Ludwig-Maximilians-Universität München
Fakultät für Mathematik, Informatik und Statistik
Mathematisches Institut

Diplomarbeit

Beschreibung und Implementierung eines Algorithmus zur Punktezählung elliptischer Kurven über Körpern $\mathbb{F}_q$ mit kleiner ungerader Charakteristik p nach Satoh

Alexander Niske

20. Oktober 2008

Betreuer: Prof. Dr. Otto Forster (LMU München),
Dipl.-Math. Anton Kargl (Siemens AG)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Einleitung

> Elliptic curves are everywhere –
> don't leave home without one!
>
> *(Joseph H. Silverman)*

Elliptische Kurven begegnen uns in vielen Teilgebieten der Mathematik (z. B. in der Funktionentheorie und der Zahlentheorie) und sind seit Jahrhunderten ein Gegenstand der Forschung. Stellvertretend für ihre Bedeutung in der näheren Vergangenheit erwähnen wir Andrew Wiles' Beweis[1] der *Fermatschen Vermutung*, Hendrik Lenstras Faktorisierungsverfahren[2] und den Goldwasser-Kilian Primzahltest[3]. Darüberhinaus verwendet man elliptische Kurven heutzutage in der Kryptographie (ECC – *Elliptic Curve Cryptography*). Warum dies überhaupt möglich ist und welche Vorteile u. a. kurvenbasierte Verschlüsselung besitzt, wird im Folgenden kompakt erläutert. Insbesondere wird der Zusammenhang mit dem in dieser Arbeit vorgestellten Algorithmus von Satoh deutlich werden.

Ein sicherer Austausch von (digitalen) Daten und eine vertrauliche Kommunikation wird in unserer Informationsgesellschaft immer wichtiger. In Zeiten der globalen Vernetzung spielt der Schutz von Informationen (d. h. Vertraulichkeit, Integrität, Authentizität und Verbindlichkeit) eine, in der Geschichte einmalige, herausragende Rolle.

Die Verwendung von elliptischen Kurven in der Kryptographie wurde in den 80er Jahren sowohl von Neal Koblitz[4] als auch von Victor S. Miller[5] erstmals vorgeschlagen. Auf einer elliptischen Kurve (über dem endlichen Körper $\mathbb{F}_q$) lässt sich die Struktur einer abelschen Gruppe einführen, in der das sog. diskrete Logarithmenproblem (DLP – *Discrete Logarithm Problem*) schwer lösbar ist:

> Sei $(G, +)$ eine Gruppe und $P, Q \in G$ mit $P \in \langle Q \rangle$ gegeben.
> Gesucht: $k \in \{0, 1, \ldots, \mathrm{ord}_G(Q) - 1\}$, so dass $P = kQ$.

Vereinfacht gesagt, hat diese Aufgabenstellung kryptographische Anwendungen wie Schlüsselaustausch (*Diffie-Hellman Key Exchange*), Verschlüsselung (*ElGamal Encryption*), digitale Signaturen und Hashfunktionen (siehe [BSS99, I.1]).

[1] [Wil95]
[2] [Len87]
[3] [GK99]
[4] [Kob87]
[5] [Mil86]

Es ist klar, dass die Stärke des DLP unmittelbar von $\#G$ abhängt[6]. Inzwischen hat man eine ganze Reihe von Algorithmen entwickelt, die das DLP in einer beliebigen endlichen Gruppe schneller als die Brute-Force-Methode lösen. Ohne weitere Informationen über G sind heutzutage Pollards ρ- und λ-Methoden (die ungefähr $\sqrt{\#G}$ Schritte benötigen) die besten Lösungsstrategien (siehe [Was08, 5.2.2]). Für die meisten elliptischen Kurven hat man bisher ebenfalls *keine* effizienteren Verfahren gefunden, d. h. die bekannten Methoden um das diskrete Logarithmenproblem auf elliptischen Kurven (ECDLP – *Elliptic Curve Discrete Logarithm Problem*) zu lösen, haben eine in $\ln q$ *exponentielle* Komplexität (vgl. [BSS99, I.3]). Dies führt uns unmittelbar zu den Vorteilen von ECC im Vergleich zu RSA[7].

RSA beruht auf der Schwierigkeit der Faktorisierung großer ganzer Zahlen. Allerdings kennt man mittlerweile *sub-exponentielle* Algorithmen, die dieses Problem lösen. Es ist also „einfacher“ eine große Zahl zu faktorisieren, als diskrete Logarithmen auf elliptischen Kurven zu berechnen. Konkret bedeutet dies, dass man für ECC viel kleinere Schlüssellängen im Vergleich zu RSA benötigt, um ein vergleichbares Sicherheitsniveau zu erreichen (siehe Abbildung 1; die mathematischen Überlegungen hierzu findet man in [BSS99, I.3]). Daraus ergeben sich natürlich weitere Vorteile, wie eine höhere Rechengeschwindigkeit (kryptographischer Verfahren) und geringere Systemanforderungen. Dies macht ECC insbesondere attraktiv für Geräte mit beschränkten Hardwarevoraussetzungen wie z. B. Smartcards.

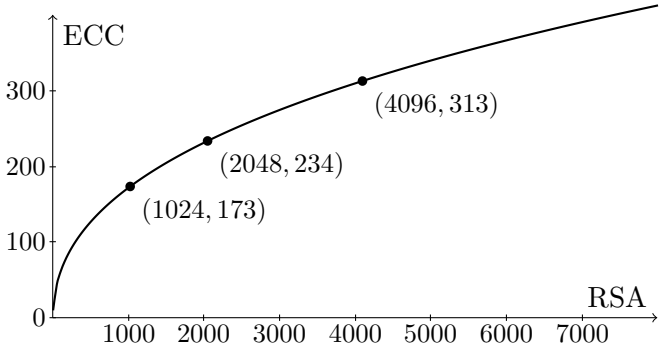


Abbildung 1: RSA vs. ECC (Schlüssellängen in Bit für vergleichbare Sicherheit)

Die kryptographische Eignung einer elliptischen Kurve hängt vor allem von der Gruppenordnung $\#G$ ab. Deshalb ist es in der Praxis von entscheidender Bedeutung diese (effizient) berechnen[8] und damit entscheiden zu können, ob eine gegebene Kurve *stark* ist.

[6] Wir nehmen an, dass $G = \langle Q \rangle$.
[7] [RSA78]
[8] Natürlich ist dieser Aspekt auch mathematisch gesehen von Interesse.

Der erste Algorithmus mit polynomieller Laufzeit, der die Gruppenordnung einer elliptischen Kurve über einem endlichen Körper bestimmt, wurde 1985 von René **S**choof beschrieben[9] (mit späteren Verbesserungen von Noam **E**lkies und Arthur O. L. **A**tkin: SEA-Algorithmus[10]). Dieser berechnet die *Spur des Frobenius-Endomorphismus* (eine ganze Zahl, die für jede Kurve charakteristisch ist und aus der sich direkt die Ordnung berechnen lässt) modulo vieler Primzahlen und konstruiert den tatsächlichen Wert mit Hilfe des *chinesischen Restsatz*[11].

Ende 1999 veröffentlichte Takakazu Satoh einen völlig neuen Ansatz: sein Algorithmus berechnet die Spur modulo einer ausreichend großen Potenz von p (der Charakteristik von $\mathbb{F}_q$). Für „kleine“ Primzahlen p ist dieses Verfahren deutlich effizienter als der SEA-Algorithmus. Inzwischen hat man Satohs *p-adischen Punktezählalgorithmus* auf unterschiedliche Weise weiterentwickelt und optimiert.

Ziel dieser Arbeit ist eine detaillierte Beschreibung und Erklärung von Satohs Algorithmus sowie eine anschließende praktische Umsetzung. Dazu führe ich in *Kapitel 1* elliptische Kurven allgemein ein und formuliere eine Reihe von wichtigen Aussagen und Eigenschaften, auf die ich in der gesamten Arbeit zurückgreife. In *Kapitel 2* stelle ich den Körper der p-adischen Zahlen sowie endliche unverzweigte Erweiterungen von $\mathbb{Q}_p$ vor und lege damit den Grundstein für Satohs Algorithmus. Schließlich wird in *Kapitel 3* die Berechnung der Spur des gelifteten Frobenius-Endomorphismus $\phi_q^{\uparrow} \in \mathrm{End}(E^{\uparrow})$ vorgestellt. Dabei verwende ich die formale Gruppe von $E^{\uparrow}$. Da $\mathrm{char}\, \mathbb{Q}_q = 0$, kann man $\mathrm{Tr}(\phi_q^{\uparrow}) \in \mathbb{Z}$ exakt (und nicht nur modulo p) bestimmen und erhält die gesuchte Gruppenordnung $\#E(\mathbb{F}_q)$ unter Verwendung der Gleichungen $\mathrm{Tr}(\phi_q) = \mathrm{Tr}(\phi_q^{\uparrow})$ und

$$\#E(\mathbb{F}_q) = q + 1 - \mathrm{Tr}(\phi_q).$$

Abschließend wird in *Kapitel 4* die Implementierung von Satohs Algorithmus und ein damit erstelltes Programm zur Suche nach elliptischen Kurven $E/\mathbb{F}_q$ mit primer Gruppenordnung $\#E(\mathbb{F}_q)$ präsentiert.

An dieser Stelle möchte ich allen, die mir bei der Entstehung dieser Diplomarbeit geholfen haben, ganz herzlich danken.

Ich danke vor allem Herrn Prof. Dr. Otto Forster, der mir dieses faszinierende Thema anvertraute und den Kontakt zur Siemens AG herstellte. Seine umfassende Unterstützung und große Hilfsbereitschaft ermutigte mich stets.

Ich danke ganz besonders Herrn Anton Kargl (Mitarbeiter der Siemens AG) für seine hervorragende Betreuung und Hilfsbereitschaft in den letzten Monaten sowie seine

[9] [Sch85]
[10] [CF06, Algorithm 17.25]
[11] [For96, Satz 6.6]

wertvollen Hinweise und Anregungen für meine Arbeit. Seine Begeisterung für dieses Thema und seine fortwährende Unterstützung und Förderung haben mich stets motiviert. Zudem danke ich Herrn Dr. Bernd Meyer, der sich immer für die Fortschritte meiner Arbeit interessierte, für meine Fragen stets ein offenes Ohr hatte und mir in Diskussionen hilfreiche Tipps für meine Implementierung gab.

Für das angenehme Arbeitsklima während meiner neunmonatigen Tätigkeit danke ich allen Mitarbeitern des Kompetenzfeldes Kryptographie der Siemens AG (Abteilung CT IC 3).

Ebenso danke ich Herrn Prof. Takakazu Satoh für seine Hilfsbereitschaft und die freundliche und ausführliche Beantwortung meiner Fragen.

Des Weiteren danke ich Veronika Ertl, Sebastian Queißer und Thomas Weber herzlich für ihre Kommentare und Hilfe. Außerdem danke ich ganz besonders Hanna Roth für ihre Unterstützung und Aufmunterung sowie ihren Rückhalt während der Entstehung dieser Diplomarbeit.

Zum Schluss danke ich meiner Familie, die mich während meines gesamten Studiums unterstützt hat.

1 Elliptische Kurven

Wir verwenden im Folgenden einige Grundbegriffe und Definitionen aus der algebraischen Geometrie. Einen Überblick findet man in Anhang A.

1.1 Weierstraß-Gleichungen

Definition 1.1. Sei K ein Körper. Eine *elliptische Kurve über* K ist eine glatte projektive Kurve E vom Grad 3 über K, die durch eine Gleichung der Form

$$Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 =: w(X, Y, Z) = 0$$

gegeben ist. Wir schreiben auch E/K um zu verdeutlichen, dass $a_1, a_2, a_3, a_4, a_6 \in K$.

Aus dieser Definition ergibt sich unmittelbar

Folgerung 1.2. *Sei E/K eine elliptische Kurve. Dann hat E genau einen Punkt im Unendlichen, nämlich $\mathcal{O} \coloneqq (0 : 1 : 0)$.*

Beweis. Um die Punkte im Unendlichen zu finden, setzen wir in der Kurvengleichung $Z = 0$ und erhalten $-X^3 = 0$. Also ist $(0 : 1 : 0)$ der einzige unendlich-ferne Punkt der Kurve. □

In den meisten Fällen genügt es deshalb die Gleichung des affinen Teils der elliptischen Kurve zu betrachten, die sog. (lange) *Weierstraß-Gleichung*:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{1.1}$$

Definition 1.3. Sei E/K eine elliptische Kurve. Für jeden Erweiterungskörper $L \supset K$ ist die *Menge der L-rationalen Punkte von E* gegeben durch

$$E(L) \coloneqq \{P \in \mathbb{P}_2(L) : \ w(P) = 0\}.$$

Wir verwenden allerdings fast immer die Einbettung

$$\mathbb{A}_2(L) \ni (x, y) \mapsto (x : y : 1) \in \mathbb{P}_2(L),$$

und identifizieren $E(L)$ mit der Menge

$$\{(x, y) \in \mathbb{A}_2(L) : \ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}.$$

Statt $E(\overline{K})$ schreibt man oft nur E. Folgende Größen, die von den Koeffizienten der elliptischen Kurve abhängen, sind allgemein gebräuchlich:

$$\begin{gathered} b_2 \coloneqq a_1^2 + 4a_2, \quad b_4 \coloneqq a_1a_3 + 2a_4, \quad b_6 \coloneqq a_3^2 + 4a_6, \\ b_8 \coloneqq a_1^2a_6 - a_1a_3a_4 + 4a_2a_6 + a_2a_3^2 - a_4^2, \\ c_4 \coloneqq b_2^2 - 24b_4, \quad c_6 \coloneqq -b_2^3 + 36b_2b_4 - 216b_6, \\ \Delta \coloneqq -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6, \quad j \coloneqq c_4^3/\Delta. \end{gathered} \tag{1.2}$$

Δ nennt man die *Diskriminante* und j die *j-Invariante* der Kurve. Es gilt

$$4b_8 = b_2b_6 - b_4^2 \quad \text{und} \quad 1728\Delta = c_4^3 - c_6^2.$$

Um die Abhängigkeit von der gegebenen Kurve zu betonen, schreiben wir auch $\Delta(E)$ bzw. $j(E)$. Es gilt stets $\Delta(E) \neq 0$.

Lemma 1.4. *Eine Gleichung der Form* (1.1) *definiert genau dann eine elliptische Kurve, wenn die Diskriminante* Δ *nicht verschwindet.*

Beweis. Wir behandeln hier nur den Spezialfall

$$E : \ y^2 = g(x) \coloneqq x^3 + ax + b, \quad a, b \in K, \quad \operatorname{char} K \neq 2, 3.$$

Die allgemeine Aussage wird in [Sil86, III., Proposition 1.4 (a)] bewiesen.

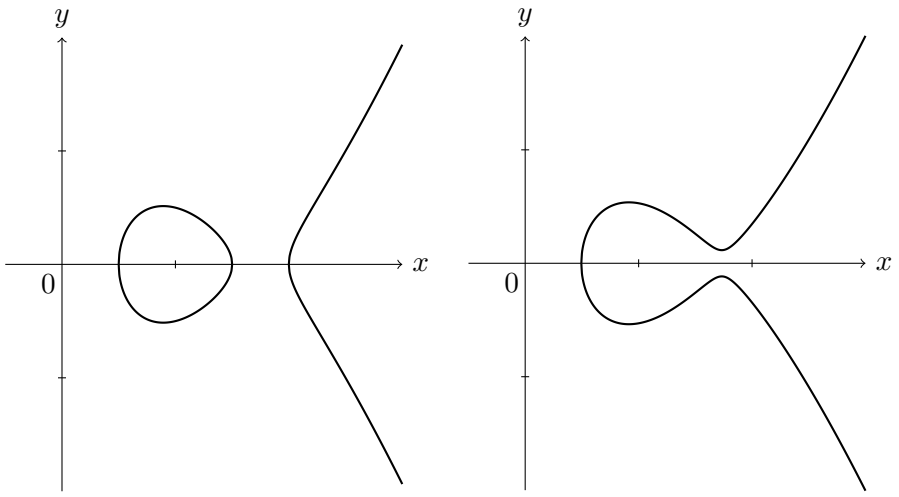
Man rechnet leicht nach, dass E in $\mathcal{O}$ glatt ist. Sei zunächst E auch im Endlichen singularitätenfrei, d. h. $(-g'(\xi), 2\eta) \neq (0, 0)$ für alle $(\xi, \eta) \in E(\overline{K}) \setminus \{\mathcal{O}\}$. Wir schreiben

$$g(x) = (x - x_1)(x - x_2)(x - x_3), \quad x_1, x_2, x_3 \in \overline{K}$$

und verifizieren (vgl. [Was08, 2.1]), dass

$$((x_1 - x_2)(x_1 - x_3)(x_2 - x_3))^2 = -(4a^3 + 27b^2) = \Delta/16.$$

Angenommen $\Delta = 0$. Dann hat $g(x)$ offenbar eine mehrfache Nullstelle $\lambda \in \overline{K}$ und nach [Bos06, 2.6, Satz 2] gilt $g'(\lambda) = 0$. Also ist $(\lambda, 0) \in E(\overline{K})$ ein singulärer Punkt der Kurve. Widerspruch! Die andere Richtung zeigt man analog. □



(a) $y^2 = x^3 - 4x^2 + 4{,}75x - 1{,}5$ (b) $y^2 = x^3 - 3{,}97x^2 + 4{,}75x - 1{,}5$

Abbildung 2: Elliptische Kurven über $\mathbb{R}$

Eine Vorstellung von elliptischen Kurven gibt Abbildung 2 – wobei der unendlich-ferne Punkt in der affinen Ebene nicht sichtbar ist.

Wir möchten die Gleichung (1.1) vereinfachen. Zunächst definieren wir, welche Abbildungen die Struktur einer elliptischen Kurve (im Sinne der algebraischen Geometrie) erhalten. Dieses Vorgehen ist in der Literatur üblich, siehe beispielsweise [BSS99, III.1] und [Eng99, Definition 2.24]. Die folgende Definition kann man aber auch als Aussage verstehen (vgl. [Hus04, 3., (2.8)]).

Definition 1.5. Seien E' und E elliptische Kurven über $K \subset L$. Man nennt E' und E *isomorph über* L, wenn es eine Abbildung $\phi :\ E' \to E$,

$$(X : Y : Z) \mapsto (u^2X + rZ : u^2sX + u^3Y + tZ : Z) \ \text{(„admissible change of variables“)}$$

gibt, wobei $u \in L^*$ und $r, s, t \in L$.

Die Umkehrabbildung $\phi^{-1} :\ E \to E'$,

$$(X : Y : Z) \mapsto (u^{-2}X + (-u^{-2}r)Z : u^{-2}(-u^{-1}s)X + u^{-3}Y + (u^{-3}rs - u^{-3}t)Z : Z)$$

hat (natürlich) die gleiche Gestalt wie ϕ. Zwischen den Koeffizienten von E' und E

besteht folgender Zusammenhang (vgl. [Sil86, Table 1.2]):

$$\begin{aligned}
ua_1' &= a_1 + 2s,\\
u^2a_2' &= a_2 - sa_1 + 3r - s^2,\\
u^3a_3' &= a_3 + ra_1 + 2t,\\
u^4a_4' &= a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st,\\
u^6a_6' &= a_6 + ra_4 + r^2a_2 + r^3 - ta_3 - t^2 - rta_1,
\end{aligned}$$

$$\begin{aligned}
u^2b_2' &= b_2 + 12r,\\
u^4b_4' &= b_4 + rb_2 + 6r^2,\\
u^6b_6' &= b_6 + 2rb_4 + r^2b_2 + 4r^3,\\
u^8b_8' &= b_8 + 3rb_6 + 3r^2b_4 + r^3b_2 + 3r^4,
\end{aligned}$$

$$\begin{aligned}
u^4c_4' &= c_4,\\
u^6c_6' &= c_6,\\
u^{12}\Delta' &= \Delta,\\
j' &= j.
\end{aligned}$$

Isomorphe Kurven haben also dieselbe j-Invariante. Umgekehrt sind zwei elliptische Kurven mit identischer j-Invariante isomorph über $\overline{K}$ (siehe [Sil86, III., Proposition 1.4 (b)]). Einen Spezialfall beweisen wir in Satz 1.7.

Lemma 1.6. *Sei E'/K eine elliptische Kurve, wobei* $\operatorname{char} K \neq 2, 3$. *Dann sind E' und*

$$E:\ y^2 = x^3 + \left(\frac{c_4'}{-48}\right)x + \left(\frac{c_6'}{-864}\right)$$

isomorph über K.

Beweis. Wir definieren $u \coloneqq 1$, $r \coloneqq b_2'/12$, $s \coloneqq a_1'/2$ und $t \coloneqq a_3'/2$. Man rechnet nach, dass E' und E isomorph sind. □

Die vorliegende Arbeit wird sich im Folgenden ausschließlich mit elliptischen Kurven über Körpern der Charakteristik ungleich 2 und 3 beschäftigen. Wir arbeiten deshalb nur noch mit elliptischen Kurven, die durch eine sog. *kurze Weierstraß-Gleichung*

$$y^2 = x^3 + ax + b \tag{1.3}$$

gegeben sind. (1.2) vereinfacht sich dann zu:

$$b_2 = 0, \quad b_4 = 2a, \quad b_6 = 4b, \quad b_8 = -a^2, \quad c_4 = -48a, \quad c_6 = -864b,$$
$$\Delta = -16(4a^3 + 27b^2) = \frac{c_4^3 - c_6^2}{1728}, \quad j = \frac{1728 \cdot 4a^3}{4a^3 + 27b^2}.$$

Satz und Definition 1.7. *Seien*

$$E' : \ y^2 = x^3 + a'x + b' \quad \textit{und} \quad E : \ y^2 = x^3 + ax + b$$

elliptische Kurven über K, wobei $\operatorname{char} K \neq 2, 3$*. Falls $j' = j$, so sind E' und E isomorph über $\overline{K}$ und es existiert $0 \neq u \in \overline{K}$, so dass*

$$(a, b) = (u^4 a', u^6 b').$$

Wir nennen E einen quadratischen Twist *von E', wenn $u^2 \in K$, aber $u \notin K$.*

Beweis. Falls $a' = 0$, so folgt $j' = 0$ und damit auch $a = 0$. Da $\Delta', \Delta \neq 0$, gilt $b', b \neq 0$ und wir wählen $u \in \overline{K}$ mit $b = u^6 b'$.

Falls $a' \neq 0$, so gilt $j' \neq 0$ und damit $a \neq 0$. Wir wählen $u \in \overline{K}$ mit $a = u^4 a'$. Es gilt

$$\frac{4a^3}{4a^3 + 27b^2} = \frac{j}{1728} = \frac{j'}{1728} = \frac{4a'^3}{4a'^3 + 27b'^2} = \frac{4u^{-12}a^3}{4u^{-12}a^3 + 27b'^2} = \frac{4a^3}{4a^3 + 27u^{12}b'^2}$$

und daraus folgt $b^2 = (u^6 b')^2$, also $b = \pm u^6 b'$. Im Fall $b = u^6 b$ sind wir fertig. Ansonsten definieren wir $\widetilde{u} \coloneqq iu$, wobei $i \in \overline{K}$ eine Lösung der Gleichung $X^2 + 1 = 0$ ist. Dann gilt $a = \widetilde{u}^4 a'$ sowie $b = \widetilde{u}^6 b'$ und die erste Behauptung des Satzes ist bewiesen.

Vermöge

$$\phi : \ E' \to E, \ (X : Y : Z) \mapsto (u^2 X : u^3 Y : Z)$$

sind E' und E isomorph über $\overline{K}$. □

Bemerkung. Falls $j' \notin \{0, 1728\}$, so gilt stets $u^2 \in K$. Andernfalls folgt aus der Definition der j-Invariante, dass $a = a' = 0$ bzw. $b = b' = 0$. Über u^2 kann man dann keine allgemeine Aussage treffen.

Sei K weiterhin ein Körper mit $\operatorname{char} K \neq 2, 3$. Wie man leicht überprüft, gilt für

$$E : \ y^2 = x^3 + \left(\frac{3j}{1728 - j}\right) x + \left(\frac{2j}{1728 - j}\right), \quad j \in K \setminus \{0, 1728\}, \tag{1.4}$$

dass $\Delta(E) \neq 0$ und $j(E) = j$. Zusammen mit $y^2 = x^3 + 1$ (j-Invariante 0) und $y^2 = x^3 + x$ (j-Invariante 1728) erhalten wir also, dass es zu jedem $j \in K$ bis auf Isomorphie (eventuell über $\overline{K}$) genau *eine* elliptische Kurve E/K mit $j(E) = j$ gibt.

1.2 Gruppen-Struktur

Auf einer elliptischen Kurve lässt sich durch eine einfache geometrische Konstruktion die Struktur einer abelschen Gruppe einführen. Diese Arbeit wird darauf nicht detaillierter eingehen. Interessierte Leser finden ausführliche Beschreibungen in [For96], [Was08] und [Sil86]. Als Ergebnis notieren wir

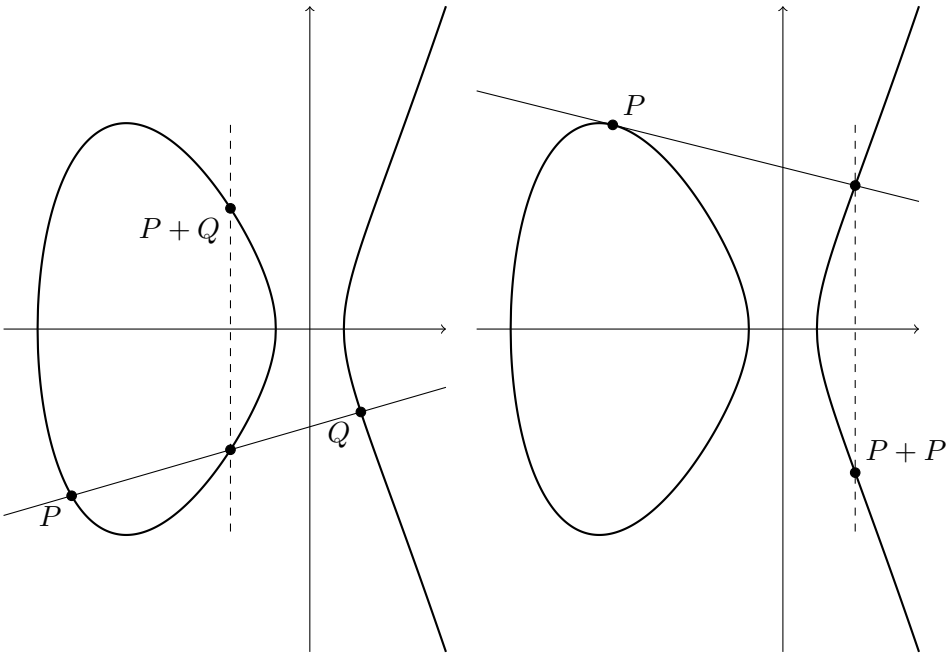
Satz 1.8 (Gruppengesetz)**.** *Sei E/K eine elliptische Kurve. Durch folgende Vorschriften wird E zu einer abelschen Gruppe (und $E(K)$ zu einer Untergruppe von E):*

a) Das Nullelement ist der unendlich-ferne Punkt.

b) Schneidet eine Gerade die Kurve in drei Punkten P, Q, R, so gilt $P + Q + R = \mathcal{O}$.

Beweis. [Sil86, III., Proposition 2.2]. □

Sei $E: \ y^2 = x^3 + 4x^2 - \frac{1}{4}x - 1$ eine elliptische Kurve über $\mathbb{Q}$. Abbildung 3 veranschaulicht das Gruppengesetz auf $E(\mathbb{R})$.



(a) Addition von zwei Punkten (b) Verdoppelung eines Punktes

Abbildung 3: Das Gruppengesetz auf elliptischen Kurven

Für eine konkrete Berechnung oder sogar eine Implementierung der Gruppen-Struktur benötigt man allerdings explizite algebraische Formeln:

Lemma 1.9. *Sei E eine elliptische Kurve über K, die durch eine lange Weierstraß-Gleichung gegeben ist:*

$$E: \ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

Seien weiter $P_1, P_2 \in E(\overline{K}) \setminus \{\mathcal{O}\}$, $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$. Dann gilt

$$-P_1 = (x_1, -y_1 - a_1x_1 - a_3).$$

Ist $P_2 \neq -P_1$ (d. h. $P_1 + P_2 \neq \mathcal{O}$), so definiere

$$(\lambda, \mu) \coloneqq \begin{cases} \left(\dfrac{y_2 - y_1}{x_2 - x_1}, \dfrac{y_1x_2 - y_2x_1}{x_2 - x_1}\right) & \text{falls } x_1 \neq x_2, \\ \left(\dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}, \dfrac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3}\right) & \text{falls } x_1 = x_2. \end{cases}$$

Es gilt $P_1 + P_2 = (x_3, y_3)$, wobei

$$\begin{aligned} x_3 &= \lambda^2 + a_1\lambda - a_2 - x_1 - x_2, \\ y_3 &= -(\lambda + a_1)x_3 - \mu - a_3. \end{aligned}$$

Beweis. [Sil86, III., Group Law Algorithm 2.3]. □

Bemerkung. Für eine effiziente Implementierung von elliptischen Kurven kann es von Vorteil sein, zur projektiven Schreibweise der Punkte zurückzukehren. Weitere Details sind beispielsweise in [BSS99] beschrieben. Eine Einführung in Jacobi- und Edwards-Koordinaten findet man in [Was08].

1.3 Isogenien

Definition 1.10. Seien E und E' elliptische Kurven über K. Eine *Isogenie* zwischen E und E' ist ein Morphismus

$$\phi: \ E \to E' \quad \text{mit} \quad \phi(\mathcal{O}) = \mathcal{O}.$$

E und E' heißen *isogen*, wenn es eine nicht-konstante Isogenie $\phi: \ E \to E'$ gibt. Die Menge aller Isogenien zwischen E und E' wird mit $\mathrm{Hom}(E, E')$ bezeichnet. Man nennt $\mathrm{End}(E) \coloneqq \mathrm{Hom}(E, E)$ *Endomorphismenring* von E. Betrachtet man nur über K definierte Isogenien, so schreibt man auch $\mathrm{Hom}_K(E, E')$ bzw. $\mathrm{End}_K(E)$.

Nach [Sil86, II., Theorem 2.3] ist jede nicht-konstante Isogenie *surjektiv*. $\mathrm{Hom}(E, E')$ wird durch punktweise Addition zu einer Gruppe:

$$(\phi + \psi)(P) \coloneqq \phi(P) + \psi(P). \tag{1.5}$$

Definiert man weiter das Produkt als Komposition von Isogenien, so wird $\mathrm{End}(E)$ zu einem (nicht zwangsläufig kommutativen) *Ring*:

$$\phi \cdot \psi \coloneqq \phi \circ \psi. \tag{1.6}$$

Nach [Sil86, III., Proposition 4.2] ist der Endomorphismenring *nullteilerfrei* und hat die Charakteristik 0. Als weitere wichtige Eigenschaft notieren wir

Satz 1.11. *Sei $\phi \in \mathrm{Hom}(E, E')$ eine Isogenie. Dann ist ϕ ein Gruppenhomomorphismus, d. h. es gilt*

$$\phi(P + Q) = \phi(P) + \phi(Q) \quad \textit{für alle } P, Q \in E.$$

Beweis. [Sil86, III., Theorem 4.8]. □

Beispiel 1.12. Sei $0 \neq m \in \mathbb{Z}$. Dann definiert

$$[m] : \ E \to E, \ P \mapsto \underbrace{Q + \cdots + Q}_{|m|}, \quad \text{wobei } Q \coloneqq \begin{cases} P & \text{falls } m > 0, \\ -P & \text{falls } m < 0 \end{cases}$$

nach [Sil86, III., Proposition 4.2] eine nicht-konstante Isogenie. Die konstante Isogenie, welche jeden Punkt auf $\mathcal{O}$ abbildet, wird mit [0] abgekürzt.

Falls $\mathbb{Z} \subsetneq \mathrm{End}(E)$, so sagt man auch E habe *komplexe Multiplikation*, wobei die ganzen Zahlen in natürlicher Weise eingebettet werden:

$$\mathbb{Z} \ni m \mapsto [m] \in \mathrm{End}(E). \tag{1.7}$$

Wenn keine Verwechslungen zu befürchten sind, so schreiben wir wieder m statt $[m]$.

Um weitere Eigenschaften von Isogenien zu untersuchen, benötigen wir den Begriff des *Funktionenkörpers* einer elliptischen Kurve. Wir übernehmen die Definition von Husemöller[1] (siehe auch Abschnitt 3.2).

Definition 1.13. Sei $E/K : \ y^2 = x^3 + ax + b$ eine elliptische Kurve. Der Funktionenkörper von E über K ist

$$K(E) \coloneqq \mathrm{Quot}(\ K[x, y]/(y^2 - x^3 - ax - b)\ ).$$

[1] [Hus04, 12., (3.2)]

Eine nicht-konstante Isogenie $\phi \in \mathrm{Hom}(E, E')$ induziert eine Inklusion der Funktionenkörper durch

$$\phi^* : \overline{K}(E') \to \overline{K}(E), \ f \mapsto f \circ \phi.$$

Nach [Sil86, II., Theorem 2.4] ist die Körpererweiterung $\phi^*\left(\overline{K}(E')\right) \subset \overline{K}(E)$ endlich. Wir definieren den *Grad* von ϕ als

$$\deg(\phi) \coloneqq \left[\, \overline{K}(E) : \phi^*\left(\overline{K}(E')\right) \,\right]$$

und setzen $\deg(0) \coloneqq 0$. Analog dazu definiert man den *separablen* und *inseparablen Grad* von ϕ:

$$\deg_s(\phi) \coloneqq \left[\, \overline{K}(E) : \phi^*\left(\overline{K}(E')\right) \,\right]_s, \quad \deg_i(\phi) \coloneqq \left[\, \overline{K}(E) : \phi^*\left(\overline{K}(E')\right) \,\right]_i.$$

ϕ heißt *(in)separabel*, wenn die induzierte Körpererweiterung (in)separabel ist, d. h. wenn $\deg_s(\phi) = \deg(\phi)$ ($\deg_i(\phi) = \deg(\phi)$). Wir verweisen auf [Bos06, 3.6] und zitieren folgende Aussage:

Satz 1.14. *Sei $K \subset L$ eine endliche Körpererweiterung. Falls* $\mathrm{char}\, K = 0$*, so gilt*

$$[L : K] = [L : K]_s.$$

Falls $\mathrm{char}\, K = p > 0$*, so existiert ein $r \in \mathbb{N}$, so dass*

$$[L : K] = [L : K]_s\, p^r,$$

d. h. $[L : K]_i = p^r$.

Beweis. [Bos06, 3.6, Satz 8]. □

Satz und Definition 1.15. *Sei $0 \neq \phi \in \mathrm{Hom}(E, E')$. Dann gibt es genau eine Isogenie $\widehat{\phi} : E' \to E$, so dass*

$$\widehat{\phi} \circ \phi = \deg(\phi) \in \mathrm{End}(E).$$

$\widehat{\phi}$ heißt duale Isogenie *von ϕ und man setzt $\widehat{0} \coloneqq 0$. Außerdem gelten folgende Eigenschaften:*

a) $\widehat{\widehat{\phi}} = \phi$ und $\deg(\widehat{\phi}) = \deg(\phi)$.

b) Für $m \in \mathbb{Z}$ gilt $m = \widehat{m}$ und $\deg(m) = m^2$.

c) Sei $\lambda \in \mathrm{Hom}(E', E'')$. Dann gilt $\widehat{\lambda \circ \phi} = \widehat{\phi} \circ \widehat{\lambda}$.

d) Sei $\psi \in \mathrm{Hom}(E, E')$. Dann gilt $\widehat{\phi + \psi} = \widehat{\phi} + \widehat{\psi}$.

Beweis. [Sil86, III., Theorem 6.1 und 6.2]. □

Korollar und Definition 1.16. *Sei $\phi \in \mathrm{End}(E)$ eine Isogenie. Wir definieren die* Spur *von ϕ als*

$$\mathrm{Tr}(\phi) \coloneqq \phi + \widehat{\phi} \in \mathrm{End}(E).$$

Es gilt

$$\mathrm{Tr}(\phi) = 1 + \deg(\phi) - \deg(1 - \phi) \in \mathbb{Z}.$$

Beweis. Unter Verwendung von (1.5), (1.6) und (1.7) gilt nach Satz 1.15:

$$\begin{aligned}\mathrm{End}(E) \ni \deg(1-\phi) = (1-\phi)\widehat{(1-\phi)} = (1-\phi)(1-\widehat{\phi}) = 1 - \widehat{\phi} - \phi + \phi\widehat{\phi} \\ = 1 - (\phi + \widehat{\phi}) + \deg(\phi).\end{aligned}$$ □

Bemerkung. Wir haben die Definition der Spur aus [Hus04, 12., (4.2)] übernommen. Einen anderen Zugang liefert das Studium des sog. *Tate-Moduls* einer elliptischen Kurve. Details dazu finden sich in [Sil86, III., §7 und V., §2]. Im Übrigen folgt aus Korollar 1.16, dass $\widehat{\phi} = \mathrm{Tr}(\phi) - \phi$ über K definiert ist, wenn bereits $\phi \in \mathrm{End}_K(E)$.

Proposition 1.17. *Sei ϕ eine Isogenie mit*

$$\deg(\phi) = 1.$$

Dann ist ϕ ein Isomorphismus.

Beweis. [Sil86, II., Corollary 2.4.1]. □

Hilfreich für das Verständnis ist sicherlich folgender Blickwinkel auf Isogenien. Wir zitieren dabei aus [Was08, 2.9 und 12.2].

Sei $0 \neq \phi \in \mathrm{Hom}_K(E, E')$. Dann gibt es $r_1(X), r_2(X) \in K(X)$, so dass

$$\phi(P) = (r_1(x), r_2(x) \cdot y) \in E'(\overline{K}) \quad \text{für alle } (x,y) = P \in E(\overline{K}) \setminus (\ker \phi).$$

Schreibt man

$$r_1(X) = \frac{p(X)}{q(X)}$$

mit $p(X), q(X) \in K[X]$, $\gcd(p(X), q(X)) = 1$, so gilt

$$q(x) = 0 \iff (x,y) \in (\ker \phi) \setminus \{\mathcal{O}\}.$$

Falls $q(x) \neq 0$, so ist $r_2(x)$ definiert (wobei $(x,y) \in E(\overline{K}) \setminus \{\mathcal{O}\}$). Es gilt

$$\deg(\phi) = \max(\deg(p), \deg(q)).$$

ϕ ist separabel, wenn die formale Ableitung von $r_1(X)$ nicht verschwindet, d. h.

$$\frac{p'q - pq'}{q^2} \neq 0.$$

Beispiel 1.18. Sei $E/K:\ y^2 = x^3+ax+b$ eine elliptische Kurve, wobei $\operatorname{char} K \neq 2, 3$. Für $2 \in \operatorname{End}(E)$ gilt mit obiger Notation (vgl. [Was08, Example 2.5]):

$$r_1(X) = \frac{X^4 - 2aX^2 - 8bX + a^2}{4(X^3 + aX + b)}.$$

Man vergleiche das Ergebnis mit Satz 1.15 b) und Satz 1.24.

1.4 Torsionspunkte

Torsionspunkte sind Punkte auf elliptischen Kurven mit endlicher Ordnung. Formal definieren wir für eine elliptische Kurve E über dem Körper K und $m \in \mathbb{N}$

$$E[m] \coloneqq \ker [m] = \{P \in E(\overline{K}) :\ [m](P) = \mathcal{O}\}$$

als die Menge aller *m-Torsionspunkte*. Statt $E[m] \cap E(K)$ schreibt man auch $E(K)[m]$. Wie man leicht zeigt, ist $E[m]$ eine Untergruppe von $E(\overline{K})$ (ebenso $E(K)[m]$ von $E(K)$).

Satz 1.19. *Seien E und E' elliptische Kurven über K und $0 \neq \phi \in \operatorname{Hom}(E, E')$. Dann gilt für alle $P \in E'(\overline{K})$, dass*

$$\#\phi^{-1}(P) = \deg_s(\phi).$$

Insbesondere gilt $\#\ker\phi = \#\phi^{-1}(\mathcal{O}) = \deg_s(\phi) < \infty$.

Beweis. [Sil86, III., Theorem 4.10]. □

Satz und Definition 1.20. *Sei E/K eine elliptische Kurve und $0 \neq m \in \mathbb{Z}$. Falls* $\operatorname{char} K$ *kein Teiler von m ist, so gilt*

$$E[m] \cong \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}.$$

Falls $\operatorname{char} K = p \neq 0$*, so gilt*

$$\begin{aligned} \textit{entweder} \quad & E[p^e] = \{0\} \quad \textit{für alle } 0 \neq e \in \mathbb{N} \\ \textit{oder} \quad & E[p^e] \cong \mathbb{Z}/p^e\mathbb{Z} \quad \textit{für alle } 0 \neq e \in \mathbb{N}. \end{aligned}$$

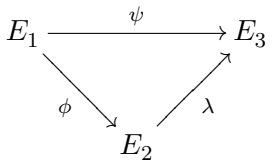
Im ersten Fall heißt E supersingulär*, im zweiten Fall* gewöhnlich.

Beweis. [Sil86, III., Corollary 6.4]. □

Lemma 1.21. *Sei $\phi \in \mathrm{Hom}(E_1, E_2)$ separabel, $0 \neq \psi \in \mathrm{Hom}(E_1, E_3)$ und*

$$\ker \phi \subset \ker \psi.$$

Dann existiert eine eindeutig bestimmte Isogenie $\lambda : E_2 \to E_3$ mit $\psi = \lambda \circ \phi$, so dass das folgende Diagramm kommutiert:



Beweis. [Sil86, III., Corollary 4.11]. □

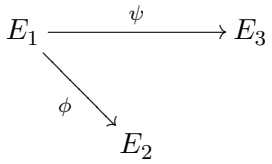
Proposition 1.22. *Sei G eine endliche Untergruppe von E_1. Dann gibt es eine eindeutig bestimmte elliptische Kurve E_2 und eine separable Isogenie $\phi : E_1 \to E_2$, so dass*

$$\ker \phi = G.$$

Sei weiter $\psi \in \mathrm{Hom}(E_1, E_3)$ separabel mit $\ker \psi = G$. Dann sind E_2 und E_3 isomorph.

Beweis. Die erste Aussage wird in [Sil86, III., Proposition 4.12] gezeigt.

Wir betrachten nun also die folgende Situation, wobei $\ker \phi = \ker \psi$:



Nach Lemma 1.21 existiert $\lambda \in \mathrm{Hom}(E_2, E_3)$, so dass

$$\psi = \lambda \circ \phi.$$

Daraus folgt

$$\deg(\psi) = \deg(\lambda) \cdot \deg(\phi).$$

Nach Satz 1.19 gilt somit

$$\#\ker \psi = \deg(\lambda) \cdot \#\ker \phi,$$

d. h. $\deg(\lambda) = 1$. Die Behauptung folgt aus Proposition 1.17. □

Bemerkung. Man bezeichnet E_2 häufig mit E_1/G. Allerdings weist Silverman in [Sil86, III., Remark 4.13.1] darauf hin, dass es a priori keinen Grund gibt, warum die Faktorgruppe E_1/G einer elliptischen Kurve entsprechen sollte. Man vergleiche in diesem Zusammenhang die obigen Aussagen mit dem bekannten Homomorphiesatz[2] und [Bos06, 1.2, Korollar 7], denn wie wir bereits erwähnt haben, ist jede nicht-konstante Isogenie surjektiv. Die Weierstraß-Gleichung für E_2 und die explizite Berechnung von ϕ werden in [Vél71] vorgestellt (siehe auch Satz 3.18). Wir halten noch folgende Aussage fest: Sei E_1 über K definiert und G invariant unter $\mathrm{Gal}(\overline{K}/K)$. Dann sind auch E_2 und ϕ über K definiert (vgl. [Sil86, III., Remark 4.13.2]).

Sei E eine elliptische Kurve über dem endlichen Körper $\mathbb{F}_q$, dann gilt offensichtlich $\#E(\mathbb{F}_q) < \infty$. Nach Lagrange[3] ist $\#E(\mathbb{F}_q)[m]$ für $m \in \mathbb{N}$ ein Teiler von $\#E(\mathbb{F}_q)$.

