Eine Implementierung des Algorithmus von Atkin zur Berechnung der Punktanzahl elliptischer Kurven über endlichen Primkörpern der Charakteristik größer drei

Diplomarbeit

von

Markus Maurer

nach einem Thema von Prof. Dr. Buchmann,
am Fachbereich 14 - Informatik -
der Universität des Saarlandes

Saarbrücken, den 29. Juli 1994

Eidesstattliche Erklärung:

„Hiermit versichere ich an Eides Statt, daß ich diese Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen benutzt habe.“

Saarbrücken, den 29. Juli 1994

Markus Maurer

Mein Dank gilt Herrn Prof. Dr. Buchmann für dieses sehr interessante Thema.

Ich möchte mich besonders bei Volker Müller für die intensive Betreuung bedanken. Er hatte stets Zeit für meine Fragen, und er hat mich durch sein positives Denken motiviert. In vielen Gesprächen war er ein wertvoller Ratgeber.

Victor Shoup gebührt zum einen Dank für die zur Verfügungstellung von effizienten Routinen, die er teilweise extra auf unsere Bedürfnisse zugeschnitten hat. Zum anderen hat er uns durch viele Tips und Anregungen bei der Implementierung geholfen.

Weiterhin bedanke ich mich besonders bei Frank Lehmann. Während der vielen Tag- und Nachtstunden, die wir bei der Implementierung des Algorithmus verbracht haben, herrschte immer eine freundliche und lockere Atmosphäre.

Bedanken möchte ich mich auch bei allen Angehörigen des Lehrstuhls Prof. Buchmann, die uns während unserer Arbeit unterstützt haben; insbesondere bei Franz-Dieter Berger, Thomas Denny, Andreas Müller, Thomas Papanikolaou, Thomas Sosnowski und Jörg Zayer.

Mein Dank gilt auch meinen Eltern, ohne die mein Studium unter solchen optimalen Bedingungen nicht möglich gewesen wäre und die mich, ebenso wie meine Freundin Annette Bühring, stets unterstüzt und bei Rückschlägen aufgemuntert haben.

Inhaltsverzeichnis

Vorwort

In dieser Arbeit beschreiben wir eine Implementierung eines Algorithmus von Oliver Atkin, mit der wir für eine elliptische Kurve, welche über dem endlichen Primkörper $\mathbb{F}_p$ der Charakteristik p größer drei definiert ist, die Anzahl der $\mathbb{F}_p$-rationalen Punkte bestimmen können. Ein wichtiger Bestandteil des Atkin-Algorithmus ist die Berechnung sogenannter modularer Polynome. Die Implementierung von Algorithmen zur Bestimmung dieser Polynome ist der Inhalt der Diplomarbeit von Frank Lehmann ([Le94]); wir gehen in dieser Arbeit davon aus, daß uns die modularen Polynome zur Verfügung stehen. Bei der Beschreibung des Algorithmus stützen wir uns auf die Dissertation von Volker Müller ([Mü94]), der die von Oliver Atkin, Noam Elkies und anderen entwickelten Ideen ausgearbeitet und auf endliche Körper der Charakteristik größer drei übertragen hat. Die im Rahmen der Arbeit von Frank Lehmann und dieser Arbeit entstandene Implementierung ist das Ergebnis der Zusammenarbeit von Frank Lehmann, Volker Müller, Victor Shoup und mir.

Die Anzahl der Punkte auf einer elliptischen Kurve ist vor allem in der Kryptographie von Interesse. Neal Koblitz ([Ko86]) und Victor Miller ([Mi86]) haben 1986 vorgeschlagen, Kryptosysteme mit Hilfe von elliptischen Kurven zu konstruieren. Die Sicherheit dieser Systeme gründet sich auf der Schwierigkeit, diskrete Logarithmen in der Punktgruppe elliptischer Kurven zu berechnen. Der beste bekannte Algorithmus zur Lösung dieses Problems ist das Verfahren von Pohlig-Hellman ([PoHe78]), welches allgemein in endlichen, abelschen Gruppen anwendbar ist. Die Laufzeit dieses Verfahrens ist proportional zur Wurzel des größten Primfaktors der Gruppenordnung der Punktgruppe. Um die Sicherheit eines solchen Kryptosystems zu gewährleisten, ist es somit notwendig, die Gruppenordnung zu berechnen und zu überprüfen, ob sie einen großen Primfaktor besitzt.

Mit den Algorithmen von Daniel Shanks und René Schoof ([Sh70],[Sc85],[Mü91]) und deren Kombination ([BuMü91]) war es möglich, die Anzahl der $\mathbb{F}_p$-rationalen Punkte einer elliptischen Kurve zu bestimmen, wobei $p \approx 10^{42}$ sein konnte. Durch die ersten Verbesserungen des Schoof-Algorithmus und den dadurch entstandenen Grundalgorithmus von Atkin ([At88]) konnte 1992 diese Grenze für p auf 75 Dezimalstellen erhöht werden. Schließlich gelang es Atkin in Zusammenarbeit mit Elkies ([At92], [El]) den Algorithmus so zu erweitern, daß er Gruppenordnungen für Charakteristiken $p \approx 10^{225}$ berechnen konnte. Die größte Charakteristik p, für die inzwischen die Anzahl der $\mathbb{F}_p$-rationalen Punkte einer elliptischen Kurve bestimmt werden konnte, besitzt 375 Dezimalstellen; diese Berechnung wurde mit unserer verteilten Implementierung des Atkin-Algorithmus durchgeführt und dauerte ungefähr eine Woche. An diesen Größenordnungen erkennt man, welchen Fortschritt der Algorithmus von Atkin auf diesem Gebiet gebracht hat.

Diese Arbeit gliedert sich wie folgt:

Im ersten Kapitel geben wir eine exakte Formulierung der Problemstellung an und formulieren grundlegende Definitionen und Sätze. Die Kapitel 2 und 3 beschreiben den Grundalgorithmus von Atkin. Kapitel 2 enthält die Weiterentwicklung des Schoof-Algorithmus; Kapitel 3 beschäftigt sich mit einer von Atkin modifizierten Version des Babystep-Giantstep-Algorithmus, welcher auf Shanks zurückgeht. In Kapitel 4 befassen wir uns mit der Erweiterung des Grundalgorithmus. Wir fassen den gesamten Algorithmus in Kapitel 5 nochmal zusammen und geben einige Daten von durchgeführten Berechnungen an. In Kapitel 6 beschreiben wir die entstandenen Programme, und schließlich geben wir in Kapitel 7 an, was von der Implementierungsseite her noch getan werden könnte.

Die Laufzeiten, die in den einzelnen Kapiteln angegeben werden, beziehen sich auf Berechnungen, die auf Maschinen des Typs SUN-SPARC station ELC mit 22 MIPS durchgeführt wurden.

Notationen

In dieser Arbeit verwenden wir die folgenden Bezeichnungen:

l	ungerade Primzahl
p	Primzahl größer 3
E	elliptische Kurve
$\mathbb{N}$	Menge der natürlichen Zahlen größer 0
$\mathbb{N}_0$	Menge der natürlichen Zahlen einschließlich 0
$\mathbb{Z}$	Ring der ganzen Zahlen
$\mathbb{Q}$	Körper der rationalen Zahlen
$\mathbb{R}$	Körper der reellen Zahlen
$\mathbb{C}$	Körper der komplexen Zahlen
$\mathbb{F}_p$	endlicher Primkörper der Charakteristik p
$\mathbb{F}_p^*$	Multiplikativgruppe von $\mathbb{F}_p$
$R[x_1, \ldots, x_n]$	Polynomring über dem Ring R
$R(x_1, \ldots, x_n)$	Körper der rationalen Funktionen über dem Ring R
1_G	das neutrale Element einer multiplikativen Gruppe G
$ord(g)$	Ordnung eines Gruppenelementes g
ggT	größter gemeinsamer Teiler
$a \equiv_l b$	$a \equiv b$ modulo l
$Grad(f)$	Polynomgrad des Polynoms f
$(h_1, h_2) \bmod f(x)$	h_1, h_2 sind rationale Funktionen, die ein formales Punktvielfaches repräsentieren; dabei sind Zähler und Nenner jeweils modulo $f(x)$ reduziert
$\mathcal{O}(f)$	$\{g : \mathbb{N}_0 \to \mathbb{N}_0 \mid$ es ex. $c > 0, n_0 \in \mathbb{N}_0$ mit $g(n) \leq c \cdot f(n) \forall n \geq n_0\}$, wobei $f : \mathbb{N}_0 \to \mathbb{N}_0$
$\lfloor x \rfloor$	größte Zahl $n \in \mathbb{N}_0$, so daß $n \leq x$, $x \in \mathbb{R}_{>0}$
$\lceil x \rceil$	kleinste Zahl $n \in \mathbb{N}_0$, so daß $n \geq x$, $x \in \mathbb{R}_{>0}$
$\Im(z)$	Imaginärteil von $z \in \mathbb{C}$

Kapitel 1

Grundlagen

Im ersten Kapitel definieren wir den Begriff der elliptischen Kurve und erläutern die Problemstellung, die Berechnung der Gruppenordnung einer Untergruppe der Punktmenge einer elliptischen Kurve. Mit der Einführung einer speziellen Abbildung, dem sogenannten Frobeniusendomorphismus, und besonderen Gruppen, den Torsionsgruppen, lernen wir zwei für den Atkin-Algorithmus grundlegende Objekte kennen. Die Torsionsgruppen stehen in enger Beziehung zu den sogenannten Divisionspolynomen, welche wir in diesem Kapitel definieren, die wir aber erst in Kapitel 4 in vollem Umfang benutzen werden. Weiterhin führen wir den Begriff des Endomorphismenringes ein. Dieser führt uns zum Satz von Hasse, welcher erste Informationen zur Lösung unseres Problems liefert. Darauf aufbauend werden wir unser weiteres Vorgehen in zwei Phasen unterteilen, die in den nachfolgenden Kapiteln beschrieben werden. Wir beginnen mit folgender

1.1. Definition *Sei K ein Körper der Charakteristik $p \neq 2, 3$ mit algebraischem Abschluß $\overline{K}$. Eine* **elliptische Kurve über K**, *bezeichnet durch E/K, ist ein Paar (a, b) mit $a, b \in K$ und $4a^3 + 27b^2 \neq 0$. Mit $E(\overline{K})$ bezeichnen wir die* **Menge aller Punkte auf der elliptischen Kurve**,

$$E(\overline{K}) = \Big\{(x, y) \mid x, y \in \overline{K} \;\; mit \;\; y^2 = x^3 + ax + b\Big\} \cup \{\mathcal{O}\}$$

$$mit \quad \mathcal{O} := (\infty, \infty).$$

1.2. Bemerkung Eine elliptische Kurve heißt definiert über K, falls die Parameter $a, b \in K$ sind. Ist im Kontext klar, daß eine Kurve über dem Körper K definiert ist, so schreiben wir statt E/K auch einfach E. Die Kurvengleichung $y^2 = x^3 + ax + b$ wird die **kurze Weierstrass-Gleichung** genannt, die man im Fall $char(K) \neq 2, 3$ aus der allgemeinen Weierstrass-Gleichung für elliptische Kurven herleiten kann, vgl. [Si86, Kap.III, §1, S.46]. Da wir in dieser Arbeit an „großen" Charakteristiken interessiert sind, können wir die kurze Weierstrassform und damit die im folgenden angebenen Additionsformeln für Punkte verwenden. Falls nichts anderes gesagt wird, so bezeichnen wir in dieser Arbeit mit E eine elliptische Kurve gemäß Definition 1.1, die über dem endlichen Primkörper $\mathbb{F}_p$ der Charakteristik $p > 3$ definiert ist.

Auf obiger Punktmenge $E(\overline{K})$ läßt sich eine Verknüpfung definieren, i.a. additiv geschrieben, mit welcher $E(\overline{K})$ zu einer abelschen Gruppe wird.

1.3. Satz *Die Punktmenge einer elliptischen Kurve, $E(\overline{K})$ gemäß Definition 1.1, bildet mit der im folgenden definierten Addition eine abelsche Gruppe.*

- *$\mathcal{O}$ ist das neutrale Element von $E(\overline{K})$.*
- *Zu einem Punkt $\mathcal{O} \neq P = (x, y) \in E(\overline{K})$ definieren wir als negativen Punkt*

$$-P := (x, -y) \in E(\overline{K}).$$

- *Für Punkte P_1 und P_2 mit $P_2 = -P_1$ definieren wir die Summe als $P_1 + P_2 := \mathcal{O}$.*
- *Für Punkte $P_1 = (x_1, y_1)$ und $P_2 = (x_2, y_2)$ mit $P_2 \neq -P_1$ ist die Summe $P_1 + P_2 := (x_3, y_3)$, wobei*

$$\begin{aligned} x_3 &= -x_1 - x_2 + \lambda^2 \\ y_3 &= -y_1 + \lambda \cdot (x_1 - x_3) \\ \textit{mit} & \\ \lambda &= \frac{y_1 - y_2}{x_1 - x_2}, \quad \textit{falls} \quad P_1 \neq P_2. \\ \lambda &= \frac{3x_1^2 + a}{2y_1}, \quad \textit{falls} \quad P_1 = P_2. \end{aligned}$$

Beweis: vgl.[Mü91, Kap.1]. ■

Wir sind nun an einer Untergruppe der Punktmenge einer elliptischen Kurve interessiert, der Menge der K-rationalen Punkte.

1.4. Definition *Unter den Voraussetzungen von Definition 1.1 bezeichnen wir mit*

$$E(K) = \left\{ (x, y) \mid x, y \in K \;\; \textit{mit} \;\; y^2 = x^3 + ax + b \right\} \cup \{\mathcal{O}\}$$

$$\textit{mit} \quad \mathcal{O} := (\infty, \infty)$$

die Menge aller **K-rationalen Punkte auf** *E.*

1.5. Bemerkung Da K ein Körper und E über K definiert ist, sieht man anhand der Definition der Punktaddition direkt, daß $E(K)$ wirklich eine Untergruppe von $E(\overline{K})$ ist.

Damit können wir unsere Aufgabenstellung, die wir in dieser Arbeit behandeln werden, wie folgt formulieren:

> Berechne zu gegebener Charakteristik $p > 3$ und Elementen $a, b \in \mathbb{F}_p$ mit $4a^3 + 27b^2 \neq 0$, die die Kurve E über $\mathbb{F}_p$ definieren, die Anzahl der $\mathbb{F}_p$-rationalen Punkte auf E.

1.6. Bemerkung Die Anzahl der $\mathbb{F}_p$-rationalen Punkte einer elliptischen Kurve über einem endlichen Körper, also insbesondere über dem Primkörper $\mathbb{F}_p$, ist endlich, da die Kurvengleichung $y^2 = x^3 + ax + b$ nur endlich viele Lösungen besitzt. Damit macht es Sinn, diese Größe zu bestimmen.

Wir definieren im folgenden Begriffe und Größen, die zur Beschreibung des Algorithmus von Atkin benötigt werden.

1.7. Definition *Für eine beliebige elliptische Kurve $E : y^2 = x^3 + ax + b$, die über dem Körper K ($char(K) \neq 2, 3$) definiert ist, bezeichnen wir als* **j-Invariante von E**

$$j(E) = \frac{1728 \cdot 4a^3}{4a^3 + 27b^2}.$$

1.8. Definition *Sei K ein Körper, $char(K) \neq 2, 3$ und E eine elliptische Kurve über K, so verstehen wir unter dem* **Endomorphismenring $\mathbf{End(E(\overline{K}))}$** *die Gesamtheit aller Gruppenhomomorphismen von $E(\overline{K})$ nach $E(\overline{K})$; für $\phi, \psi \in End(E(\overline{K}))$ und $P \in E(\overline{K})$ definieren wir*

$$\begin{aligned} (\phi + \psi)(P) &:= \phi(P) + \psi(P) \\ (\phi \circ \psi)(P) &:= \phi(\psi(P)). \end{aligned}$$

1.9. Lemma *Der Endomorphismenring von $E(\overline{K})$ ist ein Integritätsbereich mit Charakteristik 0.*

Beweis: [Si86, Prop.4.2, S.71]. ■

Der Begriff des Endomorphismenringes läßt sich auf die Untergruppen von $E(\overline{K})$, z.B. auf die K-rationalen Punkte $E(K)$, übertragen. Gruppenhomomorphismen von elliptischen Kurven haben spezielle Eigenschaften bezüglich ihrer Darstellung; man bezeichnet sie als **Isogenien**, vgl. [Si86, Kap.III, §4].

1.10. Definition *Können wir einen Homomorphismus $i : E_1(\overline{\mathbb{F}_p}) \longrightarrow E_2(\overline{\mathbb{F}_p})$, wobei E_1, E_2 über einem endlichen Erweiterungskörper von $\mathbb{F}_p$ definiert sind, in der Form*

$$i \cdot (x, y) = (h_1(x, y), h_2(x, y)) \;\; \forall \;\; (x, y) \in E_1(\overline{\mathbb{F}_p})$$

mit $h_1, h_2 \in \mathbb{F}_p(x, y)$ schreiben, so bezeichnen wir den Homomorphismus als **$\mathbb{F}_\mathbf{p}$-Isogenie**, *und wir sagen, die Kurve E_1 ist* **$\mathbb{F}_\mathbf{p}$-isogen** *zu E_2.*

Wir haben den Begriff der $\mathbb{F}_p$-Isogenie eingeführt, da wir in Kapitel 2 sehen werden, daß der Atkin-Algorithmus nur dann funktioniert, wenn man die Existenz bestimmter Isogenien ausschließen kann. Weiterhin darf die betrachtete Kurve nicht supersingulär sein.

In unserer Situation der elliptischen Kurve E über dem Primkörper $\mathbb{F}_p$ nennen wir E **supersingulär**, falls

$$\#E(\mathbb{F}_p) = p + 1,$$

(vgl. [Mü94, Abschnitt über Spezialfälle]).

Zur Berechnung der Gruppenordnung von $E(\mathbb{F}_p)$ benötigen wir einen besonderen Endomorphismus der Punktmenge einer elliptischen Kurve. Er bildet die Grundlage der in dieser Arbeit vorgestellten Ideen.

1.11. Definition *Sei E eine elliptische Kurve über dem endlichen Primkörper $\mathbb{F}_p$. Die Abbildung*

$$\begin{aligned} \Phi_E : \quad E(\overline{\mathbb{F}_p}) &\longrightarrow E(\overline{\mathbb{F}_p}) \\ (x, y) &\longmapsto (x^p, y^p) \end{aligned}$$

heißt der **Frobeniusendomorphismus von E**.

1.12. Lemma *Der Frobeniusendomorphismus Φ_E ist ein Element aus dem Endomorphismenring $End(E(\overline{\mathbb{F}_p}))$.*

Beweis: Wir beachten die beiden folgenden Tatsachen:

1. Für $x, y \in \overline{\mathbb{F}_p}$ gilt : $(x + y)^p = x^p + y^p$.
2. Für $\alpha \in \mathbb{F}_p$ gilt : $\alpha^p = \alpha$.

Somit erhalten wir

$$\begin{aligned} (y^p)^2 &= (x^3 + ax + b)^p \\ &= (x^3)^p + (ax)^p + b^p \\ &= (x^p)^3 + ax^p + b \end{aligned}$$

und damit die Abgeschlossenheit. Die Homomorphieeigenschaft erhalten wir durch Anwendung von 1. und 2. auf die Additionsformeln für Punkte aus Satz 1.3. ■

Im Algorithmus von Atkin betrachten wir den Frobeniusendomorphismus zumeist nicht auf der gesamten Punktgruppe $E(\overline{\mathbb{F}_p})$, sondern auf einer speziellen Untergruppe, der n-Torsionsgruppe.

1.13. Definition *Für $n \in \mathbb{N}$ heißt die Menge*

$$E[n] = \left\{ P \in E(\overline{\mathbb{F}_p}) \mid n \cdot P = \mathcal{O} \right\} \tag{1.1}$$

die **n-Torsionsgruppe** *von $E(\overline{\mathbb{F}_p})$.*

1.14. Lemma *Die n-Torsionsgruppe ist eine Untergruppe von $E(\overline{\mathbb{F}_p})$, und der Frobeniusendomorphismus Φ_E eingeschränkt auf $E[n]$ liegt im Endomorphismenring von $E[n]$.*

Beweis:

1. $E[n]$ ist abgeschlossen, da mit $P, Q \in E[n]$ auch $n \cdot (P + Q) = nP + nQ = \mathcal{O}$ gilt. Wegen $n \cdot (-P) = -(nP) = \mathcal{O}$ liegen auch die inversen Elemente in $E[n]$.
2. Für $P \in E[n]$ gilt $n \cdot \Phi_E(P) = \Phi_E(nP) = \Phi_E(\mathcal{O}) = \mathcal{O}$. ■

Eine Aussage über die Struktur der n-Torsionsgruppe liefert uns der folgende Satz. Insbesondere sehen wir, daß $E[n]$ nicht die triviale Untergruppe von $E(\overline{\mathbb{F}_p})$ ist.

1.15. Satz *Ist der $ggT(n, p) = 1$, so ist $E[n] \cong \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$.*

Beweis: [Si86, Cor. 6.4, Seite 89]. ■

Die n-Torsionsgruppen stehen in engem Zusammenhang zu besonderen Polynomen, den Divisionspolynomen. Im Algorithmus von Schoof benutzt man die Divisionspolynome, um die Gruppenordung von $E(\mathbb{F}_p)$ zu berechnen. Wir werden mit den Methoden von Kapitel 4 Teiler dieser Polynome bestimmen und mit Hilfe der Teiler die Gruppenordnung ermitteln.

1.16. Definition *Sei E eine elliptische Kurve über $\mathbb{F}_p$. Für $n \geq -1$ ist das* **n-te Divisionspolynom** *$\psi_n \in \mathbb{F}_p[x, y]$ gegeben als*

- $\psi_{-1}(x, y) = -1$
- $\psi_0(x, y) = 0$
- $\psi_1(x, y) = 1$
- $\psi_2(x, y) = 2y$
- $\psi_3(x, y) = 3x^4 + 6ax^2 + 12bx - a^2$
- $\psi_4(x, y) = 4y(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3)$
- $\psi_{2n}(x, y) = \frac{\psi_n(x,y)}{2y} \cdot (\psi_{n+2}(x, y)\psi_{n-1}^2(x, y) - \psi_{n-2}(x, y)\psi_{n+1}^2(x, y))$ *für $n \geq 3$*
- $\psi_{2n+1}(x, y) = \psi_{n+2}(x, y)\psi_n^3(x, y) - \psi_{n+1}^3(x, y)\psi_{n-1}(x, y)$ *für $n \geq 2$.*

Dabei sind a und b die Konstanten aus der Kurvengleichung $E : y^2 = x^3 + ax + b$.

1.17. Lemma *Für das n-te Divisionspolynom gilt*

$$\psi_n(x,y) \in \begin{cases} y \cdot \mathbb{F}_p[x,y^2], & \textit{falls } n \textit{ gerade ist.} \\ \mathbb{F}_p[x,y^2], & \textit{falls } n \textit{ ungerade ist.} \end{cases}$$

Beweis: Induktion über n mit Fallunterscheidung, ob n gerade oder ungerade. ■

Die Divisionspolynome sind Polynome in den beiden Unbestimmten x und y. Um mit einer Unbestimmten rechnen zu können, beachten wir die Kurvengleichung $E : y^2 = x^3 + ax + b$ und ersetzen alle y^2-Terme. Damit erhalten wir

1.18. Definition *Die* **reduzierten Divisionspolynome** $f_n(x) \in \mathbb{F}_p[x]$ *sind definiert als*

$$f_n(x) = \begin{cases} \frac{1}{y}\left(\psi_n(x,y)/(y^2 - x^3 - ax - b)\right), & \textit{falls } n \textit{ gerade} \\ \psi_n(x,y)/(y^2 - x^3 - ax - b), & \textit{falls } n \textit{ ungerade.} \end{cases}$$

Das n-te reduzierte Divisionspolynom hat als Nullstellen gerade die x-Koordinaten aller Punkte aus der n-Torsionsgruppe. Genauer gilt das folgende

1.19. Lemma *Sei $P = (x,y) \in E(\overline{\mathbb{F}_p})$ aus der Punktmenge der elliptischen Kurve E mit $2 \cdot P \neq \mathcal{O}$ und sei $n \geq -1$. Dann ist $n \cdot P = \mathcal{O}$ genau dann, wenn $f_n(x) = 0$.*

Beweis: [La78] ■

Wir kommen nun zu einem wichtigen Satz, der die Lage der Gruppenordnung $\#E(\mathbb{F}_p)$ beschreibt und die Verbindung zum Frobeniusendomorphismus herstellt. Er wird der Ausgangspunkt für unsere weiteren Überlegungen sein.

1.20. Satz (Hasse) *Sei $c = p + 1 - \#E(\mathbb{F}_p)$. So gilt $|c| \leq 2\sqrt{p}$ und der Frobeniusendomorphismus Φ_E erfüllt im Endomorphismenring von $E(\overline{\mathbb{F}_p})$ die Gleichung*

$$\Phi_E^2 - c \cdot \Phi_E + p = 0. \tag{1.2}$$

Beweis: [Si86, Theorem 1.1, Seite 131]. ■

Hierbei sind die Endomorphismen $i : E(\overline{\mathbb{F}_p}) \longrightarrow E(\overline{\mathbb{F}_p})$ definiert durch

$$i \cdot P = \underbrace{P + \cdots + P}_{i-mal} \text{ für } P \in E(\overline{\mathbb{F}_p}).$$

1.21. Bemerkung Das Element c aus Satz 1.20 bezeichnet man als die **Spur des Frobeniusendomorphismus**. Das Intervall $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$, in dem die Anzahl der $\mathbb{F}_p$-rationalen Punkte liegt, trägt den Namen **Hasse-Intervall**.

Der Satz von Hasse bedeutet für uns, daß wir die Anzahl der $\mathbb{F}_p$-rationalen Punkte einer elliptischen Kurve E bestimmen können, indem wir die Spur c des Frobeniusendomorphismus berechnen. Die Gruppenordnung von $E(\mathbb{F}_p)$ erhalten wir dann als

$$\#E(\mathbb{F}_p) = p + 1 - c. \tag{1.3}$$

Weiterhin ist die Spur c betragsmäßig kleiner gleich $2\sqrt{p}$.

Die Grundüberlegungen im Algorithmus von Atkin sind die gleichen wie im Algorithmus von Schoof, vgl. [Mü91, Kap.5]. Ausgehend vom Frobeniusendomorphismus und dem Satz von Hasse führen wir die Bestimmung der Gruppenordnung $\#E(\mathbb{F}_p)$ auf die Bestimmung der Spur c des Frobeniusendomorphismus zurück. Diese Spur berechnen wir modulo kleiner Primzahlen l und setzen die Kongruenzen mit dem Chinesischen Restsatz zusammen. Im Schoof-Algorithmus erhält man hierbei durch Rechnungen mit Divisionspolynomen c modulo l exakt, wobei die auftretenden Polynome den Grad $\frac{l^2-1}{2}$ besitzen. Im Verfahren von Atkin besitzen die auftretenden Polynome nur den Grad $l+1$. Dafür erhalten wir die Spur c modulo l aber nicht exakt, sondern wir bestimmen eine Liste von Resten modulo l, in der c modulo l enthalten ist. Deshalb müssen wir nach dem Zusammensetzen mittels Chinesischem Restsatz noch einen Babystep-Giantstep-Algorithmus verwenden, um die gesuchte Spur c bestimmen zu können.

Um die Spur des Frobeniusendomorphismus zu berechnen, teilen wir unser weiteres Vorgehen in zwei Phasen auf:

Phase 1 Wir wählen „genügend“ viele, „kleine“, ungerade Primzahlen $l \neq p$ und bestimmen jeweils eine Liste von Kongruenzen modulo l, in welcher die Spur c modulo l enthalten ist.

Phase 2 Wir kombinieren die Reste mit Hilfe des Chinesischen Restsatzes und finden aus der entstehenden Liste mit einem Babystep-Giantstep-Algorithmus die Spur c des Frobeniusendomorphismus heraus.

Diese beiden Phasen werden wir in den beiden nächsten Kapiteln genau beschreiben.

Kapitel 2

Bestimmung von c modulo l

Wir beschreiben in diesem Kapitel, auf welche Art wir Informationen über die Spur c des Frobeniusendomorphismus Φ_E modulo einer ungeraden Primzahl $l \neq p$ gewinnen, d.h. wie wir eine Liste von Kongruenzen modulo l bestimmen können, in der c modulo l enthalten ist. Wir werden sehen, daß der Frobeniusendomorphismus, den wir in Analogie zum Schoof-Algorithmus auf die l-Torsionsgruppe einschränken, einer Gleichung genügt, die c modulo l als Koeffizienten hat. Wir werden Informationen über die Nullstellen dieser Gleichung und damit Informationen über c modulo l erhalten, indem wir den Bezug zu einem bestimmten Polynom, dem l-ten modularen Polynom, herstellen. Wir leiten den Algorithmus zuerst theoretisch her und gehen anschließend auf praxisrelevante Details ein.

Im folgenden bezeichne für unsere elliptische Kurve E über dem Primkörper $\mathbb{F}_p$, $p > 3$, gemäß Definition 1.1, c die Spur des Frobeniusendomorphismus Φ_E sowie l eine ungerade Primzahl ungleich p.

2.1 Die Theorie

Zuerst leiten wir mit Hilfe des Satzes von Hasse (vgl. Satz 1.20) eine Gleichung her, in der die Spur c modulo l als Koeffizient auftritt. Dazu betrachten wir den Frobeniusendomorphismus Φ_E eingeschränkt auf die l-Torsionsgruppe; da im Zusammenhang klar ist, auf welcher Gruppe wir den Frobeniusendomorphismus betrachten, bezeichnen wir im folgenden auch seine Einschränkung mit dem Symbol Φ_E.

2.1. Satz *Die Einschränkung des Frobeniusendomorphismus Φ_E auf die l-Torsionsgruppe $E[l]$ genügt im Endomorphismenring von $E[l]$ der Gleichung*

$$\Phi_E^2 - c_l \cdot \Phi_E + q \equiv 0, \tag{2.1}$$

wobei $c_l \equiv c$ modulo l und $q \equiv p$ modulo l ist.

Beweis:

Nach Satz 1.20 gilt im Endomorphismenring von $E(\overline{\mathbb{F}_p})$

$$\Phi_E^2 - c \cdot \Phi_E + p \equiv 0.$$

Die Endomorphismen Φ_E, c, p liegen im Endomorphismenring von $E[l]$, somit gilt diese Gleichung auch dort. Zerlegen wir $n \in \mathbb{N}$ in der Form $n = k \cdot l + n_l$ mit $0 \leq n_l < l$, so gilt für einen l-Torsionspunkt $Q \in E[l]$:

$$n \cdot Q = (k \cdot l + n_l) \cdot Q = n_l \cdot Q.$$

Wenden wir dies auf c und p an, so erhalten wir Gleichung (2.1). ■

Somit haben wir unsere Aufgabe c modulo l zu bestimmen darauf reduziert, den Wert c_l in der Gleichung (2.1) zu berechnen. Diese Gleichung besagt, daß der Frobeniusendomorphismus Φ_E aus dem Endomorphismenring von $E[l]$ Nullstelle des Polynoms

$$\chi(X) = X^2 - c_l X + q \in \mathbb{F}_l[X] \tag{2.2}$$

ist. Da $\chi(X)$ vom Grad 2 ist, zerfällt dieses Polynom in $\mathbb{F}_{l^2}$, d.h.

$$X^2 - c_l X + q = (X - \alpha) \cdot (X - \beta) \tag{2.3}$$

mit $\alpha, \beta \in \mathbb{F}_{l^2}$.

Die Idee von Atkin ist nun, den Koeffizienten c_l zu bestimmen, indem wir Kenntnis über die Nullstellen α, β gewinnen. Diese Nullstellen können wir durch die folgenden drei, disjunkten Fälle charakterisieren:

1. $\alpha, \beta \in \mathbb{F}_l$ mit $\alpha = \beta$.
2. $\alpha, \beta \in \mathbb{F}_l$ mit $\alpha \neq \beta$.
3. $\alpha, \beta \in \mathbb{F}_{l^2} \backslash \mathbb{F}_l$ mit $\alpha \neq \beta$.

Der Fall $\alpha, \beta \in \mathbb{F}_{l^2} \backslash \mathbb{F}_l$ mit $\alpha = \beta$ kann wegen

$$\alpha = \beta \Rightarrow (X - \alpha)^2 = X^2 - c_l X + q \Rightarrow 2\alpha = c_l \in \mathbb{F}_l$$

nicht auftreten.

Könnten wir die Nullstellen α, β bestimmen, so erhielten wir die gesuchte Spur als $c_l = \alpha + \beta$. Das Verfahren von Atkin liefert aber nicht die Nullstellen direkt, sondern es entscheidet, welcher der drei oben angegebenen Fälle für die Lage von α und β vorliegt; außerdem erhalten wir in den Fällen 2 und 3 die Ordnung d von $\frac{\alpha}{\beta}$ in $\mathbb{F}_l$ bzw. $\mathbb{F}_{l^2}$.

Nehmen wir einmal an, wir wüßten, welcher Fall vorliegt. Dann können wir auf folgende Art und Weise eine Liste von Resten modulo l, die c_l enthält, bestimmen:

1. $\alpha, \beta \in \mathbb{F}_l$ mit $\alpha = \beta$

$$\begin{aligned} \Rightarrow \quad X^2 - c_l X + q = (X-\alpha)^2 &= X^2 - 2\alpha X + \alpha^2 \\ \Rightarrow \quad \alpha = \pm\sqrt{q}, \; c_l &= 2\alpha \\ \Rightarrow \quad c_l &= \pm 2\sqrt{q} \end{aligned}$$

   Wir erhalten also zwei mögliche Werte.

2. $\alpha, \beta \in \mathbb{F}_l$ mit $\alpha \neq \beta$, $ord(\frac{\alpha}{\beta}) = d$

$$\begin{aligned} \Rightarrow X^2 - c_l X + q &= (X-\alpha)\cdot(X-\beta) \\ &= X^2 - 2(\alpha+\beta)X + \alpha\beta \end{aligned}$$

   Wir berechnen nun alle Elemente in $\mathbb{F}_l^*$, die für $\frac{\alpha}{\beta}$ in Frage kommen, d.h. alle Elemente der Ordnung d:

$$\gamma_1, \ldots, \gamma_{\varphi(d)} \in \mathbb{F}_l^*, \;\; ord(\gamma_i) = d,$$

   wobei φ die Euler'sche φ-Funktion ist. Durch Koeffzientenvergleich erhalten wir die folgenden Gleichungen:

$$\alpha \cdot \beta = q, \;\; \alpha + \beta = c_l, \;\; \frac{\alpha}{\beta} = \gamma.$$

   Ersetzen wir α durch $\beta \cdot \gamma$ und lösen nach q bzw. c_l auf, so erhalten wir

$$q = \beta^2 \cdot \gamma, \;\; c_l = \beta \cdot \gamma + \beta.$$

   Berechnen wir β und setzen es in der Gleichung für c_l ein, so gilt:

$$\beta = \pm\sqrt{\frac{q}{\gamma}}, \;\; c_l = \beta \cdot (\gamma + 1),$$

   und wir bekommen die gesuchte Gleichung für c_l als

$$c_l = \pm\sqrt{\frac{q}{\gamma}} \cdot (1+\gamma).$$

   Von jedem Element γ erhalten wir so zwei Werte für c_l. Da mit $\gamma \in \mathbb{F}_l^*$ aber auch γ^{-1} Ordnung d hat, erhalten wir wegen

$$\begin{aligned} \pm\sqrt{\frac{q}{\gamma}} \cdot (1+\gamma) &= \pm\sqrt{\frac{q}{\gamma} \cdot \gamma^2} \cdot (1+\gamma) \cdot \gamma^{-1} \\ &= \pm \cdot \sqrt{\frac{q}{\gamma^{-1}}} \cdot (1+\gamma^{-1}) \end{aligned}$$

   für inverse Elemente die gleichen Werte für c_l und somit genau $\varphi(d)$ viele Kongruenzen für c modulo l.

3. $\alpha, \beta \in \mathbb{F}_{l^2}$ mit $\alpha \neq \beta$, $ord(\frac{\alpha}{\beta}) = d$, analog zu 2. mit Rechnung in $\mathbb{F}_{l^2}$.

Damit haben wir mögliche Werte für c modulo l bestimmt. Wir müssen jetzt einen Weg finden, wie wir die Lage der Nullstellen α und β ermitteln können. Dazu stellt Atkin die Verbindung zu einem Polynom, dem l-ten modularen Polynom, her. Diese Problemreduktion erfolgt in 3 Stufen:

Stufe 1 Die Invarianz von Untergruppen von $E[l]$ unter dem Frobeniusendomorphismus.

Stufe 2 Die Lage von j-Invarianten von speziellen elliptischen Kurven.

Stufe 3 Der Zerfällungstyp des l-ten modularen Polynoms.

Im ersten Schritt stellen wir den Bezug zu den Untergruppen der l-Torsionsgruppe her. Dazu geben wir das folgenden Lemma an, welches uns Einblicke in die Struktur der Torsionsgruppe liefert.

2.2. Lemma *Für eine Primzahl $l \neq p$ hat die l-Torsionsgruppe $E[l]$ folgende Eigenschaften:*

1. *$E[l]$ ist 2-dimensionaler $\mathbb{F}_l$-Vektorraum und $\#E[l] = l^2$.*
2. *Jeder Punkt $Q \in E[l] \backslash \{\mathcal{O}\}$ hat Ordnung l.*
3. $$E[l] = \dot{\bigcup_{1 \leq i \leq l+1}} (C_{l,i} \backslash \{\mathcal{O}\}) \quad \cup \{\mathcal{O}\},$$

   wobei $C_{l,i}$, $1 \leq i \leq l+1$, die zyklischen Untergruppen der Ordnung l von $E[l]$ sind.

Beweis:

1. folgt aus Satz 1.15.

2. gilt, da $ord(Q) > 1$ und $ord(Q) \mid l$.

3. Für $Q_1 \in E[l] \backslash \{\mathcal{O}\}$ definieren wir $C_{l,1} := \langle Q_1 \rangle$ also nach 2. eine zyklische Untergruppe der Ordnung l. Für $Q_2 \in E[l] \backslash C_{l,1}$ definieren wir $C_{l,2} := \langle Q_2 \rangle$. Dann gilt $C_{l,1} \cap C_{l,2} = \{\mathcal{O}\}$, da die Untergruppen Primzahlordnung haben.

Setzen wir das Verfahren fort, so erhalten wir

$$E[l] = \dot{\bigcup_{1 \leq i \leq M}} (C_{l,i} \backslash \{\mathcal{O}\}) \quad \cup \{\mathcal{O}\}.$$

Da $\# (C_{l,i} \backslash \{\mathcal{O}\}) = l - 1$ und $\# (E[l] \backslash \{\mathcal{O}\}) = l^2 - 1$, ist

$$M = \frac{l^2 - 1}{l - 1} = l + 1.$$ ■

Wir betrachten im folgenden Satz den Zusammenhang zwischen der Invarianz der Untergruppen einer l-Torsionsgruppe unter Potenzen des Frobeniusendomorphismus und der Lage der Nullstellen α, β des Polynoms $\chi(X)$.

2.3. Satz *Für* $1 \le i \le l+1$ *sei* $d_i \in \mathbb{N}$ *die kleinste positive Zahl mit* $\Phi_E^{d_i}(C_{l,i}) = C_{l,i}$.

Dann gilt:

1. $\chi(X) = (X-\alpha)^2$

$$\begin{array}{rll} \Rightarrow & \textit{entweder} & d_i = 1, 1 \le i \le l+1, \\ & \textit{oder} & d_j = 1,\ \textit{für genau ein}\ 1 \le j \le l+1 \\ & & \textit{und}\ d_i = l\ \textit{für}\ i \ne j,\ 1 \le i \le l+1, \end{array}$$

d.h. entweder sind alle Untergruppen unter Φ_E *invariant, oder es ist genau eine Untergruppe invariant unter* Φ_E *und alle anderen sind invariant unter* Φ_E^l.

2. $\chi(X) = (X-\alpha)\cdot(X-\beta), \quad \alpha, \beta \in \mathbb{F}_l, \quad \alpha \ne \beta$

$$\begin{array}{rl} \Rightarrow & \textit{es ex.}\ i_1 \ne i_2,\ 1 \le i_1, i_2 \le l+1,\ \textit{mit}\ d_{i_1} = d_{i_2} = 1\ \textit{und für} \\ & 1 \le i \le l+1,\ i \ne i_1, i_2,\ \textit{ist}\ d_i = d = ord(\frac{\alpha}{\beta}),\ d \in \mathbb{N}_{>1}, \end{array}$$

d.h. genau 2 Untergruppen sind invariant unter Φ_E *und alle anderen unter* Φ_E^d.

3. $\chi(X) = (X-\alpha)\cdot(X-\beta), \quad \alpha, \beta \in F_{l^2}, \quad \alpha \ne \beta$

$$\Rightarrow \ \textit{für}\ \ 1 \le i \le l+1\ \textit{ist}\ d_i = d = ord(\tfrac{\alpha}{\beta}),\ d \in \mathbb{N}_{>1},$$

d.h. alle Untergruppen sind invariant unter Φ_E^d.

Beweis: [Mü94, Kap. 3] ■

Können wir also feststellen, wie sich die Untergruppen C_l von $E[l]$ unter dem Frobeniusendomorphismus verhalten, so wissen wir wie die Nullstellen α, β von $\chi(X)$ liegen; außerdem erhalten wir in den Fällen 2. und 3. die Ordnung von $\frac{\alpha}{\beta}$. Wir sollten beachten, daß diese Korrespondenz zwischen Lage der Nullstellen und dem Invarianzverhalten der Untergruppen, wegen der Disjunktheit der Fälle, eine äquivalente Umformung ist.

