wir

$$\left(\frac{a}{n}\right) = \prod_{j=1}^{s} \left(\frac{a}{p_j}\right)^{e_j} = \prod_{j \in J} \left(\frac{a}{p_j}\right)^{e_j} = \prod_{j \in J} (-1)^{e_j} = (-1)^m.$$

□

2.7. Bezeichnungen aus der Komplexitätstheorie

Polynomialzeit Algorithmus

DEFINITION. Ein Algorithmus

$$\begin{array}{rccc} AL: & x & \mapsto & AL(x) \\ & \{0,1\}^* & \to & \{0,1\}^* \end{array}$$

ist **polynomialzeit**, wenn die Anzahl $T_{AL}(x)$ der Turingmaschinenschritte beschränkt ist mit

$$T_{AL}(x) = O\left(|x|^k\right) \quad , k \text{Konstante.}$$

$AL: \mathbb{N} \to \mathbb{N}$ ist polynomialzeit, wenn $T_{AL}(n) = O\left(\left(\log_2 n\right)^k\right)$.
Die Klasse aller Sprachen $L \subseteq \{0,1\}^*$, deren charakteristische Funktion χ_L in Polynomialzeit berechenbar ist, wird mit ***P*** bezeichnet.

Bis heute ist es offen, ob Prim $\in P$ gilt.

NP

DEFINITION. ***NP*** ist die Klasse aller Sprachen L, für die eine polynomialzeit Relation R und ein $k \in \mathbb{N}$ existiert, so dass gilt

$$x \in L \quad \Leftrightarrow \quad \exists y \in \{0,1\}^* : |y| < |x|^k \text{ und } (x,y) \in R$$

Ein Element aus NP ist also eine Sprache, für deren Elemente es einen kurzen und effizienten Beweis der Sprachzugehörigkeit gibt (nämlich die polynomialzeit Relation R).

Probabilistisch Polynomialzeit

DEFINITION. Eine Sprache $L \subseteq \{0,1\}^*$ ist in **probalistisch Polynomialzeit** entscheidbar, wenn es ein $k \in \mathbb{N}$ und ein polynomialzeit Test gibt mit

$$L = \left\{ x \in \{0,1\}^* \mid \underbrace{\#\left\{y \in \{0,1\}^* \mid \text{Test}(x,y) = 1, |y| \leq |x|^k\right\}}_{>0} \right\}$$

und

$$\#\left\{y \in \{0,1\}^* \mid \text{Test}(x,y) = 1, |x| \leq |x|^k\right\} > 0 \quad \Leftrightarrow \quad \#\{\} \geq \frac{2^{|x|^k}}{2}$$

Sei R die Klasse dieser Sprachen: $P \subsetneq R \subsetneq NP$.

COROLLARY 2.7.1. *$\mathbb{N} \setminus Prim \in R$ (Menge der zusammengesetzten Zahlen) ist probabilistisch Polynomialzeit.*

BEWEIS. Nehme als Test den MR- oder den SS-Test. Es gilt $k = 1$, Laufzeit linear in der Anzahl der arithmetischen Operationen modulo n (nämlich $O\left(\log_2 n\right)$ Operationen). Die Anzahl der der Bitoperationen modulo n hat eine Laufzeit von $O\left(\left(\log_2 n\right)^3\right)$. □

PROPOSITION 2.7.2. *(Adleman, Huang 1987) Prim $\in R$.*

Dieser Satz besagt, dass man Primheitsbeweise erwürfeln kann und wird nicht bewiesen.

PROPOSITION 2.7.3. *Sei Prim $:= \left\{x \in \{0,1\}^* \mid x \text{ ist Binaerdarstellung einer Primzahl}\right\}$. Dann gilt Prim $\in NP$.*

BEWEIS. Da $\mathbb{Z}_p^*$ zyklisch ist, falls p prim ist und ein erzeugendes Element a der Ordnung $p-1$ besitzt, genügt es, wenn man als Beweis ein Element a vorweisen kann, dessen Ordnung $\text{ord}_p(a) = p-1$ ist. Ein Beweis für die Primheit von p könnte also ein a sein, für das gilt

$$\begin{aligned} a^{n-1} &\equiv 1 \mod n \\ \forall q \mid n-1 : a^{\frac{n-1}{q}} &\not\equiv 1 \mod n \end{aligned}$$

da ein solches a für q prim genau die Ordnung $n-1$ hat. Wäre dies nicht der Fall, so wäre $n = c \cdot \text{ord}_n(a)$ und damit $a^{\frac{n-1}{q}} \equiv a^{\frac{c}{q}\text{ord}_n(a)} \equiv 1 \mod n$. Widerspruch.

Ein schneller Beweis (in polynomial Zeit) der Primheit von n funktioniert so, dass man $n-1$ in Faktoren zerlegt, und für diese wieder die Primheit zeigt und die Relation $a^{\frac{n-1}{p_i}} \not\equiv 1 \mod n$ (das Faktorisieren wird im anschliessenden Kapitel behandelt werden). □

KAPITEL 3

Faktorisierung

3.1. RSA-Schema, Faktorisierung

Sei $n = p_1 \cdot p_2$, p_1, p_2 prim ≥ 3, $p_1 \neq p_2$, $\varphi(n) = (p_1 - 1)(p_2 - 1)$.

Sei $\mathrm{ggT}(e, \varphi(n)) = 1$, mit $1 < e < \varphi(n)$ und $d := e^{-1} \mod \varphi(n)$ (kann mit dem euklidischen Algorithmus bestimmt werden).

Nach dem *Satz von Bezout* gilt:

$$e \cdot a + \varphi(n) \cdot b = \mathrm{ggT}(e, \varphi(n)) = 1 \quad \text{und} \quad e \cdot a = 1 \mod \varphi(n) \Leftrightarrow a = e^{-1} \mod \varphi(n).$$

Nach *Legendre* gilt

$$x^{ed} = x^{1+v\varphi(n)} = x \mod n \quad \text{fuer x} \in \mathbb{Z}_n^*, \text{ sogar } \forall x \in \mathbb{Z}_n \cong [0, n-1[.$$

Das *Verschlüsseln* von Nachrichten $m \in \mathbb{Z}_n$ funktioniert dann wie folgt:

$$E : m \mapsto m^e \mod n,$$

und *entschlüsselt* wird mittels

$$D : c \mapsto c^d \mod n.$$

Nach Legendre gilt: $D \circ E = id_{\mathbb{Z}_n}$ und $e \circ D = id_{\mathbb{Z}_n}$. Dabei unterscheiden wir die Schlüssel:

- *geheimer Schlüssel:* $p_1, p_2, \varphi(n), d$
- *öffentlicher Schlüssel:* n, e

Die Sicherheit des RSA-Verfahrens beruht auf der Schwierigkeit, die Zahl n in zwei Primfaktoren zu zerlegen.

LEMMA 3.1.1. *Folgende Probleme sind pol. Zeit äquivalent:*

1. zerlege n, $n \mapsto p_1 \cdot p_2$.

2. berechne $\varphi(n)$, $n \mapsto (p_1 - 1)(p_2 - 1)$.

BEWEIS. „2. $\leq_{pol.}$ 1.": Angenommen, $n \mapsto p_1 p_2$ kann in $T(|n|)$-Schritten berechnet werden, dann berechne $\varphi(n) = (p_1 - 1)(p_2 - 1)$.

„1. $\leq_{pol.}$ 2.": Aus $\varphi(n)$ erhält man zwei lineare Gleichungen

$$\begin{aligned} p_1 + p_2 &= n - \varphi(n) + 1 \\ p_1 - p_2 &= \sqrt{(p_1 + p_2)^2 - 4n} \end{aligned}$$

aus diesen beiden Gleichungen kann man nun p_1 und p_2 berechnen. □

3.2. Grundlagen der Faktorisierung

LEMMA 3.2.1. *Sei $n = \prod_{i=1}^{r} p_i^{e_i}$ ungerade, dann gilt $|QR_n| = \frac{\varphi(n)}{2^r}$.*

BEWEIS. Es ist $\mathbb{Z}_{p^e}^* = \left\{g, g^2, \ldots, g^{p^{e-1}(p-1)}\right\}$ zyklisch, also ist $\left|QR_{p^e}\right| = \frac{\varphi(p^e)}{2}$ (die Hälfte der Elemente, nämlich die mit geraden Exponenten, sind also quadratische Reste). Mit Anwendung des Chin. Restsatzes (s.u.) ergibt sich also $|QR_n| = \frac{\varphi(n)}{2^r}$ und der Rest folgt von selbst.

$$\begin{array}{ccccccc} \mathbb{Z}_n^* & \cong & \mathbb{Z}_{p_1^{e_1}} & \times & \ldots & \times & \mathbb{Z}_{p_r^{e_r}} \\ \supseteq & & \supseteq & & & & \supseteq \\ QR_n & \cong & QR_{p_1^{e_1}} & \times & \ldots & \times & QR_{p_r^{e_r}} \end{array}$$

$\rightarrow$ die $QR_{p_i^{e_i}}$ sind jeweils nur halb so groß wie die $\mathbb{Z}_{p_i^{e_i}}$. □

LEMMA 3.2.2. *Sei $n = \prod_{i=1}^{r} p_i^{e_i}$. Dann ist*

$$\begin{array}{rcl} Qu : \mathbb{Z}_n^* & \rightarrow & QR_n \\ x & \mapsto & x^2 \end{array}$$

eine $(2^r, 1)$-Abb. und ein Gruppenhomomorphismus mit $\left|Qu^{-1}(x)\right| = 2^r$.

Sei $Sqrt =_{def} Qu^{-1}$.

BEWEIS. Wie oben gezeigt wurde gilt $|QR(n)| = \varphi(n)/2^r$, es sind also in $\mathbb{Z}_n^*$ genau 2^r Zahlen *keine* Quadrate. Die Funktion *Qu* bildet nun genau diese „nicht-Quadrate" auf Quadrate ab, und tut dies unter Erhaltung der Gruppenstruktur mit $(ab)^2 = Qu(a \cdot b) = Qu(a)\,Qu(b) = a^2 b^2 = (ab)^2$. □

LEMMA 3.2.3. *Sei $n = \prod_{i=1}^{r} p_i^{e_i}$ ungerade, $x^2 = y^2 \mod n$ mit $x \neq \pm y \mod n$.*
Dann gilt:

$$(x+y)(x-y) = kn \quad mit\, k \in \mathbb{Z} \quad \Rightarrow \quad ggT(x \pm y, n) \neq \{1, n\}$$

sind nicht-triviale Faktoren von n. Wenn, wie bei RSA, $n = pq$ mit p und q prim, dann ist $ggT(x+y,n) \cdot ggT(x-y,n) = n$.

BEWEIS. Es gilt

$$x \not\equiv y \mod n \quad \Leftrightarrow \quad x \neq \pm y + kn,\, x+y \notin n\mathbb{Z},\, x-y \notin n\mathbb{Z}$$

$\Rightarrow$ weder $(x+y)$ noch $(x-y)$ haben n als Teiler, deren Produkt wird jedoch von n geteilt
$\Rightarrow$ ggT $(x \pm y, n) \notin \{1, n\}$. □

Wir können n also faktorisieren, indem wir zwei Quadratwurzeln einer Zahl mod n finden, die sich nicht nur durch das Vorzeichen unterscheiden. Wie kann man aber die Quadratwurzel in $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ berechnen?

Sei $\varphi(p^e) = q2^k$. Dann ist $\left(\mathbb{Z}_{p^e}^*\right)^{2^k} = \left\{a^{2^k} \mid a \in \mathbb{Z}_{p^e}^*\right\}$ eine zyklische Untergruppe von $\mathbb{Z}_{p^e}^*$. Mit $\langle g \rangle = \mathbb{Z}_{p^e}^*$ ist jedes $a \in \mathbb{Z}_{p^e}^*$ darstellbar $a = g^x$. Damit lässt sich jedes Element $a' \in \left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ mit $a' = a^{2^k}$ darstellen als $a' = (g^x)^{2^k} = g^{x2^k}$. Da x nur q verschiedene

Werte annehmen kann (wegen $\varphi(p^e) = q2^k$) gibt es also q Elemente in $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ $\Rightarrow$ $\left|\left(\mathbb{Z}_{p^e}^*\right)^{2^k}\right| = q$.

In $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ ist das Quadrieren nach Lemma 3.2.2 eine $(1,1)$-Abbildung und Gruppenisomorphismus, also werden wir in $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ die Wurzel ziehen können.

Die Wurzel einer Potenz wird gezogen, indem der Exponenten halbiert wird. Also muss im Exponenten mit 2^{-1} multipliziert werden. Da $\varphi\left(\left(\mathbb{Z}_{p^e}^*\right)^{2^k}\right) = q$ ist, potenzieren wir zur Berechnung der Wurzel von $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ mit $2^{-1} \mod q$:

$$\left(a^{2^{-1}} \mod q\right)^2 \equiv \left(a^{\frac{1+q}{2}}\right)^2 \equiv a^{1+q} \equiv (*)$$

Da $\operatorname{ord}_{\left(\mathbb{Z}_{p^e}^*\right)^{2^k}}(a)$ die Gruppenordnung q teilt, können wir für q auch $l \cdot \operatorname{ord}_{\left(\mathbb{Z}_{p^e}^*\right)^{2^k}}(a)$ schreiben:

$$(*) \equiv a^{1+l\cdot \operatorname{ord}_{\left(\mathbb{Z}_{p^e}^*\right)^{2^k}}(a)} \equiv a \cdot \left(\underbrace{a^{\operatorname{ord}_{\left(\mathbb{Z}_{p^e}^*\right)^{2^k}}(a)}}_{\equiv 1 \text{ wg. Fermat}}\right)^l \equiv a \mod \left(\mathbb{Z}_{p^e}^*\right)^{2^k}.$$

Die Multiplikation eines $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ mit sich selbst erfordert $\left(\log_2 p^e\right)^2$ Bitoperationen. Die Potenzierung mit Exponenten $O(q)$ lässt sich mit Hilfe der schnellen Exponentiation in Zeit $O\left(\log_2 q\right)$ bewerkstelligen. Daraus folgt:

COROLLARY 3.2.4. *Sei $\varphi(p^e) = q2^k$, dann ist die Berechnung*

$$\begin{aligned} sqrt: \left(\mathbb{Z}_{p^e}^*\right)^{2^k} &\to \left(\mathbb{Z}_{p^e}^*\right)^{2^k} \\ a &\mapsto a^{2^{-1}} \mod q \end{aligned}$$

in Polynomialzeit möglich, nämlich in $O\left(\left(\log_2 q\right) \cdot \underbrace{\left(\log_2 p^e\right)^2}_{\text{Multiplikationen}}\right)$ Bitoperationen.

Sei immer noch $\varphi(p^e) = q2^k$. Wir können jetzt ein paar mathematische Gegebenheiten sammeln, die sich dann zu einem Algorithmus zusammenfügen lassen, mit dem man in $\mathbb{Z}_{p^e}^*$ Wurzeln ziehen kann.

FACT 3.2.5. *Angenommen wir haben $b \in QNR_{p^e}$, dann ist $b^{\frac{\varphi(p^e)}{2}} \equiv -1 \mod p^e$, denn*

$$b^{\frac{\varphi(p^e)}{2}} \equiv b^{\frac{p^{e-1}(p-1)}{2}} \equiv \left(\underbrace{b^{\frac{p-1}{2}}}_{b \in QNR}\right)^{p^{e-1}} \equiv (-1)^{\overbrace{p^{e-1}}^{ungerade}} \equiv -1 \mod p^e.$$

Wir werden ein solches $b \in QNR_{p^e}$ willkürlich auswählen und im Algorithmus dann benutzen, wenn wir ein $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i}$ nach $\left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}}$ transponieren wollen.

FACT 3.2.6. *Für ein* $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i} = \left\{c^{2^i} \mid c \in \mathbb{Z}_{p^e}^*\right\}$ *gilt*

$$a^{\varphi(p^e)\cdot 2^{-i}} \equiv \left(c^{2^i}\right)^{\varphi(p^e)\cdot 2^{-i}} \equiv c^{\varphi(p^e)} \equiv 1 \mod p^e .$$

Da $\left(\mathbb{Z}_{p^e}^*\right)^{2^i}$ *eine Untergruppe von* $\mathbb{Z}_{p^e}^*$ *ist, gilt auch* $a \in \mathbb{Z}_{p^e}^*$. *Sei* $\mathbb{Z}_{p^e}^* = \langle g \rangle$ *und* $a = g^x$ *mit* $x \in \mathbb{Z}_{\varphi(p^e)}$. *Dann folgt*

$$\begin{aligned} & a^{\varphi(p^e)2^{-i}} && \equiv && 1 \mod p^e \\ \Leftrightarrow \quad & (g^x)^{\varphi(p^e)2^{-i}} && \equiv && 1 \mod p^e \\ \Leftrightarrow \quad & g^{x2^{-i}\varphi(p^e)} && \equiv && 1 \mod p^e \\ \Leftrightarrow \quad & x\tfrac{\varphi(p^e)}{2^i} && \equiv && 0 \mod \varphi(p^e). \end{aligned}$$

Damit $x\frac{\varphi(p^e)}{2^i}$ *ein Vielfaches von* $\varphi(p^e)$ *ist, muss* x *ein Vielfaches von* 2^i *sein:*

$$\Rightarrow \quad 2^i \mid x \quad \Rightarrow \quad a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i}.$$

Wir haben also einen Test gefunden, mit dem wir überprüfen können, ob $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i}$ ist.

Wir können das $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i}$ nach $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ transponieren, indem wir nach einem i mit $0 \le i < k$ suchen, so dass a eine 2^i-te Potenz, aber keine 2^{i+1}-te aus $\mathbb{Z}_{p^e}^*$ ist. Dann muss nämlich eine Transponation einsetzten, so dass a so „verändert" wird, dass es zu einer 2^{i+1}-ten Potenz aus $\mathbb{Z}_{p^e}^*$ wird. Ein Werkzeug zur Suche nach einem solchen i liefert folgende Tatsache.

FACT 3.2.7. *Ist* a *eine* 2^i*-te Potenz, aber keine* 2^{i+1}*-te eines Elements aus* $\mathbb{Z}_{p^e}^*$, *also*

$$\begin{aligned} a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i} \quad & \Leftrightarrow \quad \textit{es gibt ein } c \in \mathbb{Z}_{p^e}^*, \textit{ so dass } a = c^{2^i}, \\ \textit{und} \quad a \notin \left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}} \quad & \Leftrightarrow \quad \textit{es gibt kein } d \in \mathbb{Z}_{p^e}^*, \textit{ so dass } a = d^{2^{i+1}}, \end{aligned}$$

so gilt

$$a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i} \setminus \left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}} \quad \Leftrightarrow \quad a^{\varphi(p^e)\cdot 2^{-i-1}} \equiv -1 \mod p^e,$$

denn wegen

$$a^{\frac{\varphi(p^e)}{2^{i+1}}} \equiv \left(c^{2^i}\right)^{\frac{\varphi(p^e)}{2^{i+1}}} \equiv c^{\frac{\varphi(p^e)}{2}} \equiv \left(c^{\varphi(p^e)}\right)^{\frac{1}{2}} \equiv 1^{\frac{1}{2}} \mod p^e$$

muss die Wurzel $\equiv \pm 1$ *sein. Angenommen, die Wurzel ist* $\equiv 1$, *also*

$$\left(c^{\varphi(p^e)}\right)^{\frac{1}{2}} \equiv \left(c^{\frac{1}{2}}\right)^{\varphi(p^e)} \equiv 1 \mod p^e \quad \Rightarrow \quad c^{\frac{1}{2}} \in \mathbb{Z}_{p^e}^*,$$

dann gäbe es ein $d \in \mathbb{Z}_{p^e}^*$, *so dass* $c = d^2$ *und damit* $a = c^{2^i} = \left(d^2\right)^{2^i} = d^{2^{i+1}}$, *was im Widerspruch zu* $a \notin \left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}}$ *steht. Also ist die Wurzel* $\equiv -1$.

Wir haben also eine Möglichkeit gefunden, nach Stellen zu Suchen, an denen „Transponationsbedarf" besteht. Das Transponieren selbst ist nicht schwer:

FACT 3.2.8. *Multiplizieren wir ein* $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i} \setminus \left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}}$ *und die* 2^i*-te Potenz eines* $b \in QNR_{p^e}$*, so erhalten wir ein Element aus* $\left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}}$*, da gilt*

$$\left(ab^{2^i}\right)^{\varphi(p^e)2^{-i-1}} \equiv (-1) \cdot b^{\varphi(p^e)/2} \equiv (-1)^2 \equiv 1 \mod p^e.$$

Wie können also die Wurzel einer Zahl $a \in \mathbb{Z}_{p^e}^*$ ziehen, indem wir sie nach $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ hochtransponieren und dort die Wurzel ziehen, was sich als sehr einfach erwiesen hat. Das Ergebnis muss dann wieder nach $\mathbb{Z}_{p^e}^*$ runtertransponiert werden. Damit haben wir einen Algorithmus zur Berechnung der Quadratwurzel in $\mathbb{Z}_{p^e}^*$.

Algorithm 12 Berechnung der Quadratwurzel in $\mathbb{Z}_{p^e}^*$

EINGABE: $a \in QR_{p^e}$, $\varphi(p^e) = q2^k$ und $b \in QNR_{p^e}$ (kann in prob. pol. Zeit gefunden werden).
AUSGABE: $\sqrt{a} \mod p^e$.

(1) $\overline{a} := a$.
(2) **** COUNTER zählt die Mult. mit* b^{2^i} *zum späteren runtertransp. ****
COUNTER $= 1$.
(3) **** Solange* $\overline{a}$ *noch nicht nach* $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ *hochtransponiert ist ****
while $\overline{a}^q \not\equiv 1 \mod p^e$ **do**
(a) Bestimme das kleinste i mit $1 \leq i \leq k$, für das gilt
$$\overline{a}^{\varphi(p^e)2^{-i-1}} \equiv -1 \mod p^e$$
also $a \in \left(\mathbb{Z}_{p^e}^*\right)^{2^i} \setminus \left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}}$.
(b) $\overline{a} := \overline{a}b^{2^i} \in \left(\mathbb{Z}_{p^e}^*\right)^{2^{i+1}}$.
(c) **** In (4) wird die Wurzel gezogen, deshalb Mult. mit* $b^{2^{i-1}}$ *****
COUNTER $=$ COUNTER $\cdot\, b^{2^{i-1}}$
(4) **** Wurzelziehen in* $\left(\mathbb{Z}_{p^e}^*\right)^{2^k}$ *durch Exponentiation mit* $\frac{q+1}{2}$ *****
$\overline{a} :\equiv \overline{a}^{\frac{q+1}{2}} \mod \left(\mathbb{Z}_{p^e}^*\right)^{2^k}$.
(5) **** Jetzt muss das Ergebnis wieder nach* $\mathbb{Z}_{p^e}^*$ *transponiert werden ****
$\overline{a} := \overline{a} \cdot \text{COUNTER}^{-1}$.
(6) **return** $\pm\overline{a}$.

3.2.1. Berechnung der Quadratwurzel in $\mathbb{Z}_{p^e}^*$.

COROLLARY 3.2.9. *Die Funktion sqrt* : $QR_{p^e} \to Z_{p^e}^*$ *ist polynomialzeit, sofern ein* $b \in QNR_{p^e}$ *gegeben ist.*

BEWEIS. Diese Aussage gilt wegen dem oben angegebenen Algorithmus. Eine Laufzeitanalyse zeigt, dass die Schleife in Schritt 3 maximal k-mal durchlaufen wird. Alle anderen Schritte sind sowieso polynomialzeit. □

PROPOSITION 3.2.10. *Zu $n \in \mathbb{N}$, $n \notin Prim$, $n \neq p^e$ ungerade, sind folgende Probleme prob. pol. Zeit äquivalent:*

(1) Zerlege n, $n \mapsto \prod_{i=1}^{r} p_i^{e_i}$
(2) Berechne $sqrt : QR_n \to \mathbb{Z}_n^*$ (mit einem Quadratwurzelalgorithmus können wir eine Zahl also zerlegen)

BEWEIS. „$(2) \leq_{pol.} (1)$": Es ist $n = \prod_{i=1}^{r} p_i^{e_i}$ gegeben. Erzeuge $a_i \in QR_{p_i^{e_i}}$, $i = 1, \ldots, r$ (das geht in prob. pol. Zeit, da die Hälfte der Elemente einer zyklischen Gruppen – und $\mathbb{Z}^*_{p_i^{e_i}}$ ist eine zyklische Gruppe – quadratische Reste sind. Man muss sich also nur eine Zahl $a_i \in_R \mathbb{Z}^*_{p_i^{e_i}}$ zufällig wählen und testen, ob $a_i^{p_i^{e_i-1}(p_i-1)/2} \equiv 1 \mod p_i^{e_i}$ ist, was in pol. Zeit geht).
Da es nach (1) möglich ist n zu zerlegen, berechne man ein $a \in \mathbb{Z}_n^*$ aus $a_i = a \mod p_i^{e_i}$. Wir wissen nun, dass $a \in QR_{p_i^{e_i}}$, da ja $a_i \in QR_{p_i^{e_i}}$ ist. Damit ist die Abbildung $a \mapsto sqrt(a_i) \in \mathbb{Z}^*_{p_i^{e_i}}$ in Polynomialzeit möglich, denn zum Ziehen der Wurzel aus einem QR müssen wir diesen nur mit der Hälfte der Gruppenordnung $\frac{\varphi(p_i^e i)}{2}$ potenzieren.

$$\begin{array}{ccccccc} (& sqrt(a_1) & , \ \ldots \ , & sqrt(a_r) &) & \mapsto & sqrt(a) \\ & \in & & \in & & & \in \\ & \mathbb{Z}^*_{p_1^{e_1}} & & \mathbb{Z}^*_{p_r^{e_r}} & & \underset{\text{CRT}}{\cong} & \mathbb{Z}_n^* \end{array}$$

„$(1) \leq_{pol.} (2)$": Für die Zerlegung von n kann folgender Algorithmus durchgeführt werden:

(a) Wähle zufällig ein $a \in_R \mathbb{Z}_n^*$ und setze $b := sqrt(a^2) \mod n$.
(b) Teste, ob $\text{ggT}(a \pm b, n) \notin \{1, n\}$, denn dann hat man damit einen nicht trivialen Faktor $a \pm b$ von n gefunden.
(c) Wenn die Zerlegung von n nunmehr noch nicht nur aus Primfaktoren besteht, so fahre mit den nicht-Primpotenzen bei (a) fort.

Für die Laufzeitabschätzung gilt für $r \geq 2$:

$$Ws_a[\text{ggT}(a \pm b, n) \notin \{1, n\}] = \frac{\#\text{guenstige Faelle}}{\#\text{moegliche Faelle}} = \frac{2^r - 2}{2^r} = 1 - 2^{-r+1} \geq \frac{1}{2},$$

also wird in Schritt (b) mit Wahrscheinlichkeit $\geq 1/2$ ein nichttrivialer Faktor von n gefunden.

(*Bew.:* (siehe auch Lemma 3.2.3) Nach Konstruktion gilt $a^2 = b^2 \mod n$. a/b ist eine zufällige Quadratwurzel von $1 \mod n$ genau dann, wenn a eine zufällige Quadratwurzel von $b^2 \mod n$ ist.
Wir können zu $n = \prod_{i=1}^{r} p_i^{e_i}$ die Wurzeln in $\mathbb{Z}^*_{p_i^{e_i}}$ berechnen. In jeder Untergruppe erhalten wir zwei, nämlich die positive und die negative, Quadratwurzeln. Mittels des CRT können wir diese auf 2^r Zahlen in $\mathbb{Z}_n^*$ abbilden. Dies sind die 2^r möglichen Fälle. a und $-a$ sind genau die zwei Fälle, die wir zur -Bildung nicht gebrauchen können, weshalb die Anzahl der günstigen Fälle $2^r - 2$ ist.)

Da wir in polynomieller Zeit die Wurzel berechnen können und in prob. pol. Zeit ein geeignetes a finden können, läuft dieser Algorithmus in prob. pol. Zeit. □

Wie wir in Algorithmus 12 gesehen haben, können wir in $\mathbb{Z}^*_{p^e}$ die Quadratwurzel effizient berechnen. Wir können eine Zahl, bei der ein Primzahltest ein negatives Ergebnis geliefert hat, also sogar zerlegen:

PROPOSITION 3.2.11. *Sei $n = \prod_{\nu=1}^{r} p_\nu^{e_\nu}$, $r \geq 2$, n ungerade, $n-1 = q2^k$, q ungerade. Mit dem Fermat-Test kann ein zufälliger Nicht-Zeuge $a \in_R \mathbb{Z}^*_n \setminus F(n)$ gefunden werden, mit dem der Miller-Rabin-Test gemacht werden kann. Dann gilt mit $Ws_a \geq 1 - 2^{-r+1}$, dass a kein Zeuge im Fermat-Test aber ein Zeuge im MR-Test ist.*

Für dieses a gilt:

$$\exists i, 0 \leq i < k : \underbrace{ggT\left(a^{q2^i} - 1 \mod n, n\right)}_{\text{Zerlegung von } n} \notin \{1, n\}.$$

BEWEIS. Wir verwenden einen Nicht-Zeugen a aus dem Fermat-Test. Damit $a \in MR(n)$ ist, muss gelten:

$$a^q \not\equiv 1 \mod n \text{ AND } \forall i, 0 \leq i < k : a^{q2^i} \not\equiv -1 \mod n.$$

Sei $i_\nu(a) = \min\left\{k \mid a^{q2^k} = 1 \mod p_\nu^{e_\nu}\right\}$. Da a ein Zeuge im MR-Test ist, gilt $a^{q2^i} \not\equiv -1 \mod n$ für alle $1 \leq i < k$. Betrachten wir nun a in den Moduln $p_\nu^{e_\nu}$. Dann gilt

$$a^{q2^{i_\nu(a)}} \equiv 1 \mod p_\nu^{e_\nu} \quad \Rightarrow \quad a^{q2^{i_\nu(a)-1}} \equiv -1 \mod p_\nu^{e_\nu} \quad \forall 1 \leq \nu \leq r$$

Falls nun $i_1(a) = \ldots = i_r(a)$ ist, dann gilt

$$a^{q2^{i_1(a)}} \equiv 1 \mod p_l^{e_l} \quad \forall 1 \leq l \leq r \quad \Rightarrow \quad a^{q2^{i_1(a)}} \equiv 1 \mod n,$$

was aber im Widerspruch zu der Annahme stünde, dass a ein Zeuge im MR-Test ist. Um genau diesen Fall zu vermeiden, dürfen die $i_\nu(a)$ nicht gleich sein.
Die Wahrscheinlichkeit, dass alle $i_\nu(a)$ gleich sind, ist nach Fakt 2 in Satz 2.5 auf Seite 27

$$\begin{aligned} & Ws\left(i_1(a) = i_2(a) = i_3(a) = \ldots = i_r(a)\right) \\ = \; & Ws\left(a_\nu^{q2^i} = i_1(a) \mod p_\nu^{e_\nu} \text{ fuer } \nu = 2, \ldots, r\right) \\ \leq \; & \left(\frac{1}{2}\right)^{r-1} = 2^{-r+1} \end{aligned}$$

Mit $Ws_a \geq 1 - 2^{-r+1}$ sind die $i_\nu(a)$ nicht alle gleich. Sei nun

$$\min_\nu := \min\{i_\nu(a) \mid \nu = 1, \ldots, r\} \text{ und } \max_\nu := \max\{i_\nu(a) \mid \nu = 1, \ldots, r\}$$

und damit

$$\text{ggT}\left(a^{q2^i} - 1, n\right) \notin \{1, n\} \quad \forall i, \min_\nu \leq i < \max_\nu,$$

da es im Bereich $\min_\nu \leq i < \max_\nu$ mindestens zwei ν_1, ν_2 gibt mit $\nu_1 \neq \nu_2$, $1 \leq \nu_1, \nu_2 \leq r$. Denn dann gilt:

$$\left.\begin{aligned} a^{q2^{i_{\nu_2}(a)}} &\not\equiv 1 \mod p_{\nu_1}^{e_{\nu_1}} \\ a^{q2^{i_{\nu_2}(a)}} &\equiv 1 \mod p_{\nu_2}^{e_{\nu_2}} \end{aligned}\right\} \rightarrow a^{q2^{i_{\nu_2}(a)}} \not\equiv 1 \text{ oder } n \mod n$$

In diesem Fall weiß man also, dass $a^{q2^{i_{\nu_2}(a)}}$ bezüglich dem Modul $p_{\nu_2}^{e_{\nu_2}}$ einen gemeinsamen ggT mit n besitzt, jedoch nicht bezüglich dem Modul $p_{\nu_1}^{e_{\nu_1}}$. Daraus folgt, dass $\mathrm{ggT}\left(a^{q2^{i_{\nu_2}(a)}}, n\right) \notin \{1, n\}$ ist.

Zusammenfassend: Mit Wahrscheinlichkeit $\geq 1 - 2^{-r+1}$ sind die $i_\nu(a)$ *nicht* alle gleich. In diesem Fall findet sich mindestens ein echter Teiler von n (für die Konstruktion werden dann zwei unterschiedliche $i_\nu(a)$ benötigt). □

CONCLUSION. Zufällige Nicht-Zeugen im Fermat-Test liefern also eine Zerlegung von n. Diese kann in Zeit $O\left(k \log_2 n\right)$ berechnet werden.

3.3. Die $p-1$-Methode

Es gibt Faktorisierungsmethoden, die vor allem Zahlen mit bestimmten Eigenschaften gut zerlegen können. Solche Zahlen müssen als RSA-Modul vermieden werden, da diese die Sicherheit des Verfahrens mindern. Das $p-1$-Verfahren von John Pollard ist zur Faktorisierung von zusammengesetzten Zahlen n geeignet, die einen Primfaktor p haben, für den $p-1$ nur kleine Primfaktoren hat.

γ-glatt

DEFINITION. **(γ-glatt, Glattheitszahl)** Sei $\gamma \in \mathbb{N}$ positiv. Eine Zahl $n \in \mathbb{N}$ heisst **γ-glatt**, wenn alle Primfaktoren von n kleiner oder gleich γ sind:

$$n = \prod_{i=1}^{k} p_i^{e_i} \quad \gamma - \text{glatt} \quad \Longleftrightarrow \quad \forall i : p_i \leq \gamma.$$

Glattheitszahl

Man nennt das kleinste gemeinsame Vielfache aller Primzahlpotenzen $\leq \gamma$ „die **Glattheitszahl** von γ ":

$$gl(\gamma) := \mathrm{kgV}(2, 3, \ldots, \gamma) = \prod_{p_i^{e_i} \leq \gamma < p_i^{e_i+1}} p_i^{e_i}.$$

So lässt sich die Definition auch formulieren als

$$n \text{ ist } \gamma - \text{glatt} \quad \Leftrightarrow \quad n \mid gl(\gamma).$$

Nach dem Primzahlsatz gilt

$$\begin{aligned} \pi(\gamma) &= \#\{p \leq \gamma \mid p \text{ prim}\} \\ &= \gamma / \ln \gamma + O\left(\gamma / (\ln \gamma)^2\right). \end{aligned}$$

Also gilt

$$gl(\gamma) \leq \gamma^{\pi(\gamma)} \leq e^{\gamma}.$$

3.3.1. Pollard's $p-1$- Algorithmus nach [Stinson, S. 151f]. Der ursprüngliche $p-1$-Algorithmus von Pollard wird in Algorithmus 13 wie in [**Stinson**, 4.8 S. 151] beschrieben.

BEWEIS. (Korrektheit von Algorithmus 13 auf der nächsten Seite) Angenommen, p ist ein Primteiler von n und $p-1$ ist γ-glatt. Für jede Primzahlpotenz s, die $(p-1)$ teilt, ist dann $s \leq \gamma$ und es gilt $(p-1) \mid \gamma!$.
Am Ende der **for**-Schleife in Schritt 2 gilt

$$a \equiv 2^{\gamma!} \mod n,$$

Algorithm 13 Pollard's $p-1$-Algorithmus nach [**Stinson**]

EINGABE: $n \in \mathbb{N}$ ungerade, obere Grenze $\gamma \in \mathbb{N}$.
AUSGABE: $d \in \mathbb{N}$ mit $d \neq 1, n$ und $d \mid n$ oder -1, falls kein solches d gefunden wurde.

(1) $a = 2$.
(2) *** *Berechne* $a^{\gamma!}$ ***
for $j = 2$ to γ **do**
(a) $a = a^j \mod n$.
(3) *** *Berechne* $ggT\left(a^{\gamma!}-1, n\right)$ ***
$d := \mathrm{ggT}(a-1, n)$.
(4) **if** $1 < d < n$ **then**
(a) **return**(d).
(5) **else**
(a) **return**(-1).

also wegen $p \mid n$ auch

$$a \equiv 2^{\gamma!} \mod p.$$

Weil p prim ist, gilt wegen Fermat

$$2^{p-1} \equiv 1 \mod p$$

und mit $(p-1) \mid \gamma!$ (wegen der Annahme, dass $p-1$ γ-glatt ist) auch

$$a \equiv 2^{\gamma!} \equiv 2^{k\cdot(p-1)} \equiv \left(2^{p-1}\right)^k \equiv 1^k \equiv 1 \mod p.$$

Damit wissen wir, dass wegen

$$p \mid (a-1)$$

und der Voraussetzung

$$p \mid n$$

gilt, dass

$$p \mid \mathrm{ggT}(a-1, n),$$

vorausgesetzt natürlich, dass dieser ggT existiert.

Damit ist eine Faktorisierung von n gelungen. Um eine komplette Primfaktorzerlegung von n zu erhalten, muss man nur die Faktoren auf Primheit überprüfen und falls diese nicht prim sind, den Algorithmus erneut anwenden. □

Laufzeit: Zur Laufzeit lässt sich sagen, dass in Schritt 2 für die Berechnung von $a^{\gamma!}$ insgesamt $O(\gamma \log \gamma)$ viele Bitoperationen benötigt. Die Berechnung des ggT in Schritt 3 benötigt weiterhin $O\left((\log n)^2\right)$ viele Bitoperationen. Bleibt also eineLaufzeit von $O(\gamma \log \gamma)$.

3.3.2. Pollard's $p-1$- Algorithmus nach [Blömer, Abschnitt 10.1]**.** Eine Version von Blömer, in der mehr Parameter als bei der Version von Stinson „eingestellt" werden können ist Algorithmus 14.

Die Korrektheit kann man ähnlich wie die von Algorithmus 13 zeigen:

Sei $a \in_R \mathbb{Z}_n$, p prim, $p < B_2$ mit $p \mid n$ und $p-1$ B_1-glatt (alle Primfaktoren von $p-1$ sind $\leq B_1$). Mit $p_i^{e_i} \leq B_2 < p_i^{e_i+1}$ gilt dann

$$a^{\Pi_{i=1}^{l} p_i^{e_i}} \equiv \left(a^{p-1}\right)^{\frac{\Pi_{i=1}^{l} p_i^{e_i}}{p-1}} \equiv 1 \mod p$$

Algorithm 14 Pollard's $p-1$-Algorithmus nach [**Blömer**]

EINGABE: $n \in \mathbb{N}$ nicht prim, Schranken $B_1, B_2 \in \mathbb{N}$, $B_2 < B_1$.
AUSGABE: $d \in \mathbb{N}$ $d \neq 1, n$ und $d \mid n$.

(1) **do**
  (a) Berechne ($\rightarrow$Sieb des Erathostenes) alle Primzahlen $p_1, p_2, \ldots, p_l$ kleiner als B_1.
  (b) Setze $k := 1$.
  (c) *** *Berechne* $gl_{B_1}(B_2)$ ***
  for $1 \leq i \leq l$ **do**
    (i) Setze $k = k \cdot p_i^{e_i}$ mit $p_i^{e_i} \leq B_2 < p_i^{e_i+1}$.
  (d) Wähle $a \in_R \mathbb{Z}_n$.
  (e) Berechne $d := \mathrm{ggT}(a^k - 1, n)$.
(2) **while**$(d \in \{1, n\})$
(3) **return**(d).

und damit

$$p \mid a^{\Pi_{i=1}^{l} p_i^{e_i}} - 1.$$

Wegen $p \mid n$ folgt dann

$$p \mid \mathrm{ggT}\left(a^{\Pi_{i=1}^{l} p_i^{e_i}} - 1, n\right).$$

Achtung: Angenommen, n ist von der Form $n = p \cdot q$ mit p und q prim und beide B_1-glatt, dann würde zusätzlich zu dem oben gesagten gelten

$$a^{\Pi_{i=1}^{l} p_i^{e_i}} \equiv \left(a^{p-1}\right)^{\frac{\Pi_{i=1}^{l} p_i^{e_i}}{p-1}} \equiv 1 \mod q$$

und damit

$$a^{\Pi_{i=1}^{l} p_i^{e_i}} \equiv 1 \mod n.$$

Der ggT von $a^{\Pi_{i=1}^{l} p_i^{e_i}}$ und n wäre dann also entweder 1 oder n!