Wir verlassen nun kurz die Theorie und wenden uns einer praktischen Fragestellung zu. Für kryptographische Anwendungen soll $\#E(\mathbb{F}_q) =: k$ eine große Primzahl sein[4]. Es gilt dann $E(\mathbb{F}_q)[l] = \{\mathcal{O}\}$ für alle Primzahlen $l < k$. Besitzt beispielsweise E einen $\mathbb{F}_q$-rationalen 2-Torsionspunkt ungleich $\mathcal{O}$, so ist $\#E(\mathbb{F}_q)$ durch 2 teilbar. Man braucht also die Gruppenordnung gar nicht mehr zu berechnen – und kann sich der nächsten (zufällig generierten) elliptischen Kurve zuwenden. Insbesondere ist dieses Verfahren effizient, wenn man (im Vergleich zur Bestimmung von $\#E(\mathbb{F}_q)$) schnell entscheiden kann, ob E echte $\mathbb{F}_q$-rationale l-Torsionspunkte hat. Man testet also bis zu einer gewissen Schranke S die Gruppenordnung auf Primteiler und bestimmt $\#E(\mathbb{F}_q)$ erst dann exakt, wenn alle Tests negativ ausfallen.

Mit Hilfe einer einfachen Rechnung überzeugt man sich davon, dass bereits mit einer relativ klein gewählten Schranke viele Zufallskurven ausgesiebt werden können. Wir nehmen dazu an, dass bei einer zufälligen elliptischen Kurve $E/\mathbb{F}_q$ auch $k = \#E(\mathbb{F}_q)$ zufällig ist (siehe Abschnitt 4.2). Die Wahrscheinlichkeit, dass k (mindestens) einen Primteiler $l \leq S < k$ besitzt, ist gerade

$$\mathrm{P}_S := \mathrm{P}\left(\gcd\left(k, \prod_{\substack{l\in\mathbb{P}\\ l\leq S}} l\right) > 1\right) = \sum_{\substack{l\in\mathbb{P}\\ l\leq S}} l^{-1} \prod_{\substack{m\in\mathbb{P}\\ m<l}} (1 - m^{-1})$$

$$= \tfrac{1}{2} + \tfrac{1}{6} + \tfrac{1}{15} + \tfrac{4}{105} + \cdots .$$

Testet man folglich $\#E(\mathbb{F}_q)$ auf Teilbarkeit durch die ersten fünf Primzahlen, so wird man im Durchschnitt ca. 80 Prozent aller zufällig erzeugten elliptischen Kurven unmittelbar verwerfen können – und nur bei den verbleibenden 20 Prozent muss man

[2] [Bos06, 1.2, Satz 6]
[3] [Bos06, 1.2, Korollar 3]
[4] In der Praxis erzeugt man solange Zufallskurven, bis die gewünschte Eigenschaft auftritt.

Tabelle 1: Wahrscheinlichkeiten von Primteilern

S	P_S	$\pi(S)$
2	0,5	1
3	0,666 67	2
5	0,733 33	3
7	0,771 43	4
11	0,792 21	5
⋮	⋮	⋮
53	0,863 91	16
⋮	⋮	⋮
251	0,899 65	54

tatsächlich die Gruppenordnung bestimmen. Diesen Wert entnimmt man Tabelle 1; weitere Details sind in [FGH01] beschrieben.

Es wird nun eine Möglichkeit vorgestellt, Primteiler von $\#E(\mathbb{F}_q)$ zu ermitteln, ohne den Wert exakt zu kennen. Mit elliptischen Kurven über endlichen Körpern werden wir uns im nächsten Abschnitt noch einmal ausführlicher beschäftigen.

Definition 1.23. Sei E/K eine elliptische Kurve der Gestalt (1.3) und $\operatorname{char} K \neq 2, 3$. Für $n \in \mathbb{N}$ definieren wir $f_n(X) \in K[X]$ wie folgt[5]:

$$
\begin{aligned}
f_0(X) &\coloneqq 0,\\
f_1(X) &\coloneqq 1,\\
f_2(X) &\coloneqq 1,\\
f_3(X) &\coloneqq 3X^4 + 6aX^2 + 12bX - a^2,\\
f_4(X) &\coloneqq 2X^6 + 10aX^4 + 40bX^3 - 10a^2X^2 - 8abX - 2a^3 - 16b^2,
\end{aligned}
$$

sowie für $n \geq 5$ und $k \coloneqq \left\lfloor \frac{n}{2} \right\rfloor \in \mathbb{N}$

$$
f_n \coloneqq \begin{cases} f_k(f_{k+2}f_{k-1}^2 - f_{k-2}f_{k+1}^2) & \text{falls } n \text{ gerade,} \\ cd_k - e_k & \text{falls } n \text{ ungerade, } k \text{ gerade,} \\ d_k - ce_k & \text{falls } n \text{ ungerade, } k \text{ ungerade,} \end{cases}
$$

[5] vgl. [CF06, 4.4.5.a]

wobei

$$\begin{aligned} c(X) &\coloneqq 16(X^3 + aX + b)^2, \\ d_k &\coloneqq f_{k+2} f_k^3, \\ e_k &\coloneqq f_{k-1} f_{k+1}^3. \end{aligned}$$

Es gilt $f_n(X) \in \mathbb{Z}[a, b, X]$ für alle $n \in \mathbb{N}$.

Satz 1.24. *Sei K ein Körper*[6] *und*

$$E: \ y^2 = x^3 + ax + b$$

eine elliptische Kurve über K. Für $P \in E(\overline{K})$ und $n \in \mathbb{N}$ gilt

$$[n](P) = \begin{cases} \mathcal{O} & \text{falls } P \in E[n], \\ \left(\frac{\phi_n(P)}{\psi_n^2(P)}, \frac{\omega_n(P)}{\psi_n^3(P)} \right) & \text{falls } P \notin E[n], \end{cases}$$

wobei

$$\begin{aligned} \psi_n(X, Y) &\coloneqq \begin{cases} f_n(X) \cdot 2Y & \text{falls } n \text{ gerade,} \\ f_n(X) & \text{falls } n \text{ ungerade,} \end{cases} \\ \phi_n(X, Y) &\coloneqq X \cdot \psi_n^2(X, Y) - \psi_{n-1}(X, Y) \cdot \psi_{n+1}(X, Y), \\ \omega_n(X, Y) &\coloneqq \frac{\psi_{2n}(X, Y)}{2\psi_n(X, Y)} = (f_{n+2} f_{n-1}^2 - f_{n-2} f_{n+1}^2) \cdot \begin{cases} \frac{1}{2} & \text{falls } n \text{ gerade,} \\ Y & \text{falls } n \text{ ungerade.} \end{cases} \end{aligned}$$

Insbesondere gilt für $n \geq 2$ und $P \in E(\overline{K}) \setminus \{\mathcal{O}\}$, dass

$$P \in E[n] \iff \psi_n(P) = 0. \tag{1.8}$$

Falls $\operatorname{char} K \nmid n$, *so gilt*

$$\chi(n) \coloneqq \deg(f_n) = \begin{cases} (n^2 - 4)/2 & \text{falls } n \text{ gerade,} \\ (n^2 - 1)/2 & \text{falls } n \text{ ungerade.} \end{cases} \tag{1.9}$$

Falls $\operatorname{char} K = p \geq 5$, *so gilt*

$$f_p(X) = \varepsilon f(X)^p,$$

wobei $\varepsilon \in K^*$ und $f(X) \in K[X]$ ein normiertes Polynom vom Grad $(p-1)/2$ ist. Insbesondere gilt

$$\chi(p) = \frac{p(p-1)}{2}.$$

[6] $\operatorname{char} K \neq 2, 3$

Beweis. [Was08, 3.2 und 9.5] und [Mad04, Proposition 2.2.19]. □

Das Polynom ψ_n heißt *n-tes Divisionspolynom*. In unseren Anwendungen wird n stets eine Primzahl sein. Wir verweisen auf Abschnitt 3.3 für weitere Details. Das Bild von P unter $[n]$ berechnet man normalerweise nicht mit obigen Formeln. Allerdings haben wir nun unser angekündigtes Hilfsmittel erhalten, um die Primteiler von $\#E(\mathbb{F}_q)$ zu bestimmen: Besitzt $X^3 + aX + b \in \mathbb{F}_q[X]$ eine Nullstelle $\xi \in \mathbb{F}_q$, so gilt nach (1.8), dass $(\xi, 0) \in E(\mathbb{F}_q)[2]$. Folglich ist 2 ein Teiler von $\#E(\mathbb{F}_q)$. Um zu ermitteln, ob ein Polynom $F(X) \in \mathbb{F}_q[X]$ eine Nullstelle in $\mathbb{F}_q$ besitzt, berechnet man einfach einen größten gemeinsamen Teiler in $\mathbb{F}_q[X]$:

$$\gcd(X^q - X, F(X)) = \gcd((X^q \bmod F(X)) - X, F(X)).$$

Falls $X^q - X$ und $F(X)$ nicht teilerfremd sind, so ist die Existenz (mindestens) einer Nullstelle von $F(X)$ in $\mathbb{F}_q$ gezeigt. Für größere Primzahlen verfahren wir analog, wobei wir aber darauf zu achten haben, dass es zu einer x-Koordinate auch eine geeignete y-Koordinate geben muss: Besitzt beispielsweise $f_3(X) \in \mathbb{F}_q[X]$ eine Nullstelle $\xi \in \mathbb{F}_q$ *und* existiert $\eta \in \mathbb{F}_q$ mit $\eta^2 = \xi^3 + a\xi + b$, so ist (ξ, η) nach (1.8) ein $\mathbb{F}_q$-rationaler 3-Torsionspunkt. Wir betonen noch einmal, dass

$$\deg(\gcd(X^q - X, X^2 - \xi^3 - a\xi - b)) > 0 \iff \exists\, \eta \in \mathbb{F}_q :\ \eta^2 = \xi^3 + a\xi + b.$$

Divisionspolynome spielen auch für Satohs Algorithmus zur Bestimmung von $\#E(\mathbb{F}_{p^n})$ eine wichtige Rolle: Wir berechnen dabei p-te Divisionspolynome nicht nur über $\mathbb{F}_q$, sondern auch über einem Erweiterungskörper der p-adischen Zahlen[7] ($\operatorname{char} \mathbb{Q}_p = 0$). Letztlich ist es die Abhängigkeit $\deg(f_p) = O(p^2)$, die Satohs Algorithmus nur für „kleine“ Charakteristiken (des Grundkörpers $\mathbb{F}_q$) praktikabel macht.

Außer der rekursiven Variante gibt es noch eine weitere Methode um Divisionspolynome zu berechnen. Für ein festes p hat diese den Vorteil, Vorberechnungen durchführen zu können. Wir verwenden die im Folgenden vorgestellte Methode bis $p = 59$ in $\mathbb{Q}_{p^n}$ (und für eine Startnäherung in $\mathbb{F}_{p^n}$ sogar für „alle“ p). Ab $p = 61$ wird nach unseren Testläufen einerseits der Speicheraufwand relativ groß und andererseits die Verarbeitung eines Polynoms mit so hohem Grad zu langsam. Da wir nur $f_p(X) \bmod h(X)$ für ein gewisses Polynom $h(X) \in \mathbb{Q}_{p^n}[X]$ benötigen, verfolgen wir dann eine andere Strategie und berechnen $f_p(X) \bmod h(X)$ rekursiv.

Lemma 1.25 (McKee)**.** *Sei K ein Körper*[8] *und*

$$E :\ y^2 = x^3 + ax + b$$

[7] Mit dem Körper der p-adischen Zahlen beschäftigen wir uns im nächsten Kapitel ausführlich.
[8] $\operatorname{char} K \neq 2, 3$

eine elliptische Kurve über K. Dann gilt für jedes ungerade $n \in \mathbb{N}$, dass

$$f_n(X) = \sum_{t \in \mathbb{Z}} \beta_t(n) X^t,$$

wobei

$$\beta_t(n) \coloneqq \sum_{\substack{(r,s) \in \mathbb{Z} \times \mathbb{Z} \\ d_{r,s} = \chi(n) - t}} \alpha_{r,s}(n) a^r b^s \in \mathbb{Z}[a,b], \qquad d_{r,s} \coloneqq 2r + 3s,$$

$$\alpha_{r,s}(n) \coloneqq \begin{cases} 0 \quad \textit{falls} \;\; r < 0 \;\vee\; s < 0 \;\vee\; d_{r,s} > \chi(n), \\ n \quad \textit{falls} \;\; r = 0 \;\wedge\; s = 0, \\ \big((n^2 + 3 - 2d_{r,s})(n^2 - 6 + 6d_{r,s})\alpha_{r-1,s}(n) \\ -(n^2 + 5 - 2d_{r,s})(3n^2 + 9 - 6d_{r,s})\alpha_{r,s-1}(n) \\ +36(r+1)n^2\alpha_{r+1,s-1}(n) \\ -8(s+1)n^2\alpha_{r-2,s+1}(n)\big) \cdot (12d_{r,s}^2 + 6d_{r,s})^{-1} \quad \textit{sonst.} \end{cases}$$

Es gilt $\alpha_{r,s}(n) \in \mathbb{Z}$ für alle $(r,s) \in \mathbb{Z} \times \mathbb{Z}$ und $\beta_t(n) = 0$ für fast alle $t \in \mathbb{Z}$.

Beweis. [McK94, Main Lemma]. □

Die elliptische Kurve $E/\mathbb{Q}_{p^n}$, für die wir das p-te Divisionspolynom später berechnen wollen, wird in der Form (1.4) gegeben sein. Folglich erhalten wir mit $J \coloneqq j(E)$ und $Q \coloneqq J/(1728 - J)$, dass

$$\beta_t(p) = \sum_{\substack{(r,s) \in \mathbb{Z} \times \mathbb{Z} \\ d_{r,s} = \chi(p) - t}} 3^r 2^s \alpha_{r,s} Q^{r+s} = \sum_{\substack{(r,s) \in \mathbb{Z} \times \mathbb{Z} \\ d_{r,s} = \chi(p) - t}} \widetilde{\alpha}_{r,s} Q^{r+s}.$$

Die $\widetilde{\alpha}_{r,s}$ lassen sich unabhängig von E vorberechnen. Wir verzichten auf weitere Details und vergleichen stattdessen in Tabelle 2 auf der nächsten Seite die Laufzeiten (in Sekunden) der beiden vorgestellten Berechnungswege: Das p-te Divisionspolynom wird zweimal über derselben zufälligen Kurve $E/\mathbb{Q}_{p^n}$ zur Präzision N berechnet. p^n liegt dabei in der für kryptographische Anwendungen realistischen Größenordnung von 2^{160}, wobei n ebenfalls prim sein soll. Alle Berechnungen wurden mit Hilfe von C++/NTL[9] auf einer Intel Core2 Duo @ 2 GHz Architektur ausgeführt. Die vorberechneten $\widetilde{\alpha}_{r,s}$ wurden für jedes Paar (p, N) unkomprimiert in Dateien gespeichert, dessen Größe s (in Kilobyte[10]) man der vorletzten Spalte entnimmt. Man bemerke, dass $t_{\text{mckee}} < t_{\text{rekursiv}}$ für alle $p < 89$.

Tabelle 2: Berechnung von p-ten Divisionspolynomen in $\mathbb{Z}_{p^n}$

p	n	N	$\deg(f_p)$	t_{rekursiv}	t_{mckee}	s	$\pi(p)$
5	71	38	12	0,03	n. m.	0,4	3
7	59	32	24	0,14	0,01	1,3	4
11	47	26	60	0,34	0,02	8,4	5
13	47	26	84	0,44	0,05	18,4	6
17	41	22	144	0,80	0,06	50,5	7
19	41	22	180	1,09	0,09	81,5	8
23	37	20	264	1,64	0,17	168,5	9
29	37	20	420	2,67	0,45	453,1	10
31	37	20	480	3,28	0,53	599,8	11
37	31	17	684	3,16	0,67	1 092,3	12
41	31	17	840	4,22	1,02	1 692,8	13
43	31	17	924	4,47	1,28	2 064,1	14
47	29	16	1 104	4,56	1,64	2 840,5	15
53	29	16	1 404	7,16	2,59	4 738,6	16
59	29	16	1 740	8,97	3,94	7 465,9	17
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
79	29	16	3 120	17,14	14,33	25 617,6	22
83	29	16	3 444	19,64	18,11	31 472,0	23
89	29	16	3 960	20,81	23,22	42 256,6	24

Tabelle 3: Berechnung von p-ten Divisionspolynomen in $\mathbb{F}_{p^n}$

p	n	$\deg(f_p)$	t_{rekursiv}	t_{mckee}	s	$\pi(p)$
83	29	3 403	2,83	0,03	89	23
107	29	5 671	5,00	0,06	201	28
137	23	9 316	6,47	0,08	439	33
163	23	13 203	10,47	0,11	754	38
191	23	18 145	13,52	0,16	1 227	43
223	23	24 753	21,30	0,23	2 001	48
241	23	28 920	22,75	0,28	2 589	53
271	23	36 585	30,19	0,36	3 810	58
293	23	42 778	31,02	0,45	4 892	62

Noch eindrucksvollerere Resultate erhält man in Charakteristik p: Sei

$$E: \; y^2 = x^3 + 3\gamma x + 2\gamma$$

eine zufällige elliptische Kurve über $\mathbb{F}_{p^n}$ ($p \geq 5$) und $f_p(X) \in \mathbb{F}_{p^n}[X]$ das p-te Divisionspolynom von E. In Tabelle 3 auf der vorherigen Seite stellen wir erneut die Laufzeiten der beiden Berechnungsmethoden von f_p gegenüber.

Die Geschwindigkeitsunterschiede sind gravierend[11]! Wir weisen darauf hin, dass die vorberechneten Daten in diesem Fall nur von p abhängen, also *unabhängig* vom Grad der Körpererweiterung $\mathbb{F}_p \subset \mathbb{F}_{p^n}$ verwendet werden können.

Mir ist trotz dieser Ergebnisse keine Anwendung von McKees Methode in der Praxis bekannt. Natürlich kann man die $\beta_t(p)$ auch mit Hilfe der rekursiven Definition des Divisionspolynoms und multivariater Polynomarithmetik (in den zwei Variablen X und Q/γ) berechnen. Dieser Weg ist allerdings sehr ineffizient.

1.5 Elliptische Kurven über endlichen Körpern

Das Ziel dieser Arbeit ist die Beschreibung und Implementierung eines Algorithmus zur Bestimmung der Gruppenordnung $\#E(\mathbb{F}_q)$ einer elliptischen Kurve E über dem endlichen Körper $\mathbb{F}_q$. Erfüllt die Gruppenordnung gewisse Voraussetzungen, so kann man die Kurve in der Kryptographie verwenden.

Um Wiederholungen zu vermeiden, definieren wir für den gesamten Abschnitt die folgenden Rahmenbedingungen: Sei $p \geq 5$ stets eine Primzahl. Den endlichen Körper mit $q := p^n$ ($n \geq 1$) Elementen bezeichnen wir mit $\mathbb{F}_q$ bzw. $\mathbb{F}_{p^n}$.

Die intuitive Bestimmung von $\#E(\mathbb{F}_q)$ haben wir bereits kennengelernt: Besitzt das Element $0 \neq \xi^3 + a\xi + b \in \mathbb{F}_q$ für $\xi \in \mathbb{F}_q$ eine „Wurzel“ η in $\mathbb{F}_q$, so gilt $(\xi, \eta), (\xi, -\eta) \in E(\mathbb{F}_q)$. Durchläuft ξ alle Elemente von $\mathbb{F}_q$ und zählt man dabei die konstruierbaren $\mathbb{F}_q$-rationalen Punkte auf E, erhält man auf triviale Weise $\#E(\mathbb{F}_q)$. Diese Vorgehensweise lässt sich nun formal folgendermaßen zusammenfassen, wobei wir für $a \in \mathbb{F}_q$ das aus der Zahlentheorie bekannte *Legendre-Symbol* verallgemeinern:

$$\left(\frac{a}{\mathbb{F}_q}\right) := \begin{cases} 0 & \text{falls } a = 0, \\ 1 & \text{falls } \exists\, b \in \mathbb{F}_q^* : \; b^2 = a, \\ -1 & \text{sonst.} \end{cases}$$

[9] [Sho]

[10] Wir verwenden dabei die Konvention 1 kB = 1 000 Byte.

[11] Dies liegt insbesondere daran, dass „McKee“ die Darstellung $f_p(X) = \varepsilon f(X)^p$ ausnutzt (was in der rekursiven Berechnung *nicht* möglich ist).

Lemma 1.26. *Sei $E:\ y^2 = x^3 + ax + b$ eine elliptische Kurve über $\mathbb{F}_q$. Dann gilt*

$$\#E(\mathbb{F}_q) = q + 1 + \sum_{\xi \in \mathbb{F}_q} \left(\frac{\xi^3 + a\xi + b}{\mathbb{F}_q} \right).$$

Beweis. Wie man sich überlegt, existieren zu $\xi \in \mathbb{F}_q$ genau $1 + \left(\frac{\xi^3+a\xi+b}{\mathbb{F}_q}\right)$ $\mathbb{F}_q$-rationale Punkte mit x-Koordinate ξ. Zusammen mit dem unendlich-fernen Punkt $\mathcal{O}$ erhält man

$$\#E(\mathbb{F}_q) = 1 + \sum_{\xi \in \mathbb{F}_q} \left(1 + \left(\frac{\xi^3 + a\xi + b}{\mathbb{F}_q} \right) \right) = 1 + q + \sum_{\xi \in \mathbb{F}_q} \left(\frac{\xi^3 + a\xi + b}{\mathbb{F}_q} \right). \qquad \square$$

Mit obigem Verfahren kann man die Gruppenordung (auch unter Zuhilfenahme des Computers) offensichtlich nur für „kleine“ endliche Körper bestimmen.

Wir fixieren nun zwei elliptische Kurven über $\mathbb{F}_q$ mit identischer j-Invariante:

$$E':\ y^2 = x^3 + a'x + b', \qquad E:\ y^2 = x^3 + ax + b, \qquad j' = j.$$

Was für ein Zusammenhang besteht zwischen $\#E'(\mathbb{F}_q)$ und $\#E(\mathbb{F}_q)$?

Wir verwenden die Notation von Satz 1.7. Falls $u^2 \in \mathbb{F}_q$, so können wir $\#E(\mathbb{F}_q)$ aus $\#E'(\mathbb{F}_q)$ berechnen (und umgekehrt). Diese Aussage ist trivial, wenn bereits u in $\mathbb{F}_q$ liegt. E ist dann nämlich isomorph zu E' über $\mathbb{F}_q$ und es gilt

$$\#E'(\mathbb{F}_q) = \#E(\mathbb{F}_q).$$

Den Fall $u \notin \mathbb{F}_q$ behandeln wir in Lemma 1.27. Wir erinnern daran, dass $u^2 \in \mathbb{F}_q$ stets erfüllt ist, wenn $j' \notin \{0, 1728\}$. Gilt jedoch $j' = 0$ ($j' = 1728$), so wissen wir nur, dass

$$b = u^6 b' \quad (a = u^4 a').$$

Deshalb muss im konkreten Fall zunächst geprüft werden, ob ein $v \in \mathbb{F}_q$ existiert, mit

$$v^3 = u^6 = b/b' \quad (v^2 = u^4 = a/a').$$

Lemma 1.27. *Sei E ein quadratischer Twist von E'. Dann gilt*

$$\#E'(\mathbb{F}_q) + \#E(\mathbb{F}_q) = 2(q + 1). \tag{1.10}$$

Beweis. Nach Satz 1.7 existiert $0 \neq u \in \overline{\mathbb{F}}_q$ mit

$$a = u^4 a', \quad b = u^6 b' \quad \text{und} \quad u^2 =: v \in \mathbb{F}_q^*.$$

v ist nach Voraussetzung allerdings kein Quadrat in $\mathbb{F}_q$.

Wir definieren $g(x) \coloneqq x^3 + a'x + b'$, $g_v(x) \coloneqq v^3 g(x/v)$, d. h.

$$E' : \ y^2 = g(x), \quad E : \ y^2 = g_v(x).$$

Man geht nun analog zum Beweis von Lemma 1.26 vor und zählt $\mathbb{F}_q$-rationale Punkte simultan auf beiden Kurven:

Für $\xi \in \mathbb{F}_q$ gilt $g_v(\xi) = 0 \iff g(\xi/v) = 0$ und wir erhalten auf beiden Kurven je *einen* Punkt, nämlich $(\xi, 0) \in E(\mathbb{F}_q)$ und $(\xi/v, 0) \in E'(\mathbb{F}_q)$. Falls $0 \neq g_v(\xi)$ ein Quadrat in $\mathbb{F}_q$ ist, so ist $g(\xi/v)$ kein Quadrat in $\mathbb{F}_q$ und man erhält *zwei* Punkte auf $E(\mathbb{F}_q)$ und *keinen* auf $E'(\mathbb{F}_q)$. Zur Begründung überlegt man sich folgendes: Angenommen $g_v(\xi) = c^2$ und $g(\xi/v) = d^2$ (mit $c, d \in \mathbb{F}_q$). Dann gilt $v^3 = (c/d)^2$. Daraus folgt $v = (c/dv)^2$. Widerspruch zur Voraussetzung, dass v kein Quadrat in $\mathbb{F}_q$ ist! Analog ist $g_v(\xi)$ kein Quadrat in $\mathbb{F}_q$, wenn $g(\xi/v)$ ein Quadrat in $\mathbb{F}_q$ ist.

Durchläuft $\xi \in \mathbb{F}_q$ alle q Körperelemente, erhält man also stets *zwei* $\mathbb{F}_q$-rationale Punkte, die zur Summe $\#E'(\mathbb{F}_q) + \#E(\mathbb{F}_q)$ beitragen. Zusammen mit dem unendlichfernen Punkt (gezählt für beide Kurven), folgt die behauptete Identität (1.10). □

Bekanntlich definiert die Abbildung

$$\sigma : \ \mathbb{F}_{p^n} \to \mathbb{F}_{p^n}, \quad a \mapsto a^p$$

einen Körperisomorphismus, den sog. *Frobenius-Automorphismus* von $\mathbb{F}_{p^n}$. Diese Abbildung lässt sich auf elliptische Kurven übertragen.

Definition 1.28. Sei $E : \ y^2 = x^3 + ax + b$ eine elliptische Kurve über $\mathbb{F}_q$. Für $k \in \mathbb{N}$ definieren wir

$$E^{(k)} : \ y^2 = x^3 + \sigma^k(a)x + \sigma^k(b).$$

Dann ist $E^{(k)}$ eine elliptische Kurve, denn $0 \neq \sigma^k(\Delta(E)) = \Delta(E^{(k)})$. Offenbar gilt $(E^{(k)})^{(l)} = E^{(k+l)}$ für $l \in \mathbb{N}$ und $E^{(k)} = E^{(k \bmod n)}$. Definiert man weiter

$$\phi_{p,k} : \ E^{(k)} \to E^{(k+1)}, \ P \mapsto \begin{cases} \mathcal{O} & \text{falls } P = \mathcal{O}, \\ (\sigma(x), \sigma(y)) & \text{falls } P = (x, y) \in E^{(k)}(\overline{\mathbb{F}}_q) \setminus \{\mathcal{O}\}, \end{cases}$$

so ist $\phi_{p,k}$ eine Isogenie und wir erhalten folgenden Zyklus von elliptischen Kurven:

$$E = E^{(0)} \xrightarrow{\phi_{p,0}} E^{(1)} \xrightarrow{\phi_{p,1}} \cdots \xrightarrow{\phi_{p,n-2}} E^{(n-1)} \xrightarrow{\phi_{p,n-1}} E^{(n)} = E. \quad (1.11)$$

$\phi_q \coloneqq \phi_{p,n-1} \circ \phi_{p,n-2} \circ \cdots \circ \phi_{p,0} \in \mathrm{End}(E)$ heißt *Frobenius-Endomorphismus* von E.

Man überzeugt sich davon, dass $\phi_{p,k}$ tatsächlich eine wohldefinierte Isogenie ist:

Für $P = (x, y) \in E^{(k)} \setminus \{\mathcal{O}\}$ gilt $y^2 = x^3 + \sigma^k(a)x + \sigma^k(b)$. Es folgt

$$(\sigma(y))^2 = \sigma(y^2) = \sigma(x^3 + \sigma^k(a)x + \sigma^k(b)) = (\sigma(x))^3 + \sigma^{k+1}(a)\sigma(x) + \sigma^{k+1}(b)$$

und damit $\phi_{p,k}(P) \in E^{(k+1)}$.

Proposition 1.29. *Mit den eingeführten Bezeichnungen und $r, s \in \mathbb{Z}$ gilt:*

a) $\phi_q \in \mathrm{End}(E)$ *ist inseparabel und* $\deg(\phi_q) = q$.

b) $r + s\phi_q \in \mathrm{End}(E)$ *ist separabel* $\iff p \nmid r$.

c) E *ist supersingulär* $\iff \mathrm{Tr}(\phi_q) \equiv 0 \pmod{p}$.

d) Falls $j(E) \notin \mathbb{F}_{p^2}$, *so ist* E *gewöhnlich und damit* $\widehat{\phi}_q = \mathrm{Tr}(\phi_q) - \phi_q$ *separabel.*

Beweis. [Sil86, II., Proposition 2.11, III., Corollary 5.5 und V., Theorem 3.1] und [Was08, Proposition 4.31]. □

Wir stellen nun den Zusammenhang zwischen der Spur des Frobenius und der Gruppenordnung $\#E(\mathbb{F}_q)$ her. Anschließend formulieren wir die berühmte Abschätzung von Hasse.

Satz 1.30. *Sei* $E: y^2 = x^3 + ax + b$ *eine elliptische Kurve über* $\mathbb{F}_q$. *Dann gilt*

$$\#E(\mathbb{F}_q) = q + 1 - \mathrm{Tr}(\phi_q).$$

Beweis. Es gilt offenbar

$$E(\mathbb{F}_q) = \ker(1 - \phi_q).$$

Nach Satz 1.19 gilt $\#\ker(1 - \phi_q) = \deg_s(1 - \phi_q)$ und da $1 - \phi_q$ separabel ist (Proposition 1.29 b), folgt

$$\#\ker(1 - \phi_q) = \deg(1 - \phi_q).$$

Korollar 1.16 liefert nun das Gewünschte:

$$\#E(\mathbb{F}_q) = \deg(1 - \phi_q) = \deg(\phi_q) + 1 - \mathrm{Tr}(\phi_q) = q + 1 - \mathrm{Tr}(\phi_q). \qquad \square$$

Satz 1.31 (Hasse)**.**

$$\left|\#E(\mathbb{F}_q) - q - 1\right| = \left|\mathrm{Tr}(\phi_q)\right| \leq 2\sqrt{q}.$$

Beweis. Nach [Sil86, III., Corollary 6.3] ist deg : $\mathrm{End}(E) \to \mathbb{N}$ eine positiv definite quadratische Form, für die nach [Sil86, V., Lemma 1.2] folgende Abschätzung gilt:

$$|\deg(\psi - \phi) - \deg(\phi) - \deg(\psi)| \leq 2\sqrt{\deg(\phi)\deg(\psi)}, \quad \phi, \psi \in \mathrm{End}(E). \qquad \square$$

Die elliptische Kurve $E/\mathbb{F}_q : y^2 = x^3 + ax + b$ ist für $m \geq 1$ auch über $\mathbb{F}_{q^m}$ definiert. Das folgende Lemma gehört zu den faszinierendsten Resultaten im Zusammenhang mit elliptischen Kurven über endlichen Körpern.

Lemma 1.32. *Sei $t_m \in \mathrm{End}(E)$ für $m \in \mathbb{N}$ durch folgende Vorschrift rekursiv definiert:*

$$t_0 \coloneqq 2, \quad t_1 \coloneqq \mathrm{Tr}(\phi_q), \quad t_{m+1} \coloneqq t_1 t_m - q t_{m-1}.$$

Dann gilt $\mathrm{Tr}(\phi_q^m) = t_m$ und

$$\#E(\mathbb{F}_{q^m}) = q^m + 1 - t_m.$$

Beweis. Wir beweisen die erste Aussage durch Induktion nach m. Der Fall $m = 0, 1$ ist trivial (man beachte: $\phi_q^0 = 1$). Es gilt nun für $m \geq 1$, dass

$$\begin{aligned} \mathrm{End}(E) \ni 0 &= \phi_q^2 - (\phi_q + \widehat{\phi}_q)\phi_q + \widehat{\phi}_q \phi_q \\ &= \phi_q^2 - \mathrm{Tr}(\phi_q)\phi_q + q \\ \implies 0 &= \phi_q^2 \phi_q^{m-1} - \mathrm{Tr}(\phi_q)\phi_q \phi_q^{m-1} + q\phi_q^{m-1} \\ &= \phi_q^{m+1} - \mathrm{Tr}(\phi_q)\phi_q^m + q\phi_q^{m-1} \end{aligned}$$

und ebenso

$$0 = \widehat{\phi}_q^{m+1} - \mathrm{Tr}(\phi_q)\widehat{\phi}_q^m + q\widehat{\phi}_q^{m-1}.$$

Wir addieren beide Gleichungen und erhalten

$$\phi_q^{m+1} + \widehat{\phi}_q^{m+1} = \mathrm{Tr}(\phi_q)(\phi_q^m + \widehat{\phi}_q^m) - q(\phi_q^{m-1} + \widehat{\phi}_q^{m-1}).$$

Aus Satz 1.15 c) folgt $\mathrm{Tr}(\phi_q^k) = \phi_q^k + \widehat{\phi}_q^k$ und somit gilt nach Induktionsvoraussetzung

$$\mathrm{Tr}(\phi_q^{m+1}) = \mathrm{Tr}(\phi_q)\,\mathrm{Tr}(\phi_q^m) - q\,\mathrm{Tr}(\phi_q^{m-1}) = t_1 t_m - q t_{m-1} = t_{m+1}.$$

Schließlich folgt aus Satz 1.30 mit $q^m = p^{nm}$ und $\phi_q^m = \phi_{q^m}$, dass

$$\#E(\mathbb{F}_{q^m}) = q^m + 1 - \mathrm{Tr}(\phi_{q^m}) = q^m + 1 - t_m. \qquad \square$$

Bemerkung. Nach Korollar 1.16 ist die Spur eines Endomorphismus eine ganze Zahl. Wir können also $\#E(\mathbb{F}_{q^m})$ für *alle* $m \geq 2$ effizient berechnen – vorausgesetzt wir kennen $\#E(\mathbb{F}_q)$. Man beachte allerdings, dass die Koeffizienten der elliptischen Kurve E in dem „kleinen" Unterkörper $\mathbb{F}_q \subset \mathbb{F}_{q^m}$ liegen.

Proposition 1.33. *Sei* $E: y^2 = g(x) \coloneqq x^3 + ax + b$ *eine elliptische Kurve über* $\mathbb{F}_q$. *Mit* $\mathbb{F}_q[x] \ni g(x)^{(p-1)/2} = \sum_{\nu=0}^{3(p-1)/2} c_\nu x^\nu$ *gilt*

$$\mathrm{Tr}(\phi_q) \bmod p = \mathrm{N}_{\mathbb{F}_q/\mathbb{F}_p}(c_{p-1}) = \prod_{\nu=0}^{n-1} \sigma^\nu(c_{p-1}).$$

Beweis. [Mad03, Proposition A.14]. □

Den Sonderfall $j(E) \in \mathbb{F}_{p^2}$ möchten wir zunächst formal und anschließend an einem konkreten Beispiel untersuchen. Dabei treten unterschiedliche Probleme auf und es lohnt sich diese, für ein tieferes Verständnis, näher zu betrachten.

Sei also im Folgenden $E: y^2 = x^3+ax+b$ eine elliptische Kurve über $\mathbb{F}_{p^n}$, wobei $n \geq 2$. Weiter nehmen wir an, dass $j \coloneqq j(E) \in \mathbb{F}_{p^2} \subset \mathbb{F}_{p^n}$, d. h. $n \equiv 0 \pmod 2$ und $\sigma^2(j) = j$. Wie wir später sehen werden, können wir die Gruppenordnung $\#E(\mathbb{F}_q)$ nicht mit Hilfe von Satohs Algorithmus bestimmen. Vorausgesetzt die Charakteristik p ist nicht zu groß, können wir $\#E(\mathbb{F}_q)$ allerdings effizient durch eine Kombination von Lemma 1.26 und Lemma 1.32 berechnen. Um Fallunterscheidungen zu vermeiden, setzen wir voraus, dass $j \notin \mathbb{F}_p$ (insbesondere also $j \notin \{0, 1728\}$).

Zunächst müssen wir prüfen, ob unsere Kurve E auch tatsächlich über $\mathbb{F}_{p^2}$ definiert ist. Man überzeuge sich davon, dass dies nicht zwingend der Fall ist, selbst wenn $j(E) \in \mathbb{F}_{p^2}$. Falls nötig überführen wir E also in die isomorphe Kurve

$$E'/\mathbb{F}_{p^2}: y^2 = x^3 + a'x + b',$$

wobei $a' \coloneqq 3j/(1728 - j)$, $b' \coloneqq 2j/(1728 - j)$ (vgl. (1.4)). Es spielt jetzt noch keine Rolle, ob wir dabei auf einem quadratischen Twist von E gelandet sind oder nicht.

Nach Lemma 1.26 genügt es nun den (kleinen) Unterkörper $\mathbb{F}_{p^2} \subset \mathbb{F}_{p^n}$ zu „durchlaufen" und dabei $\mathbb{F}_{p^2}$-rationale Punkte zu zählen. Dadurch ergibt sich allerdings folgendes Problem: Um den Körper $\mathbb{F}_q$ darzustellen, verwendet man normalerweise die Isomorphie

$$\mathbb{F}_{p^n} \cong \mathbb{F}_p[X]/(f(X)),$$

wobei $f(X) \in \mathbb{F}_p[X]$ ein normiertes und über $\mathbb{F}_p$ irreduzibles Polynom vom Grad n ist. Ein Element $e \in \mathbb{F}_{p^n}$ lässt sich dann eindeutig schreiben als

$$e = e_0 + e_1\underline{X} + e_2\underline{X}^2 + \cdots + e_{n-1}\underline{X}^{n-1},$$

wobei $e_\nu \in \mathbb{F}_p$ für alle $0 \leq \nu < n$ und $\underline{X} := X + (f(X))$. Doch wie finden wir in dieser Darstellung alle Elemente $e \in \mathbb{F}_{p^2} \setminus \mathbb{F}_p$?

Es gilt offenbar *nicht* $\mathbb{F}_{p^2} \cong \{e_0 + e_1\underline{X} : e_0, e_1 \in \mathbb{F}_p\}$. Man könnte ein erzeugendes Element $d \in \mathbb{F}_q$ der zyklischen Gruppe $\mathbb{F}_{p^2} \setminus \{0\}$ suchen und mit dessen Potenzen $\mathbb{F}_{p^2}$ durchlaufen. Das hätte in einer Implementierung den zusätzlichen Vorteil, dass man die Arithmetik in $\mathbb{F}_q$ weiterhin verwenden könnte und dort bereits eine Darstellung von j, a' und b' kennt. Ein solches d zu finden ist allerdings nicht einfach. Wir gehen einen anderen Weg und konstruieren stattdessen einen Körper-Isomorphismus

$$\Phi : \mathbb{F}_p[Y]/(g(Y)) \to \mathbb{F}_{p^2} \subset \mathbb{F}_p[X]/(f(X)),$$

wobei $g(Y) \in \mathbb{F}_p[Y]$ ein normiertes und über $\mathbb{F}_p$ irreduzibles Polynom vom Grad 2 ist. Unser Ziel ist die Bestimmung von $\Phi^{-1}(j) \in \mathbb{F}_p[Y]/(g(Y)) \cong \mathbb{F}_{p^2}$. Anschließend können wir alle Berechnungen in $\mathbb{F}_p[Y]/(g(Y))$ ausführen, wobei wir nun tatsächlich die „schöne“ Darstellung

$$\mathbb{F}_{p^2} \cong \{e_0 + e_1\underline{Y} : e_0, e_1 \in \mathbb{F}_p\}$$

erhalten haben.

Man „kennt“ Φ bereits, wenn man $\Phi(\underline{Y})$ kennt. Daher genügt es eine Nullstelle von $g(Y)$ in $\mathbb{F}_p[X]/(f(X))$ zu finden:

$$0 = \Phi(0) = \Phi(g(\underline{Y})) = g(\Phi(\underline{Y})).$$

Hierfür benötigt man eventuell Computerunterstützung. Das Urbild von j findet man nun wie folgt: Sei $\Phi^{-1}(j) = e_0 + e_1\underline{Y}$. Wir wollen $e_0, e_1 \in \mathbb{F}_p$ bestimmen. Naiv könnte man alle p^2 Kombinationen durchprobieren, bis die Gleichung

$$\Phi(e_0 + e_1\underline{Y}) = e_0 + e_1\Phi(\underline{Y}) = j$$

erfüllt ist. Es reichen jedoch maximal p Versuche aus, wie folgende Abhängigkeit zeigt: j lässt sich schreiben als $j = j_0 + i\underline{X}$, wobei $j_0 \in \mathbb{F}_p$ und $i \in \mathbb{F}_q$. Ebenso gilt

$$\Phi(\underline{Y}) = s_0 + r\underline{X}, \quad s_0 \in \mathbb{F}_p,\ r \in \mathbb{F}_q.$$

Zusammenfassend erhält man

$$j_0 + i\underline{X} = j = e_0 + e_1\Phi(\underline{Y}) = e_0 + e_1(s_0 + r\underline{X}) = e_0 + e_1 s_0 + e_1 r\underline{X},$$

d. h. e_0 ist durch die Gleichung $j_0 = e_0 + e_1 s_0$ bereits eindeutig festgelegt. Falls $a, b \in \mathbb{F}_{p^2}$, so kann man auf die Berechnung von $\Phi^{-1}(j)$ verzichten und findet stattdessen auf die gleiche Weise eine Darstellung der Kurvenparameter in $\mathbb{F}_p[Y]/(g(Y))$. Ansonsten

definiert man E' wie angegeben und hat mit $k := \Phi^{-1}(j)$ folgende Darstellung von a' und b' in $\mathbb{F}_p[Y]/(g(Y))$:

$$\Phi^{-1}(a') = 3k/(1728-k), \quad \Phi^{-1}(b') = 2k/(1728-k).$$

Nachdem man $\#E'(\mathbb{F}_{p^2})$ berechnet hat, wendet man die Rekursionsformel aus Lemma 1.32 an und erhält $\#E'(\mathbb{F}_q)$. Es bleibt schließlich noch zu klären, ob E' ein quadratischer Twist von E ist. Nach Satz 1.7 ist dies genau dann der Fall, wenn

$$\frac{ab'}{ba'} = \frac{2a}{3b} =: v \in \mathbb{F}_q$$

kein Quadrat in $\mathbb{F}_q$ ist. Mit (1.10) gilt also

$$\#E(\mathbb{F}_q) = \left(1 - \left(\frac{v}{\mathbb{F}_q}\right)\right)(q+1) + \left(\frac{v}{\mathbb{F}_q}\right)\#E'(\mathbb{F}_q).$$

Beispiel 1.34. Sei $E:\ y^2 = x^3 + ax + b$ eine elliptische Kurve über $\mathbb{F}_{17^6}$, wobei

$$\begin{aligned}\mathbb{F}_{17^6} &\cong \mathbb{F}_{17}[X]/(7 + X + X^6),\\ a &= 1 + \underline{X} + 14\underline{X}^2 + 8\underline{X}^3 + 11\underline{X}^4 + 10\underline{X}^5,\\ b &= 2 + 14\underline{X} + 11\underline{X}^2 + 6\underline{X}^3 + 6\underline{X}^4 + 11\underline{X}^5.\end{aligned}$$

Gesucht: $\#E(\mathbb{F}_{17^6})$.

Man berechnet zunächst $j := j(E) = 12 + 5\underline{X} + 11\underline{X}^2 + 14\underline{X}^3 + 6\underline{X}^4 + 14\underline{X}^5$ und verifiziert, dass $j \in \mathbb{F}_{289}$, aber $a, b \in \mathbb{F}_{17^6} \setminus \mathbb{F}_{289}$. Um eine zu E isomorphe Kurve E' über $\mathbb{F}_{17^2} \cong \mathbb{F}_{17}[Y]/(3 + Y^2)$ zu konstruieren, bestimmt man anschließend in $\mathbb{F}_{17^6}$ eine Lösung der Gleichung $Y^2 = -3$:

$$\left(15 + 4\underline{X} + 2\underline{X}^2 + \underline{X}^3 + 15\underline{X}^4 + \underline{X}^5\right)^2 = 14 = -3.$$

Mit obiger Notation gilt dann

$$\Phi^{-1}(j) = 6 + 14\underline{Y} \in \mathbb{F}_{17}[Y]/(3 + Y^2), \quad \Phi^{-1}(a') = 9 + 3\underline{Y}, \quad \Phi^{-1}(b') = 6 + 2\underline{Y}.$$

Weiter erhält man $\#E'(\mathbb{F}_{289}) = 289$, d. h. $t_1 = 1$. Die Rekursionsformel liefert

$$\begin{aligned}t_2 &= t_1^2 - 289t_0 = 1 - 289 \cdot 2 = -577,\\ t_3 &= t_1 t_2 - 289t_1 = -577 - 289 = -866.\end{aligned}$$

Daraus folgt $\#E'(\mathbb{F}_{17^6}) = \#E'(\mathbb{F}_{289^3}) = 289^3 + 1 - t_3 = 24\,138\,436$.