Um im zweiten Schritt die Werte $d_i,\ 1 \le i \le l+1$, aus Satz 2.3 zu erhalten, betrachten wir die Lage der j-Invarianten von speziellen elliptischen Kurven.

2.4. Lemma *Ist* C *eine endliche Untergruppe von* $E(\overline{\mathbb{F}_p})$, *so existiert eine bis auf Isomorphie eindeutige elliptische Kurve* E/C *und eine Isogenie* $\psi : E(\overline{\mathbb{F}_p}) \to (E/C)(\overline{\mathbb{F}_p})$ *mit* $Kern(\psi) = C$.

Beweis: [Si86, Proposition 4.12, S. 78] ■

Die Schreibweise E/C suggeriert, daß es sich um eine Faktorgruppe handelt. Tatsächlich ist E/C eine elliptische Kurve, die zur formalen Faktorgruppe $E(\overline{\mathbb{F}_p})/C$ isomorph ist. Bezeichnen wir die j-Invariante von E/C mit j/C, so gilt mit den Notationen von Satz 2.3 der folgende

2.5. Satz *Sei E eine nicht supersinguläre Kurve über $\mathbb{F}_p$, so daß keine $\mathbb{F}_p$-Isogenie zu einer elliptischen Kurve mit j-Invariante 0 oder 1728 existiert. Dann gilt für $1 \leq i \leq l+1$:*

$$d_i = min\left\{k \in \mathbb{N} \mid j/C_{l,i} \in \mathbb{F}_{p^k}\right\}, \tag{2.4}$$

wobei d_i die entsprechende Potenz von Φ_E gemäß Satz 2.3 ist.

Beweis: [Mü94, Kor. 3.12]. ■

Also müssen wir zu jedem $C_{l,i} \leq E[l]$ den kleinsten Erweiterungskörper von $\mathbb{F}_p$ bestimmen, in dem die j-Invariante $j/C_{l,i}$ liegt und erhalten den gesuchten Wert d_i als Grad der Körpererweiterung $(1 \leq i \leq l+1)$.

2.6. Bemerkung Anhand dieses Satzes sehen wir, daß 3 Typen von elliptischen Kurven nicht mit diesem Algorithmus behandelt werden können. Ist aber die elliptische Kurve E supersingulär oder $\mathbb{F}_p$-isogen zu einer Kurve mit j-Invariante 0 oder 1728, so stellen wir dies mit einem Verfahren, welches in Abschnitt 2.4 beschrieben wird, fest; in diesen Fällen läßt sich die Gruppenordnung $\#E(\mathbb{F}_p)$ leicht berechnen.

Den <u>dritten Reduktionsschritt</u> liefert uns die Tatsache, daß die j-Invarianten $j/C_{l,i}$, $1 \leq i \leq l+1$, genau die Nullstellen eines Polynoms, des **l-ten modularen Polynoms** zur Kurve E über $\mathbb{F}_p$, sind. Wir können das l-te modulare Polynom $\Phi_l(x,y) \in \mathbb{Z}[x,y]$ über $\mathbb{Z}$ berechnen und erhalten das entsprechende Polynom für unsere Kurve E über $\mathbb{F}_p$, indem wir die y-Koordinate von $\Phi_l(x,y)$ durch die j-Invariante der Kurve ersetzen und die Koeffizienten modulo p reduzieren. Das so berechnete Polynom bezeichnen wir mit $\Phi_l(x, j(E)) \in \mathbb{F}_p[x]$ bzw. nur mit Φ_l. Die Bestimmung des Polynoms über $\mathbb{Z}$ ist unabhängig von der Charakteristik und der Kurve und erfordert deshalb eine einmalige Vorberechnung. Die Herleitung der modularen Polynome ist theoretisch und praktisch aufwendig. Die Theorie hierzu wird eingehend in [Mü94] dargestellt. Die Berechnung modularer Polynome ist der Inhalt von [Le94]; insbesondere wird hier die Implementierung von Algorithmen zur Polynombestimmung beschrieben. Wir werden im folgenden annehmen, daß uns das l-te modulare Polynom $\Phi_l(x,y) \in \mathbb{Z}[x,y]$ zur Verfügung steht.

2.7. Satz *Seien E eine elliptische Kurve über $\mathbb{F}_p$, $p > 3$, die nicht supersingulär oder $\mathbb{F}_p$-isogen zu einer elliptischen Kurve mit j-Invariante 0 oder 1728 ist und $C_{l,i}, 1 \leq i \leq l+1$, alle Untergruppen von $E(\overline{\mathbb{F}_p})$ der Ordnung l. Dann sind genau die j-Invarianten $j/C_{l,i}, 1 \leq i \leq l+1$, die Nullstellen des l-ten modularen Polynoms $\Phi_l(x, j(E)) \in \mathbb{F}_p[x]$, und diese Nullstellen sind alle verschieden.*

Beweis: [Mü94, Abschnitt über modulare Polynome]. ■

Bezeichnen wir für das Polynom $\Phi_l(x, j(E))$ mit

$$(d_1 \;\; d_2 \;\; \dots \;\; d_n) =: \textbf{Zerfällungstyp} \text{ von } \Phi_l(x, j(E))$$

die Tatsache, daß das Polynom n über $\mathbb{F}_p$ irreduzible Faktoren hat, wobei der k-te irreduzible Faktor den Grad d_k besitzt, so erhalten wir aufgrund von Satz 2.3 die folgenden Korrespondenzen:

Lage der Nullstellen	Zerfällungstyp
$\alpha, \beta \in \mathbb{F}_l$ mit $\alpha = \beta$	$\underbrace{(11\ldots1)}_{l+1}$ oder $(1l)$
$\alpha, \beta \in \mathbb{F}_l$ mit $\alpha \neq \beta, ord(\frac{\alpha}{\beta}) = d$	$(11\underbrace{d\ldots d}_{\frac{l-1}{d}})$
$\alpha, \beta \in \mathbb{F}_{l^2} \backslash \mathbb{F}_l$ mit $\alpha \neq \beta, ord(\frac{\alpha}{\beta}) = d$	$\underbrace{(dd\ldots d)}_{\frac{l+1}{d}}$

und dies sind alle Möglichkeiten für die Lage der Nullstellen α, β; also kann auch das l-te modulare Polynom nur auf eine der angegebenen Arten zerfallen. Daß in den Fällen $\alpha \neq \beta$ alle irreduziblen Faktoren von $\Phi_l(x, j(E))$ den gleichen Grad d haben, wird durch folgenden Satz aus der Körpertheorie gerechtfertigt. Dabei beachten wir, daß nach Satz 2.3 alle Nullstellen der irreduziblen Faktoren im gleichen Erweiterungskörper von $\mathbb{F}_p$ liegen.

2.8. Satz *Ist p eine Primzahl, α im algebraischen Abschluß von $\mathbb{F}_p$ und $\phi(X) \in \mathbb{F}_p[X]$ irreduzibel vom Grad $d > 1$ mit $\phi(\alpha) = 0$, so ist $\alpha \in \mathbb{F}_{p^d} \backslash \mathbb{F}_{p^{d-1}}$.*

Beweis: [Mey2, Satz 6.3.3 b, S. 19]. ■

Damit haben wir die Charakterisierung der Nullstellen α, β des Polynoms

$$X^2 - c_l X + q \in \mathbb{F}_l[X]$$

auf die Berechnung des Zerfällungstyps des l-ten modularen Polynoms $\Phi_l(x, j(E)) \in \mathbb{F}_p[x]$ reduziert, und wir können mögliche Werte der Spur c des Frobeniusendomorphismus modulo l berechnen, falls wir Φ_l und dessen Zerfällungsverhalten kennen.

Um den Zerfällungstyp zu bestimmen, wenden wir den folgenden Satz über endliche Körper an.

2.9. Satz *Es sei $p \in \mathbb{N}$ eine Primzahl und $n \in \mathbb{N}$.*

1. *Der Zerfällungskörper von $X^{p^n} - X$ über $\mathbb{F}_p$ hat p^n Elemente.*
2. *Ein Körper K mit p^n Elementen ist der Zerfällungskörper von $X^{p^n} - X$ über seinem Primkörper $\mathbb{F}_p$.*

Beweis: [Mey2, Satz 6.7.2] ■

Wir wissen, daß $\Phi_l(x, j(E)) \in \mathbb{F}_p[x]$ auf eine der 4 Arten

1. $(11 \cdots 1)$
2. $(1l)$
3. $(11d \cdots d)$
4. $(d \cdots d)$

zerfällt. Um zu ermitteln, wieviele Nullstellen von Φ_l in $\mathbb{F}_p$ liegen, berechnen wir

$$ggT(\Phi_l, X^p - X) =: g.$$

In den Fällen 1 und 2 sind wir dann fertig. Im Fall 3 ersetzen wird Φ_l durch $\frac{\Phi_l}{g}$. Danach bestimmen wir $d \geq 2$ als kleinsten Exponenten, so daß alle Nullstellen von Φ_l in $\mathbb{F}_{p^d}$ liegen, d.h.

$$\Phi_l | \left(X^{p^d} - X\right) \Longleftrightarrow X^{p^d} \equiv X \mod \Phi_l$$

und erhalten in den Fällen 3 und 4 den Zerfällungstyp für Φ_l.

2.2 Der Algorithmus in der Praxis

Wir haben gesehen, auf welche Weise die Spur des Frobeniusendomorphismus modulo l mit dem Zerfällungsverhalten des l-ten modularen Polynoms zusammenhängt, und wie wir Informationen über diese Spur berechnen können. Wir geben nun den entsprechenden Algorithmus an und gehen auf die Berechnung von x^{p^d} modulo dem modularen Polynom sowie auf die Bestimmung von Elementen der Ordnung d in $\mathbb{F}_l$ bzw. $\mathbb{F}_{l^2}$ ein.

<u>Berechnung der Spur des Frobeniusendomorphismus modulo einer ungeraden Primzahl</u>

EINGABE: Charakteristik $p > 3$, Kurve E über $\mathbb{F}_p$, ungerade Primzahl $l \neq p$

AUSGABE: Liste von Resten modulo l, unter denen sich die Spur c des Frobeniusendomorphismus modulo l befindet.

(1) Lese das l-te modulare Polynom $\Phi_l(x, y) \in \mathbb{Z}[x, y]$ ein, substituiere y durch die j-Invariante der Kurve E, reduziere die Koeffizienten modulo p und erhalte $\Phi_l(x, j(E)) \in \mathbb{F}_p[x]$.

Berechnung der Spur des Frobeniusendomorphismus
(Fortsetzung)

Bestimme den Zerfällungstyp des l-ten modularen Polynoms

Berechne x^p modulo Φ_l.
$g(x) = x^p - x$ modulo Φ_l.
if $(g(x) == 0)$ **then**
  ZerfTyp = $(11\cdots1)$.
else
  $h(x) = ggT(g(x), \Phi_l)$.
  if $(Grad(h) == 1)$ **then**
    ZerfTyp = $(1l)$.
  else
    if $(Grad(h) == 2)$ **then**
      $g(x) = \frac{\Phi_l}{h(x)}$.
      Berechne das kleinste d, so daß $x^{p^d} \equiv x$ modulo $g(x)$.
      ZerfTyp = $(11d\cdots d)$.
    else
      if $(Grad(h) == 0)$ **then**
        Berechne das kleinste d, so daß $x^{p^d} \equiv x$ modulo Φ_l.
        ZerfTyp = $(d\cdots d)$.
      fi
    fi
  fi
fi

Berechne anhand des Zerfällungstyps mögliche Werte für c_l

if (ZerfTyp == $(11\cdots1)$ oder $(1l)$) **then**
  return ($\pm 2\sqrt{p}$ modulo l.)
else
  if (ZerfTyp == $(11d\cdots d)$) **then**
    Bestimme alle Elemente γ_i, $1 \leq i \leq \varphi(d)$,
    der Ordnung d in $\mathbb{F}_l^*$.
    return ($\pm\sqrt{\frac{p}{\gamma_i}} \cdot (1+\gamma_i)$, $1 \leq i \leq \varphi(d)$.)

Berechnung der Spur des Frobeniusendomorphismus
(Fortsetzung)

(29) **else**
(30) **if** (ZerfTyp == $(11d \cdots d)$) **then**
(31) Bestimme alle Elemente γ_i, $1 \leq i \leq \varphi(d)$,
der Ordnung d in $\mathbb{F}_{l^2}^*$.
(32) **return** $(\pm\sqrt{\frac{p}{\gamma_i}} \cdot (1 + \gamma_i)$, $1 \leq i \leq \varphi(d)$.)
(33) **fi**
(34) **fi**
(35) **fi**

Um diesen Algorithmus implementieren zu können, benötigen wir

- eine Polynomarithmetik für lange Zahlen über $\mathbb{F}_p$.
- die Berechnung von x^p und x^{p^d} modulo einem Polynom.
- Arithmetiken über $\mathbb{F}_l$ und $\mathbb{F}_{l^2}$.
- Wurzelziehen in $\mathbb{F}_l^*$ und $\mathbb{F}_{l^2}^*$.
- die Bestimmung aller Elemente der Ordnung d in $\mathbb{F}_l^*$ bzw. $\mathbb{F}_{l^2}^*$.

Als **Polynomarithmetik** verwenden wir die von Victor Shoup geschriebene Bibliothek GF.a, welche auf der Langzahlarithmetik von Arjen Lenstra aufbaut (vgl. [Sh94_1]). Diese Bibliothek ist in $C++$ geschrieben und somit portabel. Ein wichtiger Vorteil dieser Arithmetik ist die Verwendung von FFT und die ausgefeilte Implementierung des Chinesischen Restsatzes zur Multiplikation von Polynomen.

Zur Berechnung von $\mathbf{x^p}$ **mod** $\mathbf{f(x)}$ verwendet Victor Shoup den Algorithmus der Schnellen Exponentiation in einer modifizierten Form. Normalerweise wird der Algortihmus zur Schnellen Exponentiation so implementiert wie auf Seite 25 angegeben. Schreiben wir eine Binärzahl so, daß das höchste Bit der Zahl links steht, so laufen wir in diesem Algorithmus von rechts nach links über die Binärdarstellung von p. Wir müssen in jedem Schleifendurchlauf das Polynom h quadrieren und das Produkt $erg \cdot h$ modulo f berechnen, falls das entsprechende Bit eins ist; also benötigen wir maximal $2 \cdot log(p)$ viele Polynommultiplikationen. Führen wir nun die Schnelle Exponentiation von links durch, d.h. laufen wir durch p vom höchsten zum niedrigsten Bit, so können wir diese Multplikation $erg \cdot h$ mod f durch eine Multiplikation $erg \cdot x$ mod f ersetzen. Anschaulich basiert dieses Vorgehen auf der Tatsache, daß

$$x^m = \begin{cases} (x^{m/2})^2, & \text{falls } m \text{ gerade.} \\ (x^{(m-1)/2})^2 \cdot x, & \text{falls } m \text{ ungerade.} \end{cases}$$

Schnelle Exponentiation

EINGABE: $p \in \mathbb{N}$, $f(x) \in \mathbb{F}_p[x]$
AUSGABE: $x^p \bmod f(x)$

```
erg = 1; h = x;
while (p > 1) do
   if (p&1) then
      erg = erg · h mod f;
   fi
   h = h · h mod f;
   p = p >> 1;
od
if (p == 1) then
   erg = erg · h mod f;
fi
```

Wir beginnen am niedrigsten Bit von p und führen gemäß der angegebenen Vorschrift eine Zerlegung von x^p durch. Dann starten wir am Zerlegungsende und berechnen x^p gemäß unserer Zerlegung, wobei wir in jedem Einzelschritt quadrieren bzw. mit x multiplizieren. Die Zerlegung ist nur eine theoretische Überlegung. In der Praxis starten wir am höchsten Bit mit dem Polynom $h(x) = x$ und laufen über p bis zum niedrigsten Bit. Gehen wir zu einem niedrigeren Bit über, so quadrieren wir h; ist dieses Bit eins, so multiplizieren wir zusätzlich h mit x.

Beispiel: $p = 11 = (1011)_2$
Wir starten mit $h(x) = x$. Beim Übergang vom vierten zum dritten Bit quadrieren wir h und erhalten x^2. Gehen wir zum zweiten Bit über, so quadrieren wir wieder, multiplizieren zusätzlich mit x und haben somit x^5 berechnet. Genauso verfahren wir im letzten Schritt und erhalten x^{11}.

Die Multiplikation mit x modulo f ist im Vergleich zu einer richtigen Multiplikation billig. Sie entspricht einem Shift mit Skalarmultiplikation und anschließender Addition. Dabei berechnen wir uns das Polynom $x^{deg(f)} \bmod f$ vor.

Multiplikation mit x modulo f(x)

EINGABE: $f(x)$, $x^{deg(f)}$ mod f, $h(x) = \sum_{i=0}^{deg(f)} a_i \cdot x^i$
AUSGABE: $h(x) \cdot x$ mod $f(x)$

(1) **if** $(deg(h) < deg(f) - 1)$ **then** /* keine Reduktion nötig */
(2) $h(x) = \sum_{i=1}^{deg(h)+1} a_{i-1} \cdot x^i$ /* Shift*/
(3) **else**
(4) $h(x) = \sum_{i=0}^{deg(f)-1} c_i \cdot x^i$ /* Shifte h
$c_0 = b_0 \cdot a_{deg(f)-1}$ und reduziere durch
$c_i = a_{i+1} + a_{deg(f)-1} \cdot b_i$ Skalarmult. und Addition */
dabei ist b_i der i-te Koeffizient von $x^{deg(f)}$ mod f
(5) **fi**

Beispiel:
Seien $f(x) = x^3 + x + 1$ über $\mathbb{F}_5$ und $h(x) = 2x^2 + 2x + 3$. Wir berechnen zuerst $x^{deg(f)}$ mod f:

$$x^3 \equiv 4x + 4 \bmod f.$$

Shiften wir $h(x)$, so erhalten wir:

$$\begin{aligned} x \cdot h(x) &= x \cdot (2x^2 + 2x + 3) \\ &= 2x^3 + 2x^2 + 3x. \end{aligned}$$

Wir reduzieren modulo f, indem wir x^3 mod f einsetzen und die Koeffizienten zusammenfassen:

$$\begin{aligned} x \cdot h(x) &\equiv 2 \cdot (4x + 4) + 2x^2 + 3x \bmod f \\ &\equiv 2x^2 + x + 3 \bmod f. \end{aligned}$$

Damit können wir für die Berechnung von x^p modulo $f(x)$ den folgenden Algorithmus formulieren:

x^p mod $f(x)$ durch Schnelle Exponentiation Von Links

EINGABE: $p,\ \ f(x) \in \mathbb{F}_p[x],\ \ x^{deg(f)}$ mod $f(x)$
AUSGABE: x^p mod $f(x)$

```
n = Anzahl der Bits von p;            /* n-tes Bit steht links */
if (n == 0 ) then                     /* Exponent ist 0 */
   erg = 1;
else
   erg = x;                           /* höchstes Bit ist 1 */
   while (n > 1) do
      erg = erg · erg;                /* Übergang zum */
      n − −;                          /* (n − 1)-ten Bit */
      if ( n-tes Bit von p ist 1) then
         erg = erg · x mod f;
      fi
   od
fi
```

Beispiel:

$$\begin{aligned}
f(x) &= x^3 + 2x^2 + 3x + 3 \ \text{ über } \mathbb{F}_5,\ \ p = 11 \\
x &\equiv x \ \bmod\ f \\
x^2 &\equiv x^2 \ \bmod\ f \\
x^4 &\equiv x^2 + 3x + 1 \ \bmod\ f \\
x^5 &\equiv x^2 + 3x + 2 \ \bmod\ f \\
x^{10} &\equiv 2x^2 + 2x + 2 \ \bmod\ f \\
x^{11} &\equiv 3x^2 + x + 4 \ \bmod\ f.
\end{aligned}$$

Nachdem wir x^p modulo Φ_l berechnet haben, bestimmen wir $h(x) := ggT(x^p - x, \Phi_l)$, und im Fall $Grad(h) = 0$ oder $Grad(h) = 2$ suchen wir den kleinsten Exponenten $d \geq 2$, so daß $x^{p^d} \equiv x$ modulo Φ_l. Wir nehmen an, daß wir uns in dieser Situation befinden und setzen

$$f(x) = \begin{cases} \Phi_l(x, j(E)), & \text{falls } Grad(h) = 0, \\ \Phi_l(x, j(E))/h(x), & \text{falls } Grad(h) = 2 \end{cases}$$

sowie $Grad(f) := n$. Wir können annehmen, daß $x^p \bmod f(x)$ berechnet ist, denn im Fall $Grad(h) = 2$ erhalten wir aus $r(x) \equiv x^p \bmod \Phi_l$:

$$\begin{aligned} x^p &= q(x) \cdot \Phi_l + r(x) \\ &= q(x) \cdot h(x) \cdot f(x) + q'(x) \cdot f(x) + r'(x) \\ &= (q(x) \cdot h(x) + q'(x)) \cdot f(x) + r'(x) \end{aligned}$$

mit $0 \leq Grad(r') < Grad(f)$, also $r'(x) \equiv x^p \bmod f(x)$. Wir werden im folgenden beschreiben, wie wir den kleinsten Exponenten $d \geq 2$ mit $x^{p^d} \equiv x \bmod f(x)$ berechnen können.

Zunächst beschäftigen wir uns mit der Berechnung von $x^{p^{i+j}} \bmod f$ aus x^{p^i} und x^{p^j}. Sind $g(x) \equiv x^{p^i} \bmod f$ und $h(x) \equiv x^{p^j} \bmod f$, $i, j \in \mathbb{N}$, vom Grad kleiner n berechnet, so erhalten wir, wegen $g(x)^p = g(x^p)$ (p ist die Charakteristik):

$$x^{p^{i+j}} \equiv_f \left(x^{p^i}\right)^{p^j} \equiv_f (g(x))^{p^j} \equiv_f g\left(x^{p^j}\right) \equiv_f g(h(x)).$$

Zur Berechnung von $x^{p^{i+j}} \bmod f$ aus x^{p^i}, $x^{p^j} \bmod f$ müssen wir also ein Polynom an einem anderen Polynom auswerten. Dazu können wir das in [GaSh92, Abschnitt 2] beschriebene Verfahren benutzen. Wollen wir für einen Wert $m \in \mathbb{N}$ das Polynom $x^{p^m} \bmod f$ berechnen, so können wir wieder das Verfahren der Schnellen Exponentiation verwenden; die dabei auftretenden Operationen sind keine Polynommultiplikationen mehr, sondern Polynomkompositionen.

Um den kleinsten Exponenten $d \geq 2$ zu ermitteln, so daß $x^{p^d} \equiv x \bmod f$ ist, überlegen wir uns zuerst, daß d ein Teiler von $d(f) = n$ sein muß. Adjungieren wir eine Nullstelle ρ von $f(x)$ zu $\mathbb{F}_p$, so ist der Erweiterungsgrad von $\mathbb{F}_p(\rho)/\mathbb{F}_p$ kleiner gleich n, also ist $\mathbb{F}_p(\rho) \subset \mathbb{F}_{p^n}$. Da die Erweiterung $\mathbb{F}_p(\rho)/\mathbb{F}_p$ normal ist, liegen alle Nullstellen von $f(x)$ in $\mathbb{F}_p(\rho)$. Damit liegt der Zerfällungskörper $\mathbb{F}_p(\rho) \cong \mathbb{F}_{p^d}$ von $f(x)$ in $\mathbb{F}_{p^n}$. Aus dem Gradsatz folgt $d \mid n$ (vgl. [Mey2, Satz 6.2.6, S. 14]).

Ein erster Ansatz zur Bestimmung von d wäre, sukzssive alle Potenzen $x^{p^i} \equiv (x^p)^{p^{i-1}} \bmod f$, $i = 2, 3, 4, \ldots, d$, zu berechnen; mit diesem Verfahren würden wir d Polynomkompositionen durchführen. Victor Shoup verwendet in seiner Implementierung ein anderes Verfahren. Zuerst zerlegen wir $d(f) = n$ in seine Primfaktoren:

$$n = \prod_{i=1}^{r} p_r^{e_r}, \quad p_i \in \mathbb{P}, e_i \in \mathbb{N}_0, 1 \leq i \leq r.$$

Zur Vereinfachung nehmen wir erst einmal an, daß in dieser Zerlegung alle Primfaktoren in einfacher Potenz auftreten, d.h. $e_i = 1$, $1 \leq i \leq r$. Dann ist es unser Ziel alle Polynome $x^{p^{n/p_i}}$, $1 \leq i \leq r$, zu bestimmen. Haben wir dies erreicht, so gilt:

$$x^{p^{n/p_i}} \not\equiv x \bmod f \Leftrightarrow p_i \mid d, \quad 1 \leq i \leq r.$$

Da alle Primteiler von d auch Primteiler von n sind, erhalten wir damit die Primfaktorzerlegung von d.

Um die Potenzen $x^{p^{n/p_i}}$, $1 \leq i \leq r$, zu berechnen, bauen wir auf bestimmte Art und Weise einen Baum auf. Wir betrachten dazu das Beispiel in Abbildung 2.1. Dort bezeichnen wir

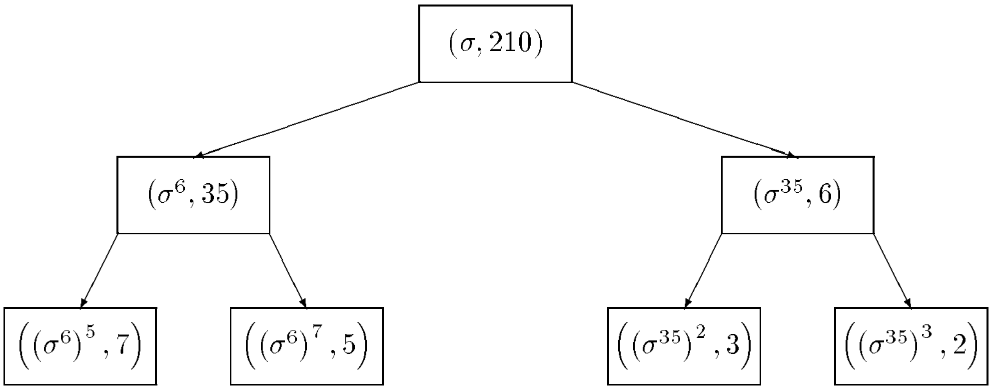


Abbildung 2.1: $n = 210 = 2 \cdot 3 \cdot 5 \cdot 7$.

mit σ^i das Polynom x^{p^i} mod f und definieren $(\sigma^i)^j := \sigma^{i \cdot j}$ $(i, j \in \mathbb{N})$. Wir beginnen an der Wurzel des Baumes und setzen voraus, daß σ bereits bestimmt ist. Im ersten Schritt schreiben wir n als $n = 210 = 6 \cdot 35$ und berechnen σ^6 als auch σ^{35}. Anschließend bestimmen wir $(\sigma^6)^5$, $(\sigma^6)^7$, $(\sigma^{35})^2$ sowie $(\sigma^{35})^3$ und haben somit $\sigma^{n/7}$, $\sigma^{n/5}$, $\sigma^{n/3}$ und $\sigma^{n/2}$ ermittelt.

Allgemein schreiben wir n als Produkt zweier, möglichst gleich großer Faktoren n_1 und n_2, berechnen σ^{n_1} sowie σ^{n_2} und verfahren mit n_1 und n_2 in der gleichen Weise, bis in der Potenz von σ nur noch ein Primfaktor von n fehlt. In jeder Stufe des Baumes ist das Produkt aller neu zu berechnenden Potenzen gerade n; im Beispiel in der ersten Stufe $6 \cdot 35$, in der zweiten $5 \cdot 7 \cdot 2 \cdot 3$. Wir verwenden zur Berechnung der Potenzen jeweils Schnelle Exponentiation und führen in jeder Stufe $\mathcal{O}(log(n))$ viele Polynomkompositionen durch; um beispielsweise $(\sigma^6)^5$ zu berechnen, bestimmen wir $\left(x^{p^6}\right)^{p^6} = x^{p^{12}}$, $\left(x^{p^{12}}\right)^{p^{12}} = x^{p^{24}}$ und $\left(x^{p^{24}}\right)^{p^6} = x^{p^{30}}$. Da der Baum aus $\mathcal{O}(log(r))$ Stufen besteht, erhalten wir insgesamt $\mathcal{O}(log(r) \cdot log(n))$ viele Polynomkompositionen. Wir können folgende rekursive Beschreibung des Verfahrens angeben:

<u>*σPotenz*</u>

EINGABE: (σ^{n_1}, n_2), $n_1, n_2 \in \mathbb{N}$, nur Primfaktoren in einfacher Potenz.
AUSGABE: $(\sigma^{n_1})^{(n_2/p)}$, wobei p alle Primteiler von n_2 durchläuft.

(1) $n_2 = m_1 \cdot m_2$. /* $m_1 \neq 1$ o. $m_2 \neq 1$ */
(2) **if** $(m_1 == 1$ oder $m_2 == 1)$ **then**
(3) Gib σ^{n_1} aus.
(4) **else**
(5) $\sigma Potenz\,((\sigma^{n_1})^{m_1}, m_2)$.
(6) $\sigma Potenz\,((\sigma^{n_1})^{m_2}, m_1)$.
(7) **fi**

Starten wir mit Eingabe (σ, n), so erhalten wir die gesuchten Potenzen $x^{p^{(n/p_i)}}$, $1 \leq i \leq r$. Treten in der Primzahlzerlegung von n Primfaktoren in Potenzen größer eins auf, so können wir auf ähnliche Weise vorgehen. Wir berechnen zuerst

$$\tau_i := \sigma^{n/(p_i^{e_i})}, \ \ 1 \leq i \leq r,$$

mit dem beschriebenen Verfahren; dabei müssen wir nicht mehr die einzelnen Primzahlen, sondern die Primzahlpotenzen isolieren. Anschließend bestimmen wir für jedes τ_i, $1 \leq i \leq r$, sukzessive

$$\tau_i^{p_i^k}, \ \ 1 \leq k \leq e_i,$$

und finden die p_i-Potenz, welche in der Primzahlzerlegung von d auftritt (Beispiel s. Abbildung 2.2).

Wegen

$$\sum_{i=1}^{r} e_i \cdot log\,(p_i) = \sum_{i=1}^{r} log\,(p_i^{e_i}) = log \prod_{i=1}^{r} p_i^{e_i} = log(n),$$

erhalten wir auch hier $\mathcal{O}(log(r) \cdot log(n))$ viele Polynomkompositionen (vgl. [Sh94_2, Lemma 6.1, Theorem 6.2]).

Für die **Arithmetik in $\mathbb{F}_l$** verwenden wir als Darstellung der Restklassen modulo l die kleinsten nicht negativen Reste, d.h. $0, 1, \ldots, l-1$; zum multiplikativen Invertieren von Elementen aus $\mathbb{F}_l^*$ benutzen wir den Erweiterten Euklidischen Algorithmus (vgl. [Mü91, S.31]). Zur Berechnung von Quadratwurzeln verwenden wir den RESSOL-Algorithmus (vgl. [Mü91, S.35]).

Für die **Arithmetik in $\mathbb{F}_{l^2}$** betrachten wir folgenden Zusammenhang:

$$\mathbb{F}_{l^2} \cong \mathbb{F}_l[x]/(x^2 - \rho) \cong \mathbb{F}_l(\lambda)$$

$$\begin{tikzcd}
 & \boxed{(\sigma, 200)} \arrow[dl] \arrow[dr] & \\
\boxed{(\sigma^8, 25)} \arrow[d] & & \boxed{(\sigma^{25}, 8)} \arrow[d] \\
\boxed{\left((\sigma^8)^5, 5\right)} & & \boxed{\left((\sigma^{25})^2, 4\right)} \arrow[d] \\
 & & \boxed{\left((\sigma^{25})^4, 2\right)}
\end{tikzcd}$$

Abbildung 2.2: $n = 2^3 \cdot 5^2 = 8 \cdot 25 = 200$.

mit $\rho \in \mathbb{F}_l, \sqrt{\rho} \notin \mathbb{F}_l, \lambda \in \mathbb{F}_{l^2}, \lambda^2 = \rho$.

Wir wählen uns ein beliebiges Element $\rho \in \mathbb{F}_l^*$ und testen durch Berechnung des Legendre-Symbols $(\frac{\rho}{l})$ (vgl. [Mü91, S.34]), ob ρ ein quadratischer Nichtrest modulo l ist. Da in $\mathbb{F}_l^*$ genauso viele quadratische Reste wie Nichtreste, nämlich $\frac{l-1}{2}$, existieren, können wir erwarten, höchstens zweimal ein Element ρ wählen zu müssen, um die gewünschte Eigenschaft zu erhalten. Da das Polynom $x^2 - \rho \in \mathbb{F}_l[x]$ über $\mathbb{F}_{l^2}$ zerfällt, existiert ein $\lambda \in \mathbb{F}_{l^2}$ mit $\lambda^2 = \rho$.

Damit können wir Elemente $x, y \in \mathbb{F}_{l^2}$ in der Form

$$\begin{aligned} x &= x_1 + x_2 \cdot \lambda \\ y &= y_1 + y_2 \cdot \lambda \end{aligned} \quad \text{mit } x_1, x_2, y_1, y_2 \in \mathbb{F}_l$$

schreiben und in der Implementierung durch die Tupel (x_1, x_2) bzw. (y_1, y_2) repräsentieren.

Wir erhalten:

$$\begin{array}{ccccc} (x_1, x_2) & + & (y_1, y_2) & = & (x_1 + y_1, x_2 + y_2) \\ (x_1, x_2) & \cdot & (y_1, y_2) & = & (x_1 y_1 + x_2 y_2 \rho, x_1 y_2 + x_2 y_1). \end{array}$$

Für $x = (x_1, x_2) \in \mathbb{F}_{l^2}^*$ gilt:

$$\frac{1}{x} = \left(\frac{x_1}{x_1^2 - x_2^2 \rho}, \frac{-x_2}{x_1^2 - x_2^2 \rho} \right).$$

Das Wurzelziehen für Elemente $x = x_1 + x_2 \cdot \lambda \in \mathbb{F}_{l^2}$ mit $x_2 \neq 0$ führen wir auf das Wurzelziehen in $\mathbb{F}_l$ zurück. Schreiben wir $(y_1, y_2) = \sqrt{(x_1, x_2)}$, so erhalten wir

$$\begin{aligned} y_2 &= \pm \sqrt{\frac{x_1}{2\rho} \pm \frac{1}{2} \sqrt{\frac{x_1^2}{\rho^2} - \frac{x_2^2}{\rho}}} \\ y_1 &= \frac{x_2}{2 y_2}; \end{aligned}$$

dabei fassen wir „$\sqrt{z}$“ als die Operation auf, welche testet, ob z modulo l ein Quadrat ist und falls ja, eine Wurzel in $\mathbb{F}_l$ zieht.

Um **alle Elemente der Ordnung d** in $G := \mathbb{F}_l^*$ oder $G := \mathbb{F}_{l^2}^*$ zu bestimmen, berechnen wir zuerst einen Erzeuger a von G; dieser existiert, da die Multiplikativgruppe eines endlichen Körpers stets zyklisch ist. Unser Algorithmus zur Bestimmung eines Erzeugenden beruht auf den folgenden Sätzen.

2.10. Satz *Sei $(G, \cdot)$ eine endliche Gruppe. Dann gilt:*

$$a \in G \text{ erzeugt } G \Longleftrightarrow ord(a) = \#G.$$

Beweis: folgt direkt aus der Definition der Ordnung eines Elementes. ■

2.11. Satz *In einer endlichen Gruppe ist die Ordnung eines jeden Elementes der Gruppe ein Teiler der Gruppenordnung.*

Beweis: [Mey1, S.48, Kor.2] ■

Wir faktorisieren die Gruppenordnung von G ($l-1$ oder l^2-1) und wählen uns solange verschiedene Elemente aus G, bis wir einen Erzeuger gefunden haben. Wir testen, ob ein Element $a \in G$ die Gruppe erzeugt, indem wir für alle echten Teiler d der Gruppenordnung überprüfen, ob $a^d \neq 1_G$. Da in unserem Algorithmus $l < 1000$ ist, können wir die Gruppenordnung durch Probedivision faktorisieren. Haben wir einen Erzeuger a von G gefunden, so können wir alle Elemente aus G mit Ordnung d bestimmen. Es gilt

2.12. Satz *Sei $(G, \cdot)$ eine endliche Gruppe der Ordnung n, $a \in G$ mit $\langle a \rangle = G$. Dann ist für $0 \leq k < n$*

$$ord(a^k) = \frac{n}{ggT(n,k)}.$$

Beweis: [Mey1, Satz 1.6.16, S. 44] ■

Somit haben genau die Elemente $a^{i \cdot (n/d)} \in G$ mit $1 \leq i < d$, $ggT(i,d) = 1$, Ordnung d.

Obwohl dieser Teil des Algorithmus die Gesamtlaufzeit des Atkin-Algorithmus nicht wesentlich beeinflußt, da l klein ist, wollen wir dieses Verfahren noch verbessern, indem wir nicht alle $\varphi(d)$ Elemente der Ordnung d berechnen, sondern nur $\frac{\varphi(d)}{2}$. Dazu betrachten wir die beiden folgenden Sätze.

2.13. Lemma *Ist $l \in \mathbb{N}$ Primzahl, $k \in \mathbb{N}$, so ist $\overline{-1} = \overline{l-1}$ das einzigste Element der Ordnung 2 in $\mathbb{F}_{l^k}^*$.*

Beweis: $\overline{(l-1)}\cdot\overline{(l-1)} = \overline{(l^2-2l+1)} = \overline{1}$; damit hat $\overline{(l-1)}$ Ordnung 2. Da in endlichen, zyklischen Gruppen für jeden Teiler m der Gruppenordnung genau eine Untergruppe mit m Elementen existiert, ist $\left\{\overline{1}, \overline{(l-1)}\right\}$ die einzigste Untergruppe mit 2 Elementen. ■

2.14. Satz *Seien $a \in G = \langle a \rangle$ eine endliche, zyklische Gruppe der Ordnung n, $3 \leq d \in \mathbb{N}$ und $i \cdot \frac{n}{d}$, $1 \leq i \leq d$ mit $ggT(i,d) = 1$, die Exponenten, so daß $a^{i\cdot(n/d)}$ genau die Elemente der Ordnung d in G sind. Sortieren wir diese Exponenten aufsteigend und bezeichnen sie mit $e_1, \ldots, e_{\varphi(d)}$, so ist*

$$a^{e_j} \cdot a^{e_{\varphi(d)-j}} = 1_G, \;\; j = 1, \ldots, \frac{\varphi(d)}{2}.$$

Beweis: Für die Euler'sche φ-Funktion gilt die folgende Gleichung

$$\varphi(d) = d - \sum_{0<m<d} \varphi(m), \;\; d \in \mathbb{N},$$

(vgl. [Mey1, Satz 6.8.2, Korollar, S. 40]). Durch Induktion über d sieht man, daß $\varphi(d)$ für $d \geq 3$ stets gerade ist. Weiterhin gilt für $0 \leq k \leq n$

$$\left(a^k\right)^{-1} = a^{n-k},$$

und da für $1 \leq s, t \leq n$ gilt, daß $n - s \geq n - t$ ist, folgt die Behauptung. ■

Diese Sätze können wir in den Fällen, in denen wir für $d \geq 2$ Elemente der Ordnung d in $G = \mathbb{F}_l^*$ bzw. $G = \mathbb{F}_{l^2}^*$ berechnen, folgendermaßen anwenden:

- $d = 2$. Das einzigste Element der Ordnung 2 ist $\overline{l-1}$ und damit ist

  $$c \equiv 0 \;\text{ modulo }\; l,$$

  vgl. Berechnung von c in Abschnitt 2.1.

- $d \geq 3$. Da multiplikativ inverse Elemente der Ordnung d die gleichen Werte für c modulo l liefern (vgl. Abschnitt 2.1) brauchen wir, unter Berücksichtigung von Satz 2.14, nur die Elemente

  $$a^{e_1}, \ldots, a^{e_{(\varphi(d)/2)}}$$

  berechnen.

Damit können wir den Algorithmus zur Berechnung von Elementen der Ordnung d formulieren.

Elemente der Ordnung d

EINGABE: $G = \mathbb{F}_l^*$ bzw. $G = \mathbb{F}_{l^2}^*$, d

AUSGABE: $x_1, \ldots, x_{\frac{\varphi(d)}{2}}$, Elemente der Ordnung d

BESTIMME ERZEUGER $a \in G$

(1) $n = \#G$. /* $n = l - 1$ bzw. $n = l^2 - 1$ */
(2) Berechne alle echten Teiler $d_1, \ldots, d_m$ von n. /* Faktorisieren durch Probedivision */
(3) Wähle $a \in G$ beliebig.
(4) **do**
(5) $i = 1$. /* Berechne die Ordnung von a */
(6) **while** $(i \leq m$ und $a^{d_i} \neq 1_G)$ **do**
(7) $i = i + 1$.
(8) **od**
(9) **if** $(i \leq m$ /* zu kleine Ordnung */$)$ **then**
(10) wähle neues Element $a \in G$.
(11) **fi**
(12) **while** $(i \leq m)$

BESTIMME ELEMENTE DER ORDNUNG d

(13) Berechne $i_1 \cdot \frac{n}{d}, \ldots, i_{\varphi(d)} \cdot \frac{n}{d}$
mit $1 \leq i_j < d$ teilerfremd zu d.
(14) Sortiere diese Elemente aufsteigend und benenne sie $e_1, \ldots, e_{\varphi(d)}$.
(15) Berechne $x_1 = a^{e_1}$.
(16) **for** $(j = 2;\ j \leq \frac{\varphi(d)}{2};\ j++)$ **do**
(17) $x_j = x_{j-1} \cdot a^{e_j - e_{j-1}}$.
(18) **od**
(19) **return** $(x_1, \ldots, x_{\frac{\varphi(d)}{2}})$

Damit haben wir alle Einzelalgorithmen zusammengetragen, um Informationen über die Spur c des Frobeniusendomorphismus modulo l berechnen zu können. Wir geben ein Beispiel für die Spurberechnung an.