Zur Bedeutung von B_2 läßt sich sagen, dass p unbedingt $\leq B_2$ sein muss, weil es sein kann, dass wenn dies nicht der Fall ist, p einen Primteiler $> p_i^{e_i}$ besitzt und somit $p \mid \prod_{i=1}^{l} p_i^{e_i}$ nicht gilt.

Im Gegensatz zu der Version von Stinson in Algorithmus 13 können in der Version von Blömer zwei Schranken „eingestellt" werden:

(1) B_1 ist eine obere Schranke für die Primfaktoren von $p-1$, wobei $p \mid n$ gilt und
(2) B_2 ist eine obere Schranke für die Primzahlpotenzen, deren Vielfaches die Glattheitszahl $gl(p-1)$ beschreiben.

In einer Erweiterung nach [**Montgomery, Silverman**] ist der Algorithmus aber auch dann erfolgreich, wenn $\left|\mathbb{Z}_p^*\right|$ B_1-glatt ist und einen einfachen Primfaktor zwischen B_1 und B_2 besitzt mit $B_2 \gg B_1$. In diesem Fall berechnen wir im Voraus

$$T = \left(a^{2m}, a^{4m}, \ldots, a^{rm}\right)$$

wobei $r \geq$ dem grössten Abstand zweier Primzahlen in $(B_1, B_2]$ sei. Wir definieren

$$Q_s = a^{ks} \mod n, \quad B_1 < s \leq B_2.$$

Wenn p_j die j-te Primzahl im Intervall $(B_1, B_2]$ ist, dann ist

$$(1) \quad Q_{p_1} \equiv a^{kp_1} \mod n \quad \text{und} \quad Q_{p_{j+1}} \equiv a^{k\left(p_{j+1}-p_j\right)} Q_{p_j} \mod n,$$

wobei die $a^{k\left(p_{j+1}-p_j\right)}$ in unserer vorausberechneten Tabelle T nachgesehen werden können. Zuletzt akkumulieren wir das Produkt

$$(2) \quad P = \prod_{j=1}^{\pi\left(B_2\right)-\pi\left(B_1\right)} \left(Q_{p_j} - 1\right) \mod n$$

und berechnen in regelmässigen Abständen $\mathrm{ggT}(P,n)$. Dieser Schritt ist erfolgreich, wenn $p-1 \mid ks$ für eine Primzahl s mit $B_1 < s \leq B_2$. Der auf diese Art und Weise implementierte Algorithmus benötigt für jede Primzahl im Intervall $(B_1, B_2]$ je eine Multiplikation modulo n von (1) und (2). Zusammen mit der Berechnung der Tabelle T ergibt sich damit eine Laufzeit von

$$2\left(\pi\left(B_2\right) - \pi\left(B_1\right)\right) + O\left(\left(\ln B_2\right)^2\right)$$

modularen Multiplikationen.

Zurück zu Algorithmus 14: Wie man sieht kann es sein, dass der Algorithmus nie hält (die *while*-Schleife in Schritt 1 also nie abbricht) – z.B. in dem oben skizzierten Fall. Dafür gibt es aber Situationen, in denen der Algorithmus auf jeden Fall mit einem echten Teiler von n terminiert:

PROPOSITION. *Sei $n \in \mathbb{N}$ mit zwei Primfaktoren $p,q \in \mathbb{P}$ mit $p \neq q$. Dann wird mit dem Algorithmus ein Teiler $d \notin \{1,n\}$ von n gefunden, wenn mit $p \leq B_2$ gilt*

(1) $p-1$ ist B_1-glatt
$\Leftrightarrow$ $p-1$ wird nur von Primteilern $\leq B_1$ geteilt.
(2) $q-1$ ist nicht B_1-glatt
$\Leftrightarrow$ $q-1$ besitzt einen Primteiler $r > B_1$.

BEWEIS. Sei $z \in \mathbb{Z}_q^*$ mit $\mathrm{ord}_q(z) = r > B_1$. So ein z exisitiert, da $\mathbb{Z}_q^*$ zyklisch ist mit Ordnung $q-1$ und $r \mid q-1$ nach 2 (nämlich $z = g^{(q-1)/r}$).

Sei nun $a \in \mathbb{Z}_n$ mit $a \equiv z \mod q$. Da $r > B_1$ prim, kann r nicht $k = gl\left(B_2\right)$ teilen. Also ist $a^k \not\equiv 1 \mod q$. Nach Voraussetzung für p gilt $p-1 \mid k$. Daher ist $a^k \equiv 1 \mod p$. Es gilt also, dass zwar p, nicht jedoch q $a^k - 1$ teilt. Daraus folgt aber $\mathrm{ggT}\left(a^k - 1, n\right) \notin \{1,n\}$. □

Probleme des Algorithmus, die bei EK nicht auftreten

Der Algorithmus hat also Probleme, wenn für alle Primteiler p von n die Zahlen $p-1$ etwa die gleichen Primteiler haben (in diesem Fall findet Algorithmus keine Lösung, da dann die exklusive Glattheit eines Primteilers nicht gewährt ist), oder sich nur in sehr großen Primteilern unterscheiden (B_2 muss sehr groß gewählt werden, was dann die Schleife in Schritt 1c) sehr aufwendig werden lässt). Die Laufzeit hängt sehr stark von der Form von n und der Gruppenstruktur von $\mathbb{Z}_n$ ab. An dieser Stelle greifen die Elliptischen Kurven, da man sich mit ihrer Hilfe viele Gruppen modulo n konstruieren kann, also nicht mehr nur noch $\mathbb{Z}_n$ zur Verfügung hat (siehe dazu Kapitel 4.6.5 ab Seite 78). Bei der Verwendung Elliptischer Kurven wird das $p-1$-Verfahren anstelle auf der Gruppe $\mathbb{Z}_n^*$ auf der Gruppe der Punkte einer Elliptischen Kurve angwendet.

3.3.3. Pollard's $p-1$- Algorithmus nach [Menezes et al., Kap. 3.2.3]. Diese Variante wurde nicht in der Vorlesung besprochen, stellt jedoch die Basis für Algorithmus 16 dar.

Die Idee hinter dem $p-1$-Algorithmus beschreiben Menezes et al. wie folgt:

Sei γ die Glattheitsschranke und Q das kleinste gemeinsame Vielfache aller Primzahlpotenzen $\leq n$ von Primzahlen $\leq \gamma$. Es gilt

$$q^l \leq n \quad \Rightarrow \quad l \ln q \leq \ln n \quad \Rightarrow \quad l \leq \left\lceil \frac{\ln n}{\ln q} \right\rceil$$

und damit

$$Q = gl(\gamma) \prod_{q \leq \gamma} q^{\lfloor \ln n / \ln q \rfloor} \quad \text{mit} q \in \mathbb{P}, q \leq \gamma.$$

Wenn p ein Primfaktor von n ist mit $p-1$ ist γ-glatt, dann $p-1 \mid Q$ und für alle a mit $\mathrm{ggT}(a,p) \equiv 1$ gilt nach Fermat $a^Q \equiv 1 \mod p$ und damit ist $a^Q - 1 \equiv 0 \mod p$, also ein Vielfaches von p. Diese Tatsachen werden nun in dem folgenden Algorithmus verwendet:

Algorithm 15 Pollard's $p-1$-Algorithmus nach [**Menezes et al.**]

EINGABE: $n \in \mathbb{N}$ zusammengesetzt und n keine Primzahlpotenz, Glattheitsschranke γ.
AUSGABE: $d \notin \{1,n\}$ mit $d \mid n$ oder -1.

(1) Wähle $a \in_R [2, n-1]$.
(2) Berechne $d = \mathrm{ggT}(a,n)$.
(3) **if** $d \geq 2$ **then**
  (a) **return**(d)**.**
(4) **for** jede Primzahl $q \leq \gamma$ **do**
  (a) $l := \left\lfloor \frac{\ln n}{\ln q} \right\rfloor$.
  (b) $a := q^{q^l} \mod n$.
(5) **** a entspricht jetzt a^Q ****
  $d := \mathrm{ggT}(a-1, n)$.
(6) **if** $d \in \{1,n\}$ **then**
  (a) **return(**-1**).**
(7) **else**
  (a) **return**(d)**.**

3.3.4. Verbesserter $p-1$-Algorithmus mit Miller-Rabin. In der Vorlesung wurde eine Kombination aus Pollard und Miller-Rabin besprochen, die Verbesserungen gegenüber der „herkömmlichen" Variante bringen soll (Prof. Schnorr: "wir quadrieren solange, bis es nichts mehr bringt").

BEWEIS. (Korrektheit von Algorithmus 16 auf der nächsten Seite) Der Algorithmus bekommt wie im Algorithmus 13 auf Seite 41 zwei Eingabewerte n und γ.

Im Beweis zu Algorithmus 13 haben wir gesehen, dass gilt

$$a^{gl(\gamma)} \equiv 1 \mod p.$$

Wir können $gl(\gamma) = q2^k$ schreiben, wobei aus $2^k \leq \gamma$ wegen der Glattheit sind alle Primfaktorpotenzen $\leq \gamma$ folgt, dass $k = \lfloor log_2 \gamma \rfloor$ ist. Sei nun

Algorithm 16 Kombination von Pollard's $p-1$-Algorithmus und Miller-Rabin

EINGABE: $n = \prod_{v=1}^{r} p_v^{e_v}$ ungerade mit einfachem Primteiler $p = p^e$ und γ, wobei gilt, dass $p-1$ γ-glatt ist.

AUSGABE: $d \notin \{1,n\} : d \mid n$ oder -1, falls ein solches d nicht gefunden wurde.

(1) Wähle $a \in_R \mathbb{Z}_n^*$.
(2) $k := \lfloor log_2 \gamma \rfloor$.
(3) **** Berechnung von q mit* $gl(\gamma) = q2^k = kgV(2,3,\ldots,\gamma)$ *****
$q := \mathrm{kgV}(3,5,7,\ldots,\gamma)$ (falls γ ungerade ist, ansonsten bis $\gamma - 1$).
(4) **for** $i = 0,\ldots,k-1$ **do**
  (a) **** Miller-Rabin ****
  if $\mathrm{d} := \mathrm{ggT}\left(a^{q2^i} - 1, n\right) \notin \{1,n\}$ **then**
    (i) Beende Schleife ($i := k$).
(5) **if** $d \notin \{1,n\}$ **then**
  (a) **return(**d**).**
(6) **else**
  (a) **return(**-1**).**

$$q := \mathrm{kgV}(3,5,7,9,\ldots,\gamma).$$

Achtung: diese „unauffällige" Zuweisung erfordert die Berechnung des kgV mit einem großen Rechenaufwand; vergleiche hierzu [**Menezes et al.**, Algorithmus 3.14 Schritt 3].

Da $p-1$ γ-glatt ist, gilt:

$$\underbrace{(p-1)}_{=\left|\mathbb{Z}_p^*\right|} \mid gl(\gamma) = q \cdot 2^k \quad \Rightarrow \quad a^{q2^k} \equiv 1 \mod p.$$

Im ursprünglichen $p-1$-Algorithmus wird zur Ermittlung des Faktors p der ggT $\left(a^{q2^k} - 1, n\right)$ berechnet. Diese Variante durchläuft in Schritt 4 eine **for**-Schleife, in der für $i = 0,\ldots,k-1$ ein ggT $\left(a^{q2^i} - 1, n\right) \notin \{1,n\}$ gesucht wird (während im ursprünglichen Pollard also nur ein ggT berechnet wurde, verlangt diese Algorithmus bis zu $log_2 \gamma - 1$ ggT-Berechnungen). Wenn ein solcher ggT gefunden wird, dann liefert er auch einen Faktor von n.

Nach Satz 3.2.11 ist die Wahrscheinlichkeit, dass man einen solchen Teiler findet

$$Ws \geq 1 - \frac{1}{2^{r-1}}.$$

In der **for**-Schleife in Schritt 4 wird nun der ggT von $a^{q2^i} - 1$ und n berechnet. Da gilt

$$a^{q2^k} \equiv 1 \mod p \quad \Rightarrow \quad p \mid \left(a^{q2^k} - 1\right)$$

und p nach Voraussetzung ein Teiler von n ist, teilt p auch ggT $\left(a^{q2^i} - 1, n\right) \notin \{1,n\}$ (natürlich vorausgesetzt, dass dieser nicht $= 1$ ist). Damit wurde mit dem ggT wie im Algorithmus 13 auf Seite 41 ein Faktor von n gefunden, der mit dem Algorithmus weiter zerlegt werden kann, sofern er nicht schon selbst prim ist. Im Gegensatz zum ursprünglich von Pollard veröffentlichten Algorithmus ist hier die Laufzeit kürzer, da die Schleife nicht maximal γ-mal sondern nur höchstens $k - 1 = \lfloor log_2 \gamma \rfloor - 1$-mal durchlaufen werden muss. □

Nach Satz 3.2.11 ist die Wahrscheinlichkeit, dass man mit dem Algorithmus einen Teiler findet

$$Ws\left[\text{der Algorithmus findet einen Teiler von } n\right] \geq 1 - \frac{1}{2^{r-1}}.$$

PROPOSITION 3.3.1. ***(Primzahlsatz)*** *Für die Anzahl der Primzahlen kleiner oder gleich einer beliebigen Zahl gilt*

$$\Pi(\gamma) := \#\{p \leq \gamma \mid p \text{ prim}\} = \gamma/\ln\gamma + o\left(\gamma/(\ln\gamma)^2\right)$$

Für die Glattheitszahl $gl(\gamma)$ gilt

$$gl(\gamma) = \prod_{p_i^{e_i} \leq \gamma} p_i^{e_i} \leq \gamma^{\Pi\gamma} \leq \gamma^{\frac{\gamma}{\ln\gamma}} = e^{\gamma}.$$

Damit ist die Abbildung $a \mapsto a^q$ in Zeit $\log_2(e^{\gamma}) \leq \left(\log_2 e\right) - \gamma$ nicht durchführbar für $\gamma \geq 2^{100}$.

Implikationen auf RSA

Doch nun zurück zum Ausgangspunkt, nämlich dem RSA-Algorithmus. Als Modul wird dort eine Zahl n als Produkt zweier Primzahlen p und q verwendet. Die Sicherheit des RSA-Verfahrens beruht darauf, dass es sehr schwer ist, n in seine Primfaktoren (hier p und q) zu zerlegen. Da es mit Hilfe des $p-1$-Algorithmus recht einfach ist, n zu zerlegen, falls $p-1$ oder $q-1$ kleine Primfaktoren besitzen, sollten p und q entsprechend gewählt werden, beispielsweise $p = 2p_1 + 1$ mit $p_1$und p prim (und q genauso). In diesem Fall ist n gegenüber dem $p-1$-Algorithmus von Pollard resistent, da dann $\varphi(p) = p - 1 = 2p_1$ sehr groß ist und somit im Algorithmus 16 auf der vorherigen Seite ein entsprechend großes $\gamma \geq p_1$ verwendet werden muss, was zu einer sehr großen Laufzeit des Algorithmus führt.

Um den RSA-Algorithmus also effektiv einsetzen zu können, muss man in der Lage sein, sehr große Primzahlen der Form $p = 2p' + 1$ zu erzeugen, für die auch p' prim ist.

3.3.5. Pollard's $p-1$-Algorithmus für G_n. Bei den bisherigen Betrachtungen zum $p-1$-Algorithmus von Pollard wurde immer vorausgesetzt, dass alle Berechnungen in $\mathbb{Z}_n^*$ durchgeführt werden. Bei genauerer Betrachtung jedoch erkennt man, dass der Algorithmus auch dann funktioniert, wenn man an Stelle von $\mathbb{Z}_n^*$ eine beliebige abelsche Gruppe G_n verwendet, so dass

- $G_n \underset{\text{Teilmenge}}{\subseteq} \mathbb{Z}_n^t$.
- die Funktionen $(a,b) \mapsto a \cdot b$ und $(a,b) \mapsto a/b$ rationale Funktionen $\mathbb{Z}_n^t \times \mathbb{Z}_n^t \to \mathbb{Z}_n^t$ sind.
- n einen einfachen Primteiler $p = p^e$ besitzt mit
  - $\left|G_p\right|$ γ-glatt
  - $\left|G_p\right|$ gerade (schließt triviale Untergruppen aus)
  - G_p zyklisch.

Das heisst also, dass die Arithmetik von G_n über $\mathbb{Z}_n$ ist. Nach dem CRT gilt für $n = \prod_{i=1}^{r} p_i^{e_i}$

$$G_n \cong G_{p_1^{e_1}} \times \ldots \times G_{p_r^{e_r}}.$$

Da es sich hier um einen Isomorphismus handelt, der ja alle Strukturen erhält, sind auch die $G_{p_i^{e_i}}$ Gruppen, da G_n nach Voraussetzung eine Gruppe ist.

Algorithm 17 Verallgemeinerung von Pollard's $p-1$-Algorithmus

EINGABE: n habe einfachen Primteiler $p=p^e$, $\left|G_p\right|$ ist γ-glatt ($\mathbb{Z}_n^* \rightsquigarrow G_n$, $\mathbb{Z}_p^* \rightsquigarrow G_p$)

(1) Wähle $a \in_R G_n$.
(2) Sei $gl(\gamma) = q2^k = kgV(2,3,\ldots,\gamma)$ mit $k := \lfloor log_2 \gamma \rfloor$ und $q = \text{kgV}(3,5,7,\ldots,\gamma)$ (falls γ ungerade ist, ansonsten bis $\gamma-1$).
(3) **for** $i=0,\ldots,k-1$ **do**
   (a) Teste, ob für $a^{q2^i} =: \left(a_{i,1},\ldots,a_{i,t}\right) \in \mathbb{Z}_n^t$ gilt: ggT $\left(a_{i,\nu}-\varepsilon_\nu, n\right) \notin \{1,n\}$ mit $a_{i,\nu}, \varepsilon_\nu \in \mathbb{Z}_n$
(4) Falls ein solcher ggT gefunden wurde, dann ist dieser ein Teiler von n. Ansonsten war der Algorithmus nicht erfolgreich.

Die 1 in G_n wird in dem Algorithmus ersetzt durch

$$1_{G_n} = \left(\varepsilon_1,\ldots,\varepsilon_t\right) \in \mathbb{Z}_n^t \quad \text{mit} \quad \varepsilon_{i,\nu} := \varepsilon_i \mod p_\nu^{e_\nu} \, \forall i = 1,\ldots,t,\ \nu = 1,\ldots,r.$$

PROPOSITION 3.3.2. *Sei p ein einfacher Primteiler von $n = \prod_{\nu=1}^r p_\nu^{e_\nu}$, $\left|G_{p_1^{e_1}}\right|$ γ-glatt, $\left|G_{p_\nu^{e_\nu}}\right|$ gerade für $\nu = 1,\ldots,r$ und G_p zyklisch. Dann geht die Abbildung*

$$n \quad \mapsto \quad \textit{Zerlegung von } n$$

mit $Ws_a \geq 1-2^{-r+1}$ in $O\left(\gamma + k\log_2 n\right)$ arithmetischen Schritten modulo n. Die O-Konstante enthält die Anzahl der arithmetischen Schritte zur Berechnung von $(a,b) \mapsto a \cdot b$.

BEWEIS. In diesem Fall kann der $p-1$-Algorithmus von Pollard angewendet werden, von dem bereits bekannt ist, dass er mit Wahrscheinlichkeit $\geq 1-2^{-r+1}$ eine Zerlegung von n findet. Im dem Algorithmus werden maximal $\gamma-1$ Exponentiationen durchgeführt, von denen jede Zeit $O\left(2\log_2 \gamma\right)$ benötigt (sofern man den Algorithmus „square-and-multiply"verwendet, s. [**Stinson**, 4.8.1 The p-1 Method]). Die Berechnung des ggT kann in Zeit $O\left(\left(\log_2 n\right)^2\right)$ erfolgen, wenn man den Euklidischen Algorithmus benutzt. Daraus ergibt sich eine Gesamtlaufzeit von $O\left(\gamma\log_2\gamma + k\left(\log_2 n\right)^3\right)$ Bitoperationen und entsprechend $O\left(\gamma + k\log_2 n\right)$ arithmetischen Schritten. □

3.4. Pollard's ρ-Algorithmus

Genau wie der $p-1$-Algorithmus von Pollard ist dieser Algorithmus ein „special-purpose"-Algorithmus für das Finden kleiner Faktoren einer zusammengesetzten Zahl n.

Sei

$$f: \quad [1,n] \quad \to \quad [1,n]$$

eine Zufallsfunktion und $x_0 \in_R [1,n]$ zufällig gewählt. Es sei nun

$$x_{i+1} = f(x_i) \quad \forall i \geq 0.$$

Da die x_i nur endlich viele Werte annehmen können (nämlich genau $(n+1)$-viele), kann man erwarten, dass die Folge $x_0, x_1, \ldots$ irgendwann in einen Zyklus kommt, nämlich genau dann, wenn ein $x_j = x_i$ ist mit $j > i$. Die Folge der Elemente bis zu diesem Zyklus nennen wir den **Vorlauf** (tail) mit der erwarteten Länge $\mu = \sqrt{\pi n/8}$. Dieser wird gefolgt von der sog. **Periode** (cycle) der erwarteten Länge $\lambda = \sqrt{\pi n/8}$. In der Praxis kann es nach

[**Menezes et al.**, Kapitel 3.2.2] bei der Faktorisierung und dem diskreten Logarithmus ein Problem darstellen, genau die i und j zu finden, für die gilt $x_i = x_j$. Wie man leicht einsieht wird die Anzahl der Vergleiche sehr groß, wenn man x_i mit allen x_j für $j < i$ vergleichen möchte. Aus diesem Grund vergleichen wir nur x_{2i} und x_i miteinander (s.u.).

Was bringt es nun für Vorteile für die Faktorisierung von n, wenn man den Vorlauf μ und die Periode λ mit dem Algorithmus berechnet hat? Nun, hat man μ und λ, so weiss man, dass weitere Iterationen in der **for**-Schleife keine neuen Erkenntnisse bringen. In diesem Fall kann man also die ggT-Suche abbrechen.

Doch nun zuerst der Algorithmus selbst:

Algorithm 18 Pollard's ρ-Algorithmus

EINGABE: Seien $n = p \cdot q$, $\mathrm{ggT}(p,q) = 1$ und $F : [1,n] \to [1,n]$ eine rationale Zufallsfunktion mit Arithmetik modulo n, nämlich

- $F = f/g$ mit $f, g \in \mathbb{Z}_n[x]$, so dass der CRT anwendbar ist mit $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$
- $F \in_R [1,n]^{[1,n]}$, die Funktion wird also zufällig aus allen Funktionen $f : [1,n] \to [1,n]$ ausgewählt.

AUSGABE: $d \notin \{1,n\}$ mit $d \mid n$ oder -1.

(1) $x_0 := 1$.
(2) **** Nach im Mittel $1.3\sqrt{n}$ wird eine Zahl doppelt gewählt sein ****
for $i = 1, \ldots, \mu + \lambda$ **do**
(a) $x_{2i-1} := F\left(x_{2i-2}\right)$.
(b) $x_{2i} := F\left(x_{2i-1}\right)$.
(c) $d := \mathrm{ggT}\left(x_{2i} - x_{i,n}\right)$.
(d) **if** $d \neq 1$ **then**
(i) **return(**d**).**
(3) **return(**-1**).**

Korrektheit: Der Algorithmus ist probabilistisch Polynomialzeit, liefert also nicht in jedem Fall ein korrektes Ergebnis!

Laufzeit: $O\left(3(\mu+\lambda)(\log n)^2\right) = O\left(\sqrt{\pi p/8}(\log n)^2\right)$.

Wir verwenden also eine Zufallsfunktion

$$f: \quad [1,n] \quad \longrightarrow \quad [1,n]$$

und hoffen, dass wir im Laufe der Berechnung der x_i irgendwann auf ein i stossen, so dass $x_i \equiv x_{2i} \mod r$ mit $r \mid n$ (noch unbekannt, aber $x_i \not\equiv x_{2i} \mod n$, da dann $\mathrm{ggT}\left(x_{2i} - x_i, n\right) \notin \{1,n\}$ ist). Eigentlich genügt schon ein beliebiges j mit $x_i \equiv x_j \mod p$, aber alle möglichen i,j-Kombinationen auszuprobieren ist ineffizient – die Betrachtung von $2i$ und i ist effizienter, hat aber kürzere Laufzeit ($\rightarrow$Lemma 3.4.1). Dazu überlege man sich, dass wenn man einen n-seitigen Würfel wirft, eine Zahl nach im Mittel $\leq 1.3\sqrt{n}$ zum zweiten mal gewürfelt wird (für $n \geq 100$). Wenn wir also eine Zufallsfolge von Zahlen aus $\mathbb{Z}_n$ generieren, dann kommt im Mittel nach $1.3\sqrt{n}$ Würfen eine Zahl doppelt vor. Nehmen wir an, dass $p \mid n$ gilt, dann haben wir nach etwa $1.3\sqrt{p}$ Schritten zwei Zahlen generiert, für die gilt $x_i \equiv x_j \mod p$, aber $x_i \not\equiv x_j \mod n$, was ergibt, dass $\mathrm{ggT}\left(x_i - x_j, n\right) \notin \{1,n\}$ ein nichttrivialer Teiler von n ist. Vergleichen wir nun nach jedem Wurf die neu gewürfelte Zahl mit

allen bisher gewürfelten, so macht das etwa $2\sqrt{p}\left(2\sqrt{p}-1\right)=2p$ Vergleiche – also mehr, als die triviale Methode zum Finden eines Faktors benötigt. Aus diesem Grund betrachten wir nur den $\mathrm{ggT}\left(x_{2i}-x_i,n\right)$, was sich als genauso effektiv erweist, wie die langsame Methode oben, aber wesentlich weniger Verleiche benötigt:

Angenommen, es gibt i,j mit $x_i \equiv x_j \mod p$ mit $1 \leq i < j < 2\sqrt{p}$. Dann setzen wir $i+d=j$ mit $d \geq 1$ und der Zufallsfunktion $f(x)=x^2+1$:

$$x_{i+1} \equiv x_i^2+1 \equiv x_{i+d}^2+1 \equiv x_{i+d+1} \mod p,$$

und damit

$$x_{i+r} \equiv x_{i+d+r} \mod p$$

für alle $r>0$. Unter den Zahlen $i,i+1,\ldots,i+d-1$ gibt es nun genau ein Vielfaches von d, so dass $i+k=ed$ ist und damit

$$x_{ed} \equiv x_{i+k} \equiv x_{i+k+d} \equiv x_{ed+d} \mod p,$$

und ebenso

$$\begin{aligned} x_{ed+d} &\equiv x_{(ed+d)+d} \mod p \\ x_{ed+2d} &\equiv x_{(ed+2d)+d} \mod p \\ &\vdots \end{aligned}$$

und damit

$$x_{ed} \equiv e_{2ed} \mod p.$$

Wegen $ed=i+k<i+d\leq j<2\sqrt{p}$ schliessen wir, dass aus $x_i \equiv x_j \mod p$ für $1 \leq i < j < 2\sqrt{p}$ folgt, dass es ein $t<2\sqrt{p}$ gibt, nämlich $t=ed$, so dass $x_t \equiv x_{2t} \mod p$. Also ist die Berechnung von $\mathrm{ggT}\left(x_{2t}-x_t,n\right)$ für $1 \leq t < 2\sqrt{p}$ zu einem p führen, das ein Teiler von n ist (in der Hoffnung, dass nicht $x_{2t}-x_t$ von n geteilt wird).

LEMMA 3.4.1. *Für das kleinste i mit $x_{2i}=x_i$ gilt: $\mu \leq i \leq \mu+\lambda$.*

BEWEIS. Für $r>i$ gilt

$$\begin{aligned} x_r = x_i \quad &\Leftrightarrow \quad \lambda \mid r-i \quad \text{und} \quad i \geq \mu \\ \text{und somit} \quad x_{2i} = x_i \quad &\Leftrightarrow \quad \lambda \mid i \quad \text{und} \quad \mathrm{i} \geq \mu. \end{aligned}$$

□

Wir hoffen nun, dass $x_{2i} \equiv x_i \mod q$ und $x_{2i} \not\equiv x_i \mod n$, also $\mathrm{ggT}\left(x_{2i}-x_i,n\right) \notin \{1,n\}$.

LEMMA 3.4.2. *Die Wahrscheinlichkeit, dass die Summe aus Vorlauf und Periode kleiner als k ist, ist*

$$1-e^{-\binom{k}{2}/p} \overset{2)}{\leq} Ws[\mu+\lambda<k] \overset{1)}{\leq} \binom{k}{2}/p.$$

BEWEIS. 1) Es gilt

$$Ws[\mu+\lambda<k] = Ws\left[x_0,\ldots,x_{k-1} \text{ nicht alle paarweise verschieden}\right]$$
$$\leq \sum_{0\leq i<j\leq k-1} Ws\left[x_i = x_j\right] \leq \underbrace{\binom{k}{2}}_{\#\text{moeglicher Paare}} \quad /p.$$

2) Wieder gilt

$$Ws[\mu+\lambda\geq k] = Ws\left[x_0,\ldots,x_{k-1} \text{ alle paarweise verschieden}\right]$$
$$= \prod_{i=1}^{k-1} \underbrace{\left(1-\frac{i}{p}\right)}_{=(p-i)/p} \quad \text{(da nach jedem gezogenen Element eines weniger uebrig ist)}$$
$$\leq \prod_{i=1}^{k-1}\left(1-\frac{1}{p}\right)^{i-1} \leq \left(1-\frac{1}{p}\right)^{\binom{k}{2}} \underset{k\to\infty}{<} 1-e^{-\binom{k}{2}/p}.$$

□

Wie oben besprochen erwarten wir ein doppeltes Auftreten einer Zahl im Mittel nach $k \geq 2\sqrt{p}(1+\varepsilon)$ Schritten, womit sich ergibt $\binom{k}{2}/p \geq (1+\varepsilon)$ und somit

$$\begin{aligned} Ws[\mu+\lambda\geq k] &\leq e^{-(1+\varepsilon)} \\ \Rightarrow \quad E[\mu+\lambda] &\leq \sqrt{2p}(1+\varepsilon). \end{aligned}$$

PROPOSITION 3.4.3. *Für den Erwartungswert der Summe von Vorlauf und Periode im ρ-Algorithmus gilt*

$$E[\mu+\lambda] \underset{p\to\infty}{\longrightarrow} \sqrt{\pi p/2}.$$

Die Erwartungswerte von Vorlauf und Periode sind gleich:

$$E[\mu] = E[\lambda] \underset{p\to\infty}{\longrightarrow} \sqrt{\pi p/8}.$$

BEWEIS. Es gilt für die Verteilungsfunktion von $\mu+\lambda$

$$\varphi(x) = \frac{1}{\sqrt{2\pi p}} e^{-\frac{x^2}{2p}} \cdot \frac{x^2}{p} \quad \text{mit} x \in]-\infty,+\infty[$$

Für den Erwartungswert von $\varphi(x)$ gilt damit

$$\int_{-\infty}^{+\infty} \varphi(x)\, x dx = 1, \quad E[\mu+\lambda] \underset{p\to\infty}{\longrightarrow} \sqrt{\pi p/2}$$
$$\sum_{k=1}^{p} e^{-\frac{k^2}{2p}} \frac{k^2}{p} \underset{p\to\infty}{\longrightarrow} \sqrt{2\pi p} \underbrace{\int_0^{\infty} \varphi(x)\, dx}_{\frac{1}{2}} = \sqrt{\pi p/2}$$

und damit

$$E[\mu] \underset{p\to\infty}{\longrightarrow} \sqrt{\pi p/8}.$$

□

3.4.1. Einschub: „Kleine Nullstellen von Polynomen: Angriffe auf RSA". Hier kommt ein Einschub von Marc Fischlin, siehe auch [**M. Fischlin**, Webseite der MI].

Algorithm 19 Faktorisieren mit Pollard-ρ

EINGABE: $n = pq$, $\mathrm{ggT}(p,q) = 1$.
AUSGABE: $d \notin \{1,n\}$ mit $d \mid n$.

(1) $a_0 = 2$
(2) **do**
  (a) $a_i := a_{i-1}^2 + 1 \mod n$.
  (b) $a_{2i} := \left(a_{2i-2}^2 + 1\right)^2 + 1 \mod n$.
(3) **while** $\mathrm{ggT}\left(a_{2i} - a_i, n\right) \notin \{1,n\}$
(4) **return(d).**

3.4.2. Faktorisieren mit Pollard-ρ. Angenommen, die Funktion

$$a \quad \mapsto \quad a^2 + 1 \mod p$$

verhält sich wie eine Zufallsfunktion, dann $\exists i \leq \mu + \lambda \leq \sqrt{\pi p/2}$, so dass $x_{2i} \equiv x_i \mod p$ (s.o.).

3.5. Kombination von Pollard $p-1$ und Pollard ρ

Gibt es eine Kombination von Pollard $p-1$ und Pollard ρ ?

Stufe 1: $gl(\gamma) = q2^k$, $a \in_R \mathbb{Z}_n^*$
Teste $\mathrm{ggT}(a^{q2^i} - 1, n) \notin \{1,n\}$ für $i = 0,\ldots,k$
Wir sind wegen der Exponentiation $a^{q2^i} \in (\mathbb{Z}_n^*)^{q2^k}$ in einer kleinen Untergruppe, in der wir das Pollard-ρ-Verfahren anwenden wollen.

Stufe 2: Benötigt wird eine modulare (für CRT) Zufallsfunktion $F : (\mathbb{Z}_n^*)^{q2^k} \to (\mathbb{Z}_n^*)^{q2^k}$. Sie muss $\mathbb{Z}$-rational sein mit $F = f(x)/g(x)$ mit $f,g \in \mathbb{Z}[x]$ und **modular** in dem Sinne, dass gilt

$$x_i \equiv x_j \mod p \quad \Rightarrow \quad F(x_i) = F\left(x_j\right) \mod p.$$

Damit können wir Algorithmus 20 formulieren.

Die geforderte Modularität von F impliziert für die Folge $x_i \mod p$, dass gilt

$$E[\mu + \lambda] \approx \sqrt{\pi p/2}.$$

Leider sind keine $\mathbb{Z}$-rationalen Zufallsfunktionen bekannt und in $(\mathbb{Z}_n^*)^{q2^k}$ ist nur die Multiplikation, jedoch nicht die Addition verfügbar, da diese aus $(\mathbb{Z}_n^*)^{q2^k}$ herausführt.

Eine Zufallsfunktion

$$F: \quad (\mathbb{Z}_n^*)^{q2^k} \quad \to \quad (\mathbb{Z}_n^*)^{q2^k},$$

die nicht modular ist, kann jedoch einfach gebaut und deren Zufälligkeit sogar bewiesen werden ($\to$Algorithmus 21).

Die Rekursion der x_i beruht auf der Arbeit von [**Brent1986**, S. 149-163].

Algorithm 20 Kombination von Pollard ρ und $p-1$ mit modularer Zufallsfunktion

EINGABE: $n, \gamma \in \mathbb{N}$ mit $n = pq$ und $\mathrm{ggT}(p,q) = 1$, modulare Zufallsfunktion
$F : (\mathbb{Z}_n^*)^{q2^k} \to (\mathbb{Z}_n^*)^{q2^k}$ $\mathbb{Z}$-rational.
AUSGABE: $d \notin \{1,n\}$ mit $d \mid n$ oder -1.

(1) $d := -1$.
(2) $a \in_R \mathbb{Z}_n^*$.
(3) $x_0 := a^{q2^k}$.
(4) **for** $i = 1, \ldots, \gamma$ **do**
  (a) $x_{i+1} := F(x_i)$.
  (b) $x_{2i} := F(x_{2i-1})$.
  (c) **if** $\mathrm{ggT}(x_{2i} - x_i, n) \notin \{1,n\}$ **then**
    (i) $d := \mathrm{ggT}(x_{2i} - x_i, n)$.
    (ii) Breche Schleife ab ($i := \gamma + 1$).
(5) **return(d).**

Algorithm 21 Kombination von Pollard ρ und $p-1$ mit nicht-modularer Zufallsfunktion

EINGABE: $n, \gamma \in \mathbb{N}$ mit $n = pq$ und $\mathrm{ggT}(p,q) = 1$.
AUSGABE: $d \notin \{1,n\}$ mit $d \mid n$ oder -1.

(1) $a \in_R \mathbb{Z}_n^*$.
(2) $x_0 := a^{q2^k}$.
(3) **for** $i = 1, \ldots, \gamma$ **do**
  (a) $b_i \in_R \{0,1\}$.
  (b) **** Jeder Wert wird mit $Ws_{b_i} = 1/2$ zufällig zugewiesen ****

$$x_{i+1} := \begin{cases} x_i^2 & , \text{ falls } b_i = 0 \\ x_i^2 x_0 & , \text{ falls } b_i = 1 \end{cases}$$

(4) $d := -1$.
(5) **** Alle x_i miteinander vergleichen ****
  for $i = 1, \ldots, \gamma - 1$ **do**
  (a) **for** $j = i+1, \ldots, \gamma$ **do**
    (i) **if** $\mathrm{ggT}\left(x_i - x_j, n\right) \notin \{1,n\}$ **then**
      $d := \mathrm{ggT}(x_{2i} - x_i, n) \notin \{1,n\}$.
      Breche Schleife ab ($i := \gamma$, $j := y + 1$).
(6) **return(d).**

Es müssen also alle Indizes (γ^2 viele)berechnet werden, während bei Pollard-ρ nur γ-viele berechnet werden müssen.

Bei Pollard-ρ war das Hauptargument:

$$\left.\begin{array}{c} X_i = x_i \mod p \\ X_\mu = X_{\lambda+\mu} \mod p \end{array}\right\} \underset{\rho}{\Rightarrow} X_{\lambda+\mu} = X_{\lambda+\mu+1}$$

Wegen der zufällig gewählten Bits gibt es diese periodische Eigenschaft nicht mehr. Deshalb müssen wir über die vollen i, j iterieren und haben eine denkbar schlechte Laufzeit („dafür haben wir eine schöne Zufallsfunktion.").

LEMMA 3.5.1. *x_i ist zufällig in $x_o^I = \{x_o^i \mid i \in [0,2^i-1]\}$ also*

$$x_i \in_R x_0^{[0,2^i-1]} = \left\{x_0^0, x_0^1, x_0^2, \ldots, x_0^{2^i-1}\right\}$$

BEWEIS. (durch Induktion über i)
Verankerung: $x_o^0, x_o^1 \in_R x_0^{[0,1]}$ stimmt (entweder x_0^0 quadrieren oder zusätzlich mit x_0 multiplizieren).
Annahme: Sei $x_i = x_0^a$ mit $a \in_R [0,2^i-1]$.
Induktionsschritt: Dann gilt $x_{i+1} = x_0^b$ mit $b = \begin{cases} 2a \\ 2a+1 \end{cases}$ mit $Ws_b = 1/2$.
Diese Tatsache kann man sich in der Binärdarstellung verdeutlichen, wenn man bedenkt, dass x_{i+1} gleich der Zahl x_i ist, die um 1 nach links geshiftet wurde. Mit Wahrscheinlichkeit $1/2$ wird dann das Bit ganz rechts gesetzt:

$$\boxed{a_{i+1}} \quad \cong \quad \boxed{a_i \quad \mid b_{i+1}}$$

Somit folgt:

$$a_{i+1} \in_R [0,2^{i+1}-1] = 2[0,2^i-1]+1$$

□

Nach diesem Lemma ist die in Algorithmus 21 verwendete Zufallsfunktion tatsächlich zufällig – aber sie ist nicht modular, denn aus $x_i \equiv x_j \mod p$ folgt nicht, dass gilt $b_i = b_j$.

PROPOSITION 3.5.2. *Sei $n = \prod_{\nu=1}^r p_\nu^{e_\nu}$, $m := \left|\left(\mathbb{Z}^*_{p_1^{e_1}}\right)^{q2^k}\right|$ mit $q2^k = kgV(2,\ldots,\gamma)$. Dann findet Algorithmus 21 mit $Ws \geq 1 - e^{-\binom{\gamma}{2}/m}$ den Teiler $p_1^{e_1}$ von n.*

REMARK. Zur Anwendung dieses Satzes ist eine wichtige Voraussetzung, dass m klein ist. Das erfordert aber, dass $\left|\mathbb{Z}^*_{p_1^{e_1}}\right|$ bis auf einen großen Primteiler γ-glatt ist, was ist nur dann erfüllt, wenn p_1 ein einfacher Primfaktor ist, also $e_1 = 1$.

BEWEIS. Wende Lemma 3.4.2 auf die Zerlegung

$$(\mathbb{Z}_n)^{q2^k} \quad = \quad \left(\mathbb{Z}_{p_1^{e_1}}\right)^{q2^k} \times \ldots \times \left(\mathbb{Z}_{p_r^{e_r}}\right)^{q2^k}$$

an und setze $p = p_1^{e_1}$. Für *zufällige und statistisch unabhängige* $x_i \mod p \in \left(\mathbb{Z}_p^*\right)^{q2^k}$ gilt dann:

$$Ws\left(\exists 0 \leq i < j \leq \gamma : x_i \equiv x_j \mod p\right) \quad \geq \quad 1 - e^{-\binom{\gamma}{2}/m},$$

wobei $m = \left|\left(\mathbb{Z}_p^*\right)^{q2^k}\right|$.