Mit $\mathbb{F}_{17} \ni 2 \cdot 3^{-1} = 12$ berechnet man

$$12a \cdot b^{-1} = 6 + \underline{X} + \underline{X}^2 + 11\underline{X}^3 + 5\underline{X}^5 = \left(2 + 15\underline{X} + 5\underline{X}^2 + 16\underline{X}^3 + 9\underline{X}^4 + 4\underline{X}^5\right)^2.$$

Also ist E' kein quadratischer Twist von E und es folgt

$$\#E(\mathbb{F}_{17^6}) = \#E'(\mathbb{F}_{17^6}) = 24\,138\,436.$$

Angenommen die Berechnung von $\#E'(\mathbb{F}_{289})$ dauert 100 ms (diese Größenordnung wurde mit einer ARIBAS[12] Implementierung auf der bereits vorgestellten Architektur erreicht), dann würde die triviale Bestimmung von $\#E'(\mathbb{F}_{17^6})$ nach Lemma 1.26 über zwei Stunden dauern! Hierbei vernachlässigen wir sogar noch, dass die Arithmetik in $\mathbb{F}_{17^6}$ langsamer ist als in $\mathbb{F}_{289}$. Wir verzichten auf weitere Abschätzungen. Die Mächtigkeit der Rekursionsformel aus Lemma 1.32 sollte deutlich geworden sein.

[12] [For]

2 Der kanonische Lift einer elliptischen Kurve

2.1 Der Körper der p-adischen Zahlen

Die p-adischen Zahlen wurden Anfang des zwanzigsten Jahrhunderts von dem Mathematiker Kurt Hensel (1861–1941) „erfunden“. Eine detaillierte Einführung in dieses interessante Themengebiet findet man in [Bac64], [Kob84] und [Neu92].

Sei im Folgenden $p \in \mathbb{N}$ stets eine Primzahl und für $k, m \in \mathbb{N}$ $(k \leq m)$

$$[k, m] \coloneqq \{l \in \mathbb{N} : \ k \leq l \leq m\} \subset \mathbb{N}.$$

Jede natürliche Zahl $a \in \mathbb{N}$ besitzt eine (eindeutige) *p-adische Entwicklung*

$$a = a_0 + a_1 p + \cdots + a_n p^n, \quad \text{wobei } a_i \in [0, p-1] \text{ (für } i \in [0, n]).$$

Die Koeffizienten erhält man durch wiederholte Division mit Rest:

$$\begin{aligned}
a_0 &= a \bmod p, \\
a_1 &= ((a - a_0)/p) \bmod p, \\
a_2 &= (((a - a_0)/p - a_1)/p) \bmod p, \\
a_3 &= ((((a - a_0)/p - a_1)/p - a_2)/p) \bmod p, \quad \text{usw.}
\end{aligned}$$

Negative Zahlen hingegen besitzen keine *endliche* p-adische Entwicklung, da das angegebene Verfahren in diesem Fall nicht abbricht.

Definition 2.1. Eine *ganze p-adische Zahl* ist eine formale unendliche Reihe

$$\sum_{\nu=0}^{\infty} a_\nu p^\nu = a_0 + a_1 p + a_2 p^2 + \cdots, \quad \text{wobei } a_\nu \in [0, p-1] \text{ (für } \nu \in \mathbb{N}).$$

Die Menge der ganzen p-adischen Zahlen wird mit $\mathbb{Z}_p$ bezeichnet.

Satz 2.2. *Sei $a \in \mathbb{Z}$. Die Restklassen $a + p^n\mathbb{Z} \in \mathbb{Z}/p^n\mathbb{Z}$ sind in eindeutiger Darstellung durch*

$$a_0 + a_1 p + a_2 p^2 + \cdots + a_{n-1} p^{n-1} + p^n \mathbb{Z}$$

gegeben, wobei $a_i \in [0, p-1]$.

Beweis. [Neu92, II., (1.2)]. □

Jede ganze Zahl a definiert also eine Folge von Restklassen

$$r_n \coloneqq a + p^n\mathbb{Z} \in \mathbb{Z}/p^n\mathbb{Z}, \quad n = 1, 2, \ldots,$$

für die nach obigem Satz gilt:

$$\begin{aligned} r_1 &= a_0 + p\mathbb{Z}, \\ r_2 &= a_0 + a_1 p + p^2\mathbb{Z}, \\ r_3 &= a_0 + a_1 p + a_2 p^2 + p^3\mathbb{Z}, \quad \text{usw.} \end{aligned}$$

mit eindeutig bestimmten und gleichbleibenden Koeffizienten $a_i \in [0, p-1]$. Wir nennen die dadurch definierte ganze p-adische Zahl

$$\sum_{\nu=0}^{\infty} a_\nu p^\nu \in \mathbb{Z}_p$$

p-adische Entwicklung von a.

Beispiel 2.3.

$$\begin{aligned} 251 &= 2 + 2\cdot 3 + 0\cdot 3^2 + 0\cdot 3^3 + 0\cdot 3^4 + 1\cdot 3^5 && \in \mathbb{Z}_3, \\ -9 &= 1 + 3\cdot 5 + 4\cdot 5^2 + 4\cdot 5^3 + \cdots && \in \mathbb{Z}_5, \\ 1023 &= 1 + 6\cdot 7 + 6\cdot 7^2 + 2\cdot 7^3 && \in \mathbb{Z}_7. \end{aligned}$$

Um nun auf $\mathbb{Z}_p$ eine Addition und Multiplikation zu definieren, benutzen wir eine andere Darstellung. Wir betrachten $\sum_{\nu=0}^{\infty} a_\nu p^\nu$ nicht als Folge der ganzzahligen Partialsummen

$$s_n = \sum_{\nu=0}^{n-1} a_\nu p^\nu \in \mathbb{Z},$$

sondern als Folge der Restklassen $s_n + p^n\mathbb{Z} \in \mathbb{Z}/p^n\mathbb{Z}$. Die Folgenglieder liegen dann in verschiedenen Ringen $\mathbb{Z}/p^n\mathbb{Z}$, allerdings sind diese durch die kanonischen Projektionen

$$\mathbb{Z}/p\mathbb{Z} \xleftarrow{\lambda_1} \mathbb{Z}/p^2\mathbb{Z} \xleftarrow{\lambda_2} \mathbb{Z}/p^3\mathbb{Z} \xleftarrow{\lambda_3} \cdots$$

verbunden, und es gilt

$$\lambda_n(s_{n+1} + p^{n+1}\mathbb{Z}) = s_n + p^n\mathbb{Z}. \tag{2.1}$$

Nach [Neu92, II., (1.3)] erhält man dadurch eine Bijektion zwischen $\mathbb{Z}_p$ und dem *projektiven Limes* der Ringe $\mathbb{Z}/p^n\mathbb{Z}$, welcher wie folgt definiert ist:

$$\varprojlim \mathbb{Z}/p^n\mathbb{Z} \coloneqq \{(x_n)_{n\geq 1} \in \prod_{n=1}^{\infty} \mathbb{Z}/p^n\mathbb{Z} : \ \lambda_n(x_{n+1}) = x_n\}.$$

Wir identifizieren also $\mathbb{Z}_p$ mit $\varprojlim \mathbb{Z}/p^n\mathbb{Z}$ und erhalten den (nullteilerfreien) *Ring der ganzen p-adischen Zahlen*, in dem Addition und Multiplikation komponentenweise erklärt sind. $\sum_{\nu=0}^{\infty} a_\nu p^\nu \in \mathbb{Z}_p$ ist offenbar[1] genau dann eine Einheit, wenn $a_0 \neq 0$. Für eine Implementierung der Arithmetik in $\mathbb{Z}_p$ genügt es wegen (2.1), eine ausreichend große Genauigkeit (Präzision) $N \geq 1$ zu wählen und anschließend nur in $\mathbb{Z}/p^N\mathbb{Z}$ zu rechnen.

Definition 2.4. $\mathbb{Q}_p \coloneqq \mathrm{Quot}(\mathbb{Z}_p)$ heißt *Körper der p-adischen Zahlen.*

Sei $a \in \mathbb{Q}$ eine beliebige rationale Zahl. Wir schreiben

$$a = \frac{b}{c} \cdot p^m \quad \text{mit } b, c, m \in \mathbb{Z}, \quad \gcd(bc, p) = 1.$$

b/c liegt dann in $\mathbb{Z}_p$ und wenn $\sum_{\nu=0}^{\infty} a_\nu p^\nu$ die zugehörige p-adische Entwicklung ist, so nennen wir

$$a_0 p^m + a_1 p^{m+1} + a_2 p^{m+2} + \cdots \tag{2.2}$$

p-adische Entwicklung von a.

Beispiel 2.5.

$$\begin{aligned}
251 &= (2, 8, 8, 8, 8, 251, 251, \ldots) \in \varprojlim \mathbb{Z}/3^n\mathbb{Z} \\
\frac{1}{251} &= (2, 8, 17, 71, 152, 395, 1124, \ldots) \\
&= 2 + 2 \cdot 3 + 1 \cdot 3^2 + 2 \cdot 3^3 + 1 \cdot 3^4 + 1 \cdot 3^5 + 1 \cdot 3^6 + \cdots \in \mathbb{Z}_3 \\
\frac{1}{753} &= 2 \cdot 3^{-1} + 2 + 1 \cdot 3 + 2 \cdot 3^2 + 1 \cdot 3^3 + 1 \cdot 3^4 + 1 \cdot 3^5 + \cdots \in \mathbb{Q}_3
\end{aligned}$$

Wir möchten nun noch eine weitere Möglichkeit vorstellen den Körper der p-adischen Zahlen einzuführen.

Definition 2.6. Für $a \in \mathbb{Z} \setminus \{0\}$ und eine Primzahl p definieren wir

$$\mathrm{ord}_p(a) \coloneqq \max\{k \in \mathbb{N} : \ p^k \mid a\}.$$

Man setzt $\mathrm{ord}_p(0) \coloneqq \infty$. Außerdem verallgemeinert man für $a \in \mathbb{Q}^*,\ a = b/c,\ b, c \in \mathbb{Z}$

$$\mathrm{ord}_p(a) \coloneqq \mathrm{ord}_p(b) - \mathrm{ord}_p(c).$$

[1] $\varprojlim \mathbb{Z}/p^n\mathbb{Z} \ni 1 = (1, 1, 1, \ldots)$

Diese Definition ist unabhängig von der Wahl der Repräsentanten b und c. Die Funktion $\mathrm{ord}_p : \mathbb{Q} \to \mathbb{Z} \cup \{\infty\}$ heißt *p-adische Exponentialbewertung* von $\mathbb{Q}$. Wie man leicht nachprüft, gilt für $a, b \in \mathbb{Q}$

(i) $\mathrm{ord}_p(a) = \infty \iff a = 0$,

(ii) $\mathrm{ord}_p(ab) = \mathrm{ord}_p(a) + \mathrm{ord}_p(b)$,

(iii) $\mathrm{ord}_p(a \pm b) \geq \min\{\mathrm{ord}_p(a), \mathrm{ord}_p(b)\}$,

wobei $x + \infty = \infty$, $\infty + \infty = \infty$ und $\infty > x$ für alle $x \in \mathbb{Q}$ gelten soll.

Definition 2.7. Der *p-adische Absolutbetrag* ist durch

$$|\ |_p : \mathbb{Q} \to \mathbb{Q}, \quad a \mapsto |a|_p := p^{-\mathrm{ord}_p(a)},$$

gegeben und erfüllt wegen (i), (ii) und (iii) die Bedingungen einer *Norm* auf $\mathbb{Q}$:

(iv) $|a|_p = 0 \iff a = 0$,

(v) $|ab|_p = |a|_p \, |b|_p$,

(vi) $|a \pm b|_p \leq \max\{|a|_p, |b|_p\}$ $(\leq |a|_p + |b|_p)$.

Wie man sich überlegt, folgt aus $|a|_p \neq |b|_p$, dass $|a \pm b|_p = \max\{|a|_p, |b|_p\}$. Diese Eigenschaft wird auch als „isosceles triangle principal“ bezeichnet (man stelle sich das durch a, b und 0 gegebene „Dreieck“ mit den Seitenlängen $|a - 0|_p$, $|b - 0|_p$ und $|a - b|_p$ vor).

Wir nennen eine Norm $|\ |$ *nicht-archimedisch*, wenn die verschärfte Abschätzung (vi) gilt. Dies ist nach [Neu92, II., (3.6)] genau dann der Fall, wenn $|n|$ für alle $n \in \mathbb{N}$ beschränkt ist. Der gewöhnliche Absolutbetrag $|\ |_\infty$ ist keine nicht-archimedische Norm. Nach Ostrowski[2] ist jede nicht-triviale Norm auf $\mathbb{Q}$ äquivalent[3] zu $|\ |_\infty$ oder $|\ |_p$ (für eine Primzahl p).

Wir definieren nun den Körper der p-adischen Zahlen aufs Neue, indem wir genauso vorgehen wie bei der Konstruktion des Körpers der reellen Zahlen.

Unter einer *Cauchyfolge* bzgl. $|\ |_p$ verstehen wir eine Folge $(x_n)_{n \in \mathbb{N}}$ rationaler Zahlen, so dass es zu jedem $\varepsilon > 0$ eine natürliche Zahl n_0 gibt mit

$$|x_n - x_m|_p < \varepsilon \quad \text{für alle } n, m \geq n_0.$$

[2] [Kob84, Theorem 1]
[3] [Kob84, I., 2.]

Die Menge aller Cauchyfolgen in $\mathbb{Q}$ bildet dann einen Ring A, die Nullfolgen ein maximales Ideal $M \subset A$ (vgl. [Bac64, II., 1.]), und wir definieren den Körper der p-adischen Zahlen als den Restklassenkörper

$$\mathbb{Q}_p \coloneqq A/M.$$

Wir setzen den p-adischen Absolutbetrag auf $\mathbb{Q}_p$ fort, indem wir $x = (x_n)_{n\in\mathbb{N}} + M$ den Betrag

$$|x|_p \coloneqq \lim_{n\to\infty} |x_n|_p \in \mathbb{R}$$

zuordnen. Der Grenzwert existiert, da $(|x_n|_p)_{n\in\mathbb{N}}$ eine Cauchyfolge in $\mathbb{R}$ ist:

$$\Big|\, |x_n|_p - |x_m|_p \,\Big|_\infty \leq |x_n - x_m|_p\,.$$

Zudem ist er unabhängig von der Wahl des Repräsentanten $(x_n)_{n\in\mathbb{N}}$. Man kann sogar zeigen[4], dass sich die Norm stabilisiert, d. h.

$$\exists\, i_0 \in \mathbb{N} :\ |x_i|_p = \left|x_j\right|_p \quad \text{für alle } i,j \geq i_0 \quad (\Longrightarrow\ |x|_p = \left|x_{i_0}\right|_p).$$

Mit anderen Worten: $|x|_p$ ist eine rationale Zahl! Nach [Bac64, II., 1.] ist $|\ |_p$ nichtarchimedisch.

Satz 2.8. *Der Körper der p-adischen Zahlen ist bezüglich* $|\ |_p$ vollständig.

Beweis. [Bac64, II., 1.]. □

Bemerkung. Ein Körper $(K, |\ |)$ heißt *vollständig*, wenn jede Cauchyfolge (bzgl. $|\ |$) von Elementen aus K einen Grenzwert in K besitzt. In der obigen Situation betrachten wir also eigentlich Cauchyfolgen *von Cauchyfolgen*.

Die rationalen Zahlen bettet man auf natürliche Weise in $\mathbb{Q}_p$ ein:

$$\mathbb{Q} \ni a \mapsto (a, a, a, \ldots) + M \in A/M,$$

also folgt $\operatorname{char} \mathbb{Q}_p = 0$. Nach [Neu92, II., (2.3)] bildet die Menge

$$\mathbb{Z}_p \coloneqq \{x \in \mathbb{Q}_p :\ |x|_p \leq 1\}$$

einen Unterring von $\mathbb{Q}_p$. Es gilt offenbar $\mathbb{Z}_p^* = \{x \in \mathbb{Q}_p :\ |x|_p = 1\}$. Außerdem sind durch die folgende (unendliche) Kette von Hauptidealen bereits *alle* von Null verschiedenen Ideale des Ringes $\mathbb{Z}_p$ gegeben:

$$\mathbb{Z}_p \ \supset\ p\mathbb{Z}_p \ \supset\ p^2\mathbb{Z}_p \ \supset\ \cdots$$

[4] [Kob84, I., 4.]

$p\mathbb{Z}_p$ ist also das einzige maximale Ideal und $\mathbb{Z}_p$ per Definition[5] ein *diskreter Bewertungsring*. Für $0 \neq x \in \mathbb{Q}_p$ ist entweder $x \in \mathbb{Z}_p$ oder $x^{-1} \in \mathbb{Z}_p$. Nach [Neu92, II., (2.4)] gilt für $n \geq 1$

$$\mathbb{Z}_p/p^n\mathbb{Z}_p \cong \mathbb{Z}/p^n\mathbb{Z}. \tag{2.3}$$

Der Zusammenhang mit unserer ursprünglichen Definition der ganzen p-adischen Zahlen wird nun durch die Isomorphie

$$\mathbb{Z}_p \cong \varprojlim \mathbb{Z}/p^n\mathbb{Z}$$

hergestellt (siehe [Neu92, II., (2.5)]). Etwas anschaulicher ist sicher der folgende

Satz 2.9. *Jedes $x \in \mathbb{Z}_p$ besitzt genau eine Darstellung $x = (x_n)_{n\in\mathbb{N}} + M \in A/M$, so dass für alle $i \in \mathbb{N}$ gilt:*

a) $x_i \in [0, p^{i+1} - 1]$ *und*

b) $x_{i+1} \equiv x_i \pmod{p^{i+1}}$.

Beweis. [Kob84, Theorem 2]. □

Wir sind damit an unserem ursprünglichen Ausgangspunkt angelangt, denn die Folge $(x_n)_{n\in\mathbb{N}}$ lässt sich nun schreiben als

$$(a_0,\ a_0 + a_1p,\ a_0 + a_1p + a_2p^2, \ldots), \quad a_i \in [0, p-1],$$

definiert also die bekannte Folge von Partialsummen, die wir mit $\sum_{\nu=0}^{\infty} a_\nu p^\nu$ bezeichnet haben.

Falls $x \in \mathbb{Q}_p \setminus \mathbb{Z}_p$, folgt $|x|_p = p^m$ mit $m \geq 1$. Für $y := x \cdot p^m$ gilt dann $y \in \mathbb{Z}_p$ und wenn $\sum_{\nu=0}^{\infty} a_\nu p^\nu$ die nach obigem Satz existierende zugehörige p-adische Entwicklung ist, so ordnen wir $x = y \cdot p^{-m}$ folgende Darstellung zu (vgl. (2.2)):

$$\sum_{\nu=-m}^{\infty} a_{\nu+m} p^\nu.$$

Abschließend führen wir für $x, y \in \mathbb{Q}_p$, $N \geq 1$ noch die Notation $x \equiv y \pmod{p^N}$ ein, womit wir ausdrücken, dass $|x - y|_p \leq p^{-N}$, d. h. $(x-y)/p^N \in \mathbb{Z}_p$ bzw. $x - y \in p^N\mathbb{Z}_p$. Falls x und y sogar ganze Zahlen sind, stimmt diese Definition mit der gewöhnlichen Interpretation überein. Für $a \in \mathbb{Z}_p$ sei

$$a \bmod p^N := \sum_{\nu=0}^{N-1} a_\nu p^\nu \in [0, p^N - 1] \subset \mathbb{Z}_p,$$

[5] [Neu92, I., (11.3)]

wobei $\sum_{\nu=0}^{\infty} a_\nu p^\nu$ die p-adische Entwicklung von a ist. Es gilt dann trivialerweise

$$a \equiv \sum_{\nu=0}^{N-1} a_\nu p^\nu \pmod{p^N}.$$

Um für $a \in \mathbb{Z}_p^*$ die Approximation $a^{-1} \bmod p^N$ zu berechnen, reicht es aus

$$b \coloneqq a \bmod p^N + p^N\mathbb{Z} \in (\mathbb{Z}/p^N\mathbb{Z})^*$$

zu invertieren. b^{-1} lässt sich schreiben als $b^{-1} = c + p^N\mathbb{Z}$ (mit $c \in [0, p^N - 1]$) und es gilt $a^{-1} \bmod p^N = c$.

Bevor wir uns mit endlichen Körpererweiterungen von $\mathbb{Q}_p$ beschäftigen, stellen wir noch eine erste p-adische Variante der bekannten Newtoniteration[6] vor. Es stellt sich heraus, dass wir uns dabei (im Gegensatz zum reellen Fall) keine Sorgen um die Konvergenz zu machen brauchen.

Satz 2.10 (Hensel)**.** *Sei $f(X) \in \mathbb{Z}_p[X]$ mit formaler Ableitung $f'(X)$ und $a_0 \in \mathbb{Z}_p$, so dass $f(a_0) \equiv 0 \pmod{p}$ und $f'(a_0) \not\equiv 0 \pmod{p}$. Dann existiert genau eine ganze p-adische Zahl $a \in \mathbb{Z}_p$, so dass*

$$f(a) = 0 \quad \textit{und} \quad a \equiv a_0 \pmod{p}.$$

Beweis. [Kob84, Theorem 3]. □

Aus dem (konstruktiven) Beweis ergibt sich direkt Algorithmus 1 auf der nächsten Seite. Man kann Nullstellen allerdings viel effektiver berechnen; die Genauigkeit N verdoppelt sich dabei in jedem Schritt (siehe Algorithmus 3 auf Seite 44). Wir haben die „langsame“ Variante dennoch vorgestellt, weil sie nur *eine* modulare Inversion benötigt (Zeile 3).

Die Startnäherung a_0 gewinnt man normalerweise durch Nullstellensuche in $\mathbb{F}_p$ (was für „große“ p keinesfalls trivial ist[7]). Wir führen deshalb bereits an dieser Stelle für $N \geq 1$ die *kanonischen Projektionen* ein:

$$\pi_N : \ \mathbb{Z}_p[X] \to (\mathbb{Z}/p^N\mathbb{Z})[X], \ f(X) = \sum_{\nu=0}^{d} c_\nu X^\nu \mapsto \sum_{\nu=0}^{d} (c_\nu \bmod p^N + p^N\mathbb{Z})X^\nu.$$

Wenn man $\mathbb{Z}_p$ mit $\varprojlim \mathbb{Z}/p^n\mathbb{Z}$ identifiziert, betrachtet man also jeweils das N-te Folgenglied. Es gilt

$$\deg(\pi_1(f(X))) = \deg(f(X)) \iff c_d \in \mathbb{Z}_p^*.$$

[6] [For04, §17]

[7] Eine Quadratwurzel in $\mathbb{F}_q$ kann man z. B. mit [BSS99, Algorithm II.8] (nach einer Methode von Tonelli und Shanks) berechnen.

Algorithmus 1 Einfache Nullstellensuche in $\mathbb{Z}_p$

Eingabe: $f(X) \in \mathbb{Z}_p[X]$ und $a_0 \in \mathbb{Z}_p$ wie in Satz 2.10, sowie $N \geq 1$.
Ausgabe: $a \in \mathbb{Z}_p$, so dass $f(a) \equiv 0 \pmod{p^N}$ und $a \equiv a_0 \pmod{p}$.

```
1: procedure FIND_ROOT(f, a_0, N)
2:     a ← a_0 mod p
3:     b ← −f'(a)^{−1} mod p
4:     for i ← 2 to N do
5:         c ← (f(a) mod p^i)/p^{i−1}                ▷ f(a) ≡ 0 (mod p^{i−1})
6:         a ← a + (bc mod p) · p^{i−1}
7:     end for
8:     return a
9: end procedure
```

2.2 Unverzweigte Erweiterungen von $\mathbb{Q}_p$

Wir behandeln im Folgenden endliche Körpererweiterungen $K \supset \mathbb{Q}_p$. Um Verwechslungen zu vermeiden, werden wir in diesem Zusammenhang die *Norm* im Sinne der Algebra[8] ausschließlich mit

$$\mathrm{N}_{K/\mathbb{Q}_p}(a) \in \mathbb{Q}_p \quad (a \in K)$$

bezeichnen. Beispielsweise gilt[9] $\mathrm{N}_{K/\mathbb{Q}_p}(a) = (-1)^n c_0$ für $K = \mathbb{Q}_p(a)$, wobei $n \geq 2$ der Grad der Körpererweiterung und $\sum_{\nu=0}^{n} c_\nu X^\nu \in \mathbb{Q}_p[X]$ das Minimalpolynom von a über K ist.

Wir möchten den p-adischen Absolutbetrag $|\ |_p$ auf K fortsetzen, d. h. eine Norm $|\ |$ auf K angeben, so dass $|x| = |x|_p$ für alle $x \in \mathbb{Q}_p \subset K$. Zunächst kann man zeigen[10], dass es höchstens *eine* Norm auf K gibt, die diese Anforderung erfüllt. Die Existenz formulieren wir als

Satz 2.11. *Sei $K/\mathbb{Q}_p$ eine endliche Körpererweiterung vom Grad n. Dann ist die eindeutige Fortsetzung des p-adischen Absolutbetrags auf K gegeben durch*

$$|a|_p \coloneqq \left|\mathrm{N}_{K/\mathbb{Q}_p}(a)\right|_p^{1/n} \quad (a \in K).$$

Außerdem ist K bezüglich der nicht-archimedischen Norm $|\ |_p \colon K \to \mathbb{Q}$ vollständig.

[8] [Bos06, 4.7, Definition 1]
[9] [Bos06, 4.7, Lemma 2]
[10] [Kob84, III., 2.]

Beweis. [Kob84, Theorem 11] bzw. [Neu92, II., (4.8)]. □

Unter den Voraussetzungen des obigen Satzes gilt nun, dass

$$R \coloneqq \{a \in K : \ |a|_p \leq 1\}$$

ein Unterring von K ist, mit dem einzigen maximalen Ideal

$$M \coloneqq \{a \in K : \ |a|_p < 1\}.$$

Der Restklassenkörper R/M ist eine endliche Erweiterung von $\mathbb{F}_p$ und es gilt[11]

$$[R/M : \mathbb{F}_p] \leq n.$$

Definition 2.12. Mit den eingeführten Bezeichnungen heißt die Körpererweiterung $K/\mathbb{Q}_p$ *unverzweigt*, wenn $[R/M : \mathbb{F}_p] = n$.

Die unverzweigten Körpererweiterungen von $\mathbb{Q}_p$ (Charakteristik 0) sind also direkt mit den Erweiterungen $\mathbb{F}_{p^n} \supset \mathbb{F}_p$ (Charakteristik p) verbunden. Wir fassen die für uns wichtigsten Aussagen zusammen:

Satz 2.13. *Sei $K \supset \mathbb{Q}_p$ eine unverzweigte Erweiterung vom Grad n. Dann gilt*

$$K \cong \mathbb{Q}_p[X]/(F(X)),$$

wobei $F(X) \in \mathbb{Z}_p[X]$ normiert, $\deg(F(X)) = n$ *und $\pi_1(F(X))$ über $\mathbb{F}_p$ irreduzibel ist. Umgekehrt ist jede Erweiterung dieser Form unverzweigt. Mit obiger Notation gilt*

$$R \cong \mathbb{Z}_p[X]/(F(X)), \quad M = pR \quad \textit{und} \quad R/M \cong \mathbb{F}_p[X]/(\pi_1(F(X))).$$

Beweis. [Mad03, Proposition A.7] □

Nach [Mad03, Corollary A.8] gibt es bis auf Isomorphie nur *eine* unverzweigte Erweiterung $K/\mathbb{Q}_p$ mit $[K : \mathbb{Q}_p] = n$. Diese bezeichnen wir mit $\mathbb{Q}_{p^n}$. Analog setzen wir

$$\mathbb{Z}_{p^n} \coloneqq \{a \in \mathbb{Q}_{p^n} : \ |a|_p \leq 1\}. \tag{2.4}$$

Mit Satz 2.13 und $q = p^n$ gilt also $\mathbb{Z}_q \cong \mathbb{Z}_p[X]/(F(X))$ und $\mathbb{Z}_q/p\mathbb{Z}_q \cong \mathbb{F}_q$. Allgemein erhalten wir für $N \geq 1$ (vgl. (2.3)):

$$\mathbb{Z}_q/p^N\mathbb{Z}_q \cong (\mathbb{Z}/p^N\mathbb{Z})[X]/(\pi_N(F(X))).$$

[11] [Kob84, III., 2.]

Dieser Isomorphismus stellt die Grundlage für unsere Implementierung der Arithmetik in $\mathbb{Z}_q$ dar. Genauer lässt sich jedes $a \in \mathbb{Z}_q$ eindeutig schreiben als

$$a = \sum_{\nu=0}^{n-1} a_\nu \overline{X}^\nu, \qquad \text{wobei } a_\nu \in \mathbb{Z}_p \text{ und } \overline{X} \coloneqq X + (F(X)).$$

Eine Approximation von a zur Präzision N ist nun gegeben durch:

$$a \bmod p^N \coloneqq \sum_{\nu=0}^{n-1} (a_\nu \bmod p^N)\overline{X}^\nu =: b.$$

Es gilt $a - b \in p^N\mathbb{Z}_q$, wofür man auch $a \equiv b \pmod{p^N}$ schreibt. Nicht ganz korrekt kann man sich $a \bmod p$ bereits als Element von $\mathbb{F}_q$ vorstellen. Genauer definieren wir die Abbildung

$$\begin{aligned} \pi :\ \mathbb{Z}_q \cong \mathbb{Z}_p[X]/(F(X)) &\to (\mathbb{Z}/p\mathbb{Z})[X]/(\pi_1(F(X))) \cong \mathbb{F}_q, \\ \sum_{\nu=0}^{n-1} a_\nu \overline{X}^\nu &\mapsto \sum_{\nu=0}^{n-1} ((a_\nu \bmod p) + p\mathbb{Z})\underline{X}^\nu, \end{aligned}$$

wobei $\underline{X} \coloneqq X + (\pi_1(F(X)))$. Diese etwas schwerfällige Formulierung täuscht darüber hinweg, dass sie sich in einem Programm sehr leicht umsetzen lässt – Elemente aus $\mathbb{Z}_q$ werden normalerweise als Vektoren $v \in [0, p^N - 1]^n$ gespeichert. Allgemein nennen wir $a \in \mathbb{Z}_q$ *Lift* von $\mathbb{F}_q \ni w = \sum_{\nu=0}^{n-1} w_\nu \underline{X}^\nu$, wenn $\pi(a) = w$. Außerdem setzen wir π auf dem Polynomring $\mathbb{Z}_q[X]$ koeffizientenweise fort. Mit $\pi^{-1}(w) \in \mathbb{Z}_q$ bezeichnen wir das Element

$$\sum_{\nu=0}^{n-1} b_\nu \overline{X}^\nu,$$

wobei $b_\nu \in [0, p-1]$ und $b_\nu + p\mathbb{Z} = w_\nu$ für alle $\nu \in [0, n-1]$.

Fast immer ist der endliche Grundkörper $\mathbb{F}_q \cong \mathbb{F}_p[X]/(f(X))$ und nicht $\mathbb{Z}_q$ unser Ausgangspunkt. Wir fassen die Koeffizienten von $f(X) \in \mathbb{F}_p[X]$ als Elemente der Menge $[0, p-1]$ auf[12] und erhalten ein Polynom über $\mathbb{Z}$, mit dem wir die Körpererweiterung $\mathbb{Q}_q \supset \mathbb{Q}_p$ darstellen. Ab sofort arbeiten wir nur noch mit diesem „trivialen Lift“ von $f(X)$.

Nach [CF06, Proposition 3.22] ist die Erweiterung $\mathbb{Q}_q/\mathbb{Q}_p$ galoissch. Die Galoisgruppe $\mathrm{Gal}(\mathbb{Q}_q/\mathbb{Q}_p)$ ist zyklisch und isomorph zu $\mathrm{Gal}(\mathbb{F}_q/\mathbb{F}_p) = \langle\sigma\rangle$. Sie besitzt genau einen erzeugenden Automorphismus Σ mit der Eigenschaft

$$\pi(\Sigma(a)) = \sigma(\pi(a)) \quad (a \in \mathbb{Z}_q).$$

[12] Jedes $a \in \mathbb{Z}/p\mathbb{Z}$ besitzt eine Darstellung $a = b + p\mathbb{Z}$, wobei $b \in [0, p-1]$.

Σ heißt *Frobenius-Substitution* auf $\mathbb{Q}_q$ (vgl. [Mad03, Corollary A.11]). Während σ jedoch recht einfach zu berechnen ist, gilt dies für Σ nicht mehr.

Wir stellen nun ein sehr effizientes Verfahren[13] vor, um

$$\mathbb{Q}_p \ni \mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(a) = \prod_{\nu=0}^{n-1} \Sigma^\nu(a)$$

für $a \in \mathbb{Q}_q$ zu berechnen. Es wurde von Harley in einer E-Mail[14] an die NMBRTHRY Mailingliste vorgeschlagen. Mit diesem Hilfsmittel lässt sich der ursprünglichen Algorithmus von Satoh deutlich verkürzen und schneller machen. Es gilt $\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(p^k a) = p^{kn}\,\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(a)$ (für $k \in \mathbb{Z}$). Man kann sich also auf den Fall $a \in \mathbb{Z}_q$ beschränken.

Proposition 2.14. *Mit den eingeführten Bezeichnungen gilt für $a \in \mathbb{Z}_q$, dass*

$$\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(a) = \mathrm{res}(F(X), A(X)),$$

wobei $\mathbb{Z}_q \cong \mathbb{Z}_p[X]/(F(X))$ und $\mathbb{Z}_p[X] \ni A(X) \coloneqq \sum_{\nu=0}^{n-1} a_\nu X^\nu$. Die Koeffizienten von $A(X)$ sind dabei durch die eindeutige Darstellung

$$\mathbb{Z}_q \ni a = a_0 + a_1\overline{X} + \cdots + a_{n-1}\overline{X}^{n-1}$$

gegeben, wobei $\overline{X} \coloneqq X + (F(X))$.

Beweis. Wir wiederholen zunächst die Definition der *Resultante* $\mathrm{res}(F(X), A(X))$ als Determinante der sog. *Sylvester-Matrix* (siehe [Bos06, 4.4]):

$$\mathrm{res}(F(X), A(X)) \coloneqq \begin{vmatrix}
c_n & \cdots & c_0 & & & \\
 & c_n & \cdots & c_0 & & \\
 & & c_n & c_{n-1} & \cdots & c_0 \\
a_{n-1} & \cdots & a_1 & a_0 & & \\
 & a_{n-1} & \cdots & a_1 & a_0 & \\
 & & a_{n-1} & \cdots & a_0 & \\
 & & & a_{n-1} & \cdots & a_0
\end{vmatrix},$$

wobei $F(X) = \sum_{\nu=0}^{n} c_\nu X^\nu$. Die obige Matrix hat $2n-1$ Zeilen (und Spalten). Man beachte, dass wir z. B. $a_{n-1} = 0$ nicht ausschließen. Es gilt

$$F(X) = \prod_{\nu=0}^{n} (X - \Sigma^\nu(\overline{X})).$$

[13] Weitere Möglichkeiten werden in [CF06, 12.8.5] beschrieben.
[14] [Har02]

Man rechnet nun einfach nach, dass

$$\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(a) = \prod_{\nu=0}^{n-1} \Sigma^\nu(a) = \prod_{\nu=0}^{n-1} A(\Sigma^\nu(\overline{X})) = \mathrm{res}(F(X), A(X)),$$

wobei die letzte Gleichheit aus [Bos06, 4.4, Korollar 9] folgt. □

Bemerkung. Anstatt die Determinante der Sylvester-Matrix zu berechnen, wird in [CF06, 12.8.5.c] vorgeschlagen, eine Variante von [Moe73] zu verwenden. Wir benutzen in der endgültigen Implementierung die von NTL bereitgestellte Methode zur Berechnung von Resultanten.

2.3 Newtoniteration und quadratischer Hensel-Lift

Wir stellen zunächst den *quadratischen Hensel-Lift* in einer sehr allgemeinen Fassung vor. Daraus leiten wir einen effizienten Algorithmus ab, um Inverse in $\mathbb{Z}_q$ zu berechnen. Danach zeigen wir, wie man Nullstellen von Polynomen mit Hilfe einer Newtoniteration liften kann. Schließlich formulieren wir eine Modifikation von Satoh[15], die wir später benötigen werden.

Sei im Folgenden R ein kommutativer Ring mit Einselement und $I \subset R$ ein Ideal. Für $x, y \in R$ schreiben wir $x \equiv y \pmod{I}$, wenn $x - y \in I$. Falls $I = zR$ mit $z \in R$, so schreiben wir kurz $x \equiv y \pmod{z}$.

Lemma 2.15. *Seien $f, g, h, a, b \in R$ mit*

$$f \equiv gh \pmod{I} \quad \textit{und} \quad ag + bh \equiv 1 \pmod{I}.$$

Dann existieren $G, H, A, B \in R$, so dass

$$f \equiv GH \pmod{I^2}, \quad AG + BH \equiv 1 \pmod{I^2},$$

$G \equiv g \pmod{I}$ und $H \equiv h \pmod{I}$.

Beweis. Wir definieren $I \ni m := f - gh$, sowie $G := g + bm$, $H := h + am$.
Dann gilt $G \equiv g \pmod{I}$, $H \equiv h \pmod{I}$ und

$$\begin{aligned} f - GH &= f - (g + bm)(h + am) \\ &= f - gh - m(ga + bh) - abm^2 \\ &\equiv m(1 - (ga + bh)) \pmod{I^2} \\ &\equiv 0 \pmod{I^2}. \end{aligned}$$

[15] [Sat00]

Mit $I \ni w \coloneqq 1 - (aG + bH)$ und $A \coloneqq a + aw$, $B \coloneqq b + bw$ gilt

$$\begin{aligned} AG + BH &= (a + aw)G + (b + bw)H \\ &= (aG + bH) + w(aG + bH) \\ &= (1 - w)(1 + w) \\ &= 1 - w^2 \\ &\equiv 1 \pmod{I^2}. \end{aligned}$$

□

Konkret setzen wir nun $R = \mathbb{Z}_q$ und $I = p\mathbb{Z}_q$. Sei $g \in \mathbb{Z}_q^*$ und $N \geq 1$. Wir möchten g^{-1} zur Präzision N bestimmen, d. h. ein $H \in \mathbb{Z}_q$ angeben, so dass $gH \equiv 1 \pmod{p^N}$. Mit $h \coloneqq \pi^{-1}(\pi(g)^{-1})$ gilt $gh \equiv 1 \pmod{p}$, denn $\pi(g)$ ist in $\mathbb{F}_q$ invertierbar. Außerdem haben wir die triviale Darstellung

$$h \cdot g + 0 \cdot h \equiv 1 \pmod{p}.$$

Wir können also Lemma 2.15 anwenden; die Rekursionsvorschrift lautet

$$h \leftarrow h + h(1 - gh).$$

Zusammenfassend ergibt sich Algorithmus 2 auf der nächsten Seite (vgl. [CF06, Algorithm 12.10]).

Bemerkung. Wir möchten in unseren Algorithmen stets mit einer möglichst *kleinen* Genauigkeit arbeiten. Angenommen $N = 33$, dann ist es effizienter diese Präzision mit dem Schema

$$1 \to \ 2 \to \ 3 \to \ 5 \to \ 9 \to 17 \to 33$$

zu erreichen, anstatt

$$1 \to \ 2 \to \ 4 \to \ 8 \to 16 \to 32 \to 33$$

zu benutzen (obwohl alle Zwischenergebnisse zur angegebenen Genauigkeit *korrekt* sind). Wir nehmen dabei an, dass das betrachtete Verfahren die Präzision in jedem Schritt verdoppelt. Am einfachsten ist es also, den Algorithmus rekursiv zu formulieren und die im rekursiven Aufruf benötigte Genauigkeit mit $\lceil N/2 \rceil$ festzulegen. Man erhält normalerweise eine nicht end-rekursive Programmskizze. Um die Notation in der iterativen Variante zu erleichtern, definieren wir die Abbildung

$$\text{precision} : \ \mathbb{N} \setminus \{0\} \to \mathcal{P}(\mathbb{N}), \ N \mapsto \{\lceil N \cdot 2^{-k} \rceil : \ k \in \mathbb{N}\} \setminus \{1\}.$$

Es gilt $\#\,\text{precision}(N) = \lceil \log_2 N \rceil$. Eine Initialisierung der Form

for $i \leftarrow \text{precision}(N)$ **do**

soll nun wie folgt interpretiert werden: Die Variable i durchläuft die Menge

$$\text{precision}(N) \subsetneq [2, N] \subset \mathbb{N}$$

in *aufsteigender* Reihenfolge. Die **for**-Schleife wird übersprungen, falls precision(N) keine Elemente enthält.

Algorithmus 2 Invertieren in $\mathbb{Z}_q$

Eingabe: $g \in \mathbb{Z}_q^*$ und $N \geq 1$.
Ausgabe: $h \in \mathbb{Z}_q$, so dass $gh \equiv 1 \pmod{p^N}$.

```
1: procedure INVERSE(g, N)
2:     h ← g^{-1} mod p
3:     for i ← precision(N) do
4:         h ← (h + h(1 − gh)) mod p^i
5:     end for
6:     return h
7: end procedure
```

Lemma 2.16 (Hensel)**.** *Sei $f(X) \in \mathbb{Z}_q[X]$ mit formaler Ableitung $f'(X)$ und $a_0 \in \mathbb{Z}_q$, so dass $f(a_0) \equiv 0 \pmod{p}$ und $f'(a_0) \not\equiv 0 \pmod{p}$. Dann besitzt $f(X)$ genau eine Nullstelle $a \in \mathbb{Z}_q$ mit $a \equiv a_0 \pmod{p}$ und die Folge*

$$a_{k+1} \coloneqq a_k - \frac{f(a_k)}{f'(a_k)} \in \mathbb{Z}_q, \qquad \text{(Newtoniteration)}$$

konvergiert für $k \geq 0$ quadratisch gegen a. Genauer gilt

(i) $f(a_k) \equiv 0 \pmod{p^{2^k}}$,

(ii) $a_{k+1} \equiv a_k \pmod{p^{2^k}}$ *und*

(iii) $\lim_{k\to\infty} a_k = a$.

Beweis. Zunächst besitzt $f(X)$ höchstens eine (einfache) Nullstelle $a \in \mathbb{Z}_q$ mit der geforderten Eigenschaft. Andernfalls hätte $\pi(f(X)) \in \mathbb{F}_q[X]$ eine mehrfache Nullstelle in $\pi(a_0) \in \mathbb{F}_q$ – nach [Bos06, 2.6, Satz 2] ein Widerspruch zur Voraussetzung $\pi(f'(a_0)) \neq 0$.

Wir beweisen die beiden Aussagen (i) und (ii) durch Induktion. Der Induktionsanfang $k = 0$ ist klar nach Voraussetzung. Sei nun $k \geq 0$. Wiederholte Division mit Rest in $\mathbb{Z}_q[X]$ liefert nach [Bos06, 2.1, Satz 4] eine Darstellung

$$f(X) = \sum_{\nu=0}^{d} c_\nu (X - a_k)^\nu, \tag{2.5}$$

wobei $d \coloneqq \deg(f(X)) \geq 1$ und $c_\nu \in \mathbb{Z}_q$ für alle $\nu \in [0, d]$. Es gilt $c_\nu = \frac{f^{(\nu)}(a_k)}{\nu!}$, d. h.

$$f(X) = \sum_{\nu=0}^{d} \frac{f^{(\nu)}(a_k)}{\nu!}(X - a_k)^\nu, \qquad \text{(Taylorentwicklung)}$$

wie man durch ν-faches Ableiten von (2.5) und Einsetzen verifiziert (unter Verwendung von $\operatorname{char} \mathbb{Z}_q = 0$). Daraus folgt

$$\begin{aligned} f(a_{k+1}) &= f(a_k) + f'(a_k)(a_{k+1} - a_k) + \sum_{\nu=2}^{d} c_\nu (a_{k+1} - a_k)^\nu \\ &= f(a_k) + f'(a_k)\left(\frac{-f(a_k)}{f'(a_k)}\right) + \left(\frac{-f(a_k)}{f'(a_k)}\right)^2 \cdot \sum_{\nu=2}^{d} c_\nu \left(\frac{-f(a_k)}{f'(a_k)}\right)^{\nu-2} \\ &\equiv f(a_k) + f'(a_k)\left(\frac{-f(a_k)}{f'(a_k)}\right) \pmod{p^{2^{k+1}}} \\ &\equiv 0 \pmod{p^{2^{k+1}}}. \end{aligned}$$

Dabei verwenden wir, dass nach Induktionsannahme

$$f(a_k)^2 \equiv 0 \pmod{p^{2 \cdot 2^k}}$$

und $f'(a_k)^{-1} \in \mathbb{Z}_q$, denn $a_k \equiv a_0 \pmod{p}$ und damit $f'(a_k) \not\equiv 0 \pmod{p}$. Per Definition gilt $a_{k+2} \equiv a_{k+1} \pmod{p^{2^{k+1}}}$ und wir sind fertig.