Beispiel: $p = 109$, $E = (3011, 68)$, j-Invariante $j(E) = 62$

- $l = 11$

  Wir lesen das modulare Polynom $\Phi_{11}(x,y) \in \mathbb{Z}[x,y]$ ein, reduzieren die Koeffizienten modulo 109, substituieren die j-Invariante und erhalten

$$\begin{aligned}\Phi_{11}(x,62) &= x^{12}+55x^{11}+59x^{10}+90x^9+73x^8\\ &+60x^7+83x^6+57x^5+64x^4+8x^3\\ &+104x^2+86x+93.\end{aligned}$$

  Wir berechnen $x^{109} \bmod \Phi_{11}(x,62)$ und erhalten

$$ggT(x^{109}-x,\ \ \Phi_{11}(x,62)) = x^2+51x+29.$$

  Also liegen 2 Nullstellen von Φ_l in $\mathbb{F}_{109}$; dividieren wir die beiden Nullstellen heraus, dann erhalten wir $d = 10$ als kleinsten Exponenten, so daß dieses Polynom über $\mathbb{F}_{p^d}$ zerfällt. Daher ist der Zerfällungstyp

$$(1\ \ 1\ \ 10).$$

  Wir finden für $\mathbb{F}_{11}^*$ den Erzeuger $\overline{2}$ und erhalten als Elemente der Ordnung 10 $\overline{2}$ und $\overline{8}$. Damit bekommen wir als mögliche Werte für die Spur c modulo 11

$$\overline{1}\ \ \overline{4}\ \ \overline{7}\ \ \overline{10}.$$

- $l = 13$

  Als Polynom erhalten wir

$$\begin{aligned}\Phi_{13}(x,62) &= x^{14}+26x^{13}+107x^{12}+41x^{11}+98x^{10}\\ &+58x^9+49x^8+40x^7+9x^6+66x^5\\ &+68x^4+47x^3+103x^2+30x+13,\end{aligned}$$

  und der ggT mit $x^{109}-x$ ergibt sich als

$$ggT(x^{109}-x,\ \ \Phi_{13}(x,62)) = 1,$$

  also liegt keine Nullstelle in $\mathbb{F}_{109}$. $\Phi_{13}(x,62)$ zerfällt über $\mathbb{F}_{p^{14}}$ und besitzt somit den Zerfällungstyp (14).
  Wir finden den quadratischen Nichtrest $\overline{2} \in \mathbb{F}_{13}$, und ein Erzeuger von $\mathbb{F}_{13^2}^*$ ist das Element $(\overline{1},\overline{2})$; Elemente der Ordnung 14 sind

$$\begin{aligned}(\overline{1},\overline{2})^{12} &= (\overline{8},\overline{8})\\ (\overline{1},\overline{2})^{36} &= (\overline{9},\overline{12})\\ (\overline{1},\overline{2})^{60} &= (\overline{3},\overline{11}).\end{aligned}$$

  Daraus ergeben sich für c modulo 13 die möglichen Werte

$$\overline{1}\ \ \overline{3}\ \ \overline{5}\ \ \overline{8}\ \ \overline{10}\ \ \overline{12}.$$

2.3 Laufzeiten

In diesem Abschnitt geben wir einige Laufzeiten für die Berechnung des Zerfällungstyps eines modularen Polynoms $\Phi_l(x, j(E))$ an, d.h. für die Bestimmung von $x^p \bmod \Phi_l(x, j(E))$ sowie des Zerfällungsgrades d. Für die Spurberechnung, die sich an die Bestimmung des Zerfällungstyps anschließt, brauchen wir keine Laufzeittabellen anzugeben; bei Charakteristiken mit 250 Dezimalstellen und l ungefähr 500 liegt die Berechnungsdauer unter 2 Sekunden.

Wir beginnen mit der Bestimmung von x^p modulo $\Phi_l(x, j(E))$ über $\mathbb{F}_p$, wobei Φ_l den Grad $l+1$ besitzt. In der ersten Spalte stehen jeweils die verschiedenen Werte für l. Die Zahlen in der Kopfzeile (30,40,...) geben jeweils die Anzahl der Dezimalstellen der Charakteristik p an; die Zeiten in den ersten beiden Tabellen sind in Sekunden angegeben.

l	30	40	50	60
11	1	2	3	5
23	3	8	13	19
31	6	13	21	32
41	9	19	29	39
53		21	32	43
61		22	34	46
73		38	59	79
83			75	96
97				89
107				96

l	70	80	90	100
11	7	11	15	19
23	26	41	55	69
31	45	70	93	118
41	55	74	95	118
53	61	83	108	132
61	66	89	116	144
73	113	150	193	238
83	132	169	218	266
97	127	167	232	270
107	135	179	251	286
127	145	192	404	308
131		317	419	499
149		324	437	512
167				540
191				566
211				596
233				632

l	150	200	250	375
53	6 min	12 min	22 min	45 min
101	12 min	25 min	41 min	1 h 50 min
151	22 min	44 min	1 h 11 min	2 h 44 min
199	25 min	49 min	1 h 21 min	3 h 27 min
257	43 min	1 h 25 min	2 h 18 min	4 h 10 min
307	46 min	1 h 31 min	2 h 29 min	6 h 18 min
353		1 h 36 min	2 h 41 min	7 h 03 min
401		1 h 43 min	2 h 53 min	7 h 39 min
449		1 h 55 min	3 h 05 min	8 h 17 min
503			3 h 14 min	8 h 58 min
569				10 h 42 min
601				11 h 03 min
653				11 h 20 min
701				14 h 33 min
751				15 h 24 min
787				15 h 38 min

An diesen Zeiten kann man gut erkennen, daß sich die FFT-Multiplikation an Zweier-Potenzen orientiert. Wenn der Polynomgrad die nächste Zweier-Potenz übersteigt, also eine doppelt so große Punkt-Wert Repräsentation für die Polynome notwendig wird, ist jeweils ein Sprung in den Berechnungszeiten festzustellen.

Als nächstes betrachten wir einige Laufzeiten zur Berechnung des Zerfällungsgrades d des modularen Polynoms $\Phi_l(x, j(E))$ unter der Voraussetzung, daß $x^p \bmod \Phi_l(x, j(E))$ berechnet ist. In der ersten Spalte stehen wieder die verschiedenen Werte für l. Für jede Charakterstik, die jeweils durch die Anzahl ihrer Dezimalstellen beschrieben ist, sind zwei Spalten angegeben. In der ersten steht jeweils der Zerfällungsgrad d. In der zweiten steht die Zeit zur Berechnung dieses Grades; wiederum sind die Zeiten in den beiden ersten Tabellen in Sekunden angegeben.

	30		40		50		60	
l	d	Zeit	d	Zeit	d	Zeit	d	Zeit
11	12	0	10	0	12	0	4	0
23	24	2	8	4	12	4	2	2
31	8	2	6	7	32	7	5	8
41	8	5	21	8	40	9	40	12
53			27	19	2	6	18	27
61			62	15	31	20	31	24
73			4	15	74	34	37	41
83					82	40	21	51
97							32	76
107							106	73

	70		80		90		100	
l	d	Zeit	d	Zeit	d	Zeit	d	Zeit
11	4	0	–	–	10	5	12	1
23	8	7	4	8	4	9	11	10
31	3	14	5	13	15	23	30	24
41	21	15	21	19	40	20	3	18
53	52	19	9	43	52	29	52	25
61	31	29	62	38	62	45	60	46
73	2	14	6	59	37	74	9	56
83	82	57	41	72	21	91	84	88
97	98	99	7	122	7	144	96	120
107	106	88	106	113	106	136	9	140
127	18	116	128	80	32	78	64	103
131			13	145	130	241	132	218
149			74	237	148	279	75	323
167					84	429	21	312
191							190	350
211							35	461
233							13	299

	150		200		250		375	
l	d	Zeit	d	Zeit	d	Zeit	d	Zeit
53	18	29 sec	52	2 min	27	4 min	27	4 min
101	5	88 sec	51	5 min	102	8 min	100	17 min
151	76	146 sec	150	17 min	152	16 min	50	28 min
199	25	229 sec	22	18 min	198	31 min	50	40 min
257	258	502 sec	64	24 min	16	15 min	256	51 min
307			51	60 min	308	1 h 36 min	153	2 h 07 min
353			177	76 min	177	1 h 56 min	177	2 h 14 min
401					201	2 h 11 min	400	2 h 32 min
449					448	2 h 43 min	9	3 h 34 min
503					504	3 h 09 min	251	4 h 11 min
601							300	6 h 25 min
653							109	5 h 01 min
787							393	9 h 35 min

2.4 Behandlung der Spezialfälle

Wir haben bei der Herleitung von Atkins's Grundalgorithmus gesehen, daß die Kurve E über $\mathbb{F}_p$ nicht supersingulär und nicht $\mathbb{F}_p$-isogen zu einer Kurve mit j-Invariante 0 oder 1728 sein darf. In diesem Abschnitt werden wir erklären, wie wir bestimmen können, ob einer dieser Spezialfälle für die Kurve E zutrifft, und wie wir in diesen Fällen die Gruppenordnung berechnen. Wir nutzen aus, daß die Spezialfälle dazu äquivalent sind, daß die Gruppenordnung $\#E(\mathbb{F}_p)$ bestimmte Werte annimmt. Wir berechnen diese Werte und bestimmen mit einem Ausscheidungsverfahren, ob einer dieser Werte die Gruppenordnung ist. Falls dies zutrifft, sind wir fertig; ansonsten sind die Voraussetzungen des Atkin-Algorithmus erfüllt, und wir können mit ihm die gesuchte Gruppenordnung berechnen.

Wir bestimmen zuerst die möglichen Werte der Gruppenordnung, die zu den Spezialfällen korrespondieren. Anhand der Definition aus Kapitel 1 wissen wir, daß die Supersingularität von E dazu äquivalent ist, daß $\#E(\mathbb{F}_p) = p + 1$. Für den anderen Spezialfall verwenden wir den folgenden

2.15. Satz *Sei $\gamma \in \mathbb{C}$ mit $\gamma^2 + \gamma + 1 = 0$, $i = \sqrt{-1} \in \mathbb{C}$ und E $\mathbb{F}_p$-isogen zu einer Kurve mit j-Invariante 0 oder 1728. Dann ist $\#E(\mathbb{F}_p)$ von folgender Form:*

$$\#E(\mathbb{F}_p) = p + 1 - \mathit{Spur}(z),$$

mit

$$z \in \{\pm\pi, \pm i\pi\},\ \textit{wobei } \pi \in \mathbb{Z}[i] \textit{ ein Element der Norm } p \textit{ ist,}$$

$$\textit{oder}$$

$$z \in \{\pm\tau, \pm\gamma\tau, \pm\gamma^2\tau\},\ \textit{wobei } \tau \in \mathbb{Z}[\gamma] \textit{ ein Element der Norm } p \textit{ ist.}$$

Beweis: [Mü94, Abschnitt über Spezialfälle]. ■

Wir müssen also je ein Element der Norm p in $\mathbb{Z}[i]$ oder $\mathbb{Z}[\gamma]$ bestimmen und erhalten dann die in Satz 2.15 angegebenen möglichen Werte für $\#E(\mathbb{F}_p)$. Definieren wir für $D \in \mathbb{Z}$, $D \equiv 0, 1 \bmod 4$, D kein Quadrat in $\mathbb{Z}$, das Element

$$\omega = \frac{D + \sqrt{D}}{2},$$

so können wir die Ringe $\mathbb{Z}[i]$ bzw. $\mathbb{Z}[\gamma]$ folgendermaßen darstellen:

$$\begin{aligned} \mathbb{Z}[i] &= \mathbb{Z}[\omega] \text{ mit } D = -4, \\ \mathbb{Z}[\gamma] &= \mathbb{Z}[\omega] \text{ mit } D = -3. \end{aligned}$$

Ein Element $z = (x + y\omega) \in \mathbb{Z}[\omega]$ hat als Spur (Sp) und Norm (N) folgende Werte:

$$Sp(z) = 2x + yD,\ \ N(z) = Sp(z) + y^2 \cdot \frac{D \cdot (D-1)}{4},$$

und z hat genau dann Norm p, falls gilt:

$$4p = (2x + yD)^2 - y^2 D \tag{2.5}$$

(vgl. [BuMü92, Abschnitt 3.4]).

Um Elemente der Norm p in $\mathbb{Z}[\omega]$ zu bestimmen, verwenden wir die in [Co93] beschriebene Funktion *cornacchia()*; diese erhält als Parameter D und p und liefert $(x_1, y_1) \in \mathbb{Z}^2$ mit

$$x_1^2 + |D|y_1^2 = 4p,$$

falls ein solches Paar existiert. Setzen wir also für $z = (x + y\omega) \in \mathbb{Z}[\omega]$

$$x = \frac{x_1 - y_1 D}{2},\ \ y = y_1,$$

so sehen wir anhand von Gleichung (2.5), daß z ein Element der Norm p in $\mathbb{Z}[\omega]$ ist. Starten wir die *cornacchia*-Funktion mit den Parametern $(-3, p)$, so können wir ein Element z der Norm p in $\mathbb{Z}[\gamma]$ bestimmen; mit $(-4, p)$ erhalten wir ein entsprechendes Element z in $\mathbb{Z}[i]$.

Um die verschiedenen Spuren aus Satz 2.15 zu ermitteln, berechnen wir zuerst die Spuren von ω und ω^2:

$$\begin{aligned} Sp(\omega) &= \frac{D + \sqrt{D}}{2} + \frac{D - \sqrt{D}}{2} = D, \\ Sp\left(\omega^2\right) &= \frac{1}{4} \cdot Sp\left(D^3 + 3D^2\sqrt{D} + 3D^2 + \sqrt{D}^3\right) \\ &= \frac{D^3 + 3D^2}{4}. \end{aligned}$$

Wir berechnen nun alle Werte für die Spur c, die sich aus den Spuren der Elemente mit Norm p in $\mathbb{Z}[i]$ ergeben. Seien dazu $D = -4$, $(x_1, y_1) \in \mathbb{Z}^2$ mit $x_1^2 + |D|y_1^2 = 4p$ und $z = x + y\omega = \frac{1}{2}(x_1 - y_1 D) + y_1\omega \in \mathbb{Z}[\omega]$ ein Element der Norm p. Dann gilt:

$\pm z$: $Sp(z) = 2x + yD = x_1$ und somit $c = \pm x_1$.

$\pm iz$: $\omega = \frac{1}{2} \cdot (-4 + 2i) = i - 2$ und damit $i = \omega + 2$. Für die Spur erhalten wir:

$$\begin{aligned} Sp(i \cdot z) &= Sp((\omega + 2) \cdot (x + y\omega)) \\ &= Sp(\omega \cdot (x + y\omega)) + 2 \cdot Sp(x + y\omega) \\ &= x \cdot Sp(\omega) + y \cdot Sp(\omega^2) + 2 \cdot (2x + yD) \\ &= -2y_1 \end{aligned}$$

und somit $c = \pm 2y_1$.

Damit erhalten wir 4 mögliche Werte für die Gruppenordnung.

Zur Berechnung der Spuren in $\mathbb{Z}[\gamma]$ seien $D = -3$, $(x_1, y_1) \in \mathbb{Z}^2$ mit $x_1^2 + |D|y_1^2 = 4p$ und $z = x + y\omega = \frac{1}{2}(x_1 - y_1 D) + y_1\omega \in \mathbb{Z}[\omega]$ ein Element der Norm p. Wir wissen, daß γ Nullstelle der Gleichung $x^2 + x + 1$ ist; also können wir γ als

$$\gamma = \frac{-1 + \sqrt{-3}}{2}$$

wählen und erhalten

$$\gamma = \omega + 1, \;\; \gamma^2 = -(\omega + 2).$$

Für c erhalten wir die folgenden Werte:

$\pm z$: $Sp(z) = 2x + yD = x_1$ und somit $c = \pm x_1$.

$\pm \gamma z$:

$$\begin{aligned} Sp(\gamma z) &= Sp((\omega + 1) \cdot z) \\ &= Sp(\omega \cdot z) + Sp(z) \\ &= x \cdot Sp(\omega) + y \cdot Sp(\omega^2) + 2x + yD \\ &= -\frac{1}{2} \cdot (x_1 + 3y_1) \end{aligned}$$

und somit $c = \pm\frac{1}{2} \cdot (3y_1 + x_1)$.

$\pm \gamma^2 z$:

$$\begin{aligned} -Sp(\gamma^2 z) &= -Sp(-(\omega + 2) \cdot z) \\ &= Sp(\omega z) + 2 \cdot Sp(z) \\ &= x \cdot Sp(\omega) + y \cdot Sp(\omega^2) + 4x + 2yD \\ &= \frac{1}{2} \cdot (x_1 - 3y_1) \end{aligned}$$

und somit $c = \pm\frac{1}{2} \cdot (3y_1 - x_1)$.

Wir erhalten 6 mögliche Werte für die Gruppenordnung.

Wir gehen im folgenden davon aus, daß uns eine Liste von Werten $w_1, \ldots, w_n$ im Hasse-Intervall vorliegt, für die wir überprüfen müssen, ob sie die Gruppenordnung $\#E(\mathbb{F}_p)$ beinhaltet. Dazu verwenden wir ein Ausscheidungsverfahren, welches auf der praktischen Erfahrung beruht, daß für einen zufälligen Punkt $Q \in E(\mathbb{F}_p)$ und einen Wert w im Hasse-Intervall, der nicht die Gruppenordnung ist, fast immer $w \cdot Q \neq \mathcal{O}$ gilt, d.h. w ist kein Vielfaches der Punktordnung von Q. Wir wählen uns solange verschiedene Punkte $Q \in \mathbb{E}(\mathbb{F}_p)$, bis von den Werten $w_1, \ldots, w_n$ nur noch ein oder kein Wert übriggeblieben ist. Ist kein Wert übriggeblieben, so kann für die Kurve E kein Speziallfall vorliegen. Ist ein Wert w übrig, so verwenden wir den in [Mü94, Abschnitt über Beweis der Gruppenordnung] beschriebenen Algorithmus, um zu überprüfen, ob w die Gruppenordnung ist.

Wir verbessern das Ausscheidungsverfahren, indem wir einen Kandidaten w nicht nur auf der Kurve E, sondern zusätzlich auf einer anderen Kurve, dem sogenannten Twist von E, testen.

2.16. Definition *Sei $E = (a, b)$ eine elliptische Kurve über $\mathbb{F}_p$ und $d \in \mathbb{F}_p^*$ ein quadratischer Nichtrest. Dann bezeichnen wir eine Kurve $E' = (d^2a, d^3b)$ als* **Twist von E**, *in Zeichen $twistE$.*

Der Twist von E besitzt die schöne Eigenschaft, daß seine Anzahl der $\mathbb{F}_p$-rationalen Punkte in engem Zusammenhang zu $\#E(\mathbb{F}_p)$ steht.

2.17. Lemma *Ist für E über $\mathbb{F}_p$ die Anzahl der $\mathbb{F}_p$-rationalen Punkte $\#E(\mathbb{F}_p) = p+1-c$, so erhalten wir für den Twist von E*

$$\#twistE(\mathbb{F}_p) = p + 1 + c.$$

Beweis:

Wir können $\#E(\mathbb{F}_p)$ schreiben als

$$\#E(\mathbb{F}_p) = 1 + \sum_{x \in \mathbb{F}_p} V\left(x^3 + ax + b\right)$$

wobei für $\alpha \in \mathbb{Z}$ gilt:

$$V(\alpha) = \begin{cases} 1, & \alpha \equiv 0 \bmod p, \\ 2, & \alpha \text{ ist Quadrat mod } p, \\ 0, & \text{sonst.} \end{cases}$$

Vergleichen wir dies mit dem Legendre-Symbol

$$\left(\frac{\alpha}{p}\right) = \begin{cases} 0, & \alpha \equiv 0 \bmod p \\ 1, & \alpha \text{ ist Quadrat mod } p \\ -1, & \text{sonst,} \end{cases}$$

so sehen wir, daß

$$\begin{aligned}\#E(\mathbb{F}_p) &= p+1-c \\ &= 1+\sum_{x\in\mathbb{F}_p^*} V\left(x^3+ax+b\right) \\ &= 1+\sum_{x\in\mathbb{F}_p^*}\left(\left(\frac{x^3+ax+b}{p}\right)+1\right) \\ &= p+1+\sum_{x\in\mathbb{F}_p^*}\left(\frac{x^3+ax+b}{p}\right).\end{aligned}$$

Sei $d \in \mathbb{F}_p^*$ ein quadratischer Nichtrest. Können wir zeigen, daß

$$-\sum_{x\in\mathbb{F}_p^*}\left(\frac{x^3+ax+b}{p}\right) = \sum_{x\in\mathbb{F}_p^*}\left(\frac{x^3+d^2ax+d^3b}{p}\right),$$

so erhalten wir die Behauptung. Klammern wir d^3 aus und verwenden wir die Multiplikativität des Legendre-Symbols, so erhalten wir:

$$\sum_{x\in\mathbb{F}_p^*}\left(\frac{x^3+d^2ax+d^3b}{p}\right) = \sum_{x\in\mathbb{F}_p^*}\left(\frac{d^3}{p}\right)\left(\frac{(x/d)^3+a(x/d)+(b/d)}{p}\right).$$

Da d ein quadratischer Nichtrest und $d \not\equiv 0$ ist, gilt $\left(\frac{d^3}{p}\right) = -1$. Weiterhin durchläuft mit x auch x/d ganz $\mathbb{F}_p$ und deshalb gilt:

$$\sum_{x\in\mathbb{F}_p^*}\left(\frac{(x/d)^3+a(x/d)+(b/d)}{p}\right) = \sum_{x\in\mathbb{F}_p^*}\left(\frac{x^3+ax+b}{p}\right).$$

Daraus folgt die Behauptung. ■

2.18. Bemerkung Die im Beweis hergeleitete Beziehung

$$\#E(\mathbb{F}_p) = p+1+\sum_{x\in\mathbb{F}_p^*}\left(\frac{x^3+ax+b}{p}\right)$$

kann man verwenden, um die Gruppenordnung zu berechnen. Dieser sogenannte Charaktersummen-Algorithmus ([Mü91, Kap.1, S.12]) ist allerdings nur für Charakteristiken p kleiner 1000 praktikabel. Den quadratischen Nichtrest $d \in \mathbb{F}_p^*$ bestimmen wir durch zufällige Auswahl von Elementen, wie wir dies schon in Abschnitt 2.2 zur Charakterisierung von $\mathbb{F}_{l^2}$ erläutert haben.

Wir erhalten den folgenden Algorithmus zur Überprüfung der Spezialfälle:

Überprüfung der Spezialfälle

EINGABE: Kurve E über $\mathbb{F}_p$.

AUSGABE: $\#E(\mathbb{F}_p)$ oder „Kein Spezialfall“.

MÖGLICHE GRUPPENORDNUNGEN.

```
n = 1.
ω_1 = p + 1.
if (cornacchia(x_1, y_1, −3, p)) then
    c_1 = x_1.
    c_2 = 1/2 · (3y_1 + x_1).
    c_3 = 1/2 · (3y_1 − x_1).
    for (i = 1; i ≤ 3; i + +) do
        w_{n+2i−1} = p + 1 + c_i.
        w_{n+2i} = p + 1 − c_i.
    od
    n = n + 6.
fi
if (cornacchia(x_1, y_1, −4, p)) then
    c_1 = x_1.
    c_2 = 2y_1.
    for (i = 1; i ≤ 2; i + +) do
        w_{n+2i−1} = p + 1 + c_i.
        w_{n+2i} = p + 1 − c_i.
    od
    n = n + 4.
fi
L := {ω_1, ..., ω_n}.
```

Überprüfung der Spezialfälle
(Fortsetzung)

Überprüfung der möglichen Gruppenordnungen.

```
Finde quadratischen Nichtrest d in F_p.
Berechne twistE = (d^2 a, d^3 b).
do
  for (i = 0; i < 10; i++) do
    Wähle neuen Punkt Q_1 ∈ E(F_p) zufällig.
    Wähle neuen Punkt Q_2 ∈ twistE(F_p) zufällig.
    for (j = 1; j ≤ #L; j++) do
      z := j-tes Element aus L.
      if (z · Q_1 ≠ O) then
        Streiche z aus L.
      else
        z := 2 · (p + 1) − z.
        if (z · Q_2 ≠ O) then
          Streiche z aus L.
        fi
      fi
    od
  od
while (#L > 1)
if (#L == 0) then
   return („Kein Spezialfall“)
else
  Versuche zu beweisen, daß z ∈ L die Gruppenordnung ist.
  if (z ist Gruppenordnung) then
    return (#E(F_p) = z.)
  else
    return („Kein Spezialfall“)
  fi
fi
```

Das Ausscheidungsverfahren liefert uns nur dann einen oder keinen Wert, wenn kein echtes Vielfaches der Gruppenordnung im Hasse-Intervall liegt. Dieser Fall kann nur dann auftreten, wenn die Charakteristik p eine gewisse Größe unterschreitet:

$$\begin{aligned} & 2 \cdot (p + 1 - 2\sqrt{p}) > p + 1 + 2\sqrt{p} \\ \Leftrightarrow \quad & p + 1 - 6\sqrt{p} > 0 \\ \Rightarrow \quad & \sqrt{p} > 3 + \sqrt{8}. \end{aligned}$$

Wegen $3 + \sqrt{8} \approx 6$, kann für $p \geq 37$ nur ein Vielfaches der Gruppenordnung im Hasse-Intervall liegen. Für $p < 37$ können wir den erwähnten Charaktersummen-Algorithmus verwenden.

Kapitel 3

Die Babystep-Giantstep-Phase

3.1 Die theoretische Herleitung

Im vorangegangenen Kapitel haben wir beschrieben, wie wir zur Kurve E mögliche Werte der Spur c des Frobeniusendomorphismus modulo einer ungeraden Primzahl $l \neq p$ berechnen können. Wir werden nun erklären, auf welche Weise wir mit diesen Informationen die Spur c und damit die Gruppenordnung $\#E(\mathbb{F}_p) = p + 1 - c$ bestimmen können. Da wir bei diesem Verfahren einen probabilistischen Test verwenden, ist die auf diese Weise ermittelte Anzahl der $\mathbb{F}_p$-rationalen Punkte nur mit hoher Wahrscheinlichkeit richtig; die Korrekheit muß in einem weiteren Schritt noch bewiesen werden. In der Praxis waren die berechneten Gruppenordnungen stets korrekt.

Seien $N \in \mathbb{N}$ und $l_i \in \mathbb{N},\ 1 \leq i \leq N$, paarweise teilerfremd, so daß

$$\begin{aligned} c &\equiv c_i \bmod l_i \text{ mit } c_i \in \{c_{i,1}, \ldots, c_{i,n_i}\},\ 1 \leq i \leq N \\ M &:= \prod_{i=1}^{N} l_i > 4\sqrt{p} \\ K &:= \prod_{i=1}^{N} n_i \end{aligned}$$

d.h. wir kennen n_i mögliche Werte für c modulo l_i, $\ 1 \leq i \leq N$.

Wir setzen die Kongruenzen (Relationen) modulo l_i, $1 \leq i \leq N$, mit dem Chinesischen Restsatz (vgl. [Co93]) zu Resten modulo M zusammen und erhalten eine Liste $c_1, \ldots, c_K$, in der c modulo M enthalten ist. Nach dem Satz von Hasse (Satz 1.20) ist $\#E(\mathbb{F}_p) = p+1-c$ mit $|c| \leq 2\sqrt{p}$, also $|c| < \frac{M}{2}$. Damit können wir c als absolut kleinsten Rest modulo M berechnen; repräsentieren wir die Restklassen in der Liste $c_1, \ldots, c_K$ durch ihre absolut kleinsten Reste, so wissen wir, daß die gesuchte Spur c in dieser Liste enthalten ist.

Um den richtigen Wert c aus dieser Liste herauszufinden, führen wir einen probabilistischen Test durch, indem wir einen Punkt $Q \in E(\mathbb{F}_p)$ zufällig wählen und überprüfen, ob

$$(p + 1 - c_i) \cdot Q = \mathcal{O}, \quad 1 \leq i \leq K.$$

Ist dieser Test für ein c_{i_0}, $1 \leq i_0 \leq K$, erfolgreich, so erhalten wir als wahrscheinliche Gruppenordnung $\#E(\mathbb{F}_p) = p + 1 - c_{i_0}$. Wir verwenden dann den in [Mü94, Beweis der Gruppenordnung] beschriebenen Algorithmus, um die Korrektheit der ermittelten Ordnung zu überprüfen.

Wir wollen uns kurz damit beschäftigen, warum es sinnvoll ist, den beschriebenen probabilistischen Test zu verwenden. Es existiert eine Konstante $c > 0$, so daß für hinreichend großes p ein Punkt der Ordnung größer gleich $\sqrt{p}$ mindestens mit der Wahrscheinlichkeit $\frac{c}{loglog(p)}$ gewählt wird (vgl. [Mü91, Kap. 3]). Gehen wir also davon aus, daß wir einen Punkt mit genügend großer Ordnung wählen, so ist die Anzahl der Vielfachen der Punktordnung im Hasse-Intervall, die ungleich der Gruppenordnung sind, kleiner gleich 4. Also ist die Wahrscheinlichkeit, ein solches Vielfaches zu treffen $\frac{1}{\sqrt{p}}$ und damit die Wahrscheinlichkeit, daß bei positivem Test nicht die Gruppenordnung gefunden wird $\frac{K}{\sqrt{p}}$. Insbesondere durch die Methoden, welche in Kapitel 4 beschrieben werden, ist der Wert K um einige Dezimalstellen kleiner als $\sqrt{p}$.

Das vorgeschlagene Verfahren hat den Nachteil, daß $\mathcal{O}(K)$ viele Berechnungen von Punktvielfachen nötig sind. Deshalb modifizieren wir diese Vorgehensweise, indem wir eine **Babystep-Giantstep** Strategie verwenden. Das Prinzip dieses Verfahrens wurde zum ersten Mal 1970 von Daniel Shanks vorgestellt ([Sh70]). Wir transformieren den Test $(p + 1 - c_i) \cdot Q = \mathcal{O}$, $1 \leq i \leq K$, zu einem Vergleich $b_i \cdot Q = g_j \cdot Q$, $1 \leq i, j \leq \sqrt{K}$. Wir berechnen alle Punkte $b_i \cdot Q$, $1 \leq i \leq \sqrt{K}$, (Babysteps) und speichern diese in einer Tabelle B. Dann bestimmen wir solange Punkte $g_i \cdot Q$, $1 \leq i \leq \sqrt{K}$, (Giantsteps) bis wir für einen Index i_0 den Punkt $g_{i_0} \cdot Q$ in der Tabelle B gefunden haben. Ist der in der Tabelle gefundene Punkt von der Form $b_{j_0} \cdot Q$, so erhalten wir als wahrscheinliche Gruppenordnung $b_{j_0} - g_{i_0}$. Wir werden im folgenden Atkin's Idee vorstellen, um eine Aufteilung der Relationen in Babysteps und Giantsteps zu erhalten.

Wieder kombinieren wir die Kongruenzen mit Hilfe des Chinesischen Restsatzes, aber anstatt mit den kleinen Moduln l_i, $1 \leq i \leq N$, ein großes Produkt M zu bilden, berechnen wir drei Produkte M_1, M_2, M_3. Dabei fassen wir in M_3 alle Moduln l_i mit $n_i = 1$ zusammen und teilen die übrigen Moduln in zwei Hälften, mit denen wir M_1 und M_2 bilden. Die dabei entstehenden Listen von Kongruenzen für M_1 und M_2 bezeichnen wir mit L_1 sowie L_2, den absolut kleinsten Rest von c modulo M_3 mit c_3.

Wir erhalten

$$\begin{aligned} c &\equiv c_3 \text{ modulo } M_3 \\ c &\equiv \text{Listenelement aus } L_1 \text{ modulo } M_1 \\ c &\equiv \text{Listenelement aus } L_2 \text{ modulo } M_2 \end{aligned}$$

$$\text{mit } M_1 \cdot M_2 \cdot M_3 = \prod_{i=1}^{N} l_i.$$

Um einen Babystep-Giantstep Algorithmus anwenden zu können, müssen wir die Berechnung von Punktvielfachen in 2 Teile zerlegen; wir können die gesuchte Spur c auf folgende Art darstellen.

3.1. Satz *Es existieren* $R_1, R_2 \in \mathbb{Z}$ *mit* $\mid R_1 \mid < \frac{M_1}{2}$, $\mid R_2 \mid < M_2$, *so daß*

$$c = c_3 + M_3 \cdot (R_1 \cdot M_2 + R_2 \cdot M_1). \tag{3.1}$$

Beweis: Zuerst zerlegen wir c modulo M_3:

$$c = c_3 + M_3 \cdot q, \quad q \in \mathbb{Z}, \quad |c_3| \text{ minimal.}$$

Den Betrag von q können wir abschätzen:

$$\begin{aligned} |q| \cdot |M_3| &= |c - c_3| \\ &\leq |c| + |c_3| \\ &< \frac{M_1 \cdot M_2 \cdot M_3}{2} + \frac{M_3}{2} \end{aligned}$$

und somit

$$|q| < \frac{M_1 \cdot M_2}{2} + \frac{1}{2} = \left\lceil \frac{M_1 \cdot M_2}{2} \right\rceil.$$

Da q ganzzahlig und $M_1 \cdot M_2 \equiv 1 \bmod 2$ ist, gilt die Ungleichung

$$|q| < \frac{M_1 \cdot M_2}{2}. \quad (*)$$

Weil M_1 und M_2 teilerfremd sind, ist das von ihnen erzeugte Ideal ganz $\mathbb{Z}$. Wir finden deshalb Zahlen $\tilde{R}_1, \tilde{R}_2 \in \mathbb{Z}$, so daß $q = \tilde{R}_1 \cdot M_2 + \tilde{R}_2 \cdot M_1$. Wählen wir $k \in \mathbb{Z}$ mit

$$\mid \underbrace{\tilde{R}_1 - k \cdot M_1}_{=:R_1} \mid < \frac{M_1}{2}, \qquad (**)$$

so erhalten wir:

$$\begin{aligned} q &= R_1 \cdot M_2 + \tilde{R}_2 \cdot M_1 + k \cdot M_1 \cdot M_2 \\ &= R_1 \cdot M_2 + \underbrace{(\tilde{R}_2 + k \cdot M_2)}_{=:R_2} \cdot M_1 \\ &= R_1 \cdot M_2 + R_2 \cdot M_1 \text{ mit } |R_1| < \frac{M_1}{2}. \end{aligned}$$

Formen wir dies um und benutzen wir die hergeleiteten Ungleichungen, so gilt

$$\begin{aligned} |R_2 \cdot M_1| &= |q - R_1 \cdot M_2| \\ &\leq |q| + |R_1 \cdot M_2| \\ &\underset{(**)}{\overset{(*)}{<}} M_1 \cdot M_2 \end{aligned}$$

und somit $|R_2| < M_2$. ■

Die Werte R_1 bzw. R_2, die wir zur Darstellung von c benötigen, können wir aus Formel (3.1) berechnen, indem wir diese Gleichung modulo M_1 bzw. M_2 betrachten. Lösen wir die

Gleichung nach R_1 bzw. R_2 auf, so stellen wir fest, daß wir zu ihrer Berechnung c modulo M_1 bzw. M_2 brauchen. Diese Reste sind aber in den Listen L_1 und L_2 enthalten:

$$\begin{aligned} c - c_3 &= M_3 R_1 M_2 + M_3 R_2 M_1 \quad \text{mit} \\ R_1 &\equiv \frac{c - c_3}{M_2 M_3} \bmod M_1 \\ R_2 &\equiv \frac{c - c_3}{M_1 M_3} \bmod M_2. \end{aligned}$$

Da wir wissen, daß $|R_1| < \frac{M_1}{2}$, ist R_1 nicht nur kongruent zur rechten Seite, sondern gleich der rechten Seite, falls wir den absolut kleinsten Rest wählen, d.h.

$$R_1 = \left(\frac{c - c_3}{M_2 M_3} \bmod M_1, \text{ abs. kleinster Rest} \right).$$

Mit der gleichen Argumentation erhalten wir wegen $|R_2| < M_2$

$$\begin{aligned} R_2 &= \left(\frac{c - c_3}{M_1 M_3} \bmod M_2, \text{ pos. kleinster Rest} \right) \\ &\text{oder} \\ R_2 &= \left(\frac{c - c_3}{M_1 M_3} \bmod M_2, \text{ neg. kleinster Rest} \right). \end{aligned}$$

Um aus den Listenelementen die richtigen Werte für R_1 und R_2 herauszufinden, bestimmen wir wieder zufällig einen Punkt $Q \in E(\mathbb{F}_p)$ und testen auf Nullpunkt, d.h. wir überprüfen, ob

$$(p + 1 - c) \cdot Q = \mathcal{O}.$$

Formen wir dies um, so erhalten wir

$$\begin{aligned} (p + 1) \cdot Q &= c \cdot Q \\ &= (c_3 + R_1 M_2 M_3 + R_2 M_1 M_3) \cdot Q \end{aligned}$$

und so eine gesuchte Aufteilung für das Babystep-Giantstep Verfahren:

$$(p + 1 - c_3 - R_1 M_2 M_3) \cdot Q = (R_2 M_1 M_3) \cdot Q. \tag{3.2}$$

Wir berechnen für alle möglichen Listenelemente aus L_2 die Werte für R_2 und speichern jeden Punkt $(R_2 M_1 M_3) \cdot Q$ und den zugehörigen Wert R_2 in einer Tabelle ab. Für alle möglichen Werte R_1 berechnen wir den Punkt $(p+1-c_3-R_1 M_2 M_3) \cdot Q$ und suchen diesen in der Tabelle. Finden wir den Punkt, so entnehmen wir den entsprechenden R_2-Wert der Tabelle und berechnen die wahrscheinliche Gruppenordnung

$$\# E(\mathbb{F}_p) = p + 1 - c_3 - R_1 M_2 M_3 - R_2 M_1 M_3. \tag{3.3}$$

Wir fassen dieses Verfahren in dem folgenden Algorithmus zusammen.

Babystep-Giantstep-Algorithmus

EINGABE: Moduln $l_i \in \mathbb{N}$, $1 \leq i \leq N$, paarweise teilerfremd, wobei $c \equiv c_i \bmod l_i$ mit $c_i \in \{c_{i,1}, \ldots, c_{i,n_i}\}$, $1 \leq i \leq N$ und $\prod_{i=1}^{N} l_i > 4\sqrt{p}$.

AUSGABE: wahrscheinliche Gruppenordnung $\#E(\mathbb{F}_p) = p + 1 - c$.

BILDUNG DER MODULN

(1) Kombiniere die Kongruenzen von $c \bmod l_i$, $1 \leq i \leq N$, mit dem Chinesischen Restsatz zu
$c \equiv c_3$ modulo M_3
$c \equiv$ Listenelement aus L_1 modulo M_1
$c \equiv$ Listenelement aus L_2 modulo M_2.

(2) Wähle einen zufälligen Punkt $Q \in E(\mathbb{F}_p)$.

BABYSTEPS

(3) Definiere eine Feld B von Punkten der Größe $2 \cdot \#L_2$.

(4) Definiere eine Feld I von Indizes der Größe $2 \cdot \#L_2$.

(5) **for** $(i = 0; i < \#L_2; i++)$ **do**

(6) $z := i$-tes Element aus L_2.

(7) $R_2 = \left(\frac{z-c_3}{M_1 M_3} \bmod M_2, \text{ pos. kleinster Rest}\right)$.

(8) $B[2i] = (R_2 M_1 M_3) \cdot Q$.

(9) $I[2i] = R_2$.

(10) $R_2 = \left(\frac{z-c_3}{M_1 M_3} \bmod M_2, \text{ neg. kleinster Rest}\right)$.

(11) $B[2i+1] = (R_2 M_1 M_3) \cdot Q$.

(12) $I[2i+1] = R_2$.

(13) **od**

(14) Sortiere die Tabelle B anhand der X-Koordinaten und permutiere I entsprechend.

Babystep-Giantstep-Algorithmus
(Fortsetzung)

GIANTSTEPS

(15) **for** $(i = 0; i < \#L_1; i++)$ **do**
(16) $z :=$ i-tes Element aus L_1.
(17) $R_1 = \left(\frac{z-c_3}{M_2 M_3} \bmod M_1, \text{ abs. kleinster Rest}\right)$.
(18) $G := (p + 1 - c_3 - R_1 M_2 M_3) \cdot Q$.
(19) **if** $(G == B[j], 0 \leq j < 2 \cdot \#L_2)$ **then**
(20) $R_2 := I[j]$.
(21) **return** $(\#E(\mathbb{F}_p) = p + 1 - c_3 - R_1 M_2 M_3 - R_2 M_1 M_3.)$
(22) **fi**
(23) **od**

3.2. Bemerkung Die Werte in der Babystep-Tabelle werden mit Quicksort aufsteigend sortiert und im Giantstep-Teil verwenden wir Binäre Suche, um Punkte in der Tabelle zu finden (vgl. [Me84]).