Für $i \geq \left(\log_2 m\right)^2$ gilt $2^i \geq m^{\log_2 m}$, somit ist $x_i \mod p$ „fast" zufällig in $\left(\mathbb{Z}_p^*\right)^{q2^k}$. Für $|i-j| \geq \left(\log_2 m\right)^2$ sind $x_i \mod p, x_j \mod p$ „nahezu" statistisch unabhängig. □

Algorithm 22 Kombination von Pollard ρ und $p-1$ mit nicht-modularer Zufallsfunktion (Alternative)

EINGABE: $n, \gamma \in \mathbb{N}$ mit $n = pq$ und $\mathrm{ggT}(p,q) = 1$.
AUSGABE: $d \notin \{1,n\}$ mit $d \mid n$ oder -1.

(1) $a \in_R \mathbb{Z}_n^*$.
(2) $x_0 := a^{q2^k}$.
(3) **for** $i = 1, \ldots, \gamma/2$ **do**
  (a) $x_{2i} := x_0^{2^i}$.
  (b) $x_{2i+1} := x_0^{3^i}$.
(4) $d := -1$.
(5) **for** $i = 1, \ldots, \gamma/2 - 1$ **do**
  (a) **for** $j = i+1, \ldots, \gamma/2$ **do**
    (i) **if** $\mathrm{ggT}\left(x_i - x_j, n\right) \notin \{1,n\}$ **then**
      $d := \mathrm{ggT}\left(x_{2i} - x_i, n\right) \notin \{1,n\}$.
      Breche Schleife ab ($i := \gamma$, $j := y+1$).
(6) **return(d).**

Eine alternative nicht-modulare Zufallsfunktion wird in Algorithmus 22 verwendet.

Ein möglicher Vorteil dieses Algorithmus ist, dass 3^i schneller wächst als 2^i. Die Unabhängigkeit von $x_0^{2^i}$ und $x_0^{3^i}$ modulo p wird plausibel.

Dies ist aber nur ein unbefriedigendes Teilergebnis. Die Anzahl der Schritte ist nicht besser als bei Betrachtung aller Exponenten $x_0, x_0^2, \ldots, x_0^n$. Die ganze Zufallsrekursion wäre also unnötig, wenn es nicht gelänge, die Anzahl der Tests zu reduzieren. Eine dafür geeignete Methode wird im nächsten Abschnitt behandelt.

3.6. FFT (Fast Fourier Transform) Polynom Multiplikation

Für die FFT benötigen wir den Begriff der „primitiven Einheitswurzel". Zuvor sei jedoch zur Erinnerung die Definition eines Rings wiederholt.

Ring

DEFINITION. **(Ring)** Ein *Ring* ist eine Menge R mit zwei Operationen

$$\begin{aligned} +: \quad & R \times R \to R \quad : \quad a+b=c \quad \forall a,b,c \in R \\ \cdot: \quad & R \times R \to R \quad : \quad a \cdot b = c \quad \forall a,b,c \in R \end{aligned}$$

und zwei besonderen Elementen 0 und 1. Es müssen folgende Axiome erfüllt sein:

(1) $(R,+)$ ist kommutativ: $a+b=b+a$ mit $e=0$ und $a+(-a)=0$.
(2) $(R,+)$ und $(R,\cdot)$ sind assoziativ: $(a+b)+c = a+(b+c)$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
(3) Distributivgesetz: $a \cdot (b+c) = a \cdot b + a \cdot c$.

In einem *kommutativen Ring mit 1* gilt zusätzlich, dass $(R,\cdot)$ kommutativ ist und ein neutrales Element 1 bzgl. der Multiplikation exisitert.

Primitive Einheitswurzel

DEFINITION. Eine **primitive Einheitswurzel** w in dem kommutativen Ring mit 1_R erfüllt folgende Eigenschaften

$$w^m = 1_R \quad \text{und} \quad w^j \neq 1_R \,\text{fuer}\, 1 \leq j < m$$
$$\sum_{j=0}^{m-1} w^{jp} = 0 \quad \text{fuer} \quad 1 \leq p < m.$$

Sei R kommutativer Ring mit Eins $1_R \in R$ und primitiver Einheitswurzel $w = w_m$.

Fouriertransformation

DEFINITION. Die **Fouriertransformation** $F_m : R^m \to R^m$ ist definiert als

$$F_m(a) = \left[\omega_m^{ij}\right]_{0 \leq i,j \leq m-1} a$$
$$F_m(a_0, \ldots, a_{m-1}) = (a'_0, \ldots, a'_{m-1}) \quad \text{mit } a'_i = \sum_{j=0}^{m-1} \omega_m^{ij} a_j,$$

a'_i ist also der Wert des Polynoms $f := \sum_{j=0}^{m-1} x^j a_j \in R[x]$ an der Stelle ω^i.

Die Berechnung von $F_m(a)$ geht in dem speziellen Fall sehr schnell, wo m eine Zweierpotenz ist (s. Lemma 3.6.1).

LEMMA 3.6.1. *Für $m = 2^k$ geht die Abbildung $a \mapsto F_m(a)$ in $2^{k-1}(k-1)$ Multiplikationen und $2^k k$ Additionen (also $O(m \log m)$ arithmetischen Schritten $(+,\cdot)$[1]) über R.*

BEWEIS. $F_m(a_0, \ldots, a_{m-1}) = (a'_0, \ldots, a'_{m-1})$, $\omega' := \omega^2$ ist primitive 2^{k-1}te Einheitswurzel (denn ω ist 2^k-te primitive Einheitswurzel).

$$a'_i = \overbrace{\sum_{j=0}^{2^{k-1}-1} \underbrace{\omega^{i2j}}_{(\omega')^{ij}} a_{2j}}^{\text{gerade Indizes}} + \overbrace{\sum_{j=0}^{2^{k-1}-1} \underbrace{\omega^{i(2j+1)}}_{(\omega')^{ij}\cdot\omega^i} a_{2j+1}}^{\text{ungerade Indizes}}$$

Wir sortieren jetzt die Matrix so um, dass erst die Zeilen mit den geraden Indizes, danach die mit den ungeraden Indizes kommen:

$$\begin{bmatrix} a'_0 \\ a'_1 \\ \vdots \\ \\ \\ a'_{m-1} \end{bmatrix} = \begin{bmatrix} & & \vdots & \\ & (\omega')^{ij} & \vdots & \\ & & \vdots & \\ \cdots & \cdots & \omega^{ij} & \\ & & & \\ & & & \end{bmatrix} \begin{matrix} a_0 \\ a_2 \\ \vdots \\ a_{m-2} \\ a_1 \\ a_2 \\ \vdots \\ a_{m-1} \end{matrix}$$

Wir betrachten die Stellen i und $i + \overbrace{2^{k-1}}^{m/2}$, denn an diesen Stellen entsteht fast der gleiche Wert, so dass wir ihn kein zweites Mal berechnen müssen: $\omega^{i+2^{k-1}} = \omega^i \cdot \omega^{2^{k-1}} = \omega^i \cdot$

[1] Wesentlich ist, dass keine Divisionen verwendet werden.

$(-1) = -\omega^i$. Es ist

$$\begin{aligned} F_m(a_0,\ldots,a_{m-1}) &= \left(F_{m/2}(a_0,a_2,\ldots,a_{m-2}),F_{m/2}(a_0,a_2,\ldots,a_{m-2})\right) + \\ &\quad \left(F_{m/2}(a_1,a_3,\ldots,a_{m-1}),-F_{m/2}(a_1,a_3,\ldots,a_{m-1})\right). \end{aligned}$$

Das Minuszeichen folgt aus $\omega^{i+2^{k-1}} = -\omega^i$. Wegen des „Störfaktors" ω_i muss im zweiten Summanden jeweils die i-te Komponente mit ω^i multipliziert werden.

Sei $M(m)/A(m)$ die Anzahl der Multiplikationen/Additionen bei $a \mapsto F_m(a)$.
Ind. Anfang: $m = 2$:

$$F_2\left(a_{0,}a_1\right) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} a_0+a_1 \\ a_0-a_1 \end{bmatrix},$$

wobei -1 der Wert der zweiten primitiven Einheitswurzel ist. Somit ist für $m = 2$ die Anzahl der Multiplikationen $M(2) = 0$ und die Anzahl der Additionen $A(2) = 2$.
Ind. Schritt: $k \to k+1$:

$$\begin{array}{rcc} & & \overbrace{\qquad\qquad}^{\text{FFT der halben Ordnung}} \quad \overbrace{\qquad}^{\text{Mult. mit allen } \omega^i} \\ M\left(2^k\right) & \leq & 2\cdot M\left(2^{k-1}\right) \quad + \quad 2^{k-1} \\ A\left(2^k\right) & \leq & 2\cdot A\left(2^{k-1}\right) + 2^k \\ & \downarrow & \text{Einsetzen der Induktionsvoraussetzung} \\ M\left(2^k\right) & \leq & 2\cdot 2^{k-2}(k-2) + 2^{k-1} = 2^{k-1}(k-1) \\ A\left(2^k\right) & \leq & 2\cdot 2^{k-1}(k-1) + 2^k = 2^k k \end{array}$$

□

LEMMA 3.6.2. $\frac{1}{m}\left[\omega_m^{ij}\right]\left[\omega_m^{-jk}\right] = I_m$ *über R.*

BEWEIS. $\frac{1}{m}\sum_{j=0}^{m-1}\omega_m^{ij-jk} = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$ □

Zirkuläre Konvolution, Polynomprodukt

DEFINITION. Die **zirkuläre Konvolution** $c = a \otimes b \in R^n$ von $a, b \in R^n$ ist $c = \left(c_0,\ldots,c_{m-1}\right)$ mit

$$c_L := \sum_{i+j \equiv L \bmod m} a_i b_j, \quad L := 0,\ldots,m-1.$$

Beachte, dass das **zirkuläre Polynomprodukt**

$$\left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{j=0}^{m-1} b_j x^j\right) \equiv \sum_{L=0}^{m-1} c_L x^L \mod (x^m-1)$$

ist, denn x^m wird mit 1 identifiziert.

$c = a \otimes b$ besteht aus den Koeffizienten des zirkulären Produktpolynoms $f \cdot g \mod (x^m-1)$.

Der Konvolutionssatz nimmt nun eine sehr einfache Form an:

PROPOSITION 3.6.3. ***Konvolutionssatz*** *für zirkuläre Konvolution und zirkuläres Polynomprodukt:*
Für $a, b \in R^m$ gilt $F_m(a \otimes b) = F_m(a) \cdot F_m(b)$.

COROLLARY 3.6.4. *Die Abbildung $a, b \mapsto a \otimes b$ mit $a, b \in R^m$ geht mit $3 \cdot 2^{k-1}(k-1) + 2^k$ Multiplikationen und $3 \cdot 2^k k$ Additionen über R.*

BEWEIS.

2 mal	F_m	$2^{k-1}(k-1)$ Mult., $2^{k+1}k$ Add.
	F_m^{-1}	$2^{k-1}(k-1)$ Mult., $2^k k$ Add.
		2^k gliedw. Mult.
	Insgesamt :	$2^{k-1}3\cdot(k-1)+2^k$ Mult., $3\cdot 2^k k$ Add.

(Wir vernachlässigen die 2^k Divisionen durch 2^k zu $F_m^{-1}(a) = \frac{1}{m}\left[\omega^{-ij}\right] a$) □

Für $k \geq 2$ erfordert die FFT-Berechnung von $a \otimes b$ weniger Multiplikationen als der triviale Algorithmus, der 2^{2^k} Schritte braucht.

Algorithm 23 Zirkuläres Polynomprodukt $f \cdot g \mod x^m - 1, R = \mathbb{Z}_n$

EINGABE: $a, b \in \mathbb{Z}_n^m$, $f = \sum_{i=0}^{m-1} a_i x^i$, $g = \sum_{j=0}^{m-1} b_j x^j$, $m = 2^k$

AUSGABE: $c = a \otimes b \in \mathbb{Z}_n^m$, $f \cdot g = \sum_{i=0}^{m-1} c_i x^i \mod (x^m - 1)$

(1) Wähle Primzahlen $p_1, \ldots, p_t$ mit $p_i = 1 \mod 2^k$ und

$$P := \prod_{i=1}^{t} p_i > 2^k \frac{n^2}{4}.$$

(2) Berechne 2^k-te primitive Einheitswurzel $\omega_i \mod p_i$.

(3) Berechne das zirkuläre Produkt

$$a \otimes b = F_{2^k}^{-1}\left(F_{2^k}(a) \cdot F_{2^k}(b)\right) \mod p_i \text{ fuer } i = 1, \ldots, t$$

Mit $(\cdot)$ ist hier die komponentenweise Multiplikation gemeint.

$$(a \otimes b \mod p_i, i = 1, \ldots, t) \overset{CRT}{\mapsto} a \otimes b \in [-\frac{p}{2}, \frac{p}{2}[^{2^k} \mapsto a \otimes b \mod n$$

Korrektheit: (Algorithmus 23)

$$\sum_{i+j \equiv L \mod 2^k} a_i b_j \in \left[-\frac{p}{2}, \frac{p}{2}\right] \text{ fuer } a_i, b_j \in \left[-\frac{n}{2}, \frac{n}{2}\right]$$

$$\sum_{i+j \equiv \nu \mod 2^k} a_i b_j \leq 2^k n^2/4 < P.$$

Die Repräsentanten $a_i, b_j \in \mathbb{Z}_n$ wählt man in $[-n/2, n/2[$ und $\mathbb{Z}_n \cong [-n/2, n/2[$.

Laufzeit: Bei gegebenem $p_1, \ldots, p_t$, Wurzeln $\omega_1, \ldots, \omega_t$ und deren Inversen $\omega_1^{-1}, \ldots, \omega_t^{-1}$ und $\sigma_i = \delta_{ij} \mod p_j$, $1 \leq i, j \leq t$ benötigt Schritt 3 $\left(3 \cdot 2^{k-1}(k-1) + 2^k\right) \cdot t$ Multiplikationen in $\mathbb{Z}_{p_i}$, $i = 1, \ldots, t$.

Praktischer Hinweis: Man wählt die Module $p_i \in [0, 2^{32}[$ und zwar möglichst groß, fängt also „von hinten" an zu suchen. Ansonsten kann man auch auf die doppelte Bitlänge wechseln: $p_i \in [0, 2^{64}[$.

LEMMA 3.6.5. *Sei p prim. Dann gilt: Es gibt eine m-te primitive Einheitswurzel $w \in \mathbb{Z}_p^*$ mit $w^m \equiv 1 \mod p$ und $w^j \not\equiv 1 \mod p$ für $1 \leq j < m \Leftrightarrow p \equiv 1 \mod m$.*

BEWEIS. „⇒": Sei w eine m-te primitive Einheitswurzel in $\mathbb{Z}_p^*$, dann gilt $w^m \equiv 1 \mod p$, also gilt nach Fermat $p - 1 \mid m$.
Gleichzeitig gilt für $1 \leq j < m$ $w^j \not\equiv 1 \mod p$ und damit $p - 1 \nmid j$ mit $j < m$. Daraus folgt

$m = p-1$, also $p \equiv 1 \mod m$ (eigentlich klar, da $\mathbb{Z}_p^*$ zyklisch ist, also ein erzeugendes Element g besitzt, und alle $a \in \mathbb{Z}_p^*$ auch als $a = g^i$ geschrieben werden können. Damit a m-te primitive Einheitswurzel in $\mathbb{Z}_p^*$ ist, muss also gelten $im = p-1$, also $p \equiv 1 \mod m$).

„$\Leftarrow$": Sei $p \equiv 1 \mod m$. Es gilt $\left|\mathbb{Z}_p^*\right| = p-1$ und damit $m \mid p-1$.

Sei Ψ_m eine $(m,1)$-Abbildung:

$$\Psi_m : \quad x \quad \mapsto \quad x^m \mod p$$

mit

$$\#\Psi_m^{-1}(a) = \begin{cases} m & , a \in \left\{\Psi_m(x) \mid x \in \mathbb{Z}_p^*\right\} \\ 0 & , \text{sonst.} \end{cases} .$$

$\Psi_m^{-1}(1)$ ist eine multiplikative Gruppe der Ordnung m. Sei $\langle w \rangle = \Psi_m^{-1}(1)$, dann ist w primitive Einheitswurzel modulo p. □

Unser Ziel ist es, $\mathrm{ggT}\left(x_{2i} - x_{j+1}, n\right) \notin \{1,n\}$, $1 \le i,j \le \gamma/2$ zu berechnen:

Algorithm 24 Schnelle Berechnung von $\mathrm{ggT}\left(x_{2i} - x_{2j+1}, n\right)$ mit $1 \le i,j \le \gamma/2$

---

(1) Sei $f :\equiv \prod_{j=1}^{\gamma/2}\left(x - x_{2j+1}\right) \mod n$.
(2) **for** $i = 1,\ldots,\gamma/2$ **do**
  (a) Berechne $f\left(x_{2i}\right)$.
(3) $X := \prod_{i=1}^{\gamma/2} f\left(x_{2i}\right)$.
(4) Teste, ob $\mathrm{ggT}(X,n) \notin \{1,n\}$.

Laufzeit: Die Anzahl der arithmetischen Schritte mittels FFT und der zirkulären Konvolution $\otimes$ist $O\left(\gamma\left(\log_2 \gamma\right)^2\right)$: Die Schleife in Schritt 2 benötigt $O\left((\gamma/2)\left(\log_2 \gamma\right)^2\right)$-viele Schritte. Die Berechnung des Produkts in Schritt 3 benötigt nochmal $O((\gamma/2))$-viele Schritte. Die Berechnung des ggT benötigt wie gehabt $O\left(\log_2 \gamma\right)$-viele Schritte.

Jetzt gilt es zu zeigen, dass man mit dieser Methode schnell rechnen kann. Das geht mit der Fourier-Transformation:

3.6.1. Einschub: „Effiziente Arithmetik in $\mathbb{Z}_m$ und $\mathbb{F}_{p^n}$". Ein Einschub von Roger Fischlin, siehe auch [**R. Fischlin**, Webseite der MI].

3.6.2. Pollard's $p-1$-Algorithmus mit FFT [**Montgomery, Silverman**, Section 2]**.** In diesem Abschnitt soll die von Herrn Schnorr ausgeteilten Arbeit von [**Montgomery, Silverman**] besprochen werden. Es geht dabei um die Anwendung der Fourier-Transformation in der Stufe 2 in Pollard's $p-1$-Algorithmus. Um Verwirrungen zu vermeiden richtet sich die Notation nach der der Arbeit von Montgomery und Silverman.

Seien $n = p \cdot q$ mit p,q prim ≥ 3 und $B_1 < B_2$ Glattheitsschranken (kommt später). Definieren wir nun das Produkt der grössten Primzahlpotenzen $\le B_1$

$$M := \prod_{p_i^{\alpha_i} \le B_1 < p_i^{\alpha_i+1}} p_i^{\alpha_i}$$

und $gl\left(B_1\right)$ mit $O\left(B_1\right)$ Multiplikationen modulo n.

Stufe 1: Wähle $a \in_R \mathbb{Z}_n^*$, $H := a^M \mod n$ und teste, ob gilt

$$\mathrm{ggT}(H-1,n) \notin \{1,n\}.$$

$$\left(\left|\mathbb{Z}_p^*\right| B_1 - \text{glatt} \quad \Rightarrow \quad a^M = 1 \mod p \quad \Rightarrow \quad p \mid \mathrm{ggT}(H-1,n)\right)$$

Ist das der Fall, so wurde ein Faktor von n gefunden.

Stufe 2: Angenommen, $\left|\mathbb{Z}_p^*\right| = q_1 \cdot q_2$ mit q_1 B_1-glatt und $B_1 < q_2 \leq B_2$ und q_2 prim (wobei q_2 kein Primteiler von M sein kann).

Berechne nun

$$Q_{p_i} := H^{p_i} \quad i = 1,2,\ldots \quad mit\, B_1 < p_1 < p_2 < \ldots \leq B_2,\, p_i \,\text{prim}$$

und

$$P := \prod_j \left(Q_{p_j} - 1\right) \mod n.$$

Teste nun, ob der $\mathrm{ggT}(P,n) \notin \{1,n\}$ ist (nach Voraussetzung ist $p \mid (\underbrace{Q_{q_2}}_{\in [0.n[} - 1)$ und $p \mid P$).

Laufzeit: Wir wissen, dass sich $Q_{p_{j+1}} := Q_{p_j} \cdot H^{p_{j+1}-p_j}$ in $\leq 2 \cdot \left(p_{j+1} - p_j\right)$ Multiplikationen $\mod n$ berechnen läßt. Demnach ergibt sich eine Laufzeit mit $\leq 2\left(B_2 - B_1\right)$ Multiplikationen modulo n (die 2 kommt von der binären Methode).

Ziel ist aber eine Laufzeitverbesserung, so dass $B_2 \sim B_1^2$ ist. Dazu benötigen wir den nächsten Abschnitt.

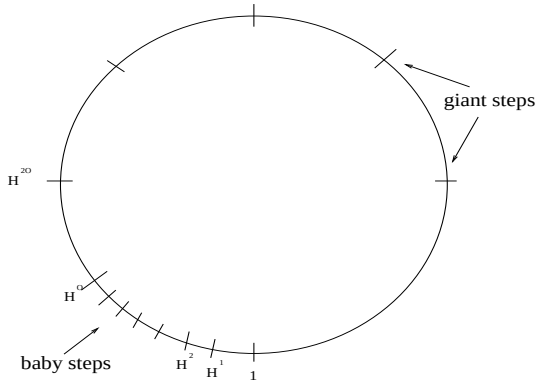
3.6.3. Die baby step, giant step Methode in $\langle H \mod p\rangle$. Sei p unbekannt unter der Annahme, dass gilt

$$\left|\langle H \mod p\rangle\right| \leq B_2.$$

Setze zunächst die Schranke

$$\Theta = \left\lceil \sqrt{B_2} \right\rceil.$$

ABBILDUNG 3.6.1. Zyklische Gruppe $\langle H \mod p\rangle$.



Nach Annahme (von Stufe 2) und weil $\Theta = \left\lceil \sqrt{B_2} \right\rceil$ ist, gibt es $0 < u, v \leq \Theta$ mit

$$H^{v\Theta} = H^u$$

(bedeutet: einer der giant steps fällt in das Intervall der baby steps).

Ziel ist es nun, u und v in $O\left(\Theta\left(\log_2\Theta\right)^2\right)$ mit einer arithmetischen Anzahl an Schritten modulo n zu finden (zum Vergleich: durch triviales Ausprobieren kriegt man eine Laufzeit von $O\left(\Theta^2\right)$ Schritten hin, indem man für alle $0 \leq u, v \leq \Theta$ testet, ob gilt ggT $\left(H^{v\theta} - H^u, n\right) \notin \{1, n\}$). An dieser Stelle kommt die FFT ins Spiel, denn mit ihrer Hilfe werden wir die gewünschte Laufzeit erreichen können.

REMARK. Wir reden hier über eine Gruppe, deren Elemente wir nicht kennen, da p unbekannt ist. Wäre p bekannt, so könnten wir uns einfach die Elemente erzeugen, sie sortieren und die giant steps in die baby steps einordnen und so einen Match finden. Die Laufzeit dieses Verfahrens wäre dann $O\left(\Theta\log_2\Theta\right)$.

In [**Buchmann1999**, Kapitel 9.3] wird die Babystep-Giantstep-Methode von Shanks zur Berechnung des diskreten Logarithmus besprochen $\alpha = \gamma^x$ mit $\langle\gamma\rangle = \mathbb{Z}_p$. Man setzt dort

$$m = \sqrt{n}$$

und formuliert

$$x = qm + r \quad 0 \leq r < m.$$

Mit Hilfe der BSGS-Methode möchten wir nun q und r berechnen:

$$\begin{aligned} & \gamma^{qm+r} &=& \gamma^x = \alpha \\ \Rightarrow \quad & (\gamma^m)^q &=& \alpha\gamma^{-r}. \end{aligned}$$

Die *baby steps* sind nun die Elemente der Menge

$$B = \left\{\left(\alpha\gamma^{-r}, r\right) \mid 0 \leq r < m\right\}.$$

Ist ein Element von β gleich $(1, r)$, also $\alpha\gamma^{-r} = 1$, dann hat man eine Lösung $x = r$ gefunden, denn aus

$$\alpha\gamma^{-r} = 1 = (\gamma^m)^q$$

folgt, dass mq ein Vielfaches der Gruppenordnung von γ sein muss und damit

$$\alpha\gamma^{-r} = 1 \quad \Leftrightarrow \quad \gamma^r = \alpha = \gamma^x.$$

Findet man kein solches Paar, so berechnet man die Giantsteps

$$\delta = \gamma^m$$

und prüft, ob für $q = 1, 2, 3, \ldots$ das Gruppenelement δ^q als erste Komponente in einem Element aus B vorkommt (also $(\delta^q, r) \in B$). Sollte dies der Fall sein, so gilt

$$\alpha\gamma^{-r} = \delta^q = \gamma^{qm},$$

und damit

$$\alpha = \gamma^{qm+r} \quad \Rightarrow \quad x = qm + r.$$

Die giantsteps bilden die Menge $G = \{(\gamma^m)^q \mid q = 1, 2, 3, \ldots\}$.

Ein Nachteil des Algorithmus ist die Große Anzahl der Elemente (baby steps), die gespeichert werden müssen. Wenn man jedoch annimmt, dass die Anzahl der Vergleiche zum Auffinden von Elementen aus B mittels einer Hashtabelle konstant gehalten werden können, benötigt der Algorithmus $O\left(\sqrt{|G|}\right)$-viele Vergleiche und Multiplikationen und muss genausoviele Elemente speichern. In der Praxis ist der Algorithmus für $|G| > 2^{160}$ nicht mehr einsetzbar, da dann die Laufzeit und der Speicherplatz nicht mehr akzeptabel sind (wenn für eine Zahl etwa 2 Byte Speicherplatz benötigt wird, so wird für $|G| > 160$ zur Speicherung aller Zahlen $O\left(2 \cdot 2^{80}\right) = O\left(2 \cdot \left(2^{40}\right)^{20}\right)$, also mehr als 2 Millionen Terabyte benötigt.

3.6.4. FFT-Implementation von Stufe 2 [Montgomery, Silverman, Section 5]. Sei

$$f(x) = \prod_{0<u\leq\Theta} (x - H^u) \in \mathbb{Z}_n[x], \quad \operatorname{grad}(f) = \Theta$$

Damit unsere Lemmas „zuschlagen" können, nehmen wir an, dass zusätzlich gilt

$$\Theta = 2^k.$$

Damit verfahren wir nun wie folgt:

Stufe 2) a): Berechne f rekursiv mittels zirkulärer Konvolution der Ordnung $2, 4, \ldots, 2^k$ (die Ordnung der zirkulären Konvolution ist gleich zweimal die Ordnung des Polynoms). Eine zirkuläre Konvolution der Ordnung 2^x verursacht nach Korollar 3.2.4 auf Seite 35 folgende Kosten für Multiplikationen modulo n

$$\begin{array}{rcl} 2^k & \text{kostet} & O\left(2^k k\right) \\ 2^{k-1} & \text{kostet} & O\left(2^k 2k\right) \\ \vdots & \vdots & \vdots \\ 2 & \text{kostet} & O\left(2^k k^2\right). \end{array}$$

Die Kosten für die zirkuläre Konvolution betragen also $O\left(\underbrace{2^k}_{=\Theta} k^2\right)$ mit $1,5\sqrt{B_2}\cdot\frac{1}{4}\left(\log_2 B_2\right)^2$ Multiplikationen modulo n (die konstanten Faktoren sind also wie man sieht klein).
Damit haben wir aber nur die baby steps – wir wollen jedoch die Differenz der baby steps und der giant steps. Deshalb folgt jetzt

Stufe 2) b): Berechne $f\left(H^{v\Theta}\right)$ für $0 < v \leq \Theta$ (die giant steps) und dann $\prod_{0<v\leq\Theta} f\left(H^{v\Theta}\right)$ und teste schließlich, ob

$$\operatorname{ggT}\left(\prod_{0<v\leq\Theta} f\left(H^{v\Theta}\right), n\right) \notin \{1, n\}$$

ist.

PROPOSITION 3.6.6. *Die „FFT-Berechnung" $f\left(H^{v\Theta}\right)$ für $0 < v \leq \Theta$ geht in $O\left(\Theta \log_2 \Theta\right)$ arithmetischen Schritten modulo n (es werden nur Additionen und Multiplikationen benötigt!).*

Damit ist Schritt 2) b) einfacher als Schritt 2) a), auch wenn in [**Montgomery, Silverman**] etwas anderes behauptet wird. Der Satz soll nun in einem eigenen Kapitel bewiesen werden.

3.6.5. Berechnung von Polynomwerten entlang einer geometrischen Progression. Sei R kommutativer Ring mit 1_R und $f \in R[x]$ ein Polynom vom Grad $\leq m-1$ mit $f \in R[x]$:

$$f = \sum_{i=0}^{m-1} a_i x^i \quad , x_i \in R.$$

Berechne die Werte von f entlang der geometrischen Progression er^i für $i = 0, \ldots, m-1$ mit $e, r \in R$:

$$f\left(e \cdot r^i\right) \quad 0 \leq i < m, \quad e, r \in R \text{ beliebig } (\text{in unserer Anwendung ist } e = 1).$$

Nun erweitern wir die diskrete FFT zu

$$\begin{array}{rccc} F_m: & R^m & \to & R^m \\ & \vec{a} & \mapsto & \left[r^{ij}\right]_{0\le i,j<m} \cdot \vec{a}, \end{array}$$

so dass der transformierte Vektor

$$F_m\left(a_0,\ldots,a_{m-1}\right) = \left(a'_0,\ldots,a'_{m-1}\right)$$

aus den Polynomwerten

$$a'_i = \sum_{j=0}^{m-1} r^{ij}\cdot a_j = f\left(r^i\right)$$

besteht.

Den allgemeineren Fall mit $e \neq 1$ brauchen wir in unserer Anwendung zwar nicht, trotzdem ist aber auch dieser interessant:

Für $\vec{a},\vec{b} \in R^m$ sei das **gliedweise Produkt**

$$\vec{a} * \vec{b} = \left(a_0b_0, a_1b_1, \ldots, a_{m-1}b_{m-1}\right) \in R^m$$

und

$$\begin{array}{rcl} e^* & := & \left(\overset{0}{=1}, e^1, \ldots, e^{m-1}\right) \in R^m \\ r^* & := & \left(\overset{0}{=1}, r^1, \ldots, r^{m-1}\right) \in R^m. \end{array}$$

Es gilt

$$\begin{array}{rcl} F_m\left(e^* * \overrightarrow{a}\right) & = & \left(\sum_{j=0}^{m-1} r^{ij}a_je^j \mid i=0,\ldots,m-1\right) \\ & = & \left(f\left(er^i\right) \mid i=0,\ldots,m-1\right). \end{array}$$

LEMMA 3.6.7. *(wird für den Beweis von Satz 3.6.6 auf der vorherigen Seite benötigt) Im Fall $m = 2^k$ geht die Abbildung*

$$\vec{a} \mapsto F_m(\vec{a}) \in R^m$$

bei gegebenem r^* und r^{2k} in $3\cdot 2^k(k-1)+1$ Multiplikationen und $2^k\cdot k$ Additionen/Subtraktionen über k (es wird also keine Division benötigt!).

BEWEIS. Die Trennung gerader und ungerader Indizes j führt zu

$$\begin{array}{rcl} a'_i & = & \sum_{j=0}^{2^k-1} r^{ij}a_j \\ & = & \underbrace{\sum_{j=0}^{2^{k-1}-1} r^{2ij}a_{2j}}_{\text{gerade Indizes}} + \underbrace{\sum_{j=0}^{2^{k-1}-1} r^{2ij+i}a_{2j+1}}_{\text{ungerade Indizes}}. \end{array}$$

Die Transformation $F_{m/2} = F_{2^{k-1}}$ bezieht sich auf r^2, dann gilt:

$$\begin{aligned}\overrightarrow{a'} &= \left(F_{m/2}\left(a_0, a_2, \ldots a_{m-1}\right), r^{2^k} F_{m/2}\left(a_0, a_2, \ldots, a_{m-2}\right)\right) \\ &\quad + \left(F_{m/2}\underbrace{\left(a_1, \ldots, a_{m-1}\right)}_{\text{ungerade Indizes}}, F_{m/2}\underbrace{\left(a_1, \ldots, a_{m-1}\right)}_{\text{ungerade Indizes}}\right) * r^*.\end{aligned}$$

Was kostet diese Rekursion für $k \geq 2$?

$$\text{Multiplikationen}: \quad M\left(2^k\right) \leq 2 \cdot M\left(2^{k-1}\right) + \underbrace{2^{k-1} + \overbrace{2^k - 1}^{\text{Mult. in } r^*}}_{=3\cdot 2^{k-1}-1}$$

$$\text{Additionen}: \quad A\left(2^k\right) \leq 2 \cdot A\left(2^{k-1}\right) + 2^k$$

Ind. Anf.: $k = 2$: $\begin{bmatrix} 1 & 1 \\ 1 & r^{2^{k-1}} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} a_0 + a_1 \\ a_0 + a_1 \cdot r^{2^{k-1}} \end{bmatrix}$ also $M(2) = 1, A(2) = 2$

Ind. Schritt: $k \to k+1$:

$$\begin{aligned} M\left(2^k\right) &\leq 2 \cdot M\left(2^{k-1}\right) + 3 \cdot 2^{k-1} - 1 \\ A\left(2^k\right) &\leq 2 \cdot A\left(2^{k-1}\right) + 2^k \\ &\downarrow \text{ Induktionsvoraussetzung} \\ M\left(2^k\right) &\leq 2 \cdot \left(3 \cdot 2^{k-2}(k-2) + 1\right) + 3 \cdot 2^{k-1} - 1 = 3 \cdot 2^{k-1}(k-1) + 1 \\ A\left(2^k\right) &\leq 2 \cdot \left(2^{k-1}(k-1)\right) + 2^k = 2^k k \end{aligned}$$

□

Die Berechnung

$$\vec{d}, e, r \mapsto \vec{d}, e^*, r^* \mapsto e^* * \vec{d}, r^*$$

geht in $\left(3 \cdot 2^k - 5\right)$ Multiplikationen. Somit folgt aus

$$F_{2^k}\left(e^* * \vec{d}\right) = \left(f\left(er^i \mid i = 0, \ldots, 2^k - 1\right)\right)$$

und Lemma 3.6.6 der folgende Korollar:

COROLLARY 3.6.8. *Sei $f \in R[x]$ vom Grad $m-1$, $m = 2^k$.*
Dann geht

$$e, r, \vec{a} \mapsto \left(f\left(er^i\right) \quad i = 0, \ldots, 2^{k-1}\right)$$

in $3 \cdot 2^{k-1}(k+1) - 4$ Multiplikationen und $2^k \cdot k$ Additionen.

Vergleich mit der Methode in [Montgomery, Silverman]:

In Sektion 5 wird vorgeschlagen, für $n = 2^k$ die Berechnung

$$e, r, \vec{a} \mapsto \underset{\in R^{2^{k+1}}}{\vec{y}, \vec{z}} \mapsto \vec{y} \otimes \vec{z} \mapsto \left(f\left(er^i\right) \quad i = 0, \ldots, 2^k - 1\right)$$

mit einer zirkulären Konvolution $\vec{y}, \vec{z} \mapsto \vec{y} \otimes \vec{z}$ der Ordnung 2^{k+1} durchzuführen. Dieses geht nach Korollar 3.6.4 in Zeit $3 \cdot 2^k(k-1) + 3 \cdot 2^{k+1} + 3$ Multiplikationen und $3 \cdot 2^{k+1}(k+1)$

Additionen über R. Ausserdem benötigt die Transformation $\vec{a}, e, r \mapsto \vec{y}, \vec{z}$ und $\vec{y} \otimes \mapsto \left(f\left(er^i\right) \mid 0 \leq i < 2^k \right)$ noch $O\left(2^k\right)$ Schritte mit Division.

CONCLUSION. Die Methode von Korollar 3.6.8 auf der vorherigen Seite ist doppelt so schnell wie [**Montgomery, Silverman**] und benötigt weder Divisionen noch primitive Einheitswurzeln, $2^{-1} \in R$ nicht erforderlich.

3.6.6. Division durch 2 in $\mathbb{Z}_n$, n ungerade. Wird benötigt bei der inversen FFT

$$\mathrm{FFT}_{2^k}^{-1}(\vec{a}) = \frac{1}{2^k}\left[w_{2^k}^{-ij}\right]_{0 \leq i,j < 2^k} \vec{a}$$

mit

$$2^{-1} = \frac{n+1}{2} \mod n.$$

Weil $n+1$ gerade ist, entsteht die Binärdarstellung von $2^{-1} \mod n$ durch Wegstreichen des niedrigsten Bits der Binärdarstellung von n.

Für $a \in [0, n[\cong \mathbb{Z}_n$ gilt

$$a2^{-1} = \begin{cases} a/2 & a \text{ gerade} \\ \underbrace{\frac{a-1}{2}}_{\in \left[0, \frac{n+1}{2}\right[} + \underbrace{\frac{n+1}{2}}_{0 \mod n} & a \text{ ungerade} \end{cases} \mod n.$$

CONCLUSION. Die Kosten der Division durch 2 entsprechen der einer „halben" Addition: Im Fall „a gerade" mus lediglich das kleinste Bit auf 0 gesetzt werden und im Fall „a ungerade" mus von $a-1$ das kleinste Bit gestrichen und $(n+1)/2$ aufaddiert werden.

3.6.7. Stufe II von Pollard $p-1$ und EK-Methode. Gegeben sei $x_i, y_i \in \mathbb{Z}_n$, $i = 0, \ldots, m-1$. Teste $\mathrm{ggT}\left(x_i - y_j, n\right) \notin \{1, n\}$ für $0 \leq i, j < m$. Dabei bilden die x_i, y_j keine geometrische Progression. Um die triviale Methode mit $O\left(m^2\right)$ Schritten zu schlagen, berechnet man

$$f(x) = \prod_{j=1}^{m} \left(x - y_j\right) \in \mathbb{Z}_n[x]$$

und $f(x_i)$, $i = 1, \ldots, m$. Es gilt

$$\prod_{i=1}^{m} f(x_i) = \prod_{1 \leq i,j \leq m} \left(x_i - y_j\right).$$

3.6.8. Auswertung eines Polynoms vom Grad n an n Punkten
[Borodin, Moenck1974]. Sei R kommutativer Ring mit Division und N-ter primitiver Einheitswurzel $w = w_N$ mit $N = 2^k$.

EXAMPLE. $R = \mathbb{Z}_{p_1 \cdot \ldots \cdot p_r}$ mit $p_i \equiv 1 \mod 2^k$ für $i = 1, \ldots, r$.

Das Produkt zweier Polynome vom Grade m und n mit $m + n < N$ hat N Koeffizienten und kann in

$$\frac{9}{2} N \log N + 5N + Ldt$$

arithmetischen Operationen berechnet werden. Das entspricht einer zirkulären Konvolution der Ordnung 2^{k+1} bei der die führenden Koeffizienten der Eingabe-Polynome Null sind.

Nach Korollar 3.6.4 ergibt das

$$\begin{array}{ll} 1{,}5\cdot 2^{k+1}k+O\left(2^k\right) \text{ Mult.} & 2\cdot 2^{k+1}k+O\left(2^k\right) \text{ Add.} \\ 3N\log N \text{ Mult.} & 6N\log N \text{ Add.} \end{array}$$

3.6.9. Auswertung eines Polynoms $P\in R[x]$ an einer Stelle a. Division durch $(x-a)$ mit Rest

$$p(x)=q(x)\cdot(x-a)+r(x)$$

mit $\operatorname{grad}(r(x))=0$. Also $p(a)=r$.