Die Folge $(a_k)_{k \in \mathbb{N}}$ schreiben wir als Folge der Partialsummen

$$a_0,\ a_0 + (a_1 - a_0),\ a_0 + (a_1 - a_0) + (a_2 - a_1), \ldots$$

und erhalten eine Cauchyfolge in $\mathbb{Z}_q$, da die einzelnen Summenglieder nach (ii) eine Nullfolge bilden und die Norm auf $\mathbb{Q}_q$ nicht-archimedisch ist. Nach Satz 2.11 existiert also

$$a \coloneqq \lim_{k \to \infty} a_k \in \mathbb{Q}_q,$$

wobei der Grenzwert nach [Ser62, II., Proposition 1] bereits in $\mathbb{Z}_q$ liegt. Schließlich folgt aus (i), dass $f(a) = 0$. □

Aus dem Beweis ergibt sich unmittelbar, dass ein Iterationsschritt die Genauigkeit $N \geq 1$ einer beliebigen Approximation $b \in \mathbb{Z}_q$ mit $f(b) \equiv 0 \pmod{p^N}$ und $f'(b) \in \mathbb{Z}_q^*$ verdoppelt, d. h. für

$$c \coloneqq b - \frac{f(b)}{f'(b)} \in \mathbb{Z}_q$$

gilt $f(c) \equiv 0 \pmod{p^{2N}}$ und $c \equiv b \pmod{p^N}$. Wie bereits erwähnt, sollte in einer effizienten Implementierung stets mit einer möglichst *kleinen* Genauigkeit gerechnet werden: Es genügt

$$\mathbb{Z}_q \ni s \coloneqq f'(b)^{-1} \bmod p^N \tag{2.6}$$

zu verwenden, denn mit $f(b) = rp^N$, $f'(b)^{-1} = s + tp^N$ $(r, t \in \mathbb{Z}_q)$ gilt

$$c = b - \frac{f(b)}{f'(b)} = b - (s + tp^N)rp^N \equiv b - srp^N \pmod{p^{2N}}.$$

Als Ergebnis formulieren wir Algorithmus 3.

Algorithmus 3 Newtoniteration

Eingabe: $f(X) \in \mathbb{Z}_q[X]$, $a \in \mathbb{Z}_q$, $N \geq 1$, wobei $f(a) \equiv 0 \pmod{p}$ und $f'(a) \in \mathbb{Z}_q^*$.
Ausgabe: $b \in \mathbb{Z}_q$, so dass $f(b) \equiv 0 \pmod{p^N}$ und $b \equiv a \pmod{p}$.

```
1: procedure NEWTON(f, a, N)
2:     b ← a mod p
3:     for i ← precision(N) do
4:         s ← f'(b)^{-1} mod p^{⌈i/2⌉}        ▷ INVERSE(f'(b) mod p^{⌈i/2⌉}, ⌈i/2⌉)
5:         b ← (b − s · f(b)) mod p^i
6:     end for
7:     return b
8: end procedure
```

Bei genauer Betrachtung bemerkt man, dass Algorithmus 2 auf Seite 42 im Grunde eine Newtoniteration ist: Sei $g \in \mathbb{Z}_q^*$, dann gilt für $\mathbb{Z}_q[X] \ni f(X) \coloneqq gX - 1$, dass

$$\text{INVERSE}(g, N) = \text{NEWTON}(f, g^{-1} \bmod p, N),$$

wobei wir nach (2.6) Zeile 4 streichen und Zeile 5 durch

$$b \leftarrow b(1 - f(b)) \bmod p^i$$

ersetzen.

Wir möchten mit Hilfe der obigen Überlegungen auf elegante Weise Quadratwurzeln in $\mathbb{Z}_{p^n}$ berechnen ($p \neq 2$). Seien also $d, e \in \mathbb{Z}_q$ mit $d = e^2$. Wir nehmen zudem an, dass d (und damit auch e) in $\mathbb{Z}_q$ invertierbar ist. Dies stellt allerdings keine echte Einschränkung dar, denn mit $d = p^k d'$, $k \in \mathbb{N}$, $d' \in \mathbb{Z}_q^*$ gilt $2 \mid k$ und d' ist ein Quadrat in $\mathbb{Z}_q^*$.

Gesucht ist für $N \geq 1$ die Approximation $e \bmod p^N$. Wir können diesen Wert bereits mit dem Aufruf

$$\text{NEWTON}(X^2 - d, e \bmod p, N)$$

berechnen lassen, wobei wir die Startnäherung $e \bmod p$ nach [BSS99, Algorithm II.8] bestimmen.

Einen optimierten Algorithmus erhält man folgendermaßen: Statt $e \bmod p^N$ berechnen wir $e^{-1} \bmod p^N$ als Nullstelle von $\mathbb{Z}_q[X] \ni f(X) \coloneqq dX^2 - 1$. Der Vorteil ist, dass man Zeile 4 durch

$$s \leftarrow 2^{-1} b \bmod p^{\lceil i/2 \rceil}$$

ersetzen kann. Es genügt natürlich $2^{-1} \bmod p^{\lceil N/2 \rceil}$ nur *einmal* zu berechnen (z. B. mit dem erweiterten euklidischen Algorithmus, siehe [CF06, Algorithm 10.42]). Wir fassen unsere Anpassungen in Algorithmus 4 zusammen:

Algorithmus 4 Wurzelziehen in $\mathbb{Z}_q$

Eingabe: $d, e \in \mathbb{Z}_{p^n}^*$, $N \geq 1$, wobei $p \neq 2$ und $e^2 \equiv d \pmod{p}$.
Ausgabe: $b \in \mathbb{Z}_q$, so dass $b^2 \equiv d \pmod{p^N}$ und $b \equiv e \pmod{p}$.

```
1: procedure SQRT(d, e, N)
2:     b ← e^{-1} mod p
3:     c ← 2^{-1} mod p^{⌈N/2⌉}
4:     for i ← precision(N) do
5:         b ← b(1 + c(1 − db^2)) mod p^i
6:     end for
7:     b ← db mod p^N
8:     return b
9: end procedure
```

Lemma 2.17 (Satoh)**.** *Sei $p \geq 3$, $N \geq 1$ und $\mathbb{Z}_{p^n}[X] \ni F(X) = D(X)E(X)$ mit $F'(X) \equiv 0 \pmod{p}$ und $F'(X) \not\equiv 0 \pmod{p^2}$. Sei weiter $h(X) \in \mathbb{Z}_{p^n}[X]$ ein normiertes Polynom mit den folgenden Eigenschaften:*

a) $\pi(h)$ ist quadratfrei und teilerfremd zu $\pi(p^{-1}F')$.

b) $h(X) \equiv E(X) \pmod{p^N}$.

c) $F(X) \equiv g(X)h(X) \pmod{p^{N+1}}$.

Dann gilt für

$$H(X) \coloneqq h(X) + \left(\frac{F(X)h'(X)}{F'(X)} \bmod h(X) \right), \tag{2.7}$$

dass $H(X) \equiv h(X) \pmod{p^N}$, $H(X) \equiv E(X) \pmod{p^{2N}}$ und

$$F(X) \equiv G(X)H(X) \pmod{p^{2N+1}}.$$

Beweis. Wir verwenden die Notation von [BSS05, Lemma VI.6]. Satoh beweist die Aussage in [Sat00, Lemma 2.1]. □

Bemerkung. Da $\pi(h)$ teilerfremd zu $\pi(p^{-1}F)$ ist, haben wir eine Darstellung[16]

$$a(X)(F'(X)/p) + b(X)h(X) \equiv 1 \pmod{p}.$$

Diese können wir nach Lemma 2.15 im Ring $\mathbb{Z}_{p^n}[X]$ zur Genauigkeit N liften:

$$A(X)(F'(X)/p) + B(X)h(X) \equiv 1 \pmod{p^N}.$$

Nach c) gibt es $g(X), c(X) \in \mathbb{Z}_{p^n}[X]$, so dass $F(X) = g(X)h(X) + c(X)p^{N+1}$. Also gilt

$$H(X) \equiv h(X) + \big(h'(X)A(X)c(X)p^N \bmod h(X)\big) \pmod{p^{2N+1}},$$

wobei man $A(X) \bmod h(X)$ wie folgt rekursiv berechnet (siehe Lemma 2.15):

$$A(X) \bmod h(X) \leftarrow (2A(X) - A(X)^2(F'(X)/p)) \bmod h(X).$$

2.4 Modulare Polynome

Das (klassische) *n-te modulare Polynom* $\Phi_n(X, Y) \in \mathbb{Z}[X, Y]$ spielt für Satohs Algorithmus eine wichtige Rolle. Wir benutzen es, um den (noch zu definierenden) kanonischen Lift einer elliptischen Kurve $E/\mathbb{F}_q$ zu berechnen. Auch ein anderes Verfahren zur Bestimmung der Gruppenordnung, der Schoof-Elkies-Atkin (SEA) Algorithmus, benutzt modulare Polynome ([BSS99, VII.], [Sch95]). Bereits der ursprüngliche Algorithmus[17] von Schoof (veröffentlicht 1985) war wesentlich effizienter als die zur damaligen Zeit bekannten Methoden, wie z. B. „Baby Step, Giant Step“ (siehe [Was08, 4.3.4]).

Die Definition des modularen Polynoms gehört in den Bereich der elliptischen Kurven über $\mathbb{C}$. Eine detaillierte Einführung in dieses (umfangreiche) Themengebiet ist hier jedoch nicht möglich. Wir fassen daher nur kurz einige wichtige Aussagen zusammen. Alle Beweise und ausführliche Erläuterungen findet der Leser in [Lan73], [KK98], [FB06] und [Was08].

Ein *Gitter* in $\mathbb{C}$ ist eine Untergruppe $\Lambda \subset \mathbb{C}$ der Gestalt

$$\Lambda = \mathbb{Z}\omega_1 + \mathbb{Z}\omega_2,$$

wobei $\omega_1, \omega_2 \in \mathbb{C}$ reell linear unabhängig sind. Zwei Gitter Λ, Λ' heißen *ähnlich*, wenn ein $\alpha \in \mathbb{C}^*$ existiert, so dass $\Lambda = \alpha\Lambda'$. Jedes Gitter Λ ist ähnlich zu einem Gitter $\Lambda_\tau := \mathbb{Z} + \mathbb{Z}\tau$ mit $\tau \in \mathbb{H} := \{z \in \mathbb{C} : \operatorname{Im} z > 0\}$.

[16] [For96, 4.11]
[17] [Sch85]

Sei $f:\ \mathbb{C} \to \mathbb{C} \cup \{\infty\}$ eine meromorphe Funktion. Wir nennen $\omega \in \mathbb{C}$ *Periode* von f, falls

$$f(z+\omega) = f(z) \quad \text{für alle } z \in \mathbb{C}.$$

Die Menge aller Perioden von f ist entweder ein Gitter Λ oder $\mathbb{Z}\omega$ (mit $\omega \in \mathbb{C}$). Im ersten Fall heißt f *doppelt periodisch* (bzgl. dem Gitter Λ) oder kurz *elliptisch*. Aus dem Satz von Liouville folgt unmittelbar, dass jede holomorphe elliptische Funktion konstant ist. Man interpretiert eine elliptische Funktion f als Funktion auf der Faktorgruppe $\mathbb{C}/\Lambda$:

$$\tilde{f}:\ \mathbb{C}/\Lambda \to \mathbb{C} \cup \{\infty\}, \quad \tilde{f}(z+\Lambda) \coloneqq f(z).$$

Diese Definition ist unabhängig von der Wahl des Repräsentanten, da für $z+\Lambda = y+\Lambda$ (also $z-y \in \Lambda$) gilt, dass $f(y) = f(y+z-y) = f(z)$. Wenn keine Verwechslungen zu befürchten sind, schreibt man statt $\tilde{f}$ einfach wieder f.

Als *Eisensteinreihe* der Ordnung $k \geq 3$ zum Gitter Λ bezeichnet man die folgende absolut konvergente Reihe:

$$G_k(\Lambda) \coloneqq \sum_{0 \neq \omega \in \Lambda} \frac{1}{\omega^k}.$$

Sie verschwindet für alle ungeraden k. Wir definieren die *Weierstraßsche $\wp$-Funktion* zum Gitter Λ durch

$$\wp(z) \coloneqq \frac{1}{z^2} + \sum_{0 \neq \omega \in \Lambda} \left\{ \frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right\}.$$

$\wp(z)$ ist bzgl. Λ doppelt-periodisch. Für $0 < |z| < \min\{|\omega|:\ 0 \neq \omega \in \Lambda\}$ gilt

$$\wp(z) = \frac{1}{z^2} + \sum_{k=1}^{\infty} (2k+1) G_{2k+2} z^{2k}.$$

Außerdem lässt sich *jede* elliptische Funktion f schreiben als

$$f = R_1(\wp) + R_2(\wp) \cdot \wp', \quad \text{mit } R_1, R_2 \in \mathbb{C}(X).$$

Der Torus $\mathbb{C}/\Lambda$ ist isomorph zu

$$E/\mathbb{C}:\ y^2 = x^3 - \frac{g_2(\Lambda)}{4} x - \frac{g_3(\Lambda)}{4},$$

wobei $g_2(\Lambda) \coloneqq 60 G_4(\Lambda)$ und $g_3(\Lambda) \coloneqq 140 G_6(\Lambda)$.

Genauer gilt $\Delta(\Lambda) \coloneqq \Delta(E) = g_2(\Lambda)^3 - 27g_3(\Lambda)^2 \neq 0$ und die Abbildung

$$\mathbb{C}/\Lambda \to E(\mathbb{C}),\ z \mapsto \begin{cases} \mathcal{O} & \text{falls } z = 0, \\ \big(\wp(z), \frac{1}{2}\wp'(z)\big) & \text{falls } z \neq 0, \end{cases}$$

ist ein bijektiver Gruppenhomomorphismus. Es gilt

$$\mathrm{End}(E) \cong \{\alpha \in \mathbb{C} :\ \alpha\Lambda \subset \Lambda\}.$$

Umgekehrt gibt es zu jeder elliptischen Kurve $E/\mathbb{C} :\ y^2 = x^3 + ax + b$ ein Gitter Λ, mit $a = -\frac{1}{4}g_2(\Lambda)$ und $b = -\frac{1}{4}g_3(\Lambda)$. Wir definieren die *absolute Invariante* von Λ als j-Invariante der entsprechenden elliptischen Kurve, d. h.

$$j(\Lambda) \coloneqq \frac{1728 \cdot g_2(\Lambda)^3}{\Delta(\Lambda)}.$$

Die so erhaltene (holomorphe) Funktion $j :\ \mathbb{H} \to \mathbb{C},\ \tau \mapsto j(\Lambda_\tau)$ ist surjektiv und besitzt eine Fourierentwicklung mit ganzzahligen Koeffizienten:

$$j(\tau) = q^{-1} + 744 + 196884q + 21493760q^2 + 864299970q^3 + 20245856256q^4 + \cdots,$$

wobei $q = e^{2\pi i\tau}$. Verschiedene Berechnungsmethoden dieser Koeffizienten werden in [BK03] vorgestellt.

Die Gruppe Γ aller Transformationen der oberen Halbebene der Gestalt

$$\tau \mapsto \frac{a\tau + b}{c\tau + d}, \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}(2,\mathbb{Z}) \coloneqq \{M \in \mathrm{M}(2 \times 2, \mathbb{Z}) :\ \det(M) = 1\}$$

heißt (elliptische) *Modulgruppe*. Sie wird von $\tau \mapsto \tau + 1$ und $\tau \mapsto -\frac{1}{\tau}$ erzeugt. Es gilt $\Gamma \cong \mathrm{SL}(2,\mathbb{Z})/\{\pm 1\}$. Die absolute Invariante j ist invariant unter der Modulgruppe, d. h.

$$(j \circ M)(\tau) = j(\tau) \quad \text{für alle } M \in \Gamma. \tag{2.8}$$

Umgekehrt folgt aus $j(\tau_1) = j(\tau_2)$, dass $\tau_2 = M(\tau_1)$ für ein $M \in \Gamma$. Allgemein definieren wir für $L = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in \mathrm{M}(2 \times 2, \mathbb{Z})$ mit $\det(L) > 0$ die Abbildung

$$L :\ \mathbb{H} \to \mathbb{H},\ \tau \mapsto \frac{a\tau + b}{c\tau + d}.$$

Es gilt $(KL)(\tau) = K(L(\tau))$ (für $K \in \mathrm{M}(2 \times 2, \mathbb{Z})$ mit $\det(K) > 0$).

Lemma 2.18. *Sei $n \geq 1$ und*

$$S_n \coloneqq \left\{ S = \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} \in \mathrm{M}(2 \times 2, \mathbb{Z}) :\ \det(S) = n \ \wedge\ b \in [0, d-1] \right\}.$$

Dann gibt es zu jeder Matrix $M \in \mathrm{M}(2 \times 2, \mathbb{Z})$ mit $\det(M) = n$ genau eine Matrix $S \in S_n$, so dass

$$MS^{-1} \in \mathrm{SL}(2,\mathbb{Z}).$$

Beweis. [Was08, Lemma 10.10] □

Man kann zeigen[18], dass

$$\#S_n = n \prod_{\substack{p \in \mathbb{P} \\ p \mid n}} \left(1 + \frac{1}{p}\right).$$

Für eine Primzahl p gilt also $\#S_p = p + 1$ und

$$S_p = \left\{ \begin{pmatrix} p & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & p \end{pmatrix}, \ldots, \begin{pmatrix} 1 & p-1 \\ 0 & p \end{pmatrix} \right\}.$$

Die Funktion $(j \circ S)(\tau)$ ist für $S \in S_n$ holomorph auf $\mathbb{H}$. Wir definieren folgendes Polynom über dem Ring der holomorphen Funktionen auf $\mathbb{H}$ (vgl. [FB06, III., §3]):

$$F_n(X, \tau) \coloneqq \prod_{S \in S_n} (X - (j \circ S)(\tau)) = \sum_{k=0}^{\#S_n} a_k(\tau) X^k.$$

Lemma 2.19. *Die Koeffizienten a_k sind invariant unter der Modulgruppe.*

Beweis. Sei $M \in \mathrm{SL}(2, \mathbb{Z})$. Für $S \in S_n$ gilt $\det(SM) = n$, also folgt mit Lemma 2.18 $SM = A_S M_S$, wobei $A_S \in \mathrm{SL}(2, \mathbb{Z})$ und $M_S \in S_n$. Dabei ist M_S eindeutig bestimmt. Wir können also die Abbildung $S_n \to S_n,\ S \mapsto M_S$ definieren. Wir zeigen, dass diese Abbildung bijektiv ist und damit lediglich die Elemente von S_n vertauscht: Sei nämlich $M_G = M_H$ für $G, H \in S_n$. Dann gilt $A_G^{-1} G M = A_H^{-1} H M$, also

$$A_G^{-1} G = A_H^{-1} H.$$

Mit Lemma 2.18 (Eindeutigkeit!) folgt $G = H$ und da $\#S_n < \infty$ folgt aus der Injektivität bereits die Behauptung.

Damit gilt unter Verwendung von (2.8):

$$\begin{aligned} F_n(X, M(\tau)) &= \prod_{S \in S_n} (X - j(SM(\tau))) = \prod_{S \in S_n} (X - (j \circ A_S)(M_S(\tau))) \\ &= \prod_{S \in S_n} (X - j(M_S(\tau))) \\ &= \prod_{S \in S_n} (X - j(S(\tau))) \\ &= F_n(X, \tau). \end{aligned}$$

Da F_n unter $\tau \mapsto M(\tau)$ invariant ist, gilt dies auch für die Koeffizienten a_k. □

[18] [Lan73, 5, §1]

Lemma 2.20. *Für jedes $k \in [0, \#S_n]$ existiert ein $m \in \mathbb{Z}$, so dass*

$$a_k(\tau) \in q^m \mathbb{Z}[\![q]\!],$$

wobei $q = e^{2\pi i \tau}$.

Beweis. [Was08, Lemma 10.12]. □

Proposition 2.21. *Sei $f : \mathbb{H} \to \mathbb{C}$ holomorph und invariant unter der Modulgruppe. Existiert $m \in \mathbb{Z}$, so dass $f(\tau) \in q^m \mathbb{Z}[\![q]\!]$, dann gilt $f \in \mathbb{Z}[j]$.*

Beweis. [Was08, Proposition 10.14]. □

Mit Hilfe der obigen Aussagen haben wir also gezeigt, dass

$$F_n(X, \tau) = \prod_{S \in S_n} (X - (j \circ S)(\tau)) \in \mathbb{Z}[X, j].$$

Wir fassen die absolute Invariante j als zweite Unbestimmte auf, und erhalten das *n-te modulare Polynom*

$$\Phi_n(X, Y) \in \mathbb{Z}[X, Y].$$

In unseren Anwendungen wird $n \geq 5$ stets eine Primzahl sein.

Satz 2.22. *Sei $n \geq 1$ und p eine Primzahl. Dann gilt*

a) $\Phi_n(X, Y) = \Phi_n(Y, X)$ *und*

b) $\Phi_p(X, Y) \equiv (X - Y^p)(X^p - Y) \pmod{p}$ (Kronecker-Relation).

Beweis. [Lan73, 5, §2]. □

Beispiel 2.23.

$$\begin{aligned}\Phi_2(X, Y) = {} & X^3 + Y^3 - X^2Y^2 + 1488(X^2Y + XY^2) - 162000(X^2 + Y^2) \\ & + 40773375XY + 8748000000(X + Y) - 157464000000000.\end{aligned}$$

Sei $\Phi_p(X, Y) = \sum_{r,s=0}^{p+1} c_{rs} X^r Y^s$. Das enorme Wachstum der Koeffizienten wird in [BSS99, III.8] abgeschätzt: Mit $h(\Phi_p) \coloneqq \ln(\ \max\{|c_{rs}| : \ r, s \in [0, p+1]\}\ )$ gilt

$$h(\Phi_p) = 6(p+1)\left(\left(1 - \frac{2}{p}\right)\ln(p) + O(1)\right).$$

Beispielwerte entnimmt man Tabelle 4 auf der nächsten Seite.

Tabelle 4: Logarithmen der betragsmäßig größten Koeffizienten von $\Phi_p(X, Y)$

p	$h(\Phi_p)$	$\pi(p)$
5	108,6	3
7	152,4	4
11	291,5	5
13	343,5	6
⋮	⋮	⋮
127	5 197,9	31

Anleitungen zur konkreten Berechnung modularer Polynome findet man in [CL05], [Her75], [Eng07] und [BCRS99]. Wir verwenden die von Rubinstein veröffentlichten[19] Koeffizienten der ersten 71 modularen Polynome, kennen also Φ_p für alle Primzahlen $p \leq 351$. Weil wir lediglich $\Phi_p(X, Y) \bmod p^N$ für ein $N \geq 1$ benötigen, können wir den Speicheraufwand deutlich reduzieren. Weitere Informationen findet man im nächsten Abschnitt. Wir verzichten sogar darauf die Symmetrie auszunutzen und speichern die $c_{rs} \bmod p^N$ in einer Matrix mit $(p+2)^2$ Einträgen (statt $\frac{(p+2)(p+3)}{2}$). Dies vereinfacht die Verarbeitung in der Implementierung.

2.5 Vercauterens Algorithmus

Definition 2.24. Sei $E: \ y^2 = x^3 + ax + b$ eine gewöhnliche[20] elliptische Kurve über $\mathbb{F}_q$ und $\mathcal{E}: \ y^2 = x^3 + Ax + B$ eine elliptische Kurve über $\mathbb{Q}_q$, wobei $A, B \in \mathbb{Z}_q$. $\mathcal{E}$ heißt *kanonischer Lift von* E, wenn folgende Bedingungen erfüllt sind:

(i) $(\pi(A), \pi(B)) = (a, b)$.

(ii) Der durch Reduktion mod p induzierte Ringhomomorphismus $\mathrm{End}(\mathcal{E}) \to \mathrm{End}(E)$ ist ein Isomorphismus:

$$\mathrm{End}(\mathcal{E}) \cong \mathrm{End}(E). \tag{2.9}$$

Nach [Deu41] existiert zu einer gegebenen (gewöhnlichen) Kurve $E/\mathbb{F}_q$ stets ein – bis auf Isomorphie eindeutig bestimmter – kanonischer Lift, den wir mit $E^{\uparrow}$ bezeichnen. Wir erinnern daran[21], dass E gewöhnlich ist, wenn $j(E) \notin \mathbb{F}_{p^2}$. Obige Charakterisierung des kanonischen Lifts ist nur eine von mehreren Möglichkeiten (siehe [CF06, Theorem 17.29]).

[19] [Rub]
[20] Satz 1.20
[21] Proposition 1.29 d)

Sei $\phi \in \mathrm{End}(E)$ eine Isogenie. Dann bezeichnen wir das Bild von ϕ unter (2.9) mit $\phi^\uparrow \in \mathrm{End}(E^\uparrow)$. Diese Notation verwenden wir auch in der folgenden Situation:

Satz 2.25. *Seien E und E' gewöhnliche elliptische Kurven über $\mathbb{F}_q$ und $\psi : E \to E'$ eine Isogenie. Dann gilt*

$$\mathrm{Hom}(E, E') \cong \mathrm{Hom}(E^\uparrow, E'^\uparrow) \quad \textit{und} \quad \deg(\psi) = \deg(\psi^\uparrow).$$

Beweis. [Mes72, Corollary 3.4] und [Sil94, II., Proposition 4.4]. □

Korollar 2.26. *Sei $\psi \in \mathrm{Hom}(E, E')$ und $\phi \in \mathrm{End}(E)$. Dann ist $\widehat{\psi}^\uparrow$ die duale Isogenie von $\psi^\uparrow$ und es gilt*

$$\mathrm{Tr}(\phi) = \mathrm{Tr}(\phi^\uparrow).$$

Beweis. Sei $\lambda \in \mathrm{Hom}(E'^\uparrow, E^\uparrow)$ die duale Isogenie von $\psi^\uparrow$. Es gilt $\psi \circ \widehat{\psi} = \deg(\psi) \in \mathbb{N}$. Daraus folgt mit Satz 2.25

$$\psi^\uparrow \circ \lambda = \deg(\psi^\uparrow) = \deg(\psi) = \deg(\psi)^\uparrow = (\psi \circ \widehat{\psi})^\uparrow = \psi^\uparrow \circ \widehat{\psi}^\uparrow.$$

Aus der Eindeutigkeit der Zerlegung[22] folgt $\lambda = \widehat{\psi}^\uparrow$.

Mit $\mathrm{Tr}(\phi) \in \mathbb{Z}$ gilt

$$\mathrm{Tr}(\phi) = \mathrm{Tr}(\phi)^\uparrow = (\phi + \widehat{\phi})^\uparrow = \phi^\uparrow + \widehat{\phi}^\uparrow.$$

Aus der ersten Behauptung folgt nun $\phi^\uparrow + \widehat{\phi}^\uparrow = \mathrm{Tr}(\phi^\uparrow)$ und wir sind fertig. □

Proposition 2.27. *Sei $\psi \in \mathrm{Hom}_{\mathbb{F}_q}(E, E')$. Dann ist $\psi^\uparrow$ über $\mathbb{Q}_q$ definiert. Insbesondere gilt*

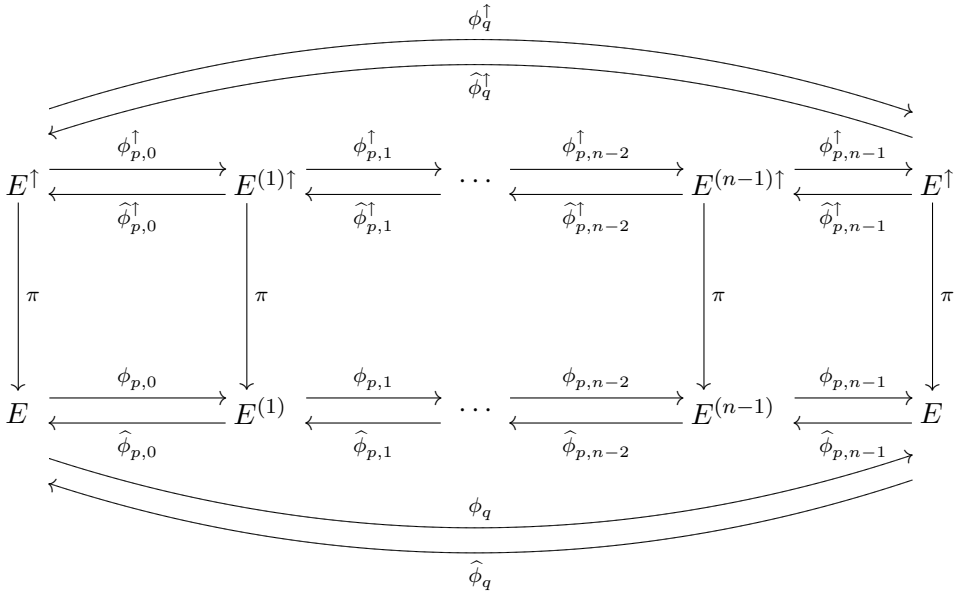
$$\phi_q^\uparrow \in \mathrm{End}_{\mathbb{Q}_q}(E^\uparrow).$$

Beweis. [Mad04, Lemma 2.2.47]. □

Ausgehend vom Zyklus (1.11) erhält man für eine gewöhnliche elliptische Kurve $E/\mathbb{F}_{p^n}$

[22] Satz 1.15

das folgende Diagramm, das die Grundlage für alle weiteren Überlegungen darstellt:

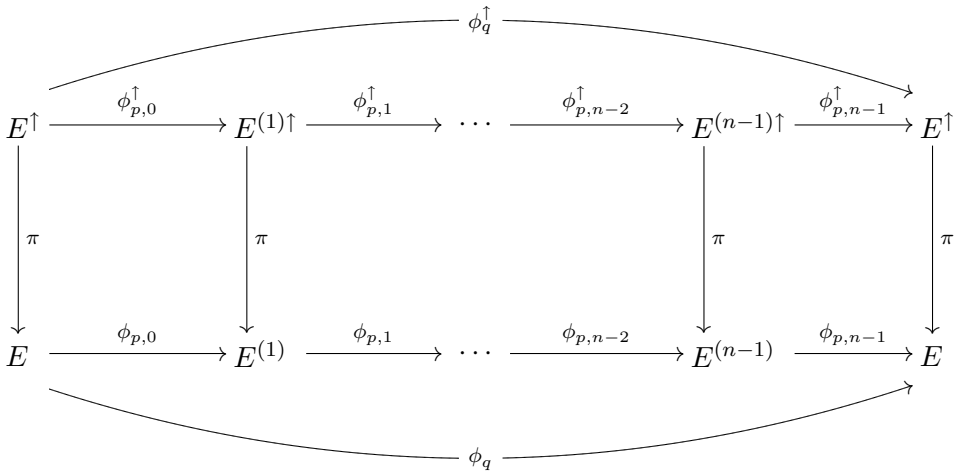


(2.10)

Dabei gilt für $i \in \{1, 2, 3, 4, 6\}$ und $k \in [0, n-1]$

$$a_i(E^{(k)\uparrow}) = \Sigma^k(a_i(E^\uparrow)). \tag{2.11}$$

In diesem Abschnitt betrachten wir allerdings nur die vereinfachte Variante:



(2.12)

Wir möchten die Gruppenordnung $\#E(\mathbb{F}_q)$ nach Satz 1.30 berechnen. In Charakteristik p können wir die Spur des Frobenius-Endomorphismus nur modulo p bestimmen[23]

[23] Proposition 1.33

– selbst wenn $\mathrm{Tr}(\phi_q)$ eine ganze Zahl ist. Die vorgestellten Zusammenhänge bieten nun die Möglichkeit, in Charakteristik 0 nach einer Lösung zu suchen!

Angenommen wir wissen, wie wir die Spur des gelifteten Frobenius bestimmen. Dann können wir den *exakten* Wert berechnen, indem wir einfach mit genügend großer Genauigkeit arbeiten. Nach der Abschätzung von Hasse[24] wählen wir ein passendes $N \geq 1$ und berechnen

$$\mathrm{Tr}(\phi_q^{\uparrow}) \bmod p^N.$$

Dies ist das Grundprinzip *aller* p-adischen Punktezählalgorithmen.

Wir fassen noch einmal zusammen: Es gilt

$$\mathrm{Tr}(\phi_q) = \mathrm{Tr}(\phi_q^{\uparrow}) = \mathrm{Tr}(\widehat{\phi_q^{\uparrow}}). \tag{2.13}$$

Es genügt also $\mathrm{Tr}(\phi_q^{\uparrow})$ *oder* $\mathrm{Tr}(\widehat{\phi_q^{\uparrow}})$ zu berechnen. Satoh entscheidet sich aus Gründen, die wir erst im nächsten Kapitel erläutern, für die zweite Variante.

Ohne konkrete Werte können wir natürlich nicht arbeiten. Zunächst konstruieren wir die j-Invariante von $E^{\uparrow}$ mit Hilfe des folgenden

Satz 2.28 (Lubin, Serre, Tate)**.** *Sei $E/\mathbb{F}_q$ eine elliptische Kurve, wobei $j(E) \notin \mathbb{F}_{p^2}$. Dann gibt es genau ein $J \in \mathbb{Z}_q$ mit*

$$\pi(J) = j(E) \quad \textit{und} \quad \Phi_p(J, \Sigma(J)) = 0.$$

Es gilt $J = j(E^{\uparrow})$.

Beweis. [LST64, 8., Theorem 3]. □

Um Berechnungen von Σ zu vermeiden, stellt man ein Gleichungssystem mit n Unbekannten über $\mathbb{Z}_q$ auf, dessen eindeutige Lösung die j-Invarianten der kanonischen Lifts von $E, \ldots, E^{(n-1)}$ sind, denn mit (2.11) gilt

$$\Sigma(j(E^{(i)\uparrow})) = j(E^{(i+1)\uparrow}).$$

Man liftet also nicht nur eine j-Invariante, sondern alle j-Invarianten des Zyklus (2.12) *simultan*. Als Startnäherung verwendet man

$$\left(j(E), j(E^{(1)}), \ldots, j(E^{(n-1)})\right) = \left(j(E), j(E)^p, \ldots, j(E)^{p^{n-1}}\right).$$

Die Idee dieses Verfahrens geht auf Satoh zurück; weitere Details sind in [Sat00] beschrieben. Das multivariate Newtonverfahren zur Lösung des Gleichungssystems wird

[24] Satz 1.31

in [FGH00] vorgestellt. Wir gehen hier einen anderen Weg und benutzen stattdessen die genauso effiziente (aber einfachere) Methode von Vercauteren[25].

In Satohs ursprünglichem Algorithmus benötigt man tatsächlich *alle* j-Invarianten des Zyklus (2.12). In unserer Variante ist dies nicht mehr der Fall. Allerdings führt gerade die Elimination der Frobenius-Substitution zu einem schnellen Algorithmus. Wir beschreiben nun detailliert die Berechnung von $j(E^{\uparrow})$.

Kennt man für $N \geq 1$ eine Approximation $J \in \mathbb{Z}_q$ mit $J \equiv j(E^{(i)\uparrow}) \pmod{p^N}$, so kann man $J' \in \mathbb{Z}_q$ mit $J' \equiv j(E^{(i+1)\uparrow}) \pmod{p^N}$ nach Lemma 2.16 berechnen:

Um die Notation zu erleichtern, bezeichne im Folgenden $\mathbb{Z}_q \ni j_i \coloneqq \pi^{-1}(j(E^{(i)}))$ die j-Invariante der entsprechenden elliptischen Kurve über $\mathbb{F}_q$. Wir weisen außerdem darauf hin, dass alle Indizes in diesem Abschnitt stets zwischen 0 und $n-1$ liegen, eventuell erst nach Reduktion modulo n (vgl. Definition 1.28). Nach Satz 2.28 gilt

$$\Phi_p(J, j_{i+1}) \equiv \Phi_p(j_i, j_{i+1}) \equiv 0 \pmod{p}.$$

Die Kronecker-Relation[26] besagt, dass

$$\frac{\partial \Phi_p}{\partial X}(j_i, j_{i+1}) \equiv (X^p - Y)\,(j_i, j_{i+1}) \equiv 0 \pmod{p}$$

und

$$\frac{\partial \Phi_p}{\partial Y}(j_i, j_{i+1}) \equiv (Y^p - X)\,(j_i, j_{i+1}) \not\equiv 0 \pmod{p},$$

denn $j_{i+1} = j_i^p$ und $j_{i+1}^p = j_i^{p^2} \neq j_i$ (da $j(E) \notin \mathbb{F}_{p^2}$ nach Voraussetzung). Mit einer Newtoniteration können wir also die eindeutige Nullstelle J' von $\Phi_p(J, Y) \in \mathbb{Z}_q[Y]$ mit $J' \equiv j_{i+1} \pmod{p}$ berechnen und es gilt

$$J' \equiv j(E^{(i+1)\uparrow}) \pmod{p^N}.$$

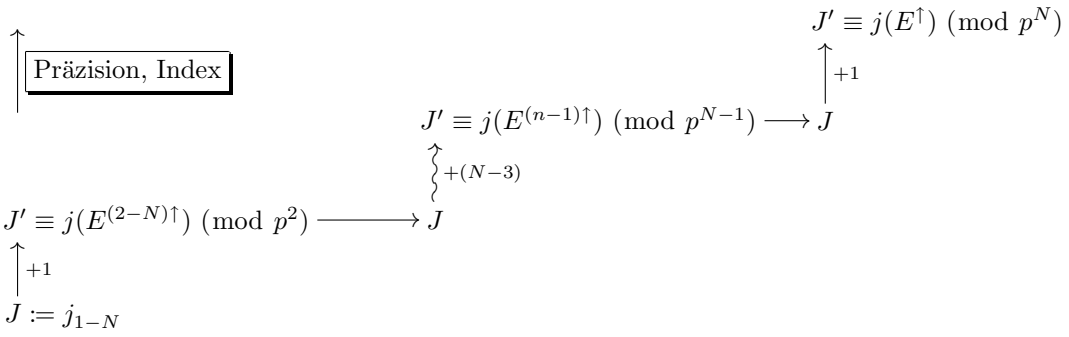
Man beachte, dass im Allgemeinen weder $J' = j(E^{(i+1)\uparrow})$ noch $J' = \Sigma(J)$ gilt.

Wie berechnet man nun aber *eine* j-Invariante zur Präzision N? Das obige Verfahren besitzt nach Lemma 2.29 die erstaunliche Eigenschaft, dass die Genauigkeit *wächst*:

$$J' \equiv j(E^{(i+1)\uparrow}) \pmod{p^{N+1}}. \tag{2.14}$$

[25] [Ver03, 3.4]
[26] Satz 2.22 b)

Wir konstruieren die Nullstelle J' also bis zur Präzision $N+1$ und können $j(E^{\uparrow})$ (zu einer beliebigen Genauigkeit) gemäß dem folgenden Diagramm iterativ berechnen:



Wir „betreten“ den Zyklus (2.12) demzufolge an der Stelle $(1-N) \bmod n$ und erreichen nach $N-1$ Schritten die Ausgangskurve.

Lemma 2.29 (Vercauteren)**.** *Sei $g(X,Y) \in \mathbb{Z}_q[X,Y]$ und $x_0, y_0 \in \mathbb{Z}_q$, so dass*

$$g(x_0,y_0) \equiv 0 \pmod{p}, \quad \frac{\partial g}{\partial X}(x_0,y_0) \equiv 0 \pmod{p} \quad \textit{und} \quad \frac{\partial g}{\partial Y}(x_0,y_0) \not\equiv 0 \pmod{p}.$$

Dann gelten folgende Aussagen:

a) Zu jedem $x \in \mathbb{Z}_q$ mit $x \equiv x_0 \pmod{p}$ gibt es genau ein $y \in \mathbb{Z}_q$, so dass

$$y \equiv y_0 \pmod{p} \quad \textit{und} \quad g(x,y) = 0.$$

b) Sei $\bar{x} \in \mathbb{Z}_q$ mit $x \equiv \bar{x} \pmod{p^N}$, $N \geq 1$. Dann gilt für das nach a) existierende eindeutige Element $\bar{y} \in \mathbb{Z}_q$ mit $\bar{y} \equiv y_0 \pmod{p}$ und $g(\bar{x},\bar{y}) = 0$, dass

$$\bar{y} \equiv y \pmod{p^{N+1}}.$$

Beweis. Aussage a) folgt aus Lemma 2.16, wobei

$$\mathbb{Z}_q[Y] \ni f(Y) \coloneqq g(x,Y).$$

Da $(\bar{x}-x)^2 \equiv 0 \pmod{p^{2N}}$, $(\bar{y}-y)^2 \equiv 0 \pmod{p^{2N}}$ und $N+1 \leq 2N$, gilt nach Taylor

$$0 = g(\bar{x},\bar{y}) \equiv g(x,\bar{y}) + \frac{\partial g}{\partial X}(x,\bar{y}) \cdot (\bar{x}-x) \pmod{p^{N+1}},$$

sowie

$$g(x,\bar{y}) \equiv g(x,y) + \frac{\partial g}{\partial Y}(x,y) \cdot (\bar{y}-y) \pmod{p^{N+1}}.$$

Wir setzen die zweite Gleichung in die Erste ein und erhalten mit $g(x, y) = 0$, dass

$$0 \equiv \frac{\partial g}{\partial Y}(x, y) \cdot (\bar{y} - y) + \frac{\partial g}{\partial X}(x, \bar{y}) \cdot (\bar{x} - x) \pmod{p^{N+1}}. \tag{2.15}$$

Weiter gilt nach Voraussetzung, dass

$$\frac{\partial g}{\partial X}(x, \bar{y}) \equiv \frac{\partial g}{\partial X}(x_0, y_0) \equiv 0 \pmod{p},$$

und damit

$$\frac{\partial g}{\partial X}(x, \bar{y}) \cdot (\bar{x} - x) \equiv 0 \pmod{p^{N+1}}.$$

Da ebenfalls nach Voraussetzung

$$\frac{\partial g}{\partial Y}(x, y) \equiv \frac{\partial g}{\partial Y}(x_0, y_0) \not\equiv 0 \pmod{p},$$

folgt aus (2.15), dass

$$0 \equiv \bar{y} - y \pmod{p^{N+1}}. \qquad \square$$

Wir wenden nun das obige Lemma auf unsere Situation an: Mit $g(X, Y) \coloneqq \Phi_p(X, Y)$, $x_0 \coloneqq j_i$, $y_0 \coloneqq j_{i+1}$ und $x \coloneqq j(E^{(i)\uparrow})$ folgt aus Satz 2.28, dass $y = j(E^{(i+1)\uparrow})$. Setzen wir also $\bar{x} \coloneqq J$ und $J' \coloneqq \bar{y}$, dann gilt (2.14).

Wir fassen abschließend alle Ergebnisse in Algorithmus 5 zusammen.

Algorithmus 5 Berechnung der j-Invariante des kanonischen Lifts von $E/\mathbb{F}_q$

Eingabe: $j \in \mathbb{F}_{p^n} \setminus \mathbb{F}_{p^2}$ und $N \geq 1$.
Ausgabe: $J \in \mathbb{Z}_q$, so dass $J \equiv j(E^{\uparrow}) \pmod{p^N}$, wobei $E/\mathbb{F}_q$ mit $j(E) = j$.

```
1: procedure LIFT_J_INVARIANT(j, N)
2:     m ← (1 − N) mod n
3:     J ← π^{-1}( σ^m(j) )
4:     for i ← 2 to N do
5:         J ← NEWTON(Φ_p(J, Y), J^p mod p, i)
6:     end for
7:     return J
8: end procedure
```

Bemerkung. Wir können den Zyklus (2.12) nur in Pfeilrichtung durchlaufen. Ein (theoretisch noch effizienteres) „forward-backward“ um die Genauigkeit nach oben zu „schaukeln“ ist leider nicht möglich. Falls $N > n$, kann man allerdings Algorithmus 5 noch verbessern, da der Zyklus dann mehr als einmal durchlaufen wird. Es

lohnt sich deshalb, bereits berechnete j-Invarianten zu speichern und diese (in einem modifizierten Algorithmus 3) statt $J^p \bmod p$ zu verwenden. In unseren Anwendungen gilt jedoch stets $N < n$.

Mit $J \coloneqq j(E^{\uparrow})$ definieren wir folgende elliptische Kurve über $\mathbb{Q}_q$ (vgl. (1.4)):

$$\mathcal{E}:\ y^2 = x^3 + \left(\frac{3J}{1728 - J}\right) x + \left(\frac{2J}{1728 - J}\right). \tag{2.16}$$

Man beachte, dass $J \notin \{0, 1728\}$, da $\pi(J) = j(E) \notin \mathbb{F}_{p^2}$. Es gilt $j(\mathcal{E}) = J = j(E^{\uparrow})$ und $\mathcal{E}$ ist der kanonische Lift von

$$E':\ y^2 = x^3 + \left(\frac{3j}{1728 - j}\right) x + \left(\frac{2j}{1728 - j}\right),$$

wobei $j \coloneqq j(E)$, also $j(E') = j(E)$.