Wählen wir die Moduln M_1, M_2 so, daß $\#L_1 \approx \#L_2 \approx \sqrt{K}$, wobei $K = \prod_{i=1}^{N} n_i$ ist, dann müssen wir nur $\mathcal{O}(\sqrt{K})$ viele Punktvielfache berechnen, im Gegensatz zu $\mathcal{O}(K)$ vielen im ersten Ansatz. Dennoch ist der angegebene Algorithmus ist dieser Form noch nicht effizient genug. Damit das Verfahren praktikabel wird, müssen wir einige „Tricks“ verwenden, die im nächsten Abschnitt beschrieben werden.

3.2 Praktische Aspekte

Wir werden in diesem Abschnitt Methoden angeben, mit denen wir die Babystep-Giantstep-Phase beschleunigen können. Der einzigste Kostenfaktor ist die Berechnung von Punktvielfachen, d.h. berechne zu $Q \in E(\mathbb{F}_p)$ und $n \in \mathbb{N}$ den Punkt $n \cdot Q$. Wir zeigen zuerst, wie wir ein solches Punktvielfaches für einen einzelnen Wert n berechnen können. Dann geben wir Verfahren an, die ausnutzen, daß wir im Algorithmus viele Punktvielfache des gleichen Punktes berechnen müssen. Dabei werden wir sehen, daß es wichtig ist, die gegebenen Kongruenzen c modulo l_i, $1 \leq i \leq N$, in bestimmter Weise auf die Moduln M_1 und M_2 zu verteilen.

Seien $Q \in E(\mathbb{F}_p)$ und $n \in \mathbb{N}$ gegeben. Um das Punktvielfache $n \cdot Q$ zu berechnen, können wir das schon in Kapitel 2 vorgestellte Prinzip der „Schnellen Exponentiation“ verwenden.

„Schnelle Exponentiation" zur Berechnung von Punktvielfachen

EINGABE: $Q \in E(\mathbb{F}_p)$, $n \in \mathbb{N}$

AUSGABE: $n \cdot Q$

```
ERG = O.
H = Q.
while (n > 1) do
  if (n&1) then
    ERG = ERG + H.
  fi
  H = 2 · H.
  n = n >> 1.
od
if (n == 1) then
  ERG = ERG + H.
fi
return (ERG)
```

Nach einer Idee von Francois Morain läßt sich dieses Verfahren noch beschleunigen, indem man Ketten von Einsen in der Binärdarstellung von n besonders behandelt (vgl. [MoOl90] bzw. [Mü91, Abschnitt 4.2.1, S. 52]).

Würden wir jedes einzelne Punktvielfache, welches wir im Algorithmus zu berechnen haben, mit Schneller Exponentiation ermitteln, so wäre dies immer noch zu teuer. Wir können aber ausnutzen, daß wir die verschiedenen Vielfachen für einen festen Punkt berechnen müssen.

3.2.1 Babysteps mit Tabelle

Um $B := 2 \cdot \#L_2$ viele Babysteps zu berechnen, müssen wir für alle $R_{2,i}$, $1 \leq i \leq B$, die Punktvielfachen $(R_{2,i} M_1 M_3) \cdot Q$ für $Q \in E(\mathbb{F}_p)$ bestimmen. Atkin schlägt vor in einer Tabelle gewisse Punktvielfache vorzuberechnen.

Sei s so gewählt, daß für „viele" aufeinanderfolgende Werte $R_{2,i}$ und $R_{2,i+1}$, $1 \leq i < B$, die Differenz

$$|R_{2,i} - R_{2,i+1}| < 2 \cdot s^2$$

ist; um dies zu unterstützen, sortieren wir die R_2-Werte aufsteigend. Die Schranke s ist ein heuristischer Wert, der in unserer Implementierung auf 200 gesetzt ist. Wir definieren $P := (M_1 M_3) \cdot Q$ und berechnen die Punkte

$$1 \cdot P,\ 2 \cdot P,\ \ldots,\ (s-1) \cdot P,\ s \cdot P,\ 2s \cdot P,\ 2 \cdot 2s \cdot P,\ 3 \cdot 2s \cdot P,\ \ldots,\ s \cdot 2s \cdot P$$

in einer Tabelle der Größe $2s$ vor. Wollen wir nun den Punkt $x \cdot P,\ 0 < x < 2s^2$, berechnen, so können wir dies mit nur einer Punktaddition tun. Dazu stellen wir x „2s"-adisch dar, d.h. wir zerlegen

$$x = x_1 \cdot 2s + x_0 \quad \text{mit } 0 \leq x_0 \leq 2s \text{ und } 0 \leq x_1 < s.$$

Ist $x_0 > s$, so modifizieren wir die Zerlegung:

$$x = \underbrace{(x_1 + 1)}_{=:x_1} \cdot 2s + \underbrace{(x_0 - 2s)}_{=:x_0}.$$

Dann gilt $0 \leq x_1 \leq s,\ -s < x_0 \leq s$ und wir erhalten:

$$x \cdot P = x_1 \cdot 2s \cdot P + x_0 \cdot P.$$

Da wir für jeden Punkt $T = (X, Y)$, der in Tabelle enthalten ist, mit einer additiven Inversion auch den Punkt $-T = (X, -Y)$ berechnen können, benötigen wir zur Bestimmung von $x \cdot P$ nur eine Punktaddition.

Um größere Vielfache x zu berechnen, können wir die Schranke s erhöhen oder die Tabelle um die Punkte

$$2 \cdot 2s^2 \cdot P,\ 3 \cdot 2s^2 \cdot P,\ \ldots,\ 2s^3 \cdot P$$

erweitern; wir können damit Punktvielfache $x \cdot P$ mit $0 < x < 2s^3$ berechnen, erhalten jedoch jeweils 2 Punktadditionen.

<u>Beispiel:</u>
Wir setzen die Schranke s auf 200, geben uns einen Punkt $P \in E(\mathbb{F}_p)$ vor und bauen wie beschrieben die Tabelle auf. Um $700 \cdot P$ zu berechnen, zerlegen wir 700 „2s"-adisch, d.h.

$$700 = 1 \cdot 2s + 300 = 2 \cdot 2s - 100.$$

Wir nehmen die Punkte $2 \cdot 2s \cdot P = 800 \cdot P$ und $100 \cdot P$ aus der Tabelle, invertieren $100 \cdot P$ und erhalten:

$$700 \cdot P = 800 \cdot P + (-100) \cdot P.$$

Damit die Abstände zwischen zwei zu berechnenden Vielfachen möglichst klein werden, berechnen wir alle R_2-Werte, speichern sie in einem Feld $R2$ und sortieren dieses. In dem Indexfeld I, in welchem wir uns zuvor die R_2-Werte gemerkt haben, speichern wir nur noch den Index des entsprechenden R_2-Wertes im Feld $R2$. Durch die Benutzung der Tabelle wird die Berechnung der Babysteps wie folgt modifiziert:

Babysteps mit Tabelle

EINGABE: Liste L_2, Moduln M_1, M_2, M_3, $c_3 \equiv c \bmod M_3$, $Q \in E(\mathbb{F}_p)$.
AUSGABE: Sortierte Babystep-Tabelle.

(1) Definiere eine Feld B von Punkten der Größe $2 \cdot \#L_2$.
(2) Definiere eine Feld I von Indizes der Größe $2 \cdot \#L_2$.
(3) Definiere eine Feld $R2$ von Vielfachen der Größe $2 \cdot \#L_2$.
(4) **for** $(i = 0; i < \#L_2; i++)$ **do**
(5) $\quad z :=$ i-tes Element aus L_2.
(6) $\quad R2[2i] = \left(\frac{z-c_3}{M_1 M_3} \bmod M_2, \text{ pos. kleinster Rest}\right)$.
(7) $\quad R2[2i+1] = \left(\frac{z-c_3}{M_1 M_3} \bmod M_2, \text{ neg. kleinster Rest}\right)$.
(8) **od**
(9) Sortiere das Feld $R2$ aufsteigend.
(10) $P := (M_1 M_3) \cdot Q$.
(11) $B[0] := R2[0] \cdot P$.
(12) $I[0] := 0$.
(13) **for** $(i = 1; i < 2 \cdot \#L_2; i{+}{+})$ **do**
(14) $\quad x = R2[i] - R2[i-1]$.
(15) $\quad$ **if** $(x < 2s^2)$ **then**
(16) $\quad\quad H := x \cdot P$ mit Tabelle.
(17) $\quad$ **else**
(18) $\quad\quad H := x \cdot P$ mit Schneller Exponentiation.
(19) $\quad$ **fi**
(20) $\quad B[i] := B[i-1] + H$.
(21) $\quad I[i] := i$.
(22) **od**
(23) Sortiere die Tabelle B anhand der X-Koordinaten und permutiere I entsprechend.

Bei diesem Verfahren wechseln wir in jedem Schritt den Bezugspunkt, im i-ten Schritt rechnen wir relativ zu $R2[i-1] \cdot P$. Halten wir den Bezugspunkt fest und berechnen alle nachfolgenden Differenzen relativ zu diesem Punkt, so können wir den ggT-Trick anwenden.

3.2.2 Babysteps mit Tabelle und ggT-Trick

Betrachten wir die Additionsformeln für Punkte (vgl. Satz 1.3), so erkennen wir, daß für jede Punktaddition ein Quotient zu berechnen ist, d.h. wir müssen ein multiplikatives Inverses, also einen erweiterten ggT, bestimmen. Addieren wir verschiedene Punkte zu einem festen Punkt hinzu, so können wir alle Nenner der jeweils zu berechnenden Quotienten zu einem Produkt zusammenfassen, dieses durch ggT-Berechnung invertieren und durch geeignete Multiplikationen die Einzelinversen extrahieren. Da Multiplikationen billiger sind als Berechnung von ggT's, erwarten wir dadurch einen Laufzeitgewinn (vgl.[Be93, S.35]). Wir schildern hier die Idee dieses Verfahrens.

Seien $a_1, \ldots, a_B \in \mathbb{F}_p^*$ gegeben und deren Inverse gesucht. Wir berechnen

$$\begin{aligned} A_0 &:= 1 \\ A_i &:= \prod_{j=1}^{i} a_i, \;\; 1 \leq i \leq B \end{aligned}$$

und erhalten

$$\begin{aligned} \frac{1}{A_B} &: \;\; \text{durch eine ggT - Berechnung}, \\ \frac{1}{a_B} &= \frac{1}{A_B} \cdot A_{B-1}, \\ \frac{1}{A_i} &= \frac{1}{A_{i+1}} \cdot a_{i+1}, \\ \frac{1}{a_i} &= \frac{1}{A_i} \cdot A_{i-1}, \;\; i = B-1, \ldots, 1. \end{aligned}$$

Seien die Babystep-Punkte $R_{2,i} \cdot P$, $0 \leq i < B$, zu berechnen. Den Bezugspunkt $R_{2,0} \cdot P$ ermitteln wir mit Schneller Exponentiation. Wir berechnen dann die Differenzpunkte

$$(R_{2,i} - R_{2,0}) \cdot P =: (x_i, y_i), \;\; 1 \leq i < B,$$

entweder mit Hilfe der Tabelle oder mit Schneller Exponentiation. Nun müssen wir diese Punkte zum Bezugspunkt addieren, d.h. wir wollen berechnen:

$$R_{2,i} \cdot P = R_{2,0} \cdot P + (R_{2,i} - R_{2,0}) \cdot P.$$

Definieren wir $(x_0, y_0) := R_{2,0} \cdot P$, so treten bei diesen Additionen die Nenner $(x_i - x_0)$, $1 \leq i < B$, auf (vgl. Satz 1.3). Diese invertieren wir simultan nach dem oben angegebenen Verfahren und erhalten $1/(x_i - x_0)$, $1 \leq i < B$. Unter Benutzung dieser simultan berechneten Inversen addieren wir die Differenzpunkte zum Bezugspunkt hinzu und erhalten so die gesuchten Babystep-Punkte.

Beispiel: $E = (30, 11)$ über $\mathbb{F}_{43}$, $s = 2$.
Sei $P = (19, 1)$ und sei $R_2 \cdot P$ für die R_2-Werte $9, 15, 17$ und 30 zu berechnen. Wir bauen zuerst die Hilfstabelle der Größe $2s$ auf, d.h. wir berechnen

$$P = (19, 1), \;\; 2P = (22, 28), \;\; 4P = (9, 35), \;\; 8P = (20, 21).$$

Dann berechnen wir $R_0 \cdot P$ mit Schneller Exponentiation, d.h.

$$R_0 \cdot P = 9 \cdot (19, 1) = (17, 39).$$

Die sich ergebenden Differenzpunkte sind

$$\begin{array}{ll} 6 \cdot P = (10, 35) & \text{mit Tabelle } (4P + 2P) \\ 8 \cdot P = (20, 21) & \text{mit Tabelle } (8P) \\ 21 \cdot P = (36, 24) & \text{mit Schneller Exponentiation.} \end{array}$$

Wollen wir diese Differenzpunkte zu $R_0 \cdot P$ addieren, so ergeben sich bei den Punktadditionen die folgenden Nenner (Differenzen der x-Koordinaten):

$$17 - 10 \equiv 7, \;\; 17 - 20 \equiv 40, \;\; 17 - 36 \equiv 24.$$

Wir invertieren diese Werte simultan. Dazu berechnen wir

$$7 \cdot 40 \equiv 22, \;\; 7 \cdot 40 \cdot 24 \equiv 12.$$

Wir invertieren 12 durch ggT-Berechnung und erhalten

$$\frac{1}{12} \equiv 18.$$

Wir multiplizieren 18 mit $7 \cdot 40$ und bekommen

$$\frac{1}{24} \equiv 22 \cdot 18 \equiv 9.$$

Multiplizieren wir $1/12$ mit 24, so erhalten wir das Inverse von $1/(7 \cdot 40)$:

$$\frac{1}{7 \cdot 40} \equiv 18 \cdot 24 \equiv 2.$$

Durch Multiplikation mit 7 bzw. 40 erhalten wir die beiden anderen Inversen:

$$\begin{array}{lll} \frac{1}{40} & \equiv 2 \cdot 7 & \equiv 14 \\ \frac{1}{7} & \equiv 2 \cdot 40 & \equiv 37. \end{array}$$

Nun addieren wir die Differenzpunkte zum Bezugspunkt $R_0 \cdot P$ und benutzen dabei die vorberechneten Nenner:

$$\begin{array}{lll} 15 \cdot P = & 9 \cdot P + 6 \cdot P = & (33, 1) \\ 17 \cdot P = & 9 \cdot P + 8 \cdot P = & (42, 25) \\ 30 \cdot P = & 9 \cdot P + 21 \cdot P = & (26, 36). \end{array}$$

Damit die Abstände zum Bezugspunkt nicht so groß werden, daß wir die Tabelle nicht mehr benutzen können, um die Differenzpunkte zu berechnen ($30 \cdot P$ im Beispiel), fassen wir die R_2-Werte in Paketen zusammen. Der Bezugspunkt ist jeweils der erste Wert eines Paketes und alle R_2-Werte in einem Paket haben zu dem ersten Wert einen Abstand kleiner $2s^2$ (vgl. Abschnitt 3.2.1). Dadurch können wir alle Differenzpunkte innerhalb eines Paketes unter Benuztung der Tabelle berechnen. Wir erhalten den folgenden Algorithmus.

Babysteps mit Tabelle und ggT-Trick

EINGABE: Liste L_2, Moduln M_1, M_2, M_3, $c_3 \equiv c \bmod M_3$, $Q \in E(\mathbb{F}_p)$, aufsteigend sortierte Liste $R2$ der R_2-Werte, Hilfstabelle der Größe $2s$.
AUSGABE: Babystep-Tabelle.

(1) Definiere eine Feld B von Punkten der Größe $2 \cdot \#L_2$.
(2) Definiere eine Feld I von Indizes der Größe $2 \cdot \#L_2$.
(3) $P := (M_1 M_3) \cdot Q$. /*Initialisierung*/
(4) $i = 1$, $j = 0$.
(5) $Q_1 := R2[0] \cdot P$ mit Schneller Exponentiation. /*1. Bezugspunkt*/
(6) $B[0] = Q_1$, $I[0] = 0$.
(7) **while** $(i < 2 \cdot \#L_2)$ **do** /*durchlaufe R_2-Werte*/
(8)   **while** $((x = R2[i] - R2[j]) < 2s^2)$ **do** /*nächstes Paket*/
(9) $P_i = x \cdot P$ mit Tabellenbenutzung.
(10) $i = i + 1$.
(11)   **od**
(12)   **for** $(k = j + 1; k < i; k + +;)$ **do**
(13)   Berechne den Nenner N_k, der bei der Punktaddition $Q_1 + P_k$ auftritt.
(14)   **od**
(15)   Berechne $Inv_k = 1/N_k$, $k = j + 1, \ldots, i - 1$ simultan. /*ggT-Trick*/
(16)   **for** $(k = j + 1; k < i; k + +)$ **do**
(17) $B[k] = Q_1 + P_k$ unter Benutzung von Inv_k. /*$= R2[k] \cdot P$*/
(18) $I[k] = k$.
(19)   **od**
(20) $j = i - 1$.
(21) $Q_1 = R2[i - 1] \cdot P$. /*neuer Bezugspunkt*/
(22) **od**
(23) Sortiere die Tabelle B anhand der X-Koordinaten und permutiere I entsprechend.

In der Praxis müssen wir hierbei noch einige Spezialfälle betrachten. Außerdem speichern wir die Differenzpunkte P_k, die Nenner N_k sowie die Inversen Inv_k in Feldern fester Größe.

3.3. Bemerkung Liegen die Werte $R_{2,i}(\ i = 1, \ldots, B)$ so dicht zusammen, daß sukzessive Werte eine Differenz kleiner s haben, bei festem Bezugspunkt aber Differenzen größer s auftreten, so verschlechtert der ggT-Trick die Laufzeit, da ohne ggT-Trick pro $R_{2,i}$ eine Punktaddition nötig ist (der Differenzpunkt steht in der Tabelle), mit ggT-Trick eine Punktaddition für den Differenzpunkt und eine ggT-Trick unterstützte Punktaddition notwendig sind.

3.2.3 Abspeichern der Babystep-Tabelle

Bisher haben wir uns mit Laufzeitoptimierung befaßt. Wollen wir größere Charakteristiken verarbeiten, so müssen wir uns auch Gedanken über den benötigten Speicherplatz machen.

Angenommen die Charakteristik p hat 200 Dezimalstellen, d.h. eine Zahl aus $\mathbb{F}_p$ benötigt in etwa 75 Bytes. Haben wir 300 000 mögliche Kongruenzen für die Spur c modulo M_2, so erhalten wir 600 000 Werte R_2 und genauso viele Punkte $(R_2 M_1 M_3) \cdot Q$, $Q \in E(\mathbb{F}_p)$. Würden wir für jeden Punkt allein die X-Koordinate abspeichern, so benötigten wir $600\,000 \cdot 75$ Bytes, also 45 MB, Speicherplatz für die Babystep-Tabelle. Setzen wir den Modul M_2 mit 16 Bytes an, dann hat das Feld $R2$, welches die R_2-Werte aufnimmt, eine Größe von $9,6$ MB. Wir werden im folgenden Modifikationen angeben, um den Speicherplatzbedarf zu reduzieren.

Wir speichern von einem Punkt $(R_2 M_1 M_3) \cdot Q$ nur das erste Wort der X-Koordinate, in unserer Implementierung 4 Bytes, ab. Dies wirkt sich auf die Suche im Giantstep-Teil aus. Finden wir einen gesuchten Wert in der Babystep-Tabelle, so wissen wir nur, daß Giantstep-Punkt und Babystep-Punkt auf dem ersten Wort der X-Koordinate übereinstimmen; deshalb müssen wir im Falle einer erfolgreichen Suche den entsprechenden Babystep-Punkt neu berechnen und die vollständigen Punktkoordinaten vergleichen. Anhand des Indexfeldes I wissen wir, welches R_2-Element im Feld $R2$ zu einem Babystep-Punkt gehört (ist die X-Koordinate des Babystep-Punktes in der Tabelle B an Stelle k gespeichert, so trägt der zugehörige R_2-Wert den Index $I[k]$ im Feld $R2$).

Bisher speichern wir im Feld $R2$ alle R_2-Werte ab. Wir können den hierfür notwendigen Platzbedarf halbieren, indem wir nur noch die positiven kleinsten Reste berechnen und sortieren. Haben wir für den Punkt $P := (M_1 M_3) \cdot Q$ das Vielfache $R_{2,+} \cdot P$ berechnet, wobei $R_{2,+}$ den positven kleinsten Rest von $\left(\frac{c-c_3}{M_1 M_3} \bmod M_2\right)$ bezeichnet, so erhalten wir den negativen kleinsten Rest als $R_{2,-} = R_{2,+} - M_2$, und wir können das entsprechende Punktvielfache mit einer Subtraktion berechnen:

$$R_{2,-} \cdot P = R_{2,+} \cdot P - M_2 \cdot P.$$

Da nur die positiven Reste im Feld $R2$ gespeichert werden, kodieren wir für einen Babystep-Punkt H, dessen X-Koordinatenanteil in $B[k]$ gespeichert ist, im Indexfeld I die folgende Information:

$$I[k] = \begin{cases} j, & \text{falls } H = R2[j] \cdot P \\ -j, & \text{falls } H = (R2[j] - M_2) \cdot P. \end{cases}$$

Wollen wir in der Giantstep-Phase den Babystep-Punkt restaurieren, dessen X-Koordinatenanteil in $B[k]$ gespeichert ist, so berechnen wir

$$\begin{aligned} R2[I[k]] \cdot P, &\quad \text{falls } I[k] \text{ positiv}, \\ R2[-I[k]] \cdot P - M_2 \cdot P, &\quad \text{falls } I[k] \text{ negativ}. \end{aligned}$$

Zur Berechnung dieser Punktvielfachen verwenden wir Schnelle Exponentiation. Da es selten vorkommt, daß zwar das erste Wort der X-Koordinaten übereinstimmt, nicht aber die vollständigen Punkte, müssen nicht „viele" solcher Punktvielfache bestimmt werden.

Haben wir K_B viele Kongruenzen modulo M_2, so erhalten wir insgesamt den folgenden Platzbedarf:

$$\begin{aligned} \text{für die Punkttabelle:} &\quad 2 \cdot K_B \text{ Worte (X-Koord.)} \\ \text{für die Indizes:} &\quad 2 \cdot K_B \text{ Worte} \\ \text{für die } R_2 - \text{Datei:} &\quad K_B \;\; R_2 - \text{Werte.} \end{aligned}$$

Dies ergibt mit den im obigem Beispiel angebenen Werten $1,2 \cdot 10^6$ Worte und 300 000 R_2-Werte. Setzen wir ein Rechnerwort mit 4 Bytes und den Modul M_2 mit 16 Bytes an, so benötigen wir sowohl für die Babystep-Tabelle als auch für das Feld $R2$ 4,8 MB.

Um beim Tabellenaufbau nicht die Punkttabelle und die R_2-Werte gleichzeitig im Speicher zu halten, speichern wir das Feld $R2$ in einer Datei ab und lesen die R_2-Werte in einzelnen Päckchen einer vorher berechneten Größe wiederein. Somit muß der Hauptspeicher des benutzten Rechners groß genug sein, um zuerst das Feld $R2$ und anschließend die Punkttabelle aufnehmen zu können.

An dieser Stelle sind noch eine Vielzahl von Varianten denkbar, um den benötigten Speicherplatz zu verkleinern und damit mehr Babysteps-Giantsteps verarbeiten zu können. Allerdings sind die Punktoperationen im Vergleich zu Rechnungen über $\mathbb{F}_p$ teuer, so daß es sinnvoller erscheint mit den in Kapitel 4 beschriebenen Methoden (Stichwort Elkies-Primzahlen) die Anzahl der Kongruenzen und damit auch den benötigten Speicherplatz zu reduzieren.

Beispiel: $E = (30, 11)$ über $\mathbb{F}_{43}$, $M_1 M_3 \cdot Q = P := (19, 1)$, $M_2 = 32$.

Seien die berechneten $R_{2,+}$-Werte $9, 15, 17$ und 30. Zuerst berechnen wir den Punkt $-M_2 \cdot P = -32 \cdot (19, 1) = (42, 15)$ vor. Die zugehörigen $R_{2,-}$-Werte sind $9 - 32 = -23,\ 15 - 32 = -17,\ 17 - 32 = -15,\ 30 - 32 = -2$. Wir rechnen die Punkte $R_{2,+} \cdot P$ aus und erhalten den jeweiligen Punkt $R_{2,-} \cdot P$ als $R_{2,+} \cdot P + (-M_2) \cdot P$. Da wir in diesem Beispiel mit kleinen Werten rechnen, ist das erste Wort der jeweiligen x-Koordinate gleich der x-Koordinate.

Wir erhalten folgende Babystep- und R_2-Tabelle:

$R_{2,+}$	$R_{2,+} \cdot P$	$R_{2,-}$	$R_{2,-} \cdot P$	$Baby-Tab.$	$Index$	$R2-Tab.$
9	$(17,39)$	-23	$(0,22)$	$B[1] = 17$	$I[1] = 1$	$R2[1] = 9$
				$B[2] = 0$	$I[2] = -1$	
15	$(33,1)$	-17	$(42,18)$	$B[3] = 33$	$I[3] = 2$	$R2[2] = 15$
				$B[4] = 42$	$I[4] = -2$	
17	$(42,25)$	-15	$(33,42)$	$B[5] = 42$	$I[5] = 3$	$R2[3] = 17$
				$B[6] = 33$	$I[6] = -3$	
30	$(26,36)$	-2	$(22,15)$	$B[7] = 26$	$I[7] = 4$	$R2[4] = 30$
				$B[8] = 22$	$I[8] = -4$	

Sortieren wir die Babystep-Tabelle, so ergibt sich folgende Konstellation:

$Nr.$	1	2	3	4	5	6	7	8
$B:$	0	17	22	26	33	33	42	42
$I:$	-1	1	-4	4	2	-3	-2	3

3.2.4 Aufteilung der Giantsteps

Wir beziehen uns in den folgenden Erläuterungen auf die Notationen von Abschnitt 3.1.

In der Giantstep-Phase müssen wir für viele Werte R_1 die Punkte $(p+1-c_3-R_1M_2M_3)\cdot Q$, $Q \in E(\mathbb{F}_p)$, ermitteln. Dazu berechnen wir uns natürlich $(p+1-c_3)\cdot Q$ sowie $P := (M_2M_3)\cdot Q$ vor. Somit besteht unsere Hauptaufgabe in der Berechnung von $R_1 \cdot P$. Da die verschiedenen Werte für R_1 in der Regel einen großen Abstand haben (s. auch Abschnitt 3.2.5) können wir nicht direkt analog zur Berechnung der Babysteps vorgehen. Stattdessen werden wir die Giantsteps nochmal unterteilen und uns dann Informationen vorberechnen.

Die Kongruenzen von c modulo M_1 sind aus den Kongruenzen modulo l_i, $i = 1,\ldots,N_G$, entstanden:

$$c \equiv c_i \bmod l_i \text{ mit } c_i \in \{c_{i,1},\ldots,c_{i,n_i}\},\ 1 \le i \le N_G.$$

Wir fassen die Moduln l_i nicht mehr zu einem Produkt M_1 zusammen, sondern verteilen die Moduln auf zwei Produkte $M_{1,1}$ und $M_{1,2}$, so daß

$$\begin{aligned} M_1 &= M_{1,1} \cdot M_{1,2} \\ c &\equiv e_{1,1} \text{ modulo } M_{1,1} \\ c &\equiv e_{1,2} \text{ modulo } M_{1,2} \end{aligned}$$

wobei $e_{1,j} \in \{c_{j,1},\ldots,c_{j,K_{G_j}}\} =: L_{1,j}$, $j = 1,2$. Für diese Aufteilung gilt der folgende

3.4. Satz *Es existieren* $R_{1,1}, R_{1,2} \in \mathbb{Z}$ *mit* $|R_{1,1}| < \frac{M_{1,1}}{2}$, $|R_{1,2}| < M_{1,2}$, *so daß*

$$R_1 = R_{1,1} \cdot M_{1,2} + R_{1,2} \cdot M_{1,1}. \tag{3.4}$$

Beweis: Wegen der Teilerfremdheit von $M_{1,1}$ und $M_{1,2}$ ist die Existenz von $R_{1,1}$ und $R_{1,2}$ klar. Außerdem kann man stets erreichen, daß $|R_{1,1}| < \frac{M_{1,1}}{2}$. Damit erhalten wir für $R_{1,2}$

$$\begin{aligned} |R_{1,2}| \cdot |M_{1,1}| &= |R_1 - R_{1,1} \cdot M_{1,2}| \\ &\leq |R_1| + |R_{1,1}| \cdot |M_{1,2}| \\ &< \frac{M_1}{2} + \frac{M_{1,1}}{2} \cdot M_{1,2} \\ &= M_1. \end{aligned}$$

Also gilt $|R_{1,2}| < M_{1,2}$. ∎

Für die Spur c erhalten wir damit die folgende Darstellung:

$$c = c_3 + M_3 \cdot ((R_{1,1}M_{1,2} + R_{1,2}M_{1,1}) \cdot M_2 + R_2 M_1)$$

und somit

$$c - c_3 = R_{1,1}M_{1,2}M_2M_3 + R_{1,2}M_{1,1}M_2M_3 + R_2M_1M_3.$$

Formen wir diese Identität nach $R_{1,i}$ um und rechnen dabei modulo $M_{1,i}$, $i = 1, 2$, so gilt nach obigem Satz:

$$\begin{aligned} R_{1,1} &= \left(\frac{c - c_3}{M_{1,2}M_2M_3} \bmod M_{1,1}, \text{ abs. kleinster Rest} \right) \\ R_{1,2} &= \left(\frac{c - c_3}{M_{1,1}M_2M_3} \bmod M_{1,2}, \text{ pos. oder neg. Rest} \right). \end{aligned}$$

Wir erhalten also soviele Möglichkeiten für $R_{1,1}$ wie es Kongruenzen von c modulo $M_{1,1}$ gibt und zweimal soviele Möglichkeiten für $R_{1,2}$ wie Kongruenzen von c modulo $M_{1,2}$ existieren.

Die Aufteilung in Babysteps und Giantsteps sieht damit folgendermaßen aus:

$$\begin{aligned} c - c_3 &= R_{1,1}M_{1,2}M_2M_3 + R_{1,2}M_{1,1}M_2M_3 + R_2M_1M_3 \\ \Leftrightarrow R_2M_1M_3 &= p + 1 - c_3 - R_{1,1}M_{1,2}M_2M_3 - R_{1,2}M_{1,1}M_2M_3. \end{aligned}$$

Die rechte Seite der zweiten Gleichung gibt die zu berechnenden Giantsteps an. Wir berechnen für den Punkt $Q \in E(\mathbb{F}_p)$ in einer inneren Schleife alle Punktvielfachen $(p + 1 - c_3) \cdot Q - (R_{1,2}M_{1,1}M_2M_3) \cdot Q$ und speichern sie in einer Tabelle. Dann addieren wir in einer äußeren Schleife jedes Punktvielfache $-(R_{1,1}M_{1,2}M_2M_3) \cdot Q$ zu allen Punkten der inneren Schleife hinzu und suchen den jeweiligen Giantstep-Punkt in der Babystep-Tabelle.

<u>Giantsteps</u>

EINGABE: Babystep-Tabelle B, Indexfeld I, Punkt $Q \in E(\mathbb{F}_p)$, Moduln $M_{1,1}, M_{1,2}, M_2, M_3$, $c_3 \equiv c \bmod M_3$, Listen $L_{1,1}$ bzw. $L_{1,2}$ von Kongruenzen der Spur modulo $M_{1,1}$ bzw. $M_{1,2}$.

AUSGABE: Wahrscheinliche Gruppenordnung $\#E(\mathbb{F}_p) = p + 1 - c$.

<u>INITIALISIERUNG</u>

(1) Definiere ein Feld INN_P von Punkten der Größe $2 \cdot \#L_{1,2}$.

(2) Definiere ein Feld INN_W von Werten der Größe $2 \cdot \#L_{1,2}$.

(3) $H := (p + 1 - c) \cdot Q$.

(4) $P := (M_2 M_3) \cdot Q$.

(5) $P_1 := M_{1,1} \cdot P$.

(6) $P_2 := M_{1,2} \cdot P$.

<u>INNERE SCHLEIFE</u>

(7) **for** $(i = 0; i < \#L_{1,2}; i{+}{+})$ **do**

(8) $\quad z := i$-tes Element in $L_{1,2}$.

(9) $\quad INN\_W[2i] = \left(\frac{z-c_3}{M_{1,1}M_2M_3} \bmod M_{1,2}, \text{ pos. Rest}\right)$.

(10) $\quad INN\_P[2i] = H - INN\_W[2i] \cdot P_1$.

(11) $\quad INN\_W[2i+1] = \left(\frac{z-c_3}{M_{1,1}M_2M_3} \bmod M_{1,2}, \text{ neg. Rest}\right)$.

(12) $\quad INN\_P[2i+1] = H - INN\_W[2i+1] \cdot P_1$.

(13) **od**

Giantsteps
(Fortsetzung)

Äussere Schleife

for $(i = 0; i < \#L_{1,1}; i{+}{+})$ **do**
  $z :=$ i-tes Element in $L_{1,1}$.
  $R_{1,1} := \left(\frac{z-c_3}{M_{1,2}M_2M_3} \bmod M_{1,1},\ \text{abs. kleinster Rest}\right)$.
  $P_3 := R_{1,1} \cdot P_2$.
  for $(j = 0; j < 2 \cdot \#L_{1,2}; j{+}{+})$ **do**
    $G := INN\_P[j] - P_3$.
    $g :=$ erstes Wort der x-Koordinate von G.
    if $(g == B[k],\ 0 \leq k < 2 \cdot \#L_2)$ **then**
      if $(I[k]$ positiv$)$ **then**
        $R_2 = R2[I[k]]$.
      else
        $R_2 = R2[-I[k]] - M_2$.
      fi
      $D := (R_2 M_1 M_3) \cdot Q$.
      if $(G == D)$ **then**
        $\#E(\mathbb{F}_p) := p + 1 - c_3 - M_2M_3 \cdot (R_{1,1}M_{1,2} + R_{1,2}M_{1,1}) - M_1M_3R_2$.
        return $(\#E(\mathbb{F}_p))$
      fi
    fi
  od
od

Der Vorteil dieser Aufteilung ist offensichtlich. Seien für die Werte $R_1^{(1)}, \ldots, R_1^{(K_G)}$ die Giantsteps zu berechnen. Da diese Werte einen großen Abstand haben (vgl. Abschnitt 3.2.5) nehmen wir an, daß ein $\Delta >> 1$ existiert, so daß

$$|R_1^{(i)} - R_1^{(i+1)}| \geq \Delta, \quad i = 1, \ldots, K_G - 1.$$

Würden wir die Giantstep-Punkte sukzessive berechnen, d.h.

$$(p + 1 - c_3 - R_1^{(i)} M_2 M_3) \cdot Q, \quad i = 1, \ldots, K(G),$$

so erhielten wir pro $R_1^{(i)}$ mindestens $log_2(\Delta)+1$, insgesamt also

$$K_G \cdot (log_2(\Delta)+1)$$

viele Punktadditionen. Teilen wir nun die Kongruenzen in etwa gleich große Hälften ($\sqrt{K_G}$) auf, so erhalten wir

$$2 \cdot \sqrt{K_G} \cdot (log_2(\Delta)+1) + \sqrt{K_G} \cdot (\ \ log_2(\Delta)+1+2\sqrt{K_G}\ \ )$$

$$= 2K_G + 3 \cdot \sqrt{K_G} \cdot (log_2(\Delta)+1)$$

viele Punktadditionen.

Weiterhin können wir die Moduln $M_{1,1}$ und $M_{1,2}$ so bilden, daß die Vielfachen der inneren Schleife etwas dichter liegen als die der äußeren Schleife. Deshalb bauen wir zur Berechnung der inneren Schleife eine Tabelle auf (vgl. Abschnitt 3.2.1). Um jeweils einen Punkt der äußeren Schleife zu allen Punkten der inneren Schleife zu addieren, können wir den in Abschnitt 3.2.2 beschriebenen ggT-Trick benutzen.

3.2.5 Aufteilung der Kongruenzen

Bevor wir in die Babystep-Giantstep-Phase eintreten, verfügen wir über Kongruenzen der Spur c modulo $l_i,\ \ i=1,\ldots,N$, d.h.

$$c \equiv c_i \bmod l_i \text{ mit } c_i \in \{c_{i,1},\ldots,c_{i,n_i}\},\ \ 1 \le i \le N.$$

Diese Relationen möchten wir nun in Babysteps, Giantsteps und die Giantsteps wiederum in eine innere und eine äußere Schleife aufteilen; die zugehörigen Moduln sind $M_2, M_1, M_{1,2}$ und $M_{1,1}$. Außerdem sammeln wird die l_i mit nur einer Kongruenz im Modul M_3.

Zuerst sortieren wir mittels Quicksort die Moduln $l_i,\ \ 1 \le i \le N$, in aufsteigender Reihenfolge nach der Anzahl der Kongruenzen. Dann fassen wir die Moduln mit nur einer Möglichkeit in M_3 zusammen und berechnen mit dem Chinesischen Restsatz $c \equiv c_3$ modulo M_3 (absolut kleinster Rest); sei nun ein Index p_1 ermittelt, so daß $l_1,\ldots,l_{p_1}$ das Produkt M_3 bilden. Die verbleibenden Moduln mit mehreren Kongruenzen verteilen wir weiter.

Betrachten wir unsere Vorgehensweise bei der Berechnung der Babysteps (vgl. Abschnitt 3.2.1) so sehen wir, daß aufeinanderfolgende Werte

$$R_2 = \left(\frac{c-c_3}{M_1 M_3} \text{ modulo } M_2,\ \text{pos. kleinster Rest}\right)$$

nahe beieinander liegen sollen. Dies ist gewährleistet, wenn es viele Kongruenzen von c modulo M_2 gibt und der Modul M_2 klein ist; aus diesem Grund geben wir folgende

3.5. Definition *Seien $l,n \in \mathbb{N}$ und $c_1,\ldots,c_n \in \mathbb{N}$ die möglichen Kongruenzen für c modulo l, so bezeichnen wir den Quotienten*

$$\frac{n}{l} =: D(l)$$

als **Dichte von l.**

Wir ordnen nun die Moduln l_i, $p_1 < i \leq N$, aufsteigend nach ihrer Dichte, so daß für $p_1 < i,j \leq N$ gilt: $D(l_i) \leq D(l_j)$. In der Praxis berechnen wir $\frac{1}{D(l)}$ als „integer“ und ordnen in umgekehrter Reihenfolge. Wir bestimmen uns nun zwei Indizes $p_{1,2}$, p_2 mit $p_1 < p_{1,2} < p_2 \leq N$ und verteilen die Moduln wie folgt:

$$\begin{array}{lll} \text{Giantsteps,} & \text{äußere Schleife} & : l_{p_1+1}, \ldots, l_{p_{1,2}} \\ \text{Giantsteps,} & \text{innere Schleife} & : l_{p_{1,2}+1}, \ldots, l_{p_2} \\ \text{Babysteps} & & : l_{p_2+1}, \ldots, l_N. \end{array}$$

Damit erhalten wir folgende Moduln:

$$\begin{aligned} M_3 &= \prod_{i=1}^{p_1} l_i \\ M_{1,1} &= \prod_{i=p_1+1}^{p_{1,2}} l_i \\ M_{1,2} &= \prod_{i=p_{1,2}+1}^{p_2} l_i \\ M_2 &= \prod_{i=p_2+1}^{N} l_i \end{aligned}$$

Anhand der Indizes $p_{1,2}$, p_2 können wir nun steuern, in welchem Verhältnis die einzelnen Moduln aufgeteilt werden sollen. Seien

$$K := \prod_{i=p_2+1}^{N} n_i$$

die Anzahl aller möglichen Kongruenzen, K_B bzw. K_G die Anzahl der Kongruenzen für die Baby- bzw. Giantsteps. Dann wählen wir p_2, so daß

$$K_B \approx K_G \approx \sqrt{K}.$$

Weiterhin ist eine obere Schranke $K_{B,max}$ für die Anzahl der Babysteps einstellbar. Ist $K_B > K_{B,max}$, so erhöhen wir p_2 solange, bis diese Schranke $K_{B,max}$ wieder unterschritten wird d.h. wir ordnen m Moduln $l_{p_2+1}, \ldots, l_{p_2+m}$ den Giantsteps zu. Nach dieser Verschiebung kann $(K_{B,max}/K_B) \approx n_{p_2+m}$ sein. Um ein zu großes Ungleichgewicht zwischen Baby- und Giantsteps zu vermeiden, suchen wir unter allen Giantstep-Moduln $l_{p_1+1}, \ldots, l_{p_2+m}$, die Moduln $l_{i_0}, \ldots, l_{i_n}$ heraus, für die

$$\prod_{j=0}^{n} n_{i_j} \leq \frac{K_{B,max}}{K_B} \text{ maximal}$$

ist und weisen sie wieder den Babysteps zu.

Die Bestimmung von $p_{1,2}$ zur Aufteilung der Giantsteps in innere und äußere Schleife erfolgt auf die gleiche Weise.