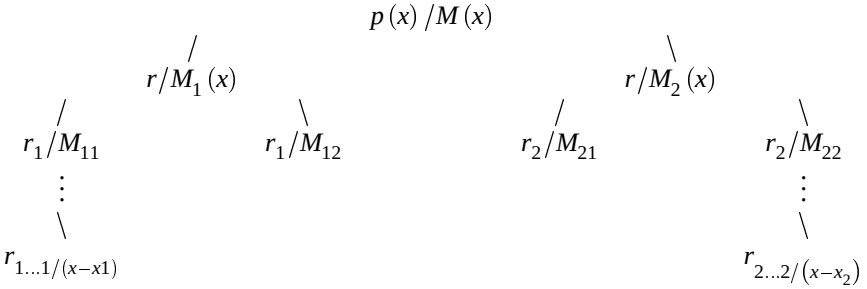
3.6.10. Auswertung an mehreren Stellen $x_1,\ldots,x_m$. Die Berechnung von $M_1(x)=\prod_{i=1}^{m}(x-x_i)$ geht in $m(\log m)^2$-vielen Schritten. Zur Division mit Rest

$$r(x)=q_iM_1(x)+r_i(x) \quad ,i=1,2$$

mit $\operatorname{grad} r_i<m/2$ berechne

$$r_1(x_1)\cdots r_1\left(x_{m/2}\right),r_2\left(x_{m/2+1}\right)\ldots,r_2(x_m)$$

(der Grad wurde halbiert). Anzahl der Divisionsschritte:



Teile $\log_2 m$

COROLLARY. *(*[**Borodin, Moenck1974**, Corollary 2, Section 6]*) Die Auswertung eines Polynoms vom Grad $N-1$ an N Stellen, $N=2^k$ geht in*

$$11\cdot N\left(\log_2 N\right)^2+O\left(N\log_2 N\right)$$

arithm. Schritten. Davon sind $10\cdot N\log_2 N$ Multiplikationen. Die wenigen Divisionen, die anfallen (in $FFT_{2^k}^{-1}$ und Polynom-Division) sollen rekursiv nach Sievking berechnet werden.

Knuth bemerkt in [**Knuth2**, S 486], dass der Algorithmus des obigen Korollars rein theoretisch ist, sofern N nicht sehr groß ist. In [**Montgomery, Silverman**] wird mit dem Satz geschlossen, dass dieser Algorithmus vermutlich nicht von praktischer Bedeutung ist. Für die Zerlegung von $n=p\cdot q$ ist das relevante N etwa $N=2^{35}$ und der Algorithmus höchst effizient.

PROPOSITION 3.6.9. *(Nussbaumer, 1982) siehe auch* [**Knuth2**, 4.6.4, Aufgabe 59]
Sei R komm. Ring mit 2^{-1}. Dann geht die zirkuläre Konvolution der Ordnung 2^k in $O\left(2^k k\right)$ Mult. und $O\left(2^k k\log_2 k\right)$ Add./Subtr. und Divisionen durch 2.

KAPITEL 4

Elliptische Kurven

(Prof. Schnorr empfiehlt für dieses Kapitel das Buch [**Blake et al.**]*)*

Sei $\mathbb{F} \subset K$ Körper (Koeffizientenkörper$\subset$Oberkörper), $A, B \in \mathbb{F}$. Die Gleichung (**kurze Weierstrass Form**)

$$E: \quad y^2 z = x^3 + Axz^2 + Bz^3 \quad (1)$$

in homogenen Koordinaten in Variablen x, y, z. Die Lösungsmenge in K ist eine elliptische Kurve $E(K)$.

Die Diskriminante ist $\Delta = -16\left(4A^3 + 27B^2\right)$. E ist singulär oder $\Delta = 0$.

Äquivalent

DEFINITION. **(Äquivalent)** Zwei Lösungen (x,y,z), $(x',y',z') \in K^3$ von (1) sind **äquivalent**, wenn

$$\exists c \in \mathbb{F} : c(x,y,z) = \left(x',y',z'\right).$$

Wir betrachten nur Lösungen $(x,y,z) \neq (0,0,0)$. $(x:y:z)$ ist die Äquivalenzklasse von (x,y,z). Die elliptische Kurve mit Koeffizienten $E_{A,B}(K)$ besteht aus den Punkten $P = (x:y:z)$ mit $x,y,z \in K$, die (1) erfüllen.

REMARK. Es gibt genau ein $(x:y:z) \in E_{A,B}(K)$ mit $z = 0$, nämlich

$$(0:y:0) \underset{def.}{=} 0,$$

„**der unendliche ferne Punkt**".

Unendlich ferner Punkt

(x,y,z) mit $z \neq 0$ ist von der Form $(x',y',1) = (x,y,z)/z$, **normierte Koordinaten** x',y', **homogene Koordinaten** x,y,z.

Die Punkte $(x:y:1) \in E_{A,B}(K)$ sind die Lösungen der Gleichung

$$y^2 = x^3 + Ax + B.$$

Die allgemeine Weierstrass-Form

$$E: \quad y^2 z + a_1 xyz + a_3 yz^2 = x^3 + a_2 x^2 z + a_4 xz^2 + a_6 z^3$$

interessiert im Folgenden eigentlich nicht, da sie durch allgemeine Variablentransformation in die spezielle Form gebracht werden kann.

4.1. Das Additionsgesetz zu $E_{a,b}(K)$

Betrachte die Gerade

$$G: \quad y - y_2 = \frac{y_1 - y_2}{x_1 - x_2}(x - x_2)$$

durch die Punkte $(x_1, y_1), (x_2, y_2)$ von $E_{a,b}(K)$.

LEMMA 4.1.1. *Jede Gerade G durch zwei Punkte von $E_{a,b}(K)$ scheidet die Kurve in einem dritten Punkt.*

BEWEIS. Der Schnitt von G und $E_{a,b}(K)$ durch zwei Punkte von $E_{a,b}(K)$ ist ein Polynom dritten Grades in x. Diese hat mit Vielfachheiten gerechnet genau drei Nullstellen im Zerfällungskörper. Wenn zwei dieser Nullstellen x_1, x_2 in K liegen, ist auch die dritte x_3 in K, denn nach Vieta ist die Summe $x_1 + x_2 + x_3$ Koeffizient des Polynoms. □

Additionsgesetz

DEFINITION. **(Additionsgesetz)** Für je 3 Punkte auf einer Geraden $P, Q, R \in E_{A,B}(K)$ gilt:

$$P + Q + R = 0, \quad \text{d.h. } P + Q = -R.$$

CLAIM. Sei $P, Q \in E_{A,B}(K), P \neq Q$. Dann schneidet die Gerade durch P, Q die Kurve $E_{A,B}(K)$ in einem „dritten" Punkt, nämlich $-R$. Im Fall $P = -Q$ ist die Gerade durch P und Q eine Parallele zur Y-Achse. In diesem Fall ist R gleich dem unendlich fernen Punkt O.

Daraus ist auch ersichtlich, warum eine elliptische Kurve keinen singulären Punkt (doppelte Nullstelle) besitzen darf ($\Leftrightarrow 4a^3 + 27b^2 = 0$): In diesem Fall hat die elliptische Kurve drei Nullstellen (eine doppelte P_1 und P_2 und eine von diesen verschiedene Q). Nach Definition müsste dann $P_1 + Q = P_2$ sein – da aber $P_1 = P_2$ ist, wäre Q dann das neutrale Elemente und somit gleich O – Widerspruch.

REMARK. Auch kann man leicht einsehen, dass eine EK über $\mathbb{F}_q$ mit $q = p^r$ maximal $2q + 1$-viele Punkte besitzen kann: den Punkt O und $2q$-viele Punkte, weil es zu jedem möglichen x-Wert (und das sind maximal q-viele) maximal 2 mögliche y-Werte gibt.

4.2. Gruppenstruktur von $E_{a,b}(K)$

Die Gruppenstruktur von $E_{a,b}(K)$ ist durch folgende Punkte gegeben:

(1) Neutrales Element: $O = (0:1:0)$ „unendlich ferner Punkt".
(2) Addition: $\forall P, Q, R \in E_{a,b}(K)$ auf einer Geraden gilt:

$$\begin{aligned} P + Q + R &= O \\ P + Q &= -R. \end{aligned}$$

(3) Inverses Element: $P = (x, y) \Rightarrow -P = (x, -y)$.

Genauer ist $E_{a,b}(K)$ sogar kommutativ:

PROPOSITION 4.2.1. *Die Gruppe $E_{a,b}(K)$ ist eine abelsche additive Gruppe.*

BEWEIS. Die Kommutativität der Addition folgt aus der Gruppeneigenschaft 2, in der es auf die Reihenfolge der Punkte P,Q,R auf der Geraden nicht ankommt. Nach dieser Eigenschaft ist die Addition entlang einer Geraden auch assoziativ, weil die Identität

$$P+Q+R=O=P'+Q'+R'$$

auf beiden Seiten kommutativ und assoziativ ist, muss die Addition assoziativ sein, so dass gilt

$$P+Q-P'=Q+R'-R$$

mit der Assoziativität auf beiden Seiten. □

4.3. Addition von $P=(x_1,y_1)$ und $Q=(x_2,y_2)$

Wir unterscheiden folgende Fälle:

(1) $\underline{P=O}$:
In diesem Fall ist $-P=O$ und für jedes beliebige Element $Q\in E_{a,b}$ gilt $P+Q=Q$.

(2) $\underline{P=-Q \quad\Leftrightarrow\quad (x_1,y_1)=(x_2,-y_2)}$:
Wenn P, der an der X-Achse gespiegelte Punkt Q ist, dann gilt $P+Q=O$.

(3) $\underline{P\neq Q \text{ mit } x_1\neq x_2}$:
Die Gerade g durch P,Q definiert durch

$$g(x)=\alpha x+\beta.$$

Nach Einsetzen der Punkte P,Q ergibt sich

$$y_1=\alpha x_1+\beta \quad \text{und} \quad y_2=\alpha x_2+\beta$$

und damit

$$g(x)=\frac{y_2-y_1}{x_2-x_1}(x-x_1)+y_1=\alpha(x-x_1)+y_1,$$

die Steigung der Gerade ist also $\alpha=\frac{y_2-y_1}{x_2-x_1}$ und $\beta=y_1-\alpha x_1$.
Der Schnitt $g(x)\bigcap E_{A,B}(K)$ ist somit

$$(\alpha x+\beta)^2=x^3+ax+b.$$

Es gibt also einen Schnittpunkt für jede Wurzel der Gleichung

$$x^3-(\alpha x+\beta)^2+ax+b.$$

Wie wir bereits wissen sind P,Q und somit auch $(x_1,\alpha x_1+\beta),(x_2,\alpha x_2+\beta)$ Schnittpunkte und ergo x_1,x_2 Wurzeln des obigen Ausdrucks. Weil in einem monischen Polynom die Summe der Wurzeln gleich dem negativen Koeffizienten des zweithöchsten Exponenten ist, schliessen wir, dass für die dritte Wurzel gilt

$$x_3=\alpha^2-x_1-x_2.$$

Das führt zu einem Ausdruck für x_3 und somit für beide Koordinaten von $P+Q=(x_3,-(\alpha x_3+\beta))$, oder mit x_1,x_2,y_1,y_2 ausgedrückt:

$$\begin{aligned} x_3 &= \left(\frac{y_2-y_1}{x_2-x_1}\right)^2-x_1-x_2 \\ y_3 &= -y_1+\left(\frac{y_2-y_1}{x_2-x_1}\right)(x_1-x_3). \end{aligned}$$

(4) $\underline{P = Q}$:
Es gilt $(x_1, y_1) = (x_2, y_2)$ und statt $\alpha = (y_1 - y_2) / (x_1 - x_2)$ setzen wir nun die Steigung α der Geraden auf die Steigung der EK in dem Punkt P (also die erste Ableitung, die Steigung der Tangente):

$$\left.\begin{array}{rcl} y^2 & = & x^3 + ax + b \\ \frac{\partial x}{\partial y} & = & \frac{3x^2+a}{2y} \end{array}\right\} \Rightarrow \alpha = \frac{3x_1^2 + a}{2y_1}$$

also

$$g(x) = \frac{3x_1^2 + a}{2y_1} x + \beta$$

und nach Einsetzen von P

$$\begin{array}{rcl} y_1 & = & \frac{3x_1^2 + a}{2y_1} x_1 + \beta \\ \Leftrightarrow \quad \beta & = & y_1 - \frac{3x_1^2 + a}{2y_1} x_1, \end{array}$$

also

$$g(x) = \alpha x + y_1 - \alpha x_1 = \alpha (x - x_1) + y_1.$$

Damit ist

$$\begin{array}{rcl} x_3 & = & \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \\ y_3 & = & -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3). \end{array}$$

REMARK. Diese Addition ist nur im Fall $\text{char}(\mathbb{F}) \neq 2, 3$ möglich, im Fall $\text{char}(\mathbb{F}) = 2, 3$ muss eine andere Form der Weierstrass-Gleichung gewählt werden.

Laufzeit:

Operation auf EK	#Mult.	#Div.	#Add.
Addition	2	1	6
Punktverdopplung	4	1	6

REMARK. Nach diesen Betrachtungen wird die wichtigste Eigenschaft elliptischer Kurven deutlich: sie bilden alle abelsche Gruppen bzgl. der Gruppenoperation „+". Diese Gruppe ist jedoch nicht unbedingt zyklisch, dafür aber immer ein Produkt zweier zyklischer Gruppen. Sie ist also immer isomorph zu

$$\prod_{p|n} \mathbb{Z}/p^\alpha \times \mathbb{Z}/p^\beta$$

mit $\alpha \geq 1$ und $\beta \geq 0$.

Multiplikation auf EK's

DEFINITION. (**Multiplikation auf EK's**)
Sei E eine elliptische Kurve modulo n, $P \in E$. Für $m \in \mathbb{N}$ wird mP (in dem Fall, dass keine Addition scheitert) rekursiv definiert durch:

(1) $0P = O$.
(2) m ungerade: $mP = (m-1)P + P$.
(3) m gerade: $mP = \frac{m}{2}P + \frac{m}{2}P$.

In diesem Zusamenhang soll noch die Menge der elliptischen Kurven modulo p definiert werden:

(E_p): Elliptische Kurve mod p

(Elliptische Kurve mod p) Sei E eine elliptische Kurve, definiert durch $f(x) = x^3 + ax + b$. Dann ist E modulo p definiert durch

$$(E)_p : \quad y^2 = x^3 + (a)_p x + (b)_p$$

mit $(x)_p \equiv x \mod p$. Ist $P = (x,y) \neq O$ aus E, dann ist

$$(P)_p = \left((x)_p, (y)_p\right)$$

und damit

$$P \in E \quad \Rightarrow \quad (P)_p \in (E)_p .$$

Die Auswirkungen auf die Multiplikation auf EK's beschreibt folgendes Lemma:

LEMMA 4.3.1. *Sei $n \in \mathbb{N}$ und p Primteiler von n. Mit E elliptische Kurve modulo n und $P, Q \in E$ gilt dann*

(1) $(P)_p = O \Leftrightarrow P = O$.
(2) Falls $P+Q$ definiert ist und $\neq O$, dann ist $(P+Q)_p \neq O$.
(3) Falls $P+Q$ definiert ist, so ist $(P+Q)_p = (P)_p + (Q)_p$, wobei die erste Addition auf E und die zweite auf $(E)_p$ stattfindet.
(4) Falls mP definiert ist, so ist $(mP)_p = m(P)_p$.

PROPOSITION 4.3.2. *Sei $P \in E$ ein Punkt auf der elliptischen Kurve E modulo n, dann kann mP in Zeit $O\left(\log m \cdot \log^2 n\right)$ bestimmt werden.*

BEWEIS. Nach der obenstehenden Definition der Multiplikation wird der Algorithmus $\log m$-oft aufgerufen. Jeder Aufruf benötigt eine Zeit von $O\left(\log^2 n\right)$ (das ist die Zeit, die die Multiplikation bzw. Division mit EEA benötigt). □

4.4. Die Ordnung von $E_{a,b}(\mathbb{F}_q)$

Frobenius Spur

DEFINITION. Sei $\mathbb{F}_q$ endlicher Körper mit q Elementen und

$$\#E_{a,b}(\mathbb{F}_q) = q + 1 - t.$$

Dann heisst t die **Frobenius Spur**.

Über die Grösse der Frobenius Spur macht der folgende Satz von Hasse eine Aussage.

Satz von Hasse

PROPOSITION 4.4.1. ***(Satz von Hasse)***
Die Ordnung (Anzahl der Elemente) einer Elliptischen Kurve $E_{a,b}(\mathbb{F}_q)$ schwankt um $q+1$ um maximal $2\sqrt{p}$:

$$\#E_{a,b}(\mathbb{F}_q) = q + 1 - t \quad mit\ |t| \leq 2\sqrt{q}$$

Im Folgenden soll die mittlere Frobenius Spur $|t|$ für $a, b \in \mathbb{F}_q$ analysiert werden. Betrachte dazu die Zerlegung

$$\mathbb{F}_q = QR_q \cup QNR_q \cup \{0\}.$$

Für $f(x) = x^3 + ax + b$ gilt dann

$$\#E_{a,b}(\mathbb{F}_q) = 2 \cdot \#f^{-1}(QR_q) + \#f^{-1}(0) + 1,$$

wobei der Faktor 2 von $(x, \pm y)$ kommt. Setze nun

$$\mathbb{F}_q' = \mathbb{F}_q - f^{-1}(0) \quad \text{und} \quad q' := \#\mathbb{F}_q'.$$

Für zufällige $a, b \in_R \mathbb{F}_q$ gilt im Mittel $\#f^{-1}(0) = 1$ und $q' = q - 1$. Betrachten wir nun für $a, b \in_R \mathbb{F}_q$ die Zufallsvariablen

$$X_x = \begin{cases} 1 & , f(x) \in QR_q \\ 0 & , f(x) \in QNR_q \end{cases} \quad \text{fuer } x \in \mathbb{F}_q',$$

dann ergibt sich

$$\begin{aligned} E_{a,b}[X_x] &= \frac{1}{2} \\ Var[X_x] &=_{def} E_{a,b}\left[X_x - E_{a,b}[X_x]\right]^2 = \frac{1}{4}. \end{aligned}$$

Nach der **Tschebycheff-Ungleichung** gilt für reelwertige Zufallsvariablen X für $\varepsilon \geq 0$

$$Ws_X\left[|X - E[X]| \geq \varepsilon\right] \leq Var[x]/\varepsilon^2.$$

Die X_x mit $x \in \mathbb{F}_q'$ sind wegen $a, b \in_R \mathbb{F}_q$ paarweise statistisch unabhängig. Somit gilt

$$Var\left(\sum_{x \in \mathbb{F}_q'} X_x\right) = \sum_{x \in \mathbb{F}_q'} Var(X_x) = \frac{1}{4} q'.$$

Daraus folgt

$$Ws_{a,b}\left[\left|\sum_{x \in \mathbb{F}_q'} X_x - q'/2\right| \geq \sqrt{q'}\right] \leq \frac{1}{4} q'/q' = \frac{1}{4}$$

und wegen

$$\#f^{-1}(QR_q) = \sum_{x \in \mathbb{F}_q'} X_x$$

gilt nach Tschebycheff im Mittel für $a, b \in_R \mathbb{F}_q$

$$E_{a,b}\left(\left|\#f^{-1}(QR_q) - \frac{q'}{2}\right|\right) < \sqrt{q'}.$$

Der Satz von Hasse zeigt dann, dass im worst-case gilt

$$\left|\#f^{-1}(QR_q) - \frac{q'}{2}\right| \leq \sqrt{q}.$$

4.5. Skalare Multiplikation auf $E_{a,b}$

Die Exponentiation $P \mapsto P^m$ schreibt man für die additive Gruppe $E_{a,b}$ als skalare Multiplikation

$$P \quad \mapsto \quad [m]P.$$

H.W. Lenstra analysiert in [**Lenstra1987**, S. 649-673] die Anzahl der Isomorphieklassen von elliptischen Kurven $E_{a,b}(\mathbb{F}_q)$ gewichtet mit $\left(\#Aut\, E_{a,b}(\mathbb{F}_q)\right)^{-1}$.

Nach Deuring (1941) gilt

$$\#'\left\{E_{a,b} \mid a, b \in \mathbb{F}_q, \#E_{a,b} = q + 1 - t\right\}_{|\cong \mathbb{F}_q} = H\left(t^2 - 4q\right).$$

Kroneckerklassenzahl, Diskriminante

DEFINITION. **(Kroneckerklassenzahl, Diskriminante)** Dabei ist $H(\Delta)$ die *Kroneckerklassenzahl* der quadratischen Formen

$$ax^2+bxy+cy^2=[x,y]\left[\begin{array}{cc} q & b/2 \\ b/2 & c \end{array}\right]\left[\begin{array}{c} x \\ y \end{array}\right]$$

mit negativer *Diskriminante* $\Delta=b^2-4ac$.

PROPOSITION 4.5.1. *(H.W. Lenstra) Es gilt mit Ausnahmen als Laufzeitabschätzung*

$$\Omega\left(\frac{\sqrt{-\Delta}}{\log|\Delta|}\right)\leq H(\Delta)=O\left(\sqrt{-\Delta}\log|\Delta|\,(\log\log\Delta)^2\right).$$

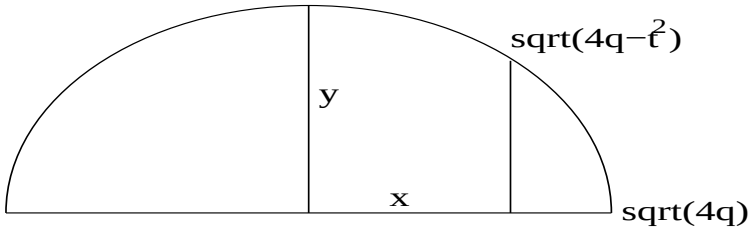
Angewandt auf unser Problem ergibt das:

PROPOSITION 4.5.2. *(H.W. Lenstra nach Deuring) Es gibt $c_1,c_2>0$, so dass für alle Primzahlen $q>3$ und für alle $t\in\mathbb{Z}$ mit $|t|\leq 2\sqrt{q}$ gilt*

$$\begin{aligned} c_1\sqrt{q}/\log q &\leq \#'\left\{(a,b)\in\mathbb{F}_q^2 : \#E_{a,b}(\mathbb{F}_q)=q+1-t\right\}_{|\cong\mathbb{F}_q} \\ &\leq c_2\sqrt{q}\log q\log\log q. \end{aligned}$$

Das Resultat nach Deuring bedeutet, dass $\#E_{a,b}$ etwa wie die y-Koordinate eines zufälligen Punktes in der 2-dimensionalen Kugel mit Radius $\sqrt{4q}$ verteilt ist:

ABBILDUNG 4.5.1. Verteilung von $\#E_{a,b}$



4.6. Primzahltest mit EK (Algorithmus von Goldwasser-Kilian)

Bevor wir zur Faktorisierung mit elliptischen Kurven kommen, soll noch ein Primzahltest auf Basis elliptischer Kurven vorgestellt werden. Seine genaue Form kann in [**Blömer**, Seite 60 (AGK)] nachgelesen werden.

PROPOSITION 4.6.1. *Sei $n\in N$, $ggT(6,n)=1$. Sei E eine EK und $m\in\mathbb{N}$. n ist eine Primzahl, falls gilt*

1) es existiert eine Primzahl q mit $q\mid m$ und $q>\left(n^{\frac{1}{4}}+1\right)^2$,
2) ein Punkt $P\in E$ existiert mit mP und $\frac{m}{q}P$ sind definiert und $mP=O$, $\frac{m}{q}\neq O$.

BEWEIS. Ist n nicht prim, so existiert ein Primteiler p von n mit $p<\sqrt{n}$ und $n=pq$. Nach dem Satz von Hasse wissen wir, dass gilt

$$\#(E)_p\leq p+1+2\sqrt{p}=(p+1)^2\leq\left(n^{1/4}+1\right)^2<q,$$

die Ordnung von $(E)_p$ ist also kleiner als q.

Liegt der Punkt $(P)_p$ auf $(E)_p$, dann gilt $m(P)_p = O$ (m ist ein Vielfaches der Gruppenordnung, Lemma 4.3.1) und $\frac{m}{q}(P)_p \neq O$.Also teilt q die Odnung von $(P)_p$, was ein Widerspruch zu der Annahme ist, dass q grösser ist als die Gruppenordnung. □

Genauso gilt die Umkehrung.

PROPOSITION 4.6.2. *Sei p prim und E EK modulo p mit $|E| = m$. Falls ein Primteiler q von m existiert mit $q > \left(p^{1/4}+1\right)^2$, dann existiert ein Punkt $P \in E$ mit $mP = O$ und $\frac{m}{q}P \neq O$.*

BEWEIS. Wie oben gilt $m \leq p + 2\sqrt{p} + 1 = \left(\sqrt{p}+1\right)^2 < q^2$. Es gilt also $q \mid m$ und $q^2 \nmid m$, also existiert ein Punkt P der Ordnung q auf E mit $mP = O$ und $\frac{m}{q}P \neq O$, denn sonst müsste gelten $q \mid \frac{m}{q}$ und somit $q^2 \mid m$. □

Korrektheit: In den Schleifen in Schritt 2.b) und 2.d) werden die Eigenschaften von Satz 4.6.1 überprüft. Diese stellen ein hinreichendes Kriterium für die Primheit von n dar, allerdings gelten diese Eigenschaften nur für ein q, das prim ist. Aus diesem Grund wird in Schritt 3. nocheinmal der Algorithmus mit q aufgerufen.

Jetzt kann es natürlich sein, dass die Schleife im 3. Schritt unendlich lang läuft, da, falls n zusammengesetzt ist, kein q existieren muss, so dass q prim und $m = 2q$ ist. In diesem Fall „rettet" uns der in Schritt 1.b) parallel aufgerufenen MR-Algorithmus, der eine erwartete polynomielle Laufzeit besitzt.

Laufzeit: Der Algorithmus hat eine erwartet polynomielle Laufzeit. Ein Beweis hierzu befindet sich in [**Blömer**, Satz 9.5]. Im Gegensatz zum MR-Algorithmus ist die Ausgabe von GK immer korrekt, dafür seine Ausführung nicht so effizient wie bei MR. Allerdings liegt der Laufzeitanalyse eine Vermutung zu Grunde, deren Korrektheit noch nicht bewiesen ist:

CLAIM 4.6.3. Es existiert ein $c \in \mathbb{N}$, so dass für $m \in \mathbb{N}$ groß genug im Intervall $[m - 2\sqrt{m} + 1, m - 2\sqrt{m} + 1]$ mindestens $\sqrt{m}/\log^c(m)$ Zahlen von der Form $2q$ mit q prim sind.

4.6.1. EK Faktorisierung von $n = pq$. Der Faktorisierungsalgorithmus mit Elliptischen Kurven wird keine vollständige Faktorisierung einer Zahl n finden, sondern nur einen Teiler d von n.

Wegen des CRT gilt $\mathbb{Z}_n \cong \mathbb{Z}_p \times \mathbb{Z}_q$.

Dies funktioniert am besten für $p \ll q$.

4.6.2. Arithmetik auf $\mathbb{Z}_n$.

PROPOSITION 4.6.4. *Es gibt absolute Konstanten $0 < \alpha, \beta > 1$, so dass für alle Primzahlen $q \geq 3$ gilt:*
Für einen α-Anteil der Paare $(a,b) \in \mathbb{F}_q^2$ mit $4a^3 + 27b^2 \neq 0$:

$$\#E_{a,b} \in_{AR} \left(q - \sqrt{q}, q + \sqrt{q}\right)$$

(AR bedeutet in diesem Zusammenhang „approximativ gleichverteilt). Die $\|\ \|_1$-Distanz zur Gleichverteilung ist $\leq \beta$ (die Differnzen der Werte werden aufsummiert).

Algorithm 25 Goldwasser-Kilian - AGK($n, ORIGINAL_n$)

EINGABE: $n \in \mathbb{N}$, $\text{ggT}(6,n) = 1$ und $ORIGINAL_n \in \{true, false\}$
AUSGABE: n „prim": *true* oder *false.*

(1) **if** $ORIGINAL_n$ **then**
  (a) **if** $n \leq 2^{30}$ **then**
    **** Bei kleinen Zahlen kann ein trivialer Test erfolgen ****
      (i) Teste mittels Division durch alle Zahlen $\leq \sqrt{n}$, ob n prim ist.
      (ii) **if** n zusammengesetzt **then**
        (A) **return**(*false*).
      (iii) **else**
        (A) **return**(*true*).
  (b) **else**
    **** Bei großen Zahlen parallel den MR-Test anwenden ****
      (i) Starte parallel MR-Test mit n und immer neuem $a \in_R \mathbb{Z}_n$.
      (ii) **if** MR findet $a \in_R \mathbb{Z}_n$ als Beweis, dass n nicht prim **then**
        (A) **return**(*false*).
(2) **do**
  (a) **** Versuchen, eine EK E zu konstruieren ****
    Suche $a, b \in_R \mathbb{Z}_n$ mit $4a^3 + 27b^2 \not\equiv 0 \mod n$.
  (b) **** Ist die EK korrekt?* *******
    do
      (i) **if** $\text{ggT}\left(4a^3 + 27b^2, n\right) \neq 1$ **then**
        (A) **return**(*false*).
      (ii) Berechne $m = |E|$, mit E def. durch $f(x) = x^3 + ax + b$ (Schoof, Satz 4.7.5).
      (iii) **if** Berechnung scheitert $\vee$ $m > n + 2\sqrt{n} + 1$ **then**
        (A) **return**(*false*).
  (c) **while** $\neg(m = 2q$ mit q prim)
  (d) **** Stimmen die Gruppeneigenschaften in E?* *******
    do
      (i) Suche $x \in \mathbb{Z}_n$ mit $\left(\frac{x^3+ax+b}{n}\right) = 0$ oder 1.
      (ii) **if** $\left(\frac{x^3+ax+b}{n}\right) = 0 \wedge x^3 + ax + b \not\equiv 0 \mod n$ **then**
        (A) **return**(*false*).
      (iii) **else**
        (A) Berechne $y \in \mathbb{Z}_{n_i}$ mit $y^2 = x^3 + ax + b$.
      (iv) Setze $P = (x, y)$ und berechne mP und $2P$.
      (v) **if** Addition nicht möglich **then**
        (A) **return**(*false*).
      (vi) **if** $mP \neq O$ **then**
        (A) return(*false*).
  (e) **while** $(2P = O)$
(3) **** Ist q mit $n = kq$ überhaupt prim? **** **while**($\neg$AGK(q, $false$)).
(4) **return**(*true*).

Algorithm 26 Zirkuläres Polynomprodukt EK-Faktorisierung von $n = pq$

EINGABE: $n = p \cdot q$
AUSGABE: $d \mid n$ mit $d \notin \{1, n\}$

(1) Wähle zufällig $P = (x,y) \in_R \mathbb{Z}_n^2$ und $A \in_R \mathbb{Z}_n$ und bestimme B, so dass B auf der Kurve liegt: $B := y^2 - x^3 - Ax$. Also $P = (x,y) \in E_{A,B}(\mathbb{Z}_n)$.
(2) Wähle Schranke γ.
(3) $gl(\gamma) := \text{kgV}(2,3,\ldots,\gamma) = \prod_{p_i^{e_i} \leq \gamma < p_i^{e_i+1}} p_i^{e_i}$.
(4) Berechne $gl(\gamma) \cdot P = (\overline{x}, \overline{y}) \in \mathbb{Z}_n^2$.
(5) Teste, ob $\text{ggT}(\overline{x}, n) \notin \{1, n\}$
$\#E_{A,B}(\mathbb{Z}_p)$ γ-glatt $\Rightarrow \overline{x} \equiv 0 \mod p$
$p \mid \text{ggT}(\overline{x}, n)$
$\#E_{A,B}(\mathbb{Z}_p), \#E_{A,B}(\mathbb{Z}_q)$ statistisch unabhängig

Seien X, Y reelwertige ZV, dann gilt

$$\|X - Y\|_1 = \sum_a |Ws[X = a] - Ws[Y = a]|.$$

Algorithm 27 Elliptische Kurven Methode (EKM) zu $n = p \cdot q$

EINGABE: $n \in \mathbb{N}$ mit $n = p \cdot q$
AUSGABE: $d \mid n$ mit $d \notin \{1, n\}$

(1) **do**
(a) **** Der Startwert von γ ist willkürlich ****
Wähle $(a,x,y) \in_R \mathbb{Z}_n^3$, $b := y^2 - x^3 - ax$ und $P := (x,y) \in E_{a,b}(\mathbb{Z}_n)$. $\gamma := 10$
(b) **** Die Addition findet auf der EK statt ****
$(\overline{x}, \overline{y}) := gl(\gamma) \cdot P$
(2) **** Im Fall $ggT(\overline{x}, n) = n$ muss man eine andere EK wählen ****
while $(\text{ggT}(\overline{x}, n) = 1)$

Der Witz bei der EKM ist, dass Additionen auf der EK stattfinden - und wenn diese scheitern, so wurde ein Teiler von n gefunden:

LEMMA 4.6.5. *Ist E eine EK modulo n und sind P, Q Punkte auf E, so dass $P + Q$ nicht definiert ist, so findet die EKM einen echten Teiler von n.*

BEWEIS. Seien also $P = (x_1, y_1)$ und $Q = (x_2, y_2)$, mit $P \neq Q$ und $x_1 \neq x_2$, dann ist nach Abschnitt 4.3 auf Seite 69

$$P + Q = (x_3, y_3) = \left(\left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2, -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1} \right) (x_1 - x_3) \right).$$

Es muss also das Inverse von $x_2 - x_1$ modulo n berechnet werden. Da die Summe nach Voraussetzung nicht definiert ist, ist $x_2 - x_1 \equiv 0 \mod n$ und das Inverse kann nicht berechnet werden, da gilt $\text{ggT}(x_2 - x_1, n) \neq 1$ (Satz von Bezout). Wegen $x_1 \neq x_2$ ist $x_2 - x_1 \neq 0$ und somit $\text{ggT}(x_2 - x_1, n) \notin \{1, n\}$, also ist der ggT ein echter Teiler von n.

Ist nun $P = Q$, dann ist nach Abschnitt 4.3 auf Seite 69

$$P + P = \left(\left(\frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1, -y_1 + \left(\frac{3x_1^2 + a}{2y_1} \right) (x_1 - x_3) \right).$$

Wieder ist die Addition nicht definiert, wenn das Inverse zu y_1 nicht berechnet werden kann. dann gilt aber nach dem Satz von Bezout $\text{ggT}(y_1, n) \neq 1$. Da aber auch gleichzeitig gilt $y_1 \neq n$ folgt, dass $\text{ggT}(y_1, n) \notin \{1, n\}$ und somit ein nicht-trivialer Teiler von n ist. □

4.6.3. Arithmetik auf $E_{a,b}(\mathbb{Z}_n)$. Sei $R := P + Q$, $(x_3, y_3) := (x_1, y_1) + (x_2, y_2)$ (die Addition findet auf der elliptischen Kurve statt). Wende nun die Formel für $E_{a,b}(K)$ für einen Körper K an – Problem: unter Umständen bereitet die Division Schwierigkeiten:

$$\lambda = \frac{y_1 - y_2}{x_1 - x_2} \quad , \quad \lambda = \frac{3x_1^2 + a}{2y_1}.$$

Ersetze dann p durch n.

Sonderfall 1: $x_1 \equiv x_2 \mod p, x_1 \not\equiv x_2 \mod q$ (oder umgekehrt)
Zerlege $n = p \cdot q$ mit $p := \text{ggT}(x_1 - x_2, n)$ und berechne (x_3, y_3) modulo p und q. Setze nun mittels CRT zusammen ($\mathbb{Z}_n \cong \mathbb{Z}_{\text{,}} \times \mathbb{Z}_q$).

Sonderfall 2: $(x_1, y_1) = (x_2, y_2)$, $\text{ggT}(2y_1, n) \neq 1$
Man zerlege $n = p \cdot q$ mit $p := \text{ggT}(x_1 - x_2, n)$.

PROPOSITION 4.6.6. *Sei $n = p \cdot q$, $p \neq q$, p, q prim.*
Dann gilt

$$E_{a,b}(\mathbb{Z}_n) \cong E_{a,b}(\mathbb{Z}_p) \times E_{a,b}(\mathbb{Z}_q)$$

ist abelsche Gruppe mit Polynomialzeitarithmetik.

Fall 1: $n = p_1 \cdot p_2 \cdot \ldots \cdot p_r$ paarweise verschieden

$$E_{a,b}(\mathbb{Z}_n) \cong E_{a,b}(\mathbb{Z}_{p_1}) \times \ldots \times E_{a,b}(\mathbb{Z}_{p_r}).$$

Fall 2: $n = p^2 \cdot q$

$$E_{a,b}(\mathbb{Z}_{p^2}) \ncong E_{a,b}(\mathbb{Z}_p) \times E_{a,b}(\mathbb{Z}_p).$$

Es steht also kein CRT mehr zur Verfügung, die Methode sollte aber trotzdem funktionieren...

Erfolgswahrscheinlichkeit: $p \mid \overline{x} \quad \Leftrightarrow \quad 2 \cdot (\overline{x}, \overline{y}) = O \mod p$

$$\#E_{a,b}(\mathbb{Z}_p) \text{ ist } \gamma - \text{glatt} \quad \Rightarrow \quad gl(\gamma) \cdot P \equiv O \mod p$$

(die Division wird durch ggT-Bildung durchgeführt und ist damit nicht aufwändiger als die Multiplikation)
Die Ergebnisse „$\#E_{a,b}(\mathbb{Z}_p)$ ist γ-glatt" und „$\#E_{a,b}(\mathbb{Z}_q)$ ist γ-glatt" sind stat. unabhängig, weil $(a, b) \in_R \mathbb{Z}_n^2$.

Sei $p < q$.
Erfolgswahrscheinlichkeit: $Ws_{a,b}\left[\#E_{a,b}(\mathbb{Z}_p)\ \gamma - \text{glatt}\right]$

Laufzeit: $p \mapsto gl(\gamma) \cdot P$ binäre Exponentiation mit $\log_2 \left(\underbrace{gl(\gamma)}_{\leq \gamma^{\pi(\gamma)} \leq e^{\gamma + o(\ln \gamma)^{-1}}} \right) = \log_2 e \cdot \gamma$

Additionen auf $E_{a,b}(\mathbb{Z}_n)$.
Kosten $O(\gamma)$ arithm. Operationen modulo n.

4.6.4. Dickman's Funktion.

$\Psi(x,y)$

DEFINITION. $\psi(x,y) =_{def.} \#\{z \in [1,x] : \text{ohne Primfaktoren} > y\}$ $x,y \in \mathbb{N}$. Dies entspricht der Definition in [**Blömer**, Definition 10.3]

$$\psi(x,y) = \#\{z \in [1,x] \mid zy - \text{glatt}\}.$$

PROPOSITION 4.6.7. ***(Canfield, Erdös, Pomerance)***
Für $\varepsilon > 0$, $u(x) : \mathbb{R} \to \mathbb{R}^+$ *mit* $u(x) \leq \log^{1-\varepsilon}(x)$ *für* $x \to \infty$. *Dann gilt:*

$$\psi\left(x, x^{1/u}\right) = x \cdot u^{-u+o(1)} \quad \textit{fuer } x \to \infty.$$

REMARK. Eine Funktion $g(x)$ ist $o(1)$, falls $g(x) \to 0$ für $x \to \infty$. Der Exponent in $u^{-u+o(1)}$ ist also bis auf einen Fehlerterm $-u$, der allerdings für $x \to \infty$ gegen 0 geht.
Die Bedingung $u(x) \leq \log^{1-\varepsilon}(x)$ bedeutet für das zweite Argument in ψ: $x^{1/u} \geq e^{\log^{\varepsilon}(x)} \geq \log^{c}(x)$ für jedes $c > 0$.

EXAMPLE. $U = 2$: $\psi\left(x, \sqrt{x} = x \cdot \frac{1}{4^{1+o(n)}}\right)$.

$L(n)$

DEFINITION. $L(n) =_{def.} e^{\sqrt{\log n \cdot \log\log n}}$ und $L_n(\beta) =_{def.} L(n)^{\beta}$

Außerdem wird eine neue Glattheitseigenschaft eingeführt. Zur Erinnerung:

$$n\,\gamma - \text{glatt} \quad \Leftrightarrow \quad n \mid \text{kgV}(2,\ldots,\gamma).$$

γ-glatt*

DEFINITION. **($\gamma - \text{glatt}^*$)**

$$n\,\gamma - \text{glatt}^* \quad \Leftrightarrow \quad n \text{ist frei von Primfaktoren} > \gamma.$$

COROLLARY 4.6.8. *Für* $n \to \infty$, $\beta > 0$ *gilt*

$$\psi(n, L_n(\beta)) = n \cdot L_n\left(-\frac{1}{2\beta} + o(1)\right).$$

Sei p der kleinste Primfaktor von n, $p \leq B \leq \sqrt{n}$. Für $z \in_R [1,B]$:

$$Ws_z\left[zL_B(z) - glatt^*\right] = L_B\left(-\frac{1}{2\beta} + o(1)\right).$$

Minimiere: $L_B(\beta) \cdot L_B\left(\frac{1}{2\beta} + o(1)\right)$ ((Erfolgswahrscheinlichkeit)$^{-1}$)
#arithm. Schritte von EKM mit $\gamma = L_B(\beta) = L_B\left(\beta + \frac{1}{2\beta} + o(1)\right)$ minimieren!
Ergebnis: $\beta = \frac{1}{\sqrt{2}}$.

4.6.5. EKM mit EK-Wechsel. Wie beim $p-1$-Algorithmus nach Blömer bereits angedeutet, kann der Einsatz dieses Algorithmus zu Problemen führen, wenn die Gruppenstruktur von $\mathbb{Z}_n$ ungünstig ist. Dieses Problem kann man mittels elliptischer Kurven umgehen, da man hier viele Gruppen modulo n zur Verfügung hat, die analysiert werden können.