Wir arbeiten in der Implementierung stets mit $\mathcal{E} = E'^{\uparrow}$ und verzichten darauf, den kanonischen Lift von E zu bestimmen, denn es gilt

$$\mathrm{End}(E') \ni \mathrm{Tr}(\phi_q) = \pm \mathrm{Tr}(\phi_q) \in \mathrm{End}(E).$$

Zur Begründung überlegt man sich folgendes: Nach Satz 1.7 sind E' und E entweder isomorph über $\mathbb{F}_q$ oder E' ist ein quadratischer Twist von E. Kombiniert man Satz 1.30 und Lemma 1.27, so erhält man die behauptete Identität.

3 Berechnung der Spur des Frobenius-Endomorphismus

3.1 Formale Gruppen

Sei R ein kommutativer Ring mit Einselement. Wir bezeichnen den *Ring der formalen Potenzreihen in τ über R* mit $R[\![\tau]\!]$. Addition und Multiplikation sind dabei wie folgt definiert:

$$\left(\sum_{i=0}^{\infty} a_i \tau^i\right) + \left(\sum_{i=0}^{\infty} b_i \tau^i\right) = \sum_{i=0}^{\infty} (a_i + b_i)\tau^i,$$
$$\left(\sum_{i=0}^{\infty} a_i \tau^i\right) \cdot \left(\sum_{i=0}^{\infty} b_i \tau^i\right) = \sum_{i=0}^{\infty} \left(\sum_{j=0}^{i} a_j b_{i-j}\right) \tau^i.$$

Damit die Komposition $g(h(\tau))$ für $g(\tau), h(\tau) \in R[\![\tau]\!]$ wieder in $R[\![\tau]\!]$ liegt[1], muss entweder $g(\tau) \in R[\tau]$ oder $h(\tau) \in \tau R[\![\tau]\!]$ erfüllt sein. Ebenso gilt[2] für $F(X,Y) \in R[\![X,Y]\!]$ mit $F(0,0) = 0$ und $g(\tau), h(\tau) \in \tau R[\![\tau]\!]$, dass

$$F(g,h) \in \tau R[\![\tau]\!].$$

Wir definieren mit Hilfe von $F(X,Y)$ eine neue Verknüpfung auf $\tau R[\![\tau]\!]$:

$$g \oplus_F h \coloneqq F(g,h).$$

Definition 3.1. Ein (kommutatives) *formales Gruppengesetz über R* ist eine formale Potenzreihe $F(X,Y) \in R[\![X,Y]\!]$ mit den folgenden Eigenschaften:

a) $F(X,0) = X$,

b) $F(X,Y) = F(Y,X)$ und

c) $F(F(X,Y),Z) = F(X,F(Y,Z))$.

[1] Beispielsweise gilt für $g(\tau) = \sum_{i=0}^{\infty} \tau^i$ und $h(\tau) = 1 + \tau$, dass $g(h(\tau)) \notin R[\![\tau]\!]$.
[2] [Blu97, §I.1]

Zunächst folgt aus a) und b), dass

$$F(X,Y) = X + Y + XY \cdot G(X,Y), \quad G \in R[\![X,Y]\!]. \tag{3.1}$$

Nach [Blu97, §I, Lemma 2.1] ist c) wohldefiniert und es gilt

Satz 3.2. *Sei $F(X,Y) \in R[\![X,Y]\!]$ mit $F(0,0) = 0$. Dann sind folgende Aussagen äquivalent:*

a) $F(X,Y)$ ist ein formales Gruppengesetz über R.

b) $(\tau R[\![\tau]\!], \oplus_F, 0)$ ist eine abelsche Gruppe.

c) $(\tau R[\![\tau]\!], \oplus_F, 0)$ ist ein kommutativer Monoid[3]*.*

Beweis. [Blu97, §I, Proposition 2.3]. □

Wir bezeichnen die *formale Gruppe* $(\tau R[\![\tau]\!], \oplus_F, 0)$ mit $\mathcal{C}(F)$ und das inverse Element von τ mit $\iota(\tau) \in \tau R[\![\tau]\!]$:

$$\tau \oplus_F \iota = 0.$$

Es gilt $x \oplus_F \iota(x) = 0$ für alle $x \in \mathcal{C}(F)$, d. h. $\iota(x)$ ist das inverse Element von x. Die (induktive) Berechnung von $\iota(\tau) = -\tau + \cdots$ wird in [Blu97, §I, Lemma 2.2] vorgestellt.

Als praktisches Hilfsmittel zur Konstruktion von formalen Gruppen notieren wir

Lemma 3.3. *Sei $(G,+,0)$ eine abelsche Gruppe und $T : \tau R[\![\tau]\!] \to G$ eine injektive Abbildung mit $T(0) = 0$. Weiter gelte für $F(X,Y) \in R[\![X,Y]\!]$ mit $F(0,0) = 0$, dass*

$$T(g) + T(h) = T(F(g,h)) \quad \textit{für alle } g(\tau), h(\tau) \in \tau R[\![\tau]\!].$$

Dann ist $F(X,Y)$ ein formales Gruppengesetz über R (und $T : \mathcal{C}(F) \to G$ ein Monomorphismus).

Beweis. [Blu97, §I, Proposition 1.1]. □

Beispiel 3.4. a) Sei $G = R[\![\tau]\!]$ und $T : \tau R[\![\tau]\!] \to R[\![\tau]\!]$ die Inklusionsabbildung. Dann ist $F(X,Y) = X + Y$ nach Lemma 3.3 ein formales Gruppengesetz über R. Man nennt $X + Y$ *additives formales Gruppengesetz.*

[3] [Bos06, 1.1]

b) Sei $G = R[\![\tau]\!]^* = \{g(\tau) \in R[\![\tau]\!] : \ g(0) \in R^*\}$ (siehe [Blu97, §I, Lemma 1.4]) und $T : \ \tau R[\![\tau]\!] \to R[\![\tau]\!]^*, \ g(\tau) \mapsto 1 + g(\tau)$. Dann gilt

$$T(g) \cdot T(h) = (1+g)(1+h) = T(g+h+gh)$$

und wir erhalten das *multiplikative formale Gruppengesetz* $X + Y + XY$.

Definition 3.5. Eine formale Potenzreihe $U(\tau) \in \tau R[\![\tau]\!]$ heißt *Homomorphismus zwischen* $\mathcal{C}(F_1)$ *und* $\mathcal{C}(F_2)$ (kurz: $U \in \mathrm{Hom}(F_1, F_2)$), wenn

$$U(x \oplus_{F_1} y) = U(x) \oplus_{F_2} U(y) \quad \text{für alle } x, y \in \mathcal{C}(F_1),$$

d. h. $U(F_1(X,Y)) = F_2(U(X), U(Y)) \in R[\![X,Y]\!]$.

$\mathrm{Hom}(F_1, F_2)$ wird durch die Verknüpfung

$$(U+V)(x) \coloneqq U(x) \oplus_{F_2} V(x)$$

zu einer Gruppe. Man überzeugt sich davon, dass $(U+V) \in \mathrm{Hom}(F_1, F_2)$ gilt:

$$\begin{aligned}(U+V)(x \oplus_{F_1} y) &= U(x \oplus_{F_1} y) \ \oplus_{F_2} \ V(x \oplus_{F_1} y) \\ &= U(x) \ \oplus_{F_2} \ V(x) \ \oplus_{F_2} \ U(y) \ \oplus_{F_2} \ V(y) \\ &= (U+V)(x) \ \oplus_{F_2} \ (U+V)(y).\end{aligned}$$

Für $m \in \mathbb{Z}$ definieren wir $[m] \in \mathrm{Hom}(F_1, F_1) =: \mathrm{End}(F_1)$ rekursiv wie folgt:

$$[0] \coloneqq 0, \quad [m+1] \coloneqq [m] + \tau, \quad [m-1] \coloneqq [m] + \iota.$$

Die Komposition von Homomorphismen ist ebenfalls wieder ein Homomorphismus:

$$U \in \mathrm{Hom}(F_1, F_2), \ V \in \mathrm{Hom}(F_2, F_3) \implies V \circ U \in \mathrm{Hom}(F_1, F_3). \tag{3.2}$$

$\mathrm{End}(F_1)$ wird durch die Verknüpfungen $+$ und $\circ$ zu einem Ring mit Einselement τ.

Sei $f(\tau) = \sum_{i=0}^{\infty} f_i \tau^i \in R[\![\tau]\!]$. Wir bezeichnen den Koeffizienten von τ mit $\mathrm{c}_1(f) \in R$:

$$\mathrm{c}_1\left(\sum_{i=0}^{\infty} f_i \tau^i\right) = f_1.$$

Für $U, V \in \mathrm{End}(F_1)$ gelten folgende einfache Rechenregeln:

$$\mathrm{c}_1(U+V) = \mathrm{c}_1(U) + \mathrm{c}_1(V) \quad \text{und} \quad \mathrm{c}_1(V \circ U) = \mathrm{c}_1(V) \cdot \mathrm{c}_1(U). \tag{3.3}$$

Die erste Formel folgt aus (3.1) und die zweite Eigenschaft lässt sich auf den Fall (3.2) verallgemeinern.

Lemma 3.6. *Sei G_i eine abelsche Gruppe und $T_i : \mathcal{C}(F_i) \to G_i$ ein Monomorphismus ($i = 1, 2$). Weiter sei $\phi : G_1 \to G_2$ ein Gruppenhomomorphismus und $U(\tau) \in \tau R[\![\tau]\!]$ mit*

$$(\phi \circ T_1)(g) = (T_2 \circ U)(g) \quad \textit{für alle } g(\tau) \in \tau R[\![\tau]\!].$$

Dann ist $U(\tau)$ ein Homomorphismus zwischen $\mathcal{C}(F_1)$ und $\mathcal{C}(F_2)$.

Beweis. Wir rechnen nach, dass

$$\begin{array}{ccc} \mathcal{C}(F_1) & \xrightarrow{T_1} & G_1 \\ \downarrow{\scriptstyle U} & & \downarrow{\scriptstyle \phi} \\ \mathcal{C}(F_2) & \xrightarrow{T_2} & G_2 \end{array}$$

$$\begin{aligned} (T_2 \circ U)(x \oplus_{F_1} y) &= (\phi \circ T_1)(x \oplus_{F_1} y) \\ &= \phi(T_1(x) + T_1(y)) \\ &= (\phi \circ T_1)(x) + (\phi \circ T_1)(y) \\ &= (T_2 \circ U)(x) + (T_2 \circ U)(y) \\ &= T_2(U(x) \oplus_{F_2} U(y)). \end{aligned}$$

T_2 ist injektiv, also gilt $U(x \oplus_{F_1} y) = U(x) \oplus_{F_2} U(y)$ für alle $x, y \in \mathcal{C}(F_1)$. □

3.2 Die formale Gruppe einer elliptischen Kurve

Sei K ein Körper[4], $L \coloneqq \mathrm{Quot}(K[\![\tau]\!]) \supset K$ und

$$E/K : \; Y^2Z - X^3 - aXZ^2 - bZ^3 =: w(X, Y, Z) = 0$$

eine elliptische Kurve. Wir werden im Folgenden eine injektive Abbildung

$$T : \; \tau K[\![\tau]\!] \to E(L)$$

mit $T(0) = \mathcal{O}$ konstruieren und eine formale Potenzreihe $F(X, Y) \in K[\![X, Y]\!]$ (mit $F(0, 0) = 0$) bestimmen, so dass

$$T(g) + T(h) = T(F(g, h)) \quad \text{für alle } g(\tau), h(\tau) \in \tau K[\![\tau]\!].$$

Nach Lemma 3.3 ist dann $F(X, Y)$ ein formales Gruppengesetz über K.

Definition 3.7. $\mathcal{C}(F)$ heißt *formale Gruppe der elliptischen Kurve E.*

Wir haben den Funktionenkörper $K(E)$ bereits in Abschnitt 1.3 definiert. Betrachtet man E als projektive Kurve über K, so kann man $f \in K(E)$ (lokal) als Quotient von homogenen Polynomen $g, h \in K[X, Y, Z]$ mit $\deg(h) = \deg(g)$ schreiben. Dabei sind

[4] $\mathrm{char}\, K \neq 2, 3$

$f_1 = g_1/h_1$ und $f_2 = g_2/h_2$ gleich, wenn $g_1h_2 - g_2h_1$ durch w teilbar ist (siehe [CF06, 4.2] und [Sil86, I., Remark 2.9]). Aus dieser Definition ergibt sich unmittelbar, dass

$$s = t^3 + ats^2 + bs^3 \quad \text{bzw.} \quad K(E) \ni w(t, -1, s) = 0, \tag{3.4}$$

wobei $s \coloneqq -Z/Y$ und $t \coloneqq -X/Y$.

Wir fassen nun die wichtigsten Aussagen von [Blu97, §II.3] zusammen. Einen Überblick findet man auch in [CF06, 4.4.1].

Man sagt $f \in K(E)$ ist *regulär* (oder *definiert*) *im Punkt* $P \in E(\overline{K})$, wenn eine Darstellung $f = g/h$ existiert, so dass $h(P) \neq 0$. Der *Wert von* f *im Punkt* P ist dann

$$f(P) \coloneqq g(P)/h(P).$$

Er ist unabhängig von der Wahl der Repräsentanten, denn für $g/h = g_1/h_1 \in K(E)$ mit $h_1(P) \neq 0$ gilt

$$(gh_1 - g_1h)(P) = (uw)(P) = 0, \quad \text{wobei } u \in \overline{K}[X, Y, Z].$$

Beispiel 3.8.

$$K(E) \ni f = Z/X = \frac{X^2 + aZ^2}{Y^2 - bZ^2} \quad \text{und} \quad f(\mathcal{O}) = \frac{0}{1} = 0.$$

Sei $P \in E(K)$ und

$$R_P \coloneqq \{f \in K(E) : \ f \text{ ist regulär in } P\}.$$

Wie man leicht zeigt, ist R_P ein Unterring[5] von $K(E)$ und

$$\mathfrak{m}_P \coloneqq \{f \in R_P : \ f(P) = 0\}$$

ein Ideal. R_P heißt *lokaler Ring in* P. Identifiziert man die Menge der konstanten Funktionen mit K, so gilt

$$R_P = K \oplus \mathfrak{m}_P,$$

denn $f = f(P) + (f - f(P))$.

Lemma 3.9. *$R_P^* = R_P \setminus \mathfrak{m}_P$ und $\mathfrak{m}_P$ ist das einzige maximale Ideal von R_P.*

[5] Häufig wird auch die Bezeichnung $\mathcal{O}_P$ verwendet. Allerdings möchten wir den Ausdruck $\mathcal{O}_\mathcal{O}$ vermeiden.

Beweis. Sei zunächst $f \in R_P \setminus \mathfrak{m}_P$. Dann existiert eine Darstellung $f = g/h$ mit $g(P), h(P) \neq 0$. Folglich gilt mit $1/f = h/g$, dass $1/f$ in P regulär ist, also $f \in R_P^*$.

Umgekehrt gilt für $f \in R_P^*$, dass ein $g \in R_P$ existiert mit $fg = 1$. Daraus folgt $f(P)g(P) = 1$, d. h. $f(P) \neq 0$ und damit $f \notin \mathfrak{m}_P$.

Sei $I \subsetneq R_P$ ein Ideal. Daraus folgt $I \cap R_P^* = \emptyset$ und damit $I \subset \mathfrak{m}_P$. □

Proposition 3.10. *$\mathfrak{m}_P$ ist ein Hauptideal von R_P. Für $u \in \mathfrak{m}_P$ gilt*

$$\mathfrak{m}_P = (u) \quad \Longleftrightarrow \quad u \notin \mathfrak{m}_P^2.$$

Beweis. [Blu97, §II, Proposition 3.6]. □

Ein erzeugendes Element von $\mathfrak{m}_P$ heißt *Orts-Uniformisierende* im Punkt P.

Lemma 3.11. *Sei $0 \neq f \in K(E)$ und $u \in \mathfrak{m}_P \setminus \mathfrak{m}_P^2$ eine Orts-Uniformisierende im Punkt P. Dann gibt es genau ein $r \in \mathbb{Z}$, so dass*

$$f = u^r g \quad \textit{mit } g \in R_P^*.$$

Die ganze Zahl r hängt dabei nicht von u ab.

Beweis. [Blu97, §II, Corollary 3.7]. □

Wir definieren die *Bewertung von f im Punkt P* als

$$v_P(f) \coloneqq r$$

und setzen $v_P(0) \coloneqq \infty$.

Lemma 3.12. *Sei $u \in K(E)$ eine Orts-Uniformisierende in $P \in E(K)$ und $f \in K(E)^*$ mit $v_P(f) = r \in \mathbb{Z}$. Dann existiert genau eine Folge $(f_n)_{n \geq r}$ in K, so dass für alle $m \geq r$ gilt*

$$v_P\left(f - \sum_{i=r}^{m} f_i u^i\right) \geq m + 1.$$

Beweis. Durch Induktion nach m beweisen wir zunächst die Existenz. Für $m = r$ definieren wir $g \coloneqq u^{-r} f \in K(E)$ und es gilt

$$v_P(g) = v_P(u^{-r}) + v_P(f) = -r + r = 0,$$

d. h. $g \in R_P^*$. Mit $f_r := g(P) \in K^*$ gilt also

$$v_P(f - f_r u^r) = v_P(u^r(g - f_r)) = r + v_P(g - f_r) > r,$$

denn $(g - f_r)(P) = 0$.

Sei nun $m \geq r$. Nach Induktionsvoraussetzung existieren $f_r, f_{r+1}, \ldots, f_m \in K$, so dass

$$v_P\left(f - \sum_{i=r}^{m} f_i u^i\right) := l \geq m+1, \quad \text{wobei } l \in \mathbb{Z} \cup \{\infty\}.$$

Falls $l > m+1$ sind wir fertig (definiere $f_{m+1} := 0$). Andernfalls gilt für

$$g := u^{-(m+1)}\left(f - \sum_{i=r}^{m} f_i u^i\right),$$

dass g in P regulär ist, da $v_P(g) = 0$. Mit $f_{m+1} := g(P)$ gilt nun

$$v_P\left(f - \sum_{i=r}^{m+1} f_i u^i\right) = v_P(g u^{m+1} - f_{m+1} u^{m+1}) = v_P(u^{m+1}(g - f_{m+1})) \geq m+2,$$

denn $(g - f_{m+1})(P) = 0$.

Eindeutigkeit: Sei $(g_n)_{n \geq r}$ eine zweite Folge in K mit der geforderten Eigenschaft und $j \geq r$ der kleinste Index, so dass $g_j \neq f_j$. Dann gilt

$$v_P\left(f - \sum_{i=r}^{j} f_i u^i\right) \geq j+1 \quad \text{und} \quad v_P\left(f - \sum_{i=r}^{j} g_i u^i\right) \geq j+1.$$

Daraus folgt $v_P((g_j - f_j)u^j) \geq j+1$, d. h. $g_j = f_j$. Widerspruch! Hierbei haben wir verwendet, dass

$$v_P(c+d) \geq \min(v_P(c), v_P(d)) \quad \text{für alle } c, d \in K(E).$$

Diese Eigenschaft folgt unmittelbar aus Lemma 3.11. □

Genauer ist $v_P : K(E) \to \mathbb{Z} \cup \{\infty\}$ eine *Exponentialbewertung*[6] von $K(E)$. Wir gehen allerdings auf diesen Aspekt nicht weiter ein.

Das obige Lemma liefert nach [Blu97, §II, Corollary 4.2] einen injektiven *Ringhomomorphismus*

$$\Psi_u : R_P \to K[\![\tau]\!], \; f \mapsto \sum_{i=r}^{\infty} f_i \tau^i.$$

[6] [Neu92, II., §3]

Nach [Blu97, §II, Example 3.8 und 3.9] ist $t = -X/Y$ eine Orts-Uniformisierende in $\mathcal{O}$ und $v_{\mathcal{O}}(s) = 3$. Es existiert also

$$\Psi_t(s) = \sum_{i=3}^{\infty} s_i \tau^i =: S(\tau),$$

so dass $v_{\mathcal{O}}(s - \sum_{i=3}^{m} s_i t^i) \geq m+1$ für alle $m \geq 3$. Mit Hilfe der Gleichung (3.4) kann man $(s_n)_{n\geq 3}$ durch Einsetzen immer genauer approximieren. Ausgehend von $s = t^3 + O(t^7)$ erhält man schließlich folgende Rekursionsvorschrift (vgl. [Blu97, §II.4]):

$$s_0, s_1, s_2 \coloneqq 0, \quad s_3 \coloneqq 1, \quad s_n \coloneqq a \left(\sum_{i+j=n-1} s_i s_j \right) + b \left(\sum_{i+j+k=n} s_i s_j s_k \right).$$

Insbesondere gilt $S(\tau) \in \tau\mathbb{Z}[a,b][\![\tau]\!]$.

Lemma 3.13. *In $K[\![\tau]\!]$ gilt*

$$w(f, -1, S(f)) = 0 \quad \textit{für alle } f(\tau) \in \tau K[\![\tau]\!].$$

Außerdem folgt aus $w(f,-1,g) = 0$ bereits $g(\tau) = S(f)$.

Beweis. Mit $\Psi_t(t) = \tau$ und $\Psi_t(s) = S(\tau)$ folgt aus $w(t,-1,s) = 0$, dass

$$0 = \Psi_t(0) = \Psi_t(w(t,-1,s)) = w(\tau, -1, S(\tau)).$$

Daraus folgt $w(f,-1,S(f)) = 0$ für alle $f(\tau) \in \tau K[\![\tau]\!]$.

Angenommen $w(f,-1,g) = 0$, wobei $f(\tau), g(\tau) \in \tau K[\![\tau]\!]$. Wir definieren $h(\tau) \coloneqq S(f)$. Dann gilt

$$\begin{aligned} 0 &= w(f,-1,h) - w(f,-1,g) \\ &= (g-h)(-1 + af(g+h) + b(g^2 + gh + h^2)). \end{aligned}$$

Nach [Blu97, §I, Lemma 1.4] ist $-1 + af(g+h) + \cdots$ in $K[\![\tau]\!]$ invertierbar und somit gilt $g - h = 0$. □

Wir definieren die injektive Abbildung

$$T: \ \tau K[\![\tau]\!] \to E(L), \ f \mapsto (f : -1 : S(f)).$$

Man beachte, dass T nach Lemma 3.13 wohldefiniert ist, denn $w(f,-1,S(f)) = 0$. Außerdem gilt $T(0) = (0 : -1 : 0) = \mathcal{O}$.

Schließlich benötigen wir noch $F(X,Y) \in K[\![X,Y]\!]$ mit $F(0,0) = 0$ und

$$T(f) + T(g) = T(F(f,g)) \quad \text{für alle } f(\tau), g(\tau) \in \tau K[\![\tau]\!].$$

Die Konstruktion des formalen Gruppengesetzes wird im Beweis von [Blu97, §II, Theorem 6.1] durchgeführt. Wir verzichten auf Details und notieren als Ergebnis die ersten Glieder der formalen Potenzreihe (vgl. [Con99, 2.6.2]):

$$\begin{aligned} F(X,Y) = X + Y - 4a(X^2Y^3 + X^3Y^2) - 2a(XY^4 + X^4Y) - 15b(X^3Y^4 + X^4Y^3) \\ - 9b(X^2Y^5 + X^5Y^2) - 3b(XY^6 + X^6Y) + \cdots \in \mathbb{Z}[a,b][\![X,Y]\!]. \end{aligned}$$

Isogenien und Homomorphismen (zwischen den entsprechenden formalen Gruppen elliptischer Kurven) sind auf natürliche Weise miteinander verbunden. Bevor wir die konkrete Aussage in Lemma 3.15 formulieren, stellen wir das Konstruktionsprinzip vor.

Seien also E_1 und E_2 elliptische Kurven über K und $\phi \in \mathrm{Hom}_K(E_1, E_2)$. Wir bezeichnen die formale Gruppe von E_i mit $\mathcal{C}(F_i)$ $(i = 1, 2)$ und schreiben

$$\phi(X,Y,Z) = (\phi_1(X,Y,Z) : \phi_2(X,Y,Z) : \phi_3(X,Y,Z)),$$

wobei ϕ_1, ϕ_2 und ϕ_3 homogene Polynome vom Grad d über K sind. Es gilt $\phi(\mathcal{O}) = \mathcal{O}$, d. h. $\phi_1(\mathcal{O}) = \phi_3(\mathcal{O}) = 0$ und $\phi_2(\mathcal{O}) \neq 0$. Daraus folgt

$$\mathfrak{m}_\mathcal{O} \ni \phi_1/Y^d = \phi_1(X/Y, 1, Z/Y) = \phi_1(-t, 1, -s) = (-1)^d \phi_1(t, -1, s)$$

und

$$R_\mathcal{O}^* = (R_\mathcal{O} \setminus \mathfrak{m}_\mathcal{O}) \ni \phi_2/Y^d = (-1)^d \phi_2(t, -1, s),$$

also gilt

$$K(E) \ni \phi_1/\phi_2 = \frac{\phi_1(t,-1,s)}{\phi_2(t,-1,s)} \in \mathfrak{m}_\mathcal{O}.$$

Definition 3.14. Man nennt[7]

$$\Psi_t(-\phi_1/\phi_2) =: U(\tau) = \sum_{i=1}^{\infty} u_i \tau^i \in \tau K[\![\tau]\!]$$

den von ϕ *induzierten Homomorphismus zwischen* $\mathcal{C}(F_1)$ *und* $\mathcal{C}(F_2)$.

[7] In [Blu97, §II.7] fehlt das negative Vorzeichen (vermutlich unabsichtlich). Man erhält dann jedoch einen Widerspruch, da $\mathrm{c}_1(\Psi_t(X/Y)) = -1$. Unsere Definition stimmt mit [Sat00] überein.

Allerdings müssen wir noch beweisen, dass $U(\tau)$ tatsächlich ein Homomorphismus ist. Nebenbei bemerkt kann man $U(\tau)$ mit Hilfe der folgenden Umformung berechnen:

$$\Psi_t(-\phi_1/\phi_2) = -\frac{\phi_1(\tau, -1, S(\tau))}{\phi_2(\tau, -1, S(\tau))}, \tag{3.5}$$

wobei der Nenner in $K[\![\tau]\!]$ invertierbar ist, da $\phi_2(0, -1, S(0)) = \phi_2(0, -1, 0) \neq 0$.

Lemma 3.15. *Mit den eingeführten Bezeichnungen gilt:*

a) $U(\tau)$ ist ein Homomorphismus zwischen $\mathcal{C}(F_1)$ und $\mathcal{C}(F_2)$.

b) Die Abbildung $\mathrm{Hom}_K(E_1, E_2) \to \mathrm{Hom}(F_1, F_2),\ \phi \mapsto U$ ist ein Monomorphismus.

c) Sei E_3/K eine weitere elliptische Kurve, $\psi \in \mathrm{Hom}_K(E_2, E_3)$ und $V(\tau)$ der von ψ induzierte Homomorphismus. Dann induziert $\psi \circ \phi$ den Homomorphismus

$$V \circ U \in \mathrm{Hom}(F_1, F_3).$$

Beweis. [Blu97, §II, Proposition 7.1]. □

Satz 3.16 (Satoh)**.** *Sei E/K eine elliptische Kurve, $\phi \in \mathrm{End}_K(E)$ und $U(\tau) \in \tau K[\![\tau]\!]$ der von ϕ induzierte Homomorphismus. Falls $\mathrm{c}_1(U) \neq 0$, so gilt*

$$\mathrm{Tr}(\phi) = \mathrm{c}_1(U) + \frac{\deg(\phi)}{\mathrm{c}_1(U)}.$$

Beweis. Wir bezeichnen die formale Gruppe von E mit $\mathcal{C}(F)$. Es gilt

$$\mathrm{End}(E) \ni 0 = \phi^2 - \mathrm{Tr}(\phi)\phi + \deg(\phi).$$

Folglich erhalten wir mit Lemma 3.15 und $\mathrm{Tr}(\phi), \deg(\phi) \in \mathbb{Z}$

$$\mathrm{End}(F) \ni [0] = U \circ U + [-\mathrm{Tr}(\phi)] \circ U + [\deg(\phi)].$$

Insbesondere gilt mit (3.3)

$$K \ni 0 = \mathrm{c}_1([0]) = \mathrm{c}_1(U)^2 - \mathrm{Tr}(\phi) \cdot \mathrm{c}_1(U) + \deg(\phi). \qquad \square$$

Bemerkung. Satoh formuliert die Aussage in [Sat00, Proposition 4.1]. Er vermutet, dass man diesen „einfachen Zusammenhang“ bereits vor ihm kannte, allerdings fand er diesbezüglich keine Anhaltspunkte in der Literatur. Im Übrigen können wir den Satz auch direkt mit Hilfe der Gleichungen $\mathrm{Tr}(\phi) = \phi + \widehat{\phi}$ und $\deg(\phi) = \phi\widehat{\phi}$ beweisen. Die obige Herleitung vermeidet lediglich die Einführung des von $\widehat{\phi}$ induzierten Homomorphismus.

Kommen wir nun zu unserer Ausgangslage am Ende des letzten Kapitels zurück: Sei $E:\ y^2 = x^3 + ax + b$ eine gewöhnliche elliptische Kurve über $\mathbb{F}_q$ und

$$E^{\uparrow}:\ y^2 = x^3 + Ax + B \tag{3.6}$$

der kanonische Lift von E. Die Koeffizienten A, B liegen dabei in dem Unterring $\mathbb{Z}_q \subset \mathbb{Q}_q$. Wir übertragen alle Resultate in diesem Abschnitt auf den Spezialfall $K = \mathbb{Q}_q$.

Sei $\phi_q^{\uparrow} \in \mathrm{End}(E^{\uparrow})$ das Bild des Frobenius-Endomorphismus ϕ_q unter der Isomorphie $\mathrm{End}(E) \cong \mathrm{End}(E^{\uparrow})$ und $U(\tau) \in \tau\mathbb{Q}_q[\![\tau]\!]$ der von $\phi_q^{\uparrow}$ induzierte Homomorphismus[8]. Nach [Blu97, §II, Lemma 7.3] gilt[9]

$$\mathbb{Q}_q \ni \mathrm{c}_1(U) \neq 0.$$

Zudem folgt aus Proposition 1.29 a) und Satz 2.25, dass

$$q = \deg(\phi_q) = \deg(\phi_q^{\uparrow}).$$

Mit Satz 3.16 erhalten wir

$$\mathrm{Tr}(\phi_q^{\uparrow}) = \mathrm{c}_1(U) + \frac{q}{\mathrm{c}_1(U)}.$$

Bekanntlich gilt[10] $\mathrm{c}_1(U) \in \mathbb{Z}_q$ oder $\mathrm{c}_1(U)^{-1} \in \mathbb{Z}_q$. Angenommen $\mathrm{c}_1(U) \notin \mathbb{Z}_q$, dann folgt $\mathrm{c}_1(U)^{-1} \in \mathbb{Z}_q$ und wir erhalten mit

$$\mathbb{Z}_q \ni\ \mathrm{Tr}(\phi_q) - q \cdot \mathrm{c}_1(U)^{-1} = \mathrm{c}_1(U)$$

einen Widerspruch. Also gilt $\mathrm{c}_1(U) \in \mathbb{Z}_q$. Allerdings gilt $\mathrm{c}_1(U) \notin \mathbb{Z}_q^*$, denn ϕ_q ist inseparabel[11] und deshalb $\pi(\mathrm{c}_1(U)) = 0$.

$\mathbb{Z}_q \ni \mathrm{Tr}(\phi_q) - \mathrm{c}_1(U) = q \cdot \mathrm{c}_1(U)^{-1}$ ist in $\mathbb{Z}_q$ invertierbar, denn E ist gewöhnlich und demzufolge gilt nach Proposition 1.29 c)

$$0 \neq \pi(\mathrm{Tr}(\phi_q)) = \pi(\mathrm{Tr}(\phi_q) - \mathrm{c}_1(U)) = \pi(q \cdot \mathrm{c}_1(U)^{-1}).$$

Wir können somit $\mathrm{c}_1(U)$ als $\mathrm{c}_1(U) = q \cdot d = p^n \cdot d$ schreiben, wobei $d \in \mathbb{Z}_q^*$.

Um $\mathrm{Tr}(\phi_q^{\uparrow}) \bmod p^N$ für ein $1 \leq N < n$ mit Hilfe von Satz 3.16 zu bestimmen, muss man also $\mathrm{c}_1(U)$ zur Präzision $N+n$ berechnen! Zudem erweist sich die Berechnung von

[8] Nach Proposition 2.27 gilt $\phi_q^{\uparrow} \in \mathrm{End}_{\mathbb{Q}_q}(E^{\uparrow})$.

[9] Im Gegensatz zu ϕ_q ist $\phi_q^{\uparrow}$ separabel, da $\mathrm{char}\,\mathbb{Q}_q = 0$ (siehe [Bos06, 3.6, Bemerkung 4]).

[10] (2.4)

[11] Proposition 1.29 a)

$\mathrm{c}_1(U)$ als recht schwierig (vgl. [CF06, 17.3.1.d]). Man umgeht diese Probleme, indem man mit $\widehat{\phi}_q^{\uparrow}$ arbeitet:

Nach Korollar 2.26 ist $\widehat{\phi}_q^{\uparrow}$ die duale Isogenie von $\phi_q^{\uparrow}$. Es gilt

$$\widehat{\phi}_q^{\uparrow} = \mathrm{Tr}(\phi_q^{\uparrow}) - \phi_q^{\uparrow} \in \mathrm{End}_{\mathbb{Q}_q}(E^{\uparrow})$$

und $q = \deg(\phi_q^{\uparrow}) = \deg(\widehat{\phi}_q^{\uparrow})$. Sei $V(\tau) \in \tau\mathbb{Q}_q[\![\tau]\!]$ der von $\widehat{\phi}_q^{\uparrow}$ induzierte Homomorphismus. $\widehat{\phi}_q^{\uparrow}$ ist separabel und somit erhalten wir wie oben

$$\mathbb{Q}_q \ni \mathrm{c}_1(V) \neq 0.$$

Also gilt

$$\mathrm{Tr}(\widehat{\phi}_q^{\uparrow}) = \mathrm{c}_1(V) + \frac{q}{\mathrm{c}_1(V)}.$$

Man zeigt erneut, dass $\mathrm{c}_1(V) \in \mathbb{Z}_q$. Im Gegensatz zu $\mathrm{c}_1(U)$ ist

$$\mathrm{c}_1(V) \in \mathbb{Z}_q^*,$$

denn nach Proposition 1.29 d) ist $\widehat{\phi}_q$ separabel und deshalb[12]

$$\pi(\mathrm{c}_1(V)) \neq 0.$$

Mit (2.13) erhält man

Korollar 3.17. *Sei $N \leq n$. Dann gilt*

$$\mathrm{Tr}(\phi_q) \equiv \mathrm{c}_1(V) \pmod{p^N}.$$

Wir vereinfachen diese Formel mit Hilfe des Zyklus (2.10). Es gilt

$$\widehat{\phi}_q = \widehat{\phi}_{p,0} \circ \widehat{\phi}_{p,1} \circ \cdots \circ \widehat{\phi}_{p,n-1}$$

und damit

$$\widehat{\phi}_q^{\uparrow} = \widehat{\phi}_{p,0}^{\uparrow} \circ \widehat{\phi}_{p,1}^{\uparrow} \circ \cdots \circ \widehat{\phi}_{p,n-1}^{\uparrow}.$$

Sei $V_i(\tau) \in \tau\overline{\mathbb{Q}}_q[\![\tau]\!]$ der von $\widehat{\phi}_{p,i}^{\uparrow}$ induzierte Homomorphismus. Dann gilt nach Lemma 3.15

$$V = V_0 \circ V_1 \circ \cdots \circ V_{n-1}.$$

[12] [Blu97, §II, Lemma 7.3]

Mit (3.3) erhält man

$$\mathrm{c}_1(V) = \prod_{i=0}^{n-1} \mathrm{c}_1(V_i).$$

Dabei gilt $\mathrm{c}_1(V_i) \in \mathbb{Q}_q$, denn

$$\begin{aligned} p = \deg(\phi_{p,i}^{\uparrow}) &= \phi_{p,i}^{\uparrow} \circ \widehat{\phi}_{p,i}^{\uparrow} \\ \Longrightarrow \quad [p] &= U_i \circ V_i \\ \Longrightarrow \quad p \ &= \mathrm{c}_1(U_i) \cdot \mathrm{c}_1(V_i). \end{aligned}$$

Man beachte, dass $\widehat{\phi}_{p,i}^{\uparrow}$ nach Korollar 2.26 die duale Isogenie von $\phi_{p,i}^{\uparrow}$ ist und nach Satz 2.25 $\deg(\phi_{p,i}) = \deg(\phi_{p,i}^{\uparrow})$ gilt. Mit Proposition 2.27 gilt

$$\mathrm{c}_1(U_i) \in \mathbb{Q}_q,$$

wobei $U_i(\tau) \in \tau\mathbb{Q}_q[\![\tau]\!]$ der von $\phi_{p,i}^{\uparrow}$ induzierte Homomorphismus ist.

In (2.10) sind alle Quadrate zueinander konjugiert (vgl. [Ver03, Remark 3.1.1]) und deshalb gilt

$$\mathrm{c}_1(V) = \prod_{i=0}^{n-1} \mathrm{c}_1(V_i) = \mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(\mathrm{c}_1(V_i)). \tag{3.7}$$

Aus $\mathrm{c}_1(V) \in \mathbb{Z}_q$ folgt somit

$$\mathrm{c}_1(V_i) \in \mathbb{Z}_q \tag{3.8}$$

und wir erhalten

$$\mathrm{Tr}(\phi_q) \equiv \mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(\mathrm{c}_1(V_i)) \pmod{p^N}. \tag{3.9}$$

Im nächsten Abschnitt stellen wir die Berechnung von $\mathrm{c}_1(V_{n-1})$ vor. Kann man dessen Norm nur langsam (oder gar nicht) berechnen, so bestimmt man *alle* $\mathrm{c}_1(V_i)$ nach dem gleichen Schema. Auch Satoh versprach sich zunächst nichts von der Umformulierung (3.7) (siehe [Sat00, Remark 4.6]). Bereits in [Sat02] widmet er sich aber ausführlich der Normberechnung.

Inzwischen hat man mehrere schnelle Algorithmen[13] entwickelt, die in allen p-adischen Punktezählverfahren Verwendung finden. In unserer Implementierung berechnen wir die Norm nach Proposition 2.14 und erreichen eine enorme Verbesserung der Laufzeit im Vergleich zur Umsetzung von [CF06, Algorithm 17.38].

[13] [CF06, 12.8.5]

3.3 Die Formeln von Vélu und Satohs Algorithmus

Wir verzichten darauf, die Notation des letzten Abschnitts zu wiederholen.

Unser Ziel ist zunächst die Berechnung von $\mathrm{c}_1(V_{n-1}) \in \mathbb{Z}_q$. Danach sind wir in der Lage, Satohs Algorithmus vollständig zu formulieren. Ausgangspunkt ist erneut (2.10), wobei wir nur das „letzte“ Quadrat betrachten:

$$\begin{array}{ccc} E^{(n-1)\uparrow} & \xleftarrow{\widehat{\phi}^{\uparrow}_{p,n-1}} & E^{\uparrow} \\ \downarrow{\scriptstyle \pi} & & \downarrow{\scriptstyle \pi} \\ E^{(n-1)} & \xleftarrow{\widehat{\phi}_{p,n-1}} & E \end{array}$$

Wir wissen bereits, dass

$$\deg(\widehat{\phi}^{\uparrow}_{p,n-1}) = \deg(\widehat{\phi}_{p,n-1}) = \deg(\phi_{p,n-1}) = p.$$

$\widehat{\phi}^{\uparrow}_{p,n-1}$ ist *separabel*, also gilt nach Satz 1.19

$$\#\ker \widehat{\phi}^{\uparrow}_{p,n-1} = p.$$

Insbesondere ist $\ker \widehat{\phi}^{\uparrow}_{p,n-1}$ eine endliche Untergruppe von $E^{\uparrow}$ und nach Proposition 1.22 existiert eine eindeutig bestimmte elliptische Kurve $E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1}$ und eine separable Isogenie $\phi : E^{\uparrow} \to E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1}$, so dass $\ker \phi = \ker \widehat{\phi}^{\uparrow}_{p,n-1}$. Außerdem gibt es einen eindeutig bestimmten *Isomorphismus*[14] $\lambda : E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1} \to E^{(n-1)\uparrow}$ mit

$$\widehat{\phi}^{\uparrow}_{p,n-1} = \lambda \circ \phi, \tag{3.10}$$

so dass das folgende Diagramm kommutiert:

$$\begin{array}{ccc} E^{(n-1)\uparrow} & \xleftarrow{\widehat{\phi}^{\uparrow}_{p,n-1}} & E^{\uparrow} \\ & {\scriptstyle \lambda} \nwarrow \quad \swarrow {\scriptstyle \phi} & \\ & E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1} & \end{array}$$

[14] Im Beweis von Proposition 1.22 zeigen wir, dass die nach Lemma 1.21 existierende Isogenie λ in unserem Spezialfall sogar ein Isomorphismus ist.

Um $\mathrm{c}_1(V_{n-1})$ mit Hilfe von (3.10) zu berechnen, benötigen wir

Satz 3.18 (Vélu)**.** *Sei K ein Körper*[15]*,*

$$E:\ y^2 = x^3 + ax + b$$

eine elliptische Kurve über K und G eine endliche Untergruppe von $E(\overline{K})$. Wir definieren folgende Polynome in zwei Unbestimmten über K:

$$\xi(X,Y) \coloneqq X, \quad \eta(X,Y) \coloneqq Y, \quad r(X,Y) \coloneqq 3X^2 + a,$$
$$u(X,Y) \coloneqq 4(X^3 + aX + b), \quad v(X,Y) \coloneqq r(X,Y) \cdot \begin{cases} 1 & \text{falls } Y = 0, \\ 2 & \text{falls } Y \neq 0. \end{cases}$$

Sei weiter $G^+ \subset G$ so gewählt[16]*, dass*

$$G = (G \cap E[2]) \cup G^+ \cup (-G^+).$$

Nach Proposition 1.22 existiert eine eindeutig bestimmte elliptische Kurve

$$E/G:\ y^2 = x^3 + \alpha x + \beta$$

und eine separable Isogenie $\phi:\ E \to E/G$ mit $\ker\phi = G$. Es gilt

$$\alpha = a - 5\sum_{P \in S} v(P)$$

und

$$\beta = b - 7\sum_{P \in S} (u + \xi v)(P),$$

wobei $S \coloneqq ((G \cap E[2]) \cup G^+) \setminus \{\mathcal{O}\}$. Außerdem ist ϕ gegeben durch

$$P \mapsto \begin{cases} \mathcal{O} & \text{falls } P \in G, \\ \Big(\xi(P) + \sum_{Q \in S}\Big(\frac{v(Q)}{\xi(P)-\xi(Q)} + \frac{u(Q)}{(\xi(P)-\xi(Q))^2}\Big), & \\ \ \ \eta(P) - \sum_{Q \in S}\Big(\frac{2\eta(P)u(Q)}{(\xi(P)-\xi(Q))^3} + \frac{v(Q)(\eta(P)-\eta(Q))+2(\eta r)(Q)}{(\xi(P)-\xi(Q))^2}\Big)\Big) & \text{falls } P \notin G. \end{cases}$$

Beweis. [Vél71] bzw. [Was08, Theorem 12.16]. □

[15] $\mathrm{char}\, K \neq 2, 3$

[16] Für jedes Paar $P, -P \in G \setminus E[2]$ soll also genau ein Punkt in G^+ liegen.

Korollar 3.19 (Satoh)**.** *Sei $m := \#G$ ungerade und $k := \#G^+$. Dann gilt mit $h(X) := \prod_{P\in G^+}(X - \xi(P)) = \sum_{\nu=0}^{k} h_\nu X^{k-\nu}$, dass*

$$\alpha = (6-5m)a - 30(h_1^2 - 2h_2)$$

und

$$\beta = (15-14m)b - 70(-h_1^3 + 3h_1h_2 - 3h_3) + 42ah_1,$$

wobei $h_\nu := 0$, falls $\nu > k$.

Beweis. Zunächst gilt offenbar

$$G \cap E[2] = \{\mathcal{O}\},$$

denn G enthält keine Punkte der Ordnung 2 (da m ungerade ist). Daraus folgt $S = G^+$ und $k = \frac{m-1}{2}$. Außerdem gilt für $P \in G \setminus \{\mathcal{O}\}$ stets $\eta(P) \neq 0$.