3.3 Ein Beispiel und Laufzeiten

In diesem Abschnitt geben wir zuerst ein Beispiel für die Babystep-Giantstep-Phase an und im Anschluß einige Laufzeiten. Für unser Beispiel betrachten wir die Kurve

$$E = (118, 69) \quad \text{über} \quad \mathbb{F}_{65537}$$

mit den folgenden Relationen, die im ersten Teil des Algorithmus (Kapitel 2) gefunden wurden:

$$\begin{aligned} c \equiv &\ 1 \text{ oder } 2 && \text{modulo } 3 \\ c \equiv &\ 2 \text{ oder } 3 && \text{modulo } 5 \\ c \equiv &\ 0 && \text{modulo } 7 \\ c \equiv &\ 3 \text{ oder } 8 && \text{modulo } 11. \end{aligned}$$

Ordnen wir diese Relationen aufsteigend nach ihrer Dichte, so erhalten wir die folgende Konstellation:

$$\begin{aligned} c \equiv &\ 0 && \text{modulo } 7 \\ c \equiv &\ 3 \text{ oder } 8 && \text{modulo } 11 \\ c \equiv &\ 2 \text{ oder } 3 && \text{modulo } 5 \\ c \equiv &\ 1 \text{ oder } 2 && \text{modulo } 3. \end{aligned}$$

Diese Kongruenzen teilen wir in Babysteps (Modul M_2), innere Schleife der Giantsteps (Modul $M_{1,2}$) und äußere Schleife der Giantsteps (Modul $M_{1,1}$) auf; außerdem sammeln wir die Kongruenzen mit nur einer Möglichkeit für c im Modul M_3. Wir müssen in diesem Beispiel keinen Chinesischen Restsatz anwenden, da wir nur wenige Relationen haben. Die Babysteps werden von der dichtesten Relation gebildet, also gilt:

$$M_2 = 3.$$

Die nächst dichteste Relation bildet die innere Schleife der Giantsteps und die darauffolgende Relation mit mehr als einer Möglichkeit für c die äußere Schleife:

$$M_{1,2} = 5, \quad M_{1,1} = 11.$$

Wir kennen c exakt modulo 7, also:

$$M_3 = 7, \quad c_3 \equiv 0 \bmod 7.$$

Wir finden den Punkt

$$Q = (64982, 33326)$$

auf der Kurve und beginnen mit den **Babysteps**. Wie beschrieben, berechnen sich die R_2-Werte nach der Formel:

$$R_2 = \left(\frac{c - c_3}{M_1 M_3} \bmod M_2, \text{ pos. oder neg. kleinster Rest} \right).$$

Da $c_3 \equiv 0 \bmod M_2$ und $M_1 \cdot M_3 = 55 \cdot 7 \equiv 1 \bmod M_2$ ist, erhalten wir

$$\begin{array}{lrr} R_{2,+}(\text{ pos. Reste }): & 1 & 2 \\ R_{2,-}(\text{ neg. Reste }): & -2 & -1. \end{array}$$

Wir bestimmen für alle R_2-Werte den Punkt $(R_2 M_1 M_3) \cdot Q$ und berechnen uns deshalb den Punkt

$$P := (M_1 M_3) \cdot Q = 7 \cdot (64982, 33326) = (5571, 58727)$$

vor. Wir könnten nun eine Hilfstabelle aufbauen (vgl. Abschnitt 3.2.1). Dies ist aber wegen der Größe der Zahlen nicht notwendig; ein Beispiel zur Tabellenbenutzung einschließlich ggT-Trick haben wir in dem entsprechenden Abschnitt schon gegeben. Um die Punkte $R_2 \cdot P$ zu bestimmen, berechnen wir zuerst alle $R_{2,+}$-Werte und sortieren diese. Die entsprechenden Punktvielfache für $R_{2,-}$ erhalten wir als

$$R_{2,-} \cdot Q = R_{2,+} \cdot Q + (-M_2) \cdot Q.$$

Dazu bestimmen wir

$$-M_2 \cdot Q = -7 \cdot (5571, 58727) = (55024, 41024).$$

Damit können wir alle R_2-Werte sowie die Babystep-Punkte berechnen, und wir erhalten:

$R_{2,+}$	$R_{2,+} \cdot P$	$R_{2,-}$	$R_{2,-} \cdot P$	$Baby-Tab.$	$Index$	$R2-Tab.$
1	$(55024, 41024)$	-2	$(33700, 25149)$	$B[1] = 55024$	$I[1] = 1$	$R2[1] = 1$
				$B[2] = 33700$	$I[2] = -1$	
2	$(33700, 40388)$	-1	$(55024, 6810)$	$B[3] = 33700$	$I[3] = 2$	$R2[2] = 2$
				$B[4] = 55024$	$I[4] = -2$	

Sortieren wir die Babystep-Tabelle, so ergibt sich folgende Konstellation:

$Nr.$	1	2	3	4
$B:$	55024	55024	33700	33700
$I:$	1	-2	-1	1

Es fällt auf, daß in der Babystep-Tabelle jede x-Koordinate zweimal vorkommt. Dies liegt daran, daß der Wert $c_3 = 0$ ist; dadurch sind je zwei R_2-Werte additiv invers zueinander, und die entsprechenden Punktvielfachen haben die gleiche x-Koordinate. Dieses Verhalten tritt aber nur im Grundalgorithmus auf; benutzen wir den erweiterten Atkin-Algorithmus, welchen wir in Kapitel 4 beschreiben, so ist c_3 fast immer ungleich Null.

In der **Giantstep**-Phase bestimmen wir die Punktvielfachen

$$(p + 1 - c_3 - R_{1,2} M_{1,1} M_2 M_3 - R_{1,1} M_{1,2} M_2 M_3) \cdot Q;$$

deswegen berechnen wir zuerst die Punkte

$$\begin{aligned} H &:= & (p+1-c3)\cdot Q &= (17319, 19195) \\ Q_1 &:= & (-M_{1,1}M_2M_3)\cdot Q &= (65216, 28778) \\ Q_2 &:= & (-M_{1,2}M_2M_3)\cdot Q &= (4305, 27694). \end{aligned}$$

Wir beginnen mit der inneren Schleife, wobei sich die $R_{1,2}$-Werte nach folgender Formel herleiten:

$$R_{1,2} = \left(\frac{c-c_3}{M_{1,1}M_2M_3} \bmod M_{1,2}, \text{ pos. oder neg. Rest} \right).$$

Analog zur Babystep-Phase berechnen wir zuerst die $R_{1,2,+}$-Werte, d.h. die positiven Reste und daraus die negativen Reste $R_{1,2,-}$. Für die zugehörigen Punkte bedeutet dies:

$$R_{1,2,-}\cdot Q_1 = R_{1,2,+}\cdot Q_1 - M_{1,2}\cdot Q_1,$$

und dazu berechnen wir

$$-M_{1,2}\cdot Q_1 = -5\cdot(65216, 28778) = (55024, 24513).$$

Wir erhalten folgendes Bild für die innere Schleife:

$R_{1,2,+}$	$R_{1,2,+}\cdot Q_1$	$R_{1,2,-}$	$R_{2,-}\cdot Q_1$
2	$(31151, 12780)$	-3	$(4451, 58621)$
3	$(4451, 6916)$	2	$(31151, 52757)$

$R12 - Tabelle$		$R12 - Werte$	
$INN\_P[1] =$	$(31151, 12780)$	$INN\_W[1] =$	2
$INN\_P[2] =$	$(4451, 58621)$	$INN\_W[2] =$	-3
$INN\_P[3] =$	$(4451, 6916)$	$INN\_W[3] =$	3
$INN\_P[4] =$	$(31151, 52757)$	$INN\_W[4] =$	2

Dabei enthält das Feld INN_P die Punkte der inneren Schleife, und die zugehörigen $R_{1,2}$-Werte findet man im Feld INN_W.

Wir addieren alle Punkte der inneren Schleife mit H und erhalten:

$Nr.$	1	2	3	4
$INN\_P:$	$(56885, 17984)$	$(16467, 25030)$	$(61381, 38471)$	$(232, 10524)$

Die $R_{1,1}$-Werte der äußeren Schleife berechnen sich nach der Formel:

$$R_{1,1} = \left(\frac{c-c_3}{M_{1,2}M_2M_3} \bmod M_{1,1}, \text{ pos. Rest} \right).$$

Wir berechnen $M_{1,1}M_2M_3 \equiv 6 \bmod 11$ und erhalten den ersten $R_{1,1}$-Wert als

$$R_{1,1} \equiv \frac{3}{6} \equiv 6 \bmod 11.$$

Wir bestimmen nun das Punktvielfache $R_{1,1} \cdot Q_2$, addieren dieses zu den Punkten der inneren Schleife und suchen die x-Koordinate des jeweiligen Punktes in der Babystep-Tabelle:

$$6 \cdot Q_2 = (59670, 19971).$$

Addieren wir dies zum ersten Punkt der inneren Schleife, so erhalten wir

$$6 \cdot Q_2 + INN\_P[1] = (33700, 25149).$$

Wir finden diese x-Koordinate zweimal in der Babystep-Tabelle, nämlich $B[3]$ und $B[4]$. Der Index I von $B[3]$ ist $I[3] = -1$. Um den vollständigen Babystep-Punkt zu erhalten, müssen wir

$$(R2[1] - M_2)(M_1M_3) \cdot Q = (33700, 25149)$$

berechnen. Da dieser Punkt mit dem Giantstep-Punkt übereinstimmt, haben wir die wahrscheinliche Gruppenordnung gefunden und erhalten:

$$\begin{aligned}
\#E(\mathbb{F}_{65537}) &= p + 1 - c_3 - M_2M_3 \cdot (R_{1,1}M_{1,2} + R_{1,2}M_{1,1}) - M_1M_3R_2 \\
&= 65538 - 21 \cdot (6 \cdot 5 + 2 \cdot 11) - 55 \cdot 7 \cdot (-2) \\
&= 65216.
\end{aligned}$$

Im folgenden geben wir einige Laufzeiten für die Berechnung der Babysteps und Giantsteps an. Wir haben dabei Größenordnungen gewählt, die in der Praxis auftreten. In der ersten Spalte der Tabelle steht jeweils die Anzahl der Dezimalstellen der Charakteristik p. Bei den Babysteps unterscheiden wir Berechnungen die ohne Hilfstabelle (o.T.), mit Hilfstabelle (m.T.) und mit Hilfstabelle und ggT-Trick (T.+ggT) durchgeführt wurden. Zur Berechnung der Giantsteps haben wir die beschriebene Doppelschleife verwendet, d.h. wir haben die Moduln der Giantsteps in eine innere und eine äußere Schleife aufgeteilt. In der Spalte „Anzahl" haben wir für die Babysteps die Anzahl N aller Babystep-Punkte angegeben, d.h. wir haben $N/2$ $R_{2,+}$-Werte, für die wir die entsprechenden Punktvielfachen mit den verschiedenen Verfahren berechnen und $N/2$ $R_{2,-}$-Werte, für die wir jeweils eine Punktaddition durchführen. Für die Giantsteps haben wir die Anzahl N aller Punkte angegeben, die auch wirklich berechnet wurden, d.h. wir haben N Punkte getestet bis wir einen Giantstep-Punkt in der Babystep-Tabelle gefunden haben.

$dd(p)$	Babysteps				Giantsteps	
	Anzahl	o.T.	m.T.	T.+ggT	Anzahl	Zeit
50	276480	2 h 30 min	34 min	35 min	49440	13 min
60	207360	1 h 57 min	32 min	33 min	33280	9 min
70	96768	1 h 13 min	18 min	21 min	53760	18 min
80	49152	1 h 10 min	14 min	12 min	6288	3 min
90	69120	1 h 55 min	25 min	21 min	46848	24 min
100	20736		8 min	7 min	144	35 sec
100	276480		4 h 37 min	2 h 15 min	1452	1 min
200	92160		1 h 52 min	1 h 31 min	4320	9 min
250	27648		2 h 01 min	57 min	2080	8 min
375	17472		45 min	45 min	3360	24 min
375	59904		2 h 24 min	1 h 34 min	349056	24 h

Wir sehen, daß die Benutzung der Tabelle einen hohen Geschwindigkeitsgewinn bringt. Ferner stellen wir fest, daß die Anwendung des ggT-Tricks die Laufzeit nicht immer verbessert. Wir erzielen keinen Laufzeitgewinn, falls viele der zu berechnenden Differenzpunkte in der Hilfstabelle liegen (vgl. Bemerkung 3.3). Schauen wir uns an, wieviele Punkte jeweils in der Hilfstabelle lagen, so wird unsere Überlegung bestätigt:

$dd(p)$	#Differenzpunkte	
	insgesamt	in Tabelle
50	138240	138232
60	103680	103680
70	48384	48150
80	24576	10839
90	34560	17632
100	10368	6223
100	138240	952
200	46080	24771
250	13824	96
375	8736	8736
375	29952	29952

Es fällt auf, daß im zweiten Test mit der 375-stelligen Charakteristik alle Differenzpunkte in der Tabelle gefunden wurden und dennoch die Berechnung der Babysteps mit ggT-Trick schneller war. In diesem Fall lagen die R_2-Werte so dicht, daß auch bei festem Bezugspunkt die Differenzpunkte eines Päckchens in der Tabelle lagen.

Kapitel 4

Die Atkin-Elkies Erweiterung

In diesem Kapitel werden wir den Grundalgorithmus aus Kapitel 2 ausbauen; die hierbei verwendeten Ideen stammen von A.O.L. Atkin und Noam Elkies ([At92], [El]). In den vorangegangenen Kapiteln haben wir gesehen, wie wir die Spur c des Frobeniusendomorphismus modulo kleiner Primzahlen l berechnen und diese Ergebnisse mit einem Babystep-Giantstep-Algorithmus zusammensetzen können. Existieren n mögliche Werte für c modulo l, so geht n als Faktor in die Anzahl der Babysteps bzw. Giantsteps ein. Da n fast immer größer gleich 2 ist und die Punktoperationen teuer sind, können wir mit dem reinen Atkin-Algorithmus keine Gruppenordnungen für Charakteristiken mit mehr als 70 Dezimalstellen berechnen.
Beispiel:

$$E = (9051969, 11081969), \;\; p = 10^{59} + 19$$
$$c \bmod l \text{ berechnet für } 3 \leq l \leq 79$$
$$\#\text{Babysteps: } 185\,794\,560$$
$$\#\text{Giantsteps:} 1\,447\,821\,312$$

Das Verfahren von Atkin und Elkies gibt uns die Möglichkeit in den Fällen, in denen das l-te modulare Polynom eine Nullstelle in $\mathbb{F}_p$ hat, die Spur c modulo l exakt zu berechnen, d.h. l geht mit Faktor 1 in die Anzahl der Babysteps bzw. Giantsteps ein. Damit sind wir in der Lage die Gruppenordnung für Charakteristiken mit bis zu 375 Dezimalstellen zu berechnen.

Wir nehmen für den Rest des Kapitels an, daß für die ungerade Primzahl $l < p$ das l-te modulare Polynom eine Nullstelle in $\mathbb{F}_p$ hat. Betrachten wir unter dieser Voraussetzung den Zusammenhang zwischen Zerfällungsverhalten des modularen Polynoms und der Lage der Nullstellen von

$$X^2 - c \cdot X + p \in \mathbb{F}_l[X] \tag{4.1}$$

(vgl. Kapitel 2), so wissen wir, daß diese quadratische Gleichung, welche das charakteristische Polynom von $\Phi_{E_{|E[l]}}$ ist, in $\mathbb{F}_l$ zerfällt; also hat der Frobeniusendomorphismus eingeschränkt auf die l-Torsionsgruppe $E[l]$ einen **Eigenwert** α **in** $\mathbb{F}_l$. Wir werden diesen Eigenwert berechnen und erhalten dann

$$X^2 - c \cdot X + p \;\; \equiv_l \;\; (X - \alpha) \cdot (X - \beta)$$

$$\begin{aligned} p &\equiv_l \alpha \cdot \beta \\ c &\equiv_l \alpha + \beta \end{aligned}$$

und damit die Spur des Frobeniusendomorphismus modulo l als

$$c \equiv \alpha + \frac{p}{\alpha} \bmod l. \tag{4.2}$$

Wie berechnen wir den Eigenwert α ?

Zuerst betrachten wir, wie die Eigenräume zu den Eigenwerten von $\Phi_{E_{|E[l]}}$ liegen. Zur Vereinfachung bezeichnen wir auch in diesem Kapitel die Einschränkung des Frobeniusendomorphismus Φ_E auf die l-Torsionsgruppe $E[l]$ ebenfalls mit Φ_E.

4.1. Lemma *Ist $d = ord(\frac{\alpha}{\beta})$, wobei α, β die Nullstellen des charakteristischen Polynoms (4.1) sind, und zerfällt dieses über $\mathbb{F}_l$, so besteht folgender Zusammenhang zwischen Zerfällungstyp des l-ten modularen Polynoms und den Eigenräumen zu den Eigenwerten α, β von Φ_E:*

$(1l)$	$\Leftrightarrow$	*$\alpha = \beta$ mit Eigenraum $C \leq E[l]$, wobei C die unter Φ_E invariante Untergruppe der Ordnung l ist .*
$(1 \cdots 1)$	$\Leftrightarrow$	*$\alpha = \beta$ mit Eigenraum $E[l]$.*
$(11d \cdots d)$	$\Leftrightarrow$	*$\alpha \neq \beta$, wobei die beiden unter Φ_E invarianten Untergruppen die beiden verschiedenen Eigenräume der Ordnung l bilden.*

Beweis: Der Zusammenhang zwischen Zerfällungstyp und α, β folgt aus den Sätzen von Abschnitt 2.1.

$(1l)$ Es existiert nur eine unter Φ_E invariante Untergruppe, also bildet sie den Eigenraum zum Eigenwert α.

$(1 \cdots 1)$ Sei C eine der $l+1$ invarianten Untergruppen von Φ_E, $C = \langle P \rangle$, $P \in E[l]$. Dann gilt:

$$\Phi_E(P) = \gamma \cdot P \Rightarrow \gamma = \alpha$$

und C ist Unterraum des Eigenraumes. Als Vereinigung aller invarianten Untergruppen bildet $E[l]$ den Eigenraum.

$(11d \cdots d)$ Es existieren 2 verschiedene Eigenwerte α, β und 2 invariante Untergruppen $C_1, C_2 \leq E[l]$. Da ein Eigenraum eines Eigenwertes eine unter Φ_E invariante Untergruppe von $E[l]$ sein muß, ist C_1 Eigenraum zu α und C_2 Eigenraum zu β. ∎

Sei $C \leq E[l]$ eine unter Φ_E invariante Untergruppe der Ordnung l, die Unterraum des Eigenraumes zu α ist. Dann gilt für jeden Eigenvektor $v \in C$

$$\Phi_E(v) = \alpha \cdot v. \tag{4.3}$$

Wäre uns ein Eigenvektor $v \in C$ bekannt, so könnten wir mit $\alpha = 1$ starten und solange erhöhen, bis die Gleichung (4.3) erfüllt und der Eigenwert gefunden ist. Die Atkin-Elkies-Erweiterung liefert uns keinen einzelnen Eigenvektor, sondern eine Menge von x-Koordinaten von $(l-1)$ Eigenvektoren von α, gegeben als Nullstellen des Polynoms

$$f_C(x) = \prod_{\pm P \in C \setminus \{\mathcal{O}\}} (x - x(P)) \in \overline{\mathbb{F}_p}[x] \tag{4.4}$$

vom Grad $\frac{l-1}{2}$, da inverse Punkte die gleiche x-Koordinate besitzen.

Da die Galoisgruppe $Gal(f_C, \mathbb{F}_p)$ vom Frobeniusendomorphismus

$$\begin{aligned} \Phi : \quad \overline{\mathbb{F}_p} &\longrightarrow \overline{\mathbb{F}_p} \\ x &\longmapsto x^p \end{aligned}$$

erzeugt wird, und der Eigenunterraum C invariant unter dem Frobeniusendomorphismus

$$\begin{aligned} \Phi_E : \quad E[l] &\longrightarrow E[l] \\ (x, y) &\longmapsto (x^p, y^p) \end{aligned}$$

ist, permutieren die Elemente der Galoisgruppe $Gal(f_C, \mathbb{F}_p)$ die Nullstellen von f_C, und somit liegen die Koeffizienten von f_C im Fixkörper, d.h.

$$f_C(x) \in \mathbb{F}_p[x].$$

Wie wir sehen werden, können wir für $\alpha \in \mathbb{Z}$ das formale Punktvielfache $\alpha \cdot (x, y)$ des Punktes (x, y) in der Form

$$\alpha \cdot (x, y) = (h_1(x), y \cdot h_2(x))$$

schreiben, wobei $h_i(x) \in \mathbb{F}_p(x)$ $(i = 1, 2)$ sind.

Beispiel:
Für den Punkt (x, y) erhalten wir mit den Formeln zur Punktaddition aus Satz 1.3

$$2 \cdot (x, y) = \left(\frac{x^4 - 2ax^2 - 8bx + a^2}{4(x^3 + ax + b)}, y \cdot \left(-1 + \frac{(3x^2 + a) \cdot (3x^4 + 6ax^2 + 12bx - a^2)}{8 \cdot (x^3 + ax + b)^2} \right) \right).$$

Weiterhin gilt wegen $y^p = y \cdot (x^3 + ax + b)^{\frac{p-1}{2}}$

$$\Phi_E(x, y) = \left(x^p, y \cdot \left(x^3 + ax + b \right)^{\frac{p-1}{2}} \right).$$

Um den gesuchten Eigenwert α, $1 \leq \alpha < l$, zu bestimmen, testen wir für alle ganzen Werte in diesem Intervall, ob

$$\Phi_E(x, y) = \alpha \cdot (x, y) \qquad \forall \ (x, y) \in C.$$

In Polynomgleichungen bedeutet dies:

$$\begin{array}{rcll} \left(x^p, y\cdot\left(x^3+ax+b\right)^{\frac{p-1}{2}}\right) & = & (h_1(x), y\cdot h_2(x)) & \forall\ (x,y)\in C \\ & \Longleftrightarrow & & \\ x^p & = & h_1(x) & \forall\ x \text{ mit } (x,y)\in C \\ \left(x^3+ax+b\right)^{\frac{p-1}{2}} & = & h_2(x) & \forall\ x \text{ mit } (x,y)\in C. \end{array}$$

Da uns die x-Koordinaten aller Punkte aus C als Nullstellen von f_C gegeben sind, können wir alle auftretenden Polynome modulo f_C berechnen und überprüfen, ob

$$\begin{array}{rcl} x^p - h_1(x) & \equiv & 0 \text{ modulo } f_C(x) \\ \left(x^3+ax+b\right)^{\frac{p-1}{2}} - h_2(x) & \equiv & 0 \text{ modulo } f_C(x). \end{array}$$

Diese Gleichungen sind eindeutig für einen Wert $\alpha,\ 1\leq\alpha\leq l$, erfüllt, und dies ist unser gesuchter Eigenwert.

4.2. Bemerkung Dieses Verfahren lehnt sich an den Algorithmus von Schoof, vgl. [Mü91, Kap.5], an. Im Schoof-Algorithmus rechnet man allerdings modulo dem l-ten Divisionspolynom, welches den Grad $\frac{l^2-1}{2}$ hat. Unser Polynom f_C ist ein Teiler des l-ten Divisionspolynoms vom Grad $\frac{l-1}{2}$; diese Reduktion des Polynomgrades ist der entscheidende Vorteil des hier vorgestellten Verfahrens.

Es bleiben 2 Fragen offen:

1. Wie berechnen wir das Polynom $f_C(x)$?
2. Wie bestimmen wir die Punktvielfachen $\alpha\cdot(x,y)$ mod $f_C(x)$?

4.1 Berechnung des Polynoms $f_C(x)$

Zur Bestimmung von

$$f_C(x) = x^d + a_{d-1}x^{d-1} + \ldots + a_0 \in \mathbb{F}_p[x],\ \ d=(l-1)/2,$$

betrachten wir elliptische Kurven über den komplexen Zahlen $\mathbb{C}$. Wir gehen nicht näher auf die theoretischen Hintergründe ein, die ausführlich in [Mü94, Bestimmung der Koeffizienten von $f_C(x)$] beschrieben werden, sondern formulieren nur die für den Algorithmus benötigten Tatsachen. Wir erhalten über $\mathbb{C}$ eine Gleichung für das Polynom f_C; mit Hilfe dieser Gleichung führen wir einen Koeffizientenvergleich durch und bestimmen so die Koeffizienten von f_C. Die auftretenden Berechnungen können wir über $\mathbb{F}_p$ ausführen.

4.3. Definition *Unter einem* **zweidimensionalen Gitter L** $\subseteq \mathbb{C}$ *verstehen wir die Menge*

$$L = \{x_1 \cdot \omega_1 + x_2 \cdot \omega_2 \mid x_1, x_2 \in \mathbb{Z}\},$$

wobei $\omega_1, \omega_2 \in \mathbb{C}$ linear unabhängig über $\mathbb{R}$ sind.

Die **Weierstrass $\wp$-Funktion zu** L ist über $\mathbb{C}$ definiert als

4.4. Definition

$$\wp(z, L) = \frac{1}{z^2} + \sum_{w \in L \setminus \{0\}} \left(\frac{1}{(z-w)^2} - \frac{1}{w^2} \right).$$

4.5. Lemma *Fassen wir die Kurve $E = (a, b)$ über $\mathbb{F}_p$ als über $\mathbb{C}$ definierte Kurve auf, d.h. wir betrachten die Kurvenparameter als komplexe Zahlen und legen für die Punktmenge den Körper der komplexen Zahlen zugrunde, so existiert in $\mathbb{C}$ ein 2-dimensionales Gitter $L = \mathbb{Z}\lambda + \mathbb{Z}\lambda\tau,\ \lambda = \frac{\pi}{\sqrt{3}},\ \tau \in \mathbb{C}$ mit $\Im(\tau) > 0$, so daß*

$$\mathbb{C}/L \cong E(\mathbb{C})$$

mit dem Isomorphismus

$$\overline{z} \longmapsto \left(\wp(z, L), \frac{1}{2}\wp'(z, L) \right), \quad z \in \mathbb{C}.$$

Beweis: [Mü94, Kapitel über modulare Polynome] ∎

4.6. Bemerkung Wenden wir dieses Lemma auf unsere Kurve E an, so erhalten wir ein Gitter L und damit verbunden den Wert τ; L und τ seien für den Rest des Abschnitts stets diese Werte.

Für die Weierstrass'sche $\wp$-Funktion erhalten wir die folgende Entwicklung:

$$\wp(z, L) = \frac{1}{z^2} + \sum_{k=1}^{\infty} c_k \cdot z^{2k} \tag{4.5}$$

mit

$$\begin{aligned} c_1 &= -\frac{a}{5}, \qquad c_2 = -\frac{b}{7} \\ c_k &= \frac{3}{(k-2)(2k+3)} \cdot \sum_{h=1}^{k-2} c_h \cdot c_{k-1-h} \text{ für } k \geq 3. \end{aligned} \tag{4.6}$$

Wir betrachten in der Faktorgruppe $\mathbb{C}/L$ das Element $\overline{(\frac{\lambda}{l})}$ und die von diesem Element erzeugte Untergruppe

$$C' = \left\{ n \cdot \overline{\left(\frac{\lambda}{l}\right)} \mid 1 \leq n \leq l \right\}$$

der Ordnung l. Die dazu (gemäß Lemma 4.5) isomorphe Untergruppe C in $E(\mathbb{C})$ ist

$$C = \left\{ \left(\wp\left(\frac{n\lambda}{l}, L\right), \frac{1}{2}\wp'\left(\frac{n\lambda}{l}, L\right) \right) \mid 1 \le n \le l \right\}.$$

Dann existiert eine elliptische Kurve $\tilde{E} = (\tilde{a}, \tilde{b})$ über $\mathbb{C}$, die isomorph zur Faktorgruppe $E(\mathbb{C})/C$ ist, d.h. $\tilde{E}(\mathbb{C}) \cong E(\mathbb{C})/C$. Betrachten wir das Gitter $\tilde{L} = \mathbb{Z}\frac{\lambda}{l} + \mathbb{Z}\lambda\tau$, so gilt $\tilde{E}(\mathbb{C}) \cong \mathbb{C}/\tilde{L}$. Definieren wir P_1 als Summe von x-Koordinaten der Punkte in C,

$$P_1 = \sum_{n=1}^{d} \wp(\frac{n\lambda}{l}, L),$$

sowie die Koeffizienten von $\wp(z, \tilde{L})$ als $\tilde{c_k}$, $k = 1, 2, \ldots$, die sich aus $\tilde{a}, \tilde{b}$ gemäß (4.6) herleiten, so gilt der folgende

4.7. Satz *Für das Polynom f_C gilt die folgende Gleichung über $\mathbb{C}$:*

$$f_C\left(\wp(z, L)\right) = z^{1-l} \cdot \sum_{r=0}^{\infty} \frac{\left(\sum_{k=1}^{\infty} (l \cdot c_k - \tilde{c_k}) \frac{z^{2k+2}}{(2k+1)(2k+2)} - P_1 z^2\right)^r}{r!}. \tag{4.7}$$

Die Koeffizienten des Polynoms f_C liegen in $\mathbb{Q}[a, b, \tilde{a}, \tilde{b}, P_1]$.

Beweis: [Mü94, Bestimmung der Koeffizienten von $f_C(x)$] ■

Im Algorithmus wenden wir genau die gleichen Formeln über $\mathbb{F}_p$ an, d.h. wir bestimmen $\tilde{E} = (\tilde{a}, \tilde{b})$ und P_1 über $\mathbb{F}_p$, berechnen die Koeffizienten der Fourierreihe der Weierstrass'schen $\wp$-Funktion für die Kurve E und $\tilde{E}$ über $\mathbb{F}_p$ und erhalten die Koeffizienten des Polynoms $f_C(x)$ durch Koeffizientenvergleich in der im Satz angegebenen Formel. Insbesondere wird durch unsere Voraussetzung $l < p$ gewährleistet, daß bei der Reduktion modulo p nicht durch Null dividiert wird (vgl. [Mü94, Kap. 7]).

Wir werden im folgenden zuerst den Koeffizientenvergleich beschreiben und anschließend zeigen wie wir $\tilde{E}$ sowie P_1 herleiten können.

4.1.1 Der Koeffizientenvergleich

Fourierreihenarithmetik

Wie aus Gleichung (4.7) ersichtlich wird müssen wir Fourierreihen berechnen. Wir haben dazu eine Fourierreihenarithmetik implementiert, in der Reihen durch endlich viele Koeffizienten dargestellt werden, d.h. die Reihe

$$z^n \cdot \sum_{k=0}^{\infty} a_k \cdot z^k, \quad a_k \in \mathbb{F}_p,$$

wird repräsentiert durch

$$(n, a_0, \ldots, a_M), \ \ n \in \mathbb{Z},$$

wobei $M \in \mathbb{N}$ eine in Abhängigkeit vom Problem zu bestimmende Genauigkeit ist; wir sagen auch, daß die Reihe bis zum Exponenten $M+n$ bekannt ist bzw. die Präzision $M+1$ hat. Den Exponenten n bezeichnen wir auch als Startexponenten der Reihe; wir wählen diesen stets so, daß $a_0 \neq 0$ ist.

Multiplizieren wir zwei Reihen

$$f_1(z) = z^{n_1} \cdot \sum_{k=0}^{M_1} a_k \cdot z^k, \quad f_2(z) = z^{n_2} \cdot \sum_{k=0}^{M_2} b_k \cdot z^k,$$

so ist das Produkt von der Form

$$(f_1 \cdot f_2)(z) = z^{n_1+n_2} \cdot \sum_{k=0}^{min(M_1,M_2)} c_k \cdot z^k \quad \text{mit} \quad c_k = \sum_{i=0}^{k} a_i \cdot b_{k-i},$$

d.h. die Präzision des Produktes ist das Minimum der Ausgangspräzisionen.

Berechnen wir für $d \in \mathbb{N}$ durch sukzessives Multiplizieren f_1^d, so erhalten wir

$$f_1^d(z) = z^{d \cdot n_1} \cdot \sum_{k=0}^{M_1} \alpha_k \cdot z^k$$

d.h. die Präzision bleibt gleich, aber der höchste bekannte Exponent „verschiebt sich".

Weitere Informationen zur Implementierung der Fourierreihenarithmetik findet man in [Le94, Kapitel 3].

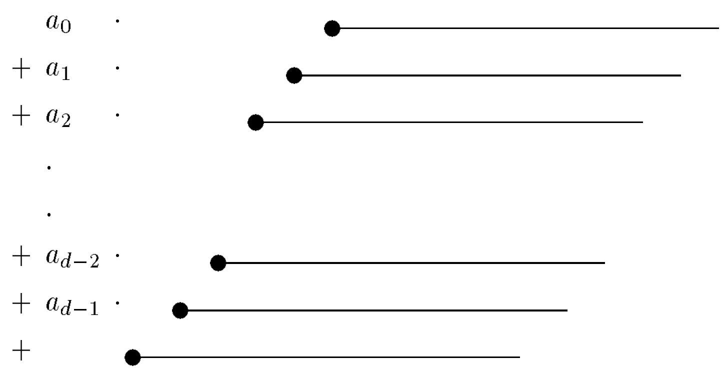
Da wir die auftretenden Fourierreihen durch endlich viele Koeffizienten darstellen, müssen wir für jede Reihe bestimmen, wieviele ihrer Koeffizienten wir zum Koeffizientenvergleich benötigen. Dazu ermitteln wir zuerst den größten Exponenten, an dem ein Vergleich der Koeffizienten durchgeführt wird und berechnen dann für jede Reihe die erforderliche Länge.

Die zu bestimmenden Koeffizienten unseres Polynoms sind mit den Potenzen der Weierstrass'schen $\wp$-Funktion verknüpft:

$$f_C(\wp(z, L)) = \wp(z, L)^d + a_{d-1} \cdot \wp(z, L)^{d-1} + \ldots + a_1 \cdot \wp(z, L) + a_0, \quad d = (l-1)/2.$$

Berechnen wir alle Potenzen sukzessive, so wissen wir, daß die Präzision der Reihen gleichbleibt, und der Startexponent wandert. Da die $\wp$-Funktion mit negativem Startexponenten, -2, beginnt, verschiebt sich die i-te Potenz um $2 \cdot (i-1)$ Exponenten nach links. Wir haben versucht, dies in Abbildung 4.1 zu veranschaulichen. In dieser Darstellung symbolisiert eine Linie rechts von a_i die Reihe $\wp(z, L)^i$, $0 \leq i \leq d-1$, und der Punkt am Anfang einer Linie markiert den Startexponenten der entsprechenden Reihe. Wir erhalten den Koeffizienten a_i des Polynoms $f_C(x)$ $\quad (i = d-1, \ldots, 0)$ durch Koeffizientenvergleich am Exponenten $-2i$; damit gilt

4.8. Lemma *Alle am Koeffizientenvergleich beteiligten Reihen müssen bis zum Exponenten 0 einschließlich bekannt sein.*



= rechte Seite von Gleichung (4.7).

Abbildung 4.1: Reihendarstellung von $f_C(\wp(z,L))$.

Die linke Seite von Gleichung (4.7)

Die Reihe $\wp(z,L)$ muß für den Koeffizientenvergleich bis zum Exponenten 0 bekannt sein. Allerdings reicht es nicht aus, die Reihe nur bis zu diesem Exponenten zu berechnen, weil auch die Potenzen von $\wp(z,L)$ benötigt werden. Da sich bei diesen der Startexponent verkleinert, die Länge aber konstant bleibt, müssen wir $\wp(z,L)$ bis zu einem Exponenten größer 0 berechnen.

Die $\wp$-Potenz mit dem kleinsten Startexponenten ist $\wp(z,L)^d$. Aus

$$\wp(z,L) = z^{-2} \cdot \sum_{k=0}^{M} a_k \cdot z^k$$

erhalten wir

$$\wp(z,L)^d = z^{-2d} \cdot \sum_{k=0}^{M} b_k \cdot z^k$$

d.h. der höchste bekannte Exponent von $\wp(z,L)^d$ ist $-2d+M$.

Also muß gelten:

$$\begin{aligned} -2d + M &= 0 \\ \Leftrightarrow \quad M &= 2d = l-1. \end{aligned}$$

Der größte bekannte Exponent von $\wp(z,L)$ muß also $M-2 = l-3$ sein; deshalb benötigen wir die Werte c_k (vgl. (4.6)) für $k = 1,\ldots,\frac{l-3}{2}$.

4.9. Satz *Für den Koeffizientenvergleich benötigen wir die Weierstrass'sche $\wp$ Funktion in der Form*

$$\wp(z,L) = z^{-2} \cdot \sum_{k=0}^{l-1} a_k \cdot z^k,$$

wobei

$$a_0 = 1, \quad a_1 = a_2 = 0,$$
$$a_k = 0 \quad \textit{für } k \geq 3 \textit{ ungerade},$$
$$a_k = c_{\frac{k-2}{2}} \quad \textit{gemäß (4.6) für } k \geq 4 \textit{ gerade}.$$

Die rechte Seite von Gleichung (4.7)

Wenden wir uns der Doppelreihe auf der rechten Seite der Gleichung (4.7) zu und definieren g als die innere Reihe

$$g(z) := \sum_{k=1}^{\infty}(lc_k - \tilde{c}_k)\frac{z^{2k+2}}{(2k+1)(2k+2)} - P_1 z^2,$$

so erhalten wir die Gestalt

$$f_C(\wp(z, L)) = z^{1-l} \cdot \sum_{r=0}^{\infty} \frac{g(z)^r}{r!}.$$

Der Startexponent der Reihe $g(z)$ sei n. Dann beginnt die r-te Potenz von g bei $n \cdot r$, und die Reihe $z^{1-l} \cdot g^r$ hat den Startexponenten $n \cdot r + 1 - l$. Da die Reihen nur bis zum Exponenten 0 bekannt sein müssen, ist die obere Grenze r_{max} für r

$$n \cdot r_{max} + 1 - l \leq 0, \quad \Leftrightarrow \quad r_{max} \leq \frac{l-1}{n}.$$

Damit wissen wir, wieviele Potenzen von $g(z)$ zu berechnen sind. Jetzt sind wir noch an der Präzision von $g(z)$ interessiert. Die Reihe $z^{1-l} \cdot g^r$ hat den kleinsten Startexponenten für $r = 1$ (für $r = 0$ braucht keine Reihe berechnet werden), nämlich $1 - l + n$. Für

$$g(z) = z^n \cdot \sum_{k=0}^{M} a_k \cdot z^k$$

ist $1 - l + n + M$ der größte Exponent von $z^{1-l} \cdot g(z)$; mit Lemma 4.8 gilt: M = l-(n+1).

4.10. Satz *Für den Koeffizientenvergleich benötigen wir*

$$g(z) = \sum_{k=1}^{M_g}(lc_k - \tilde{c}_k)\frac{z^{2k+2}}{(2k+1)(2k+2)} - P_1 z^2 \tag{4.8}$$

bis $2M_g + 2 = M + n \Leftrightarrow M_g = \frac{l-3}{2}$ *sowie*

$$\sum_{r=0}^{\lfloor\frac{l-1}{n}\rfloor} \frac{g(z)^r}{r!}. \tag{4.9}$$

Mit den berechneten Genauigkeiten erhalten wir den nachfolgend angegebenen Algorithmus.

Koeffizientenvergleich zur Bestimmung von f_C

EINGABE: $p, E, l, \tilde{E}, P_1$
AUSGABE: $f_C(x) = x^d + a_{d-1}x^{d-1} + \ldots + a_0 \in \mathbb{F}_p[x]$

(1) Berechne c_k und $\tilde{c_k}$ für $k = 1, \ldots, \frac{l-3}{2}$ gemäß (4.6).
(2) $g(z) = \sum_{k=1}^{(l-3)/2}(l \cdot c_k - \tilde{c_k})\frac{z^{2k+2}}{(2k+1)(2k+2)} - P_1 z^2$.
(3) Bestimme n als kleinsten Exponenten in $g(z)$, so daß der zugehörige Koeffizient ungleich Null ist.
(4) Berechne durch sukzessives Multiplizieren von $g(z)$ die Reihe $z^{1-l} \cdot \sum_{r=0}^{\lfloor (l-1)/n \rfloor} \frac{g(z)^r}{r!} =: RS$.
(5) Berechne $\wp(z, L) = \frac{1}{z^2} + \sum_{k=1}^{(l-3)/2} c_k \cdot z^{2k}$ und die Potenzen von $\wp(z, L)$ bis $d = (l-1)/2$.
(6) $RS = RS - \wp(z, L)^d$.
(7) **for** (i = d-1 bis 1) **do**
(8) Berechne den i-ten Koeffizienten a_i von f_C durch Koeffizientenvergleich an der Stelle z^{-2i}.
(9) $RS = RS - a_i \cdot \wp(z, L)^i$.
(10) **od**
(11) Berechne a_0 durch Koeffizientenvergleich an der Stelle z^0.

4.11. Bemerkung Um Speicherplatz zu sparen, berechnen wir in der Praxis nicht alle d Potenzen der Weierstrass'schen $\wp$-Funktion, sondern wir benutzen eine Babystep-Giantstep-Strategie und müssen somit nur $2\sqrt{d}$ viele Potenzen der Länge l im Speicher halten. Da man jede Potenz nur einmal benötigt, verschlechtert dies nicht die Laufzeit. Man kann die Potenzen nicht sukzessive berechnen und jeweils nur 2 Reihen abspeichern, da man sie in umgekehrter Reihenfolge benötigt.

Beispiel: $E = (3011, 68)$ über $\mathbb{F}_{109}$, $l = 11$.