Blömer schlägt in [**Blömer**, Kapitel 10] Algorithmus 28 vor.

Der Witz bei diesem (und den folgenden Algorithmen) ist, dass nach Lemma 4.6.5 auf Seite 76 eine fehlgeschlagene Addition auf der EK zu einem echten Teiler von n führt. Anders als beim $p-1$-Algorithmus, wo wir uns immer nur in der Gruppe $\mathbb{Z}_n$ bewegen,

Algorithm 28 EKM nach Blömer

EINGABE: $n \in \mathbb{N}$ keine Primzahlpotenz.
AUSGABE: $d \in \mathbb{N}$ mit $d \neq 1, n$ und $d \mid n$.

(1) Wähle Schranken $B_1, B_2 \in \mathbb{N}$.
(2) Berechne ($\rightarrow$Sieb des Erathostenes) alle Primzahlen $p_1, \ldots, p_l$ kleiner als B_1.
(3) **** Berechne die Faktoren des Produkts, mit dem P multipliziert wird ****
for $i = 1, \ldots, l$ **do**
(a) Berechne e_i mit $p_i^{e_i} \leq B_2 + 2\sqrt{B_2} + 1 < p_i^{e_i+1}$.
(4) **do**
(a) **do**
**** Wähle eine zufällige EK E modulo n und $P(x,y) \in_R E$ ****
(i) Wähle $(a,x,y) \in_R \mathbb{Z}_n^3$.
(ii) Setze $b := y^2 - x^3 - ax$.
(iii) Berechne $d := \mathrm{ggT}\left(4a^3 + 27b^2, n\right)$.
(b) **while** $d = n$.
(c) **** Es wurde bereits ein nicht-trivialer Teiler von n gefunden ****
if $d \neq 1, n$ **then**
(i) **return**(d).
(d) **** Diese Anweisung wird in jedem Fall ausgeführt ****
else if $d = 1$ **then**
(i) $P := (x,y)$.
(e) **** Mit der Addition auf EK berechne: $\prod_{i=1}^{l} p_i^{e_i} P$ ****
for $i = 1, \ldots, l$ **do**
(i) **for** $j = 1, \ldots, e_i$ **do**
(A) $P := p_i P$.
(5) **while** alle Addition in (e) waren möglich.
(6) Setze $d = \mathrm{ggT}(m,n)$ als den ggT, bei dem die Addition das erste Mal scheiterte.
(7) **return**(d).

können wir und hier beliebige Gruppen erzeugen, indem wir einfach andere Parameter für die EK wählen (hier: a,x,y).

In Schritt 4.a) wird eine zufällige elliptische Kurve modulo n und ein zufälliger Punkt P auf dieser konstruiert. Um die Berechnung von Quadratwurzel zu vermeiden, wird wie im Algorithmus von Goldwasser-Kilian statt (a,b,x) zufällig (a,x,y) gewählt und b dann berechnet.
Die Berechnung von $\prod_{i=1}^{l} p_i^{e_i} P$ kann wie bereits gesehen (Gruppenoperation auf elliptischen Kurven) stark von der Reihenfolge der Rechenschritte abhängen. Aus diesem Grund wird in diesem Algorithmus die Berechnung folgendermassen durchgeführt:

$$\prod_{i=1}^{l} p_i^{e_i} P \quad = \quad p_l \cdots p_2 \left(p_1 \cdots p_1 \left(p_1 P\right)\right).$$

Für die Multiplikation werden dabei die Regeln der Definition der Multiplikation auf EK auf Seite 70 angewandt – eine gescheiterte Addition liefert dabei einen echten Teiler von n! In diesem Zusammenhang sei noch erwähnt, dass eine geschickte Wahl der Schranken B_1 und B_2 zu den schwierigsten Teilen des Algorithmus zählen. Der Algorithmus findet in folgenden Fällen einen echten Teiler von n:

PROPOSITION. *Sei $n \in \mathbb{N}$. EKM findet einen echten Teiler von n falls Primteiler p,q mit $p \neq q$ von n existieren und eine EK E modulo n und ein Punkt $P \in E$ derart gewählt wurden dass folgende Bedingungen erfüllt sind:*

(1) $p \leq B_2$.
(2) Die Zahl $m_1 = \left|(E)_p\right|$ hat nur Primteiler kleiner als B_1.
(3) $m_2 = \left|(E)_q\right|$ ist nicht teilbar durch den grössten Primteiler der Ordnung von $(P)_p$ auf $(E)_p$.

BEWEIS. (Zur Definition von $(E)_p$ siehe Seite 71)
Sei w die Ordnung von $(P)_p$ auf $(E)_p$. Wegen (1) und Hasse (Seite 71) gilt dann

$$m_1 = \left|(E)_p\right| \leq B_2 + 2\sqrt{B_2} + 1.$$

Nach (2) folgt dann, dass gilt

$$m_1 = \prod_{i=1}^{l} p_i^{f_i} \quad \text{mit } 0 \leq f_i \leq e_i.$$

Damit gilt auch für w, dass $w = \prod_{i=1}^{l} p_i^{h_i}$ ist mit $0 \leq h_i \leq e_i$.

Sei p_c der größte Primteiler von w und h_c der Exponent, mit dem p_c w teilt und

$$k := \prod_{p_j < p_c} p_j^{e_j} p_c^{h_c - 1}.$$

Dann ist $k \not\equiv 0 \mod w$, aber $kp_c \equiv 0 \mod w$ (da $k = w/p_c$ ist).

Angenommen, mit diesem E und P können alle Additionen in Algorithmus 28 durchgeführt werden. Dann wären kP und $p_c kP_p$ erfolgreich berechnet worden. Da aber $k \not\equiv 0 \mod w$ ist und $kp_c \equiv 0 \mod w$, gilt

$$(kP)_p \neq O \quad \text{und} \quad (p_c(kP))_p = O.$$

Dann gilt aber nach dem Lemma bzgl. der Multiplikation auf $(E)_p$ auf Seite 71

$$kP \neq O \quad \text{und} \quad p_c(kP) = O.$$

Damit kann mit dem gleichen Lemma gefolgert werden, dass gilt

$$(kP)_q \neq O \quad \text{und} \quad (p_c(kP))_q = O.$$

Dann muss aber p_c die Ordnung von $(P)_q$ auf $(E)_q$ teilen, was aber im Widerspruch steht zu der Voraussetzung, dass p_c nicht $m_2 = \left|(E)_q\right|$ teilt.

Das bedeutet, dass bei dieser Wahl von E und P eine Addition in Schritt 4.e) nicht möglich ist und damit ein echter Teiler von n gefunden wird. □

Mittels einer Analyse der EKM nach Blömer möchte man nun gerne wissen, wie man B_1 und B_2 am geschicktesten wählt, so dass die erwartete Laufzeit möglichst gering ist: Einerseits möchte man, dass B_1 und B_2 möglichst groß sind, damit man in Schritt (4) des Algorithmus schnell zu einer EK E und einem Punkt $P \in E$ kommt, der die Eigenschaften aus dem obigen Satz erfüllt – andererseits sollen B_1 und B_2 möglichst klein gewählt werden, damit die Laufzeit (Schritte (2),(3), (4.e)) gering bleibt. Die Analyse wird in [**Blömer**, Abschnitt 10.2, Seite 68] näher ausgeführt. Dort wird zuerst die Wahrscheinlichkeit berechnet, dass eine Zahl γ-glatt ist für ein gegebenes γ (bei $\gamma = n^\alpha$ ist eine Zahl $\leq n^\alpha$ mit

Wahrscheinlichkeit $\approx L_n\left(-\frac{\alpha}{2\beta}\right) L_n(\beta)$-glatt). Dann wird analysiert, wie groß die Wahrscheinlichkeit ist, dass die EKM bei zufälliger Wahl eines Tripels $(a,x,y) \in \mathbb{Z}_n^3$ einen Teiler von n findet:

Wenn n zusammengesetzt ist und einen Primteiler $p \leq B_1$ besitzt und u der Anteil der B_1-glatten Zahlen im Intervall $\left[p-2\sqrt{p}+1, p+2\sqrt{p}+1\right]$ ist, dann existiert eine Konstante $c_0 > 0$, so dass für $(a,x,y) \in_R \mathbb{Z}_n^3$ die Wahrscheinlichkeit, dass die EKM mit diesem Tripel einen Teiler von n findet $> \frac{c_0 u}{\log p}$ ist. Damit findet die EKM unter der Voraussetzung, dass der Anteil der B_1-glatten Zahlen im Intervall $\left[p-2\sqrt{p}+1, p+2\sqrt{p}+1\right]$ gleich $L_p(-1/2\beta)$ ist, einen echten Teiler von n in erwarteter Zeit

$$e^{(1+o(1))\sqrt{\log(n)\log\log(n)}} = L_n(1+o(1)).$$

Dabei begnügt sie sich mit polynomiellem Speicherbedarf und kann derart angepasst werden, dass ihre Laufzeit von der Grösse des kleinsten Primteilers abhängt (was bei den meisten Krypto-Verfahren nicht sinnvoll ist, da hier vor allem große Primteiler verwendet werden).

In der Vorlesung wurde der Algorithmus wie folgt behandelt (Finde zu n Primfaktor $p \leq B$):

Algorithm 29 EKM mit EK-Wechsel (Schnorr)

EINGABE: $n \in N$, $\gamma \in \mathbb{N}$
AUSGABE: $d \mid n$ mit $d \notin \{1,n\}$

(1) Wähle $(a,x,y) \in_R \mathbb{Z}_n^3$, $n := y^2 - x^2 - ax$, $P := (x,y) \in E_{a,b}(\mathbb{Z}_n)$ und $\gamma := 1000$.
(2) $gl(\gamma) := kgV(2,3,\ldots\gamma) = e^{\gamma + o\left(\frac{1}{\log\gamma}\right)}$
(3) $(\overline{x},\overline{y}) := [gl(\gamma)] \cdot P$
(4) **if** bei der Berechnung von $(\overline{x},\overline{y})$ tritt ein ggT $\neq 1$ auf **then**
  (a) stoppe und gib den ggT zurück.
(5) **if** $\gamma \leq L_B\left(\frac{1}{\sqrt{2}}\right)$ **then**
  (a) $\gamma := 2\gamma$
  (b) goto 2
(6) **else**
  (a) goto 1

Bei diesem Algorithmus setzen wir

$$L_B(\beta) := e^{\beta\sqrt{\log B \cdot \log\log B}} \quad (\log = \ln)$$

Für $z \in_R [0,B]$ gilt

$$Ws_z[z \text{ ist } L_B(\beta) - \text{glatt}^*] = L_B\left(-\frac{1}{2\beta} + o(1)\right)$$

mit $o(1) \underset{B\to\infty}{\longrightarrow} 0$

$$\begin{aligned}
&(P)_p = P \in E_{a,b}(\mathbb{Z}_p)\\
&ord(P_p) \mid \left|E_{a,b}(\mathbb{Z}_p)\right|\\
&\quad ord(P_p) \mid gl(\gamma) \quad \Rightarrow \quad gl(\gamma)\cdot(P)_p = (O)_p \in E_{a,b}(\mathbb{Z}_p)\\
&\qquad\qquad\qquad\quad \Rightarrow \quad \text{es tritt ein ggT mit } p \mid \text{ggT auf}
\end{aligned}$$

4.6.6. Verteilungsannahme: Sei p kleinster Primfaktor von n und $p \leq \sqrt{n}$. Sei ausserdem $r_p \in_R \left(p-\sqrt{p}, p+\sqrt{p}\right)$ und $s_p \in_R [1,p]$. Nach Satz 4.5.2 auf Seite 73 gilt für $a,b \in_R \mathbb{Z}_p$

$$\begin{aligned} & Ws_{a,b}\left[\left|E_{a,b}(\mathbb{Z}_p)\right| \ \gamma-\text{glatt}^*\right] \\ \geq \quad & \Omega\left(\tfrac{1}{\log p} Ws_{r_p}[r_p\,\gamma-\text{glatt}^*]\right) \quad \forall \gamma \leq p. \end{aligned}$$

γ-glatt* bedeutet „*ohne Primfaktoren größer als* γ". $\Psi\left(x, x^{1/n}\right)$ in Satz 4.6.7 von Canfield, Erdös und Pomerance, bezieht sich auf $x^{1/n}$-glatt*. Dagegen erfaßt die EKM nur die p mit $\left|E_{a,b}(\mathbb{Z}_p)\right|$ γ-glatt. Der Unterschied ist nicht bedeutend.

4.6.7. GL-Annahme: Es gibt $c_1, c_2 > 0$, so dass für alle $\gamma < p$ gilt

$$Ws_{r_p}[r_p\,\gamma-\text{glatt}^*] \geq \frac{c_1}{(\log p)^{c_2}} \cdot Ws_{s_p}[s_p\,\gamma-\text{glatt}^*]$$

Diese Annahme hängt nicht von elliptischen Kurven ab. Es wird angenommen, dass der Anteil der γ-glatten[1] Zahlen im kleinen Intervall $\left(p-\sqrt{p}, p+\sqrt{p}\right)$ nicht viel kleiner ist als im großen Intervall $[1,p]$.

PROPOSITION 4.6.9. *(H. W. Lenstra) Unter der GL-Annahme findet die EKM mit EK-Wechsel jeden Primfaktor p von n mit* $p \leq B \leq \sqrt{n}$ *in* $L_B\left(\sqrt{2}+o(1)\right) = e^{\left(\sqrt{2}+o(1)\right)\sqrt{\log B \log\log B}}$ *arithm. Schritten modulo n.*

BEWEIS. Die Laufzeit von Schritt 2 ist gleich $(\text{Erfolgswahrscheinlichkeit})^{-1}$

$$\begin{aligned} \gamma &\leq L_B\left(\frac{1}{\sqrt{2}}+o(1)\right) \cdot L_B\left(\frac{1}{2\sqrt{2}}+o(1)\right) \\ &= L_B\left(\frac{1}{\sqrt{2}}+\frac{1}{2\sqrt{2}}+o(1)\right) \\ &= L_B\left(\sqrt{2}+o(1)\right) \end{aligned}$$

□

Praktisch ist $Ws_{a,b}\left[\left|E_{a,b}(\mathbb{Z}_p)\right| \ \gamma-\text{glatt}^*\right]$ sehr nahe an $Ws_{s_p}[s_p\,\gamma-\text{glatt}^*]$ aber dies ist nicht beweisbar. Für zufällige Primzahlen p gilt die GL-Annahme im Mittel.

Beim RSA-Alg. mit Schlüsseln der Länge $n = 500$ werden mit $B = 2^{250}$ $e^{\sqrt{2\log B \log\log B}} = e^{\sqrt{2\cdot 173\cdot 5{,}15}} = e^{42{,}27} \approx 2^{61}$ Schritte benötigt.

Bei $n = 2^{1000}$ sind es sogar $e^{\sqrt{2\cdot 346{,}6\cdot 5{,}89}} = e^{63{,}7} = 2^{91{,}8}$ Schritte (Rechenzeit bei dem Stand der Technik vom Jahr 2000 etwa 10 Jahre)!

[1] Der Erwartungswert bezieht sich auf die zufälligen a,b von $E_{a,b}(\mathbb{Z}_p)$.

4.7. Primheitsbeweise

Die Faktorisierung von n mit dem $(p-1)$-Verfahren von Pollard geht für Primfaktoren p mit $p-1$ γ-glatt. Bei der Elliptischen Kurven Methode (EKM) wurde diese Voraussetzung aufgehoben; dafür muss jetzt die Ordnung der Kurve $\left|E_{a,b}(\mathbb{Z}_p)\right|$ glatt sein.

LEMMA 4.7.1. [**Blömer**, Lemma 7.5]
Sei $n \in \mathbb{N}$. n ist genau dann eine Primzahl, wenn es ein $a \in \mathbb{Z}_n^*$ mit folgenden Eigenschaften gibt:

1) $a^{n-1} \equiv 1 \mod n$.

2) Für jeden Primteiler q von $n-1$ gilt

$$a^{\frac{n-1}{q}} \not\equiv 1 \mod n.$$

BEWEIS. Aus der ersten Bedingung wissen wir, dass $\mathrm{ord}_n(a) \mid n-1$. Angenommen, $\mathrm{ord}_n(a) \neq n-1$, dann ist $n-1 = c \cdot \mathrm{ord}_n(a)$ mit $c \neq 1$ ein Teiler von $n-1$. Sei nun q ein Primteiler von c, dann gilt:

$$a^{\frac{n-1}{q}} \equiv a^{\frac{c}{q} \cdot \mathrm{ord}_n(a)} \equiv \left(a^{\mathrm{ord}_n(a)}\right)^{\frac{c}{q}} \equiv 1 \mod n.$$

Dies steht aber im Widerspruch zu der zweiten Bedingung, es gilt also $ord_n(a) = n-1$. Bleibt zu zeigen, dass dann n prim ist. Dies folgt aus $ord_n(a) \mid \varphi(n)$, aber $\varphi(n) < n-1$, falls n nicht prim ist. □

Ein Verschärfung dieses Lemmas stellt der folgende Satz dar:

Satz von Pocklington

PROPOSITION 4.7.2. ***(Satz von Pocklington)***
Sei $n-1 = p^k \cdot r$, p prim und $ggT(p,r) = 1$ (k soll also maximal sein).
Falls es ein $a \in \mathbb{Z}_n^*$ gibt mit $a^{n-1} \equiv 1 \mod n$ und $ggT\left(a^{\frac{n-1}{p}} - 1, n\right) = 1$, so gilt für alle Teiler q von n, dass $q \equiv 1 \mod p^k$.

BEWEIS. Es genügt nach dem CRT diesen Satz für die Primteiler von n nachzuweisen. Sei also q Primteiler von n ($q \mid n$), dann gilt:

$$\begin{aligned}
a^{n-1} \equiv 1 \mod n &\overset{\text{CRT}}{\Rightarrow} a^{n-1} \equiv 1 \mod q \\
&\Rightarrow \mathrm{ord}_q(a) \mid n-1 \\
\mathrm{ggT}\left(a^{\frac{n-1}{p}} - 1, n\right) = 1 &\Rightarrow a^{\frac{n-1}{p}} \not\equiv 1 \mod q \\
&\Rightarrow \mathrm{ord}_q(a) \nmid \frac{n-1}{p} \\
&\Rightarrow p^k \mid \mathrm{ord}_q(a) \text{ und } \mathrm{ord}_q(a) \mid (q-1) \\
&\Rightarrow p^k \mid (q-1) \\
&\Rightarrow q \equiv 1 \mod p^k.
\end{aligned}$$

□

Damit ist die Grundlage für einen Primheitsbeweis gelegt, der nicht mehr eine vollständige Faktorisierung von $n-1$ verlangt, sondern nur noch eine Faktorisierung eines (großen) Teilers von $n-1$:

COROLLARY 4.7.3. *Sei $n-1 = f \cdot r$, $f > \sqrt{n}-1$ und $ggT(r,f) = 1$.*
Falls ein $a \in \mathbb{Z}_n^*$ existiert mit $a^{n-1} \equiv 1 \mod n$ und $ggT\left(a^{\frac{n-1}{p}} - 1, n\right) = 1$ für alle Primteiler p von f, dann ist n prim. Es gilt

$$Ws_a\left[a^{\frac{n-1}{p}} \equiv 1 \mod n\right] \leq \frac{1}{2}.$$

BEWEIS. Angenommen, n ist zusammengesetzt mit $q \leq \sqrt{n}$ Primfaktor von n. Nach Satz 4.7.2 auf der vorherigen Seite gilt

$$q \equiv 1 \mod p^k \text{ fuer alle Primzahlpotenzen } p^k \text{ mit } p^k \mid f,$$

und mit dem CRT folgt $q \equiv 1 \mod f$. Da $f > \sqrt{n}-1$ ist, muss dann aber $q > \sqrt{n}$ gelten, was einen Widerspruch darstellt. □

Kennt man also zu einem primen n von einem $\sqrt{n}$-Anteil von $n-1$ die Primfaktorzerlegung, dann kann man nach Korollar 4.7.3 einen Primheitsbeweis von n in polynomieller Zeit erwürfeln.

REMARK. Lenstra betrachtet explizit nur nichtsinguläre EK's mit

$$\Delta = -16\left(4a^3 + 27b^2\right) \neq 0.$$

Die singulären Kurven bringen für die EKM nichts, da die Verteilung der Gruppenordnung dann nicht mehr zufällig ist.

Die Vorteile der EKM sind zusammenfassend

(1) leichte Parallelisierbarkeit
(2) wenig Speicherbedarf ($O(\log n)$)
(3) kleine Primfaktoren werden sehr schnell gefunden.

PROPOSITION 4.7.4. *Sei $n \in \mathbb{N}$, $ggT(n,6) = 1$ (damit hat n weder die Primfaktoren 2 noch 3). Sei $E = E_{a,b}(\mathbb{Z}_n)$ und $m \in \mathbb{N}$.*

n ist prim, wenn gilt

(1) $\exists q \in \mathbb{P}\ q \mid m$ und $q > \left(\sqrt[4]{n}+1\right)^2$
(2) $\exists P \in E$, wobei die $[m] \cdot P$, $\left[\frac{m}{q}\right] \cdot P$ ohne nicht-triviale ggT berechenbar sind mit

$$[m] \cdot P = O,\ \left[\frac{m}{q}\right] \cdot P \neq O.$$

BEWEIS. Ang. n ist nicht prim, dann existiert ein $p \in \mathbb{P}$ mit $p \mid n$ und es gilt $p \leq \sqrt{n}$. Betrachte $(E)_p = E \mod p$. Dann gilt nach Hasse

$$\begin{aligned} \left|(E)_p\right| &\leq p + 2\sqrt{p} + 1 \\ &\leq \left(\sqrt{p}+1\right)^2 \\ &\leq \left(\sqrt[4]{n}+1\right)^2 \\ &< q. \end{aligned}$$

Also ist ggT $\left(\left|(E)_p\right|, q\right) = 1$ und $\exists u : uq = 1 \mod \left|(E_p)\right|$. Sei jetzt $P' \in (P)_p$, dann gilt in $(E)_p$

$$(m/q)\,P' = (m/q)\,(P)_p = uq\,(m/q)\,(P)_p = um\,(P)_p = u\,(O)_p = (O)_p$$

wegen der ersten Forderung in (2), was ein Widerspruch ist zu der zweiten Forderung in (2):

$$(m/q)\,P' = (m/q)\,(P)_p \neq (O)_p,$$

da $[m/q] \cdot P \neq O$ gelten soll und wegen dem CRT das gleiche auch für P' gilt. Also muss n prim sein. □

PROPOSITION 4.7.5. ***(Algorithmus von Schoof)***
Sei $E = E_{a,b}(\mathbb{Z}_p)$ mit p prim, dann kann man $|E|$ in $O(\log p)^8$ arithmetischen Schritten berechnen.

Wie man sieht ist das Verfahren nur für relativ kleine Primfaktoren praktikabel mit

$$\prod_i l_i > p + 1 + \sqrt{p} \quad \text{mit } l_i \text{ Primfaktoren.}$$

Aus diesem Satz kann man einen Algorithmus herleiten, der die Primheit einer Zahl n beweist und dafür den *Algorithmus von Schoof* einsetzt. Hierbei wählt man $(a,x,y) \in_R \mathbb{Z}_n$ und setzen $b \equiv y^2 - x^3 - ax \mod n$. Dann ist $P = (x,y)$ ein Element von $E_{a,b}(\mathbb{Z}_n)$ und wir zählen mit Hilfe des Algorithmus von Schoof die Anzahl der Punkte auf $E_{a,b}$, um die Zahl m zu finden, die im Fall n prim gleich $\left|E_{a,b}(\mathbb{F}_n)\right|$ ist. Wenn wir m nicht schreiben können als $m = kq$ mit $k \geq 2$ ist eine kleine Zahl und q wahrscheinlich prim, dann suchen wir eine neue EK. Wenn nicht, dann berechnen wir mP und kP und erhalten entweder einen nicht definierten Ausdruck (und damit einen Teiler von n), oder es können zwei Fälle eintreten: $mP \neq O$, dann ist n zusammengesetzt oder $kP = O$, dann müssen wir eine andere EK ausprobieren, oder $mP = O$ und $kP \neq O$: dann ist nach dem obigen Satz n mit Sicherheit prim, vorausgesetzt natürlich, dass q prim ist.

Im großen und ganzen ist das der Algorithmus von Goldwasser-Kilian, allerdings ist der Algorithmus von Schoof in der Praxis oft recht schwerfällig – gerade bei großen EKs. Außerdem ist es nach dem Satz von Hasse zwar so, dass in dem Intervall $\left[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}\right]$ die Zahl m enthalten ist, jedoch ist die Erfolgswahrscheinlichkeit nicht unbedingt so hoch, dass wir nach erwartet polynomieller Laufzeit ein E gefunden haben, dass unsere Voraussetzungen erfüllt. Praktische Erfahrungen zeigen jedoch, dass das die Laufzeit des Algorithmus für den praktischen Einsatz gut genu ist.

KAPITEL 5

Quadratisches Sieb

5.1. Fermat-Faktorisierung

Eng verwandt mit dem Verfahren des Quadratischen Siebs ist die sog. Fermat Faktorisierung. Diese Verfahren wird bei der Faktorisierung von $n = pq$ angewandt, wobei p und q eng beieinander liegen. Im Folgenden wird vor allem auf [**Koblitz1994**, S. 143-153] Bezug genommen.

PROPOSITION 5.1.1. *Sei n ungerade. Dann ist eine 1-zu-1-Abbildung zwischen der Faktorisierung von n der Form $n = ab$ mit $a \geq b > 0$ und Darstellungen von n der Form $n = t^2 - s^2$, wobei s und t nicht-negative ganze Zahlen sind, gegeben durch:*

$$t = \frac{a+b}{2}, \quad s = \tfrac{a-b}{2} \quad \textit{mit } a = t+s,\, b = t-s.$$

BEWEIS. Wenn man bereits über eine Faktorisierung von n der Form $n = ab$ verfügt, kann man n auch schreiben als

$$n = ab = \left(\frac{(a+b)}{2}\right)^2 - \left(\frac{(a-b)}{2}\right)^2 = t^2 - s^2.$$

Umgekehrt kann man genauso bei einem gegebenen $n = t^2 - s^2$ die rechte Seite in zwei Faktoren aufspalten und erhält dann mit den obigen Gleichungen unser $n = ab$. □

Wenn a und b sehr nahe beieinander liegen, so ist $s = (a-b)/2$ sehr klein und $t = (a+b)/2$ nur etwas grösser als $\sqrt{n}$. In diesem Fall können wir alle a und b finden, indem wir sukzessive alle Werte von $\lfloor\sqrt{n}\rfloor + 1$ an für t durchprobieren, bis wir einen gefunden haben, für den gilt

$$t^2 - n = s^2 \quad \text{ist ein quadratischer Rest.}$$

Die Idee hinter der Fermat-Faktorisierung, die zu einer noch wesentlich effizienteren Methode führt, basiert auf dem Konzept der Faktorbasen. Im Grunde genommen suchen wir die ganze Zeit nach t, s mit $t^2 \equiv s^2 \mod n$ und $t \not\equiv \pm s \mod n$. Dann finden wir nämlich sofort einen Faktor von n, indem wir den $\mathrm{ggT}(t+s, n)$ (oder $\mathrm{ggT}(t-s, n)$) ausrechnen, da gilt

$$n \mid t^2 - s^2 = (t+s)(t-s),$$

während n nicht $t+s$ oder $t-s$ teilt. D.h. $\mathrm{ggT}(t+s, n)$ ist eine echter Teiler von n und $b = n/a$ teilt $\mathrm{ggT}(t-s, n)$.

An dieser Stelle ist die Frage offen, wie man solche $t^2 \equiv s^2 \mod n$ findet. Wenn n sehr groß ist, ist dies nämlich sehr schwer. Die Idee ist nun, dass man einige b_i's wählt, für die gilt, dass $b_i^2 \mod n$ ein Produkt kleiner Primzahlpotenzen ist und so ein Produkt einer Teilmenge dieser b_i ein b ergibt, dessen Quadrat wieder kongruent ist zu einem quadratischen Rest modulo n.

Im Folgenden verstehen wir unter dem „**kleinsten absoluten Rest**" einer Zahl a modulo n eine Zahl aus $[-n/2, n/2]$, die zu a kongruent ist und schreiben diese als $a \mod n$.

5.2. Faktorbasisalgorithmus

In [**Koblitz1994**] wird das quadratische Sieb unter der Bezeichnung *Faktorbasisalgorithmus* behandelt. Wieder der Name schon sagt, ist der Begriff der *Faktorbasis* ganz elementar und soll zunächst definiert werden:

Faktorbasis

B-Zahl

DEFINITION. **(Faktorbasen)** Eine *Faktorbasis* ist eine Menge $B = \{p_1, p_2, \ldots, p_h\}$ von paarweise verschiedenen Primzahlen und $p_1 = -1$[1]. Eine Zahl b ist eine ***B*-Zahl** (bzgl. einem gegebenen n), wenn der kleinste absolute Rest $b^2 \mod n$ als Produkt von Zahlen aus B geschrieben werden kann.

Sei nun $\mathbb{F}_2^h$ ein Vektorraum über zwei Elementen, die aus h-Tupeln von 0 und 1 bestehen. Für ein gegebenes n und eine Faktorbasis B, die h Zahlen enthält, zeigen wir nun, wie ein Vektor $\vec{\varepsilon} \in \mathbb{F}_2^h$ zu jeder B-Zahl b korrespondiert. Wir schreiben also $b^2 \mod n = \prod_{j=1}^h p_j^{\alpha_j}$ und setzen die j-te Komponente $\varepsilon_j = \alpha_j \mod 2$ (also $\varepsilon_j = 0$ wenn α_j gerade, sonst $\varepsilon_j = 1$).

Angenommen, wir kennen eine Menge von B-Zahlen b_i, die zu Vektoren $\overrightarrow{\varepsilon} = (\varepsilon_{i1}, \ldots, \varepsilon_{ih})$ korrespondieren und deren Komponenten sich zu dem Nullvektor in $\mathbb{F}_2^h$ aufaddieren. Dann ist das Produkt der kleinsten absoluten Reste der $b_i^2 \mod n$ gleich dem Produkt der *geraden* Exponenten aller p_j in B. Für jedes i setzen wir a_i als den kleinsten absoluten Rest von $b_i^2 \mod n$ und schreiben $a_i = \prod_{j=1}^h p_j^{\alpha_{ij}}$. Damit erhalten wir

$$\prod_i a_i = \prod_{j=1}^h p_j^{\sum_i \alpha_{ij}},$$

wobei der Exponent aller p_j auf der rechten Seite eine gerade Zahl ist. Dann ist die rechte Seite auch ein Quadrat von $\prod_j p_j^{\gamma_j}$ mit $\gamma_j = \frac{1}{2}\sum_i \alpha_{ij}$. Also setzen wir $b = \prod_i b_i \mod n$ (der kleinste positive Rest) und $c = \prod_j p_j^{\gamma_j} \mod n$ (kleinster positiver Rest) und erhalten so zwei Zahlen b und c, die auf unterschiedliche Art und Weise konstruiert wurden (eine als Produkt der b_i's und das andere als Produkt der p_j's) und deren Quadrat kongruent ist modulo n.

Jetzt kann es natürlich passieren, dass $b \equiv \pm c \mod n$ ist. In diesem Fall müssen wir nochmal von vorne anfangen mit einer anderen Menge von B-Zahlen, deren korrespondierende Vektoren sich zum Nullvektor aufaddieren. Das passiert z.B. dann, wenn wir dummerweise B-Zahlen $< \sqrt{n/2}$ wählen: In diesem Fall sind Quadrate der B-Zahlen $< n/2$. Damit ist dann immer $b^2 \equiv c^2 \mod n$ mit $b \equiv c \mod n$, da der kleinste absolute Rest der $b_i^2 \mod n$ immer $< n/2$ ist. Koblitz ist in [**Koblitz1994**, Seite 146] der Meinung, dass dann alle Vektoren aus $\mathbb{F}_2^h$ Null-Vektoren wären – das kann ich nicht so ganz nachvollziehen. Ist z.B. $B = \{-1, 2, 3\}$ und $b_1 = 27 = 9^2 = 3^3$, dann ist der entsprechende Vektor aus $\mathbb{F}_2^h$ nicht der Nullvektor.

[1] Die -1 muss zu der Faktorbasis hinzugenommen werden, da negative Zahlen kein Quadrat sein können – die -1 in der Faktorbasis sorgt dann dafür, dass negative Zahlen nicht fälschlicher Weise für ein Quadrat gehalten werden.

Für zufällig gewählte b_i erwarten wir, weil n zusammengesetzt ist, dass b und c in mindestens der Hälfte aller Fälle (bis auf ± 1) kongruent sind modulo n. Da gilt, weil jedes Quadrat modulo n $2^r > 4$ Quadratwurzeln besitzt, wenn n r verschiedene Primteiler hat[2]. Also hat eine zufällige Quadratwurzel von b^2 nur mit einer Wahrscheinlichkeit von $2/2^r \leq 1/2$ eine Chance entweder b oder $-b$ zu sein. Gehen wir also die Prozedur oben durch, bis wir ein b und ein c gefunden haben, die einen nicht-trivialen Faktor von n liefern, dann besteht eine Wahrscheinlichkeit von höchstens 2^{-k}, dass wir mehr als k Versuche dafür brauchen.

Wie wählen wir aber in der Praxis die Faktorbasis B und die B-Zahlen b_i? Eine Methode wäre, mit einer Menge B zu starten, die aus den ersten h Primzahlen (bzw. den ersten $h-1$ Primzahlen mit der -1) besteht und dann zufällige b_i's zu wählen, bis genügend viele gefunden wurden, die B-Zahlen sind.
Eine andere Methode wäre, mit zufälligen b_i's zu starten, für die $b_i^2 \mod n$ (der kleinste absolute Rest) einen kleinen absoluten Wert hat (z.B. die b_i, die nahe an $\sqrt{kn}$ für kleine kn sind). Dann konstruiere man B derart, dass es eine kleine Menge kleiner Primzahlen ist, so dass einige der $b_i^2 \mod n$ durch Zahlen aus B ausgedrückt werden können.

Wann können wir uns sicher sein, dass wir genügend b_i gefunden haben, so dass die Summe der $\vec{\varepsilon}_i$ der Nullvektor ist? Anders formuliert: Wann können wir uns sicher sein, aus einer Menge von Vektoren aus $\mathbb{F}_2^h$ eine Untermenge gefunden zu haben, deren Summe der Nullvektor ist? Dies ist äquivalent zu der Frage nach einer Menge von Vektoren, die linear unabhängig über dem Körper $\mathbb{F}_2$ sind. Das ist nach Erkenntnissen der Linearen Algebra bei $h+1$-vielen Vektoren der Fall. Schlimmstenfalls müssen wir also $h+1$ verschiedene B-Zahlen generieren, um unser erstes Beispiel von

$$\left(\prod_i b_i\right)^2 \equiv \left(\prod_j p_j^{\gamma_j}\right)^2 \mod n$$

zu finden. Wenn h sehr groß ist, kann es sehr schwer sein eine Untermenge von Zahlen zu finden, deren Vektoren sich zum Nullvektor aufaddieren. In diesem Fall müssen wir die $h+1$ Gleichungen in eine Matrix schreiben und diese „von Hand" lösen (z.B. mit dem Gauß'schen Reduktionsverfahren).

Zusammenfassend ergibt sich damit Algorithmus 30, der auch unter der Bezeichnung „Faktorbasis Algorithmus" bekannt ist.

Dieser Algorithmus entspricht im wesentlichen dem Quadratischen Sieb, das im nächsten Abschnitt behandelt wird. Zu der Bedeutung von y schreibt Pomerance in [**Pomerance1996**], dass es einfach eine Schätzung für das größte Element der Faktorbasis ist. Mit diesem y stehen natürlich die Zahlen in Beziehung, die y-glatt sind – und das sind diejenigen, die in Schritt 3.a.i) gesucht werden.

5.2.1. Analyse. Zur Laufzeitanalyse des Algorithmus werden zunächst einige Tatsachen benötigt:

Formel von Stirling

FACT 5.2.1. ***(Formel von Stirling)*** *Für $n \to \infty$ gilt:* $\log(n!) \approx n\log(n) - n$.

Diese Aussage gilt nur für sehr große n und bedeutet, dass die Differenz auf der rechten Seite für sehr große n wesentlich langsamer wächst, als n.

[2] Ist $x^2 \equiv 1 \mod n$ mit $n = \prod_{i=1}^k p_i^{e_i}$, dann ist nach CRT $x^2 \equiv 1 \mod p_i^{e_i}$ und damit $x \equiv \pm 1 \mod p_i^{e_i}$.

Algorithm 30 Faktorbasis Algorithmus

EINGABE: $n \in \mathbb{N}$ ungerade.
AUSGABE: $d \in \mathbb{N}$ mit $d \mid n$ und $d \neq 1, n$.

(1) **** wenn n eine 50-stellige Zahl ist, wähle y z.B. als 5-stellige Zahl ****
Wähle ein $y \in \mathbb{N}$ von mittlerer Grösse.
(2) **** Wahl der Faktorbasis ****
Sei B die Menge aller Primzahlen $\leq y$ zuzüglich der -1.
(3) **do**
(a) **** $\pi(y)$-viele sollten genügen, wobei $\pi(y) = |\{p < y \mid p \text{ prim}\}|$ ****
do
(i) Wähle zufällig ein großes b_i und versuche $b_i^2 \mod n$ durch eine Produkt von Zahlen aus B auszudrücken.
(b) **until** genug b_i wurden gewählt
(c) Berechne die korrespondierenden Vektoren aus $\mathbb{F}_2^h$ mit $h = \pi(y) + 1$.
(d) **do**
(i) **** z.B. mit dem Gausschen Reduktionsverfahren ****
Bestimme eine Teilmenge der b_i.
(e) **until** die Summe der korrespondierenden $\overrightarrow{\varepsilon_i}$ addieren sich zu dem Nullvektor.
(f) **** Berechne b und c mit $b^2 \equiv c^2 \mod n$ ****
Setze $b := \prod b_i \mod n$ und $c := \prod p_i^{\gamma_i} \mod n$.
(4) **** Wenn die B-Zahlen ungeschickt gewählt wurden, dann neue B-Zahlen wählen, oder einfach andere Zeilenkombinationen der $\overrightarrow{\varepsilon}$-Matrix nehmen ****
until $b \not\equiv c \mod n$.
(5) $d := \mathrm{ggT}(b + c, n)$.
(6) **return(**d**)**.