Nach Satz 3.18 gilt

$$\begin{aligned}
\alpha = a - 5\sum_{P\in G^+} v(P) &= a - 10\sum_{P\in G^+} r(P) \\
&= a - 10\sum_{P\in G^+} (3\xi(P)^2 + a) \\
&= a - 10a\cdot\tfrac{m-1}{2} - 30\sum_{P\in G^+}\xi(P)^2 \\
&= (6-5m)a - 30\sum_{P\in G^+}\xi(P)^2
\end{aligned}$$

und

$$\begin{aligned}
\beta = b - 7\sum_{P\in G^+}(u+\xi v)(P) &= b - 7\sum_{P\in G^+}\left(4(\xi^3 + a\xi + b) + 2\xi(3\xi^2 + a)\right)(P) \\
&= b - 7\sum_{P\in G^+}\left(10\xi^3 + 6a\xi + 4b\right)(P) \\
&= b - 28b\cdot\tfrac{m-1}{2} - 70\sum_{P\in G^+}\xi(P)^3 - 42a\sum_{P\in G^+}\xi(P) \\
&= (15-14m)b - 70\sum_{P\in G^+}\xi(P)^3 - 42a\sum_{P\in G^+}\xi(P).
\end{aligned}$$

Die Behauptung folgt aus den Newton-Girard Formeln[17] (auch bekannt als Newton-Identitäten). □

[17] [Tig01, 4.2]

Angenommen wir haben die Koeffizienten α, β von $E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1}$ mit Hilfe des obigen Korollars berechnet ($G \coloneqq \ker \widehat{\phi}^{\uparrow}_{p,n-1}$ und $\#G = p$). Dann können wir den Isomorphismus

$$\lambda : \ E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1} \to E^{(n-1)\uparrow}$$

explizit angeben:

Die Weierstraß-Gleichung von $E^{(n-1)\uparrow}$ hat mit (2.11) und (3.6) die Gestalt

$$y^2 = x^3 + \Sigma^{n-1}(A)x + \Sigma^{n-1}(B).$$

Wir erinnern daran, dass $A, B \neq 0$, da $E/\mathbb{F}_q$ gewöhnlich ist. Nach Satz 1.7 existiert $0 \neq u \in \overline{\mathbb{Q}}_q$, so dass

$$(\Sigma^{n-1}(A), \Sigma^{n-1}(B)) = (u^4\alpha, u^6\beta).$$

Zudem ist λ gegeben durch

$$(X : Y : Z) \mapsto (u^2X : u^3Y : Z).$$

Es folgt

$$u^2 = \frac{\Sigma^{n-1}(B)\alpha}{\Sigma^{n-1}(A)\beta}.$$

Sei $W(\tau) \in \tau\overline{\mathbb{Q}}_q[\![\tau]\!]$ der von λ induzierte Homomorphismus. Nach (3.5) gilt

$$W(\tau) = -\frac{u^2\tau}{-u^3} = u^{-1}\tau,$$

also

$$\mathrm{c}_1(W) = u^{-1}.$$

Der erste Koeffizient des von $\phi : \ E^{\uparrow} \to E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1}$ induzierten Homomorphismus ist 1 (siehe [Vél71, (13)]). Damit folgt aus (3.10), dass

$$\mathrm{c}_1(V_{n-1}) = \mathrm{c}_1(W) \cdot 1 = u^{-1}.$$

Insbesondere gilt wegen (3.8)

$$u^{-1} \in \mathbb{Z}_q \quad \text{und} \quad u^{-2} = \frac{\Sigma^{n-1}(A)\beta}{\Sigma^{n-1}(B)\alpha} =: c \in \mathbb{Z}_q.$$

Mit (3.9) erhält man

$$\mathrm{Tr}(\phi_q) \equiv \mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(u^{-1}) \pmod{p^N}.$$

Man beachte, dass wir nur $c = u^{-2}$ kennen und folglich eine Quadratwurzel in $\mathbb{Z}_q$ berechnen müssen[18]. Das Polynom $\mathbb{Z}_q[X] \ni X^2 - c$ besitzt zwei Nullstellen – doch welche ist die „richtige“? Man verwendet folgende Umformung:

$$(\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(u^{-1}))^2 = \mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(c) =: d \in \mathbb{Z}_p.$$

Proposition 1.33 liefert

$$\mathbb{Z} \ni e \coloneqq \mathrm{Tr}(\phi_q) \bmod p.$$

Mit diesem Wert initialisieren wir nun Algorithmus 4 auf Seite 45 und erhalten die gesuchte Nullstelle von $\mathbb{Z}_p[X] \ni X^2 - d$:

$$\mathrm{Tr}(\phi_q) \equiv \textsc{sqrt}(d, e, N) \pmod{p^N}.$$

Wir begründen nun, warum wir uns gerade für die Berechnung von $\mathrm{c}_1(V_{n-1})$ entschieden haben. Dazu betrachten wir die konkrete Situation in der Implementierung:

Nach (2.16) gilt mit $\mathbb{Z}_q \ni J = j(E^{\uparrow})$, dass

$$A = \frac{3J}{1728 - J} \quad \text{und} \quad B = \frac{2J}{1728 - J}.$$

Sei $J' \coloneqq j(E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1})$. $E^{\uparrow}/\ker \widehat{\phi}^{\uparrow}_{p,n-1}$ und $E^{(n-1)\uparrow}$ sind isomorph, folglich gilt

$$J' = j(E^{(n-1)\uparrow}) \quad (= \Sigma^{n-1}(J)).$$

Mit (2.11) ist $E^{(n-1)\uparrow}$ also gegeben durch

$$y^2 = x^3 + \left(\frac{3J'}{1728 - J'}\right) x + \left(\frac{2J'}{1728 - J'}\right).$$

Es folgt

$$u^{-2} = \frac{3\beta}{2\alpha}.$$

Man muss somit $\Sigma^{n-1}(A/B)$ gar nicht berechnen! Außerdem hängen α und β nach Korollar 3.19 nur von J (und $h(X)$) ab. Es ist nun auch unerheblich, ob wir eventuell mit dem kanonischen Lift eines Twists von E arbeiten. Algorithmus 4 muss lediglich mit der korrekten Approximierung aufgerufen werden[19]:

$$\mathrm{Tr}(\phi_q) \equiv \textsc{sqrt}(\mathrm{N}_{\mathbb{Q}_q/\mathbb{Q}_p}(3\beta/2\alpha),\ \mathrm{Tr}(\phi_q) \bmod p,\ N) \pmod{p^N}.$$

[18] Wir zeigen später, dass $\alpha, \beta \in \mathbb{Z}_q$ und deshalb gilt sogar $c \in \mathbb{Z}_q^*$.

[19] In der Literatur berechnet man zumeist eine beliebige Quadratwurzel und passt erst im Anschluss das Vorzeichen an.

Abschließend stellen wir die Berechnung von $h(X)$ aus Korollar 3.19 vor:

Sei also $G \coloneqq \ker \widehat{\phi}^{\uparrow}_{p,n-1}$ und

$$h(X) \coloneqq \prod_{P \in G^+} (X - \xi(P)) = \sum_{\nu=0}^{\#G^+} h_\nu X^{(\#G^+ - \nu)}.$$

Es gilt $\#G = p$ und $\deg(h) = \#G^+ = \frac{p-1}{2}$. Ferner ist G eine *Untergruppe* von $E^{\uparrow}[p]$: Sei $P \in G$, dann folgt aus

$$\mathrm{End}(E^{\uparrow}) \ni [p] = \phi^{\uparrow}_{p,n-1} \circ \widehat{\phi}^{\uparrow}_{p,n-1},$$

dass

$$[p](P) = \phi^{\uparrow}_{p,n-1}\left(\widehat{\phi}^{\uparrow}_{p,n-1}(P)\right) = \phi^{\uparrow}_{p,n-1}(\mathcal{O}) = \mathcal{O}.$$

Insbesondere ist $h(X)$ ein *Teiler* des p-ten Divisionspolynoms von $E^{\uparrow}$, das wir mit $\psi^{\uparrow}_p(X) \in \mathbb{Z}_q[X]$ bezeichnen. Es gilt

$$\pi(\psi^{\uparrow}_p) = \psi_p,$$

wobei $\psi_p(X) \in \mathbb{F}_q[X]$ das p-te Divisionspolynom von E ist. Diese Eigenschaft folgt aus Definition 2.24. Es liegt also nahe, dass wir $h(X)$ mit Hilfe von ψ_p berechnen können, indem wir einen geeigneten Faktor liften. Dies ist allerdings nicht so einfach.

Nach Satz 1.20 gilt $\#E^{\uparrow}[p] = p^2$. Also folgt aus (1.9), dass $\psi^{\uparrow}_p$ *quadratfrei* ist. Das gilt jedoch nicht mehr für ψ_p, denn[20]

$$\psi_p(X) = \varepsilon f(X)^p, \tag{3.11}$$

wobei $\varepsilon \in \mathbb{F}_q^*$ und $f(X) \in \mathbb{F}_q[X]$ normiert mit $\deg(f) = \frac{p-1}{2}$ ist. Insbesondere gilt

$$\deg(\psi_p) = \frac{p(p-1)}{2}.$$

Man beachte: $\#E[p] = p$, da E nach Voraussetzung gewöhnlich ist. Das Polynom $f(X)$ können wir nun folgendermaßen beschreiben: $\widehat{\phi}_{p,n-1}$ ist separabel[21] und somit

$$\#\ker \widehat{\phi}_{p,n-1} = p.$$

Also gilt in $\ker \widehat{\phi}_{p,n-1} \subset E[p]$ bereits *Gleichheit* und

$$f(X) = \prod_{P \in E[p]^+} (X - \xi(P)).$$

[20] Satz 1.24
[21] [Sil86, V., Theorem 3.1]

Definition 3.20. Man nennt die Vereinigung aller endlichen unverzweigten Erweiterungen von $\mathbb{Q}_p$ *maximale unverzweigte Erweiterung von* $\mathbb{Q}_p$, in Zeichen[22]

$$\mathbb{Q}_p^{\text{unram}} := \bigcup_{n=1}^{\infty} \mathbb{Q}_{p^n}.$$

Analog wird die Menge $\{x \in \mathbb{Q}_p^{\text{unram}} : \ |x|_p \leq 1\}$ mit $\mathbb{Z}_p^{\text{unram}}$ bezeichnet. Es gilt[23]

$$\mathbb{Z}_p^{\text{unram}}/p\mathbb{Z}_p^{\text{unram}} \ \cong \ \overline{\mathbb{F}}_p = \bigcup_{n=1}^{\infty} \mathbb{F}_{p^n},$$

wobei $p\mathbb{Z}_p^{\text{unram}}$ das einzige maximale Ideal von $\mathbb{Z}_p^{\text{unram}}$ ist.

$\mathbb{Q}_p^{\text{unram}}$ ist ein viel „kleinerer“ Körper als $\overline{\mathbb{Q}}_p$ und kann oftmals anstelle von $\overline{\mathbb{Q}}_p$ verwendet werden.

Lemma 3.21.

$$\ker \widehat{\phi}_{p,n-1}^{\uparrow} = E^{\uparrow}(\mathbb{Z}_p^{\text{unram}})[p].$$

Beweis. [Sat00, Corollary 3.3]. □

Es gilt[24] nun einerseits

$$h(X) \in \mathbb{Z}_q[X]$$

und damit insbesondere $\alpha, \beta \in \mathbb{Z}_q$. Andererseits erhält[25] man

$$\pi(h) = f,$$

denn $\pi(h)$ ist quadratfrei. Wie können wir $f(X)$ berechnen? Es gibt eine viel einfachere Methode als die „p-te Wurzel“ aus $\varepsilon^{-1}\psi_p(X)$ zu ziehen:

Betrachten wir dazu das p-te Divisionspolynom von $E^{(n-1)}$, das wir mit $\widetilde{\psi}_p(X)$ bezeichnen. Es gilt

$$\widetilde{\psi}_p(X) = \delta \widetilde{f}(X)^p,$$

[22] „maximal unramified extension of $\mathbb{Q}_p$“

[23] [Kob84, III.3]

[24] Wir geben folgende Beweisskizze, die wir einer freundlichen E-Mail von Takakazu Satoh verdanken: Da $\psi_p^{\uparrow}(X) \in \mathbb{Z}_q[X]$, ist $E^{\uparrow}[p]$ stabil unter $\text{Gal}(\overline{\mathbb{Q}}_q/\mathbb{Q}_q)$. Ebenso ist $E^{\uparrow}(\mathbb{Z}_p^{\text{unram}})$ invariant unter $\text{Gal}(\overline{\mathbb{Q}}_q/\mathbb{Q}_q)$. Nach Lemma 3.21 gilt also $h(X) \in \mathbb{Q}_q[X]$. Die Behauptung folgt aus $\mathbb{Q}_q \cap \mathbb{Z}_p^{\text{unram}} = \mathbb{Z}_q$.

[25] [Sat00, S. 258]

wobei $\delta \in \mathbb{F}_q^*$ und

$$\widetilde{f}(X) \;=\; \prod_{P \in E^{(n-1)}[p]^+} (X \;-\xi(P)\;).$$

Daraus folgt

$$\widetilde{f}(X)^p = \prod_{P \in E^{(n-1)}[p]^+} (X^p - \xi(P)^p) = f(X^p),$$

d. h.

$$\widetilde{\psi}_p(X) = \delta f(X^p).$$

Wir berechnen also zunächst $\widetilde{\psi}_p(X) = \sum_{\nu=0}^{(p-1)/2} u_\nu X^{p\nu}$ und anschließend

$$f(X) = \delta^{-1}\widetilde{\psi}_p(X^{1/p}) = (u_{(p-1)/2})^{-1} \sum_{\nu=0}^{(p-1)/2} u_\nu X^\nu.$$

Die Polynome $\psi_p(X)/f(X)$ und $f(X)$ sind wegen (3.11) nicht teilerfremd. Folglich können wir Lemma 2.15 *nicht* anwenden, um $h(X)$ zu approximieren. Allerdings erfüllen $\psi_p^\uparrow(X)$ und $\pi^{-1}(f) = \pi_1(h)$ die Voraussetzungen von Lemma 2.17 (vgl. [Sat00, Lemma 3.8 und Theorem 3.9]). Insbesondere gibt es ein $g(X) \in \mathbb{Z}_q[X]$, so dass

$$\psi_p^\uparrow(X) \equiv g(X)\pi^{-1}(f(X)) \pmod{p^2}.$$

Mit unserer Notation erhält man Algorithmus 6 (vgl. [CF06, Algorithm 17.36]):

Algorithmus 6 Berechnung von $h(X) = \prod_{P \in G^+} (X - \xi(P))$, wobei $G = \ker \widehat{\phi}_{p,n-1}^\uparrow$

Eingabe: $P(X) \in \mathbb{Z}_q[X], \widetilde{\psi}_p(X) \in \mathbb{F}_q[X]$ und $N \geq 2$, s. d. $P(X) \equiv \psi_p^\uparrow(X) \pmod{p^N}$.
Ausgabe: $H(X) \in \mathbb{Z}_q[X]$, so dass $H(X) \equiv h(X) \pmod{p^{N-1}}$.

1: **procedure** LIFT_KERNEL($P, \widetilde{\psi}_p, N$)
2: $H(X) \leftarrow \pi^{-1}(\delta^{-1}\widetilde{\psi}_p(X^{1/p}))$ ▷ $\pi(P(X)) = \varepsilon f(X)^p,\ \widetilde{\psi}_p(X) = \delta f(X^p)$
3: **for** $i \leftarrow$ precision$(N-1)$ **do**
4: $H(X) \leftarrow H(X) + \left(\frac{H'(X)P(X)}{P'(X)} \bmod H(X)\right) \bmod p^{i+1}$
5: **end for**
6: **return** $H(X)$
7: **end procedure**

In einer Implementierung hat man zwei Möglichkeiten:

- Man berechnet $P(X)$ (d. h. $\psi_p^{\uparrow}(X) \bmod p^N$) *einmal* und reduziert anschließend $P(X)$ und $P'(X)$ $\lceil \log_2(N-1) \rceil$-mal modulo dem jeweiligen $H(X)$.
- Man berechnet[26] $\lceil \log_2(N-1) \rceil$-mal $P(X) \bmod H(X)$ und $P'(X) \bmod H(X)$, wobei
$$P(X) \equiv \psi_p^{\uparrow}(X) \pmod{p^{i+1}}.$$

Man sollte auf jeden Fall die Laufzeiten beider Möglichkeiten vergleichen. In unserem Programm entscheiden wir uns ab $p = 61$ für die zweite Alternative.

Warnung: $P'(X) \bmod H(X)$ lässt sich *nicht* aus $P(X) \bmod H(X)$ berechnen! Man bestimmt deshalb zunächst

$$F(X) \coloneqq P(X) \bmod H(X)^2 \qquad (\deg(F) < p-1).$$

Dann gilt $P(X) \bmod H(X) = F(X) \bmod H(X)$ und

$$P'(X) \bmod H(X) = F'(X) \bmod H(X).$$

Diesen „Trick“ empfehlen wir auch für die erste Variante (anstatt nacheinander $P(X)$ und $P'(X)$ modulo $H(X)$ zu reduzieren)!

Bemerkung. In diesem Abschnitt haben wir *theoretisch* kennengelernt, wie man in Vercauterens Algorithmus den „Schritt zurück“ gehen könnte, denn es gilt

$$j(E^{\uparrow} / \ker \widehat{\phi}_{p,n-1}^{\uparrow}) = j(E^{(n-1)\uparrow}).$$

Der sehr aufwändige Algorithmus 6 auf der vorherigen Seite ist dafür allerdings ungeeignet, da man die gewonnene Genauigkeit umgehend wieder verliert.

Wir sind nun in der Lage Satohs Algorithmus zu formulieren. Nach Satz 1.31 (Hasse) gilt

$$\left|\mathrm{Tr}(\phi_q)\right| \leq 2\sqrt{q}.$$

Also folgt mit

$$[0, p^N - 1] \ni t \coloneqq \mathrm{Tr}(\phi_q) \bmod p^N \qquad (N \coloneqq \lceil \log_p 4 + n/2 \rceil),$$

dass

$$\mathrm{Tr}(\phi_q) = \begin{cases} t & \text{falls } t \leq 2\sqrt{q}, \\ t - p^N & \text{falls } t > 2\sqrt{q}. \end{cases}$$

Wir fassen alle bisherigen Resultate in Algorithmus 7 auf der nächsten Seite zusammen. Damit endet der theoretische Teil dieser Arbeit.

[26] Damit meinen wir, dass man bereits während der rekursiven Berechnung des Divisionspolynoms (zur Präzision $i+1$) fortlaufend modulo $H(X)$ bzw. $H(X)^2$ reduziert.

Bemerkung. Man beachte, dass wir aufgrund von Algorithmus 6 auf Seite 79 zunächst alle Berechnungen zur Präzision

$$N \coloneqq \lceil \log_p 4 + n/2 \rceil + 1$$

durchführen müssen! In [CF06, Algorithm 17.38, Zeile 1] wird mit $N = \lceil \log_p 4 + n/2 \rceil$ eine zu niedrige Genauigkeit definiert.

Algorithmus 7 Punktezählung elliptischer Kurven nach Satoh

Eingabe: $E/\mathbb{F}_{p^n} : \ y^2 = x^3 + ax + b,$ wobei $j(E) \notin \mathbb{F}_{p^2}$ und $p \geq 5$.
Ausgabe: $\#E(\mathbb{F}_{p^n})$.

procedure SATOH(a, b)
  $j \leftarrow j(E)$
  $N \leftarrow \lceil \log_p 4 + n/2 \rceil + 1$
  $J \leftarrow$ LIFT_J_INVARIANT(j, N)
  $Q \leftarrow J/(1728 - J)$
  $(A, B) \leftarrow (3Q, 2Q)$
  $\gamma \leftarrow \sigma^{n-1}(j/(1728 - j))$
  $\psi_p^{\uparrow}(X) \leftarrow p$-tes Divisionspolynom von $y^2 = x^3 + Ax + B$
  $\widetilde{\psi}_p(X) \leftarrow p$-tes Divisionspolynom von $y^2 = x^3 + 3\gamma x + 2\gamma$
  $\sum_{\nu=0}^{(p-1)/2} h_\nu X^{(p-1)/2-\nu} \leftarrow$ LIFT_KERNEL$(\psi_p^{\uparrow}, \widetilde{\psi}_p, N)$
  $\alpha \leftarrow (6 - 5p)A - 30(h_1^2 - 2h_2)$
  $\beta \leftarrow (15 - 14p)B - 70(-h_1^3 + 3h_1h_2 - 3h_3) + 42Ah_1$
  $\sum_{\nu=0}^{3(p-1)/2} c_\nu x^\nu \leftarrow (x^3 + ax + b)^{(p-1)/2}$
  $t \leftarrow \mathrm{SQRT}(\mathrm{N}_{\mathbb{Q}_{p^n}/\mathbb{Q}_p}(3\beta/2\alpha), \pi^{-1}(\mathrm{N}_{\mathbb{F}_{p^n}/\mathbb{F}_p}(c_{p-1})), N - 1)$
  if $t > 2\sqrt{p^n}$ **then**
    $t \leftarrow t - p^{N-1}$
  end if
  return $p^n + 1 - t$
end procedure

Beispiel 3.22. Sei $E : \ y^2 = x^3 + ax + b$ eine elliptische Kurve über $\mathbb{F}_{5^7}$, wobei

$$\begin{aligned} \mathbb{F}_{5^7} &\cong \mathbb{F}_5[X]/(1 + X + X^7), \\ a &= 3 + 4\underline{X} + \underline{X}^2 + 3\underline{X}^4 + \underline{X}^5, \\ b &= 2 + 2\underline{X} + 3\underline{X}^2 + 3\underline{X}^3 + 4\underline{X}^4 + 4\underline{X}^5 + 3\underline{X}^6. \end{aligned}$$

Es gilt $j \coloneqq j(E) = 2 + \underline{X} + \underline{X}^2 + 4\underline{X}^4 + 4\underline{X}^5 + 3\underline{X}^6 \notin \mathbb{F}_5$ ($\mathbb{F}_{25} \not\subset \mathbb{F}_{5^7}$). Wir befolgen nun Schritt für Schritt die Anweisungen von Algorithmus 7 um die Gruppenordnung $\#E(\mathbb{F}_{78125})$ zu berechnen. Die erforderliche Präzision N beträgt

$$\lceil \log_5 4 + 7/2 \rceil + 1 = 6 \quad (5^6 = 15625).$$

Algorithmus 5 auf Seite 57 berechnet

$$j(E^{\uparrow}) \equiv 7282+7631\overline{X}+11921\overline{X}^2+11895\overline{X}^3+2064\overline{X}^4+7654\overline{X}^5+3023\overline{X}^6 \pmod{5^6}.$$

Dabei verwenden wir, dass

$$\begin{aligned}\Phi_5(X,Y) \equiv\ & 125 + 13250(X+Y) + 3749XY + 14075(X^2+Y^2)\\ & + 15000(X^2Y+XY^2) + 625X^2Y^2 + 9425(X^3+Y^3)\\ & + 3875(X^3Y+XY^3) + 9250(X^3Y^2+X^2Y^3)\\ & + 3900X^3Y^3 + 3440(X^4+Y^4) + 9000(X^4Y+XY^4)\\ & + 12250(X^4Y^2+X^2Y^4) + 8675(X^4Y^3+X^3Y^4)\\ & + 5225X^4Y^4 + 4905(X^5+Y^5) + 10925(X^5Y+XY^5)\\ & + 4325(X^5Y^2+X^2Y^5) + 11560(X^5Y^3+X^3Y^5)\\ & + 3720(X^5Y^4+X^4Y^5) + 15624X^5Y^5 + X^6 + Y^6 \pmod{5^6}.\end{aligned}$$

Mit

$$A \equiv 4036 + 9791\overline{X} + 1414\overline{X}^2 + 8801\overline{X}^3 + 7506\overline{X}^4 + 7279\overline{X}^5 + 13958\overline{X}^6 \pmod{5^6},$$

$$B \equiv 7899 + 1319\overline{X} + 6151\overline{X}^2 + 659\overline{X}^3 + 5004\overline{X}^4 + 10061\overline{X}^5 + 4097\overline{X}^6 \pmod{5^6}$$

erhält man

$$\begin{aligned}\psi_5^{\uparrow}(X) \equiv\ & 4883 + 11885\overline{X} + 13002\overline{X}^2 + 5525\overline{X}^3 + 12136\overline{X}^4 + 7469\overline{X}^5 + 2114\overline{X}^6\\ & + (11605 + 12255\overline{X} + 12490\overline{X}^2 + 10075\overline{X}^3 + 11280\overline{X}^4 + 480\overline{X}^5\\ & + 4160\overline{X}^6)X + (15170 + 4720\overline{X} + 15585\overline{X}^2 + 4750\overline{X}^3 + 6970\overline{X}^4\\ & + 7445\overline{X}^5 + 9815\overline{X}^6)X^2 + (1185 + 135\overline{X} + 9555\overline{X}^2 + 7850\overline{X}^3\\ & + 14135\overline{X}^4 + 14760\overline{X}^5 + 9570\overline{X}^6)X^3 + (12545 + 6905\overline{X} + 13080\overline{X}^2\\ & + 2100\overline{X}^3 + 10545\overline{X}^4 + 3260\overline{X}^5 + 4490\overline{X}^6)X^4 + (6344 + 8746\overline{X}\\ & + 8856\overline{X}^2 + 6095\overline{X}^3 + 3444\overline{X}^4 + 13432\overline{X}^5 + 8268\overline{X}^6)X^5\\ & + (2260 + 8440\overline{X} + 12245\overline{X}^2 + 3765\overline{X}^3 + 8840\overline{X}^4 + 8325\overline{X}^5\\ & + 4765\overline{X}^6)X^6 + (1685 + 12390\overline{X} + 13120\overline{X}^2 + 11915\overline{X}^3 + 12440\overline{X}^4\\ & + 2425\overline{X}^5 + 3565\overline{X}^6)X^7 + (7195 + 14330\overline{X} + 7015\overline{X}^2 + 15130\overline{X}^3\\ & + 3555\overline{X}^4 + 850\overline{X}^5 + 2055\overline{X}^6)X^8 + (1620 + 1220\overline{X} + 9255\overline{X}^2\\ & + 420\overline{X}^3 + 10895\overline{X}^4 + 10680\overline{X}^5 + 9985\overline{X}^6)X^9 + (232 + 13292\overline{X}\\ & + 9543\overline{X}^2 + 14412\overline{X}^3 + 12247\overline{X}^4 + 13798\overline{X}^5 + 6021\overline{X}^6)X^{10}\\ & + 5X^{12} \pmod{5^6}.\end{aligned}$$

Daraus folgt insbesondere

$$\begin{aligned}\psi_5(X) &= 3 + 2\underline{X}^2 + \underline{X}^4 + 4\underline{X}^5 + 4\underline{X}^6 \\ &\quad + (4 + \underline{X} + \underline{X}^2 + 4\underline{X}^4 + 2\underline{X}^5 + 3\underline{X}^6)X^5 \\ &\quad + (2 + 2\underline{X} + 3\underline{X}^2 + 2\underline{X}^3 + 2\underline{X}^4 + 3\underline{X}^5 + \underline{X}^6)X^{10} \\ &= (2 + 2\underline{X} + 3\underline{X}^2 + 2\underline{X}^3 + 2\underline{X}^4 + 3\underline{X}^5 + \underline{X}^6) \cdot f(X)^5,\end{aligned}$$

wobei

$$\begin{aligned}f(X)^5 &= \underline{X} + 3\underline{X}^2 + 2\underline{X}^3 + \underline{X}^4 + 3\underline{X}^6 \\ &\quad + (3 + 2\underline{X} + \underline{X}^2 + 2\underline{X}^3 + 2\underline{X}^4 + 2\underline{X}^6)X^5 + X^{10}.\end{aligned}$$

Man beachte, dass $\psi_5(X)$ das 5-te Divisionspolyom von $y^2 = x^3 + (\frac{3j}{1728-j})x + (\frac{2j}{1728-j})$ (und nicht von E) ist. Weiter gilt

$$\begin{aligned}\widetilde{\psi}_5(X) &= \underline{X} + 4\underline{X}^2 + 2\underline{X}^3 + 3\underline{X}^5 + 3\underline{X}^6 \\ &\quad + (1 + \underline{X} + 4\underline{X}^2 + \underline{X}^3 + 3\underline{X}^4 + 4\underline{X}^5 + 2\underline{X}^6)X^5 \\ &\quad + (1 + 2\underline{X} + 2\underline{X}^3 + 3\underline{X}^4 + 2\underline{X}^5 + 4\underline{X}^6)X^{10} \\ &= (1 + 2\underline{X} + 2\underline{X}^3 + 3\underline{X}^4 + 2\underline{X}^5 + 4\underline{X}^6) \cdot f(X^5),\end{aligned}$$

wobei

$$\begin{aligned}f(X^5) &= 3 + \underline{X} + 4\underline{X}^2 + \underline{X}^4 + 4\underline{X}^5 + 4\underline{X}^6 \\ &\quad + (1 + 2\underline{X} + 3\underline{X}^2 + 3\underline{X}^3 + 3\underline{X}^4 + 3\underline{X}^5 + 3\underline{X}^6)X^5 + X^{10},\end{aligned}$$

d. h.

$$\begin{aligned}f(X) &= 3 + \underline{X} + 4\underline{X}^2 + \underline{X}^4 + 4\underline{X}^5 + 4\underline{X}^6 \\ &\quad + (1 + 2\underline{X} + 3\underline{X}^2 + 3\underline{X}^3 + 3\underline{X}^4 + 3\underline{X}^5 + 3\underline{X}^6)X + X^2.\end{aligned}$$

Mit Hilfe von Algorithmus 6 auf Seite 79 erhält man

$$\begin{aligned}h(X) \equiv\ & 263 + 1596\overline{X} + 2779\overline{X}^2 + 1565\overline{X}^3 + 2216\overline{X}^4 + 894\overline{X}^5 + 274\overline{X}^6 \\ & + (2726 + 827\overline{X} + 1603\overline{X}^2 + 603\overline{X}^3 + 473\overline{X}^4 + 2418\overline{X}^5 + 2658\overline{X}^6)X \\ & + X^2 \pmod{5^5}.\end{aligned}$$

Es gilt

$$3\beta/2\alpha \equiv 2238 + 601\overline{X} + 2114\overline{X}^2 + 2614\overline{X}^3 + 924\overline{X}^4 + 2084\overline{X}^5 + 579\overline{X}^6 \pmod{5^5}$$

und

$$\mathrm{N}_{\mathbb{Q}_{78125}/\mathbb{Q}_5}(3\beta/2\alpha) \equiv 939 \pmod{5^5}.$$

Um Algorithmus 4 auf Seite 45 zu initialisieren, berechnen wir

$$(x^3 + ax + b)^2 = \cdots + (1 + 3\underline{X} + 2\underline{X}^2 + \underline{X}^4 + 2\underline{X}^5)x^4 + x^6.$$

Daraus folgt

$$\mathrm{Tr}(\phi_{5^7}) \equiv 3 = \mathrm{N}_{\mathbb{F}_{78125}/\mathbb{F}_5}(1 + 3\underline{X} + 2\underline{X}^2 + \underline{X}^4 + 2\underline{X}^5) \pmod 5.$$

Als Endergebnis erhalten wir $\mathrm{Tr}(\phi_{5^7}) = 258$, d. h.

$$\#E(\mathbb{F}_{78125}) = 78125 + 1 - 258 = 77868.$$

3.4 Komplexitätsbetrachtungen und Anmerkungen

Soweit nicht anders angegeben, wurden die folgenden Abschätzungen aus [BSS05, VI.] übernommen (siehe auch [CF06, 17.3.1]).

Der SEA-Algorithmus zur Bestimmung der Gruppenordnung $\#E(\mathbb{F}_q)$ hat eine Zeitkomplexität von $O(\ln^{2\mu+2} q)$ und eine Platzkomplexität von $O(\ln^2 q)$, wobei μ eine Konstante ist, so dass die Multiplikation von zwei m-Bit Zahlen eine Laufzeit von $O(m^\mu)$ hat. Beispielsweise gilt $\mu = 2$ für die klassische Multiplikation und $\mu = \log_2 3$ für die Multiplikation nach Karatsuba[27].

Im Vergleich dazu hat Satohs Algorithmus für ein festes p (die Charakteristik von $\mathbb{F}_{p^n} = \mathbb{F}_q$) eine Zeitkomplexität von $O(n^{2\mu+1})$ und eine Platzkomplexität von $O(n^3)$ ($O(n^2)$ mit Algorithmus 5 auf Seite 57). Die O-Konstante der Zeitkomplexität hängt allerdings stark von p ab ($O(p^2 \ln^\mu p)$). Deshalb ist der in dieser Arbeit beschriebene Algorithmus nur für kleine Charakteristiken praktikabel.

Satoh formulierte seinen Algorithmus ursprünglich für $p \geq 5$. Allerdings gelang es rasch, ihn auf Charakteristik 2 und 3 zu erweitern (siehe [FGH00] und [Skj03]). Einen Überblick über die zahlreichen Verbesserungen und Weiterentwicklungen von Satohs Algorithmus findet man in [CF06, 17.3.1.f]. Der resultierende (asymptotisch) optimale Punktezählalgorithmus von Harley hat für ein festes p eine Zeitkomplexität von $O(n^{2\mu} \ln n)$ (und eine Platzkomplexität von $O(n^2)$).

Ich möchte abschließend einige wichtige Ergebnisse dieser Arbeit hervorheben.

Zunächst haben wir in Kapitel 1 gesehen, wie wir mit McKees Formeln p-te Divisionspolynome in $\mathbb{Z}_{p^n}$ und $\mathbb{F}_{p^n}$ teilweise deutlich schneller als mit der rekursiven Methode berechnen können. Nach meinem Wissen hatte [McK94] bisher keine Anwendung in

[27] [CF06, 10.3.2]

der Praxis. Schließlich wurde der Spezialfall $j(E) \in \mathbb{F}_{p^2}$ untersucht und eine implementierungsnahe Berechnung von $\#E(\mathbb{F}_{p^n})$ mit Hilfe von Lemma 1.32 vorgestellt.

In Kapitel 2 wurden zunächst Grundalgorithmen wie INVERSE und NEWTON mathematisch hergeleitet und für den Lift der j-Invariante verwendet. Vercauterens Algorithmus, der durch seine kompakte Fassung besticht, wurde nicht nur bewiesen, sondern auch anschaulich erläutert.

Um den letzten Teil von Satohs Algorithmus zu formulieren, wird häufig das *invariante Differential* einer elliptischen Kurve (siehe [Sil86, III., §5]) verwendet. Im Gegensatz dazu benutze ich in Kapitel 3 formale Gruppen und verwende damit Satohs ursprüngliche Notation aus [Sat00] (letztlich unterscheiden sich beide Varianten nicht wesentlich). Die Idee, Satohs Algorithmus mit Hilfe einer Normberechnung zu verkürzen, wird in den Abschnitten 3.2 und 3.3 konsequent umgesetzt.

Ausführliche Informationen über die Implementierung der vorgestellten mathematischen Theorie findet man im nächsten Kapitel.

4 Implementierung

Der Rahmen für die Implementierung von Satohs Algorithmus wurde im August 2007 bei einem Treffen mit Herrn Dr. Erwin Heß und Herrn Anton Kargl festgelegt. Das fertige Programm sollte für alle Charakteristiken[1] $5 \leq p \leq 251$ die Gruppenordnung $\#E(\mathbb{F}_{p^n})$ (bei Eingabe einer beliebigen elliptischen Kurve $E/\mathbb{F}_{p^n}$) berechnen, wobei die konkrete Anwendung in der Größenordnung $p^n \approx 2^{160}$ beginnt. Eine automatisierte Suche von kryptographisch starken Kurven, d. h. in erster Linie elliptischen Kurven mit *primer* Gruppenordnung, sollte die nächste Entwicklungsstufe sein. Diese Vorgaben konnten erfolgreich erfüllt werden.

Als Programmiersprache wird `C++` verwendet, wobei zusätzlich die Bibliothek `NTL`[2] (Version `5.4.2`, kompiliert mit `GMP 4.2.4`[3]), die bereits eine Vielzahl von Datenstrukturen und Algorithmen zur Verfügung stellt, zum Einsatz kommt. Parallel zu `C++` entstand in der Anfangsphase eine Implementierung in `ARIBAS`[4], die allerdings nach der Umsetzung von Algorithmus 5 auf Seite 57 nicht weiterentwickelt wurde.

Eine obere Grenze für p ist innerhalb des Quelltextes nicht festgelegt. Eine Verwendung hängt damit theoretisch nur von der Verfügbarkeit der Koeffizienten des p-ten modularen Polynoms $\Phi_p(X, Y)$ ab, da das Programm diese nicht selbst berechnet. Tests wurden allerdings „nur“ bis zur Charakteristik 293 durchgeführt. Es sei darauf hingewiesen, dass die in Abschnitt 1.5 vorgestellte Behandlung des Spezialfalls $j(E) \in \mathbb{F}_{p^2}$ ebenfalls implementiert wurde. Somit unterliegen die Eingabedaten prinzipiell keiner Einschränkung.

Im Folgenden werden die beiden Programme `satoh` und `search` im Detail vorgestellt und einige Laufzeiten präsentiert. Die im Anhang C abgedruckten Dateien sollen den Einblick in die praktische Umsetzung von Satohs Algorithmus abrunden. Der gesamte Quellcode umfasst ca. 3 800 Zeilen und befindet sich auf der beigefügten DVD (siehe Anhang B).

---

[1] Herr Kargl entwickelte 2002 im Rahmen seiner Diplomarbeit eine Implementierung für $p = 2$, die noch heute zum Einsatz kommt.

[2] [Sho]

[3] [Gra]

[4] [For]

Mit `search` konnte ich u. a. für alle $p \in [5, 293]$ elliptische Kurven $E/\mathbb{F}_{p^n}$ finden, so dass $\#E(\mathbb{F}_{p^n}) \approx 2^{256}$ eine Primzahl ist (dabei ist der Erweiterungsgrad n ebenfalls stets prim). Die Grenzen der Kurvensuche sind damit natürlich noch längst nicht erreicht.

Beispiel 4.1. Sei $E: \; y^2 = x^3 + ax + b$ eine elliptische Kurve über

$$\mathbb{F}_{293^{23}} \cong \mathbb{F}_{293}[X]/(50 + X + X^{23}),$$

wobei

$$a = [203\ 171\ 249\ 203\ 85\ 160\ 45\ 251\ 62\ 85\ 235\ 135\ 31\ 10\ 230\ 253\ 38\ 285\ 162\ 194\ 2\ 222\ 161],$$
$$b = [258\ 15\ 96\ 71\ 263\ 213\ 23\ 13\ 72\ 65\ 158\ 249\ 95\ 281\ 188\ 246\ 268\ 216\ 165\ 17\ 19\ 250\ 135].$$

Dann gilt für $k := \#E(\mathbb{F}_{293^{23}})$, dass $\lceil \log_2(k) \rceil = 189$ und

$$k = 5469596237510178644993898339276906090271251276704713403 23 \in \mathbb{P}.$$

Die Berechnung dauerte 2:45 Minuten (ohne `GMP` ca. 27 Minuten). Dabei (und im Weiteren) wird die Notation $[e_0 e_1 \dots e_{n-1}]$ für $\mathbb{F}_{p^n} \ni \sum_{\nu=0}^{n-1} e_\nu \underline{X}^\nu$ verwendet, wobei $\underline{X} := X + (f(X)),\ \mathbb{F}_{p^n} \cong \mathbb{F}_p[X]/(f(X))$.

Alle Laufzeiten in dieser Arbeit wurden auf einer Intel Core2 Duo @ 2 GHz Architektur ermittelt. Der Quellcode wurde mit `g++ (GCC) 3.4.4` kompiliert und unter einer `Cygwin`[5]-Umgebung ausgeführt.

Wir stellen noch die Ordnerstruktur vor, in der `satoh` und `search` ausgeführt werden. Dies erleichtert das Verständnis der Programmausgabe in den nächsten beiden Abschnitten.

`cpp/Satoh2/build` In diesem Verzeichnis befinden sich die ausführbaren Dateien. Relative Pfadangaben beziehen sich also stets auf diese Position.

`divisionpolynomials/cpp` Hier werden die Vorberechnungsdaten für McKees Methode[6] zur Berechnung von Divisionspolynomen gespeichert.

`ec/cpp` In diesem Ordner werden die mit `search` gefundenen elliptischen Kurven abgelegt. Es werden folgende Daten (sortiert nach p, n) gespeichert: $f(X)$, a, b, $j(E)$ und $\#E(\mathbb{F}_{p^n})$.

`irreduciblepolynomials/cpp` `satoh` und `search` suchen (bei fehlerhafter Eingabe) in diesem Verzeichnis nach einem passenden $f(X)$ um die Körpererweiterung $\mathbb{F}_p \subset \mathbb{F}_{p^n}$ zu definieren. Sollten die benötigten Daten nicht vorhanden sein, wird automatisch ein zufälliges irreduzibles Polynom vom Grad n über $\mathbb{F}_p$ konstruiert. Normalerweise erzeugt `NTL` jedoch keine „sparse polynomials“.

[5] [Cyg]
[6] Lemma 1.25

modularpolynomials/cpp Hier werden die modulo p^N reduzierten Koeffizienten des p-ten modularen Polynoms $\Phi_p(X, Y)$ gespeichert.

modularpolynomials/dec Sollten sich in dem vorherigen Ordner (noch) keine passenden Daten befinden, wird in diesem Verzeichnis nach unreduzierten Koeffizienten gesucht.

4.1 satoh

Wir beginnen mit einem Beispiel und erläutern im Anschluss die Programmausgabe.

```
Alexander@vostro /cygdrive/e/Diplomarbeit/cpp/Satoh2/build
$ ./satoh-2.1

p? 16
p must be prime. Set p = 17.
n? 22
f? []
invalid
../../../irreduciblepolynomials/cpp/17_22.dat loaded

a? []
b? []
E: y^2 = x^3 + ax + b is singular. Use random elliptic curve.

f=[5 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
a=[3 0 11 15 4 13 10 3 16 16 15 8 6 16 11 16 4 1 5 12 8 8]
b=[9 2 0 0 13 9 2 0 3 6 16 8 11 10 11 0 12 2 7 10 6 5]
j=[16 6 0 9 13 15 1 16 15 9 11 11 7 15 15 6 9 11 6 1 10 9]
../../../modularpolynomials/cpp/phi_17_mod_13.dat loaded
lift j-invariant... ( 0.25 sec)
../../../divisionpolynomials/cpp/psi_17.dat loaded
compute 17th division polynomial... ( 0.00 sec)
lift kernel...
../../../divisionpolynomials/cpp/psi_17_mod_13.dat loaded
compute 17th division polynomial... ( 0.02 sec)
reduce division polynomial (and its derivative)... ( 0.09 sec)
reduce division polynomial (and its derivative)... ( 0.09 sec)
reduce division polynomial (and its derivative)... ( 0.09 sec)
reduce division polynomial (and its derivative)... ( 0.09 sec)
done ( 0.61 sec)
compute norm... (-0.00 sec)
trace computed (total time:  0.88 sec)
t=-62857030803368
```

```
#E(GF(17^22))=11745628765212113160048660 58
check group order... ok ( 0.03 sec)

new parameters? (y/n)
```

Zunächst wird die fehlerhafte Eingabe `16` korrigiert und die nächstgrößere Primzahl `17` gewählt. Der Erweiterungsgrad n soll `22` sein. Der endliche Körper $\mathbb{F}_{17^{22}}$ lässt sich natürlich nicht mit dem Nullpolynom definieren, allerdings befindet sich in `17_22.dat` ein gültiger Eintrag:

```
[5 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
```

Das Programm benutzt also die Isomorphie

$$\mathbb{F}_{17^{22}} \cong \mathbb{F}_{17}[X]/(5 + 2X + X^2 + X^{22}).$$

Danach werden wir aufgefordert die Koeffizienten der elliptischen Kurve einzugeben. Wir setzen naiv $a = 0$ und $b = 0$. `satoh` erkennt natürlich, dass $y^2 = x^3$ *keine* elliptische Kurve definiert und wählt stattdessen Zufallsparameter. Es folgt eine kurze Zusammenfassung, wobei die berechnete j-Invariante $j(E)$ ebenfalls mitgeteilt wird.