Seien $\tilde{E} = (73, 42)$ und $P_1 = 100$. Wir berechnen zuerst die Koeffizienten gemäß Formel (4.6):

i	1	2	3	4
c_i	30	37	82	55
$\tilde{c}_i$	29	103	26	12

Wir erhalten die Reihe g als

$$g(z) = 9z^2 + 16z^4 + 61z^6 + 39z^8 + 95z^{10}$$

und somit die rechte Seite von Gleichung (4.7):

$$RS = z^{-10} + 9z^{-8} + 2z^{-6} + 54z^{-4} + 16z^{-2} + 104z^0.$$

Wir berechnen die Weierstrass'sche $\wp$-Funktion

$$\wp(z, L) = z^{-2} + 30z^2 + 37z^4 + 82z^6 + 55z^8$$

und bestimmen als Babystep

$$\wp(z, L)^2 = z^{-4} + 60z^0 + 74z^2 + 83z^4 + 41z^6$$

sowie als Giantstep

$$\wp(z, L)^3 = z^{-6} + 90z^{-2} + 2z^0 + 3z^2 + 67z^4.$$

Wir bestimmen mit einer Multiplikation $\wp(z, L)^5$, subtrahieren dies von RS und erhalten als neue rechte Seite

$$RS = 9z^{-8} + 70z^{-6} + 87z^{-4} + 89z^{-2} + 83z^0.$$

Durch Koeffizientenvergleich an der Stelle $z^{-2\cdot 4}$ erhalten wir

$$a_4 = 9.$$

Wir berechnen mit einer Multiplikation $\wp(z, L)^4$, multiplizieren mit a_4 und subtrahieren diese Reihe von der rechten Seite:

$$RS = 70z^{-6} + 97z^{-4} + 65z^{-2} + 88z^0.$$

Wir setzen dieses Verfahren fort und erhalten

$$a_3 = 70, a_2 = 97, a_1 = 87.$$

Nach der Berechnung von a_1 hat die rechte Seite die Form $RS = 14z^0$ und damit

$$a_0 = 14.$$

Wir erhalten das gesuchte Polynom $f_C(x) = x^5 + 9x^4 + 70x^3 + 97x^2 + 87x + 14$.

Wir geben noch einige Laufzeiten für die Durchführung des Koeffizientenvergleichs an. In der ersten Spalte steht jeweils die Anzahl der Dezimalstellen der verwendeten Charakteristik. Unter der Rubrik „rechte Seite“ sind die Zeiten für die Berechnung der rechten Seite von Gleichung (4.7) angegeben; diese Zeiten beinhalten nicht die Bestimmung von $\tilde{E}$ und P_1. In der vierten Spalte stehen die Zeiten zur Berechnung der Potenzen der Weierstrass'schen $\wp$-Funktion (Babysteps und Giantsteps) und in der letzten Spalte sind die

Zeiten des eigentlichen Koeffizientenvergleichs aufgeführt.

dd(p)	l	rechte Seite	$\wp$-Potenzen	Koeff.vgl.
100	53	6 sec	1 sec	4 sec
	97	22 sec	4 sec	16 sec
	101	25 sec	5 sec	17 sec
	157	1 min 11 sec	12 sec	55 sec
	199	1 min 38 sec	15 sec	1 min 22 sec
150	103	5 min 20 sec	46 sec	2 min 40 sec
	211	2 min 50 sec	26 sec	2 min 16 sec
	317	10 min 00 sec	1 min 06 sec	7 min 02 sec
200	103	2 min 32 sec	21 sec	1 min 14 sec
	193	3 min 33 sec	33 sec	2 min 50 sec
	307	13 min 16 sec	1 min 28 sec	9 min 30 sec
	401	19 min 06 sec	1 min 54 sec	14 min 17 sec
250	103	17 min 46 sec	2 min 20 sec	8 min 28 sec
	199	5 min 01 sec	44 sec	3 min 57 sec
	313	18 min 14 sec	2 min 01 sec	12 min 55 sec
300	101	1 min 35 sec	20 sec	1 min 12 sec
	317	1 h 11 min	7 min 46 sec	51 min 11 sec
	421	34 min 59 sec	3 min 25 sec	26 min 33 sec
	523	1 h 08 min	6 min 23 sec	53 min 59 sec
375	103	5 min 42 sec	46 sec	2 min 53 sec
	223	10 min 02 sec	1 min 29 sec	8 min 05 sec
	401	45 min 50 sec	4 min 23 sec	34 min 58 sec
	503	1 h 28 min	8 min 05 sec	1 h 09 min
	839	3 h 05 min	13 min 17 sec	2 h 32 min

Es fällt auf, daß bei gleicher Charakteristik die Berechnung für kleine Moduln l länger dauert als für größere Moduln. Dies liegt daran, daß für manche Moduln die Berechnung des Polynoms f_C mehrmals durchgeführt werden muß, da aufgrund falscher Werte für $\tilde{E}$ oder P_1 ein Polynom bestimmt wurde, welches kein Teiler des l-ten Divisionspolynoms ist. Die Ursache hierfür wird im nächsten Abschnitt ersichtlich. In den obigen Berechnungen wurde die Bestimmung von f_C für $(p : 150, l : 103)$ dreimal, für $(p : 250, l : 103)$ fünfmal und für $(p : 300, l : 317)$ dreimal wiederholt.

4.1.2 Bestimmung von $\tilde{E}$ und P_1

Wir haben unsere Kurve $E = (a, b)$ als Kurve über $\mathbb{C}$ betrachtet, d.h. a, b als Werte in $\mathbb{C}$ aufgefaßt. Wir wissen, daß wir ein zweidimensionales Gitter $L = \mathbb{Z}\lambda + \mathbb{Z}\lambda\tau \subseteq \mathbb{C}$ mit

$\lambda = \frac{\pi}{\sqrt{3}}$, $\tau \in \mathbb{C}$, $\Im(\tau) > 0$, finden, so daß $E(\mathbb{C}) \cong \mathbb{C}/L$. Der Wert $\tau \in \mathbb{C}$ und das Gitter L seien nun fest.

Zu dem Gitter $\tilde{L} = \mathbb{Z}\frac{\lambda}{l} + \mathbb{Z}\lambda\tau$ bestimmen wir die Kurve $\tilde{E} = (\tilde{a}, \tilde{b})$ mit $\tilde{E}(\mathbb{C}) \cong \mathbb{C}/\tilde{L}$. Wir benötigen die Kurvenparameter $\tilde{a}, \tilde{b} \in \mathbb{C}$ sowie den Wert

$$P_1 = \sum_{n=1}^{d} \wp\left(\frac{n\lambda}{l}, L\right) \in \mathbb{C},$$

um die Gleichung (4.7) zur Berechnung des Polynoms f_C anwenden zu können. Um diese Größen zu berechnen, führen wir spezielle Reihen, die sogenannten Eisensteinreihen, ein und stellen den Zusammenhang zwischen Gittern und diesen Reihen her. Wir gehen dann von unserem Gitter L über zu den Reihen, die wir an der Stelle τ betrachten. Daraus berechnen wir die Reihen an der Stelle $l\tau$ und gelangen zu einem Gitter $\hat{L}$. Dieses setzen wir in Bezug zu dem Gitter $\tilde{L}$ und erhalten die gesuchten Werte $\tilde{a}, \tilde{b}$ und P_1.

$$E \;\rightarrow\; L \;\rightarrow\; Reihe(\tau) \;\rightarrow\; Reihe(l\tau) \;\rightarrow\; \hat{L} \;\rightarrow\; \tilde{L} \;\rightarrow\; \tilde{E}$$

4.12. Satz *Betrachten wir das zweidimensionale Gitter $\hat{L} = \mathbb{Z}\lambda + \mathbb{Z}\lambda l\tau$ mit der zugehörigen Kurve $\hat{E} = (\hat{a}, \hat{b})$, so daß $\hat{E}(\mathbb{C}) \cong \mathbb{C}/\hat{L}$, so gilt*

$$\tilde{a} = l^4 \cdot \hat{a}, \quad \tilde{b} = l^6 \cdot \hat{b}.$$

Beweis: [Mü94, Bestimmung von $\tilde{c_1}, \tilde{c_2}$ und P_1] ■

Um die Werte $\hat{a}, \hat{b}$ zu bestimmen, benötigen wir folgende

4.13. Definition *Zu $\omega \in H = \{z \in \mathbb{C} \mid \Im(z) > 0\}$ definieren wir die folgenden* **normalisierten Eisensteinreihen***:*

$$E_2(\omega) = 1 - 24 \cdot \sum_{n=1}^{\infty} \sigma_1(n) \cdot q_\omega^n$$
$$E_4(\omega) = 1 + 240 \cdot \sum_{n=1}^{\infty} \sigma_3(n) \cdot q_\omega^n$$
$$E_6(\omega) = 1 - 504 \cdot \sum_{n=1}^{\infty} \sigma_5(n) \cdot q_\omega^n$$

mit $\sigma_k(n) = \sum_{d|n} d^k$, $q_\omega = exp(2\pi i\omega)$.

4.14. Satz *Ist $G = \mathbb{Z}\lambda + \mathbb{Z}\lambda\omega$ ein Gitter und $E(\mathbb{C})$ die Punktmenge einer elliptischen Kurve $E = (a, b)$ über $\mathbb{C}$, so daß $E(\mathbb{C}) \cong \mathbb{C}/G$, so gilt:*

$$a = -3 \cdot E_4(\omega), \;\; b = -2 \cdot E_6(\omega).$$

Beweis: [Mü94, Bestimmung von $\tilde{c_1}, \tilde{c_2}$ und P_1] ■

Fassen wir diese Aussagen zusammen und wenden sie auf unser Gitter $\hat{L}$ an, so erhalten wir für die Kurve $\tilde{E}$:

$$\tilde{a} = -3 \cdot l^4 \cdot E_4(l\tau), \quad \tilde{b} = -2 \cdot l^6 \cdot E_6(l\tau).$$

Definieren wir noch $E_2^*(\tau) := E_2(\tau) - l \cdot E_2(l\tau)$, so gilt nach [Mü94] für den gesuchten Wert P_1:

$$P_1 = -\frac{l}{2} E_2^*(\tau).$$

Ausgehend von unserer Kurve $E = (a, b)$ erhalten wir nach Satz 4.14

$$E_4(\tau) = -\frac{a}{3}, \quad E_6(\tau) = -\frac{b}{2}.$$

Damit haben wir unser Problem darauf reduziert, die Werte $E_2^*(\tau), E_4(l\tau)$ und $E_6(l\tau)$ zu bestimmen, d.h. wir müssen den Übergang von τ nach $l\tau$ bewerkstelligen. Wir werden im folgenden die Formeln angeben, die notwendig sind, um einen Algorithmus zur Berechnung von $\tilde{E}$ und P_1 zu implementieren und dabei die theoretischen Hintergründe auslassen. Eine genaue Analyse der Zusammenhänge ist in [Mü94] gegeben.

Wir haben bisher stets von dem l-ten modularen Polynom gesprochen. In der Praxis verwenden wir aus Effizienzgründen nicht das l-te modulare Polynom selbst, sondern zu diesem äquivalente Polynome, d.h. Polynome, die das gleiche Zerfällungsverhalten haben. In Abhängigkeit der Primzahl l benutzen wir zwei verschiedene Arten von Polynomen; genauere Informationen hierüber werden in [Le94] gegeben. Bei der Berechnung von $\tilde{E}$ und P_1 werden wir, unterscheiden von welchem Typ das verwendete Polynom ist.

Wir geben noch eine Definition und zwei Aussagen an, die wir im folgenden benötigen werden. Mit Hilfe der Eisensteinreihen definieren wir die komplex differenzierbare Funktion

$$\begin{aligned} j: \quad H &\longrightarrow \mathbb{C} \\ \omega &\longmapsto \frac{1728 \cdot E_4^3(\omega)}{E_4^3(\omega) - E_6^2(\omega)} \end{aligned}$$

mit $H = \{z \in \mathbb{C} | \ \Im(z) > 0\}$.

Dann gelten die Beziehungen:

$$j(\tau) = \frac{1728 \cdot E_4^3(\tau)}{E_4^3(\tau) - E_6^2(\tau)}, \quad j'(\tau) = \frac{-j(\tau) \cdot E_6(\tau)}{E_4(\tau)}$$

(vgl. [Mü94, Die Situation über den komplexen Zahlen]).

Ist $E_4(\tau)$ bzw. $E_6(\tau)$ gleich Null, so ist auch der Kurvenparameter a bzw. b gleich Null. Dies ist dazu äquivalent, daß die j-Invariante von E den Wert 0 bzw. 1728 annimmt. Der Wert $j(\tau) \in \mathbb{C}$ hat die Eigenschaft, daß er bei Reduktion modulo p gerade die j-Invariante $j(E)$ der Kurve E über $\mathbb{F}_p$ liefert. Da wir durch die Überprüfung der Spezialfälle (vgl. Abschnitt 2.4) die $\mathbb{F}_p$-Isogenie der Kurve E zu einer Kurve mit j-Invariante 0 oder 1728 ausgeschlossen haben, können wir im folgenden annehmen, daß die Werte $E_4(\tau)$, $E_6(\tau)$ und $j(\tau)$ ungleich Null sind.

Berechnung von $\tilde{E}$ und P_1 für Typ 1

Unter einem Polynom vom Typ 1 verstehen wir ein zum modularen Polynom äquivalentes Polynom, welches wir auf folgende Art herleiten.

4.15. Definition *Die Fourierentwicklung der Dedekind'schen η-Funktion auf $\mathbb{C}$ wird gegeben als*

$$\begin{aligned} \eta(\omega) &= q_\omega^{1/24} \cdot \Big(1 + \sum_{n \geq 1} (-1)^n \cdot (q_\omega^{n \cdot (3n-1)/2} + q_\omega^{n \cdot (3n+1)/2})\Big) \\ & \quad \mathit{mit}\ q_\omega = \exp(2\pi i \omega). \end{aligned}$$

Definieren wir die Funktion

$$\begin{aligned} f: \ H &\longrightarrow \mathbb{C} \\ \omega &\longmapsto \left(\frac{\eta(\omega)}{\eta(l\omega)}\right)^{2s}, \end{aligned}$$

wobei $s \in \mathbb{N}$ minimal ist mit $12 \mid s \cdot (l-1)$ und $v = s \cdot (l-1)/12$ sowie die Funktion

$$\begin{aligned} g: \ H &\longrightarrow \mathbb{C} \\ \omega &\longmapsto l^s / f(\omega), \end{aligned}$$

so gelten für das zum modularen Polynom äquivalente Polynom

$$G_l(x,y) = \sum_{r=0}^{l+1} \sum_{k=0}^{v} a_{r,k} \cdot x^r \cdot y^k \in \mathbb{Z}[x,y] \tag{4.10}$$

die beiden Gleichungen

$$G_l\left(g(\tau), j(\tau)\right) = 0, \tag{4.11}$$

$$G_l\left(f(\tau), j(l\tau)\right) = 0. \tag{4.12}$$

Berechnen wir die Reihenentwicklungen für $f(\tau), g(\tau), j(\tau)$ und $j(l\tau)$, so können wir mit Hilfe dieser beiden Gleichungen einen Koeffizientenvergleich durchführen und die Koeffizienten $a_{r,k}$ von $G_l(x,y)$ bestimmen. Diese Herleitung wird in [Le94, Kap. 5] beschrieben. Wie in Abschnitt 2.1 erläutert, erhalten wir aus $G_l(x,y) \in \mathbb{Z}[x,y]$ durch Substitution der j-Invarianten und Reduktion der Koeffizienten modulo p das Polynom für die Kurve E über $\mathbb{F}_p$. Wir werden im folgenden sehen, daß wir zur Bestimmung von $\tilde{E}$ und P_1 explizit die modulo p reduzierten Koeffizienten $a_{r,k}$ von $G_l(x,y)$ benötigen.

Leiten wir $G_l\left(g(\tau), j(\tau)\right)$ nach τ ab, so können wir $g'(\tau)$ berechnen:

$$g'(\tau) = \frac{-j'(\tau) \cdot \sum_{r=0}^{l+1} \sum_{k=0}^{v} k \cdot a_{r,k} \cdot f(\tau)^r \cdot j(\tau)^{k-1}}{\sum_{r=0}^{l+1} \sum_{k=0}^{v} r \cdot a_{r,k} \cdot f(\tau)^{r-1} \cdot j(\tau)^k}.$$

Wir definieren die Größen

$$\begin{aligned}
DF &= g(\tau)\cdot\left(\frac{\partial G_l(x,y)}{\partial x}(g(\tau),j(\tau))\right)\\
DJ &= j(\tau)\cdot\left(\frac{\partial G_l(x,y)}{\partial y}(g(\tau),j(\tau))\right)\\
DFJ &= g(\tau)\cdot j(\tau)\cdot\left(\frac{\partial G_l(x,y)}{\partial x\partial y}(g(\tau),j(\tau))\right)\\
DFF &= \sum_{r=1}^{l+1}\sum_{k=0}^{v} r^2\cdot a_{r,k}\cdot g(\tau)^r\cdot j(\tau)^k\\
DJJ &= \sum_{r=0}^{l+1}\sum_{k=1}^{v} k^2\cdot a_{r,k}\cdot g(\tau)^r\cdot j(\tau)^k.
\end{aligned}$$

Dann gilt nach [Mü94, Abschnitt über Berechnung von $E_2^*(\tau)$]:

$$E_2^*(\tau) = \frac{-12}{s}\cdot\frac{E_6(\tau)\cdot DJ}{E_4(\tau)\cdot DF}.$$

Definieren wir

$$E_0(\tau) := \frac{-s}{12}\cdot\frac{DF}{DJ},$$

so erhalten wir für die Ableitung nach τ :

$$E_0'(\tau) = \frac{-s}{12}\cdot\frac{-DFF\cdot\frac{s}{12}\cdot E_2^*(\tau) - 2DFJ\cdot\frac{E_6(\tau)}{E_4(\tau)} - DJJ\cdot\frac{s}{12}\cdot E_0(\tau)\cdot\frac{E_6(\tau)}{E_4(\tau)}}{DJ}$$

(Beweis s. unten).

Somit gilt nach [Mü94, Abschnitt über Berechnung von $E_4(l\tau)$] :

$$E_4(l\tau) = \frac{E_4(\tau) - E_2^*(\tau)\cdot\left(12\cdot\frac{E_0'(\tau)}{E_0(\tau)} + 6\cdot\frac{E_4(\tau)^2}{E_6(\tau)} - 4\cdot\frac{E_6(\tau)}{E_4(\tau)} - E_2^*(\tau)\right)}{l^2}.$$

Wir haben also einen Berechnungsweg für $E_4(l\tau)$ gefunden. Für die Herleitung von $E_6(l\tau)$ bestimmen wir

$$\begin{aligned}
\Delta(\tau) &= \frac{E_4(\tau)^3}{j(\tau)}\\
\Delta(l\tau) &= \frac{\Delta(\tau)}{f(\tau)^{\frac{12}{s}}}\\
j(l\tau) &= \frac{E_4(l\tau)^3}{\Delta(l\tau)}.
\end{aligned}$$

Leiten wir die Gleichung $G_l(f(\tau), j(l\tau)) = 0$ nach τ ab und formen um, so gilt

$$j'(l\tau) = \frac{\sum_{r=1}^{l+1}\sum_{k=0}^{v} a_{r,k} \cdot r \cdot f(\tau)^{r-1} \cdot f'(\tau) \cdot j(l\tau)^k}{-\sum_{r=0}^{l+1}\sum_{k=1}^{v} a_{r,k} \cdot k \cdot l \cdot f(\tau)^r \cdot j(l\tau)^{k-1}},$$

und wir erhalten $E_6(l\tau)$ als

$$E_6(l\tau) = -\frac{E_4(l\tau) \cdot j'(l\tau)}{j(l\tau)}$$

(vgl. [Mü94, Abschnitt über Berechnung von $E_6(l\tau)$]).

Damit können wir die gesuchten Werte berechnen:

$$P_1 = -\frac{l}{2} \cdot E_2^*(\tau), \quad \tilde{a} = -3 \cdot l^4 \cdot E_4(l\tau), \quad \tilde{b} = -2 \cdot l^6 \cdot E_6(l\tau).$$

Beweis (für $E_0'(\tau)$):

Da wir $E_0(\tau)$ als

$$E_0(\tau) = \frac{-s}{12} \cdot \frac{DF}{DJ}$$

definiert haben, erhalten wir für die Ableitung

$$E_0'(\tau) = \frac{-s}{12}\frac{1}{DJ} \cdot \left(\underbrace{DF' \cdot \frac{DF}{DJ} \cdot DJ'}_{=:B}\right)$$

und

$$\begin{aligned} B \quad = \quad & \frac{g'(\tau)}{g} \cdot DF + \frac{j'(\tau)}{j(\tau)} \cdot DFJ + g(\tau)g'(\tau) \cdot S_1 \\ & - \frac{DF}{DJ}\left(\frac{j'(\tau)}{j(\tau)} \cdot DJ + \frac{g'(\tau)}{g} \cdot DFJ + j(\tau) \cdot j'(\tau) \cdot S_2\right) \end{aligned}$$

mit

$$S_1 = \sum_{r=2}^{l+1}\sum_{k=0}^{v} a_{r,k} r \cdot (r-2) \cdot g(\tau)^{r-2} \cdot j(\tau)^k$$

$$S_2 = \sum_{r=0}^{l+1}\sum_{k=2}^{v} a_{r,k} k \cdot (k-2) \cdot g(\tau)^r \cdot j(\tau)^{k-2}.$$

Wegen

$$\begin{aligned} \frac{g'(\tau)}{g(\tau)} &= \frac{-s}{12} \cdot E_2^*(\tau) = \frac{E_6(\tau)}{E_4(\tau)} \cdot \frac{DJ}{DF} \\ \frac{j'(\tau)}{j(\tau)} &= -\frac{E_6(\tau)}{E_4(\tau)} \end{aligned}$$

(vgl. [Mü94]), erhalten wir

$$\begin{aligned} B &= \frac{E_6(\tau)}{E_4(\tau)} \cdot DJ - \frac{E_6(\tau)}{E_4(\tau)} \cdot DFJ + \frac{E_6(\tau)}{E_4(\tau)} \cdot \frac{DJ}{DF} \cdot g(\tau)^2 \cdot S_1 \\ &\quad + \frac{E_6(\tau)}{E_4(\tau)} \cdot DF - \frac{E_6(\tau)}{E_4(\tau)} \cdot DFJ + \frac{E_6(\tau)}{E_4(\tau)} \cdot \frac{DF}{DJ} \cdot j(\tau)^2 \cdot S_2. \end{aligned}$$

Wie man durch Hinschreiben der Differenzen sieht, gilt

$$g(\tau)^2 \cdot S_1 = DFF - DF$$

$$j(\tau)^2 \cdot S_2 = DJJ - DJ$$

und damit

$$\begin{aligned} B &= -2 \cdot \frac{E_6(\tau)}{E_4(\tau)} \cdot DFJ + DFF \cdot \frac{E_6(\tau)}{E_4(\tau)} \cdot \frac{DJ}{DF} + \frac{E_6(\tau)}{E_4(\tau)} \cdot \frac{DF}{DJ} \cdot DJJ \\ &= -2 \cdot \frac{E_6(\tau)}{E_4(\tau)} \cdot DFJ - \frac{s}{12} \cdot E_2^*(\tau) - \frac{s}{12} \cdot DJJ \cdot E_0(\tau) \cdot \frac{E_6(\tau)}{E_4(\tau)}, \end{aligned}$$

also die gewünschte Form. ∎

Bei der Herleitung von P_1 und $\tilde{E}$ müssen wir an einigen Stellen dividieren. Da das Polynom $G_l(x, j(\tau))$ nur einfache Nullstellen hat, ist DF ungleich 0. Hat das Polynom $G_l(g(\tau), y)$ mehrfache Nullstellen, so ist $DJ = 0$. In diesem Fall können wir $E_4(l\tau)$ und $E_6(l\tau)$ nicht auf die angegebene Weise berechnen, da in den Formeln für $g'(\tau)$, $E_0(\tau)$ und $j'(l\tau)$ die auftretenden Nenner Null sind. Wir wählen deshalb einen anderen Berechnungsweg. Da $DJ = 0$ dazu äquivalent ist, daß

$$E_2^*(\tau) = 0,$$

erhalten wir

$$E_2(\tau) = l \cdot E_2(l\tau), \quad E_2'(\tau) = l^2 \cdot E_2'(l\tau)^2.$$

Wegen

$$E_4(\tau) = E_2(\tau)^2 - 12 \cdot E_2'(\tau)$$

können wir $E_4(l\tau)$ berechnen als

$$E_4(l\tau) = E_2(l\tau)^2 - 12 \cdot E_2'(l\tau) = \frac{E_4(\tau)}{l^2}.$$

Da $f(\tau)$, $j(\tau)$ und $E_4(\tau)$ ungleich Null sind, können wir $\Delta(\tau)$, $\Delta(l\tau)$ und $j(l\tau)$ wie zuvor berechnen und erhalten

$$E_6(l\tau) = \pm\sqrt{(j(l\tau) - 1728) \cdot \Delta(l\tau)}$$

(vgl. [Mü94, Bestimmung von $E_4(l\tau)$ und $E_6(l\tau)$]).

Die zwei möglichen Werte für $E_6(l\tau)$ liefern uns zwei mögliche Kurven für $\tilde{E}$. Um zu entscheiden, welche Kurve die richtige ist, führen wir mit der ersten Kurve den in Abschnitt

4.1.1 beschriebenen Koeffizientenvergleich durch und bestimmen ein Polynom f_C. Anschließend testen wir, ob dieses Polynom ein Teiler des l-ten Divisionspolynoms ist. Teilt f_C das l-te Divisionspolynom nicht, so wiederholen wir das Verfahren mit der zweiten Kurve. Die Situation, daß das Polynom $G_l(g(\tau), y)$ mehrfache Nullstellen hat, ist in der Praxis allerdings nicht aufgetreten. Es ist zur Zeit theoretisch noch nicht geklärt, ob dieser Fall überhaupt eintreten kann.

Beispiel: $E = (3011, 68)$ über $\mathbb{F}_{109}$, $l = 11$.

Das zum 11-ten modularen Polynom äquivalente Polynom aus $\mathbb{F}_{109}[x, y]$ hat die Gestalt

$$\begin{aligned} G_l(x,y) \;=\;& x^{12} + 55x^{11} + 59x^{10} + \\ & x^9 \cdot (92y + 54) + \\ & x^8 \cdot (82y + 86) + \\ & x^7 \cdot \left(25y^2 + y + 36\right) + \\ & x^6 \cdot \left(50y^2 + 62y + 21\right) + \\ & x^5 \cdot \left(104y^3 + 32y^2 + 19y + 74\right) + \\ & x^4 \cdot \left(68y^3 + 51y^2 + 50y + 97\right) + \\ & x^3 \cdot \left(23y^4 + 68y^3 + 50y^2 + 52y + 5\right) + \\ & x^2 \cdot \left(54y^4 + 24y^3 + 65y^2 + 8y + 63\right) + \\ & x \cdot \left(108y^5 + 26y^4 + 14y^3 + 31y^2 + 99y + 6\right) + 93. \end{aligned}$$

Als Nullstelle von $G_l(x, 62)$ finden wir den Wert

$$g(\tau) = 22,$$

und aus den Kurvenparametern berechnen wir

$$E_4(\tau) = 50, \;\; E_6(\tau) = 75, \;\; j(\tau) = 62.$$

Wir erhalten $E_2^*(\tau)$ über den folgenden Berechnungsweg:

$$g'(\tau) = 91,\ DF = 13,\ DJ = 92,\ DFJ = 57,\ DFF = 104,\ DJJ = 106,\ E_2^*(\tau) = 19.$$

Zur Bestimmung von $E_4(l\tau)$ berechnen wir:

$$E_0(\tau) = 89, \;\; E_0'(\tau) = 36, \;\; E_4(l\tau) = 100.$$

Weiterhin erhalten wir

$$\Delta(\tau) = 26, \;\; \Delta(l\tau) = 73, \;\; j(l\tau) = 102, \;\; j'(l\tau) = 83, \;\; E_6(l\tau) = 49$$

und somit

$$P_1 = 100, \;\; \tilde{a} = 73, \;\; \tilde{b} = 42.$$

Berechnung von $\tilde{E}$ und P_1 für Typ 2

Zur Herleitung von Polynomen vom Typ 2 verwenden wir eine auf H komplex differenzierbare Funktion $A : H \longrightarrow \mathbb{C}$, die keinen Bezug zur Dedekind'schen η-Funktion mehr hat. In [Le94, Kap. 6] wird erklärt, wie wir eine Reihenentwicklung für $A(\tau)$ herleiten können. Bezeichnen wir in dieser Reihenentwicklung den kleinsten Exponenten, so daß der zugehörige Koeffizient ungleich 0 ist, mit v, so gelten für das zum modularen Polynom äquivalente Polynom

$$G_l(x,y) = \sum_{r=0}^{l+1} \sum_{s=0}^{2v} a_{r,s} \cdot x^r \cdot y^k \in \mathbb{Z}[x,y] \tag{4.13}$$

die beiden Gleichungen

$$G_l\left(A(\tau), j(\tau)\right) = 0, \tag{4.14}$$
$$G_l\left(A(\tau), j(l\tau)\right) = 0. \tag{4.15}$$

Wie im Typ 1 - Fall werden diese beiden Gleichungen benutzt, um mit einem Koeffizientenvergleich die Koeffizienten $a_{r,s}$ des Polynoms $G_l(x,y)$ zu bestimmen, vgl. [Le94, Kap. 6].

Der Wert $j(l\tau) \in \mathbb{C}$ ist eine Nullstelle des Polynoms $G_l\left(A(\tau), y\right) \in \mathbb{Z}[A(\tau)][y]$. Daher faktorisieren wir dieses Polynom und führen für alle Nullstellen das im folgenden beschriebene Verfahren durch. Mit den so gewonnenen möglichen Werten für $\tilde{E}$ und P_1 berechnen wir, mit dem im Abschnitt 4.1.1 beschriebenen Koeffizientenvergleich, ein Polynom. Ist dieses ein Teiler des l-ten Divisonspolynoms, so war die verwendete Nullstelle wirklich $j(l\tau)$; im anderen Fall müssen wir das gesamte Verfahren mit der nächsten Nullstelle wiederholen.

Seien also $A(\tau) \in \mathbb{C}$ und $j(l\tau) \in \mathbb{C}$ gegeben.

Wir definieren die folgenden Hilfsgrößen:

$$\begin{aligned}
DF_1 &:= \left(\frac{\partial}{\partial X} G_l(x,y)\right)\left(A(\tau), j(\tau)\right) \\
DJ_1 &:= \left(\frac{\partial}{\partial Y} G_l(x,y)\right)\left(A(\tau), j(\tau)\right) \\
DF_2 &:= \left(\frac{\partial}{\partial X} G_l(x,y)\right)\left(A(\tau), j(l\tau)\right) \\
DJ_2 &:= \left(\frac{\partial}{\partial Y} G_l(x,y)\right)\left(A(\tau), j(l\tau)\right).
\end{aligned}$$

Leiten wir die Gleichung (4.14) nach τ ab und formen um, so erhalten wir

$$\begin{aligned}
A'(\tau) &= \frac{-j'(\tau) \cdot DJ_1}{DF_1} \\
&= \frac{-j'(\tau) \cdot \sum_{r=0}^{l+1} \sum_{s=1}^{2v} k \cdot a_{r,s} \cdot A(\tau)^r \cdot j(\tau)^{k-1}}{\sum_{r=1}^{l+1} \sum_{s=0}^{2v} r \cdot a_{r,s} \cdot A(\tau)^{r-1} \cdot j(\tau)^k}.
\end{aligned}$$

Leiten wir die Gleichung (4.15) nach τ ab und formen um, so erhalten wir

$$
\begin{aligned}
j'(l\tau) &= \frac{-A'(\tau)\cdot DF_2}{l\cdot DJ_2} \\
&= \frac{-A'(\tau)\cdot\sum_{r=1}^{l+1}\sum_{s=0}^{2v} r\cdot a_{r,s}\cdot A(\tau)^{r-1}\cdot j(l\tau)^k}{l\cdot\sum_{r=0}^{l+1}\sum_{s=1}^{2v} k\cdot a_{r,s}\cdot A(\tau)^r\cdot j(l\tau)^{k-1}}.
\end{aligned}
$$

Mit diesen Größen berechnen wir

$$
\begin{aligned}
E_4(l\tau) &= \frac{j'(l\tau)^2}{j(l\tau)\cdot(j(l\tau)-1728)} \\
E_6(l\tau) &= \frac{-E_4(l\tau)\cdot j'(l\tau)}{j(l\tau)}
\end{aligned}
$$

(vgl. [Mü94, Abschnitt über Berechnung von $E_4(l\tau)$ und $E_6(l\tau)$]).

Zur Berechnung von $E_2^*(\tau)$ definieren wir für $(x_1,y_1)=(A(\tau),j(\tau))$ und $(x_2,y_2)=(A(\tau),j(l\tau))$ $(i=1,2)$ folgende Werte:

$$
\begin{aligned}
DFF_i &:= \left(\frac{\partial}{\partial X\partial X}G_l(X,Y)\right)(x_i,y_i) \\
DJJ_i &:= \left(\frac{\partial}{\partial Y\partial Y}G_l(X,Y)\right)(x_i,y_i) \\
DFJ_i &:= \left(\frac{\partial}{\partial X\partial Y}G_l(X,Y)\right)(x_i,y_i).
\end{aligned}
$$

Berechnen wir

$$
\begin{aligned}
W_1 &:= \frac{-1}{A'(\tau)\cdot DF_1}\cdot\left(A'(\tau)^2\cdot DFF_1+2\cdot A'(\tau)\cdot j'(\tau)\cdot DFJ_1+j'(\tau)^2\cdot DJJ_1\right) \\
&= \frac{-1}{A'(\tau)\cdot\sum_{r=1}^{l+1}\sum_{s=0}^{2v} r\cdot a_{r,s}\cdot A(\tau)^{r-1}\cdot j(\tau)^k}\cdot \\
&\quad \left(A'(\tau)^2\cdot\sum_{r=2}^{l+1}\sum_{s=0}^{2v} r\cdot(r-1)\cdot a_{r,s}\cdot A(\tau)^{r-2}\cdot j(\tau)^k\right. \\
&\quad +2\cdot A'(\tau)\cdot j'(\tau)\cdot\sum_{r=1}^{l+1}\sum_{s=1}^{2v} r\cdot k\cdot a_{r,s}\cdot A(\tau)^{r-1}\cdot j(\tau)^{k-1} \\
&\quad \left.+j'(\tau)^2\cdot\sum_{r=0}^{l+1}\sum_{s=2}^{2v} k\cdot(k-1)\cdot a_{r,s}A(\tau)^r\cdot j(\tau)^{k-2}\right) \\
W_2 &:= \frac{-1}{A'(\tau)\cdot DF_2}\cdot\left(A'(\tau)^2\cdot DFF_2+2\cdot l\cdot A'(\tau)\cdot j'(\tau)\cdot DFJ_2+l^2\cdot j'(\tau)^2\cdot DJJ_2\right) \\
&= \frac{-1}{A'(\tau)\cdot\sum_{r=1}^{l+1}\sum_{s=0}^{2v} r\cdot a_{r,s}\cdot A(\tau)^{r-1}\cdot j(l\tau)^k}\cdot
\end{aligned}
$$

$$\left(A'(\tau)^2 \cdot \sum_{r=2}^{l+1}\sum_{s=0}^{2v} r\cdot(r-1)\cdot a_{r,s}\cdot A(\tau)^{r-2}\cdot j(l\tau)^k\right.$$
$$+2\cdot l\cdot A'(\tau)\cdot j'(l\tau)\cdot \sum_{r=1}^{l+1}\sum_{s=1}^{2v} r\cdot k\cdot a_{r,s}\cdot A(\tau)^{r-1}\cdot j(l\tau)^{k-1}$$
$$\left.+l^2\cdot j'(l\tau)^2\cdot \sum_{r=0}^{l+1}\sum_{s=2}^{2v} k\cdot(k-1)\cdot a_{r,s}A(\tau)^r\cdot j(l\tau)^{k-2}\right),$$

so gilt nach [Mü94, Abschnitt über Berechnung von $E_2^*(\tau)$]

$$\begin{aligned} E_2^*(\tau) &= 6\cdot(W_2 - W_1)\\ &+ 3\cdot\left(\frac{l\cdot j(l\tau)\cdot E_4(l\tau)}{j'(l\tau)} - \frac{j(\tau)\cdot E_4(\tau)}{j'(\tau)}\right)\\ &+ 4\cdot\left(\frac{l\cdot E_4(l\tau)\cdot E_6(l\tau)^2}{\Delta(l\tau)\cdot j'(l\tau)} - \frac{E_4(\tau)\cdot E_6(\tau)^2}{\Delta(\tau)\cdot j'(\tau)}\right), \end{aligned}$$

wobei wie im vorherigen Abschnitt

$$\begin{aligned} \Delta(\tau) &= \frac{E_4(\tau)^3}{j(\tau)}\\ \Delta(l\tau) &= \frac{E_4(l\tau)^3}{j(l\tau)}. \end{aligned}$$

Damit erhalten wir auch hier

$$P_1 = -\frac{l}{2}\cdot E_2^*(\tau),\ \ \tilde{a} = -3\cdot l^4\cdot E_4(l\tau),\ \ \tilde{b} = -2\cdot l^6\cdot E_6(l\tau).$$

Dieses Verfahren für Typ 2 - Polynome funktioniert, falls das Polynom $G_l(A(\tau), y)$ keine mehrfachen Nullstellen hat. Sollte dies der Fall sein, d.h. DJ_i $(i = 1, 2)$ Null sein, so ist zur Zeit nicht bekannt, wie P_1 und $\tilde{E}$ bestimmt werden können. Alle experimentellen Versuche deuten jedoch daraufhin, daß diese Situation nicht auftritt.

Zum Abschluß dieses Abschnittes geben wir noch einige Laufzeiten für die Berechnung von $\tilde{E}$ und P_1 an. In der ersten Spalte ist jeweils die Anzahl der Dezimalstellen der verwendeten Charakteristik angegeben. Unter der Rubrik „Typ“ ist aufgeführt, ob wir für den Modul l ein Polynom vom Typ 1 oder Typ 2 verwendet haben. Die Dauer der Berechnung von $\tilde{E}$ und P_1 ist unter anderem von der Anzahl der Koeffizienten des zum modularen Polynom äquivalenten Polynoms abhängig. Wie wir gesehen haben, existieren für Typ 1 - Polynome $(l+2)\cdot v$ und für Typ 2 - Polynome $(l+2)\cdot 2v$ viele Koeffizienten; der Wert v ergibt sich dabei aus der Herleitung des Polynoms. Unter „$\#j(l\tau)$“ haben wir angegeben, wieviele mögliche Werte für $j(l\tau)$ im Fall von Typ 2 - Moduln existierten; in Klammern ist hinzugefügt, wieviele $j(l\tau)$ getestet wurden, bis ein Teiler f_C des l-ten Divisionspolynoms gefunden wurde. Die Zeitangabe in der letzten Spalte beinhaltet die Berechung der in $\mathbb{F}_p$ liegenden Nullstellen des Polynoms $G_l(A(\tau), y) \in \mathbb{F}_p[y]$ zur Bestimmung der möglichen $j(l\tau)$-Werte; dabei sollten wir beachten, daß dieses Polynom den Grad $2v$ besitzt.

$dd(p)$	l	v	Typ	$\#j(l\tau)$	Zeit
100	53	13	1		2 sec
	97	8	1		2 sec
	157	13	1		6 sec
	199	33	1		25 sec
150	103	3	2	3 (3)	42 sec
	211	35	1		47 sec
	317	11	2	1 (1)	4 min 55 sec
200	103	3	2	1 (1)	44 sec
	193	16	1		22 sec
	307	10	2	1 (1)	8 min 09 sec
	401	10	2	1 (1)	8 min 52 sec
250	103	3	2	5 (5)	4 min 32 sec
	199	33	1		1 min 17 sec
	313	12	2	1 (1)	18 min 27 sec
300	101	25	1		33 sec
	317	11	2	3 (3)	32 min 53 sec
	421	14	2	1 (1)	42 min 06 sec
	523	15	2	1 (1)	50 min 15 sec
375	103	3	2	1 (1)	3 min 28 sec
	223	37	1		3 min 19 sec
	401	10	2	1 (1)	33 min 46 sec
	503	10	2	1 (1)	38 min 31 sec
	839	14	2	1 (1)	1 h 16 min

Wir sehen, daß die Berechnungen für Typ 2 aufwendiger sind als für Typ 1. Dies liegt vor allem an der Berechnung der Nullstellen des Polynoms $G_l(A(\tau), y)$; außerdem muß in einigen Fällen die Berechnung von $\tilde{E}$ und P_1 mehrmals durchgeführt werden. Dennoch benutzen wir für größere Moduln ($l > 250$) fast ausschließlich Typ 2 - Polynome, da die Berechnung von Typ 1 - Polynomen für große Moduln praktisch nicht mehr durchführbar ist (vgl. [Le94, Abschnitt 6.3]).

4.2 Berechnung von formalen Punktvielfachen

Wir gehen in diesem Abschnitt davon aus, daß ein Teiler $f_C(x)$ des l-ten Divisionspolynoms berechnet wurde, der zum gesuchten Eigenwert $\alpha \in \mathbb{F}_l^*$ gehörende Eigenvektoren charakterisiert. Wir erstellen eine Liste von möglichen Eigenwerten $\alpha_1, \ldots, \alpha_n$ und berechnen die formalen Punktvielfachen $\alpha_i \cdot (x, y) \bmod f_C$. Um den Eigenwert zu bestimmen, vergleichen wir diese Punkte mit den Werten des Frobeniusendomorphismus $\Phi_E\,(x, y) = (x^p, y^p) \bmod f_C$.

Zunächst schränken wir die möglichen Eigenwerte α, für die Punktvielfache $\alpha \cdot (x, y) \bmod f_C$ berechnet werden müssen, ein. Wir wissen aus dem ersten Teil des Algorithmus (vgl. Satz 2.3), ob der Frobeniusendomorphismus Φ_E eingeschränkt auf die l-Torsionsgruppe $E[l]$ nur den Eigenwert α oder noch einen zweiten Eigenwert $\beta \neq \alpha$, $\beta \in \mathbb{F}_l^*$, hat.