FACT 5.2.2. *Für $N \in \mathbb{N}$ und $u > 0$ ist die Gesamtanzahl aller nicht-negativen N-tupel α_j mit $\sum_{j=1}^{N} \alpha_j \leq u$ gleich dem Binomialkoeffizient $\binom{\lceil u \rceil + N}{N}$.*

BEWEIS. Ein einfacher Beweis ist die Analogie zu der Aufgabe, $\lceil u \rceil$ Kugeln auf N Schachteln zu verteilen. Lösung: Denke an $\lceil u \rceil + N$ viele Plätze, dan kann jeder Platz entweder von einer Kugel oder einer Zwischenwand belegt werden. Etwas komplizierter und „mathematischer" wird das im Folgenden bewiesen:

Für jedes N-tupel α_j wähle N ganze Zahlen β_j aus $\{1, 2, \ldots, \lceil u \rceil + N\}$ und setze

$$\begin{aligned} \beta_1 &= \alpha_1 + 1 \\ \beta_{j+1} &= \beta_j + \alpha_{j+1} + 1 \quad j \geq 1. \end{aligned}$$

Damit sind die β_j derart konstruiert, dass es α_j Zahlen zwischen β_{j-1} und β_j gibt. Dies ergibt eine 1-zu-1 Abbildung zwischen der Anzahl der Lösungen, die die Ungleichung erfüllen, und der Anzahl der Möglichkeiten, N-Zahlen aus der Menge von $\lceil u \rceil + N$ Zahlen zu wählen. □

Um die Laufzeit des Algorithmus zu bestimmen gilt es, die Wahrscheinlichkeit zu berechnen, dass eine zufällige Zahl kleiner als x ein Produkt von Primzahlen kleiner als y ist,

wobei y natürlich kleiner als x ist:

$$\text{Ws}\left(b = \prod_{i=1}^{k} p_i^{e_i} \in_R \mathbb{N}\right) \quad \text{mit } b < x, x < y \text{ und } p_i < y \forall i.$$

Dazu setzen wir

$$u := \frac{\log x}{\log y}.$$

Damit ist u in etwa gleich dem Koeffizient aus der Anzahl der Bits der Binärdarstellung der Zahlen x und y. Im Folgenden nehmen wir an, dass u wesentlich kleiner ist als y und wollen unter $\pi(y)$ die Anzahl der Primzahlen $\leq y$ verstehen mit

$$\pi(y) \approx \frac{y}{\log y}.$$

Wegen dem Primzahlsatz gehen wir davon aus, dass u wesentlich kleiner als $\pi(y)$ ist. Außerdem sei

$$\Psi(x,y) = |\{k \leq x \mid k \text{ ist } y-\text{glatt}\}| = \text{Anzahl der } y-\text{glatten Zahlen } \leq x.$$

Damit beschreibt $\Psi(x,y)$ die Anzahl der Zahlen $k = \prod p_j^{\alpha_j} \leq x$ mit $p_j \leq y$ prim und $\alpha_j \in \mathbb{N}$ für alle j. Offensichtlich existiert eine 1-zu-1-Abbidung zwischen den $\pi(y)$-Tupeln mit den α_j und den Zahlen $\leq x$, die y-glatt sind. Also gilt

$$\Psi(x,y) \quad \approx \quad \left|\left\{\alpha_j \mid \sum_{j=1}^{\pi(y)} \alpha_j \log p_j \leq \log x\right\}\right|.$$

Im Folgenden nehmen wir an, dass für die meisten p_j der Wert von $\log p_j$ nicht wesentlich kleiner ist als $\log y$, da die meisten Primzahlen kleiner als y in etwa die gleiche Anzahl an Stellen haben wie y. Aus diesem Grund können wir in der Ungleichung oben die meisten $\log p_j$ durch $\log y$ ersetzen und $\log x / \log y$ durch u ersetzen und erhalten damit

$$\Psi(x,y) \quad \approx \quad \left|\left\{\alpha_j \mid \sum_{j=1}^{\pi(y)} \alpha_j \leq u\right\}\right|.$$

Im Folgenden setzen wir $\pi(y)$ durch y, was zwar zu schlimmen Fehlern führen kann, sich in der Regel jedoch nicht schlimmer auf das Ergebnis auswirkt, als es genauere Approximationen von $\pi(y)$ tun würden. Unsere Abschätzung von $\Psi(x,y)$ ist damit

$$\Psi(x,y) \quad \approx \quad \left|\left\{\alpha_j \mid \sum_{j=1}^{y} \alpha_j \leq u\right\}\right|.$$

Nach Fakt 5.2.2 und $N = y$ ergibt sich nun

$$\Psi(x,y) \quad \approx \quad \binom{\lceil u \rceil + y}{y}.$$

Als nächstes berechnen wir den Logarithmus der Wahrscheinlichkeit, dass eine zufällige Zahl zwischen 1 und x ein Produkt von Primzahlen $\leq y$ ist. Diese Wahrscheinlichkeit ist

$$\text{Ws}(k \in_R [1,x] \text{ ist } y-\text{glatt}) \quad = \quad \frac{\Psi(x,y)}{x}.$$

Dazu benutzen wir die Approximation für $\Psi(x,y)$, Fakt 5.2.1 und $\log(x) = u\log(y)$:

$$\begin{aligned}
\log\left(\frac{\Psi(x,y)}{x}\right) &\approx \log\left(\binom{\lceil u\rceil + y}{y}\right) - \log(x) \\
&\approx \log\left(\frac{(\lceil u\rceil + y)!}{\lceil u\rceil ! y!}\right) - u\log(y) \\
&\approx (\lceil u\rceil + y)\log(\lceil u\rceil + y) - (\lceil u\rceil + y) - \\
&\quad (\lceil u\rceil \log\lceil u\rceil - \lceil u\rceil) - (y\log y - y) - u\log y.
\end{aligned}$$

Wir nähern nun $\lceil u\rceil$ durch u an und ersetzen $\log(u+y)$ durch $\log y$, da wir annehmen können, dass u viel kleiner als y ist:

$$\begin{aligned}
\log\left(\frac{\Psi(x,y)}{x}\right) &\approx (u+y)\log y - (u+y) - (u\log u - u) - \\
&\quad (y\log y - y) - u\log y \\
&\approx -u\log u
\end{aligned}$$

und damit

$$\frac{\Psi(x,y)}{x} \approx u^{-u}.$$

Es gilt also

$$\mathrm{Ws}(k \in_R [1,x] \text{ ist } y-\text{glatt}) \approx u^{-u} \quad \text{mit } u = \frac{\log x}{\log y}.$$

Um nun die Anzahl der Bitoperationen des Algorithmus abzuschätzen, nehmen wir an, dass unsere Faktorbasis B aus den ersten $h = \pi(y)$ Primfaktoren, also z.B. allen Primzahlen $\leq y$ besteht. Weiterhin nehmen wir an, dass B nicht die -1 enthält und wir den kleinsten *positiven* Rest (nicht den kleinsten *absoluten* Rest) $b_i^2 \mod n$ betrachten.

Die Anzahl der Bitoperationen in den einzelnen Schritten sind damit

(1) Wahl der $b_i \in_R [1,n]$ und Berechnung deren Darstellung durch den kleinsten positiven Rest von $b_i^2 \mod n$ als Produkt von Primzahlen $\leq y$, solange, bis $\pi(y)+1$ verschiedene b_i's gefunden wurden, für die $b_i^2 \mod n$ als Produkt dieser Primzahlen geschrieben werden kann.
(2) Finden einer Menge von linear abhängigen Zeilen der korrespondierenden $(\pi(y)+1)\times\pi(y)$-Matrix um die Kongruenz $b^2 \equiv c^2 \mod n$ mit $b \not\equiv \pm x \mod n$ zu finden.
(3) Wenn gilt $b \equiv \pm c \mod n$, dann wiederhole (1) und (2) solange, bis das nicht mehr gilt und berechne dann $\mathrm{ggT}(b+c,n)$.

Nach obiger Rechnung benötigen wir im Mittel u^u-viele Versuche, bis wir ein b_i gefunden haben, so dass $b_i^2 \mod n$ ein Produkt von Primzahlen $\leq y$ ist ($u = \log n/\log y$). Für große y wird u^u also klein und wir werden sehr häufig b_i mit den gewünschten Eigenschaften finden. Ein großes y gereicht uns allerdings bei der Faktorisierung der $b_i^2 \mod n$ zum Nachteil – diese müssen wir nämlich $\pi(y)+1$-mal durchführen, was sehr aufwändig ist. Wählen wir y jedoch sehr klein, so brauchen wir sehr lange, bis wir geeignete b_i gefunden haben. Wir sollten also y als Kompromiss zwischen groß und klein wählen. Deshalb machen wir zunächst ein paar grobe Annahmen zu y:

Angenommen, n ist eine r-Bit Zahl und y besitzt s-viele Bits. In diesem Fall ist u sehr nah an r/s. Wie viele Bitoperationen werden für das Testen jedes zufällig gewählten b_i's benötigt?

- Wir behaupten, dass die Anzahl der Operationen polynomiell ist in r und y, z.B. $O\left(r^l e^{ks}\right)$ für einige (sehr kleine) Zahlen k und l.
- Für das Erzeugen eines zufälligen $b_i \in [1,n]$ wird eine feste Zeit benötigt, also $O(r)$ Bitoperationen.
- Die Berechnung von $b_i^2 \mod n$ benötigt $O\left(r^2\right)$-viele Bitoperationen.
- Das sukzessive Teilen von $b_i^2 \mod n$ durch alle Primzahlen $\leq y$, die y mit gerader Potenz teilen, benötigt mit dem trivialen Algorithmus $O(yrs)$, mit $O(rs)$ als die Zeit, die benötigt wird, um eine r-Bit Zahl durch eine s-Bit Zahl zu teilen.

In Schritt (1) werden also etwa $u^u(\pi(y)+1)$-viele b_i berechnet, um $\pi(y)+1$-viele b_i zu finden, deren Quadrate modulo n aus der Faktorbasis B (also von Primzahlen $\leq y$) erzeugt werden können. Unter der Voraussetzung, dass gilt $\pi(y) \approx \frac{y}{\log y} = O(y/s)$ gilt also mit $O\left(u^u \frac{y}{s}(r^2+yrs)\right) = O\left(u^u r y^2\right)$ benötigt $O(u^{u} r y^{2}\)$ Bitoperationen}.

Schritt (2) ist polynomiell in y und r, da es sich hierbei um eine Matrix-Reduktion und das Finden von b und c modulo n handelt.

Bei jeder Durchführung der Schritte (1) und (2) besteht eine 50% Wahrscheinlichkeit für die Hoffnung auf Erfolg. Wir nehmen an, dass eine Wahrscheinlichkeit von $1-2^{-50}$ ausreicht, um einen nicht-trivialen Faktor von n zu finden – es soll also genügen, die Schritte (1) und (2) 50-mal auszuführen. Damit ergibt sich eine Gesamtlaufzeit des Algorithmus von

$$O\left(50\left(u^u r y^2 + y^j r^h\right)\right) = O\left(r^h u^u y^j\right) = O\left(r^h u^u e^{ks}\right) = O\left(r^h \left(\frac{r}{s}\right)^{\frac{r}{s}} e^{ks}\right)$$

für geeignete $h,k \in \mathbb{N}$.

Als nächstes gilt es, die y und s zu finden, für die die Laufzeit minimal wird. Da r die Anzahl der Bits von n bezeichnet, handelt es sich hier um eine Konstante, da n fest ist und damit ist $(r/s)^{r/s} e^{ks}$ bzgl. s zu minimieren. Das wiederum ist äquivalent dazu, den Logarithmus zu minimieren:

$$0 = \frac{d}{ds}\left(\frac{r}{s}\log\frac{r}{s} + ks\right) = -\frac{r}{s^2}\left(\log\frac{r}{s}+1\right) + k \approx -\frac{r}{s^2}\log\frac{r}{s} + k.$$

Wir können also s derart wählen, dass $k \approx r/s^2 \log(r/s)$ ist. In anderen Worten, sollen also die beiden Faktoren in $(r/s)^{r/s} e^{ks}$ in etwa gleich groß sein. Weil k eine Konstante ist folgt, dass s^2 etwa gleich groß ist wie $r\log(r/s) = r(\log r - \log s)$, s liegt also etwa zwischen $\sqrt{r}$ und $\sqrt{r\log r}$. Das bedeutet aber, dass $\log s$ in etwa so groß ist wie $\frac{1}{2}\log r$ und damit ergibt die obere Relation unter Substitution von $\log s \approx \frac{1}{2}\log r$

$$0 \approx -\frac{r}{2s^2}\log r + k, \quad \text{z.B.} \quad s \approx \sqrt{\frac{r}{2k}\log r}.$$

Mit diesem Wert können wir jetzt die Laufzeit des Algorithmus abschätzen, da sie für dieses s minimal sein sollte. Da $(r/s)^{r/s}$ und e^{ks} für unser optimales s etwa gleich groß sind, schätzen wir die Anzahl der Bitoperationen für die Faktorisierung einer r-Bit Zahl folgendermassen ab:

$$O\left(e^{c\sqrt{r\log r}}\right).$$

Diese Abschätzung ist natürlich sehr grob und die Vereinfachungen, die durchgeführt wurden keineswegs belegt und zudem handelt es sich um eine erwartete Laufzeit. Bis die Methode des Zahlkörpersiebs gefunden wurde, gab es keinen allgemeinen Faktorisierungsalgorithmus, der diese Laufzeit unterbot.

PROPOSITION 5.2.3. *Da gilt $r = O(\log n)$ ergibt sich für den Faktorbasis Algorithmus eine erwartete Laufzeit von*

$$O(\textit{Faktorbasis Algorithmus}) \quad = \quad O\left(e^{c\sqrt{\log n \log\log n}}\right).$$

REMARK. Wie in Abschnitt 3.1 behandelt, beruht die Sicherheit des RSA-Schemas auf der Schwierigkeit, eine sehr große Zahl $n = pq$ mit p,q prim zu faktorisieren, was polynomiell in der Anzahl r der Bits von n ist. Da diese Laufzeit in etwa durch $O\left(e^{c\log r}\right)$ beschrieben werden kann, sieht man, dass der Faktorbasis Algorithmus für große r um einiges besser ist, als z.B. die ρ-Methode, die in Zeit $O\left(\sqrt[4]{n}\right) = O\left(e^{cr}\right)$ einen Faktor von n findet.

Da es beim praktischen Einsatz von Faktorisierung sehr stark auf die Wahl von Konstanten (hier c) ankommt, können diese einen großen Einfluss auf die reale Laufzeit eines Faktorisierungalgorithmus haben – in der Theorie wird ihnen jedoch nicht so viel Beachtung geschenkt.

5.3. Das Quadratische Sieb

Im vorhergehenden Abschnitt wurde der Faktorbasis Algorithmus nach [**Koblitz1994**] besprochen. Dabei handelt es sich im wesentlichen um das nun folgende Quadratische Sieb nach [**Blömer**], das auf Satz 5.3.1 aufbaut:

PROPOSITION 5.3.1. *Sei $n \in \mathbb{N}$ ungerade, zusammengesetzt und keine Primzahlpotenz. Dann gibt es Zahlen $x,y \in \mathbb{N}$, für die gilt*

$$x \not\equiv \pm y \mod n \quad \textit{und} \quad x^2 \equiv y^2 \mod n.$$

BEWEIS. Sei also $n = n_1 n_2$ mit $\mathrm{ggT}(n_1,n_2) = 1$ und $y \in \mathbb{Z}_n^*$. nach dem CRT gibt es dann ein $x \in \mathbb{Z}_n^*$ mit

$$\begin{aligned} x &\equiv y \mod n_1 \\ x &\equiv -y \mod n_2 \end{aligned}$$

(läßt sich konstruieren). Dann gilt auch (nach CRT)

$$x^2 \equiv y^2 \mod n.$$

Bleibt zu zeigen, dass gilt $x \not\equiv \pm y \mod n$. Wäre das nicht der Fall, so wäre auch entweder $x \equiv -y \mod n_1$ oder $x \equiv y \mod n_2$. Im ersten Fall gilt aber auch $x \equiv y \mod n_1$ und damit $2x_1 \equiv 0 \mod n_1$. Da aber n und damit auch n_1 ungerade ist, folgt $n_1 \mid x$ und damit $\mathrm{ggT}(x,n) \neq 1$ – Widerspruch zu $x \in \mathbb{Z}_n^*$!
Der Fall $x \equiv -y \mod n_2$ kann analog bewiesen werden. □

Wie man in dem Beweis sieht gibt es zu jedem $y \in \mathbb{Z}_n^*$ ein $x \in \mathbb{Z}_n^*$, das konstruiert werden kann und die obigen Bedingungen erfüllt – es gibt also $\varphi(n)$-viele Paare (x,y), mit $x \not\equiv \pm y \mod n$ und $x^2 \equiv y^2 \mod n$. Von besonderer Bedeutung ist für uns in diesem Zusammenhang, dass gilt

$$\begin{aligned} x^2 \equiv y^2 \mod n \quad &\Rightarrow \quad n \mid (x-y)(x+y) \\ \text{mit } x \not\equiv \pm y \mod n \quad &\Rightarrow \quad \mathrm{ggT}(n, x-y) \notin \{1,n\}. \end{aligned}$$

Wir haben also einen echten Teiler von n gefunden!

Das *Quadratische Sieb* ist nun ein Verfahren, möglichst effizient Paare $(x,y) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$ mit $x^2 \equiv y^2 \mod n$ und $x \not\equiv \pm y \mod n$ zu finden, um so einen echten Teiler von n zu bestimmen. Die Idee des Algorithmus ist:

(1) Wähle eine Faktorbasis $\{p_1, \ldots, p_l\}$ von kleinen Primzahlen p_j mit $p_1 = -1$.
(2) Bestimme Kongruenzen $x_i^2 \equiv z_i \mod n$ mit $x_i, z_i \in \mathbb{Z}_n$ und $z_i = \prod_{j=1}^{l} p_j^{e_{ij}}$, $e_{ij} \in \mathbb{N}$ (es werden also nur z_i's verwendet, die sich aus der Faktorbasis konstruieren lassen).
(3) **do**
  (a) Erzeuge $f_i \in \{0,1\}$, wobei nicht alle $f_i = 0$ sein sollen mit $z = \prod z_i^{f_i}$ ist ein Quadrat in $\mathbb{N}$ (es existiert also ein $y \in \mathbb{N}$ mit $y^2 = z$).
  (b) Bestimme dieses y mit $y^2 = z$ und setze $x = \prod x_i^{f_i}$.
(4) **until** $x \not\equiv \pm y \mod n$ und $x^2 \equiv y^2 \mod n$.

Schon beim ersten Betrachten des Algorithmus fällt auf, dass hier Dinge verlangt werden, von denen nicht so ganz klar ist, wie sie durchgeführt werden sollen. Beginnen wir von hinten und betrachten den Fall, dass wir in Schritt 3.a) $l+1$-viele Kongruenzen $x_i^2 \equiv z_i \mod n$ gefunden haben, mit $f_i = \prod_{j=1}^{l} p_j^{e_{ij}}$. Dann soll gelten, dass

$$\prod_{i=1}^{l+1} z_i^{f_i} = \prod_{i=1}^{l+1} \left(\prod_{j=1}^{l} p_j^{e_{ij}} \right)^{f_i} = \prod_{j=1}^{l} p_j^{\sum_{i=1}^{l+1} f_i e_{ij}}$$

ein Quadrat in $\mathbb{N}$ ist. Das ist aber nur dann der Fall, wenn alle Exponenten $\sum_{i=1}^{l+1} f_i e_{ij}$ gerade sind, also

$$\begin{aligned}
\sum_{i=1}^{l+1} f_i e_{i1} &\equiv 0 \mod 2 \\
\sum_{i=1}^{l+1} f_i e_{i2} &\equiv 0 \mod 2 \\
&\vdots \\
\sum_{i=1}^{l+1} f_i e_{il} &\equiv 0 \mod 2.
\end{aligned}$$

Da dieses Gleichungssystem $l+1$ Variablen, aber nur l Gleichungen besitzt, gibt es immer eine Lösung $f_1, \ldots, f_{l+1}$, wobei nicht alle $f_i = 0$ sind. Diese Lösung kann z.B. mit dem Gauß'schen Eliminationsverfahren gefunden werden – der Algorithmus liefert also eine korrekte Lösung, wenn Schritt (2) und Schritt (3) korrekt durchgeführt wurden.

Gehen wir nun den Algorithmus Schrittweise durch:

(1) Wählen der Faktorbasis:
Für diesen Schritt werden kleine Primzahlen benötigt, die relativ leicht erzeugt werden können (z.B. mit dem Sieb des Erathostenes).
(2) Finden der Kongruenzen $x_i^2 \equiv z_i \mod n$:
Damit der Algorithmus eine gescheite Lösung findet ($x \not\equiv \pm y \mod n$), darf auf keinen Fall gelten $x_i^2 = z_i$. Aus diesem Grund sollte man $x_i > \sqrt{n}$ wählen. Die

Forderung ist, dass sich die $z_i \equiv x_i^2 \mod n$ durch die kleinen Primzahlen p_j vollständig faktorisieren lassen – am besten hält man also die z_i möglichst klein. Diese beiden Bedingungen führen zu dem Schluss, dass man die x_i in der Nähe von $m := \lceil \sqrt{n} \rceil$ wählt. Im Algorithmus setzt man $x_i := m + i$. Für kleine i ergibt sich damit $z_i \equiv x_i^2 \equiv (m+i)^2 \mod n$ ist etwa so groß wir $i \cdot m$ – bei kleinen i etwa $im \approx \sqrt{n}$.

Wie können die z_i schnell faktorisiert werden?

Trivialerweise geht das durch Division aller p_j – was allerdings nicht sonderlich effizient ist. Da $m = \lceil \sqrt{n} \rceil$ und $z_i \equiv (m+i)^2 \mod n$ ist, gilt für kleine i

$$z_i = (m+i)^2 - n \quad \left(\text{da } m^2 = n \text{ ist}\right)$$

und damit für jede Primzahl p_j

$$\begin{aligned} p_j \mid z_i &\Leftrightarrow p_j \mid (i+m)^2 - n \\ &\Leftrightarrow n \equiv (i+m)^2 \mod p_j \\ &\Leftrightarrow \left(\frac{n}{p_j}\right) = 1. \end{aligned}$$

Das bedeutet, dass p_j z_i teilt, falls die Gleichung $t^2 \equiv n \mod p_j$ lösbar ist (mit $t = i + m$). Ausserdem ist $x_i = t_1 + kp_j$, da gilt

$$\begin{aligned} x_i &= t_1 + kp_j \\ \Rightarrow \quad x_i^2 &\equiv \left(t_1 + kp_j\right)^2 \mod p_j \\ &\equiv t_1^2 + 2t_1 kp_j + \left(kp_j\right)^2 \mod p_j \\ &\equiv t_1^2 \equiv t^2 \equiv n \equiv z_i \mod p_j \end{aligned}$$

Um jetzt die z_i zu finden, die durch p_j teilbar sind, müssen die t_1, t_2 berechnet werden, die $t^2 \equiv n \mod p_j$ erfüllen (→Berechnung der Quadratwurzel, Algorithmus 12 auf Seite 37). Dann kann man alle die z_i markieren, deren Index i sich von t_1 und t_2 durch ein Vielfaches von p_j unterscheiden, da diese durch p_j teilbar sind ($x_i^2 = \left(t_{1/2} + kp_j\right)^2 \equiv z_i \mod n$). Allerdings muss auch die Teilbarkeit durch Potenzen von p_j überprüft werden – dazu folgendes Lemma:

LEMMA 5.3.2. *Sei p prim, $k \geq 1$, $a \in \mathbb{Z}$, $ggT(a,p) = 1$. Die Gleichung $t^2 \equiv a \mod p^k$ ist lösbar, wenn $\left(\frac{a}{p}\right) = 1$ ist. Eine Lösung kann dann in erwarteter polynomieller Zeit gefunden werden.*

Damit ergibt sich der Algorithmus 31.

Falls in Schritt 5 die Anzahl der i mit $L[i] = 1$ zu klein ist, dann wird man in der Praxis B_2 erhöhen um genügend Gleichungen für das Lösen des Gleichungssystems in 8 zu erhalten.

In Schritt 4(a)ii werden die $L[i]$ nur durch p_j geteilt, deren Index i die Eigenschaft hat $p_j^k \mid \left(i - t_{1/2}\right)$ – auf diese Weise werden die Einträge, bei denen es sich um B-Zahlen handelt auf 1 gebracht.

Die Faktoren in Schritt 5 können auch berechnet werden, indem man sich die Faktoren in Schritt 4 merkt, was allerdings zu einem erheblich grösseren Speicherbedarf führt.

Algorithm 31 Quadratisches Sieb (QS)

EINGABE: $n \in \mathbb{N}$ ungerade, keine Primzahlpotenz.
AUSGABE: $d \in \mathbb{N}$ mit $d \neq 1, n$ und $d \mid n$.

(1) Wähle Schranken $B_1, B_2 \in \mathbb{N}$. Berechne ($\rightarrow$Sieb des Erathostenes und Alg. für das Jacobisymbol) alle Primzahlen $p_1, p_2, \ldots, p_l$ kleiner als B_1 mit $\left(\frac{n}{p_j}\right) = 1$ und $p_1 = -1$.
(2) Setze $m := \lceil \sqrt{n} \rceil$.
(3) **for** $i = 1, \ldots, B_2$ **do**
  (a) Berechne $z_i \equiv (m+i)^2 \mod n$.
  (b) Erstelle Liste L mit $L[i] := z_i$.
(4) **for** $j = 1, \ldots, l$ **do**
  (a) **for** $j = 1, \ldots, l$ **do**
    (i) Bestimme alle Lösungen t_1, t_2 von $t^2 \equiv n \mod p_j^k$.
    (ii) Bestimme in L alle Positionen i, so dass $i - t_1$ oder $i - t_2$ durch p_j^k teilbar sind und teile die entsprechenden Listeneinträge durch p_j.
(5) Bestimme die Anzahl der i mit $L[i] = 1$.
(6) **if** Anzahl$< l+1$ **then**
  (a) **return(null)**.
(7) **else**
  (a) Berechne für die ersten $l+1$ Listeneinträge $L[i]$ mit $L[i] = 1$ die Faktorisierung $\prod_{j=1}^{l} p_j^{e_{ij}}$ von z_i durch sukzessives Dividieren durch die Elemente der Faktorbasis.
(8) Löse das Gleichungssystem

$$\sum_{i=1}^{l+1} f_i e_{ij} \equiv 0 \mod 2 \quad , j = 1, \ldots, l.$$

(9) **for** Lösungen $f_1, \ldots, f_{l+1}$ **do**
  (a) Setze

$$y := \prod_{j=1}^{l} p_j^{\frac{1}{2}\sum_{i=1}^{l+1} f_i e_{ij}} \mod n \quad \text{und} \quad x := \prod_{i=1}^{l+1} x_i^{f_i} \mod n.$$

(10) Berechne $d := \mathrm{ggT}(x-y, n)$.
(11) **return(**d**)**.

Zur Erinnerung sei an dieser Stelle noch das „Sieb des Erathostenes" erklärt (Algorithmus 32).

Man kann sich gut vorstellen, dass einige der Zahlen in Algorithmus 32 mehrfach aus P entfernt werden. Diese Zahlen haben dementsprechend auch mehr Primfaktoren – woraus man folgern kann, dass diese Zahlen auch viele kleine Primfaktoren besitzen.

Wird das Sieb des Erathostenes derart eingesetzt, dass die Schleife in 2 nicht bis $\sqrt{n}$ läuft sondern nur bis y, und wir dann in jedem Schritt alle Zahlen aus P, die durch k teilbar sind eben durch k teilen, dann werden alle die Zahlen aus P, die y-glatt sind am Ende des Algorithmus zu 1. Leider findet man mit dieser Methode nicht alle y-glatten Zahlen (60 wird z.B. zu 2) – Probleme machen die höheren Potenzen von Primzahlen kleiner als y. Dieses Problem kann gelöst werden, indem auch durch Potenzen von Primzahlen dividiert wird.

Algorithm 32 Sieb des Erathostenes

EINGABE: $n \in \mathbb{N}$.
AUSGABE: $P = \{p \mid p \text{ prim und } p < n\}$.

(1) Sei $P := \{2, \ldots, n\}$.
(2) **for** $i = 2, \ldots, \sqrt{n}$ **do**
  (a) $k := i$.
  (b) **while** $k < n$ **do**
    (i) Entferne k aus P.
    (ii) $k := k + i$.
(3) **return(P).**

5.3.1. Analyse. Zu Analyse des Quadratischen Siebs benötigen wir Annahme 5.3.3.

CLAIM 5.3.3. Seien $0 < \alpha, \beta < 1$. Sei $B_2 = L_n(\alpha)$. Für ein zufälliges $1 \leq i \leq B_2$ ist mit Wahrscheinlichkeit $L_n(-1/(4\beta))$ die Zahl $z_i = (\lceil \sqrt{n} \rceil + i)^2$ $L_n(\beta)$-glatt, d.h. die z_i verhalten sich etwa wie zufällige Zahlen zwischen 1 und $\sqrt{n}$.

REMARK. Für diese Behauptung gibt es keinen Beweis, beobachtungsgemäß scheint sie aber zu stimmen.

Wenn gilt

$$B_2 = L_n(\alpha) \quad \text{und} \quad i \leq B_2,$$

dann folgt

$$m + i \quad = \quad n^{1/2+o(1)},$$

und deshalb auch

$$(m+i)^2 \mod n \quad = \quad n^{1/2+o(1)}.$$

Die Behauptung 5.3.3 kann also auch so interpretiert werden, dass die Funktion $(x+i)^2 \mod n$ die Zahlen im Intervall $[1, L_n(\alpha)]$ gleichmässig auf die Zahlen um $\sqrt{n}$ herum verteilt.

Setzt man nun $B_1 = L_n(\beta)$ und $i \leq B_2$, dann ist die Grösse l der Faktorbasis von Schritt 1 kleiner als $L_n(\beta)/2$. Mit der Behauptung ist dann die erwartete Anzahl der in Schritt 4 berechneten z_i, die sich vollständig durch die p_i der Faktorbasis faktorisieren lassen, $L_n(\alpha) L_n(-1/(4\beta))$ mit

$$\begin{aligned} L_n(\alpha) = B_2 \quad &: \quad \text{Anzahl der Zahlen, die betrachtet werden} \\ L_n(-1/(4\beta)) \quad &: \quad \text{Wahrscheinlichkeit, dass eine dieser Zahlen } B_1 - \text{glatt ist.} \end{aligned}$$

Damit das Quadratische Sieb einen Faktor von n findet, muss diese Anzahl an B_1-glatten Zahlen grösser als die Faktorbasis sein, also

$$L_n(\alpha) L_n(-1/(4\beta)) > L_n(\beta)/2.$$

Setzt man nun

$$\alpha = \beta + 1/(4\beta),$$

so ist das sogar mit hoher Wahrscheinlichkeit der Fall, weil

$$L_n(\alpha) = L_n(\beta + 1/(4\beta)) = L_n(\beta) L_n(1/(4\beta)).$$

Eingesetzt in die Gleichung oben ergibt das

$$L_n(\alpha) L_n(-1/(4\beta)) = L_n(\beta) L_n(1/(4\beta)) L_n(-1/(4\beta)) = L_n(\beta) > L_n(\beta)/2.$$

Zur Bestimmung der Laufzeit wird nun jeder einzelne Schritt betrachtet:

- Schritt 1: Die Laufzeit wird durch die Berechnung der Jacobisymbole bestimmt. Diese werden mit Algorithmus 8 in Zeit $O\left(B_1 \log^2(n)\right) = L_n(\beta + o(1))$ berechnet.
- Schritt 2: Die Quadratwurzel aus n kann mit Algorithmus 12 in Polynomialzeit berechnet werden.
- Schritt 3: Dieser Schritt benötigt Zeit $O\left(B_2 \log^2(n)\right) = L_n(\alpha + o(1))$.
- Schritt 4: Für festes k, j ist die Laufzeit zur Bestimmung der t_1, t_2 erwartet polynomiell (siehe Lemma 5.3.2). Für das Ändern der Listeneinträge wird pro Listeneintrag, der geändert wird wieder polynomielle Zeit benötigt. Insgesamt also pro j,k Zeit $O\left(\log^c(n) L_n(\alpha)/p_j^k\right)$ für eine Konstante c. Damit ist die Laufzeit dieses Schritts

$$\begin{aligned} O\left(L_n(\alpha)\log^c(n)\sum_{j,k} 1/p_j^k\right) &= O\left(L_n(\alpha)\log^c(n)\sum_{i=1}^{n} 1/i\right) \\ = O\left(L_n(\alpha)\log^{c+1}(n)\right) &= L_n(\alpha + o(1)). \end{aligned}$$

- Schritte 5 bis 7a: linear.
- Schritte 8: Zur Berechnung der Laufzeit benötigen wir den Satz 5.3.4 von Wiedemann (s.u.).
- Schritt 9: Hier dominiert die Berechnung von x und y. Dafür werden $O\left(B_1 \log(n)\right) = O(L_n(\beta)\log(n))$ vielen Multiplikationen modulo n bewältigt werden. Die Primzahlen sind alle kleiner als B_1 und die Exponenten kleiner als $\log(n)$. Damit ergibt sich eine Laufzeit von $O\left(L_n(\beta)\log^3(n)\right) = L_n(\beta + o(1))$.

PROPOSITION 5.3.4. ***(Satz von Wiedemann)*** *Sei A eine $n_1 \times n_2$-Matrix, $n_2 > n_1$, mit Einträgen 0 oder 1. Sei w die Anzahl der 1 in A.*

In Zeit $O\left(n_1(n_1+w)\log^c(n)\right)$ kann ein Vektor $x \neq 0$ mit Einträgen 0 oder 1 berechnet werden, so dass $Ax \equiv b \mod 2$ ist (das mod ist hier komponentenweise zu verstehen). Dabei ist c eine Konstante.

In Schritt 8 ist die Anzahl der Zeilen kleiner als $B_1 = L_n(\beta)$ und die Spalten entsprechen den Faktorisierungen der z_i. Da $z_i < n$ ist, kann es pro Spalte nur $\log(n)$ Einträge geben, die nicht 0 sind. Die Anzahl der Spalten ist also ebenfalls etwa $B_1 = L_n(\beta)$. Somit ist hier $w \leq L_n(\beta)\log(n)$ und die Laufzeit für Schritt 8

$$O\left(L_n(\beta) L_n(\beta) \log^{c+1}(n)\right) = L_n(2\beta + o(1)).$$

Damit ergibt sich eine Gesamtlaufzeit des QS von

$$L_n(\alpha + o(1)) + L_n(2\beta + o(1)) = L_n(\beta + 1/(4\beta) + o(1)) + L_n(2\beta + o(1))$$

(durch Setzen von $L_n(\alpha) = L_n(\beta + 1/(4\beta))$). Zum Minimieren des Ausdrucks sollte man eine Umformung vornehmen:

$$L_n(\beta + 1/(4\beta)) = L_n(2\beta) \quad \Leftrightarrow \quad \beta + 1/(4\beta) = 2\beta.$$

Das führt auf $\beta = 1/2$ und die Laufzeit des QS ist damit:

$$\begin{aligned} L_n(\beta + 1/(4\beta) + o(1)) + L_n(2\beta + o(1)) &= L_n\left(2\frac{1}{2} + o(1)\right) + L_n\left(2\frac{1}{2} + o(1)\right) \\ &= 2L_n(1 + o(1)). \end{aligned}$$

PROPOSITION 5.3.5. *Sei n ungerade und keine Primzahlpotenz. Falls Annahme 5.3.3 richtig ist, so findet das QS mit hoher Wahrscheinlichkeit einen nicht-trivialen Teiler von n in Zeit*

$$e^{(1+o(1))\sqrt{\log(n)\log\log(n)}} = L_n(1 + o(1)).$$

Wie man sieht ist die Laufzeit der EKM und des QS gleich – dies gilt jedoch nur theoretisch. In der Praxis ist die EKM viel schneller als das QS, wenn n relativ kleine Primfaktoren besitzt. Im Fall $n = pq$ mit p, q prim und etwa von der Grösse $\sqrt{n}$ ist dagegen das QS der Champ.

KAPITEL 6

Zahlkörpersieb

Bei dem Zahlkörpersieb handelt es sich um den momentan schnellsten bekannten Faktorisierungsalgorithmus. Zunächst einige Grundlagen aus der Zahlentheorie.

6.1. Grundlagen aus der Zahlentheorie

algebraisch, ganz algebraisch

DEFINITION. **(algebraisch, ganz algebraisch)** Eine Zahl $\alpha \in \mathbb{C}$ heisst *algebraisch*, wenn es ein Polynom $f(X) \in \mathbb{Q}[X]$ gibt mit $f(\alpha) = 0$. Für ein Polynom $f \in \mathbb{Q}$ mit $f(X) = \sum_{i=0}^{n} f_i X^i$, $f_n \neq 0$, heisst f_n der *führende Koeffizient* von f und n der Grad von f ($\operatorname{grad}(f) = n$).

Die Zahl α heisst *ganz algebraisch*, wenn es ein Polynom $f(X) \in \mathbb{Z}[X]$ gibt mit führendem Koeffizienten 1 und $f(\alpha) = 0$.

Ein Beispiel für eine algebraische Zahl ist $\frac{1}{\sqrt{2}}$, sie ist jedoch nicht ganz algebraisch. Dagegen ist $\sqrt{2}$ ganz algebraisch (da Nullstelle von $f(X) = x^2 - 2$). Nicht algebraische Zahlen sind z.B. π und e.

LEMMA 6.1.1. *Summe und Produkt zweier ganzer algebraischer Zahlen sind algebraisch – die ganzen algebraischen Zahlen bilden also einen Ring.*
Summe und Produkt zweier algebraischer Zahlen sind wieder algebraisch – die algebraischen Zahlen bilden also einen Körper.

BEWEIS. Dieses Lemma wird nicht bewiesen. □

In den komplexen Zahlen $\mathbb{C}$ sind nicht immer die Inversen einer ganz algebraischen Zahl wieder ganz algebraisch – das wirkt sich unmittelbar auf das Produkt zweier ganz algebraischer Zahlen aus. Die ganz algebraischen Zahlen verhalten sich in etwa wie die ganzen Zahlen $\mathbb{Z}$. In $\mathbb{Q}$ besteht sogar ein direkter Zusammenhang:

LEMMA 6.1.2. *Eine Zahl $\alpha \in \mathbb{Q}$ ist ganz algebraisch genau dann, wenn $\alpha \in \mathbb{Z}$.*

BEWEIS. „⇐": Offensichtlich sind alle Zahlen aus $\mathbb{Z}$ ganz algebraisch ($z \in \mathbb{Z} : f(X) = X - z$).

„⇒": Sei $\alpha = p/q$ ganz algebraisch mit $p, q \in \mathbb{Z}$ und $\operatorname{ggT}(p,q) = 1$, dann existiert ein Polynom $f(X) = \sum_{i=0}^{n} f_i X^i$ mit $f_n = 1$ und $f(\alpha) = 0$ und damit

$$\frac{p^n}{q^n} = -\sum_{i=0}^{n-1} f_i \left(\frac{p}{q}\right)^i \quad \Leftrightarrow \quad p^n = -\sum_{i=0}^{n-1} f_i p^i q^{n-i}.$$

Daraus folgt aber, dass p^n von q geteilt wird – Widerspruch zur Annahme! □

In den rationalen Zahlen können die ganzen algebraischen Zahlen also auch einfach nur „ganz" genannt werden – in diesem Fall bezeichnen die beiden Begriffe das gleiche.

Wie man nun einfach einsehen kann, ist also $\sqrt{1/2}$ nicht algebraisch – denn dann wäre nach Lemma 6.1.1 auch $1/2 = \left(\sqrt{1/2}\right)^2$ algebraisch, was ein Widerspruch zu Lemma 6.1.2 darstellen würde.

Minimalpolynom

DEFINITION. **(Minimalpolynom)** Sei $\alpha \in \mathbb{C}$ algebraisch. Das Polynom $f(X) \in \mathbb{Q}[X]$ kleinsten Grades mit führendem Koeffizienten 1 und mit $f(\alpha) = 0$ heisst das *Minimalpolynom* von α $f = \mathrm{MP}(\alpha)$. Der Grad des Minimalpolynoms von α heisst der Grad von α, nämlich $\mathrm{grad}(\alpha)$.

Gauß'sches Lemma

LEMMA 6.1.3. ***(Gauß'sches Lemma)*** *Das Minimalpolynom f einer ganzen algebraischen Zahl α ist $f(X) \in \mathbb{Z}[X]$.*
Ist g ein Polynom in $\mathbb{Q}[X]$ mit $g(\alpha) = 0$, dann teilt f das Polynom g in $\mathbb{Q}[X]$.
Das Minimalpolynom ist also ein irreduzibles Polynom in $\mathbb{Q}[X]$, das nur von dem konstanten Polynom $q \in \mathbb{Q}$ geteilt wird.

Aus dem ersten Teil folgt, dass man sich nur das Minimalpolynom einer algebraischen Zahl ansehen muss, um entscheiden zu können, ob diese ganz algebraisch ist.

Algebraischer Zahlkörper

DEFINITION. **(Algebraischer Zahlkörper)** Sei $\alpha \in \mathbb{C}$ algebraisch mit Minimalpolynom $f(X) = \sum_{i=0}^{n} f_i X^i$. Die Menge

$$\mathbb{Q}(\alpha) = \left\{ \gamma \in \mathbb{C} \mid \gamma = \sum_{i=0}^{n-1} c_i \alpha^i, c_i \in \mathbb{Q} \right\}$$

mit der Addition und der Multiplikation in $\mathbb{C}$ heisst „der von α erzeugte *algebraische Zahlkörper*".

LEMMA 6.1.4. *Der algebraische Zahlkörper ist ein Körper.*

BEWEIS. Damit $\mathbb{Q}(\alpha)$ ein Körper ist müssen die Körperaxiome gelten. Bezüglich der Addition ist $\mathbb{Q}(\alpha)$ auf jeden Fall abgeschlossen. Diese Abgeschlossenheit gilt auch für die Multiplikation, da man $\alpha^n := -\sum_{i=0}^{n-1} f_i \alpha^i$ setzen kann und damit die hohen Potenzen wieder „klein kriegt". Da die Addition und die Multiplikation in $\mathbb{C}$ abläuft, gelten auch das Distributiv-, das Kommutativ- und das Assoziativgesetz. Bleibt die Inversenbildung: Ist das Inverse von $\gamma \in \mathbb{Q}(\alpha)$, gebildet in $\mathbb{C}$ wieder in $\mathbb{Q}(\alpha)$? Sei dazu $\gamma = \sum_{i=0}^{n-1} c_i \alpha^i$ mit $c_i \in \mathbb{Q}$. Sei $g(X) = \sum_{i=0}^{n-1} c_i X^i$. Da $f(X) = \sum_{i=0}^{n} f_i X^i$ irreduzibel ist und der Grad von g kleiner als der von f ist, gilt $\mathrm{ggT}(f,g) = 1$ (der ggT wird in $\mathbb{Q}[X]$ gebildet). Jetzt kann wie bei den ganzen Zahlen gezeigt werden, dass es Polynome $u, v \in \mathbb{Q}[X]$ gibt mit

$$1 = \mathrm{ggT}(f,g) = u(X) f(X) + v(X) g(X).$$

Setzt man nun α ein, so erhält man

$$1 = v(\alpha) g(\alpha).$$

Da v nur rationale Koeffizienten besitzt, ist $v \in \mathbb{Q}(\alpha)$, und damit das Inverse von g. □

Im Folgenden können wir uns bei algebraischen Zahlkörpern auf diejenigen beschränken, die durch ganze algebraische Zahlen erzeugt werden.