Bevor das Programm die j-Invariante des kanonischen Lifts von E approximiert, werden die Koeffizienten von $\Phi_{17}(X, Y)$ zur Genauigkeit

$$N = \lceil \log_{17} 4 + 22/2 \rceil + 1 = 13$$

geladen. Alternativ erhält man Ausgaben wie z. B.

```
../../../modularpolynomials/dec/phi_17.dec loaded
../../../modularpolynomials/cpp/phi_17_mod_13.dat written
../../../modularpolynomials/cpp/phi_17_mod_13.dat loaded
```

oder

```
../../../modularpolynomials/cpp/phi_17_mod_24.dat loaded
../../../modularpolynomials/cpp/phi_17_mod_13.dat written
../../../modularpolynomials/cpp/phi_17_mod_13.dat loaded
```

`satoh` prüft also auch, ob die benötigten Daten vielleicht zu einer höheren Genauigkeit (in diesem Fall modulo 17^{24}) vorliegen. Anschließend werden wir informiert, dass $j(E^{\uparrow}) \bmod 17^{13}$ in 0,25 Sekunden berechnet wurde.

Um das 17-te Divisionspolynom $\widetilde{\psi}_{17}(X) \in \mathbb{F}_{17^{22}}[X]$ von $E^{(21)}$ zu berechnen, werden dann die nach Lemma 1.25 vorberechneten Daten geladen bzw. erst erstellt:

```
compute coefficients of the 17th division polynomial (mod 17)... ( 0.00 sec)
../../../divisionpolynomials/cpp/psi_17.dat written
../../../divisionpolynomials/cpp/psi_17.dat loaded
```

Die Berechnung von $h(X)$ mod 17^{12} dauert insgesamt 0,61 Sekunden. Dabei informiert das Programm über zeitintensive Zwischenschritte. In unserem Beispiel wird zunächst das 17-te Divisionspolynom von $E^{\uparrow}$ zur Präzision 13 berechnet und anschließend viermal[7] modulo dem jeweiligen $H(X)$ reduziert. Für größere p erhält man stattdessen (vgl. Abschnitt 3.3) eine Ausgabe der Form

```
lift kernel...
compute reduced 89th division polynomial recursively... ( 2.94 sec)
compute reduced 89th division polynomial recursively... ( 2.92 sec)
compute reduced 89th division polynomial recursively... ( 3.72 sec)
compute reduced 89th division polynomial recursively... ( 5.06 sec)
done (15.58 sec)
```

Die Programmausgabe endet mit der berechneten Spur des Frobenius sowie

$$\#E(\mathbb{F}_{17^{22}}) = 17^{22} + 1 - (-62857030803368).$$

Ein abschließender Test überprüft, ob

$$[117456287652121131600486 6058](P) = \mathcal{O},$$

wobei $P \in E(\mathbb{F}_{17^{22}}) \setminus \{\mathcal{O}\}$ zufällig gewählt wird.

Wie man an der Ausgabe einer zweiten Berechnung (`new parameters? (y/n) y`) erkennen kann, bleiben die bereits geladenen Daten gespeichert. Ein Neustart von `satoh` mit den obigen Eingaben verwendet nebenbei bemerkt *dieselben* „zufälligen“ Koeffizienten a und b, was vorallem zu Testzwecken hilfreich ist. Bei `search` ist dies natürlich nicht der Fall. Schließlich möchte man nach einer ersten abgebrochenen Suche nicht nocheinmal dieselben elliptischen Kurven überprüfen oder (viel schlimmer) immer dieselbe elliptische Kurve finden.

Betrachten wir noch den Spezialfall $j(E) \in \mathbb{F}_{289}$:

```
a? [-1]
b? [-3]

f=[5 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
a=[16]
b=[14]
```

[7] # precision$(13 - 1) = 4$

```
j=[7]
j in GF(17)
t=22707972935975
#E(GF(17^22))=117456287652112575100112671 5
check group order... ok ( 0.09 sec)

new parameters? (y/n) y

a? [16 2 10 3 6 2 12 10 10 9 2 14 13 12 13 16 2 4 8 14 12 14]
b? [2 13 14 11 5 13 10 14 14 16 13 6 8 10 8 2 13 9 1 6 10 6]

f=[5 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
a=[16 2 10 3 6 2 12 10 10 9 2 14 13 12 13 16 2 4 8 14 12 14]
b=[2 13 14 11 5 13 10 14 14 16 13 6 8 10 8 2 13 9 1 6 10 6]
j=[8 16 12 7 14 16 11 12 12 4 16 10 2 11 2 9 16 15 13 10 11 10]
j in GF(17^2)
t=66547133948621
#E(GF(17^22))=117456287652108191184011 4069
check group order... ok ( 0.08 sec)
```

`satoh` benutzt dabei die theoretischen Überlegungen aus Abschnitt 1.5.

In Tabelle 5 auf Seite 94 fassen wir für $p \in [5, 113]$ und $p^n \approx 2^{160}$ die Laufzeiten (in Sekunden) von `satoh` (Version 2.1) zusammen. Der Erweiterungsgrad n ist dabei ebenfalls eine Primzahl. Die Körpererweiterung $\mathbb{F}_p \subset \mathbb{F}_{p^n}$ besitzt also keine echten Zwischenkörper. Die Werte $t_{\text{j\_invariant}}$, t_{kernel} und t entsprechen den folgenden Zeilen der Programmausgabe:

```
lift j-invariant... ( 0.25 sec)
done ( 0.61 sec)
trace computed (total time:  0.88 sec)
```

In diesem Fall ist also $t = 0{,}88$, $t_{\text{j\_invariant}}/t \approx 0{,}28$ und $t_{\text{kernel}}/t \approx 0{,}69$. Man beachte, dass t_{kernel} *nicht* die Laufzeit von Algorithmus 6 auf Seite 79 wiedergibt, da in der Implementierung nur $\widetilde{\psi}_p(X)$ und N als Parameter übergeben werden.

Der Wert $t_{\text{j\_invariant}}/t$ sinkt kontinuierlich auf ca. 8 Prozent, während t_{kernel}/t schließlich über 90 Prozent beträgt. Dies liegt an der bereits erwähnten Abhängigkeit

$$\deg(\psi_p^{\uparrow}) = \frac{p^2 - 1}{2} = O(p^2),$$

d. h. die Berechnung des p-ten Divisionspolynoms von $E^{\uparrow}$ ist für „große“ Primzahlen sehr aufwändig (selbst wenn wir „nur“ $\psi_p^{\uparrow}(X) \bmod H(X)$ berechnen). Der enorme

Zeitgewinn[8] durch die Umformulierung (3.7) ist nun offensichtlich: Berechnet man *alle* $c_1(V_i)$ $(i \in [0, n-1])$, so erhöht sich die Gesamtlaufzeit ungefähr[9] um den Wert

$$(n-1) \cdot t_{\text{kernel}}.$$

Tabelle 5 setzen wir in Abbildung 4 fort ($p \in [127, 293]$). Die Entwicklung von t in Abhängigkeit von $n \in [30, 55]$ entnimmt man Abbildung 5 auf Seite 95. Dabei liegt p im Bereich $[5, 37]$, wobei wir den Verlauf für $p \geq 17$ nur skizzieren.

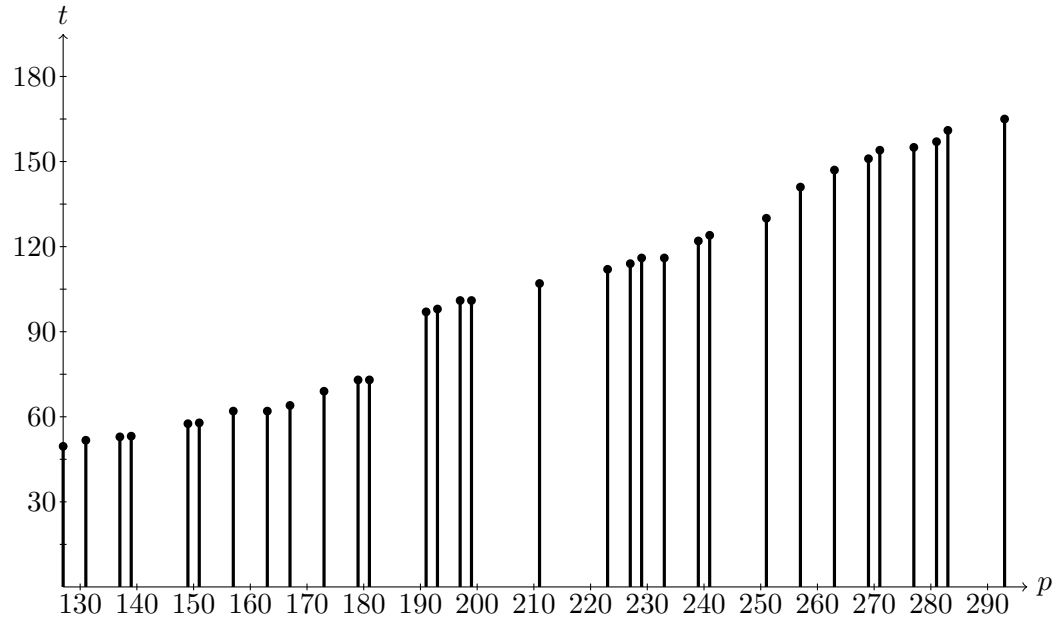


Abbildung 4: Laufzeiten satoh, $p \in [127, 293]$, $n = 23$

4.2 search

Das Programm search benutzt die gleichen Komponenten wie satoh und dient der komfortablen Suche nach elliptischen Kurven $E/\mathbb{F}_{p^n}$ mit primer Gruppenordnung $\#E(\mathbb{F}_{p^n})$. Diese Kurven können in der Kryptographie verwendet werden[10], falls die

[8] Voraussetzung ist natürlich eine schnelle Normberechnung. Wir benutzen in der Implementierung die von NTL bereitgestellte Methode zur Berechnung von Resultanten (siehe Proposition 2.14).

[9] Wir vernachlässigen, dass man in diesem Fall *alle* j-Invarianten des Zyklus (2.12) liften muss.

[10] Nebenbei bemerkt genügt es, dass $\#E(\mathbb{F}_{p^n})$ durch eine große Primzahl teilbar ist (vgl. [Was08, 5.2.3]).

Tabelle 5: Laufzeiten `satoh`, $p \in [5, 113]$, $p^n \approx 2^{160}$ (n prim)

p	n	t	$t_{\text{j\_invariant}}/t$	t_{kernel}/t	$\pi(p)$
5	71	2,45	0,780	0,192	3
7	59	1,98	0,662	0,318	4
11	47	2,91	0,419	0,564	5
13	47	4,33	0,333	0,661	6
17	41	4,52	0,325	0,670	7
19	41	5,17	0,284	0,710	8
23	37	6,17	0,240	0,755	9
29	37	9,00	0,216	0,778	10
31	37	10,62	0,184	0,812	11
37	31	8,22	0,145	0,852	12
41	31	10,67	0,125	0,873	13
43	31	11,47	0,124	0,874	14
47	29	11,84	0,110	0,889	15
53	29	16,22	0,105	0,892	16
59	29	19,41	0,097	0,901	17
61	29	19,58	0,101	0,897	18
67	29	22,47	0,101	0,898	19
71	29	24,62	0,101	0,897	20
73	29	25,94	0,095	0,903	21
79	29	32,64	0,081	0,917	22
83	29	34,36	0,083	0,915	23
89	29	39,16	0,085	0,913	24
97	29	45,59	0,088	0,910	25
101	29	50,56	0,082	0,916	26
103	29	50,80	0,084	0,914	27
107	29	53,00	0,084	0,915	28
109	29	56,27	0,081	0,917	29
113	29	57,06	0,083	0,915	30

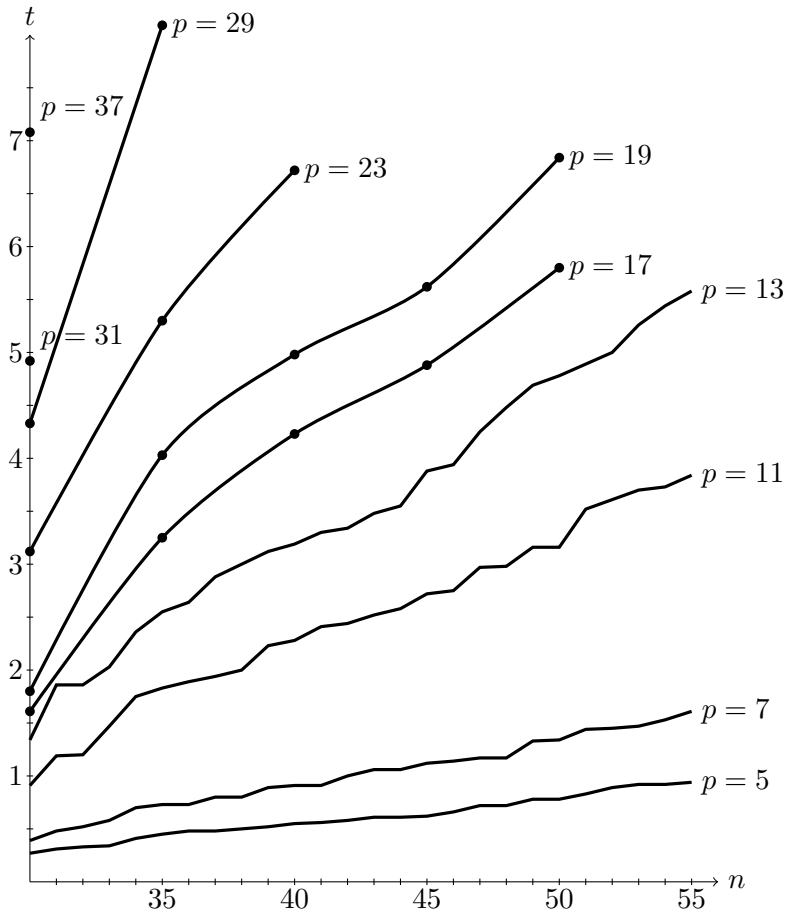


Abbildung 5: Laufzeiten `satoh`, $p \in [5, 13],\ n \in [30, 55]$

Ordnung groß genug ist ($> 2^{160}$). Man beachte, dass die Kurve noch weitere Bedingungen erfüllen muss; diese werden in der vorliegenden Implementierung allerdings nicht überprüft. Einen Überblick findet man in [BSS99, V.7].

Wir beginnen erneut mit einem Beispielaufruf:

```
Alexander@vostro /cygdrive/e/Diplomarbeit/cpp/Satoh2/build
$ ./search-2.1

p? 7
n? 19
f? []
invalid
```

```
../../../irreduciblepolynomials/cpp/7_19.dat loaded

early-abort strategy? (y/n) n

.
.
.

f=[6 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
a=[0 6 5 0 5 1 6 3 6 6 0 5 1 2 2 6 4 0 4]
b=[5 3 1 0 6 1 6 3 3 1 2 2 0 6 3 0 4 3]
j=[6 4 1 4 5 0 0 1 4 3 4 2 5 0 4 0 1 2 1]
lift j-invariant... ( 0.05 sec)
compute 7th division polynomial... (-0.00 sec)
lift kernel...
compute 7th division polynomial... (-0.00 sec)
reduce division polynomial (and its derivative)... ( 0.02 sec)
reduce division polynomial (and its derivative)... (-0.00 sec)
reduce division polynomial (and its derivative)... ( 0.00 sec)
reduce division polynomial (and its derivative)... ( 0.00 sec)
done ( 0.06 sec)
compute norm... (-0.00 sec)
trace computed (total time:  0.11 sec)
t=61083839
#E(GF(7^19))=11398895124289305
check group order... ok ( 0.02 sec)

order of twisted curve is prime...
a=[3 5 1 4 6 4 5 0 2 4 3 3 0 2 4 4 6]
b=[1 3 6 2 0 2 4 3 1 3 6 3 2 5 1 4 4 0 3]
#E(GF(7^19))=11398895246456983
check group order... ok ( 0.02 sec)

../../../ec/cpp/7_19.dat written
../../../ec/aribas/7_19.dat written

Elliptic curve with prime order found after 12 tries.
Running Satoh on 12 curves.
Total time:  1.62 sec

new search? (y/n)
```

Die Programmausgabe geben wir verkürzt wieder. Ein prüfender Blick enthüllt nichts Außergewöhnliches: Es werden solange Zufallskurven erzeugt, bis $k \coloneqq \#E(\mathbb{F}_{p^n})$ eine

Primzahl ist[11]. Die berechnete Gruppenordnung wird dabei stets überprüft:

$$[k](P) \stackrel{?}{=} \mathcal{O}.$$

Zudem untersucht `search`, ob

$$2(p^n + 1) - k$$

eine Primzahl ist (vgl. Lemma 1.27).

Wir starten das Programm erneut (mit $p^n = 127^{23}$), wobei wir allerdings ein „frühes Abbruchverfahren“ (`early-abort strategy`) benutzen:

```
Alexander@vostro /cygdrive/e/Diplomarbeit/cpp/Satoh2/build
$ ./search-2.1

p? 127
n? 23
f? []
invalid
../../../irreduciblepolynomials/cpp/127_23.dat loaded

early-abort strategy? (y/n) y
l-torsion point search for l<=? 5

f=[13 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
a=[34 81 51 18 18 42 110 123 9 100 5 26 39 68 106 93 111 122 38 2 16 105 117]
b=[22 22 27 67 6 27 26 126 32 59 76 101 70 23 29 4 107 82 63 52 25 12 17]
j=[71 48 29 38 93 32 125 48 78 27 112 3 57 21 38 86 74 55 25 71 21 49 92]
search small prime factor of #E(GF(127^23))  2  ( 0.02 sec)

throw curve away...

f=[13 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
a=[17 9 13 67 120 6 103 94 7 92 84 65 8 30 41 50 26 19 2 1 94 126 62]
b=[58 116 46 15 67 110 81 82 85 39 57 1 118 94 9 72 8 107 74 67 77 23 123]
j=[33 47 67 4 121 96 104 9 18 11 32 41 41 45 53 71 33 106 91 64 8 76 9]
search small prime factor of #E(GF(127^23))  .  .  5  ( 0.34 sec)

throw curve away...

.
.
.
```

[11] Dieser einfache Ansatz ist tatsächlich die bevorzugte Methode um kryptographisch starke elliptische Kurven zu generieren (vgl. [BSS99, Algorithm VI.1]). Eine andere Möglichkeit wird in [BSS99, VIII.] vorgestellt.

```
f=[13 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
a=[7 91 76 63 106 10 62 104 97 118 126 113 92 42 90 55 125 76 98 41 6 114 1]
b=[37 76 45 33 2 34 95 82 107 53 60 63 110 32 65 66 100 102 91 54 56 84 43]
j=[64 96 10 16 38 85 72 86 108 15 69 57 3 112 46 37 115 19 110 92 35 95 90]
search small prime factor of #E(GF(127^23))  .  .  .  ( 0.33 sec)
lift j-invariant... ( 3.52 sec)
compute 127th division polynomial... ( 0.06 sec)
lift kernel...
compute reduced 127th division polynomial recursively... ( 6.80 sec)
compute reduced 127th division polynomial recursively... ( 6.75 sec)
compute reduced 127th division polynomial recursively... (10.81 sec)
compute reduced 127th division polynomial recursively... (16.91 sec)
done (44.33 sec)
compute norm... (-0.00 sec)
trace computed (total time: 47.94 sec)
t=-124017885667043527531483 5
#E(GF(127^23))=24405380440023204613644700962000461850461834368 19
check group order... ok ( 0.06 sec)

../../../ec/cpp/127_23.dat written
../../../ec/aribas/127_23.dat written

Elliptic curve with prime order found after 145 tries.
Running Satoh on 23 curves.
Total time: 18:45

new search? (y/n)
```

Man erkennt den Vorteil dieses Verfahrens: die Berechnung der Gruppenordnung k dauert ca. 48 Sekunden; einen Primteiler ≤ 5 von k findet[12] `search` aber bereits in maximal 0,34 Sekunden. In diesem Fall wird die Berechnung sofort abgebrochen und eine neue Kurve erzeugt. Die Gruppenordnung k wird natürlich nur noch bei der „letzten“ Kurve überprüft.

Ist es außergewöhnlich, dass wir nach „nur“ 145 Versuchen eine passende Kurve finden? Wir beantworten diese Frage mathematisch:

Nach Hasse[13] gilt

$$p^n + 1 - 2\sqrt{p^n} \;\leq\; k \;\leq\; p^n + 1 + 2\sqrt{p^n}.$$

[12] siehe Abschnitt 1.4
[13] Satz 1.31

Der *Primzahlsatz*[14] besagt, dass

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

In dem angegebenen Intervall liegen also ungefähr

$$\pi_{p,n} \coloneqq \frac{p^n + 1 + 2\sqrt{p^n}}{\ln(p^n + 1 + 2\sqrt{p^n})} - \frac{p^n + 1 - 2\sqrt{p^n}}{\ln(p^n + 1 - 2\sqrt{p^n})}$$

Primzahlen. Für eine zufällige elliptische Kurve $E/\mathbb{F}_{p^n}$ ist deshalb $k = \#E(\mathbb{F}_{p^n})$ mit der Wahrscheinlichkeit

$$\frac{\pi_{p,n}}{4\sqrt{p^n}}$$

eine Primzahl. Dabei nehmen wir an, dass $\#E(\mathbb{F}_{p^n})$ im Hasse-Intervall gleichverteilt ist (siehe [BSS99, VI.5.]).

In unserem Beispiel gilt

$$\frac{\pi_{127,23}}{4\sqrt{127^{23}}} \approx 0{,}009.$$

Nach $0{,}009^{-1} \approx 111$ Zufallskurven erwartet man also im Durchschnitt einen „Treffer“. Die Wahrscheinlichkeit nach 300 Zufallskurven noch *keine* geeignete gefunden zu haben, beträgt nur etwa 6,6 Prozent ($\approx 0{,}991^{300}$). Tatsächlich ist die Wahrscheinlichkeit noch kleiner, da wir mit Satohs Algorithmus *zwei* Ordnungen (quadratischer Twist) erhalten, die unabhängig voneinander prim sein können.

Insofern kann man sagen, dass unsere Suche etwas langsam war. Schnellere Sucherfolge sind realistisch (und wurden in mehreren Testläufen auch beobachtet).

Es bleibt noch zu klären, ob der Anteil von $145 - 23 = 122$ ausgesiebten Kurven zu erwarten war. Nach Tabelle 1 auf Seite 14 gilt

$$\mathrm{P}_5 \approx 0{,}73,$$

d. h. von 145 zufälligen elliptischen Kurven sollten wir ca. $0{,}73 \cdot 145 \approx 106$ aussieben können. Diese Vorhersage stimmt mit dem beobachteten Wert gut überein (ein einzelnes Beispiel ist natürlich nicht sehr aussagekräftig; größere Abweichungen sind normal).

[14] [FB06, VII., 4.5]

4.3 Ausblick

Die Verwendung von NTL hat eine Reihe von Vorteilen. An dieser Stelle erwähne ich das umfangreiche Paket von (schnellen) Algorithmen und Datenstrukturen, mit dessen Unterstützung ich mich ganz auf die Umsetzung von Satohs Algorithmus konzentrieren konnte. In Verbindung mit GMP gelang es zudem in neue Geschwindigkeitsregionen vorzudringen.

Es ist fraglich, ob man bei einem Verzicht auf NTL überhaupt in die Nähe der vorgestellten Laufzeiten gekommen wäre – von dem (erheblichen) zusätzlichen Programmieraufwand ganz zu schweigen. Allerdings möchte ich zwei Verbesserungsmöglichkeiten nicht unerwähnt lassen:

Zum einen besitzt NTL keine eigene p-adische Arithmetik. Dies stellt natürlich zunächst einmal kein Problem dar, denn es ist bekannt wie man Elemente aus $\mathbb{Z}_p$ bzw. $\mathbb{Z}_q$ repräsentieren kann. Konkret kennt ein laufendes Programm den aktuellen *Modulus* („current modulus“), der *vor* der ersten Verwendung von Objekten der Klasse ZZ_p (d. h. ganzen Zahlen modulo p) initialisiert werden muss (in diesem Fall mit p^N). Dies kann man sich vereinfacht so vorstellen, dass ein Objekt der Klasse ZZ_p zwar eine ganze Zahl $\in [0, p^N - 1]$ repräsentiert, sich den Modulus aber nicht als Attribut „merkt“, sondern ihn zur Laufzeit aus einer statischen Variable erfährt (tatsächlich ist die Situation komplizierter). Elemente aus

$$\mathbb{Z}_{p^n} \cong \mathbb{Z}_p[X]/(F(X))$$

werden durch Polynome vom Grad $\leq n-1$ über $\mathbb{Z}/p^N\mathbb{Z}$ approximiert. Dazu verwende ich die Klassen ZZ_pX („Polynome mit Koeffizienten aus ZZ_p“) bzw. ZZ_pE. In der Klasse ZZ_pE ist die Reduktion modulo $F(X)$ bereits integriert. Intern werden Polynome einfach als Koeffizientenvektoren gespeichert.

Der von NTL verwendete Mechanismus ist leider recht unflexibel: Der Modulus kann zwar geändert werden, bereits erstellte Objekte darf man dann aber nicht mehr verwenden (selbst wenn es mathematisch sinnvoll wäre). Ich zitiere aus der Dokumentation:

> „Please note that for efficiency reasons, NTL does not make any attempt to ensure that variables declared under one modulus are not used under a different one. If that happens, the behavior of a program in this case is completely unpredictable.“
> `http://www.shoup.net/ntl/doc/tour-ex4.html`

> „Note, however, that if a ZZ_p object is created under one modulus and then used in any way (except destroyed) under another, program behavior is not predictable. This condition is not explicitly checked for, but an

> error is likely to be raised. One should also not presume that things will work properly if the modulus is changed, but its value happens to be the same [...]“
> `http://www.shoup.net/ntl/doc/ZZ_p.txt`

Warum ist das relevant? In allen vorgestellten Algorithmen genügt es, Zwischenschritte zu einer kleineren Genauigkeit durchzuführen. Aufgrund der geschilderten Problematik verzichte ich jedoch darauf und arbeite stets zur Präzision N. Objekte jedesmal zu sichern und nach einem Moduluswechsel wiederherzustellen ist recht teuer und lohnt sich nach meinen Tests nur bei der Berechnung von Divisionspolynomen. Generell gibt es weitere Möglichkeiten, auf die ich nicht eingegangen bin, da sie die Laufzeit nur unwesentlich verringern würden (es sei an das zumeist „ungünstige“ Verhältnis $t_{\text{j\_invariant}}/t$ erinnert). Für kleine Charakteristiken ist eine verbesserte p-adische Arithmetik jedoch durchaus wünschenswert.

Der zweite Punkt, den ich ansprechen möchte, betrifft die von `NTL` durchgeführte Reduktion modulo $F(X)$. Es stellt sich heraus, dass die Anzahl der „Nullkoeffizienten“ von $F(X)$ für die Geschwindigkeit der Reduktion bedeutungslos ist (auch unter Verwendung der Klasse `ZZ_pXModulus`), d. h. `NTL` verzichtet an dieser Stelle auf eine Optimierungsmöglichkeit: Eine einfache Implementierung von [CF06, Algorithm 11.31] („Division by a sparse polynomial“) war beispielsweise in Testläufen stets schneller als der von `NTL` verwendete Algorithmus. Normalerweise ist es effizienter, die Körpererweiterung $\mathbb{F}_p \subset \mathbb{F}_{p^n}$ bzw. $\mathbb{Z}_p \subset \mathbb{Z}_{p^n}$ mit einem „sparse polynomial“ zu definieren. Dies hat jedoch in der aktuellen Programmversion keine Auswirkungen.

Was für Konsequenzen wurden daraus gezogen? Zunächst einmal muss man Aufwand mit Nutzen vergleichen. Für den Lift der j-Invariante mit Algorithmus 5 auf Seite 57 ist es durchaus sinnvoll die Reduktion modulo $F(X)$ selbst zu verwalten (dies ist auch leicht zu realisieren). Allerdings gehe ich davon aus, dass eine Optimierung an dieser Stelle *keine* bedeutenden Auswirkungen auf die Gesamtlaufzeit hat (vgl. Tabelle 5 auf Seite 94). Warum dann keine Anpassungen im Bereich „lift kernel“? Für die Berechnung von $\psi_p^{\uparrow}(X) \bmod p^N$ und in der Implementierung von Algorithmus 6 auf Seite 79 benutze ich die Klasse `ZZ_pEX`, um eine Polynomarithmetik *über* der „ersten“ Polynomarithmetik zu realisieren. Eine Beeinflussung der Multiplikation von Koeffizienten „aus“ `ZZ_pE` mit Hilfe einer eigenen Reduktionsmethode ist dann aber nicht mehr praktikabel. Die beste Lösung wäre eine neue Polynomarithmetik über `ZZ_pX`, die aber nicht realisiert wurde.

Es wurde auch versucht direkt im Quellcode von `NTL` Veränderungen vorzunehmen (teilweise mit Erfolg). Allerdings kann man sich selbst davon überzeugen, dass dies nicht so trivial ist. Änderungen an einer Stelle können daher unvorhersehbare Auswirkungen in anderen Teilen des Programms haben. Aus Gründen der Stabilität und

Verlässlichkeit der Implementierung wurde auch dieser Ansatz nicht weiterverfolgt. Insbesondere bleibt damit das Programm ohne Einschränkungen verwendbar.

Die obigen Bemerkungen machen deutlich, dass es noch Optimierungsmöglichkeiten und damit Potenzial für schnellere Laufzeiten gibt – im Rahmen einer Diplomarbeit waren diese jedoch nicht zu verwirklichen.

A Algebraische Varietäten

Wir fassen im Folgenden [Sil86, I.] zusammen. Dem Leser ohne Hintergrund in algebraischer Geometrie soll damit der Einstieg in diese Arbeit erleichtert werden. Einen ähnlichen Überblick findet man in [CF06] und [Hus04]. Wir verweisen zudem auf [Har77].

Sei K ein Körper und $\overline{K}$ der *algebraische Abschluss* von K.

Definition A.1. Der *n-dimensionale affine Raum über K* ist definiert als

$$\mathbb{A}_n = \mathbb{A}_n(\overline{K}) \coloneqq \{(x_1, \ldots, x_n) : \ x_\nu \in \overline{K}\}.$$

Die *Menge der K-rationalen Punkte von $\mathbb{A}_n$* ist gegeben durch

$$\mathbb{A}_n(K) \coloneqq \{(x_1, \ldots, x_n) : \ x_\nu \in K\}.$$

Definition A.2. Sei $I \subset \overline{K}[X_1, \ldots, X_n]$ ein Ideal. Eine Menge der Gestalt

$$V = V_I \coloneqq \{P \in \mathbb{A}_n : \ f(P) = 0 \text{ für alle } f \in I\}$$

heißt (*affine*) *algebraische Menge*. Man definiert das *Verschwindungsideal von V* als

$$I(V) \coloneqq \{f \in \overline{K}[X_1, \ldots, X_n] : \ f(P) = 0 \text{ für alle } P \in V\}.$$

Mit $I(V/K)$ bezeichnet man das Ideal

$$I(V) \cap K[X_1, \ldots, X_n].$$

V heißt *definiert über K*, wenn $I(V)$ durch Polynome aus $K[X_1, \ldots, X_n]$ erzeugt werden kann (in Zeichen: V/K). Dies ist genau dann der Fall, wenn

$$I(V) = I(V/K)\overline{K}[X_1, \ldots, X_n]$$

(vgl. [Sil86, I., Remark 1.2]). Ist V über K definiert, so ist die *Menge der K-rationalen Punkte von V* gegeben durch

$$V(K) \coloneqq V \cap \mathbb{A}_n(K).$$

Beispiel A.3. Die algebraische Menge

$$V: \ X^n + Y^n = 1$$

ist über $\mathbb{Q}$ definiert. Nach der *Fermatschen Vermutung* gilt für $n \geq 3$, dass

$$V(\mathbb{Q}) = \begin{cases} \{(1,0),(0,1),(-1,0),(0,-1)\} & \text{falls } n \text{ gerade,} \\ \{(1,0),(0,1)\} & \text{falls } n \text{ ungerade.} \end{cases}$$

Definition A.4. Eine affine algebraische Menge V heißt (*affine*) *Varietät*, wenn $I(V)$ ein Primideal in $\overline{K}[X_1, \ldots, X_n]$ ist. Sei V/K eine Varietät. Der *affine Koordinatenring von* V/K ist definiert als

$$K[V] \coloneqq K[X_1, \ldots, X_n]/I(V/K).$$

$K[V]$ ist ein Integritätsbereich und man nennt $K(V) \coloneqq \mathrm{Quot}(K[V])$ *Funktionenkörper von* V/K.

Definition A.5. Sei V eine Varietät. Die *Dimension von* V (in Zeichen: $\dim(V)$) ist der Tranzendenzgrad[1] von $\overline{K}(V)$ über $\overline{K}$.

Beispielsweise hat $\mathbb{A}_n$ die Dimension n. Eine Varietät der Dimension 1 heißt (*algebraische*) *Kurve*.

Definition A.6. Sei V eine Varietät und $I(V)$ durch $f_1, \ldots, f_m \in \overline{K}[X_1, \ldots, X_n]$ erzeugt. V heißt *singularitätenfrei* oder *glatt in* $P \in V$, wenn

$$\mathrm{Rang}\left(\left(\frac{\partial f_i}{\partial X_j}(P)\right)_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}\right) = n - \dim(V).$$

Beispiel A.7. Sei V durch ein nicht konstantes Polynom $f \in \overline{K}[X_1, \ldots, X_n]$ gegeben:

$$V: \ f(X_1, \ldots, X_n) = 0.$$

Dann gilt[2] $\dim(V) = n - 1$. Folglich ist $P \in V$ genau dann ein singulärer Punkt, wenn

$$\frac{\partial f}{\partial X_1}(P) = \cdots = \frac{\partial f}{\partial X_n}(P) = 0.$$

[1] [Bos06, 7.1]
[2] [Sil86, Example 1.4]

Definition A.8. Der *n-dimensionale projektive Raum über K* ist definiert als

$$\mathbb{P}_n = \mathbb{P}_n(\overline{K}) \coloneqq (\mathbb{A}_{n+1} \setminus \{0\})/\sim$$

mit folgender Äquivalenzrelation:

$$(x_0, \ldots, x_n) \sim (y_0, \ldots, y_n) \ :\Longleftrightarrow\ \exists\ 0 \neq \lambda \in \overline{K} :\ y_\nu = \lambda x_\nu\ \forall\ \nu \in [0, n].$$

Die Äquivalenzklasse von $(x_0, \ldots, x_n) \in \mathbb{A}_{n+1} \setminus \{0\}$ wird mit $(x_0 : \ldots : x_n) \in \mathbb{P}_n$ bezeichnet. Die *Menge der K-rationalen Punkte von* $\mathbb{P}_n$ ist gegeben durch

$$\mathbb{P}_n(K) \coloneqq \{(x_0 : \ldots : x_n) \in \mathbb{P}_n :\ \exists\ 0 \neq \lambda \in \overline{K} :\ \lambda x_\nu \in K\ \forall\ \nu \in [0, n]\}.$$

Man beachte, dass aus $(x_0 : \ldots : x_n) \in \mathbb{P}_n(K)$ *nicht* folgt, dass $x_\nu \in K$. Allerdings gilt für jedes $x_i \neq 0$, dass $x_\nu/x_i \in K$ (für alle $\nu \in [0, n]$).

Definition A.9. Ein Polynom $f \in \overline{K}[X_0, \ldots, X_n]$ heißt *homogen vom Grad d*, wenn

$$f(\lambda X_0, \ldots, \lambda X_n) = \lambda^d f(X_0, \ldots, X_n) \quad \text{für alle } \lambda \in \overline{K}.$$

In diesem Fall lässt sich f schreiben als

$$f(X_0, \ldots, X_n) = \sum_{\nu_0 + \cdots + \nu_n = d} a_{\nu_0 \cdots \nu_n} X_0^{\nu_0} \cdots X_n^{\nu_n}.$$

Ein Ideal $I \subset \overline{K}[X_0, \ldots, X_n]$ heißt *homogen*, wenn es von homogenen Polynomen erzeugt wird.

Definition A.10. Sei $I \subset \overline{K}[X_0, \ldots, X_n]$ ein homogenes Ideal. Eine Menge der Gestalt

$$V = V_I \coloneqq \{P \in \mathbb{P}_n :\ f(P) = 0 \text{ für alle homogenen } f \in I\}$$

heißt (*projektive*) *algebraische Menge*. Das (*homogene*) *Verschwindungsideal von V*, das man mit $I(V)$ bezeichnet, ist das von

$$\{f \in \overline{K}[X_0, \ldots, X_n] :\ f \text{ ist homogen und } f(P) = 0 \text{ für alle } P \in V\}$$

erzeugte Ideal in $\overline{K}[X_0, \ldots, X_n]$. V heißt *definiert über K* (in Zeichen: V/K), wenn $I(V)$ durch homogene Polynome aus $K[X_0, \ldots, X_n]$ erzeugt werden kann. In diesem Fall ist die *Menge der K-rationalen Punkte von V* gegeben durch

$$V(K) \coloneqq V \cap \mathbb{P}_n(K).$$

Beispiel A.11. Eine *Hyperebene* ist durch eine Gleichung der Form

$$a_0 X_0 + a_1 X_1 + \cdots + a_n X_n = 0$$

gegeben, wobei $(a_0, \ldots, a_n) \in \mathbb{A}_{n+1} \setminus \{0\}$.

Definition A.12. Eine projektive algebraische Menge V heißt (*projektive*) *Varietät*, wenn $I(V)$ ein Primideal in $\overline{K}[X_0, \ldots, X_n]$ ist.

Wir stellen nun den Zusammenhang zwischen projektiven und affinen algebraischen Mengen her. Zunächst einmal gibt es viele Möglichkeiten, um $\mathbb{A}_n$ in $\mathbb{P}_n$ einzubetten, beispielsweise mit Hilfe der folgenden Inklusionen ($\nu \in [0, n]$):

$$\varphi_\nu : \ \mathbb{A}_n \to \mathbb{P}_n, \ (x_1, \ldots, x_n) \mapsto (x_1 : x_2 : \ldots : x_\nu : 1 : x_{\nu+1} : \ldots : x_n).$$

Sei U_ν das Komplement von $X_\nu = 0$, d. h.

$$U_\nu \coloneqq \{(x_0 : \ldots : x_n) \in \mathbb{P}_n : \ x_\nu \neq 0\}.$$

U_ν lässt sich bijektiv auf $\mathbb{A}_n$ abbilden:

$$\varphi_\nu^{-1} : \ U_\nu \to \mathbb{A}_n, \ (x_0 : \ldots : x_n) \mapsto \left(\frac{x_0}{x_\nu}, \ldots, \frac{x_{\nu-1}}{x_\nu}, \frac{x_{\nu+1}}{x_\nu}, \ldots, \frac{x_n}{x_\nu} \right).$$

Man überzeugt sich davon, dass diese Abbildungen wohldefiniert sind.

Sei V eine projektive algebraische Menge. Mit V_ν bezeichnen wir die affine algebraische Menge $\varphi_\nu^{-1}(V \cap U_\nu)$. Es gilt

$$I(V_\nu) = \{f(Y_1, \ldots, Y_\nu, 1, Y_{\nu+1}, \ldots, Y_n) : \ f(X_0, \ldots, X_n) \in I(V)\}.$$

Man beachte, dass

$$\mathbb{P}_n = \bigcup_{\nu=0}^{n} U_\nu \quad \text{und} \quad V = \bigcup_{\nu=0}^{n} (V \cap U_\nu) = \bigcup_{\nu=0}^{n} \varphi_\nu(V_\nu).$$

Die *Dehomogenisierung nach* X_ν

$$f(X_0, \ldots, X_n) \rightsquigarrow f(Y_1, \ldots, Y_\nu, 1, Y_{\nu+1}, \ldots, Y_n)$$

lässt sich umkehren: Sei $f \in \overline{K}[Y_1, \ldots, Y_n]$. Wir definieren

$$f_\nu(X_0, \ldots, X_n) \coloneqq X_\nu^{\deg(f)} \cdot f \left(\frac{X_0}{X_\nu}, \ldots, \frac{X_{\nu-1}}{X_\nu}, \frac{X_{\nu+1}}{X_\nu}, \ldots, \frac{X_n}{X_\nu} \right).$$

Man nennt $f_\nu \in \overline{K}[X_0, \ldots, X_n]$ die *Homogenisierung von f nach X_ν*.

Beispiel A.13. Sei $\overline{K}[Y_1, Y_2] \ni f(Y_1, Y_2) \coloneqq Y_2^2 - Y_1^3 - aY_1 - b$. Dann ist die Homogenisierung von f nach X_2 gegeben durch

$$\begin{aligned} f_2(X_0, X_1, X_2) &= X_2^3 \cdot f \left(\frac{X_0}{X_2}, \frac{X_1}{X_2} \right) \\ &= X_1^2 X_2 - X_0^3 - aX_0 X_2^2 - bX_2^3. \end{aligned}$$

Definition A.14. Sei $V = V_I$ eine affine algebraische Menge. Der *projektive Abschluss von* V (bzgl. ν) ist definiert als

$$\mathbb{P}_n \supset \overline{V} \coloneqq V_{\langle\{f_\nu^{\text{h}}:\ f\in I\}\rangle}.$$

Man bezeichnet $\overline{V} \setminus \varphi_\nu(V)$ als *Punkte im Unendlichen* oder *unendlich-ferne Punkte.*

Proposition A.15. *a) Sei V eine affine Varietät. Dann ist $\overline{V}$ eine projektive Varietät und es gilt*

$$\varphi_\nu(V) = \overline{V} \cap U_\nu.$$

b) Sei V eine projektive Varietät. Dann ist $\varphi_\nu^{-1}(V \cap U_\nu)$ eine affine Varietät und es gilt entweder

$$\varphi_\nu^{-1}(V \cap U_\nu) = \emptyset \quad \textit{oder} \quad V = \overline{\varphi_\nu^{-1}(V \cap U_\nu)}.$$

c) Sei V/K eine affine (projektive) Varietät. Dann ist $\overline{V}$ ($\varphi_\nu^{-1}(V \cap U_\nu)$) über K definiert.

Beweis. [Sil86, I., Proposition 2.6]. □

Es existiert stets ein $\nu \in [0, n]$, so dass $V \cap U_\nu \neq \emptyset$ (vgl. [CF06, 4.1.1.d]).

Definition A.16. Sei V/K eine projektive Varietät und $\varphi_\nu^{-1}(V \cap U_\nu) \neq \emptyset$. Die *Dimension von* V ist definiert als

$$\dim(V) \coloneqq \dim(\varphi_\nu^{-1}(V \cap U_\nu)).$$

Definition A.17. Sei V eine projektive Varietät und $P \in V \cap U_\nu \neq \emptyset$. V heißt *singularitätenfrei* oder *glatt in* P, wenn $\varphi_\nu^{-1}(V \cap U_\nu)$ glatt in $\varphi_\nu^{-1}(P)$ ist.

Definition A.18. Seien V_1 und V_2 projektive Varietäten. Eine *rationale Abbildung* $\phi :\ V_1 \to V_2$ ist eine Abbildung der Form

$$\phi = (\phi_0(X_0, \ldots, X_n) : \ldots : \phi_n(X_0, \ldots, X_n)),$$

mit homogenen Polynomen $\phi_\nu \in \overline{K}[X_0, \ldots, X_n]$ und folgenden Eigenschaften:

(i) $\deg(\phi_i) = \deg(\phi_j)$ für alle $i, j \in [0, n]$.

(ii) Es existiert ein $\nu \in [0, n]$, so dass $\phi_\nu \notin I(V_1)$.

(iii) Für alle $f \in I(V_2)$ gilt $f(\phi) \in I(V_1)$.

Definition A.19. Eine rationale Abbildung $\phi :\ V_1 \to V_2$ heißt *regulär in* $P \in V_1$, wenn homogene Polynome $\psi_0, \ldots, \psi_n \in \overline{K}[X_0, \ldots, X_n]$ existieren, so dass

(i) $\deg(\psi_i) = \deg(\psi_j)\ \forall\ i,j \in [0,n]$,

(ii) $\phi_i\psi_j \equiv \phi_j\psi_i \pmod{I(V_1)}\ \forall\ i,j \in [0,n]$ und

(iii) $\exists\ \nu \in [0,n] :\ \psi_\nu(P) \neq 0$.

In diesem Fall definieren wir $\phi(P) \coloneqq (\psi_0(P) : \ldots : \psi_n(P))$. Ein *Morphismus* ist eine rationale Abbildung, die in jedem Punkt regulär ist.

Definition A.20. Seien V_1 und V_2 projektive Varietäten. V_1 und V_2 heißen *isomorph*, wenn es Morphismen $\phi :\ V_1 \to V_2$ und $\psi :\ V_2 \to V_1$ gibt, so dass

$$\psi \circ \phi = \mathrm{id}_{V_1} \quad \text{und} \quad \phi \circ \psi = \mathrm{id}_{V_2}\,.$$

V_1/K und V_2/K heißen *isomorph über* K, wenn ϕ und ψ über K definiert werden können.