Existiert nur ein Eigenwert, so folgt aus der charakteristischen Gleichung des Frobeniusendomorphismus, daß $\alpha^2 \equiv p$ modulo l. Es existieren somit 2 Kandidaten für den Eigenwert und wir müssen ein Punktvielfaches berechnen.

Existieren zwei Eigenwerte $\alpha \neq \beta$, so kennen wir aus dem ersten Teil des Algorithmus die Ordnung d von $\frac{\alpha}{\beta} \equiv_l \frac{\alpha^2}{p}$. Wir bestimmen alle $\varphi(d)$ Elemente der Ordnung d in $\mathbb{F}_l^*$ $(\gamma_1, \ldots, \gamma_{\varphi(d)})$ und erhalten durch die Gleichung

$$\alpha = \pm\sqrt{p \cdot \gamma_i},\ \ 1 \leq i \leq \varphi(d),$$

maximal $2 \cdot \varphi(d)$ viele Kandidaten für den Eigenwert α. Dabei muß nicht jedes Element $p \cdot \gamma_i$ ein Quadrat in $\mathbb{F}_l$ sein.

In dieser Liste ersetzen wir die Kandidaten α mit $\alpha > \frac{l-1}{2}$ durch ihr additives Inverses $l-\alpha$. Haben wir $(x_1, y_1) = (l - \alpha) \cdot (x, y) \bmod f_C$ berechnet, so überprüfen wir $\alpha \cdot (x, y)$ durch den Vergleich $(x_1, -y_1) = (x^p, y^p)$. Durch diese Modifikation müssen wir nur Vielfache kleiner gleich $\frac{l-1}{2}$ berechnen.

Wir gehen im folgenden davon aus, daß eine Liste

$$\alpha_1, \ldots, \alpha_n \text{ mit } |\alpha_i| \leq \frac{l-1}{2},\ \ 1 \leq i \leq n,$$

von möglichen Eigenwerten gegeben ist, und für einen Kandidaten α bestimmen wir

$$\alpha \cdot (x, y) = (h_1(x), y \cdot h_2(x)),\ \ h_1, h_2 \in \mathbb{F}_p(x),$$

und überprüfen, ob

$$\begin{aligned} x^p &\equiv h_1(x) \text{ modulo } f_C \\ \left(x^3 + ax + b\right)^{\frac{p-1}{2}} &\equiv h_2(x) \text{ modulo } f_C \\ \left(x^3 + ax + b\right)^{\frac{p-1}{2}} &\equiv -h_2(x) \text{ modulo } f_C. \end{aligned}$$

Da die Berechnung von $x^p \bmod f_C$ bzw. $y^p \bmod f_C$ teuer ist, verwenden wir folgende Heuristik:

Stimmt bei den formalen Punkten $\Phi_E(x, y) \bmod f_C$ und $\alpha \cdot (x, y) \bmod f_C$, $\alpha \in \mathbb{F}_l^*$, die y-Koordinate überein, so ist α der gesuchte Eigenwert.

Aufgrund dieser Heuristik vergleichen wir bei der Eigenwertsuche nur die y-Koordinaten und sparen die Berechnung von $x^p \bmod f_C$. Es ist theoretisch möglich, daß wir durch diese Verfahrensweise einen falschen Eigenwert und damit die Spur c modulo l falsch bestimmen. Dann finden wir entweder in der anschließenden Babystep-Giantstep-Phase keine Übereinstimmung, oder wir erhalten eine falsche Gruppenordnung, die dann durch das Beweisverfahren aufgedeckt wird. Bei unseren experimentellen Untersuchungen ist dieser Fall jedoch nie eingetreten; dies legt nahe, daß es sinnvoll ist, diese Heuristik zu verwenden.

Für die Berechnung der Punktvielfachen haben wir in unserer Implementierung zwei Ansätze verfolgt:

- Rechnen mit Divisionspolynomen.
- Rechnen mit rationalen Funktionen unter Benutzung der Punktadditionsformeln.

Je nach zur Verfügung stehendem Speicherplatz und Anzahl der zu testenden Punktvielfachen wählen wir eine der beiden Methoden. Wir werden beide Vorgehensweisen im folgenden beschreiben.

4.2.1 Berechnung von Punktvielfachen mit Divisionspolynomen

Wir haben in Kapitel 1 die zur Kurve $E/\mathbb{F}_p$ korrespondierenden Divisionspolynome bereits definiert und gesehen, daß die Nullstellen des n-ten Divisionspolynoms genau die x-Koordinaten der Punkte in der n-Torsionsgruppe sind. Der nachfolgende Satz liefert uns eine Berechnungsvorschrift, um mit Hilfe der Divisionspolynome Punktvielfache zu bestimmen.

4.16. Satz *Sei $n \geq 1$ und $P = (x, y) \in E(\overline{\mathbb{F}_p})$ ein Punkt der elliptischen Kurve E mit $n \cdot P \neq \mathcal{O}$. Dann gilt*

$$n \cdot P = \left(x - \frac{\psi_{n-1}\psi_{n+1}}{\psi_n^2}, \frac{\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2}{4y\psi_n^3} \right).$$

Dabei fasse man das k-te Divisionspolynom ψ_k als $\psi_k(x, y)$ auf.

Beweis: [La78]. ■

4.17. Bemerkung In unserer Situation können wir den Satz anwenden, da wir mit l-Torsionspunkten rechnen, und die zu bestimmenden Vielfachen kleiner als l sind.

Um die möglichen Eigenwerte $\alpha_1, \ldots, \alpha_n$ zu testen, berechnen wir sukzessive alle Divisionspolynome $\psi_1, \ldots, \psi_{\alpha_n+2}$ und damit die entsprechenden Punktvielfachen mittels Satz 4.16. Dabei wird in unserer Implementierung das n-te Divisionspolynom

$$\psi_n(x, y) \in \begin{cases} y \cdot \mathbb{F}_p[x, y^2], & \text{falls } n \text{ gerade}, \\ \mathbb{F}_p[x, y^2], & \text{falls } n \text{ ungerade}, \end{cases}$$

durch das Polynom in x und den Index n repräsentiert. Bei der Verknüpfung von Divisionspolynomen müssen wir dann Fallunterscheidungen durchführen und gegebenenfalls y^2 Terme durch $x^3 + ax + b$ ersetzen. Wir erhalten im ersten Ansatz den folgenden Algorithmus:

<u>Eigenwertbestimmung mit Divisionspolynomen</u>

EINGABE: $f_C \in \mathbb{F}_p[X]$, Liste $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_l^*$
AUSGABE: Eigenwert α des Frobeniusendomorphismus $\Phi_{E_{|E[l]}}$

(1) Berechne $y^{p-1} \equiv (x^3 + ax + b)^{(p-1)/2} \bmod f_C$.
(2) $j = 1$.
(3) **for** $(m = 1; m \leq \alpha_n; m++)$ **do**
(4)   **if** $(m == \alpha_j)$ **then**
(5)   Berechne $\psi_{m+2}, \psi_{m-1}^2, \psi_{m-2}, \psi_{m+1}^2$ und ψ_m^3 (mod f_C).
(6)   Berechne den Zähler der y-Koordinate von $\alpha_j \cdot (x, y)$:
        $g(x) \equiv_{f_C} (\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2)$.
(7)   Multipliziere den Nenner der y-Koord. mit y^{p-1}:
        $h(x) \equiv_{f_C} 4 \cdot \psi_m^3 \cdot y^{p-1}$.
(8)   **if** (m gerade) **then**
(9) $h(x) \equiv_{f_C} h(x) \cdot y^4$.
(10)   **fi**
(11)   **if** ($g(x) \equiv_{f_C} h(x)$) **then**
(12)   **return** (α_j)
(13)   **else**
(14)   **if** ($g(x) \equiv_{f_C} -h(x)$) **then**
(15)   **return** $(l - \alpha_j)$
(16)   **fi**
(17)   **fi**
(18) $j = j + 1$.
(19)   **fi**
(20) **od**

Die rekursive Definition der Divisionspolynome bewirkt, daß die Berechnung der Polynome in Schritt (5) aufwendig wird. Zum einen benötigen wir zur Berechnung von

$$n \cdot (x, y) \text{ die Polynome mit Index } \quad n-1, n, n+1, n+2,$$

zum anderen zur Berechnung von

$$\psi_m \text{ die Polynome mit Index } \begin{cases} n-2, n-1, n, n+1, n+2, & \text{für } m = 2n \\ n-1, n, n+1, n+2, & \text{für } m = 2n+1 \end{cases}$$

sowie jeweils die zweite und dritte Potenz dieser Polynome. Um das Verfahren effizient zu machen, speichern wir uns Divisionspolynome, die wir einmal berechnet haben, in einer Tabelle. Da wir sukzessive alle Polynome $\psi_1, \ldots, \psi_{\alpha_n+2}$ berechnen, benötigen wir viel Platz. Um den Platzbedarf zu verringern, speichern wir nicht alle $\alpha_n + 2$ Polynome mit ihren Potenzen. Dazu analysieren wir die rekursive Formel zur Berechnung der Divisionspolynome.

Welches Polynom wird nach der Berechnung von $\psi_m, m \geq 1$, nicht mehr benötigt ?

- m ungerade:
  Das Polynom mit kleinsten Index, daß bei der Berechnung von ψ_m benötigt wurde, ist $\psi_{\frac{m-1}{2}-1}$. Da $m+1$ gerade ist, brauchen wir zur Berechnung von ψ_{m+1} das Polynom mit Index $\frac{m+1}{2} - 2 = \frac{m-1}{2} - 1$. Also dürfen wir dieses Polynom nicht streichen.
- m gerade:
  Hier ist der kleinste Index $\frac{m}{2} - 2$. Zur Berechnung von ψ_{m+1} benötigen wir nur noch ψ mit Index $\frac{m+1-1}{2} - 1 = \frac{m}{2} - 1$. Also können wir nach der Berechnung von ψ_m das Polynom $\psi_{\frac{m}{2}-2}$ aus unserer Liste herausstreichen.

Wir erhalten also in Schritt m ein Polynom ψ_m hinzu und können, im Fall m gerade, das Polynom $\psi_{\frac{m}{2}-2}$ streichen. Da das höchste zu berechnende Divisionspolynom den Index $\alpha_n + 2$ hat, wobei $\alpha_n \leq \frac{l}{2}$ ist, und die Polynome modulo f_C vom Grad $\frac{l-1}{2}$ reduziert sind, können wir die Anzahl der Polynomkoeffizienten, die in diesem Verfahren gespeichert werden müssen, größenordnungsmäßig als

$$\frac{l}{4} \cdot \frac{l}{2} \cdot 3 = \frac{3}{8} l^2 \tag{4.16}$$

angeben; Faktor 3, da wir von jedem Polynom auch die zweite und dritte Potenz benötigen. Damit erhalten wir den folgenden, verfeinerten Algorithmus zur Berechnung von Punktvielfachen mittels Divisionspolynomen:

Eigenwertbestimmung mit Divisionspolynomen

EINGABE: $f_C \in \mathbb{F}_p[X]$, Liste $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_l^*$
AUSGABE: Eigenwert α des Frobeniusendomorphismus $\Phi_{E_{|E[l]}}$

(1) Berechne $y^{p-1} \equiv (x^3 + ax + b)^{(p-1)/2} \bmod f_C$.
(2) Definiere ein dreidimensionales Feld D
von Divisionspolynomen der Größe $(\alpha_n + 5) \times 3$.
(3) Berechne $D[i][j] = \psi_i^j \bmod f_C$, $i = -1, 0, 1, 2$, $j = 1, 2, 3$.
(4) $i = 1$.
(5) **for** $(m = 1; m \leq \alpha_n; m++)$ **do**
(6) Berechne $D[m+2][j] = \psi_{m+2}^j \bmod f_C$ $(j = 1, 2, 3)$
unter Benutzung der in D gespeicherten Polynome.
(7) **if** $(m == \alpha_i)$ **then**
(8) Berechne den Zähler der y-Koordinate von $m \cdot (x, y)$:
$g(x) \equiv_{f_C} (D[m+2][1] \cdot D[m-1][2] - D[m-2] \cdot D[m+1][2])$
(9) Multipliziere den Nenner der y-Koord. mit y^{p-1}:
$h(x) \equiv_{f_C} 4 \cdot D[m][3] \cdot y^{p-1}$.
(10) **if** (m gerade) **then**
(11) $h(x) \equiv_{f_C} h(x) \cdot y^4$.
(12) **fi**
(13) **if** $(g(x) \equiv_{f_C} h(x))$ **then**
(14) **return** (α_i)
(15) **else**
(16) **if** $(g(x) \equiv_{f_C} -h(x))$ **then**
(17) **return** $(l - \alpha_i)$
(18) **fi**
(19) **fi**
(20) $i = i + 1$.
(21) **fi**
(22) **if** ($m + 2$ ist gerade) **then**
(23) Streiche die Polynome $D[(m/2) - 1][j]$ $(j = 1, 2, 3)$ in D.
(24) **fi**
(25) **od**

Die Laufzeit dieses Algorithmus wird durch die Anzahl der durchgeführten Polynomoperationen bestimmt, dabei ist die Multiplikation der dominierende Anteil.

Zur Bestimmung des m-ten Divisionspolynoms benötigen wir

$$\begin{array}{rl} \text{3 Mult.,} & \text{falls } m = 2n \\ \text{2 Mult. } + \text{1 Mult. mit } y^4, & \text{falls } m = 2n+1 \end{array}$$

und 2 Multiplikationen für die zweite und dritte Potenz, insgesamt 5 Multiplikationen.

Da wir den Punktvergleich nur mit der y-Koordinate durchführen, erhalten wir für die Berechnung von $n \cdot P$

	n ungerade	n gerade
Zähler :	2 Mult.	2 Mult.
Nenner :	0 Mult.	1 Mult. mit y^4

sowie 1 Multiplikation für den Vergleich mit y^{p-1}, insgesamt 4 Multiplikationen.

Testen wir $\alpha_1, \ldots, \alpha_n$, so erhalten wir als Anzahl der durchzuführenden Polynommultiplikationen

$$5 \cdot \alpha_n + 4 \cdot n. \tag{4.17}$$

<u>Beispiel:</u> $E = (3011, 68)$ über $\mathbb{F}_{109}$, $l = 11$.

Wir erhalten als Teiler des 11-ten Divisionspolynoms

$$f_C(x) = x^5 + 9x^4 + 70x^3 + 97x^2 + 87x + 14,$$

und die zu testenden Kandidaten für den Eigenwert sind 2, 3, 4 und 5. Im folgenden geben wir für die Divisionspolynome nur die Polynome in x an. Wir beginnen mit

$$\psi_0(x) = 0, \;\; \psi_2(x) = 2, \;\; \psi_2(x) = 2.$$

Im ersten Schleifendurchlauf ($m = 1$) berechnen wir das 3-te Divisionspolynom

$$\psi_3(x) = 3x^4 + 81x^2 + 53x63, \;\; \psi_3^2(x) = 28x^4 + 40x^2 + 93x + 38,$$

und da der erste zu testende Eigenwert zwei ist, müssen wir keinen Vergleich durchführen. Für $m = 2$ berechnen wir

$$\psi_4(x) = 96x^4 + 51x^3 + 44x^2 + 51x + 36, \;\; \psi_4^2(x) = 98x^4 + 22x^3 + 5x^2 + 36x + 92$$

sowie $\psi_2^3(x) = 8$. Als Zähler der y-Koordinate von $2 \cdot (x, y)$ erhalten wir $g(x) = \psi_4(x)$; multiplizieren wir den Nenner der y-Koordinate mit y^p, so ergibt sich das Polynom in x als

$$h(x) = 45x^4 + 3x^3 + 4x^2 + 26x + 9.$$

Da die beiden Polynome weder gleich noch additiv invers zueinander sind, ist ± 2 nicht der gesuchte Eigenwert. Wir können die beiden Polynome ψ_0 und ψ_0^2 aus unserer Liste streichen. Im nächsten Schleifendurchlauf ($m = 3$) bestimmen wir das 5-te Divisionspolynom

$$\psi_5(x) = 21x^4 + 13x^3 + 77x^2 + 25x + 43, \;\; \psi_5^2(x) = 97x^4 + 91x^3 + 65x^2 + 37x + 63.$$

sowie die dritte Potenz $\psi_3^3(x)$. Wir erhalten als Zähler der y-Koordinate von $3 \cdot (x, y)$ das Polynom in x

$$g(x) = 95x^4 + 30x^3 + 85x^2 + 64x + 80.$$

Das Produkt von Nenner und y^p ergibt sich als

$$h(x) = 95x^4 + 30x^3 + 85x^2 + 64x + 80,$$

und damit ist der gesuchte Eigenwert 3. Also ist

$$c \equiv 10 \bmod 11.$$

4.2.2 Berechnung von Punktvielfachen mit rationalen Funktionen

Außer der Berechnung von formalen Punktvielfachen mit Divisionspolynomen haben wir den Ansatz der Bestimmung von Punktvielfachen mittels rationaler Funktionen verfolgt. Dieses Verfahren basiert auf der Idee, den formalen Punkt $n \cdot (x, y)$, $n \in \mathbb{N}$, mit den Additionsgesetzen aus Satz 1.3 zu bestimmen.

Wir berechnen das n-fache von formalen Punkten, indem wir mit dem Tupel

$$\left(\frac{x}{1}, y \cdot \frac{1}{1}\right)$$

von rationalen Funktionen starten und darauf die Additionsformeln für Punkte einer elliptischen Kurve anwenden (vgl. Satz 1.3). Anhand der Additionsformeln und mit Induktion sieht man, daß ein solches formales Punktvielfaches immer von der Form

$$n \cdot (x, y) = (h_1(x), y \cdot h_2(x)) \text{ mit } h_1, h_2 \in \mathbb{F}_p(x) \tag{4.18}$$

ist; dabei ersetze man stets die y^2-Terme durch $x^3 + ax + b$. Deshalb wird das formale Punktvielfache $n \cdot (x, y)$ durch das Tupel (h_1, h_2) repräsentiert.

Beispiel:
Rechnen wir mit der Kurve $(3011, 1968)$ über $\mathbb{F}_{109}$ und verwenden als Modul das Polynom $f(x) = x^5 + 9x^4 + 13x^3 + 16x^3 + 47x + 23$, so erhalten wir

$$5 \cdot (x, y) \equiv_f \left(\frac{65x^4 + 34x^3 + 46x^2 + 84x + 77}{17x^4 + 5x^3 + 33x^2 + 72x + 75}, y \cdot \frac{57x^4 + 36x^3 + 8x^2 + 60x + 41}{65x^4 + 71x^3 + 78x^2 + 5x + 23}\right).$$

Bei der Addition zweier solcher Tupel müssen wir darauf achten, welches Vielfache durch ein Tupel repräsentiert wird; gleiche Vielfache müssen wir mit der Formel für „$2 \cdot P$“, verschiedene Vielfache mit der Formel für „$P + Q$“ verknüpfen (vgl. Satz 1.3). Die Entscheidung, ob es sich um gleiche oder verschiedene Vielfache handelt, hängt stets von dem Punkt ab, mit dem man die rationalen Funktionen spezialisiert. Da wir alle Polynome modulo dem Polynom f_C reduzieren, also mit l-Torsionspunkten rechnen, die alle die gleiche Ordnung l haben, repräsentiert in unserer Situation ein Tupel (h_1, h_2) für alle Punkte das gleiche Vielfache. Außerdem berechnen wir nur Vielfache kleiner l, so daß wir die Punktordnungen nicht überschreiten.

Wir haben drei verschiedene Strategien zur Berechnung von Punktvielfachen implementiert, die je nach Struktur der zu berechnenden Vielfachen $\alpha_1, \ldots, \alpha_n$ (die Kandidaten für den Eigenwert) unterschiedlich gut geeignet sind.

1. Alle α_i, $i = 1, \ldots, n$ sind kongruent modulo einer Zahl $\Delta \in \mathbb{N}$.

   Wir berechnen mittels paralleler Schneller Exponentiation $\alpha_1 \cdot (x, y) \bmod f_C$ und $\Delta \cdot (x, y) \bmod f_C$ und erhalten

   $$\alpha_i \cdot (x, y) = (\alpha_1 + (i-1) \cdot \Delta) \cdot (x, y) \bmod f_C, \;\; i = 2, \ldots, n.$$

2. Die möglichen Eigenwerte liegen dicht, d.h. $\alpha_n \leq 2n$.

   Wir berechnen die Punktvielfachen $m \cdot (x, y) \bmod f_C$, $1 \leq m \leq \alpha_n$, sukzessive nur durch Addition mit (x, y). Die unten angegebene Analyse zeigt, daß eine solche Punktaddition schneller durchzuführen ist, als eine Addition beliebiger Punkte.

3. Wir benutzen eine Babystep-Giantstep-Strategie.

   Dazu berechnen wir für $m = \lceil \sqrt{\alpha_n} \rceil$ die Babystep-Punkte

   $$i \cdot (x, y) \bmod f_C, \;\; i = 1, \ldots, m-1.$$

   Anschließend bestimmen wir sukzessive die Giantsteps

   $$i \cdot m \cdot (x, y) \bmod f_C, \;\; i = 1, \ldots, \frac{n}{m}.$$

   Dabei suchen wir **nicht** den jeweiligen Giantstep G in der Babystep-Tabelle, sondern wir berechnen mit Hilfe der Tabelle die möglichen Vielfachen $\alpha_j \cdot (x, y)$, $1 \leq j \leq n$, die mit diesem Giantstep „erreichbar“ sind, d.h. alle Vielfachen, die sich in der Form G + Babystep-Punkt darstellen lassen. Wir können keine Suche durchführen, da wir dazu den kompletten Punkt $(x^p, y^p) \bmod f_C$ benötigten; von diesem kennen wir aber nur die y-Koordinate. Wir haben uns eine Strategie überlegt, um einen vollständigen Babytstep-Giantstep-Algorithmus zu verwenden; diese ist allerdings noch nicht implementiert (s. unten).

Ein Vorteil des Rechnens mit rationalen Funktionen ist der geringe Speicherplatzbedarf. In den Fällen 1 und 2 benötigt man eine konstante Anzahl von Tupeln von rationalen Funktionen. Die Babystep-Giantstep-Strategie hat den bekannten Platzverbrauch von $\sqrt{\alpha_n}$ vielen Tupeln.

Beispiel:
Für $l \approx 800$ und einer Größe von 200 Bytes für p müssen wir höchstens $20 \cdot 4 = 80$ Polynome vom Grad 400 verwalten, was einen Speicherplatzverbrauch von 6,4 MB zur Folge hat. Dies ist im Vergleich zum Speicherplatzbedarf der Divisionspolynome (48 MB) wenig.

Wir analysieren nun die Polynomoperationen, die für die Addition zweier Punktvielfachen notwendig sind, wie wir dies schon bei der Berechnung von Punktvielfachen mittels Divisionspolynomen getan haben. Seien dazu

$$\begin{aligned} P &:= \left(\tfrac{a_1}{b_1}, y \cdot \tfrac{c_1}{d_1}\right) \\ & \qquad\qquad\qquad \text{mit } a_i, b_i, c_i, d_i \in \mathbb{F}_p[x]/(f_C(x)), \;\; i = 1, 2, \\ Q &:= \left(\tfrac{a_2}{b_2}, y \cdot \tfrac{c_2}{d_2}\right) \end{aligned}$$

so erhalten wir mit den Notationen von Satz 1.3

- Berechnung von „$2 \cdot P$“

$$\begin{aligned}
\lambda &= \frac{(3 \cdot a_1^2 + a \cdot b_1^2) \cdot d_1}{2 \cdot b_1^2 \cdot c_1} \quad (2\times \text{ Quadrieren, } 2\times \text{ Multiplizieren}),\\
x_3 &= -\frac{2 \cdot a_1 \cdot b_1}{b_1^2} + \lambda^2 \quad (2\times \text{ Quadrieren, } 4\times \text{ Multiplizieren}),
\end{aligned}$$

  und 8 Multiplikationen für y_3, also insgesamt 4 Quadrierungen und 14 Multiplikationen.

- Berechnung von „$P + Q$“

$$\begin{aligned}
\lambda &= \frac{(c_1 d_2 - c_2 d_1) \cdot b_1 b_2}{(a_1 b_2 - a_2 b_1) \cdot d_1 d_2} \quad (8\times \text{ Multiplizieren}),\\
x_3 &= -\frac{a_1 b_2 + a_2 b_1}{b_1 b_2} + \lambda^2 \quad (2\times \text{ Quadrieren, } 3\times \text{ Multiplizieren}),
\end{aligned}$$

  unter Berücksichtigung der Ergebnisse aus der Berechnung von λ und 8 Multiplikationen für y_3, also insgesamt 4 Quadrierungen, 19 Multiplikationen.

- Berechnung von „$(x, y) + Q$“

$$\begin{aligned}
\lambda &= \frac{(d_2 - c_2) \cdot b_2}{(x b_2 - a_2) \cdot d_2} \quad (2\times \text{ Multiplizieren}),\\
x_3 &= -\frac{x b_2 + a_2}{b_2} + \lambda^2 \quad (5\times \text{ Multiplizieren}),\\
y_3 &= -1 + \lambda \left(\frac{x}{1} - x_3 \right) \quad (2\times \text{ Multiplizieren}),
\end{aligned}$$

  dabei berechnen wir $a + \frac{c}{d} = \frac{ad+c}{d}$, wobei a konstant oder $a = x$.

  Insgesamt 9 Multiplikationen.

Wir haben hierbei die im Polynomgrad linearen Operationen vernachlässigt. Weiterhin sollten wir beachten, daß eine rationale Addition 3 Multiplikationen erfordert und eine Multiplikation mit x modulo f_C einem Shift mit Skalarmultiplikation und Addition (vgl. Abschnitt 2.2) also größenordnungsmäßig einer Addition entspricht. Wir sehen, daß Inkrementieren halb so teuer ist wie eine normale Addition, und Punktverdopplung ist etwas billiger als die Addition zweier verschiedener Punkte.

Wir geben im folgenden ein Beispiel für die Bestimmung des Eigenwertes mit rationalen Funktionen an. Zur Vereinfachung benutzen wir die folgende Schreibweise für Polynome:

$$[a_0\, a_1 \ldots a_n] := \sum_{i=0}^{n} a_i \cdot x^i.$$

Beispiel: Sei $E = (3011, 68)$ über $\mathbb{F}_{103}$, $l = 11$.

Wir erhalten als Teiler des 11-ten Divisionspolynoms

$$f_C(x) = [55\ \ 45\ \ 59\ \ 4\ \ 67\ \ 1]$$

und als Kandidaten für den Eigenwert 1, 3, 4 und 5. Da die möglichen Eigenwerte dicht liegen, berechnen wir sukzessive alle Punktvielfachen zwischen eins und fünf. Zuerst berechnen wir $y^{p-1} \bmod f_C$ als

$$y^{p-1} = [66\ \ 15\ \ 52\ \ 52\ \ 92].$$

Da y^{p-1} ungleich ± 1 ist, bestimmen wir

$$\begin{aligned} 2 \cdot (x, y) &= \left(\frac{[61\ \ 74\ \ 55\ \ 0\ \ 1]}{[66\ \ 96\ \ 0\ \ 4]}, y \cdot \frac{[43\ \ 37\ \ 101\ \ 24\ \ 73]}{[37\ \ 43\ \ 28\ \ 82\ \ 8]} \right), \\ 3 \cdot (x, y) &= \left(\frac{[3\ \ 62\ \ 52\ \ 51\ \ 39]}{[100\ \ 55\ \ 13\ \ 27\ \ 19]}, y \cdot \frac{[92\ \ 35\ \ 87\ \ 76\ \ 28]}{[18\ \ 58\ \ 102\ \ 31\ \ 44]} \right). \end{aligned}$$

Für die y-Koordinate von $3 \cdot (x, y)$ finden wir heraus, daß

$$y^{p-1} \neq \pm \frac{[43\ \ 37\ \ 101\ \ 24\ \ 73]}{[37\ \ 43\ \ 28\ \ 82\ \ 8]}.$$

Auch für $4 \cdot (x, y)$ finden wir keine Übereinstimung. Wir erhalten schließlich

$$5 \cdot (x, y) = \left(\frac{[18\ \ 49\ \ 34\ \ 97\ \ 66]}{[15\ \ 65\ \ 5\ \ 87\ \ 40]}, y \cdot \frac{[61\ \ 77\ \ 4\ \ 34\ \ 40]}{[52\ \ 36\ \ 29\ \ 66\ \ 52]} \right)$$

sowie

$$y^{p-1} \cdot [52\ \ 36\ \ 29\ \ 66\ \ 52] \equiv_{f_C} [61\ \ 77\ \ 4\ \ 34\ \ 40].$$

Damit ist der gesuchte Eigenwert 5 und für die Spur gilt:

$$c \equiv 8 \bmod 11.$$

Wir wollen noch ein weiteres Verfahren für die Ermittlung des Eigenwertes ansprechen, daß allerdings noch nicht implementiert ist. Die Praxis hat gezeigt, daß es für größere Primzahlen l teuer wird, den Zerfällungsgrad des l-ten modularen Polynoms vom Grad $l + 1$ zu bestimmen; die Kosten hierfür sind erfahrungsgemäß mindestens so hoch wie für die Berechnung von x^p modulo f_C vom Grad $\frac{l-1}{2}$. Weiterhin sind die oben angegebenen Strategien zur Bestimmung des Eigenwertes immer noch teuer. Deshalb schlagen wir eine neue Strategie vor. Wir berechnen x^p und y^p modulo f_C, berechnen dafür den Zerfällungsgrad des l-ten modularen Polynoms nicht mehr und verwenden zur Suche des Eigenwerts einen „richtigen“ Babystep-Giantstep-Algorithmus mit Suche in der Babystep-Tabelle.

Wir zerlegen die Berechnung von $\alpha \cdot (x, y) \bmod f_C$ für $1 \leq \alpha < l$ in

$$\alpha \cdot (x, y) = i \cdot (x, y) + j \cdot G \cdot (x, y),$$

$$\text{wobei } l > G := 2^k \geq \left\lceil \sqrt{l} \right\rceil, k \text{ minimal.}$$

Damit erhalten wir

$$\begin{aligned} \text{Babysteps} &: \ i = 1, 2, \ldots, G - 1 \\ \text{Giantsteps} &: \ j = 1, 2, \ldots, j_0, \end{aligned}$$

wobei $j_0 = \lfloor \frac{l}{2^k} \rfloor$, da $l \not\equiv 0 \bmod 2$. Man kann sich leicht überlegen, daß die obige Zweierpotenz zwischen $\lceil \sqrt{l} \rceil$ und l in unserer Situation stets existiert.

Beachten wir die Analyse über die Anzahl der durchgeführten Polynommultiplikationen bei der formalen Punktaddition mit rationalen Funktionen, so sehen wir, daß ein Babystep nur halb so teuer ist wie ein Giantstep (inkrementieren statt Addition zweier beliebiger Punkte). Deshalb führen wir durch die Wahl von G als nächste Zweierpotenz größer $\sqrt{l}$ mehr Baby- als Giantsteps durch und beschleunigen damit gleichzeitig die Schnelle Exponentiation zur Berechnung von $G \cdot (x, y)$ modulo f_C. Um Platz zu sparen, speichern wir nicht die Babysteps, sondern die Giantsteps in einer Tabelle ab.

Eigenwertbestimmung mit vollständigem Babystep-Giantstep-Algorithmus

EINGABE: E, p, l, f_C

AUSGABE: Eigenwert α des Frobeniusendomorphismus $\Phi_{E_{|E[l]}}$

---

(1) Berechne k minimal mit $l > 2^k \geq \lceil \sqrt{l} \rceil$.

(2) Berechne $g(x) = (x^3 + ax + b)^{(p-1)/2} \bmod f_C$.

(3) Berechne $2^k \cdot (x, y) \bmod f_C$ mit Schneller Exponentiation.

GIANTSTEPS

(4) **for** ($j = 1; j \leq l/2^k; j++$) **do**

(5) $\quad G[j] = j \cdot (2^k \cdot (x, y)) \equiv (h_1(x), y \cdot h_2(x)) \bmod f_C$.

(6) $\quad$ **if** ($h_2(x) \equiv_{f_C} g(x)$) **then**

(7) $\qquad$ **return** $(j \cdot 2^k)$

(8) $\quad$ **fi**

(9) **od**

(10) Berechne $x^p \bmod f_C$.

Eigenwertbestimmung mit vollständigem ...
(Fortsetzung)

BABYSTEPS

(11) **for** ($i = 1; i \leq 2^k - 1; i++$) **do**
(12) $B = ((x^p, y^p) - i \cdot (x, y)) \bmod f_C$.
(13) **if** (es ex. $1 \leq j \leq l/2^k$ mit $G[j] = B$) **then**
(14) **return** $(i + j \cdot 2^k)$
(15) **fi**
(16) **od**

Wir wollen diesen Algorithmus noch etwas modifizieren. Der Grund für die Darstellung formaler Punktvielfacher mittels rationaler Funktionen liegt in der Berechnung des Bruches $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ bei der Punktaddition (vgl. Satz 1.3). Warum invertieren wir nicht den Nenner modulo f_C und repräsentieren die formalen Punktvielfachen als Tupel von Polynomen ? Dazu betrachten wir den folgenden

4.18. Satz *Ist $d' = ord(\alpha)$ in $\mathbb{F}_l^*$, α Eigenwert des Frobeniusendomorphismus $\Phi_{E_{|E[l]}}$,*

$$d = \begin{cases} d', & \text{falls } d' \text{ ungerade} \\ \frac{d'}{2}, & \text{falls } d' \text{ gerade}, \end{cases}$$

so zerfällt f_C über $\mathbb{F}_p$ in $\frac{l-1}{2d}$ irreduzible Faktoren vom Grad d.

Beweis:

$$\Phi_E^d(x, y) = \alpha^d \cdot (x, y) = \pm(x, y)$$

wobei $\alpha^{\frac{d'}{2}} = -1$, da Ordnung 2. Somit gilt

$$x^{p^d} = x \quad \forall \ x = x(P), P \in C \backslash \{\mathcal{O}\},$$

wobei $C \leq E[l]$ die unter Φ_E invariante Untergruppe ist, welche das Polynom f_C bestimmt, und d ist minimal mit dieser Eigenschaft, denn:

Läge eine Nullstelle $x = x(P), P \in C\backslash\{\mathcal{O}\}$ von f_C in einem kleineren Erweiterungskörper von $\mathbb{F}_p$, d.h. $x^{p^i} = x$ mit $i < d$, so müßten schon alle Nullstellen von f_C in diesem kleineren Körper liegen, da P ein Erzeuger von C ist. Das wäre aber ein Widerspruch, da die Ordnung von α nicht d sein könnte. ∎

Wir wissen also, daß $f_C(x)$ reduzibel ist, und somit muß nicht jedes Polynom aus $R := \mathbb{F}_p[x]/(f_C(x))$ invertierbar sein. Wollen wir jedoch ein Polynom ψ, welches keine Einheit im Ring R ist, invertieren, so erhalten wir mit dem $ggT(\psi, f_C)$ einen Teiler von f_C vom Grad $k \cdot d < (l-1)/2,\ k \in \mathbb{N}$. Aber die Nullstellen dieses Teilers sind ebenfalls x-Koordinaten von Eigenvektoren zum gesuchten Eigenwert α; deshalb starten wir in diesem Fall die Eigenwertsuche wieder von neuem und rechnen alle auftretenden Polynome modulo dem gefundenen Teiler. Wegen des geringeren Polynomgrades, wird diese Suche sehr viel schneller sein als die Suche mit f_C.

Rechnen wir die Summe zweier ungleicher, nicht zueinander inverser, formaler Punkte („P+Q“ in Satz 1.3) als Polynome aus, so müssen wir eine erweiterte ggT-Berechnung und 3 Polynommultiplikationen durchführen. Die Praxis zeigt, daß dies, bei unseren Polynomgraden kleiner 500, schneller ist, als die Addition zweier solcher formaler Punkte mittels rationaler Funktionen. Da die Giantsteps nach der Additionsvorschrift für „P+Q“ gebildet werden, benutzen wir Polynome anstatt rationaler Funktionen. Außerdem benötigen wir somit nur 2 Polynome zur Punktrepräsentation anstatt 4 Polynomen, wodurch die zu speichernde Tabelle kleiner wird. Die Berechnung der Babysteps wird nicht verändert, da wird dort nur Inkrementieren und dies ist billiger als ggT-Berechnung.

Analysieren wir den Babystep-Giantstep-Algorithmus, so erhalten wir:

1. $18 \cdot k$ Polynommultiplikationen zur Berechnung von $2^k \cdot (x, y) \bmod f_C$.
2. $\lfloor \frac{l}{2^k} \rfloor \cdot (1 \text{ ggT } + 3 \text{ Mult. }) + \frac{l}{2^k}$ Vergleiche für die Giantsteps.
3. $(2^k - 1) \cdot 9$ Mult. $+ (2^k - 1) \cdot \frac{l}{2^k}$ Vergleiche für die Babysteps.

Für $2^k \approx \sqrt{l}$ wird dies zu

1. $12\sqrt{l} + 18 \log(\sqrt{l}) - 9$ Multiplikationen,
2. $\sqrt{l}$ ggT - Berechnungen,
3. $2\sqrt{l} - 1$ Vergleichen.

4.2.3 Laufzeiten

Wir geben im folgenden einige Laufzeiten für die Bestimmung des Eigenwertes zum einem mit Divisionspolynomen zum anderen mit rationalen Funktionen an.

In der ersten Spalte ist die jeweilige Charakteristik durch die Anzahl ihrer Dezimalstellen angegeben. Unter „K“ steht die Anzahl der Kandidaten für den Eigenwert, und in der darauffolgenden Spalte ist die Anzahl der wirklich getesteten Kandidaten angegeben. Die Spalte mit der Überschrift „EW“ gibt jeweils das Vielfache an, für welches eine Übereinstimmung mit y^p gefunden wurde. Unter „Div.pol.“ sind die Zeiten für das Auffinden des Eigenwertes mittels Divisionspolynomen angegeben. In der darauffolgenden Spalte stehen die Zeiten, die mit rationalen Funktionen benötigt wurden; unter „V“ ist dabei das verwendete Verfahren angegeben (i: sukzessive inkrementieren, b: Babystep-Giantstep).

Schließlich steht in der letzten Spalte jeweils die Zeit zur Berechnung von $y^p \bmod f_C$, welche in den beiden anderen Zeiten nicht enthalten ist.

dd(p)	l	K	G	EW	Div.pol.	rat.Fkt.	V	y^p
100	53	24	6	−6	3 sec	9 sec	i	1 min 28 sec
	97	32	27	−39	1 min 38 sec	2 min 34 sec	i	2 min 09 sec
	157	48	23	−37	2 min 40 sec	4 min 42 sec	i	4 min 11 sec
	199	60	41	68	6 min 09 sec	9 min 44 sec	i	4 min 35 sec
150	103	16	12	38	2 min 15 sec	3 min 47 sec	b	5 min 39 sec
	211	48	31	66	9 min 22 sec	15 min 59 sec	i	12 min 00 sec
	317	78	39	73	19 min 01 sec	32 min 55 sec	i	21 min 56 sec
200	103	32	7	9	29 sec	1 min 08 sec	i	11 min 36 sec
	193	16	13	83	14 min 16 sec	18 min 42 sec	b	23 min 21 sec
	307	32	28	139	45 min 59 sec	1 h 02 min	b	42 min 58 sec
	401	20	17	160	57 min 25 sec	46 min 41 sec	b	49 min 42 sec
250	103	16	7	17	1 min 47 sec	4 min 33 sec	b	19 min 13 sec
	199	60	10	18	3 min 21 sec	7 min 31 sec	i	39 min 12 sec
	313	96	55	85	45 min 35 sec	1 h 15 min	i	1 h 11 min
300	107	52	12	12	1 min 48 sec	3 min 16 sec	i	33 min
	317	156	90	92	1 h 16 min	1 h 53 min	i	1 h 55 min
	421	12	2	43	20 min	35 min	b	2 h 21 min
	523	56	34	−137	2 h 31 min	3 h 36 min	b	3 h 37 min
375	103	32	22	34	9 min	14 min	i	55 min
	223	72	10	20	8 min	17 min	i	2 h 01 min
	401	160	30	40	38 min	1 h 13 min	i	3 h 51 min
	503	250	195	−195	5 h 26 min	7 h 39 min	i	4 h 34 min
	839	418	326	−326	19 h 59 min	23 h 19 min	i	8 h 10 min

Wie wir nach der Analyse der bei der Berechnung der Punktvielfachen durchzuführenden Polynommultiplikationen erwarten konnten, finden wir den Eigenwert schneller, falls wir Divisionspolynome verwenden. Um Gruppenordnungen über Primkörpern großer Charakteristik zu berechnen, verwenden wir eine verteilte Version des Atkin-Algorithmus, die auf Basis des Systems LiPS arbeitet (vgl. Abschnitt 6.3). Auf den einzelnen Rechnern steht den unter LiPS arbeitenden Programmen verhältnismäßig wenig Speicherplatz zur Verfügung, so daß wir in der Regel rationale Funktionen verwenden. Weiterhin sehen wir, daß die Berechnung von y^p oftmals aufwendiger ist als die Bestimmung der formalen Punktvielfachen, insbesondere dann, wenn nur kleine Vielfache zum Auffinden des Eigenwertes bestimmt werden müssen.