LEMMA 6.1.5. *Sei $K = \mathbb{Q}(\alpha)$ ein algebraischer Zahlkörper. Dann existiert eine ganze algebraische Zahl γ mit $\mathbb{Q}(\gamma) = \mathbb{Q}(\alpha)$.*

BEWEIS. Sei $f(X) = \sum_{i=0}^{n} f_i X^i$ mit $f_i \in \mathbb{Q}$ das Minimalpolynom von α. Sei weiterhin q das kleinste gemeinsame Vielfache der Nenner der Koeffizienten f_i. Dann ist

$$q^n f(X) = \sum_{i=0}^{n} q^n f_i X^i = \sum_{i=0}^{n} q^{n-i} f_i (qX)^i \in \mathbb{Z}[X].$$

Sei nun $g(X) = \sum_{i=0}^{n} q^{n-i} f_i X^i$. Diese Polynom aus $\mathbb{Z}[X]$ mit führendem Koeffizienten 1

$$i = n: \quad q^{n-n} f_n = f_n = 1$$

und der Nullstelle αq

$$g(\alpha q) = \sum_{i=0}^{n} q^{n-i} f_i (\alpha q)^i = \sum_{i=0}^{n} q^n f_i \alpha^i = q^n f(\alpha) = 0$$

ist dann das Minimalpolynom zu der ganz algebraischen Zahl $\gamma = \alpha q$. Ausserdem ist $q\alpha \in \mathbb{Q}(\alpha)$ und $\alpha \in \mathbb{Q}(\gamma)$ und damit $\mathbb{Q}(\alpha) = \mathbb{Q}(\gamma)$. □

Wegen dieses Lemmas werden wir in Zukunft immer annehmen, dass algebraische Körper von ganzen algebraischen Zahlen erzeugt werden.

$K = Q(\alpha)$

DEFINITION. Sei $K = \mathbb{Q}(\alpha)$ ein algebraischer Zahlkörper. Mit $\mathbb{Z}_K$ bezeichnen wir **die Menge der in K enthaltenen ganzen algebraischen Zahlen**.

LEMMA 6.1.6. *$\mathbb{Z}_K$ ist ein Ring und wird als Ring der ganzen Zahlen in K bezeichnet.*

Nach diesem Lemma gilt für jeden von einer ganzen algebraischen Zahl α erzeugten Körper $K = \mathbb{Q}(\alpha)$, dass

$$\mathbb{Z}[\alpha] = \left\{ \sum_{i=0}^{n-1} c_i \alpha^i \mid c_i \in \mathbb{Z} \right\} \subseteq \mathbb{Z}_K.$$

Leider gilt nicht immer eine Gleichheit der beiden Körper:

PROPOSITION 6.1.7. *Sei $m \in \mathbb{N}$ mit $p^2 \nmid n$ für alle p prim. Sei $K = \mathbb{Q}(\sqrt{m})$, dann ist*

$$\mathbb{Z}_K = \mathbb{Z}[\sqrt{m}] \quad , \quad \text{falls } m \equiv 2,3 \mod 4$$
$$\mathbb{Z}_K = \{a/2 + b/2\sqrt{m} \mid a,b \in \mathbb{Z}, a \equiv b \mod 2\} \quad , \quad \text{falls } m \equiv 1 \mod 4.$$

6.2. Einleitung zum Zahlkörpersieb

Sei $n \in \mathbb{N}$ die Zahl, die wir faktorisieren möchten, dann versuchen wir ein Polynom $f \in \mathbb{Z}[X]$ und ein $m \in \mathbb{N}$ zu finden mit f irreduzibel und führendem Koeffizienten 1 und ausserdem $f(m) \equiv 0 \mod n$. Jetzt untersuchen wir den algebraischen Zahlkörper $K = \mathbb{Q}(\alpha)$, mit α Nullstelle von f und nehmen an, dass $\mathbb{Z}_K = \mathbb{Z}[\alpha]$ ist. Die folgende Abbildung φ ist dann ein Ringhomomorphismus:

Finde $f \in \mathbb{Z}[X]$ und $m \in \mathbb{N}$ mit f irreduzibel und führendem Koeffizienten 1

$$\begin{array}{rccc} \varphi: & \mathbb{Z}[\alpha] & \longrightarrow & \mathbb{Z}_n \\ & \sum c_i \alpha^i & \longmapsto & \sum c_i m^i. \end{array}$$

α Nullstelle von f und $K = \mathbb{Q}(\alpha)$ alg. Zahlkörper

Angenommen, es gibt in $\mathbb{Z}[\alpha]$ „kleine Primfaktoren" $\pi_1, \ldots, \pi_k$, so dass sich viele Zahlen der Form $a + b\alpha$ mit $a,b \in \mathbb{Z}$ durch diese π_j faktorisieren lassen. Parallel betrachten wir eine Menge normaler Primzahlen in $\mathbb{Z}$. Können wir nun $a_i + b_i\alpha$ in $\mathbb{Z}[\alpha]$ mit Hilfe der π_j und die Zahl $a_i + b_i m$ in $\mathbb{Z}$ mit Hilfe der p_j faktorisieren, für $a_i, b_i \in \mathbb{Z}$, so erhalten wir

Ann.: $\mathbb{Z}[\alpha]$ besitzt viele kleine Primfaktoren π_i

$$\varphi(a_i + b_i\alpha) \equiv \prod \varphi\left(\pi_j\right)^{d_{ij}} \equiv a_i + b_i m \equiv \prod p_j^{e_{ij}} \mod n.$$

Suche parallel zu π_i Primzahlen p_i in $\mathbb{Z}$, die $a_i + b_i m$ faktorisieren

Hat man $k+l+1$ viele Paare $(a_1,b_1),\ldots,(a_{l+k+1},b_{l+k+1})$ mit obigen Faktorisierungen gefunden, so erhalten wir das folgende Gleichungssystem:

$$\begin{aligned}\sum_{i=1}^{l+k+1} f_i d_{ij} &\equiv 0 \mod 2,\ j=1,\ldots,k\\ \sum_{i=1}^{l+k+1} f_i e_{ij} &\equiv 0 \mod 2,\ j=1,\ldots,l.\end{aligned}$$

Dafür können wir eine Lösung $(f_1,\ldots,f_{k+l+1}) \neq (0,\ldots,0)$ finden, so dass gilt

$$\begin{aligned}x &= \prod_{j=1}^{l} \varphi\left(\pi_j\right)^{1/2\sum_{i=1}^{l+k+1} f_i d_{ij}}\\ y &= \prod_{j=1}^{l} p_j^{1/2\sum_{i=1}^{l+k+1} f_i e_{ij}}\end{aligned}$$

und $x^2 \equiv y^2 \mod n$. Wie beim Quadratischen Sieb sollte jedoch $x \neq y \mod n$ sein, was sehr wahrscheinlich gegeben ist, da die Faktorisierung in unterschiedlichen Ringen erzeugt wurde. Als letzter Schritt sollte dann $\mathrm{ggT}(x-y,n)$ einen nicht-trivialen Teiler von n liefern.

In dieser groben Skizzierung des Algorithmus sind viele Aussagen nicht näher spezifiziert (z.B. wie man die kleinen Primzahlen findet, und was das in $\mathbb{Z}[\alpha]$ überhaupt ist usw.). In den folgenden Abschnitten soll auf diese Fragen Schritt für Schritt näher eingegangen werden.

6.3. Finden der irreduziblen f

Zu Anfang des Algorithmus soll zu n ein Polynom $f \in \mathbb{Z}[X]$ und ein $m \in \mathbb{N}$ gefunden werden, das irreduzibel ist, einen führenden Koeffizienten 1 hat und $f(m) \equiv 0 \mod n$ ist. Wie kann ein solches f gefunden werden?

Seien $n \in \mathbb{N}$ mit $n \neq p^k$ mit p prim und eine Zahl $d \in \mathbb{N}$, $d \geq 2$ gegeben. Wir suchen nun ein irreduzibles Polynom f mit $\mathrm{grad}(f) = d$. Ausserdem wollen wir eine Zahl $m \in \mathbb{N}$ konstruieren, so dass $f(m) \equiv 0 \mod n$ ist. Dazu setzen wir $m = \lfloor n^{1/d} \rfloor$. Sei dann $n = \sum_{i=0}^{\infty} f_i m^i$ die m-äre Darstellung von n. Dann gilt:

LEMMA 6.3.1. *Ist $n > (4d)^d$, so ist $f_i = 0$ für alle $i \geq d+1$ und $f_d = 1$.*

BEWEIS. Wir zeigen, dass $n/2 < m^d$ – daraus folgt dann das Lemma. Da $m \geq n^{1/d} - 1$ ist, genügt es zu zeigen, dass gilt

$$n/2 \leq \left(n^{1/d} - 1\right)^d.$$

Dieses ist äquivalent zu (binomischer Lehrsatz)

$$n/2 \leq \sum_{i=0}^{d} \binom{d}{i} (-1)^i n^{(d-i)/d} = n - n\sum_{i=1}^{d} \binom{d}{i} (-1)^{i-1} n^{-i/d}.$$

Wegen

$$n\sum_{i=1}^{d} \binom{d}{i} (-1)^{i-1} n^{-i/d} \leq n\sum_{i=1}^{d} \left(d/n^{1/d}\right)^i$$

bleibt zu zeigen, dass gilt

$$\sum_{i=1}^{d} \left(d/n^{1/d} \right)^i \leq 1/2.$$

Nach der Voraussetzung ist $n > (4d)^d$ und damit

$$\frac{d}{n^{1/d}} < \frac{d}{(4d)^d} = \frac{1}{4^d d^{d-1}} \leq 1/4,$$

womit die Aussage bewiesen wäre. □

Sei also $f(x) = \sum_{i=0}^{d} f_i X^i$, dann hat dieses Polynom nach dem Lemma einen führenden Koeffizienten 1 und es gilt $f(m) \equiv 0 \mod n$. Um das Zahlkörpersieb anwenden zu können, sollte f aber auch irreduzibel sein, was nicht zwingend aus dem Lemma folgt. Es kann allerdings gezeigt werden, dass wenn f reduzibel ist, also $f = gh$ gilt, dass dann $g(m)$ oder $h(m)$ ein echter Teiler von n ist.

REMARK. Über den ganzen Zahlen $\mathbb{Z}$ können Polynome in polynomieller Zeit faktorisiert werden. Ist f also nicht irreduzibel, so kann in polynomieller Zeit ein echter Teiler von n gefunden werden.

Auf die Bedingung $n \geq (4d)^d$ wird bei der Analyse des Zahlkörpersiebs noch genauer eingegangen werden. Wie man sehen wird, sollte d sogar noch um einiges kleiner gewählt werden, als es die Bedinung verlangt.

In der Praxis hängt die Effizienz des Zahlkörpersiebs sehr stark von der Größe der Koeffizienten des Polynoms f ab. Für speziell geartete n können sehr gute Polynome f algorithmisch bestimmt werden, worauf im Folgenden eingegangen werden soll.

6.3.1. Bestimmung von f für $n = r^e - s$. (Das spezielle Zahlkörpersieb) Sei n von der Form $n = r^e - s$ mit $e \in \mathbb{N}$ und $s \in \mathbb{Z}$, $s \neq 0$. Seien außerdem r und $|s|$ „klein" und e „groß" und $\mathrm{ggT}(r,n) = 1$. Beispiele für Zahlen dieser Art sind die Fermat-Zahlen $2^{2^k} + 1$ mit $k \in \mathbb{N}$.

Für eine gegebene Gradschranke $d \in \mathbb{N}$ seien

$$\begin{aligned} k &= \min\{kd \geq e \mid k \in \mathbb{N}\} \\ t &= sr^{kd-e} \\ f(X) &= X^d - t \\ m &= r^k. \end{aligned}$$

Jetzt ist zu zeigen, dass gilt $n \mid f(m) = r^{kd} - sr^{kd-e}$. Wegen $\mathrm{ggT}(r,n) = 1$ genügt es also zu zeigen, dass gilt $n \mid r^e - s = n$ – was trivialerweise richtig ist. Damit erfüllen m und f die Bedingung $f(m) \equiv 0 \mod n$ und der führende Koeffizient von f ist 1. Ist f aber auch irreduzibel? Im allgemeinen ist dies sicherlich nicht der Fall – für bestimmte Variationen von d sind die konstruierten Polynome aber i.a. irreduzibel.

Für Zahlen der Form $n = r^e - s$ können also spezielle Methoden angewandt werden, so dass das Zahlkörpersieb noch effizienter arbeitet. Dieser Algorithmus wird das **spezielle Zahlkörpersieb** genannt.

6.4. Glattheit algebraischer Zahlen

Nachdem der erste Schritt des Algorithmus geklärt ist, fahren wir nun mit dem Finden der „kleinen" Primzahlen $\pi_j \in \mathbb{Z}[\alpha]$ fort. Dafür soll zunächst der Glattheitsbegriff, der schon von der EKM bekannt ist, auf die algebraischen Zahlen ausgedehnt werden.

Sei α ein algebraische Zahl, deren Minimalpolynom $f(X) = \sum_{i=0}^{n} f_i X^i$ ist. Seien $\alpha_0 = \alpha, \alpha_1, \ldots, \alpha_{n-1}$ Nullstellen diese Polynoms in den komplexen Zahlen $\mathbb{C}$. Die Zahlen α_i heissen dann **die zu α konjugierte Zahlen**. Sei nun

$$\begin{aligned} \gamma &= \sum_{i=0}^{n-1} c_i \alpha^i \in \mathbb{C}(\alpha) \\ \text{Norm}: \quad N(\gamma) &= \prod_{j=0}^{n-1} \left(\sum_{i=0}^{n-1} c_i \alpha_j^i \right). \end{aligned}$$

In speziellen Fällen ist die Norm dann

LEMMA 6.4.1.

$$\begin{aligned} N(\alpha) &= (-1)^n f_0. \\ N(\gamma_1\gamma_2) &= N(\gamma_1) N(\gamma_2) \quad , \textit{fuer alle } \gamma_1, \gamma_2 \in \mathbb{Q}(\alpha). \\ N(\gamma) &\in \mathbb{Z} \quad , \textit{fuer } \gamma \in \mathbb{Q}(\alpha) \textit{ ganz algebraisch.} \end{aligned}$$

Mit der dritten Zeile des Lemmas können wir nun die Glattheit einer algebraischen Zahl definieren.

B-glatt

DEFINITION. **(*B*-glatt)** Eine ganze algebraische Zahl in $\mathbb{Q}(\alpha)$ heisst B-glatt, falls $N(\gamma)$ B-glatt ist in $\mathbb{Z}$, d.h. nur Primteiler $\leq B$ besitzt.

Über die Norm wird also die Glattheit in $\mathbb{Q}(\alpha)$ mit der Glattheit in $\mathbb{Z}$ definiert.

6.5. Irreduzible Elemente und Einheiten

Wie wir oben bereits gesehen haben, werden wir beim Zahlkörpersieb-Algorithmus Elemente eines Rings von ganzen Zahlen in ihre äquivalente Primzahlen in diesen Ringen zerlegen müssen.

Irreduzible Elemente

DEFINITION. **(irreduzible Elemente)** Die zu den Primzahlen in den Ringen äquivalenten Zahlen heissen *irreduzible Elemente*.

Genauso arbeiten wir mit den Inversen von Zahlen.

Einheit

DEFINITION. **(Einheit)** Sei $K = \mathbb{Q}(\alpha)$ ein algebraischer Zahlkörper mit dem Ring der ganzen Zahlen $\mathbb{Z}_K$. Ein Element $\varepsilon \in \mathbb{Z}_K$ heisst *Einheit*, wenn sein Inverses ε^{-1} ebenfalls in $\mathbb{Z}_k$ liegt.

Wie man leicht einsieht stellt die Menge der Einheiten bzgl. der Multiplikation eine Gruppe dar. Die Einheiten dieser Gruppe sind ± 1. Im allgemeinen sind die Einheiten aller $\mathbb{Z}_K$ immer die Elemente mit Norm ± 1:

LEMMA 6.5.1. *Sei $K = \mathbb{Q}(\alpha)$ ein algebraischer Zahlkörper mit Ring der ganzen Zahlen $\mathbb{Z}_K$. Ein Element $\varepsilon \in \mathbb{Z}_K$ ist genau dann eine Einheit, wenn $N(\varepsilon) = \pm 1$ ist.*

Wie man schon in $\mathbb{Z}$ sieht, werden alle Zahlen durch die Einheiten geteilt. So gilt immer $\varepsilon\varepsilon^{-1}\gamma = \gamma$ – jedes $\gamma \in \mathbb{Z}_K$ wird also von der Einheit ε geteilt. Es gilt sogar $\varepsilon\varepsilon' = 1$, nämlich mit $\varepsilon' = \varepsilon^{-1}$. Bei der Untersuchung von Teilbarkeitsrelationen sollen diese Einheiten eine Sonderbehandlung erfahren, denn wie man schon in $\mathbb{Z}$ sieht ist die Primfaktorzerlegung nur *bis auf Einheiten* eindeutig. In beliebigen Ringen kann die Situation etwas komplizierter werden, da es hier sogar unendlich viele Einheiten geben kann. Zur Struktur der Gruppe der Einheiten in einem Ring von ganzen Zahlen gilt:

PROPOSITION 6.5.2. *Sei $K = \mathbb{Q}(\alpha)$ ein algebraischer Körper mit Ring der ganzen Zahlen $\mathbb{Z}_K$. Das Minimalpolynom von α habe r reelle und $2s$ nichtreelle Nullstellen in $\mathbb{C}$. Dann gibt es ein $m \in \mathbb{N}$, eine Einheit $\xi \in \mathbb{Z}_K$ mit $\xi^m = 1$ und $r+s-1$ weitere Einheiten $\varepsilon_1, \ldots, \varepsilon_{r+s-1}$, so dass sich jede Einheit $\varepsilon \in \mathbb{Z}_K$ eindeutig darstellen lässt als*

$$\varepsilon = \xi \prod_{i=1}^{r+s+1} \varepsilon_i^{e_i} \quad , 0 \leq k \leq m-1, e_i \in \mathbb{Z}.$$

DEFINITION. **(Fundamentaleinheiten)** Die ε_i werden die *Fundamentaleinheiten* genannt.

Fundamentaleinheiten

Die Eindeutigkeitsaussage des Satzes impliziert, dass m die kleinste Zahl mit $\xi^m = 1$ ist, und dass für keine der Einheiten ε_i eine natürliche Zahl l existiert mit $\varepsilon_i^l = 1$.

DEFINITION. **(Irreduzibles Element)** Sei $K = \mathbb{Q}(\alpha)$ ein algebraischer Zahlkörper mit Ring der ganzen Zahlen $\mathbb{Z}_K$. Ein Element $\pi \in \mathbb{Z}_K$ heisst *irreduzibel*, falls aus $\pi = \gamma_1\gamma_2$, $\gamma_i \in \mathbb{Z}_K$ folgt, dass γ_1 oder γ_2 Einheit ist $\mathbb{Z}_K$ ist.

Irreduzibles Element

Mit dem folgenden Lemma kann man zeigen, dass eine Zahl irreduzibel ist:

LEMMA 6.5.3. *Sei $\pi \in \mathbb{Z}_K$. Falls $N(\pi)$ prim ist, dann ist π irreduzibel.*

BEWEIS. Wenn $\pi = \gamma_1\gamma_2$ ist, dann folgt $N(\pi) = N(\gamma_1)N(\gamma_2)$. Da aber $N(\pi)$ prim ist, muss $N(\gamma_1) = \pm 1$ oder $N(\gamma_2) = \pm 1$ sein – also ist γ_1 oder γ_2 eine Einheit. □

Die Umkehrung des Lemmas gilt nicht unbedingt – es gibt also irreduzible Elemente, deren Norm nicht prim ist (z.B. 3 in $\mathbb{Z}\left[\sqrt{-1}\right]$).

Wenn jetzt $K = \mathbb{Q}(\alpha)$ ein algebraischer Zahlkörper mit Ring der ganzen Zahlen $\mathbb{Z}_K$ ist – gibt es dann für jedes Element $\gamma \in \mathbb{Z}_K$ eine bis auf Einheiten eindeutige Zerlegung in irreduzible Elemente?

Sind also

$$\gamma = \varepsilon \prod_{i=1}^{k} \pi_i^{e_{is}} \quad \varepsilon \text{ Einheit, } \pi_i \text{ irreduzibel, } e_{is} \in \mathbb{N}$$

$$\gamma = \varepsilon' \prod_{j=1}^{l} \rho_j^{e_{js}} \quad \varepsilon' \text{ Einheit, } \rho_j \text{ irreduzibel, } e_{js} \in \mathbb{N}.$$

zwei Zerlegungen von γ – gilt dann, dass $k = l$ ist und für alle π_i ein ρ_j und eine Einheit ε_i existiert mit $\pi_i = \varepsilon_i\rho_j$?

Natürlich kann jedes Element γ in Einheiten und irreduzible Element zerlegt werden – nur, ist diese Zerlegung auch eindeutig? Als Beispiel für die Nichteindeutigkeit wird in [**Blömer**, Beispiel 12.29] der Fall $K = \mathbb{Q}\left(\sqrt{-5}\right)$ angeführt. In diesem Körper ist $6 = 2 \cdot 3 = \left(1+\sqrt{-5}\right)\left(1-\sqrt{-5}\right)$ und $2, 3, 1+\sqrt{-5}$ und $1-\sqrt{-5}$ irreduzibel und keine Einheit. Die Zerlegung ist also nicht eindeutig.

Um grössere Umstände beim Zahlkörpersieb zu umgehen, nehmen wir an, dass in unseren Fällen stets eine eindeutige Zerlegung aller Zahlen in Einheiten und irreduzible Elemente möglich ist:

CLAIM 6.5.4. Für den Rest des Kapitels sei angenommen, dass für einen algebraischen Zahlkörper $K = \mathbb{Q}(\alpha)$ das erzeugende Element α eine ganze Zahl ist, dass $\mathbb{Z}_K = \mathbb{Z}[\alpha]$, und dass in $\mathbb{Z}_K$ jedes Element eindeutig (bis auf Einheiten) in irreduzible Elemente zerlegt werden kann.

6.6. Faktorisieren von $a + b\alpha$

Zurück zum Algorithmus: hier sollen nach dem Finden der π_i und der p_j Zahlen der Form $a_i + b_i\alpha$ in $\mathbb{Z}[\alpha]$ und $a_i + b_i m$ in $\mathbb{Z}$ faktorisiert werden ($a_i, b_i \in \mathbb{Z}$).

Seien nun die $\{\pi_1, \ldots, \pi_k\}$ die irreduziblen Elemente, deren Norm kleiner als eine Schranke B ist – die $a_i + b_i\alpha$, bei denen die Faktorisierung erfolgreich ist, sind also B-glatt nach der Definition auf Seite 106. Aber wie kann effizient festgestellt werden, ob eine Zahl durch ein irreduzibles Element π_j teilbar ist? Dazu benötigen wir zunächst den Begriff des Ideals:

Ideal

DEFINITION. Eine Teilmenge $I \subset \mathbb{Z}_K$ heisst **Ideal**, falls gilt

$$\begin{aligned} \gamma, \beta \in I &\Rightarrow \gamma + \beta \in I \\ \gamma \in I, \beta \in \mathbb{Z}_K &\Rightarrow \beta\gamma \in I. \end{aligned}$$

Ein ganz spezielles Ideal ist das Hauptideal.

Hauptideal

DEFINITION. Sei $\gamma \in \mathbb{Z}_K$, dann ist $(\gamma) = \{\beta\gamma \mid \beta \in \mathbb{Z}_K\}$ das von γ erzeugte **Hauptideal**. Erzeugen zwei Elemente γ_1 und γ_2 dasselbe Ideal, so unterscheiden sie sich nur durch Einheiten: $\gamma_1 = \varepsilon\gamma_2$ mit ε Einheit in $\mathbb{Z}_K$.

LEMMA 6.6.1. *In $\mathbb{Z}$ sind alle Ideale Hauptideale.*

BEWEIS. Sei I ein Ideal in $\mathbb{Z}$ und g das betragsmässig kleinste Element in I. Dann können wir annehmen, dass g positiv ist. Jetzt zeigen wir $(g) = I$:
Angenommen, es gilt $I \neq (g)$, dann gibt es ein Element $a \in I$ mit $a \notin I$, also $a \neq \beta g$ mit $\beta \notin \mathbb{Z}_K$. Also kann a geschrieben werden als $a = qg + r$ mit $0 < g < r$. Da $a, g \in I$, muss auch $r \in I$ sein – Widerspruch zu der Voraussetzung, dass g das kleinste positive Element in I ist. □

In den algebraischen Körpern, für die Annahme 6.5.4 gilt, gilt dann auch

LEMMA 6.6.2. *Genügt der Zahlkörper K der Annahme 6.5.4, so ist jedes Ideal in $\mathbb{Z}_K$ ein Hauptideal.*

BEWEIS. Ohne Beweis. □

Für zwei Elemente $\gamma, \beta \in \mathbb{Z}_K$ teilt γ das Element β genau dann, wenn $\beta \in (\gamma)$ – ein wichtiger Zusammenhang zwischen Idealen und der Teilbarkeitsrelation. Für uns werden im Folgenden vor allem Ideale von Interesse sein, die von irreduziblen Elementen erzeugt werden.

Primideal

DEFINITION. Ein Ideal $I \subset \mathbb{Z}_K$ heisst **Primideal**, wenn für beliebige Elemente $\gamma_1, \gamma_2 \in \mathbb{Z}_K$ aus $\gamma_1\gamma_2 \in I$ und $\gamma_1 \notin I$ immer $\gamma_2 \in I$ folgt.

LEMMA 6.6.3. *Genügt der Zahlkörper K der Annahme 6.5.4, dann ist ein Ideal $I = (\pi)$ genau dann ein Primideal in $\mathbb{Z}_K$, wenn π ein irreduzibles Element in $\mathbb{Z}_K$ ist.*

BEWEIS. „⇒": Angenommen, π ist nicht irreduzibel, dann gibt es irreduzible Element $\pi_1, \ldots, \pi_k$ mit $k \geq 2$ und eine Einheit ε, so dass sich π schreiben lässt als $\pi = \varepsilon \prod_{i=1}^{k} \pi_i$. Vor allem gilt dann auch $\prod_{i=1}^{k} \pi_i \in (\pi)$ und $\pi_i \notin (\pi)$, $i = 1, \ldots, k$, da das andernfalls bedeuten würde, dass die irreduzible π_i von π geteilt werden. Dann kann (π) aber auch kein Primideal sein!

„⇐": Angenommen, (π) ist kein Primideal, dann gibt es $\gamma_1, \gamma_2 \notin (\pi)$, so dass $\gamma_1\gamma_2 \in (\pi)$ gilt. Das bedeutet $\pi \mid \gamma_1\gamma_2$ aber $\pi \nmid \gamma_1$ und $\pi \nmid \gamma_2$. In diesem Fall gilt

$$\gamma_1\gamma_2 = k\pi \quad k \in \mathbb{Z} \quad \Rightarrow \quad \mathrm{ggT}\left(\gamma_1\gamma_2, \pi\right) > 1.$$

Also kann π aber nicht irreduzibel sein. □

Die Primideale in $\mathbb{Z}$ sind also genau die von Primzahlen erzeugten Hauptideale.

Sei nun $I = (\pi)$ Primideal in $\mathbb{Z}_K$. Dann ist $I \cap \mathbb{Z}$ ein Primideal in $\mathbb{Z}$ – allerdings ist $I \cap \mathbb{Z} \neq \mathbb{Z}$, was allerding etwas schwerer zu zeigen ist. Also ist $I \cap \mathbb{Z} = (p)$, mit p prim. Wir nennen diesen Sachverhalt *„π **liegt über** p"* oder *„p **liegt unter** π"*. Zu jedem $I = (\pi)$ gibt es also eine Primzahl $p \in \mathbb{Z}$, so dass π über p liegt. In diesem Fall gilt $p \in (\pi)$, man kann p also auch schreiben als $p = \gamma\pi$ mit $\gamma \in \mathbb{Z}_K$. Genauso wird p von den irreduziblen Elementen geteilt, die in $\mathbb{Z}_K$ ein Primideal erzeugen, dass über p liegt. Es handelt sich hier also um eine 1-zu-1-Abbildung.

Desweiteren gilt $N(p) = p^n$, mit $n = \mathrm{grad}(\alpha)$ und $K = \mathbb{Q}(\alpha)$. Ausserdem gilt $p^n = N(p) = N(\gamma)\,N(\pi)$ und demnach auch $N(\pi) = p^f$ mit $f \in \mathbb{N}$. Der folgende Satz zeigt weiter Zusammenhänge zwischen den Primzahlen in $\mathbb{Z}$ und den Primidealen in $\mathbb{Z}_K$ auf.

PROPOSITION 6.6.4. *Sei $K = \mathbb{Q}(\alpha)$ ein Zahlkörper, der Annahme 6.5.4 genügt. Sei f das Minimalpolynom von α und $p \in \mathbb{Z}$ prim. Es sei*

$$f(X) = \prod_{i=1}^{k} f_i^{e_i}(X) \mod p_i \quad f_i \text{ seien irreduzible Polynome in } \mathbb{Z}_p[X].$$

Dann gilt

(1) $(p, f_i(\alpha))$ *ist ein Primideal von* $\mathbb{Z}_K$.
(2) *Ist* $(p, f_i(\alpha)) = (\pi_i)$ *für irreduzible Elemente* π_i, *dann liegt* π *über* p *und* π^{e_i} *ist die höchste Potenz von* π_i, *die* p *noch teilt.*
(3) $N(\pi_i) = p^{grad(f_i)}$.
(4) *Es gibt eine Einheit* $\varepsilon \in \mathbb{Z}_K$ *mit* $p = \varepsilon \prod_{i=1}^{k} \pi_i^{e_i}$. *Alle anderen irreduziblen Elemente, die über* p *liegen, unterscheiden sich durch eine Einheit von einem der* π_i *mit* $i = 1, \ldots, k$.

Die Aussage (4) ergibt sich schon aus der Tatsache, dass zwei Elemente, die sich nur durch eine Einheit unterscheiden, dasselbe Hauptideal erzeugen.

Interessant ist für uns der Satz vor allem deshalb, weil es effiziente Algorithmen zur Faktorisierung von Polynomen in $\mathbb{Z}_p[X]$ gibt – es wird jedoch nichts darüber ausgesagt, wie die π_i gefunden werden können.

Im Zahlkörpersieb möchten wir Zahlen der Form $a+b\alpha$ faktorisieren. Dazu soll folgende Notation eingeführt werden: sind $\gamma_1, \gamma_2 \in \mathbb{Z}_K$ mit $\gamma_1 - \gamma_2 \in (\pi)$, dann schreiben wir $\gamma_1 \equiv \gamma_2 \mod \pi$. Dabei handelt es sich um eine echte Äquivalenzrelation und die Eigenschaften sind die gleichen wie in den ganzen Zahlen $\mathbb{Z}$ und es gilt

$$\gamma_1 \equiv \gamma_2 \mod \pi \quad \Leftrightarrow \quad \pi \mid \gamma_2 - \gamma_1.$$

PROPOSITION 6.6.5. *Sei $K = \mathbb{Q}(\alpha)$ ein Zahlkörper, der Annahme 6.5.4 genügt und f das Minimalpolynom von α. Schliesslich seien $a, b \in \mathbb{Z}$ mit $ggT(a,b) = 1$ und $p \in \mathbb{Z}$ prim.*
Es exisitiert genau dann ein irreduzibles Element $\pi \in \mathbb{Z}_K$ mit $\pi \mid a+b\alpha$, das über p liegt (also $(\pi) \cap \mathbb{Z} = (p)$), wenn ein $c_p \in \mathbb{Z}$ existiert mit $f(c_p) \equiv 0 \mod p$ und $a + bc_p \equiv 0 \mod p$.

BEWEIS. Zunächst soll die Notwendigkeit der Bedingungen gezeigt werden. Angenommen, es gibt ein $\pi \in \mathbb{Z}_K$ mit $\pi \mid a+b\alpha$ und $(\pi) \cap \mathbb{Z} = (p)$, dann gilt $p \nmid b$, da andernfalls gelten würde

$$\pi \mid a+b\alpha \Rightarrow a+b\alpha \in (\pi) \Rightarrow a \in (\pi),$$

was mit $a \in \mathbb{Z}$ und der Tatsache, dass π über p liegt, hiesse, dass auch $a \in (p)$ ist, also $p \mid a$, was ein Widerspruch zu $\mathrm{ggT}(a,b) = 1$ wäre – die Voraussetzung $\mathrm{ggT}(a,b) = 1$ ist also notwendig!

Nun konstruieren wir das c_p: aus $\pi \mid a+b\alpha$ folgt

$$a \equiv -b\alpha \mod \pi$$

und aus $\mathrm{ggT}(p,b) = 1$ folgt, dass es ein Inverses b^{-1} zu b modulo p gibt, mit dem gilt

$$-ab^{-1} \equiv \alpha \mod \pi.$$

Sei nun $c_p = -ab^{-1}$, dann gilt wegen $c_p \equiv \alpha \mod \pi$, dass $c_p - \alpha \in (\pi)$ ist. Zudem folgt aus $c_p \equiv \alpha \mod \pi$

$$f(c_p) \equiv f(\alpha) \equiv 0 \mod \pi.$$

Wegen $f(c_p) \in \mathbb{Z}$ und weil $(\pi) \cap \mathbb{Z} = (p)$ ist, folgt $f(c_p) \equiv 0 \mod p$. Das c_p besitzt also die geforderten Eigenschaften!

Zuletzt muss noch gezeigt werden, dass die als notwendig bewiesenen Voraussetzungen auch hinreichend sind. Dazu nehmen wir an, dass ein c_p existiert mit $p \mid a+b\alpha$ und $f(c_p) \equiv 0 \mod p$. Die letzte Bedingung bedeutet $(X - c_p)$ teilt $f(X)$ modulo p. Dann existiert nach Satz 6.6.4 ein irreduzibles Element π mit $(\pi) = (p, \alpha - c_p)$. Für dieses π gilt dann $\alpha \equiv c_p \mod \pi$. Wegen $a + bc_p \equiv 0 \mod p$ gilt auch $a + bc_p \equiv 0 \mod \pi$ und damit

$$0 \equiv a + bc_p \equiv a + b\alpha \mod \pi.$$

Also gilt wie gefordert $\pi \mid a+b\alpha$. □

Aus dem Satz folgt das Korollar

COROLLARY 6.6.6. *Sei $K = \mathbb{Q}(\alpha)$ ein Zahlkörper, der Annahme 6.5.4 genügt und f das Minimalpolynom von α und $a, b \in \mathbb{Z}$ mit $ggT(a,b) = 1$.*
Zu jeder Primzahl $p \in \mathbb{Z}$ gibt es höchstens ein irreduzibles π (bis auf Mulitplikation mit Einheiten natürlich), so dass $\pi \mid a+b\alpha$ und $(\pi) \cap \mathbb{Z} = (p)$. Der höchste Exponent, mit dem

$a+b\alpha$ von π geteilt wird, ist auch gleichzeitig der höchste Exponent, mit dem p die Norm von $a+b\alpha$ teilt.

BEWEIS. Nach Satz 6.6.5 gibt es zu einem π mit $\pi \mid a+b\alpha$ und $(\pi)\cap\mathbb{Z}=(p)$ ein $c_p \in \mathbb{Z}$ mit $p \mid a+b\alpha$ und $f(c_p) \equiv 0 \mod p$. Dieses c_p ist modulo p eindeutig mit $p \mid a+b\alpha$. Nach dem Beweis von Satz 6.6.5 ist π mit $(\pi)=(p,\alpha-c_p)$ bis auf Einheiten eindeutig. Da π genau einem Linearfaktor in f modulo p entspricht, gilt nach Satz 6.6.4 $N(\pi)=p$.

Sei jetzt $a+b\alpha=\varepsilon\prod_{i=1}^{k}\pi_i^{e_i}$ die Zerlegung von $a+b\alpha$ in irreduzible Elemente. Es sei $\pi_1=\pi$. Da π bis auf Einheiten das einzige irreduzible Element über p ist, das $a+b\alpha$ teilt, gilt $N(\pi_i)\neq p^f$ für alle $i\neq 1$. Damit gilt dann

$$N(a+b\alpha)=N\left(\pi_1^{e_1}\right)\prod_{i=2}^{k}N(\pi_i)^{e_i}=p^{e_1}m,$$

wobei gilt $p\nmid m$. □

Nun zurück zu unserem ursprünglichen Vorhaben: Um $a+b\alpha$ in $\mathbb{Z}_K$ zu faktorisieren, bestimmen wir mit Satz 6.6.4 für kleine Primzahlen $p\in\mathbb{Z}$ alle über p liegenden irreduziblen Primideale der Form $(p,\alpha-c)$ mit $c\in\mathbb{Z}_p$, um dann mit Hilfe von Satz 6.6.5 und Korollar 6.6.6 die Faktoren von $a+b\alpha$ zu bestimmen. Dazu muss aber erst noch gezeigt werden, wie die linearen Faktoren von Polynomen aus $\mathbb{Z}_p[X]$ berechnet werden können.

Ausserdem liefert Satz 6.6.4 für die Primideale über einer Primzahl p lediglich die Darstellung $(p,\alpha-c)$. Wie daraus ein irreduzibles Element π berechnet werden kann mit $(\pi)=(p,\alpha-c)$, wird noch gezeigt werden. Zunächst soll jedoch die Bestimmung der linearen Faktoren eines Polynoms über $\mathbb{Z}_p$ betrachtet werden.

Sei p eine Primzahl, dann gilt folgendes Lemma:

LEMMA 6.6.7. *Sei $f(X)\in\mathbb{Z}_p[X]$ mit führendem Koeffizienten 1. Dann ist*

$$ggT(X^p-X,f)=\prod_{c\in\mathbb{Z}_p,\,f(c)\equiv 0 \mod p}(X-c),$$

d.h. der grösste gemeinsame Teiler enthält nur lineare Faktoren von f.

BEWEIS. Es gilt $X^p-X=\prod_{c\in\mathbb{Z}_p}(X-c)$. Auf jeden Fall ist 0 eine Wurzel dieses Polynoms. Für die restlichen Elemente folgt die Gleichung aus dem Satz 1.2.6 von Fermat, Legendre. □

Den $\mathrm{ggT}(X^p-X,f)$ kann man mit Hilfe des euklidischen Algorithmus für Polynome sehr effizient berechnen. Damit große p mit großen Exponenten vermieden werden, rechnet man am günstigsten mit $X^p \mod f$ – errechnet also zunächst X^p durch sukzessives Quadrieren modulo f. Es gilt nämlich

$$\begin{aligned}\mathrm{ggT}(X^p-X,f) &= \mathrm{ggT}(X^p-X \mod f,f)=\mathrm{ggT}(X^p \mod f-X \mod f,f)\\ &= \mathrm{ggT}(X^p \mod f-X,f).\end{aligned}$$

Wenn f' die erste Ableitung von f ist, dann gilt

$$f(X)=\sum_{i=0}^{n}f_iX^i \quad\Rightarrow\quad f'(X)=\sum_{i=1}^{n}iX^{i-1}f_i.$$

Bei $f' = 0$ ist das Polynom f also entweder eine Konstante oder $f = g^p$ mit $g \in \mathbb{Z}_p[X]$. In letzterem Fall kann man f durch g ersetzen, ohne lineare Faktoren zu ändern. Wir können also annehmen, dass $f' \neq 0$ ist. Dann hat $f/\mathrm{ggT}(f,f')$ nur noch einfache Nullstellen oder einfache lineare Faktoren, d.h. $f/\mathrm{ggT}(f,f')$ wird von keinem Polynom der Form $(X-c)^2$ mit $c \in \mathbb{Z}_p$ geteilt.