Proposition A.21. *Sei C eine Kurve (d. h. eine projektive Varietät der Dimension 1), $V \subset \mathbb{P}_n$ eine Varietät, $P \in C$ ein glatter Punkt und $\phi :\ C \to V$ eine rationale Abbildung. Dann ist ϕ regulär in P. Insbesondere ist ϕ ein Morphismus, wenn C glatt ist.*

Beweis. [Sil86, II., Proposition 2.1]. □

Satz A.22. *Seien C_1 und C_2 Kurven und $\phi :\ C_1 \to C_2$ ein Morphismus. Dann ist ϕ entweder konstant oder surjektiv.*

Beweis. [Sil86, II., Theorem 2.3]. □

B DVD

Auf der beiliegenden DVD befinden sich folgende Dateien:

divisionpolynomials.tar.bz2 (21 907 236 Bytes) Vorberechnungsdaten um p-te Divisionspolynome in $\mathbb{F}_{p^n}$ und $\mathbb{Z}_{p^n}$ nach McKee[1] zu berechnen ($p \in [5, 293]$). `satoh` und `search` erzeugen diese Daten (falls nötig) automatisch.

gmp-4.2.4.tar.bz2 (1 710 660 Bytes) Aktuelle Version der Bibliothek GMP[2].

irreduciblepolynomials.tar.bz2 (22 638 Bytes) Irreduzible dünn besetzte Polynome über $\mathbb{F}_p$ ($p \in [5, 251]$) um Körpererweiterungen $\mathbb{F}_p \subset \mathbb{F}_{p^n}$ und $\mathbb{Q}_p \subset \mathbb{Q}_{p^n}$ ($p^n \leq 2^{160}$) zu definieren.

modularpolynomials_cpp.tar.bz2 (35 564 715 Bytes) Modulo p^N reduzierte Koeffizienten des p-ten modularen Polynoms $\Phi_p(X, Y)$ ($p \in [5, 293]$).

modularpolynomials_dec.tar.bz2 (1 053 736 181 Bytes) Koeffizienten des p-ten modularen Polynoms $\Phi_p(X, Y)$ ($p \in [2, 293]$). Diese Daten wurden von Rubinstein[3] veröffentlicht.

niske.pdf (1 590 874 Bytes) Diplomarbeit.

ntl-5.4.2.tar.gz (692 630 Bytes) Aktuelle Version von NTL[4].

readme.txt (9 235 Bytes) Englischsprachige Hilfedatei mit Kompilieranleitung und Hinweisen zur Programmbenutzung.

satoh-2.1.tar.bz2 (1 125 211 Bytes) Quelltext und mit MinGW[5] erstellte Programme für Windows. Außerdem eine Vielzahl von gefundenen starken elliptischen Kurven $E/\mathbb{F}_{p^n}$ (für alle $p \in [5, 293]$).

---

[1] Lemma 1.25
[2] [Gra]
[3] [Rub]
[4] [Sho]
[5] [Min]

C Auszug aus dem Quelltext

C.1 p_adic.h

```
1
2  /**
3   * @file p_adic.h
4   * @author Alexander Niske
5   */
6
7  #ifndef P_ADIC_H
8  #define P_ADIC_H
9
10 #include"psi.h"
11
12 /**
13  * @brief f = pi^-1(pi(a)^e).
14  */
15 void power_field(ZZ_pX& f, const ZZ_pX& a, long e);
16
17 /**
18  * @brief f = pi^-1(a^e).
19  */
20 void power_field(ZZ_pX& f, const zz_pE& a, const ZZ& e);
21
22 /**
23  * @brief f = pi^-1(a^e).
24  */
25 void power_field(ZZ_pX& f, const ZZX& a, long e);
26
27 /**
28  * @brief z = a^-1 mod p^N, start = a^-1 mod p.
29  */
30 void inverse(ZZ_pX& z, const ZZ_pX& a, long N, const ZZ_pXModulus& f, const
      ZZ_pX& start);
31
32 /**
33  * @brief Q = J/(1728-J) mod p^N.
34  */
```

```
35 void compute_Q(ZZ_pX& Q, const ZZ_pX& J, long N, const ZZ_pXModulus& f);
36
37 /**
38  * @brief Implementation of algorithm 5.
39  */
40 void lift_j_invariant(ZZ_pX& J, const zz_pE& j, const vec_ZZ_pX& phi, long p,
       long N, const ZZ_pXModulus& f);
41
42 /**
43  * @brief Implementation of algorithm 6.
44  *
45  * A = 3Q, B = 2Q.
46  */
47 void lift_kernel(ZZ_pEX& H, const zz_pEX& psi_tilde, long p, long N, const
      ZZ_pX& Q, const ZZ_pE& A, const ZZ_pE& B);
48
49 /**
50  * @brief Implementation of algorithm 4.
51  *
52  * b^2 = d mod p^N, e^2 = d mod p.
53  */
54 void sqrt(ZZ& b, const ZZ& d, const ZZ& e, long p, long N);
55
56 /**
57  * @brief res = norm(z) mod m.
58  */
59 void norm(ZZ& res, const ZZ_pX& z, const ZZ& m);
60
61 #endif
```

C.2 p_adic.cpp

```
1
2 /**
3  * @file p_adic.cpp
4  * @author Alexander Niske
5  */
6
7 #include"p_adic.h"
8
9 void power_field(ZZ_pX& f, const ZZ_pX& a, long e)
10 {
11     static ZZX aux1;
```

```
12      static zz_pX aux2;
13      conv(aux1, a);
14      conv(aux2, aux1);
15      PowerMod(aux2, aux2, e, zz_pE::modulus());
16      conv(aux1, aux2);
17      conv(f, aux1);
18  }
19
20  void power_field(ZZ_pX& f, const zz_pE& a, const ZZ& e)
21  {
22      static ZZX aux1;
23      static zz_pE aux2;
24      power(aux2, a, e);
25      conv(aux1, rep(aux2));
26      conv(f, aux1);
27  }
28
29  void power_field(ZZ_pX& f, const ZZX& a, long e)
30  {
31      static ZZX aux1;
32      static zz_pX aux2;
33      conv(aux2, a);
34      PowerMod(aux2, aux2, e, zz_pE::modulus());
35      conv(aux1, aux2);
36      conv(f, aux1);
37  }
38
39  void inverse(ZZ_pX& z, const ZZ_pX& a, long N, const ZZ_pXModulus& f, const
        ZZ_pX& start)
40  {
41      z = start;
42      static ZZ_pX aux;
43      static ZZ_pXMultiplier m;
44      build(m, a, f);
45
46      while(N!=1)
47      {
48          MulMod(aux, z, m, f);
49          sub(aux, 2, aux);
50          MulMod(z, z, aux, f); // z = z(2-za) mod f
51          N = (long)ceil(N/2.);
52      }
53  }
54
55  void compute_Q(ZZ_pX& Q, const ZZ_pX& J, long N, const ZZ_pXModulus& f)
56  {
```

```
57      static ZZ_pX start, aux;
58
59      sub(aux, 1728, J);
60      power_field(start, aux, -1);
61      inverse(Q, aux, N, f, start);
62      MulMod(Q, Q, J, f); // Q =  J/(1728-J)
63  }
64
65  /**
66   * @brief (a, b) = (g(x), g'(x)).
67   */
68  void horner(ZZ_pX& a, ZZ_pX& b, const vec_ZZ_pX& g, const ZZ_pX& x, const
        ZZ_pXModulus& f)
69  {
70      clear(a);
71      clear(b);
72      static ZZ_pXMultiplier m;
73      build(m, x, f);
74
75      static long i;
76      for(i=g.length()-1; i>0; i--)
77      {
78          MulMod(a, a, m, f);
79          add(a, a, g[i]);
80          MulMod(b, b, m, f);
81          add(b, b, a);
82      }
83
84      MulMod(a, a, m, f);
85      add(a, a, g[0]);
86  }
87
88  void lift_previous_j_invariant(ZZ_pX& I, const ZZ_pX& J, const vec_ZZ_pX& phi
        , long p, long N, const ZZ_pXModulus& f)
89  {
90      static vec_ZZ_pX h = vec_ZZ_pX(INIT_SIZE, p+2);
91      static ZZ_pXArgument arg;
92
93      build(arg, J, f, p+1);
94      // computes and stores J, J^2, ..., J^(p+1)
95
96      static long i;
97      for(i=0; i<p+2; i++) CompMod(h[i], phi[i], arg, f);
98
99      static stack<long> st;
100     while(N!=1)
```

```
101     {
102         st.push(N);
103         N = (long)ceil(N/2.);
104     }
105
106     static ZZ_pX start, z, a, b;
107
108     clear(start); // important
109
110     power_field(I, J, p); // I = pi^-1(pi(J)^p)
111
112     while(!st.empty())
113     {
114         horner(a, b, h, I, f);
115         // (a, b) = (h(I), h'(I))
116
117         if(IsZero(start)) power_field(start, b, -1);
118         // pi(b) is constant
119
120         inverse(z, b, (long)ceil(st.top()/2.), f, start);
121
122         MulMod(a, a, z, f);
123         sub(I, I, a); // I = I - h(I)/h'(I)
124         st.pop();
125     }
126 }
127
128 void lift_j_invariant(ZZ_pX& J, const zz_pE& j, const vec_ZZ_pX& phi, long p,
        long N, const ZZ_pXModulus& f)
129 {
130     static double zeit;
131     cout << "lift j-invariant..." << flush;
132     zeit = GetTime();
133     static long e, i;
134     e = 1-N;
135
136     while(e<0) e += deg(f);
137     power_field(J, j, power_ZZ(p, e));
138     // J = pi^-1( j^(p^( (1-N) mod deg(f) )) )
139
140     progress(N);
141
142     static ZZ_pX aux;
143     for(i=2; i<=N; i++)
144     {
145         lift_previous_j_invariant(aux, J, phi, p, i, f);
```

```
146         swap(J, aux);
147         progress();
148     }
149
150     done(GetTime()-zeit);
151 }
152
153 void div(ZZ_pEX& source, long p)
154 {
155     static int i, k;
156     k = deg(source) +1;
157     static ZZX aux;
158
159     for(i=0; i<k; i++)
160     {
161         conv(aux, rep(source.rep[i]));
162         div(aux, aux, p);
163         conv(source.rep[i], to_ZZ_pX(aux));
164     }
165
166     source.normalize();
167 }
168
169 void inverse(ZZ_pEX& A, const ZZ_pEX& F, const ZZ_pEXModulus& G, long N,
        const ZZ_pEX& start)
170 {
171     A = start;
172
173     static ZZ_pEX aux;
174     while(N!=1)
175     {
176         MulMod(aux, A, F, G);
177         sub(aux, 2, aux);
178         MulMod(A, aux, A, G); // A = A(2-A*F) mod G
179         N = (long)ceil(N/2.);
180     }
181 }
182
183 void lift_kernel_aux(ZZ_pEX& H, const ZZ_pEX& PSI, const zz_pEX& h, long p,
        long N)
184 {
185     static stack<long> st;
186
187     N = N-1;
188
189     while(N!=1)
```

```
190     {
191         st.push(N);
192         N = (long)ceil(N/2.);
193     }
194 
195     conv(H, h);
196 
197     static ZZ_pEXModulus F;
198     build(F, H);
199 
200     static ZZ_pEX i, j, aux1, aux2, start, F2;
201 
202     diff(i, PSI);
203     div(i, p); // i = 1/p * PSI'
204 
205     static zz_pEX f, g, a, b, d;
206 
207     conv(f, i);
208     conv(g, H);
209     rem(f, f, g);
210     XGCD(d, a, b, f, g); // a*f + b*g = gcd(f, g) = d (should be 1)
211 
212     conv(start, a);
213 
214     static double zeit;
215     while(true)
216     {
217         cout << "reduce division polynomial (and its derivative)..." << flush
                ;
218         zeit = GetTime();
219 
220         sqr(F2, F);
221         rem(aux2, PSI, F2); // aux2 = PSI mod F^2
222         diff(aux1, aux2);
223         rem(j, aux1, F);
224         div(j, p); // j = (1/p * PSI') mod F
225         rem(aux1, aux2, F); // aux1 = PSI mod F
226 
227         done(GetTime()-zeit);
228 
229         inverse(aux2, j, F, (long)ceil(st.top()/2.), start);
230         div(aux1, p);
231         MulMod(aux1, aux1, aux2, F);
232         diff(aux2, H);
233         MulMod(aux1, aux1, aux2, F);
234         add(H, H, aux1);
```

```
235
236         st.pop();
237
238         if(st.empty()) break;
239         else build(F, H);
240     }
241 }
242
243 void lift_kernel_aux2(ZZ_pEX& H, const zz_pEX& h, long p, long N, const ZZ_pE
        & A, const ZZ_pE& B)
244 {
245     static stack<long> st;
246
247     N = N-1;
248
249     while(N!=1)
250     {
251         st.push(N);
252         N = (long)ceil(N/2.);
253     }
254
255     conv(H, h);
256     static ZZ_pEXModulus F;
257     build(F, H);
258
259     static ZZ_pEX PSI, dPSIdx, aux1, aux2, start;
260
261     compute_psi_rec(PSI, dPSIdx, p, st.top()+1, A, B, F);
262     div(dPSIdx, p);
263
264     static zz_pEX f, g, a, b, d;
265
266     conv(f, dPSIdx);
267     conv(g, H);
268
269     rem(f, f, g);
270
271     XGCD(d, a, b, f, g); // a*f + b*g = gcd(f, g) = d (should be 1)
272
273     conv(start, a);
274
275     while(true)
276     {
277         inverse(aux2, dPSIdx, F, (long)ceil(st.top()/2.), start);
278         div(PSI, p);
279         MulMod(aux1, PSI, aux2, F);
```

```
280         diff(aux2, H);
281         MulMod(aux1, aux1, aux2, F);
282         add(H, H, aux1);
283 
284         st.pop();
285 
286         if(st.empty()) break;
287         else
288         {
289             build(F, H);
290             compute_psi_rec(PSI, dPSIdx, p, st.top()+1, A, B, F);
291             div(dPSIdx, p);
292         }
293     }
294 }
295 
296 void lift_kernel(ZZ_pEX& H, const zz_pEX& psi, long p, long N, const ZZ_pX& Q
        , const ZZ_pE& A, const ZZ_pE& B)
297 {
298     static double zeit;
299     zeit = GetTime();
300     cout << "lift kernel..." << endl;
301 
302     static long k, l;
303     l = (p-1)/2;
304 
305     static zz_pE aux;
306     inv(aux, coeff(psi, l*p));
307 
308     static zz_pEX h;
309     h.rep.SetLength(l+1);
310 
311     for(k=0; k<=l; k++) mul(h.rep[k], coeff(psi, k*p), aux);
312 
313     h.normalize();
314 
315     static ZZ_pEX PSI;
316     if(p<=P) // tools.h
317     {
318         load_psi(PSI, p, N, Q, ZZ_pE::modulus());
319         lift_kernel_aux(H, PSI, h, p, N);
320     }
321     else lift_kernel_aux2(H, h, p, N, A, B);
322 
323     cout << "done";
324     done(GetTime()-zeit);
```

```
325 }
326
327 void sqrt(ZZ& b, const ZZ& d, const ZZ& e, long p, long N)
328 {
329     static ZZ c, c_, d_, q, aux;
330     static stack<long> st;
331
332     conv(q, p);
333
334     static long N_ = 0;
335     if(N_ < N)
336     {
337         c = InvMod(to_ZZ(2), power_ZZ(p, (long)(ceil(N/2.))));
338         N_ = N;
339     }
340
341     while(N!=1)
342     {
343         st.push(N);
344         N = (long)ceil(N/2.);
345     }
346
347     InvMod(b, e, q); // b = e^-1 mod p
348
349     while(!st.empty())
350     {
351         power(q, p, st.top());
352         rem(c_, c, q);
353         rem(d_, d, q);
354
355         SqrMod(aux, b, q);
356         MulMod(aux, d_, aux, q);
357         sub(aux, 1, aux);
358         rem(aux, aux, q);
359         MulMod(aux, c_, aux, q);
360         add(aux, 1, aux);
361         rem(aux, aux, q);
362         MulMod(b, b, aux, q);
363
364         st.pop();
365     }
366     MulMod(b, d_, b, q);
367 }
368
369 void norm(ZZ& res, const ZZ_pX& z, const ZZ& mod)
370 {
```

```
371     static double zeit;
372     cout << "compute norm..." << flush;
373     zeit = GetTime();
374
375     static ZZX F = to_ZZX(zz_pE::modulus());
376     static ZZX Z;
377
378     conv(Z, z);
379     resultant(res, F, Z);
380     rem(res, res, mod);
381
382     done(GetTime()-zeit);
383 }
```

C.3 satoh.h

```
1
2  /**
3   * @file satoh.h
4   * @author Alexander Niske
5   */
6
7  #ifndef SATOH_H
8  #define SATOH_H
9
10 #include"p_adic.h"
11 #include"phi.h"
12 #include"ec.h"
13
14 /**
15  * @brief Initializes GF(p^n) = GF(p)[X]/(f).
16  */
17 void init(long p, long n, ZZX& f);
18
19 /**
20  * @brief Implementation of algorithm 7.
21  *
22  * k = #E(GF(p^n)) (E: y^2 = x^3 + ax + b).
23  *
24  * Searches non-trivial l-torsion points for all primes l <= limit.
25  * The computation stops (and k is set to 0) when a torsion point
26  * is found. The computed order k will never be checked when limit > 1.
27  */
```

```
28 void satoh(ZZ& k, const zz_pE& a, const zz_pE& b, long limit=1);
29
30 /**
31  * @brief Searches an elliptic curve with prime order.
32  * Uses "satoh" with the parameter "limit".
33  */
34 void search(long limit=1);
35
36 #endif
```

C.4 satoh.cpp

```
1
2  /**
3   * @file satoh.cpp
4   * @author Alexander Niske
5   */
6
7  #include"satoh.h"
8
9  void load_irred(long p, long n, ZZX& g)
10 {
11     fstream f;
12     string s = IRRED_INPUT_PATH + to_string<long>(p, dec) + "_" + to_string<
           long>(n, dec) + IRRED_INPUT_FILE_END;
13
14     f.open(s.c_str(), ios::in);
15
16     if(!f.good())
17     {
18         cout << "Build irreducible polynomial." << endl;
19         conv(g, BuildIrred_zz_pX(n));
20     }
21     else
22     {
23         f >> g;
24
25         f.close();
26
27         if((deg(g)!=n)||!DetIrredTest(to_zz_pX(g))) // better to be safe
28         {
29             conv(g, BuildIrred_zz_pX(n));
30             cout << s << " contains invalid data. Build new irreducible
                   polynomial." << endl;
```

```
31          }
32          else cout << s << " loaded" << endl;
33      }
34  }
35
36  void init(long p, long n, ZZX& f)
37  {
38      zz_p::init(p);
39      ZZ_p::init(power_ZZ(p, satoh_precision(p, n)));
40      zz_pX g = to_zz_pX(f);
41
42      if((deg(g)!=n)||!DetIrredTest(g))
43      {
44          cout << "invalid" << endl;
45          load_irred(p, n, f);
46          g = to_zz_pX(f);
47      }
48
49      zz_pE::init(g);
50      ZZ_pE::init(to_ZZ_pX(to_ZZX(g))); // better to be safe
51  }
52
53  long compute_factor(const zz_pE& a, const zz_pE& b, long limit)
54  {
55      static string s = "search small prime factor of #E(GF(" + to_string(zz_p
            ::modulus(), dec) + "^" + to_string(zz_pE::degree(), dec) + "))";
56      cout << s << flush;
57
58      static long factor;
59      factor = 1;
60
61      static double zeit;
62      zeit = GetTime();
63
64      cout.width(3);
65      if(!two_good(a, b))
66      {
67          factor = 2;
68          cout << 2 << flush;
69      }
70      else cout << "." << flush;
71
72      static PrimeSeq seq;
73      static long l;
74
75      if(factor<2)
```

```
76      {
77          seq.reset(3);
78          l = seq.next();
79
80          while(l<=limit)
81          {
82              cout.width(3);
83              if(!l_good(l, a, b))
84              {
85                  factor *= l;
86                  cout << l << flush;
87                  break;
88              }
89              else cout << "." << flush;
90              l = seq.next();
91          }
92      }
93
94      cout << " ";
95      done(GetTime()-zeit);
96
97      return factor;
98  }
99
100 void compute_trace_j_gfp(ZZ& t, const zz_p& j, const zz_pE& a, const zz_pE& b
        )
101 {
102     static long p = zz_p::modulus();
103     static long n = zz_pE::degree();
104     static ZZ q = zz_pE::cardinality();
105     static ZZ ZZp = to_ZZ(p);
106     static long x, i, sum, a2, b2;
107     static zz_pE mu_2, mu_4, mu_6;
108     static bool b_inGFp, a_inGFp;
109
110     b_inGFp = (deg(rep(b)) == 0);
111     a_inGFp = (deg(rep(a)) == 0);
112
113     if(IsZero(j)) // => a = 0
114     {
115         b2 = rep(ConstTerm(rep(b)));
116
117         if(!b_inGFp)
118         {
119             for(b2=1; b2<p; b2++)
120             {
```

```
121                 mu_6 = b / b2;
122                 if(IsCube(mu_6)) break;
123                 if(b2==p-1)
124                 {
125                     cout << endl;
126                     cout << "Could not build E2/GF(" << p << "): y^2 = x^3 +
                        b2 with the following properties:" << endl;
127                     cout << " - j(E2) = j (= 0)" << endl;
128                     cout << " - b / b2 is a cube in GF(" << p << "^" << n <<
                        ")" << endl;
129                     cout << endl;
130
131                     brute_force_trace(t, a, b);
132                     return;
133                 }
134             }
135         }
136
137         for(i=0, sum=0; i<p; i++)
138         {
139             x = PowerMod(i, 3, p);
140             x = AddMod(x, b2, p);
141             sum += Jacobi(to_ZZ(x), ZZp);
142         }
143
144         trace_recursion(t, to_ZZ(-sum), ZZp, n);
145
146         if((!b_inGFp) && (!IsSquare(mu_6))) t = -t; // E2 is a quadratic
                twist of E
147         return;
148     }
149
150     if(j==to_zz_p(1728)) // => b = 0
151     {
152         a2 = rep(ConstTerm(rep(a)));
153
154         if(!a_inGFp)
155         {
156             for(a2=1; a2<p; a2++)
157             {
158                 mu_4 = a / a2;
159                 if(IsSquare(mu_4)) break;
160                 if(a2==p-1)
161                 {
162                     cout << endl;
```

```
163                     cout << "Could not build E2/GF(" << p << "): y^2 = x^3 +
                            a2x with the following properties:" << endl;
164                     cout << " - j(E2) = j (= 1728)" << endl;
165                     cout << " - a / a2 is a square in GF(" << p << "^" << n
                            << ")" << endl;
166                     cout << endl;
167
168                     brute_force_trace(t, a, b);
169                     return;
170                 }
171             }
172         }
173
174         for(i=0, sum=0; i<p; i++)
175         {
176             x = PowerMod(i, 3, p);
177             x = AddMod(x, MulMod(a2, i, p), p);
178             sum += Jacobi(to_ZZ(x), ZZp);
179         }
180
181         trace_recursion(t, to_ZZ(-sum), ZZp, n);
182
183         if(!a_inGFp)
184         {
185             sqrt(mu_2, mu_4);
186             if(!IsSquare(mu_2)) t = -t; // E2 is a quadratic twist of E
187         }
188         return;
189     }
190
191     // now j is different from 0 and 1728
192
193     a2 = rep( (3*j) / (1728 - j) );
194     b2 = rep( (2*j) / (1728 - j) );
195
196     // E2/GF(p): y^2 = x^3 + a2x + b2 is isomorph to E
197
198     for(i=0, sum=0; i<p; i++)
199     {
200         x = PowerMod(i, 3, p);
201         x = AddMod(x, MulMod(a2, i, p), p);
202         x = AddMod(x, b2, p);
203         sum += Jacobi(to_ZZ(x), ZZp);
204     }
205
206     trace_recursion(t, to_ZZ(-sum), ZZp, n);
```

```
207
208     if(!IsSquare((2*a)/(3*b))) t = -t; // E2 is a quadratic twist of E
209 }
210
211 void compute_trace_j_gfp2(ZZ& t, const zz_pE& j, const zz_pE& a, const zz_pE&
        b)
212 {
213     static long p = zz_p::modulus();
214     static long n = zz_pE::degree();
215     static ZZ q = zz_pE::cardinality();
216     static ZZ ZZp2 = power_ZZ(p, 2);
217     static zz_pXModulus g = zz_pXModulus(BuildIrred_zz_pX(2));
218     static zz_pX k, a1, b1, x, aux;
219     static long e0, e1, sum;
220
221     send_j(k, rep(j), g);
222     InvMod(aux, 1728-k, g);
223     MulMod(a1, 3*k, aux, g);
224     MulMod(b1, 2*k, aux, g);
225
226     clear(aux);
227     clear(x);
228
229     for(e0=0, sum=0; e0<p; e0++)
230     {
231         for(e1=0; e1<p; e1++)
232         {
233             SetCoeff(aux, 1, e1);
234             SetCoeff(aux, 0, e0);
235
236             PowerMod(x, aux, 3, g);
237             add(x, x, MulMod(a1, aux, g));
238             add(x, x, b1);
239
240             sum += jacobi(x, g);
241         }
242     }
243
244     trace_recursion(t, to_ZZ(-sum), ZZp2, n/2);
245
246     if(!IsSquare((2*a)/(3*b))) t = -t; // E1 is a quadratic twist of E
247 }
248
249 long compute_trace(ZZ& t, const zz_pE& a, const zz_pE& b, long limit)
250 {
251     static double zeit;
```

```
252     static bool first = true; // first call of this method?
253     static long p = zz_p::modulus();
254     static long n = zz_pE::degree();
255     static long N = satoh_precision(p, n);
256     static ZZ jpower = zz_pE::cardinality() / p;
257     static ZZ_pXModulus f  = ZZ_pE::modulus();
258     static zz_pXModulus f_ = zz_pE::modulus();
259     static zz_pE j;
260
261     j_invariant(j, a, b);
262
263     cout << "f=" << f << endl;
264     cout << "a=" << a << endl;
265     cout << "b=" << b << endl;
266     cout << "j=" << j << endl;
267
268     if(deg(rep(j))<=0) // j in GF(p)
269     {
270         cout << "j in GF(" << p << ")" << endl;
271         compute_trace_j_gfp(t, ConstTerm(rep(j)), a, b);
272         cout << "t=" << t << endl;
273         return 1;
274     }
275
276     if(j==power(j, p*p)) // j in GF(p^2) \ GF(p)
277     {
278         cout << "j in GF(" << p << "^2)" << endl;
279         compute_trace_j_gfp2(t, j, a, b);
280         cout << "t=" << t << endl;
281         return 1;
282     }
283
284     if(limit>1 && compute_factor(a, b, limit)>1) return 0;
285
286     zeit = GetTime();
287
288     static vec_ZZ_pX phi;
289
290     if(first)
291     {
292         load_phi(phi, p, N);
293         first = false;
294     }
295
296     static ZZ_pX J, Q;
297
```

```
298     lift_j_invariant(J, j, phi, p, N, f);
299
300     static long k = (p-1)/2;
301     static ZZ_pE A, B, alpha, beta, aux, tmp;
302     static zz_pE q, a_, b_;
303     static ZZ_pEX H;
304     static zz_pEX psi;
305
306     power(j, j, jpower); // j = j^(p^(n-1))
307     sub(q, 1728, j);
308     inv(q, q);
309     mul(q, q, j); // q = j / (1728-j)
310
311     load_psi(psi, p, rep(q), f_);
312
313     compute_Q(Q, J, N, f);
314     conv(A, Q);
315     add(B, A, A); // B = 2Q
316     add(A, A, B); // A = 3Q
317
318     lift_kernel(H, psi, p, N, Q, A, B);
319
320     mul(alpha, A, 6 - 5*p);
321     sqr(aux, H.rep[k-1]);
322     sub(aux, aux, H.rep[k-2]);
323     sub(aux, aux, H.rep[k-2]);
324     mul(aux, aux, 30);
325     sub(alpha, alpha, aux);
326
327     mul(beta, B, 15 - 14*p);
328     mul(aux, A, H.rep[k-1]);
329     mul(aux, aux, 42);
330     add(beta, beta, aux);
331
332     power(tmp, H.rep[k-1], 3);
333     mul(aux, H.rep[k-1], H.rep[k-2]);
334     mul(aux, aux, 3);
335     sub(aux, aux, tmp);
336     mul(tmp, H.rep[k-3], 3);
337     sub(aux, aux, tmp);
338     mul(aux, aux, 70);
339     sub(beta, beta, aux);
340
341     mul(alpha, alpha, 2);
342     mul(beta, beta, 3);
343
```

```
344     static ZZ_pX start, z;
345
346     power_field(start, rep(alpha), -1);
347     inverse(z, rep(alpha), N, f, start);
348     MulMod(z, rep(beta), z, f);
349
350     static zz_pEX g = zz_pEX(3, 1);
351     static zz_pEX h;
352     SetCoeff(g, 1, a);
353     SetCoeff(g, 0, b);
354     power(h, g, k);
355
356     static ZZ res;
357     static ZZ m = ZZ_p::modulus() / p;
358     static ZZ c = 4 * zz_pE::cardinality();
359
360     norm(res, z, m);
361     sqrt(t, res, to_ZZ(rep(norm(h.rep[p-1]))), p, N-1);
362
363     if(sqr(t) > c) sub(t, t, m);
364
365     cout << "trace computed (total time: " << flush;
366         PrintTime2(GetTime()-zeit);
367     cout << ")" << endl << "t=" << t << endl;
368
369     return 1;
370 }
371
372 void satoh(ZZ& k, const zz_pE& a, const zz_pE& b, long limit)
373 {
374     static ZZ t;
375     static long ok, probprime;
376
377     ok = compute_trace(t, a, b, limit);
378
379     add(k, zz_pE::cardinality(), 1);
380     sub(k, k, t);
381
382     probprime = ProbPrime(k);
383
384     if(!ok) clear(k);
385     else
386     {
387         cout << "#E(GF(" << zz_p::modulus() << "^" << zz_pE::degree() << "))=
                " << k << endl;
388
```

```
389         if(limit==1 || probprime) // check order!
390         {
391             if(!check_order(k, a, b)) exit(1);
392         }
393     }
394     cout << endl;
395 }
396
397 void write_curve(const ZZ&o, const zz_pE& a, const zz_pE& b, const zz_pE& j)
398 {
399     fstream f;
400     string s = to_string<long>(zz_p::modulus(), dec) + "_" + to_string<long>(
            zz_pE::degree(), dec) + SATOH_OUTPUT_FILE_END;
401     string cpp = SATOH_OUTPUT_PATH_CPP + s;
402     string aribas = SATOH_OUTPUT_PATH_ARIBAS + s;
403
404     f.open(cpp.c_str(), ios::out|ios::app);
405
406     if(!f.good())
407     {
408         cout << "cannot write to " << cpp << endl;
409         cout << "maybe " << SATOH_OUTPUT_PATH_CPP << " does not exist" <<
                endl;
410         exit(1);
411     }
412
413     f << zz_pE::modulus() << endl;
414     f << a << endl;
415     f << b << endl;
416     f << j << endl;
417     f << o << endl;
418     f << endl;
419
420     f.close();
421     f.clear();
422
423     cout << cpp << " written" << endl;
424
425     f.open(aribas.c_str(), ios::out|ios::app);
426
427     if(!f.good())
428     {
429         cout << "cannot write to " << aribas << endl;
430         cout << "maybe " << SATOH_OUTPUT_PATH_ARIBAS << " does not exist" <<
                endl;
431         exit(1);
```

```
432     }
433
434     long i, k;
435     long p = zz_p::modulus();
436     long n = zz_pE::degree();
437     zz_pX g = zz_pE::modulus();
438
439     f << "p := " << p << ";" << endl;
440     f << "n := " << n << ";" << endl;
441
442     f << "f := (";
443     for(i=0; i<n; i++)
444     {
445         f << g.rep[i] << ", ";
446     }
447     f << g.rep[n] << ");" << endl;
448
449     k = rep(a).rep.length() - 1;
450     f << "a := (";
451     for(i=0; i<k; i++)
452     {
453         f << rep(a).rep[i] << ", ";
454     }
455     f << rep(a).rep[k] << ");" << endl;
456
457     k = rep(b).rep.length() - 1;
458     f << "b := (";
459     for(i=0; i<k; i++)
460     {
461         f << rep(b).rep[i] << ", ";
462     }
463     f << rep(b).rep[k] << ");" << endl;
464
465     k = rep(j).rep.length() - 1;
466     f << "j := (";
467     for(i=0; i<k; i++)
468     {
469         f << rep(j).rep[i] << ", ";
470     }
471     f << rep(j).rep[k] << ");" << endl;
472
473     f << "k := " << o << ";" << endl;
474     f << endl;
475
476     f.close();
477
```

```
478     cout << aribas << " written" << endl << endl;
479 }
480 
481 void search(long limit)
482 {
483     static double zeit;
484     zeit = GetTime();
485 
486     SetSeed(to_ZZ(long(time(0))));
487 
488     static ZZ o, o_twist, sum;
489     static zz_pE v, j, a, b;
490     static long p2 = zz_p::modulus()*zz_p::modulus();
491 
492     add(o, zz_pE::cardinality(), 1);
493     mul(sum, o, 2); // sum = 2*(zz_pE::cardinality() + 1)
494 
495     static ZZ k, i;
496     clear(k);
497     clear(i);
498 
499     while(true)
500     {
501         k++;
502 
503         while(true)
504         {
505             random(a); random(b);
506             if(IsZero(a) || IsZero(b)) continue;
507             if(non_singular(a, b)) break;
508         }
509 
510         j_invariant(j, a, b);
511         satoh(o, a, b, limit);
512 
513         if(IsZero(o))
514         {
515             cout << "throw curve away..." << endl << endl;
516             continue; // there was an early-abort at satoh(o, a, b, limit)
517         }
518         else
519         {
520             if(j!=power(j, p2)) i++; // j not in GF(p^2), so "Satoh" has
                    calculated the order...
521         }
522 
```

```
523         if(ProbPrime(o)) break;
524
525         sub(o_twist, sum, o);
526
527         if(ProbPrime(o_twist))
528         {
529             swap(o, o_twist);
530
531             cout << "order of twisted curve is prime..." << endl;
532
533             while(IsSquare(j)) random(j);
534
535             sqr(v, j);
536             mul(a, a, v); // a = a*j^2
537             mul(v, v, j);
538             mul(b, b, v); // b = b*j^3
539
540             j_invariant(j, a, b);
541
542             cout << "a=" << a << endl;
543             cout << "b=" << b << endl;
544
545             cout << "#E(GF(" << zz_p::modulus() << "^" << zz_pE::degree() <<
                    "))=" << o << endl;
546             if(!check_order(o, a, b)) exit(1);
547             cout << endl;
548
549             break;
550         }
551     }
552
553     write_curve(o, a, b, j);
554
555     cout << "Elliptic curve with prime order found after " << k << " tr";
556     if(k>1) cout << "ies";
557     else cout << "y";
558     cout << "." << endl;
559
560     if(i>0)
561     {
562          cout << "Running Satoh on " << i << " curve";
563          if(i>1) cout << "s";
564          cout << "." << endl;
565     }
566
567     cout << "Total time: ";
```

```
568     PrintTime2(GetTime()-zeit);
569     cout << endl;
570 
571     cerr << "\a"; // beep
572 }
```

Literaturverzeichnis

[Bac64] Bachman, George: *Introduction to p-Adic Numbers and Valuation Theory*. Academic Press, 1964.

[BCRS99] Blake, Ian, János Csirik, Michael Rubinstein und Gadiel Seroussi: *On the Computation of Modular Polynomials for Elliptic Curves*. Technischer Bericht, Hewlett-Packard Laboratories, 1999.
`http://www.math.uwaterloo.ca/~mrubinst/publications/phi.pdf`.

[BK03] Baier, Harald und Günter Köhler: *How to Compute the Coefficients of the Elliptic Modular Function $j(z)$*. Experimental Mathematics, 12(1):115–121, 2003.

[Blu97] Bluher, Antonia W.: *A Leisurely Introduction to Formal Groups and Elliptic Curves*, 1997.
`http://www.math.uiuc.edu/Algebraic-Number-Theory/0076/`.

[Bos06] Bosch, Siegfried: *Algebra*. Springer, Sechste Auflage, 2006.

[BSS99] Blake, Ian, Gadiel Seroussi und Nigel Smart: *Elliptic Curves in Cryptography*, Band 265 der Reihe *LMS Lecture Note Series*. Cambridge University Press, 1999.

[BSS05] Blake, Ian, Gadiel Seroussi und Nigel Smart: *Advances in Elliptic Curve Cryptography*, Band 317 der Reihe *LMS Lecture Note Series*. Cambridge University Press, 2005.

[CF06] Cohen, Henri und Gerhard Frey: *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete Mathematics and Its Applications. Chapman & Hall/CRC, 2006.

[CL05] Charles, Denis und Kristin Lauter: *Computing modular polynomials*. LMS Journal of Computation and Mathematics, 8:195–204, 2005.

[Con99] Connell, Ian: *Elliptic Curve Handbook*, 1999.
`http://www.math.mcgill.ca/connell/public/ECH1/`.

[Cyg] *Cygwin – a Linux-like environment for Windows*.
`http://cygwin.com/`.

[Deu41] Deuring, Max: *Die Typen der Multiplikatorenringe elliptischer Funktionenkörper.* Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, 14:197–272, 1941.

[Eng99] Enge, Andreas: *Elliptic curves and their applications to cryptography: an introduction.* Kluwer Academic Publishers, 1999.

[Eng07] Enge, Andreas: *Computing modular polynomials in quasi-linear time.* Preprint, April 2007. `http://www.lix.polytechnique.fr/Labo/Andreas.Enge/vorabdrucke/modcomp.pdf`.

[FB06] Freitag, Eberhard und Rolf Busam: *Funktionentheorie 1.* Springer, Vierte Auflage, 2006.

[FGH00] Fouquet, Mireille, Pierrick Gaudry und Robert Harley: *An extension of Satoh's algorithm and its implementation.* Journal of the Ramanujan Mathematical Society, 15(4):281–318, 2000.

[FGH01] Fouquet, Mireille, Pierrick Gaudry und Robert Harley: *Finding Secure Curves with the Satoh-FGH Algorithm and an Early-Abort Strategy.* In: *Advances in Cryptology – EUROCRYPT 2001*, Band 2045 der Reihe *Lecture Notes in Computer Science*, Seiten 14–29. Springer, 2001.

[For] Forster, Otto: *ARIBAS Interpreter for Arithmetic.* `http://www.mathematik.uni-muenchen.de/~forster/sw/aribas.html`.

[For96] Forster, Otto: *Algorithmische Zahlentheorie.* Vieweg, 1996.

[For04] Forster, Otto: *Analysis 1.* Vieweg, Siebte Auflage, 2004.

[GK99] Goldwasser, Shafi und Joe Kilian: *Primality Testing Using Elliptic Curves.* Journal of the ACM, 46(4):450–472, 1999.

[Gra] Granlund, Torbjörn: *GNU Multiple Precision Arithmetic Library.* `http://gmplib.org/`.

[Har77] Hartshorne, Robin: *Algebraic Geometry*, Band 52 der Reihe *Graduate Texts in Mathematics.* Springer, 1977.

[Har02] Harley, Robert: *Asymptotically optimal p-adic point-counting.* E-Mail an NMBRTHRY Mailingliste, Dezember 2002. `http://listserv.nodak.edu/archives/nmbrthry.html`.

[Her75] Herrmann, Oskar: *Über die Berechnung der Fourierkoeffizienten der Funktion $j(\tau)$.* Journal für die reine und angewandte Mathematik, 274:187–195, 1975.

[Hus04] Husemöller, Dale: *Elliptic Curves*, Band 111 der Reihe *Graduate Texts in Mathematics*. Springer, Zweite Auflage, 2004.

[KK98] Koecher, Max und Aloys Krieg: *Elliptische Funktionen und Modulformen*. Springer, 1998.

[Kob84] Koblitz, Neal: *p-adic Numbers, p-adic Analysis, and Zeta-Functions*, Band 58 der Reihe *Graduate Texts in Mathematics*. Springer, Zweite Auflage, 1984.

[Kob87] Koblitz, Neal: *Elliptic Curve Cryptosystems*. Mathematics of Computation, 48(177):203–209, 1987.

[Lan73] Lang, Serge: *Elliptic Functions*. Addison-Wesley, 1973.

[Len87] Lenstra, Hendrik Willem: *Factoring integers with elliptic curves*. Annals of Mathematics, 126(3):649–673, 1987.

[LST64] Lubin, Jonathan, Jean-Pierre Serre und John Tate: *Elliptic curves and formal groups*. Lecture notes prepared in connection with the seminars held at the Summer Institute on Algebraic Geometry, Whitney Estate, Woods Hole, Massachusetts, 1964.
`http://ma.utexas.edu/users/voloch/lst.html`.

[Mad03] Madsen, Marc Skov: *A General Framework for p-adic Point Counting and Application to Elliptic Curves on Legendre Form*. Preprint, November 2003.
`http://home.imf.au.dk/marc/doc/phd/legendre_elliptic.pdf`.

[Mad04] Madsen, Marc Skov: *p-adic Point Counting on Elliptic Curves*. Dissertation, Århus Universitet, Mai 2004.
`http://home.imf.au.dk/marc/doc/phd/thesis.pdf`.

[McK94] McKee, James: *Computing division polynomials*. Mathematics of Computation, 63(208):767–771, 1994.

[Mes72] Messing, William: *The Crystals Associated to Barsotti-Tate Groups: with Applications to Abelian Schemes*, Band 264 der Reihe *Lecture Notes in Mathematics*. Springer, 1972.

[Mil86] Miller, Victor S.: *Use of Elliptic Curves in Cryptography*. In: *Advances in Cryptology – CRYPTO '85*, Band 218 der Reihe *Lecture Notes in Computer Science*, Seiten 417–426. Springer, 1986.

[Min] *Minimalist GNU for Windows*.
`http://www.mingw.org/`.

[Moe73] Moenck, Robert T.: *Fast computation of GCDs.* In: *Proceedings of the 5th Annual ACM Symposium on the Theory of Computing*, Seiten 142–151. Association for Computing Machinery, 1973.

[Neu92] Neukirch, Jürgen: *Algebraische Zahlentheorie.* Springer, 1992.

[RSA78] Rivest, Ronald L., Adi Shamir und Leonard Adleman: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, 21(2):120–126, 1978.

[Rub] Rubinstein, Michael: *Classical Modular Polynomials.* `http://www.math.uwaterloo.ca/~mrubinst/modularpolynomials/phi_l.html`.

[Sat00] Satoh, Takakazu: *The Canonical Lift of an Ordinary Elliptic Curve over a Finite Field and Its Point Counting.* Journal of the Ramanujan Mathematical Society, 15(4):247–270, 2000.

[Sat02] Satoh, Takakazu: *On p-adic Point Counting Algorithms for Elliptic Curves over Finite Fields.* In: *Algorithmic Number Theory*, Band 2369 der Reihe *Lecture Notes in Computer Science*, Seiten 43–66. Springer, 2002.

[Sch85] Schoof, René: *Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p.* Mathematics of Computation, 44(170):483–494, 1985.

[Sch95] Schoof, René: *Counting points on elliptic curves over finite fields.* Journal de Théorie des Nombres de Bordeaux, 7:219–254, 1995.

[Ser62] Serre, Jean-Pierre: *Corps locaux.* Hermann, 1962.

[Sho] Shoup, Victor: *NTL: A Library for doing Number Theory.* `http://www.shoup.net/ntl/`.

[Sil86] Silverman, Joseph H.: *The Arithmetic of Elliptic Curves*, Band 106 der Reihe *Graduate Texts in Mathematics.* Springer, 1986.

[Sil94] Silverman, Joseph H.: *Advanced Topics in the Arithmetic of Elliptic Curves*, Band 151 der Reihe *Graduate Texts in Mathematics.* Springer, 1994.

[Skj03] Skjernaa, Berit: *Satoh's algorithm in characteristic 2.* Mathematics of Computation, 72(241):477–487, 2003.

[Tig01] Tignol, Jean-Pierre: *Galois' Theory of Algebraic Equations.* World Scientific, 2001.

[Ver03] Vercauteren, Frederik: *Computing zeta functions of curves over finite fields.* Dissertation, Katholieke Universiteit Leuven, November 2003. `http://homes.esat.kuleuven.be/~fvercaut/papers/thesis.ps`.

[Vél71] Vélu, Jacques: *Isogénies entre courbes elliptiques.* Comptes rendus hebdomadaires des séances de l'Académie des sciences, 273:A238–A241, 1971.

[Was08] Washington, Lawrence C.: *Elliptic Curves: Number Theory and Cryptography.* Discrete Mathematics and Its Applications. Chapman & Hall/-CRC, Zweite Auflage, 2008.

[Wil95] Wiles, Andrew: *Modular elliptic curves and Fermat's Last Theorem.* Annals of Mathematics, 141(3):443–551, 1995.

Erklärung

Hiermit erkläre ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

München, den 20. Oktober 2008