4.3 Bestimmung von c modulo 2

In den bisher vorgestellten Verfahren haben wir die Spur c des Frobeniusendomorphismus Φ_E modulo einer ungeraden Primzahl l berechnet. Es ist einfach, die Spur c auch modulo 2 und eventuell modulo 4 exakt zu bestimmen. Die Grundlage bildet der folgende

4.19. Satz (Cauchy) *Ist G eine endliche Gruppe und $p \in \mathbb{N}$ eine Primzahl, so gilt:*

$$p \mid \#G \Longleftrightarrow \text{ es ex. } a \in G \text{ mit } ord(a) = p.$$

Beweis: Die Richtung von rechts folgt aus dem Satz von Lagrange ([Mey1, Satz 1.7.7, S. 47]). Die Richtung von links nach rechts folgt aus dem 1.Sylow-Satz (vgl. [Mey1, Satz 2.2.2, S.70]) und der Tatsache, daß eine Gruppe mit Primzahlordnung zyklisch ist. ■

Da nach dem Satz von Hasse $\#E(\mathbb{F}_p) = p + 1 - c$ ist, wobei in unserer Situation p eine ungerade Primzahl ist, gilt

$$\begin{aligned} c \equiv_2 0 \quad &\Longleftrightarrow \quad \#E(\mathbb{F}_p) \equiv_2 0 \\ &\Longleftrightarrow \quad \text{es ex. } P \in E(\mathbb{F}_p) \text{ mit } ord(P) = 2. \end{aligned}$$

Für einen Punkt $P = (x, y) \in E(\mathbb{F}_p)$ der Ordnung 2 gilt $P = -P$, also $y = 0$. Somit existiert genau dann ein Punkt der Ordnung 2, wenn die Gleichung $x^3 + ax + b \in \mathbb{F}_p[x]$ mindestens eine Nullstelle in $\mathbb{F}_p$ hat. Wir erhalten

$$c \equiv_2 0 \Longleftrightarrow deg\left(ggT\left(x^p - x, x^3 + ax + b\right)\right) \geq 1.$$

Um diesen Test durchzuführen, berechnen wir zuerst $x^p \bmod x^3 + ax + b$; dies geht schnell, da der Modul den Grad 3 besitzt.

Falls $ggT\left(x^p - x, x^3 + ax + b\right) = x^3 + ax + b$ ist, besitzt $E(\mathbb{F}_p)$ 3 Punkte mit Ordnung 2 und die 2-Torsionsgruppe von $E(\overline{\mathbb{F}_p})$ liegt in $E(\mathbb{F}_p)$. Damit ist $\#E(\mathbb{F}_p) = p + 1 - c \equiv_4 0$, also $c \equiv_4 p + 1$.

Kapitel 5

Gesamtalgorithmus und Laufzeiten

Um einen Überblick zu erhalten, fassen wir in diesem Abschnitt den gesamten Algorithmus in Form eines informalen Programms zusammen. Anschließend geben wir einige Laufzeiten und Daten für die Berechnung von Gruppenordnungen an.

Atkin-Algorithmus

EINGABE: Kurvenparameter a, b der Kurve E über $\mathbb{F}_p$, Charakteristik $p > 3$.
AUSGABE: $\#E(\mathbb{F}_p)$

ÜBERPRÜFE DIE SPEZIALFÄLLE (ABSCHNITT 2.4).

(1) **if** $(4a^3 + 27b^2 == 0)$ **then**
(2) **return** (Die Parameter definieren keine elliptische Kurve.)
(3) **fi**
(4) **if** (E ist supersingulär) **then**
(5) **return** $(p + 1.)$
(6) **fi**
(7) **if** (E ist $\mathbb{F}_p$-isogen zu Kurve mit j-Inv. 0 oder 1728) **then**
(8) **return** (Berechnete Gruppenordnung.)
(9) **fi**

Atkin-Algorithmus
(Fortsetzung)

INITIALISIERUNG

(10) Bestimme eine obere Schranke $MAXKOMB$ für die Anzahl der Kombinationen in der Babystep-Giantstep-Phase.
(11) $KOMB = 1$. /*Anzahl der Kombinationen*/
(12) $PROD = 2$. /*Produkt der Primzahlen*/
(13) $i = 1$. /*Nr. der Primzahl*/
(14) Berechne $j(E)$. /*j-Invariante*/

BERECHNE DIE KONGRUENZEN.

(15) Berechne $c \bmod 2$. /*Abschnitt (4.3)*/
(16) **while** ($PROD \le 4\sqrt{p}$ und $KOMB > MAXKOMB$) **do**
(17) $l = i$-te ungerade Primzahl.
(18) Lese die Koeffizienten $a_{r,k}$ des l-ten modularen Polynoms $\Phi_l(x,y) = \sum_{r,k} a_{r,k} \cdot x^r \cdot y^k$ ein, und berechne das l-te modulare Polynom für E, $\Phi_l(x, j(E)) \in \mathbb{F}_p[x]$, durch Substitution der j-Invarianten und Reduktion mod p.
(19) Bestimme den Zerfällungstyp /*Abschnitt 2.2*/ und eine Nullstelle $f(\tau)$ in $\mathbb{F}_p$ von $\Phi_l(x, j(E))$, falls existent.
(20) **if** (Φ_l hat keine Nullstelle in $\mathbb{F}_p$) **then**

ATKIN-PRIMZAHL

(21) Berechne mögliche $c \bmod l$: /*Abschnitt 2.2*/
Bestimme quadr. Nichtrest in $\mathbb{F}_l$.
Bestimme Erzeuger von $\mathbb{F}_{l^2}$.
Bestimme alle Elemente der Ordnung d in $\mathbb{F}_{l^2}$ (d aus Zerfällungstyp).
Bestimme alle möglichen $c \bmod l$.
(22) **else**

Atkin-Algorithmus
(Fortsetzung)

Elkies-Primzahl

(23) **if** (Φ_l ist vom Typ 1 /*Abschnitt 4.1.2*/) **then**
(24) Berechne $\tilde{E}$ und P_1. /*Abschnitt 4.1.2*/
(25) Berechne f_C. /*Abschnitt 4.1*/
(26) **else**
(27) Faktorisiere $\Phi_l(f(\tau), y)$ /*Abschnitt 4.1.2*/
und erhalte alle möglichen Werte für $j(l\tau)$.
(28) **for** (alle Werte $j(l\tau)$, solange $f_C \not| \psi_l$) **do**
(29) Berechne $\tilde{E}$ und P_1. /*Abschnitt 4.1.2*/
(30) Berechne f_C. /*Abschnitt 4.1*/
(31) **if** (f_C teilt l-tes Divisionspolynom) **then**
(32) Beende FOR-Schleife.
(33) **fi**
(34) **od**
(35) **fi**
(36) Berechne Eigenwert α von Φ_E. /*Abschnitt 4.2*/
(37) $c \equiv \alpha + \frac{p}{\alpha} \bmod l$.
(38) **fi**
(39) Berechne $PROD$ und $KOMB$ neu.
(40) $i = i + 1$.
(41) **od**

Babystep-Giantstep-Phase

(42) Initialisiere die Babystep-Giantstep-Phase, d.h. berechne
die versch. Moduln und Listen. /*Abschnitt 3.1*/
(43) Berechne die Babysteps. /*Abschnitt 3.2*/
(44) Berechne die Giansteps.
(45) Beweise die gefundene Gruppenordnung. /*[Mü94, Beweis der Gruppenordnung]*/

Im folgenden geben wir einige Daten für durchgeführte Berechnungen von Gruppenordnungen an. Wir beginnen mit einem Überblick über die benötigten Laufzeiten. In der Spalte $dd(p)$ ist die Anzahl der Dezimalstellen der jeweiligen Charakteristik p aufgelistet. Unter dem Stichwort „Zerfällungstyp" ist der Laufzeitanteil für die Berechnung der Zerfällungstypen angegeben. An die Berechnung eines Zerfällungstyps für das l-te modulare Polynom schließt sich die Bestimmung der Spur c an, wobei nur die exakte Berechnung von c modulo l, d.h. Verwendung des Erweiterungsalgorithmus, für die Laufzeit von Bedeutung ist.

$dd(p)$	Zeit	Zerfällungstyp	Spurberechnung
31	2 min	86%	07%
41	9 min	83%	16%
51	20 min	79%	20%
60	45 min	70%	29%
70	1 h 13 min	72%	27%
80	2 h 20 min	69%	30%
90	4 h 39 min	72%	27%
100	8 h 40 min	72%	27%
150	44 h 21 min	70%	27%
201	174 h 18 min	70%	29%
250	425 h 55 min	67%	32%
303	1332 h 00 min	68%	31%
375	2119 h 00 min	61%	37%

Es fällt auf, daß der Anteil für die Berechnung der Zerfällunsgtypen bei der 375-stelligen Charakteristik ungefähr 10 Prozent geringer ist als bei den übrigen Berechnungen. Für diese Gruppenordnung haben wir eine etwas andere Strategie verfolgt. Ob ein Modul l eine Atkin-Primzahl oder eine Elkies-Primzahl ist, d.h. ob wir die Spur nur ungenau oder exakt bestimmen können, wissen wir, nachdem der ggT des l-ten modularen Polynoms mit $x^p - x$ berechnet ist. Davon ausgehend, daß für große Atkin-Primzahlen l sehr viele Möglichkeiten für c modulo l berechnet werden, haben wir solche großen Moduln nicht benutzt und uns damit die Berechnung des Zerfällunsgrades d gespart. Dies ist eine Strategie, die wir in Zukunft auch für kleinere Charakteristiken verwenden wollen.

Für obige Berechnungen geben wir im folgenden die verwendete Kurve sowie die Charakteristik an. Wir beschreiben das Verhältnis von Atkin zu Elkies Primzahlen (E/A), geben die Anzahl der Kombinationen für die Babystep-Giantstep-Phase und schließlich die berechnete Gruppenordnung an. Wir numerieren die Auflistung anhand der in der Tabelle angegebenen Stelligkeiten der Charakteristiken.

31 **Stellen**

$$\begin{aligned} E &= (9051969, 11081969) \\ p &= 10^{30} + 123456789123456789987657 1 \\ \mathrm{E/A} &= 5/9 \\ \#\mathrm{BG} &\approx 10^8 \\ c &= -1191647832538652 \end{aligned}$$

41 **Stellen**

$$\begin{aligned} E &= (9051969, 11081969) \\ p &= 10^{40} + 1182021 \\ \mathrm{E/A} &= 11/9 \\ \#\mathrm{BG} &\approx 10^5 \\ c &= -58513442752852825315 \end{aligned}$$

51 **Stellen**

$$\begin{aligned} E &= (9051969, 11081969) \\ p &= 10^{50} + 1234567899876543212345678987654481 \\ \mathrm{E/A} &= 13/10 \\ \#\mathrm{BG} &\approx 10^4 \\ c &= -1199078084683684623414270 6 \end{aligned}$$

60 **Stellen**

$$\begin{aligned} E &= (9051969, 11081969) \\ p &= 10^{59} + 19 \\ \mathrm{E/A} &= 16/11 \\ \#\mathrm{BG} &\approx 10^3 \\ c &= 36663314915554875804786833567 6 \end{aligned}$$

70 **Stellen**

$$\begin{aligned} E &= (9051969, 11081969) \\ p &= 10^{69} + 9 \\ \mathrm{E/A} &= 18/12 \\ \#\mathrm{BG} &\approx 10^5 \\ c &= -3784332196968673298996548298712246 6 \end{aligned}$$

80 **Stellen**

$$\begin{aligned} E &= (9051969, 11081969) \\ p &= 10^{79} + 49 \\ \mathrm{E/A} &= 21/12 \\ \#\mathrm{BG} &\approx 10^4 \\ c &= -42587421481648493716572411110335459964 82 \end{aligned}$$

90 **Stellen**

$$
\begin{aligned}
E &= (9051969, 11081969) \\
p &= 10^{89} + 31 \\
\mathrm{E/A} &= 20/19 \\
\#\mathrm{BG} &\approx 10^{5} \\
c &= -247041773797161772603677408304514295289649644
\end{aligned}
$$

100 **Stellen**

$$
\begin{aligned}
E &= (9051969, 11081969) \\
p &= 10^{99} + 289 \\
\mathrm{E/A} &= 22/26 \\
\#\mathrm{BG} &\approx 10^{4} \\
c &= -48710847154147308223714987590822768142801634966450
\end{aligned}
$$

150 **Stellen**

$$
\begin{aligned}
E &= (9051969, 11081969) \\
p &= 10^{149} + 183 \\
\mathrm{E/A} &= - \\
\#\mathrm{BG} &\approx 10^{8} \\
c &= -334951732923243334323705814040951664 10 \\
&\quad 9585623329852244233168227866046342483
\end{aligned}
$$

201 **Stellen**

$$
\begin{aligned}
E &= (9051969, 11081969) \\
p &= 10^{200} + 127392178378947984758473857485781361276 37 \\
&\quad 1263671286387647369473928758823784824611 \\
\mathrm{E/A} &= 42/75 \\
\#\mathrm{BG} &\approx 10^{8} \\
c &= -3097103000043261706675163282971314884912342064053 1 \\
&\quad 04617886327483815411056838547628128900332464438 00
\end{aligned}
$$

250 **Stellen**

$$
\begin{aligned}
E &= (9051969, 11081969) \\
p &= 10^{249} + 1291 \\
\mathrm{E/A} &= 50/97 \\
\#\mathrm{BG} &\approx 10^{8} \\
c &= -2862928389053542389986100979502413891256563 \\
&\quad 8376522979049802620357518443887406873271408 \\
&\quad 665210522602382217460818606015970032398
\end{aligned}
$$

303 **Stellen**

$$\begin{array}{rcl}
E & = & (90569, 110869) \\
p & = & 10^{302} + 399 \\
\mathrm{E/A} & = & 55/72 \\
\#\mathrm{BG} & \approx & 10^{6} \\
c & = & -7839209985339525998205342636192001225627161 31 \\
 & & 3312286929789466358953200631518568753474568 39 \\
 & & 3017477641836748249294292674422101233708313 27 \\
 & & 6092322441156548
\end{array}$$

375 **Stellen**

$$\begin{array}{rcl}
E & = & (9051969, 11081969) \\
p & = & 10^{374} + 169 \\
\mathrm{E/A} & = & 70/65 \\
\#\mathrm{BG} & \approx & 10^{5} \\
c & = & 66515880622942259450404201980659722700703805586159 2195 \\
 & & 48756894165746088741727020472212357428409089007669 4286 \\
 & & 00871013183903050823460008229976494046242985177441 2934 \\
 & & 9701163422706189949782368
\end{array}$$

Kapitel 6

Programme

6.1 Das Programm *eco*

Wir haben den Grundalgorithmus von Atkin und die Erweiterung von Atkin und Elkies in einem in C und C++ geschriebenen Programm namens *eco* realisiert. Um das Programm zu benutzen, muß der Benutzer das Kommando

eco -*m meqpath*

eingeben, wobei *meqpath* das Verzeichnis angibt, in dem die modularen Gleichungen stehen. Nach diesem Aufruf bittet das Programm den Benutzer um Eingabe der Charakteristik p und der Kurve $E = (a, b)$, wobei a, b automatisch modulo p reduziert werden. Anschließend wird die Gruppenordnung berechnet und das Ergebnis auf dem Bildschirm ausgegeben. Man kann den Programmablauf von *eco* durch einige Parameter beeinflußen; wir werden diese im folgenden erläutern:

-report Druckt markante Daten der Berechnung aus.

-Report Druckt sehr viele Berechnungsdaten aus.

-time Gibt am Ende ein Timing aus.

-Time Gibt am Ende ein detailliertes Timing aus.

-finp D Liest die Charakteristik p und die Kurve $E = (a, b)$ aus der Datei D ein; dabei ist D eine „ascii"-Datei im Format: $p\ a\ b$.

-moduls D Berechne nur c modulo l für alle l, die in der Datei D angegeben sind; dabei ist D eine „ascii"-Datei, in der die Moduln untereinander aufgelistet sind.

-proof Bewirkt, daß die Gruppenordnung, die in der Babystep-Giantstep-Phase berechnet wurde, bewiesen wird.

-rf D Speichert die Relationen c modulo l in der Datei D ab; existiert D schon, so werden die neuen Relationen angehängt. Diese Daten werden binär abgespeichert (s. Abschnitt 6.4).

-rrel D Es werden keine Relationen berechnet, sondern das Programm liest die Relationen aus der Datei D ein und startet damit die Babystep-Giantstep-Phase.

-bf D Speichert die Babystep-Tabelle in der Datei D ab.

-r2f D Speichert die Tabelle der R_2-Werte in der Datei D ab.

-rbaby Liest die R_2-Werte aus der mit $-rf2$ und die Babystep-Tabelle aus der mit $-bf$ spezifizierten Datei ein und startet die Giantstep-Phase.

-sim Rechnet in der Babystep-Giantstep-Phase mit ggT-Trick, speichert die R_2-Werte ab, und liest sie in Päckchen wieder ein.

-aonly Rechnet nur mit dem Grundalgorithmus von Atkin, d.h. benutzt keine Elkies-Erweiterung.

-noadeg Bewirkt, daß für Atkin-Primzahlen kein Zerfällungsgrad berechnet wird (nur in Verbindung mit *-moduls*).

-wsplit Schreibt für jeden Modul l den Zerfällungstyp in eine Datei; der Name der Datei ist: fout_l_r_sp_bak.

-rsplit Liest für jeden mit $-moduls$ spezifizierten Modul l den Zerfällungstyp aus einer Datei ein und berechnet anschließend die Spur modulo l; ist kein Zerfällungstyp vorhanden, wird der Modul l übersprungen.

-sponly Berechnet für die mit $-moduls$ spezifizierten Moduln l den Zerfällungstyp, speichert diesen in einer Datei und berechnet für Atkin-Primzahlen die Spur.

Die Programme, mit denen man modulare Gleichungen berechnen kann, werden in [Le94, Kapitel 7] beschrieben.

Das Programm *eco* kann über eine X11-Oberfläche **xeco** bedient werden. Diese Oberfläche wurde aus dem Programm **xfact** übernommen, welches von der Gruppe des Lehrstuhls Prof. Dr. J. Buchmann erstellt wurde, die sich mit dem Faktorisieren großer Zahlen beschäftigt. Wenn man die Oberfläche *xeco* verwendet, stehen in der derzeitigen Version die angegebenen Parameter von *eco* nicht zur Verfügung.

6.2 Funktionsnamen

In diesem Abschnitt beschreiben wir für die wichtigsten der implementierten Funktionen, welche Datei welche C-Funktionen beinhaltet, und wir erläutern stichwortartig die Aufgaben der einzelnen Funktionen.

Datei	Funktion	Aufgabe
buildLlist.C	*buildLlist()*	bekommt eine Liste von Moduln und steuert die Berechnung der Spur
readfile.C	*readpol()*	liest das modulare Polynom ein
	read_ark()	liest die Koeff. des modularen Polynoms ein (ascii - Format)
	fwrite_Int()	schreibt eine libI-Zahl in eine Datei
	fread_Int()	liest eine libI-Zahl aus einer Datei
	read_ark_bin()	liest die Koeff. des modularen Polynoms ein (binäres - Format)
	readmeqcoeff()	liest die Koeff. des modularen Polynoms ein
split_type.C	*split_type()*	berechnet den Zerfällungstyp des modularen Polynoms.
cmodl.C	*cmodl()*	bekommt den Zerfällungstyp für l und berechnet c modulo l
cmodl_atkin.C	*cmodl_atkin()*	berechnet die Spur für Atkin-Primzahlen
cmod2.C	*cmod2()*	berechnet die Spur modulo 2
divpol.C	*div_of_divpol()*	berechnet $\tilde{E}, P_1$ und den Teiler f_C des l-ten Divisionspolynoms
	compute_divpol()	bekommt $\tilde{E}$ und P_1 und führt den Koeff.vergleich zur Bestimmung von f_C durch
elkies_series.C	*wp_coeff()*	berechnet die Koeffizienten der Weierstrass'schen $\wp$-Funktion
	compute_wp()	berechnet die Weierstrass'sche $\wp$-Funktion
	coeffserie()	berechnet die „innere“ Reihe g im Koeffizientenvergleich

Datei	Funktion	Aufgabe
guess_jltau.C	*guess_jltau()*	bestimmt die möglichen Werte von $j(l\tau)$ für Typ 2 Polynome
eigenvalue.C	*eigenvalue()*	steuert die Berechnung des Eigenwertes
	evmod3()	berechnet den Eigenwert modulo 3
compute_klist.C	*compute_klist()*	berechnet Kandidaten für den Eigenwert
schoofpart.C	*schoofpart()*	berechnet den Eigenwert mit Divisionspolynomen
eigenv_rat.C	*eigenv_rat()*	berechnet den Eigenwert mit rationalen Funktionen
compute_psi.C	*compute_psi()*	berechnet das n-te Divisionspolynom
read_split_file.C	*read_split()*	liest den Zerfällungstyp ein
	write_split()	schreibt den Zerfällungstyp in eine Datei
primes.C	*eratost()*	Sieb des Erathostenes
	findord()	bestimmt ein Element der Ordnung d in $\mathbb{F}_l$
	ord()	berechnet die Ordnung eines Elementes in $(\mathbb{Z}/n\mathbb{Z})^*$
	rel_prime()	berechnet alle zu einer Zahl n relativ-primen Zahlen kleiner n
	allofd()	berechnet alle Elemente der Ordnung d in $\mathbb{F}_l$
primesl2.C	*findord_l2()*	bestimmt ein Element der Ordnung d in $\mathbb{F}_{l^2}$
	allofd_l2()	berechnet alle Elemente der Ordnung d in $\mathbb{F}_{l^2}$
sqrt_l2.C	*sqrt_l2()*	zieht die Wurzel aus Elementen in $\mathbb{F}_{l^2}$
fastex.C	*fastex()*	Schnelle Exponentiation in $\mathbb{F}_l$
	fastex_l2()	Schnelle Exponentiation in $\mathbb{F}_{l^2}$
divis.C	*divisors()*	berechnet alle Teiler einer Zahl
choose_relations.C	*choose_relations()*	berechnet die Moduln $M_{1,1}, M_{1,2}, M_2$, M_3 sowie die zugehörigen Listen für die Babystep-Giantstep-Phase

Datei	Funktion	Aufgabe
baby.C	*babysteps*()	berechnet die Babysteps
baby_simul.C	*babysteps_simul*()	berechnet Babysteps mit ggT-Trick
baby_fisi.C	*babysteps_fisi*()	berechnet Baysteps mit ggT-Trick; die R_2-Werte werden in Päckchen berechnet und sortiert
giant.C	*giantsteps*()	berechnet die Giantsteps
giant_memsi.C	$giant_memsi()$	berechnet Giantsteps mit ggT-Trick
chinese.C		verschiedene Routinen für Chinesischen Restsatz
quicksort.C		verschiedene quicksort - Routinen
binsearch.C		Binäre Suche
verify.C	*verify_order*() *simple_verify*()	beweist die Gruppenordnung führt ein Auscheidungsverfahren durch
spezial.C	*cornacchia*() *check_super*() *check*_1728_0()	berechnet Elemente der Norm p überprüft auf Supersingularität überprüft auf $\mathbb{F}_p$-Isogenie
dense_psr.C		Arithmetik für dichte Fourierreihen
sparse_psr.C		Arithmetik für dünne Fourierreihen
wepadd.C		Addition von Punkten
weppow.C		Schnelle Exponentiation für Punkte
wepadd_simul.C		Addition mit ggT-Trick
ZZ_pXrat.C		Arithmetik für rationale Funktionen
wepadd_ratpol.C		Addition von formalen Punkten mit rationalen Funktionen
weppow_ratpol.C		Schnelle Exponentiation für formale Punkte

Zusätzlich verwenden wir die Langzahl-Arithmetik (*libI.a*) von Ralf Dentzer sowie die Polynomarithmetik (*GF.a*) von Victor Shoup, die auf der Langzahl-Arithmetik (lip.c) von Arjen Lenstra aufbaut. Wir haben von Thomas Denny einige Funktionen zur Berechnung des Legendre-Symbols und zur Bestimmung von Wurzeln (Ressol-Algorithmus) übernommen. Von Thomas Sosnowski haben wir einen Primzahltest (Miller-Rabin) erhalten. Andreas Müller hat uns Funktionen zur Probedivision zur Verfügung gestellt, und die Arithmetik für elliptische Kurven haben wir aus den vorhandenen Funktionen von Franz-Dieter Berger und Volker Müller aufgebaut.

6.3 Verteilung mit LiPS

Im Rahmen unserer Implementierung des Atkin-Algorithmus haben wir auch eine verteilte Version erstellt. Dazu verwendeten wir das System LiPS, welches aus einer Bibliothek von C-Funktionen besteht, mit denen es möglich ist, eine Berechnung auf einem Netz von Workstations zu verteilen (vgl. [Li93]).

Wir geben kurz den grundlegenden Ablauf einer solchen Berechnung an. Eine Maschine, der sogenannte Master-Rechner, startet das System und erstellt eine Aufgabenliste. Anschließend verschickt er die Aufgaben an die zur Verfügung stehenden Maschinen, die sogenannten Worker; welche Rechner verfügbar sind, wird in einer speziellen Datei, der CLIENT_CONF-Datei, spezifiziert. Nach Abarbeitung der Aufgabe schickt eine Worker-Maschine die Resultate zum Master-Rechner zurück, der die Ergebnisse auswertet und dem Worker eine neue Aufgabe zuteilt, falls die Gesamtberechnung noch nicht abgeschlossen ist. Die Benutzung von LiPS ist einfach, denn dem Programmierer stehen zum Versenden und Empfangen von Nachrichten C-Funktionen zur Verfügung, wobei die Kommunikation zwischen den Rechnern vom LiPS-System übernommen wird.

Bei der Verteilung haben wir einen einfachen Weg gewählt, der sich direkt aus der Struktur des Atkin-Algorithmus ergibt. Da die Spur des Frobeniusendomorphismus modulo kleiner Primzahlen berechnet wird, erstellt der Master-Rechner eine Liste von Primzahlen und verschickt diese sukzessive, bis alle Worker-Rechner versorgt sind. Jeder Worker berechnet die Spur modulo der ihm zugeteilten Primzahl, schickt die Kongruenzen zum Master zurück und erhält die nächste Primzahl, für die dem Master-Rechner noch keine Ergebnisse vorliegen. Sind genügend Relationen eingetroffen, so wird auf einem Rechner die Babystep-Giantstep-Phase durchgeführt; dieser Teil des Algorithmus ist noch nicht verteilt implementiert.

Erst durch die Benutzung von LiPS war es uns möglich sehr große Gruppenordnungen zu berechnen. Die größte bisher durchgeführte Rechnung war die Bestimmung der Anzahl der $\mathbb{F}_p$-rationalen Punkte der elliptischen Kurve $E = (9051969, 11081969)$ über dem Primkörper der Charakteristik $p = 10^{374} + 169$. Diese Berechnung dauerte 7 Tage auf bis zu 40 Maschinen mit jeweils 22 MIPS. Die Gesamtzeit betrug 2119 h.

Die von uns geschriebenen LiPS-Programme heißen TRACE-master und TRACE-worker. Wir fassen ihre oben beschriebene Arbeitsweise in den beiden nachfolgenden, informalen Algorithmen zusammen.

TRACE-master

EINGABE: Kurve (a, b) über $\mathbb{F}_p$, Moduln $l_1, \ldots, l_N$
AUSGABE: mögliche Werte für c modulo l_i, $1 \leq i \leq N$

(1) Initialisiere LiPS.
(2) Initialisiere die zur Verfügung stehenden Worker-Maschinen $M_1, \ldots, M_n$.
(3) Initialisiere ein Feld $DONE$ der Größe N mit 0.
(4) Versende die ersten n Moduln $l_1, \ldots, l_n$ an $M_1, \ldots, M_n$.
(5) **while** (es ex. $1 \leq i \leq N$ mit $DONE[i] == 0$) **do**
(6) Warte auf das Ergebnis eines Workers.
(7) **if** (Worker w hat c mod l_j geschickt) **then**
(8) **if** $((DONE[j] == 0))$ **then**
(9) Speichere c mod l_j.
(10) $DONE[j] = 1$.
(11) **fi**
(12) Suche nächstes $1 \leq i \leq N$ mit $DONE[i] == 0$, falls existent, und sende den Auftrag zur Berechnung von c mod l_i an den Worker w.
(13) **fi**
(14) **od**
(15) Beende alle Prozesse auf den Workern.

TRACE-worker

EINGABE: %
AUSGABE: %

(1) Initialisiere LiPS.
(2) **while** (1) **do**
(3) Empfange Kurve (a, b) über $\mathbb{F}_p$ sowie Primzahl l vom Masterprozeß.
(4) Berechne c modulo l.
(5) Sende die Kongruenzen zum Master-Rechner.
(6) **od**

Ein Problem bei der Verteilung mit LiPS sind die modularen Gleichungen, die jeder Worker-Maschine zugänglich sein müssen. Da die Rechner an verschiedenen Dateisystemen angeschlossen sein können, müssen die modularen Gleichungen auf mehrere Dateisysteme kopiert werden. Wollen wir große Charakteristiken verarbeiten, z.B. $p \approx 10^{374}$, so benötigen wir die l-ten modularen Gleichungen bis $l \approx 800$, was einen Speicherplatzbedarf von 140 MB zur Folge hat. Wir haben uns eine Strategie überlegt, um dieses Problem zu begrenzen; diese ist allerdings noch nicht implementiert. Die Idee ist, die zur Verfügung stehenden Worker-Rechner in zwei Gruppen zu unterteilen, von denen nur der einen die modularen Gleichungen zugänglich sind. Die Maschinen dieser Gruppe dienen als Server für die andere Gruppe, welche die eigentliche Berechnung durchführt. Damit müssen die modularen Gleichungen nur noch auf einem Dateisystem zur Verfügung stehen.

Das l-te modulare Polynom

$$\Phi_l(x, y) = \sum_{r,k} a_{r,k} \cdot x^r \cdot y^k \in \mathbb{Z}[x, y]$$

ist durch die Koeffizienten $a_{r,k}$ gegeben, welche in einer Datei gespeichert sind. Wir erhalten daraus das l-te modulare Polynom für unsere Kurve E über $\mathbb{F}_p$, indem wir die Koeffizienten modulo p reduzieren und die j-Invariante der Kurve, $j(E)$, für die y-Koordinate einsetzen. Zur Berechnung des Zerfällungstyps benötigen wir nur das Polynom $\Phi_l(x, j(E))$ (vgl. Abschnitt 2.2). Zur Berechnung von $\tilde{E}$ und P_1 (vgl. Abschnitt 4.1.2) brauchen wir auch die Koeffizienten $a_{r,k}$ des modularen Polynoms. Deshalb teilen wir die zur Verfügung stehenden Worker-Rechner in 2 Gruppen G_1 und G_2 auf, wobei alle Maschinen aus G_1 an ein gemeinsames Dateisystem angeschlossen sind, auf dem die modularen Polynome stehen. Die Rechner aus G_1 dienen als Server, die Rechner aus G_2 berechnen die Spur c des Frobeniusendomorphismus modulo l. Sendet der Master-Rechner an $w \in G_2$ den Auftrag zur Berechnung von c modulo l, so sendet er an den nächsten freien Server $s \in G_1$ den Auftrag, die l-te modulare Gleichung einzulesen, die j-Invariante zu substituieren, die Koeffizienten zu reduzieren und das so berechnete modulare Polynom an den Worker w zu senden. Benötigt der Worker w im Laufe der Berechnung die Werte $\tilde{E}$ und P_1, so sendet er eine Nachricht und eine Nullstelle des modularen Polynoms an den Master-Rechner. Dieser gibt die Berechnung bei einem Server aus G_1 in Auftrag, der nach Bestimmung von $\tilde{E}$ und P_1 diese Werte zum Worker w schickt.

6.4 Hilfsprogramme

In diesem Abschnitt beschäftigen wir uns mit der Verwaltung der Relationen. Wir erläutern zuerst das Format, in welchem die Relationen gespeichert sind und beschreiben dann die Funktionalität einiger Programme, die die Handhabung der Relationendateien erleichtern.

Während des Programmablaufes sind die Relationen in einer Liste gespeichert. Diese Liste besteht aus Zeigern auf Strukturen, die folgendermaßen definiert sind:

```
struct {
   long  l      ;
   long *cpos   ;
   long  chosen ;
       } Llist  ;
```

Die Variable l enthält den Modul l, und in dem Feld *cpos* sind die möglichen Werte für c modulo l gespeichert; dabei gibt *cpos*[0] die Länge des Feldes an. Die Variable *chosen* wird in der Babystep-Giantstep-Phase benötigt.

Diese Strukturen werden in einer sogenannten Relationendatei binär abgelegt. Am Dateianfang steht die Anzahl der in der Datei enthaltenen Llist-Strukturen, darauf folgen die Strukturen selbst. Wir gehen im folgenden auf die Hilfsprogramme zur Handhabung von Relationendateien ein.

Das Programm **RELview** *[-d -s -p] [-l] relfile* dient zum Ansehen der Relationen. Wird es ohne Parameter aufgerufen, so wird für jede Relation $c \bmod l$, die in $relfile$ abgelegt ist, ein Tripel $[l, n, i]$ ausgegeben; n gibt die Anzahl der möglichen Werte von c modulo l an, i ist stets 1 (wird intern benutzt). Die Ausgabenreihenfolge entspricht der Anordnung in der Datei. Die einzelnen Parameter haben folgende Bedeutung:

-l Gibt die Relationen im langen Format aus, d.h. für jeden Modul l werden alle möglichen Werte von $c \bmod l$ ausgegeben.

-d Sortiert die Relationen aufsteigend nach ihrer Dichte.

-p Sortiert die Moduln l aufsteigend nach ihrer Größe.

-s Sortiert die Moduln aufsteigend nach der Anzahl der Möglichkeiten von $c \bmod l$.

Wird *RELview* mit Parametern aufgerufen, so wird eine Stelligkeitsgrenze abgefragt, die mit dem Produkt der Moduln l überschritten werden soll; anschließend wird die Nummer k der Relation ausgegeben, mit der diese Grenze überschritten wird sowie die Anzahl aller Kongruenzen für c bis zur Relation k einschließlich.

Das Programm **RELcat** *d1 d2 to d3* konkateniert die beiden Relationendateien *d1* und *d2* und legt das Ergebnis in der Datei *d3* ab.

Mit **RELexam** *[-miss] relfile* kann man sich die Moduln l ausgeben lassen, die in Datei *relfile* gespeichert sind. Verwendet man den Parameter *-miss*, so fragt das nach Programm nach einer maximalen Primzahl $lmax$ und gibt alle Primzahlen kleiner $lmax$ aus, für die keine Relation in der Datei enthalten ist.

Mit **RELmove** *relfile* werden alle Relationen aufgelistet, die in der Datei enthalten sind. Gibt der Benutzer anschließend einen Modul l ein, so wird die Relation dieses Moduls aus der Liste entfernt. Wird der Modul l wieder eingegeben, wird die Relation wieder in die Liste aufgenommen. Nach jeder Eingabe eines Moduls zeigt das Programm dem Benutzer an wie groß das Produkt aller Moduln ist, die sich in der Liste befinden und wie groß die Anzahl der Kombinationen ist. Durch Eingabe eines nicht positiven Wertes kann das Programm verlassen werden; dabei werden alle Relationen, die aus der Liste genommen wurden, in einer Sicherheitsdatei abgespeichert.

Möchte man nur bestimmte Relationen von einer Datei in eine andere kopieren, so kann man das Programm **lcp** *moduls d1 d2* verwenden. Der Parameter *moduls* ist eine Liste $l_1\ l_2 \ldots l_n$ von Moduln, deren Relationen in der Datei *d1* gesucht und dann an die Datei *d2* angehängt werden.

Kapitel 7

Ausblick

Wir haben in den vorangegangenen Kapiteln schon einige Teile der Implementierung angesprochen, die noch erweiterbar sind; wir fassen diese Punkte nochmal zusammen und fügen Ergänzungen hinzu.

Zur Eigenwertsuche mit rationalen Funktionen können wir einen vollständigen Babystep-Giantstep-Algorithmus verwenden, indem wir zusätzlich x^p modulo f_C berechnen und eine Tabelle aufbauen, in der die entsprechenden Punktvielfachen gesucht werden (vgl. Abschnitt 4.2.2).

Nach einer Idee von Francois Morain und Jean-Marc Couveignes ([CoMo94], [Mü94, Kap. 8]) ist es möglich, die Spur c des Frobeniusendomorphismus nicht nur modulo der ungeraden Primzahl l, sondern auch modulo l^i, $i \geq 1$, insbesondere l^2, zu berechnen. Wir haben dieses Verfahren schon implementiert, aber an dieser Stelle sind noch einige Verbesserungen möglich.

Wir haben in Abschnitt 6.3 die Verteilung des Atkin-Algorithmus mit LiPS beschrieben und die entstehenden Probleme mit der Handhabung der modularen Gleichungen angesprochen. Die dort beschriebene Strategie zur Lösung dieser Probleme ist noch nicht implementiert. Weiterhin ist es möglich unter Verwendung von LiPS die Babystep-Giantstep-Phase zur Bestimmung der Gruppenordnung (vgl. Kapitel 3) zu verteilen. Ebenso kann man den Babystep-Giantstep-Algorithmus zur Eigenwertsuche (Abschnitt 4.2.2) parallelisieren, indem man zuerst auf einem Rechner die Tabelle aufbaut und anschließend die Giantsteps (in unserer Situation die Babysteps) verteilt berechnet.

Literaturverzeichnis

[At88] A.O.L. Atkin: *The number of points on an elliptic curve modulo a prime*, Manuskript, veröffentlicht über das Number Theory Net, 1988

[At92] A.O.L. Atkin: *The number of points on an elliptic curve modulo a prime (ii)*, Manuskript, veröffentlicht über das Number Theory Net, 1992

[Be93] Franz-Dieter Berger: *ECM - Faktorisieren mit elliptischen Kurven*, Universität des Saarlandes, Diplomarbeit, 1993

[BuMü91] Johannes Buchmann, Volker Müller: *Computing the Number of Points of Elliptic Curves over Finite Fields*, Proceedings of ISSAC 91, ACM Press, 1991, Seite 179-82

[BuMü92] Johannes Buchmann, Volker Müller: *Primality testing*, Informatik Fachbericht 2/92, Universität des Saarlandes, Saarbrücken, 1992

[Co93] Henri Cohen: *A Course in Computational Algebraic Number Theory*, Springer-Verlag, 1993

[CoMo94] Jean-Marc Couveignes and Francois Morain: *Schoof's algorithm and isogeny cycles*, Proceedings of ANTS I, 1994

[El] Noam Elkies: *Explicit Isogenies*, Vorabdruck

[GaSh92] Joachim von zur Gathen, Victor Shoup: *Computing Frobenius maps and factoring polynomials*, Computational Complexity, 2, 187-224 (1992), Birkhäuser Verlag.

[Ko86] Neal Koblitz: *Elliptic curve cryptosystems*, Math. Comp. 48, 1987, Seite 203-209

[La78] Serge Lang: *Elliptic Curves: Diophantine Analysis*, Springer-Verlag, 1982

[Le94] Frank J. Lehmann: *Implementierung von Algorithmen zur Berechnung modularer Polynome und deren Anwendung im Algorithmus von Atkin*, Universität des Saarlandes, Diplomarbeit, 1994

[LMMS94] F. Lehmann, M. Maurer, V. Müller, V. Shoup: *Counting the Number of Points on Elliptic Curves over Finite Fields of Characteristic Greater than Three*, Proceedings of ANTS I, 1994

[Me84] Kurt Mehlhorn: *Data Structures and Algorithms 1: Sorting and Searching*, Springer-Verlag, 1984

[Mey1] Kurt Meyberg: *Algebra, Teil 1*, Carl Hanser Verlag, 1980

[Mey2] Kurt Meyberg: *Algebra, Teil 2*, Carl Hanser Verlag, 1976

[Mi86] Victor Miller: *Uses of elliptic curves in cryptography*, Advances in Cryptology: Proceedings of Crypto '85, Lecture Notes in Computer Science 218, 1986, Springer-Verlag, Seite 417-426

[MoOl90] F. Morain, J. Olivos: *Speeding up the Computations on an Elliptic Curve using Addition-Subtraction Chains*, Thèse, L'Université Claude Bernard Lyon I, 1990

[Mü91] Volker Müller: *Die Berechnung der Punktanzahl von elliptischen Kurven über endlichen Primkörpern*, Universität des Saarlandes, Diplomarbeit, 1991

[Mü94] Volker Müller: *Ein Algorithmus zur Bestimmung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakeristik größer drei*, Universität des Saarlandes, Dissertation, voraussichtliche Veröffentlichung: 1994

[PaZa94] Thomas Papanikolaou, Jörg Zayer: ALGO.STY - *ein TEX - Style für Algorithmen*, Lehrstuhl Prof. Buchmann, Universität des Saarlandes, 1994

[PoHe78] S.C. Pohlig, M.E. Hellman: *An Improved Algorithm for Computing Logarithms in GF(p) and Its Cryptographic Significance*, IEEE Transactions on Information Theory 24, 1978, Seite 106-111

[Li93] Ralf Roth, Thomas Setz: *LiPS, a system for distributed processing on workstations*, University of Saarland, 1993

[Sc85] René Schoof: *Elliptic Curves over Finite Fields and the Computation of Square Roots mod p*, Math. Comp. 44, 1985, Seite 483 - 494

[Sh70] Daniel Shanks: *Class Number, A Theory of Factorization and Genera, Proceedings Symposium Pure Mathematics*, vol. 20, American Mathematical Society, 1970, Seite 415-440

[Sh94_1] Victor Shoup: *Practical computations with large polynomials over finite fields*, in Vorbereitung (1994)

[Sh94_2] Victor Shoup: *Fast Construction Of Irreduzible Polynomials Over Finite Fields*, Journal Of Symbolic Computation, in Vorbereitung

[Si86] Joseph Silverman: *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics 106, Springer-Verlag, 1986