Um nun für Polynome, die in lineare Faktoren zerfallen, diese Faktoren zu bestimmen, genügt es, einen Algorithmus zu haben, der f in 2 Faktoren f_1, f_2 aufspaltet mit $\mathrm{grad}(f_i) < \mathrm{grad}(f)$ für $i = 1,2$. Durch rekursives Aufspalten von f_1 und f_2 können dann sukzessive die linearen Faktoren von f bestimmt werden. Die Berechnung von f_1 und f_2 beruht auf dem nun folgenden Lemma:

LEMMA 6.6.8. *$f(X) \in \mathbb{Z}[X]$ habe nur lineare Faktoren, Dann gilt*

(1) *Für jedes $a \in \mathbb{Z}_p$ wird $ggT\left((X+a)^{(p-1)/2} - 1, f\right)$ genau von linearen Polynomen $X - c$, $c \in \mathbb{Z}_p$ geteilt, für die $a + c$ ein quadratischer Rest modulo p ist.*
(2) *Ist $\mathrm{grad}(f) \geq 2$ und hat f nur einfache Nullstellen, so ist für $a \in_R \mathbb{Z}_p$ mit Wahrscheinlichkeit $> 1/2$ das Polynom $ggT\left((X+y)^{(p-1)/2} - 1, f\right)$ verschieden vom konstanten Polynom 1 und vom Polynom f selbst.*

BEWEIS. (1) Die Nullstellen von $(X+a)^{(p-1)/2} - 1$ sind nach Lemma 1.2.3 ($\mathrm{grad}_n(a) = \varphi(n)/\mathrm{ggT}(\varphi(n), d)$ für $a = g^d$) genau die $c \in \mathbb{Z}_p$, für die $a + c$ ein quadratischer Rest ist.

(2) Seien c_1, c_2 mit $c_1 \neq c_2$ Nullstellen von f. Damit $X - c_1$ und $X - c_2$ beide entweder $\mathrm{ggT}\left((X+a)^{(p-1)/2} - 1, f\right)$ teilen oder beide den ggT nicht teilen, muss gelten

$$(a + c_1)^{(p-1)/2} \equiv (a + c_2)^{(p-1)/2}.$$

Dieses ist der Fall, wenn a Nullstelle des Polynoms $(X + c_1)^{(p-1)/2} - (X + c_2)^{(p-1)/2}$ ist. Diese Polynom hat den Grad $(p-3)/2$ und hat deswegen höchstens $(p-3)/2$-viele Nullstellen und $a \in_R \mathbb{Z}_p$ ist mir Wahrscheinlichkeit $> 1/2$ keine Nullstelle, da gilt

$$\begin{aligned} \mathrm{Ws}(a \text{ ist Nullstelle}) &\leq \frac{(p-3)/2}{p} = \frac{p-3}{2p} \\ \Rightarrow \quad \mathrm{Ws}(a \text{ ist nicht Nullstelle}) &> 1 - \frac{p-3}{2p} = \frac{p+3}{2p} = \frac{1}{2}\underbrace{\left(\frac{p+3}{p}\right)}_{>1} > \frac{1}{2}. \end{aligned}$$

Für solche a ist dann auch $\mathrm{ggT}\left((X+a)^{(p-1)/2} - 1, f\right) \neq 1, f$. □

Eine direkte Folgerung aus dem Lemma ist, dass im Mittel zwei a gewählt werden müssen, bevor f in zwei Polynome f_1 und f_2 aufgespalten werden kann. Mit den vorangegangenen Ausführungen erhalten wir damit den folgenden Satz.

PROPOSITION 6.6.9. *Sei $f(X) \in \mathbb{Z}[X]$. Dann können wir in erwartet polynomieller Zeit die linearen Faktoren $X - c$ mit $c \in \mathbb{Z}_p$ von f errechnen.*

Wie kann man nun die erzeugenden Elemente π für Primideale der Form $(p, \alpha - c)$ bestimmen? Dazu werden wir einen Algorithmus skizzieren, der allerdings nicht genauer untersucht werden wird. Parallel werden wir auch einen Algorithmus kennenlernen, die Fundamentaleinheiten berechnet. Dazu folgende Bemerkungen:

REMARK 6.6.10. Die Norm eines Elementes $\gamma = \sum_{i=0}^{d-1} c_i \alpha^i$ in $\mathbb{Z}_K$ kann effizient berechnet werden, da

$$N(\gamma) = \prod_{j=0}^{d-1} \sum_{i=0}^{d-1} c_i \alpha_j^i$$

ist mit den α_j als Konjugierte von α. Um jetzt die die Norm zu bestimmen, genügt es, den obigen Ausdruck mit absolutem Fehler $1/2$ zu approximieren. Das gilt deshalb, weil die Norm eine Zahl aus $\mathbb{Z}$ liefert. Jetzt können die α_j als Nullstellen eines ganzzahligen Polynoms approximiert werden – eine gute Approximation für die α_j liefert dann auch eine gute Approximation der Norm.

Die zweite Bemerkung betrifft Elemente der Ideale $(p, \alpha - c)$.

REMARK 6.6.11. Wir benötigen einen effizienten Algorithmus, der entscheidet, ob $\gamma = \sum_{i=0}^{d-1} c_i \alpha^i \in \mathbb{Z}_K$ ein Element des Ideals ist. Dafür benötigen wir das folgende Lemma:

LEMMA 6.6.12. *Sei $p \in \mathbb{Z}$ prim und $(p, \alpha - c)$ Ideal von $\mathbb{Z}_K$ mit $c \in \mathbb{Z}_p$. Dann und nur dann ist $\gamma = \sum_{i=0}^{d-1} c_i \alpha^i \in \mathbb{Z}_K$ ein Element des Ideals, wenn $\sum_{i=0}^{d-1} c_i c^i \equiv 0 \mod p$ ist.*

BEWEIS. Sei $(\pi) = (p, \alpha - c)$. Zunächst sei die Notwendigkeit gezeigt: Ist $\gamma \in (\pi)$, dann gilt $\gamma = \sum_{i=0}^{d-1} c_i c^i \equiv 0 \mod \pi$. Da $\alpha \equiv c \mod \pi$ folgt dann, dass die Summe $\sum_{i=0}^{d-1} c_i c^i \equiv 0 \mod \pi$ ein Element aus $\mathbb{Z}$ ist und daher gilt $\sum_{i=0}^{d-1} c_i c^i \equiv 0 \mod p$.

Jetzt muss noch gezeigt werden, dass die Bedingung hinreichend ist: Ist $\sum_{i=0}^{d-1} c_i c^i \equiv 0 \mod p$ dann ist auch $\sum_{i=0}^{d-1} c_i \alpha^i \equiv 0 \mod \pi$. Da $\alpha \equiv c \mod \pi$ gilt, folgt $\sum_{i=0}^{d-1} c_i \alpha^i \equiv 0 \mod \pi$. Das heisst $\gamma = \sum_{i=0}^{d-1} c_i \alpha^i \in (\pi) = (p, \alpha - c)$. □

Wir bezeichnen im folgenden ein Polynom $f \in \mathbb{Z}$ und eine Primzahl $p \in \mathbb{Z}$ mit $R(p)$ die **Menge der Nullstellen von** $f(X) \mod p$ **in** $\mathbb{Z}_p$. Wie bisher nehmen wir dabei an, dass der von einer Nullstelle erzeugte algebraische Zahlkörper $K = \mathbb{Q}(\alpha)$ die Annahme 6.5.4 erfüllt.

In Schritt (4) wird versucht, alle erzeugende Element für die Ideale $(p, \alpha - c)$ unter den Elementen mit kleinen Koeffizienten zu finden. Hierbei ist $m(c, p)$ ein Mass dafür, wie nahe man an so einem erzeugenden Element ist (bei $m(c, p) = 1$ hat man eines gefunden). Auf diese Weise findet man aber wahrscheinlich nicht für alle Ideale ein erzeugendes Element – es sei denn, man wählt D sehr groß, was aber zu einer langen Laufzeit führen würde. Aus diesem Grund werden in Schritt (5) noch gewisse Quotienten daraufhin getestet, ob sie noch Ideale erzeugen. Dabei entstehen u.U. recht große Koeffizienten, aber man hofft darauf, dass auf diese Weise noch erzeugende Elemente von Idealen gefunden werden.

Man kann zeigen, dass für eine geschickte Wahl von D nach Schritt (4) nur noch wenige Ideale übrigbleiben, für die noch kein erzeugendes Element gefunden wurde. Theoretisch könnte man dann für die restlichen Erzeugenden einen anderen Algorithmus einsetzen – hier soll das Schritt (5) erledigen. Wie D und M gewählt werden müssen, soll hier nicht näher erläutert werden.

Die Bestimmung der Einheiten ist strukturell der Bestimmung der erzeugenden Elemente sehr ähnlich: zunächst wird versucht, die Einheiten durch Elemente mit kleinen Koeffizienten zu bestimmen. Dann versucht man zusätzliche Einheiten durch Quotientenbildung zu berechnen. In der Regel kommt man mit diesem Vorgehen zu einer vollständigen Menge von Fundamentaleinheiten – es kann allerdings passieren, dass man nicht sicher weiss,

Algorithm 33 Bestimmung irreduzibler Elemente und von Fundamentaleinheiten

EINGABE: $f \in \mathbb{Z}[X]$ irreduzibel mit $\operatorname{grad}(f) = d$, $B \in \mathbb{N}$.
AUSGABE: Für alle Primzahlen $p \leq B$ und alle $c \in R(p)$ ein $\pi_{c,p} \in \mathbb{Z}_K$ mit $(p, \alpha - c) = (\pi_{c,p})$. Ausserdem eine Menge U von Fundamentaleinheiten in $\mathbb{Z}_K$.

(1) Wähle Schranken $M, D \in \mathbb{N}$.
(2) Setze $S = \{\}$.
(3) **** Initialisierung der Variablen ****
for all $p \leq B$ *und* $c \in \mathbb{R}(p)$ **do**
(a) **** $m(c,p)$ ist ein Mass dafür, wie nahe man an einem erzeugenden Element dran ist – bei $m(c,p) = 1$ wurde eines gefunden ****
Setze $\pi_{c,p} := 1$ und $m(c,p) := M + 1$.
(4) **** Finden der erzeugenden Elemente für die Ideale $(p, \alpha - c)$ unter den Elementen mit kleinen Koeffizienten $|c_i| \leq D$ ****
for all $\gamma = \sum_{i=0}^{d-1} c_i \alpha^i$ *mit* $|c_i| \leq D$ **do**
(a) Berechne die Norm $N(\gamma)$.
(b) **if** $2 \leq N(\gamma) \leq M$ **then**
(i) Setze $S := S \cup \{\gamma\}$.
(ii) **for all** $p \leq B$ mit $N(\gamma) = kp$ *und* $k \in \mathbb{Z}$ **do**
(A) **for all** $c \in R(p)$ **do**
if $\sum_{i=0}^{d-1} c_i c^i \equiv 0 \mod p$ *und* $|k| \leq m(c,p)$ **then**
Setze $\pi_{c,p} := \gamma$ und $m(c,p) = |k|$.
(c) **if** $N(\gamma) = \pm 1$ **then**
(i) Setze $U := U \cup \{\gamma\}$.
(5) **** Da wahrscheinlich nicht für alle Ideale erzeugende Elemente gefunden wurden, werden nun noch einige Quotienten getestet ****
for all (p,c) *mit* $m(c,p) \neq 1$ *und alle* $\gamma \in S$ *mit* $N(\gamma) = m(c,p)$ **do**
(a) **if** $\pi_{c,p}/\gamma = \sum_{i=0}^{d-1} c_i \alpha^i \in \mathbb{Z}_K$ *und* $\sum_{i=0}^{d-1} c_i c^i \equiv 0 \mod p$ **then**
(i) Setze $\pi_{c,p} := \pi_{c,p}/\gamma$ und $m(c,p) = 1$.
(6) **for** je zwei Elemente $\gamma\gamma' \in S$ *mit* $N(\gamma) = \pm N(\gamma')$ **do**
(a) **if** $\gamma/\gamma' \in \mathbb{Z}_K$ **then**
(i) Setze $U = U \cup \{\gamma/\gamma'\}$.
(7) **for all** (c,p) *mit* $m(c,p) = 1$ **do**
(a) **return(**$\pi_{c,p}$**,** U**)**

ob die Menge auch wirklich vollständig ist. In der Praxis versucht man einfach mit der Ergebnismenge U, die der Algorithmus liefert, möglichst weit zu kommen. Stellt man im Verlauf des Zahlkörpersiebs fest, dass die Menge U nicht ausreicht, so geht man zu anderen (aufwendigeren) Algorithmen über, um die vollständige Menge zu berechnen.

6.7. Das Zahlkörpersieb

Alles bisher besprochene lässt sich nun zu dem eigentlichen Zahlkörpersieb-Algorithmus zusammenfassen (Algorithmus 34).

Algorithm 34 Zahlkörpersieb

EINGABE: $n \in \mathbb{N}$, n keine Primzahlpotenz.

AUSGABE: $k \in \mathbb{N}$ mit $k \neq 1, n$ und $k \mid n$ oder *null*, wenn kein Teiler gefunden wurde.

(1) Bestimme Schranke $d \in \mathbb{N}$.

(2) Bestimme irreduzibles Polynom $f \in \mathbb{Z}[X]$ mit $\operatorname{grad}(f) \leq d$ und ein $m \in \mathbb{N}$ mit $f(m) \equiv 0 \mod n$.

Sei α Nullstelle von f, dann nehmen wir an, dass $K = \mathbb{Q}(\alpha)$ Annahme 6.5.4 genügt, also insbesondere gilt $\mathbb{Z}_K = \mathbb{Z}[\alpha]$.

(3) Bestimme Schranke $B \in \mathbb{N}$.

(4) Bestimme alle Primzahlen $p_1, \ldots, p_l \in \mathbb{Z}$ mit $p_i \leq B$ prim ($\rightarrow$Sieb des Erathostenes).

(5) Setze $p_0 := -1$.

(6) **for** $i = 1, \ldots, l$ **do**

(a) Bestimme Menge $R(p_i)$ der Nullstellen von $f \mod p_i$ in $\mathbb{Z}_{p_i}$.

(7) Bestimme Schanke M.

(8) Erzeuge Liste L_1 mit Paaren (a,b) mit $a \in \mathbb{Z}$, $|a| \leq M$, $b \in \mathbb{N}$, $b \leq M$ und $\operatorname{ggT}(a,b) = 1$ inklusive $a + bm$.

(9) Erzeuge Liste L_2 mit der Norm $N(a + b\alpha)$ der Paare (a,b) aus L_1.

(10) **for all** p_i **do**

(a) **for all** $b \leq M$ **do**

(i) Bestimme $\overline{a} \in \mathbb{Z}_{p_i}$ mit $\overline{a} + bm \equiv 0 \mod p_i$.

(ii) **for all** *a mit* $|a| \leq M$ *und* $\operatorname{ggT}(a,b) = 1$, $a = \overline{a} + jp_i$, $j \in \mathbb{Z}$ **do**

(A) Teile den Eintrag für (a,b) in L_1 durch die maximale Potenz von p_i.

(b) Entferne aus L_2 die Einträge für Paare, deren Listeneintrag in L_1 ungleich 1 sind.

(11) **for all** (p_i, c) *mit* $c \in R(p_i)$ **do**

(a) Bestimme π_{c,p_i} mit $(\pi_{c,p_i}) = (p_i, a - c)$.

(12) Sei $\pi_1, \ldots, \pi_k$ die Menge der hierbei auftretenden irreduziblen Elemente, dann bestimme $\xi, \varepsilon_1, \ldots, \varepsilon_{k'}$ von Fundamentaleinheiten in $\mathbb{Z}_K$.

(13) **for all** p_i **do**

(a) **for all** $c \in \mathbb{R}(p_i)$ **do**

(i) Bestimme $\overline{a} \in \mathbb{Z}_p$ mit $\overline{a} + bc \equiv 0 \mod p$.

(ii) **for all** *a mit* $|a| \leq M$, $\operatorname{ggT}(a,b) = 1$, $a = \overline{a} + jp_i$, $j \in \mathbb{Z}$ **do**

(A) Teile den Listeneintrag $N(a + b\alpha)$ für das Paar (a,b) in L_2 durch die maximale Potenz von p_i.

(14) **for all** (a,b) *mit* Listeneintrag in L_2 ist gleich ± 1 **do**

(a) Berechne durch sukzessives Dividieren die vollständige Faktorisierung von $a + bm$ mit Hilfe von p_i

(b) Berechne die vollständige Faktorisierung der $a + b\alpha$ mit Hilfe der $\pi_{c,p_i}, \xi, \varepsilon_j$. Die vollständig faktorisierten Zahlen und ihre Faktorisierung sei dann gegeben durch

$$\begin{aligned} a_j + b_j m &= \prod_{i=0}^{l} p_i^{c_{ij}} \\ a_j + b_j \alpha &= \xi^{d_{0j}} \prod_{i=1}^{k'} \varepsilon_i^{d_{ij}} \prod_{i=1}^{k} \pi_i^{e_{ij}} \end{aligned}$$

mit $j = 1, \ldots, K$.

Fortsetzung auf Seite 116.

Algorithm 35 Zahlkörpersieb (Fortsetzung)

(15) Finde Lösung $f = (f_1, \ldots, f_K) \neq (0, \ldots, 0)$ für

$$\begin{aligned}
\sum_{j=0}^{K} f_j c_{ij} &\equiv 0 \mod 2, i = 0, \ldots, l \\
\sum_{j=0}^{K} f_j d_{ij} &\equiv 0 \mod 2, i = 0, \ldots, k' \\
\sum_{j=i}^{K} f_j e_{ij} &\equiv 1 \mod 2, i = 0, \ldots, k.
\end{aligned}$$

(16) **if** eine solche Lösung f exisitert nicht **then**
   (a) **return(null)**

(17) Setze

$$x \equiv \prod_{i=0}^{l} p_i^{\frac{1}{2}\sum_{j=1}^{K} f_j c_{ij}} \mod n$$

$$y \equiv \varphi(\xi)^{\frac{1}{2}\sum_{j=1}^{K} f_j d_{0j}} \prod_{i=1}^{k'} \varphi(\varepsilon_i)^{\frac{1}{2}\sum_{j=1}^{K} f_j d_{ij}} \prod_{i=0}^{k} \varphi(\pi_i)^{\frac{1}{2}\sum_{j=1}^{K} f_j e_{ij}} \mod n.$$

mit der Abbildung φ

$$\begin{aligned}
\varphi: \quad \mathbb{Z}[\alpha] &\longrightarrow \mathbb{Z}_n \\
\sum c_i \alpha^i &\longmapsto \sum c_i m^i.
\end{aligned}$$

(18) Berechne $k = \mathrm{ggT}(x - y, n)$.

(19) **if** $k \neq 1, n$ **then**
   (a) **return(*k*)**

(20) **else**
   (a) **return(null)**

6.8. Analyse des Zahlkörpersiebs

Eine vollständige Analyse des Zahlkörpersiebs kann leider nicht vorgenommen werden, da einige zahlentheoretische Annahmen gemacht werden, deren Gültigkeit noch nicht bewiesen ist. Da die Analyse trotzdem noch sehr schwer ist, werden einige grobe Vereinfachungen gemacht, so dass die Analyse als nicht sehr genau angesehen werden muss. So wird z.B. nicht die Wahrscheinlichkeit berechnet, mit der in Schritt (17) mit Hilfe des ggT ein echter Teiler von n gefunden wird – es kann jedoch davon ausgegangen werden, dass aufgrund der vielen nichttrivialen Lösungen des Gleichungssystems in Schritt (15) mit sehr hoher Wahrscheinlichkeit ein nichttrivialer Teiler von n in Schritt (17) gefunden wird.

Wie sollten nun die Variablen d, B und M am besten gewählt werden, damit in Schritt (15) eine nichttriviale Lösung f des Gleichungssystems gefunden wird und gleichzeitig die Laufzeit möglichst klein gehalten wird?

Die Anzahl der Gleichungen entspricht genau der Summe aus der Anzahl der Primzahlen p_i, der Anzahl der irreduziblen Elemente π_o und der Anzahl der Einheiten ξe_i:

$$\begin{aligned}\#\{\text{Gleichungen in Schritt 15}\} &= \#\{\text{Primzahlen } p_i\} \\ &\quad +\#\{\text{irreduzible Elemente } \pi_i\} \\ &\quad +\#\{\text{Einheiten } \xi, e_i\}\end{aligned}$$

Dabei gilt auf jeden Fall

$$\#\{\text{Primzahlen } p_i\} < B.$$

Da über jedem p_i höchsten d viele irreduzible Elemente liegen können, die sich nicht durch Einheiten unterscheiden, fogt

$$\#\{\text{irreduzible Elemente } \pi_i\} < dB.$$

Schliesslich gilt nach Satz 6.5.2, dass die Anzahl der ξ, ε_i durch d beschränkt ist:

$$\#\{\text{Einheiten } \xi, e_i\} < d.$$

Damit erhalten wir

$$\begin{aligned}\#\{\text{Gleichungen in Schritt 15}\} &< B + dB + d = d(B+1) + B \\ &< d(B+1) + (B+1) = (d+1)(B+1).\end{aligned}$$

Die Anzahl der Variablen in dem Gleichungssystem ist gegeben durch die Anzahl der Paare (a,b), für die $a+bm$ durch die p_i und $ab+\alpha$ durch die ξ, ε_i vollständig faktorisierbar ist. Wegen der obigen Schranke für die Anzahl der Gleichungen sollten also mindestens $(d+1)(B+1)$-viele Paare (a,b) diese Bedingungen erfüllen, damit das Gleichungssystem eine nichttriviale Lösung besitzt.

Nach Satz 6.6.5 und Korollar 6.6.6 und der Beschreibung des Zahlkörpersiebs lässt sich $a+bm$ durch p_i und $a+b\alpha$ durch π_i, ε_i genau dann faktorisieren, wenn die Zahl $(a+bm)N(a+b\alpha)$ B-glatt ist. Die Grösse dieser Zahl wird im nächste Lemma abgeschätzt und ist der hauptsächliche Grund, warum das Zahlkörpersieb eine soviel bessere Laufzeit hat als z.B. das Quadratische Sieb.

LEMMA 6.8.1. *Mit den Schranken aus der Beschreibuung des Zahlkörpersiebs gilt*

$$|(a+bm)N(a+b\alpha)| \leq 2dn^{2/d}M^{d+1}.$$

BEWEIS. Sind $\alpha_0 = \alpha, \alpha_1, \ldots, \alpha_{d-1}$ die Konjugierten von α, so ist

$$N(a+b\alpha) = \prod_{j=0}^{d-1}\left(a+b\alpha_j\right).$$

Da die α_j Nullstellen des Polynoms f sind, sind die $a+b\alpha_j$ genau die Nullstellen des Polynoms

$$g(X) = b^d f\left(\frac{x-a}{b}\right) \in \mathbb{Z}[X].$$

$g(X)$ hat den führenden Koeffizienten 1 und damit ist $\left|\prod_{j=0}^{d-1}\left(a+b\alpha_j\right)\right|$ der Absolutbetrag des konstanten Terms von g. Dieser konstante Term ist gegeben durch $g_0 = \sum_{i=0}^{d-1} f_i(-a)^i b^{d-i}$. Da $\left|f_i\right| < m$ für $i = 0, \ldots, d-1$ und $|a|, b \leq M$ ist, gilt

$$\left|g_0\right| = |N(a+b\alpha)| \leq (d+1)mM^d.$$

Da weiterhin gilt $|a+bm| \leq (M+1)\,m$ und $m \leq n^{1/d}$, folgt

$$\begin{aligned} |(a+bm)\,N(a+b\alpha)| &\leq (M+1)\,m\,(d+1)\,mM^d \\ &\leq (M+1)\,n^{1/d}\,(d+1)\,n^{1/d}M^d = n^{2/d}\left(M^{d+1}+M\right)(d+1) \\ &\leq 2dn^{2/d}M^{d+1} \end{aligned}$$

□

Im Zahlkörpersieb nehmen wir nur die (a,b) mit $\mathrm{ggT}(a,b)=1$. Dabei muss sichergestellt sein, dass es genügend viele Zahlen dieser Art gibt, was durch folgendes Lemma erreicht wird.

LEMMA 6.8.2. *Sei* $T(x) = |\{(a,b) \in \mathbb{N} \mid a,b \leq x,\, ggT(a,b)=1\}|$. *Dann gilt* $T(x) \to 6/\pi^2 x$ *für* $x \to \infty$.

BEWEIS. Ohne Beweis. □

Da wir zulassen, dass $a \in \mathbb{Z}$ ist und nicht nur $a \in \mathbb{N}$, können wir erwarten, dass es für etwa $\left(12/\pi^2\right)M^2$-viele Paare (a,b) mit $a \in \mathbb{Z}$,$b \in \mathbb{N}$,$|a| \leq M$,$b \leq M$ gilt $\mathrm{ggT}(a,b)=1$. Für die weitere Analyse machen wir folgende Annahme:

CLAIM 6.8.3. Setze $N_0 = 2dn^{2/d}M^{d+1}$. Wir nehmen an, dass sich die Zahlen $|(a+bm)\,N(a+b\alpha)|$ wir zufällige Zahlen im Intervall $[1,M]$ verhalten. Hier sind a,b wie im Zahlkörpersieb. Insbesondere nehmen wir an, dass für

$$\frac{12M^2\Psi\left(N_0,B\right)}{\pi^2 N_0} - \text{viele}$$

dieser Paare das Produkt $|(a+bm)\,N(a+b\alpha)|$ B-glatt ist (siehe auch Satz 4.6.7).

Der wesentliche Unterschied zwischen dieser Annahme und der, die für das Quadratische Sieb gemacht wurde (Annahme 5.3.3) ist die Schranke $2dn^{2/d}M^{d+1}$. Für das Quadratische Sieb war die entsprechende Schranke $\sqrt{n}$. Wir werden ausschliesslich $d \approx \sqrt{\log n}$ wählen und wegen $M \approx n^{1/d}$ eine Schranke erhalten, die deutlich kleiner als $\sqrt{n}$ ist.

Desweieren benötigen wir folgendes Lemma, dass mit Hilfe von Satz 4.6.7 bewiesen werden kann.

LEMMA 6.8.4. *Seien* $n,d \in \mathbb{N}$ *mit* $n > 2^{d^2}$. *Weiter sei* $M(n,d)$ *eine Funktion, so dass für alle* n,d *gilt*

$$\log(M) \geq \left(\frac{1}{2}+o(1)\right)\left(d\log(d)+\sqrt{(d\log(d))^2+4\log\left(n^{1/d}\log\log\left(n^{1/d}\right)\right)}\right),$$

mit dem natürlichen Logarithmus log. *Dann gilt für* $N_0 = 2dn^{2/d}M^{d+1}$ *und für* $n,d,B \to \infty$

$$\frac{M^2\Psi\left(N_0,B\right)}{N_0} \geq B^{1+o(1)}.$$

Angewandt bedeutet das, dass wenn wir im Zahlkörpersieb die Wahl

$$M,B = \exp\left(\frac{1+\varepsilon}{2}\right)\left(d\log(d)+\sqrt{(d\log(d))^2+4\log\left(n^{1/d}\log\log\left(n^{1/d}\right)\right)}\right)$$

treffen mit $\varepsilon > 0$, dann sind dieBedingungen des Lemmas erfüllt und es gilt somit

$$\frac{M^2 \Psi\left(N_0, B\right)}{N_0} \geq B^{1+\varepsilon}.$$

Mit Annahme 6.8.3 ist dann $12/\pi^2 B^{1+\varepsilon}$ die Anzahl der Paare (a,b), für die $|(a+bm)N(a+b\alpha)|$ B-glatt ist. Somit besitzt auch das Gleichungssystem in Schritt (15) $12/\pi^2 B^{1+\varepsilon}$-viele Variablen. Die Anzahl der Gleichungen hatten wir durch $(d+1)(B+1)$ abgeschätzt und wegen $d = B^{o(1)}$ ist die Anzahl der Gleichungen also $B^{1+o(1)}$. Für jedes $\varepsilon > 0$ und $n \to \infty$ erhalten wir also wesentlich mehr Variablen als Gleichungen und können somit viele nichttriviale Lösungen für das Gleichungssystem finden.

Im weiteren werden wir $\varepsilon = 0$ setzen, was zwar eigentlich nicht korrekt, aber auch nicht ganz falsch ist, die Aussage aber erheblich vereinfacht.

Jetzt soll die Laufzeit abgeschätzt werden. Oben haben wir bereits gesehen, dass die Mengen $R(p_i)$ aus Schritt (6) in polynomieller Zeit berechnet werden können. Die Berechnung der irreduziblen Elemente π_i und der Fundamentaleinheiten in den Schritten (11) und (12) sollen zunächst nicht beachtet werden. Damit können wir eine fast identische Analyse zu der des Quadratischen Siebs machen und erhalten eine Laufzeit von

$$\exp\left(d\log(d) + \sqrt{(d\log(d))^2 + 4\log\left(n^{1/d}\log\log\left(n^{1/d}\right)\right)}\right).$$

Jetzt muss nur noch d bestimmt werden, um die Laufzeit zu optimieren. Eine optimale Wahl von d ist gegeben durch

$$d = \left(3^{1/3} + o(1)\left(\log(n)/\log\log(n)\right)\right)^{1/3},$$

wieder mit log dem natürlichen Logarithmus. Dieses d erfüllt dann auch die Bedingung $n > 2^{d^2}$ aus Lemma 6.8.4 und damit auch die Bedingung $n > (4d)^d$ aus Lemma 6.3.1. Nach Einsetzen von d in die oben angegebene Laufzeit erhalten wir damit den folgenden Satz.

PROPOSITION 6.8.5. *Mit Annahme 6.5.4 und Annahme 6.8.3 ist die Laufzeit des Zahlkörpersiebs gegeben durch*

$$\exp\left(\left((64/9)^{1/3} + o(1)\right)\log(n)^{1/3}\log\log(n)^{2/3}\right).$$

Der Entscheidende Term im Exponent der Laufzeit ist $\log(n)^{1/3}$, was deutlich besser ist als der entsprechende Term $\log(n)^{1/2}$ bei der Laufzeit des Quadratischen Siebs. Die Tatsache, dass der Exponent des $\log\log(n)$-Terms jetzt $2/3$ ist und nicht wie beim Quadratischen Sieb $1/2$, fällt asymptotisch nicht ins Gewicht. Es muss allerding bedacht werden, dass die Grundoperationen im Zahlkörpersieb wesentlich komplizierter sind, als die beim Quadratischen Sieb – der Einsatz des Zahlkörpersieb lohnt damit erst ab Zahlen mit mindestens 130 Dezimalstellen.

Für Zahlen der Form $n = r^e - s$ kann die Laufzeit noch stark verbessert werden (→6.3.1). Der Faktor $(64/9)^{1/3}$ kann dabei durch $(32/9)^{1/3}$ ersetzt werden.

lassen. Da die Anzahl der benötigten Gleichungen $L_{p-1}[\alpha + o(1)]$ ist, ergibt sich hieraus die Ungleichung

$$\alpha + 2\varepsilon \geq \frac{1}{4\beta}.$$

Mit $\alpha \geq \beta$, $\alpha \approx \beta$ erhalten wir, da ε beliebig klein war, $\alpha \approx \beta \approx 1/2$. Wir erhalten dann als Laufzeit für Schritt (3) $L_{p-1}[1+\varepsilon]$, $\varepsilon > 0$ beliebig klein. Dieses ist nun eine deutliche Verbesserung gegenüber $L_{p-1}[2]$ in der ursprünglichen Variante.

Wie oben bereits erwähnt kann eine ähnliche Verbesserung in Schritt (5) erzielt werden, auf die jedoch nicht näher eingegangen werden soll.

KAPITEL 7

Anhang

7.1. Laufzeiten/Algorithmen

TABELLE 1. Basis-Algorithmen

Name des Algorithmus	Eingabe	Ausgabe	Seite	Laufzeit	Bemerkung
Euklidischer Algorithmus	$m,n \in \mathbb{N}$	$\mathrm{ggT}(m,n)$	11	$O\left(\log^2(n)\right)$ Bitoperationen	
Erweiterter Euklidischer Algorithmus	$m,n \in \mathbb{N}$	$\mathrm{ggT}(m,n) = sm+tn$	11	s.o.	Anwendung: Berechnung des Inversen
Addition modulo n	$a,b \in \mathbb{N}$	$x \equiv a+b \bmod n$		$O(\log a + \log b) = O(\log n)$	[**Menezes et al.**, Kapitel 2.4.2.]
Multiplikation modulo n	$a,b \in \mathbb{N}$	$x \equiv a \cdot b \bmod n$		$O(\log a \log b) = O\left((\log n)^2\right)$	s.o.
Multiplikation modulo n	$a,b \in \mathbb{N}$	$x \equiv a/b \bmod n$		$O(\log a \log b) = O\left((\log n)^2\right)$	s.o.
Schnelle Berechnung des Jacobi-/Legendre-Symbols	n ungerade, $a \in \mathbb{Z}_n^*$	Legendre-Symbol $\left(\frac{a}{n}\right)$	20	$O\left((\log n)^2\right)$ Bitoperationen	Laufzeit wie bei ggT
Berechnung der Quadratwurzel modulo p	p prim, $a\mathbb{Z}_p^*$ mit $\left(\frac{a}{p}\right) = 1$	$b \in \mathbb{Z}_p^*$ mit $b^2 \equiv a \bmod p$	20	$O\left((\log p)^4\right)$ Bitoperationen	
Schnelle Exponentiation	$a,e \in \mathbb{N}$	a^e	35	$O\left(\log_2 e\right)$ Bitoperationen	siehe auch Seite **??**
Berechnung der Quadratwurzel modulo p^e	$a \in QR_{p^e}$, $\varphi(p^e) = q2^k$ und $b \in QNR_{p^e}$	$\sqrt{a} \bmod p^e$	35,37	polynomialzeit	Der Algorithmus transponiert a hoch, bis gilt $a \in \left(\mathbb{Z}_{p^e}\right)^{2^k}$, zieht dann dieWurzel und transponiert anschliessen wieder runter.
Sieb des Erathostenes	$n \in \mathbb{N}$	$P = \{p \mid p \text{ prim und } p < n\}$	98	$\sqrt{n}$	Berechnet alle Primzahlen kleiner oder gleich einer gegebenen Zahl.

TABELLE 2. Primzahltests

Name des Algorithmus	Eingabe	Ausgabe	Seite	Laufzeit	Bemerkung
φ-Algorithmus	$n \in \mathbb{N}$ ungerade	n prim?	15	$O\left(\log^2(n)\right)$ Bitoperationen	Antwort „n prim„ ist mit Ws $\leq 1-\varphi(n)/(n-1)$ falsch.
Fermat-Test	$n \in \mathbb{N}$	n prim?	15	s.o.	Ist n nicht-Carmichael, dann liefert der Algorithmus mit Ws $\geq \frac{1}{2}$ einen Zeugen der Nichtprimheit, sonst ist die Erfolgsws. wie im φ-Test.
Solovay-Strassen-Test	$n \in \mathbb{N}$ ungerade	n prim?	21	$O\left((\log n)^3\right)$ Bitoperationen	Die Antwort „n prim„ ist mit Ws $\leq \frac{1}{2}$ falsch.
Miller-Rabin-Test	$n \in \mathbb{N}$ ungerade	n prim?	27	$O\left((\log n)^3\right)$ Bitoperationen	Obwohl gleiche Laufzeit wie SS, ist MR doch schneller und liefert eine Falschaussage mit geringerer Wahrscheinlichkeit.

TABELLE 3. Faktorisierungsalgorithmen

Name des Algorithmus	Eingabe	Ausgabe	Seite	Laufzeit	Bemerkung
Pollard $p-1$ (Stinson)	$n, \gamma \in N$	$d \notin \{1,n\}$ mit $d \mid n$ oder $d=-1$	41	$O(\gamma \log \gamma)$	
Pollard $p-1$ (Blömer)	$n \in \mathbb{N}, B_1, B_2 \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$ oder $d=-1$	41		B_1 ist Schranke für Primfaktoren von $p-1$, B_2 ist Schranke für Primfaktorpotenzen von $gl(p-1)$
Pollard $p-1$ (Menezes)	$n \in \mathbb{N} \setminus \mathbb{P}$ und $n \neq p^e$, Glattheitsschranke γ	$d \notin \{1,n\}$ mit $d \mid n$ oder $d=-1$	44		
Verbesserter $p-1$ (Miller-Rabin)	$n \in \mathbb{N} \setminus \mathbb{P}$ und $n \neq p^e$ mit $p \mid n$, p ist γ-glatt	$d \notin \{1,n\}$ mit $d \mid n$ oder $d=-1$	45	$O\left((k-1)(\log n)^2\right)$	$k = \lfloor \log_2 \gamma \rfloor$
Pollard ρ	$n = p \cdot q$ und F Zufallsfunktion modulo n	$d \notin \{1,n\}$ mit $d \mid n$ oder $d=-1$	48		
FFT	$a_0, \ldots, a_m$, ω ist 2^k-te primitive Einheitswurzel	$a'_i = \sum_{j=0}^{m-1} \omega_m^{ij} a_j$	55	$2^{k-1}(k-1)$ Mult., $2^k k$ Additionen	
Zirkuläres Konvolutionsprodukt	$a, b \in \mathbb{Z}_n^m$, $f = \sum_{i=0}^{m-1} a_i x^i$, $g = \sum_{j=0}^{m-1} b_j x^j$, $m = 2^k$	$c = a \otimes b \in \mathbb{Z}_n^m$, $f \cdot g = \sum_{i=0}^{m-1} c_i x^i \mod (x^m-1)$	57		
Schnelle Berechnung des ggT	$n \in \mathbb{N}$	Teiler von n	58	$O\left(\gamma (\log_2 \gamma)^2\right)$	

TABELLE 4. Elliptische Kurven

Name des Algorithmus	Eingabe	Ausgabe	Seite	Laufzeit	Bemerkung
Goldwasser-Kilian	$n \in \mathbb{N}$, $\mathrm{ggT}(n,6) = 1$	n prim oder nicht	75	erwartet polynomiell	Läßt parallel noch den MR-Test laufen.
Zirkuläres Polynomprodukt	$n \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$	57		
Elliptische Kurven Methode	$n \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$	76	erwartet polynomiell	Einfache Methode, wo die EK nicht gewechselt werden.
EKM mit EK-Wechsel nach Blömer	$n \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$	79	erwartet $L_n(1+o(1))$	
EKM mit EK-Wechsel nach Schnorr	$n \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$	81	s.o.	
Faktorbasisalgorithmus	$n \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$	90	s.o.	Sucht zufällig b_i und testet, ob b_i^2 zerlegbar durch Faktorbasis
Quadratisches Sieb	$n \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$	97	s.o.	Bestimmt die $b_i = (m+i)^2 \bmod n$
Sieb des Erathostenes	$n \in \mathbb{N}$	Primzahlen $< n$	98	$O(\sqrt{n})$	
Bestimmung von Fundamentaleinheiten	$f \in \mathbb{Z}[X]$ irreduzibel, $\mathrm{grad}(f) = d$, $B \in \mathbb{N}$	Fundamentaleinheiten und irreduzible Elemente	114		
Zahlkörpersieb	$n \in \mathbb{N}$	$d \notin \{1,n\}$ mit $d \mid n$	115	$\exp\left(\left((64/9)^{1/3}+o(1)\right)\log(n)^{1/3}\log\log(n)^{2/3}\right)$	

Index

Literaturverzeichnis

[Schnorr]	Schnorr: Skript „Diskrete Mathematik”
[Blömer]	Blömer: Skript „Algorithmen in der Zahlentheorie”
[Wohlfahrt]	Wohlfart: Buch „Einführung in die Zahlentheorie und Algebra”
[Knuth2]	Knuth: Buch „The Art of Computer Programming Vol. 2”
[Childs]	Lindsay N. Childs: Buch „A concrete introduction to higher algebra”
[Stinson]	Douglas R. Stinson: Buch „Cryptography – Theory and Practice”
[Menezes et al.]	Menezes, Oorschot, Vanstone: Buch „Handbook of Cryptography”
[Brent1986]	R.P. Brent: „Some Factorization Algorithms using Elliptic Curves.” Australian Computer Science Communications 8, 1986
[Montgomery, Silverman]	Montgomery and Silverman: „An FFT Extension to the $p-1$ Factoring Algorithm”
[M. Fischlin]	Marc Fischlin: „Kleine Nullstellen von Polynomen: Angriffe auf RSA”, http://www.mi.informatik.uni-frankfurt.de/teaching/ws0001/krypto_algo/index.html
[R. Fischlin]	Roger Fischlin: „Effiziente Arithmetik in $\mathbb{Z}_m$ und $\mathbb{F}_{p^n}$”, http://www.mi.informatik.uni-frankfurt.de/teaching/ws0001/krypto_algo/index.html
[Borodin, Moenck1974]	Borodin, Moenck: „Fast Modular Transforms”, 1974
[Blake et al.]	Ian Blake, Cradiel Serousi, Nigel Smart: „Elliptic Curves in Cryptography”, Cambridge University Paper, London Mathematical Society, Lecture Notes 265, ISBN 0-521-65374-6
[Koblitz1994]	N. Koblitz: „A Course in Number Theory and Cryptography, Second Edition”, Springer-Verlag, 1994
[Pomerance1996]	C. Pomerance: „A Tale of Two Sieves”, Notices of the AMS, Volume 43, Number 12
[Lenstra1987]	H.W. Lenstra: Annals of Mathematic 126, 1987
[Buchmann1999]	J. Buchmann: „Einführung in die Kryptographie”, Springer-Verlag
[Cohen1995]	L. Cohen: „A Course in Compuational Number Theory”, Springer-Verlag 1995, ISBN 3-540-55640-0