Masterarbeit

Polynomielle Primzahltests mit elliptischen Kurven

Erstellt von Georg Hahn
Betreuer Professor Dr. Stefan Müller-Stach

Mathematisches Institut
Johannes Gutenberg Universität Mainz

Inhaltsverzeichnis

1 Einführung

Mit dem Beweis des Mathematikers Euklid um 300 v. Chr. über die Existenz unendlich vieler Primzahlen beginnt die Suche nach effizienten Verfahren, um die Primalität von immer größeren Zahlen zu zeigen. Während es jedoch nach den Arbeiten von Fermat, Miller-Rabin und Solovay-Strassen relativ leicht ist, die Nicht-Primalität einer Zahl zu zeigen, blieben praktisch einsetzbare Primzahlbeweise, d.h. solche mit polynomieller Laufzeit, bis 2002 ein ungelöstes Problem. In diesem Jahr wurde mit dem Primzahltest nach Agrawal, Kayal und Saxena der erste Test vorgestellt, der beweisbar in polynomieller Zeit für alle Eingaben terminiert.

Diese Arbeit behandelt polynomielle Primzahltests im Allgemeinen und den Test nach Goldwasser-Kilian im Speziellen, dessen theoretischer Hintergrund auf der Theorie der sogenannten elliptischen Kurven aufbaut. Zunächst werden polynomielle Tests anhand des Algorithmus von Agrawal-Kayal-Saxena in Kapitel 2 kurz eingeführt. Die Konzentration dieser Arbeit liegt jedoch auf den Tests von Goldwasser-Kilian und Atkin-Morain. Die dazu notwendige Theorie der elliptischen Kurven (elementare Definitionen, Gruppengesetz, Isogenien, elliptische Kurven über endlichen Körpern, Satz von Hasse u.a.) werden zusammen mit der Idee von Primalitätszertifikaten (insbesondere dem Pratt-Zertifikat) in Kapitel 3 behandelt. Dazu gehören auch der Satz von Goldwasser-Kilian sowie der daraus resultierende Primzahltest und dessen Korrektheit. Zudem beinhaltet diese Arbeit eine ausführliche Laufzeitbetrachtung, die auf der Vermutung von Goldwasser-Kilian über eine Abschätzung der Primzahlverteilung in Intervallen aufbaut.

Eine Verbesserung des Goldwasser-Kilian Tests zum momentanen (Stand 2010) Standard „ECPP“ (Elliptic Curve Primality Proving) wurde durch die Arbeiten von Atkin und Morain über die sogenannte Komplexe Multiplikation möglich. Diese Methode wird in Kapitel 4 zusammen mit ihrem theoretischen Hintergrund (elliptische Kurven über $\mathbb{C}$, Weierstraß $\wp$-Funktion, j-Funktion, Fundamentaldiskriminanten, Hilbert'sche Klassenpolynome, Methode der Komplexen Multiplikation, Konstruktion elliptischer Kurven mit vorgegebener Ordnung u.a.) und dem verbesserten Test hergeleitet.

Im Rahmen der Arbeit wurden beide Tests nach Goldwasser-Kilian und Atkin-Morain sowie eine Funktion zur Verifikation bereits erstellter Primalitätszertifikate für die Mathematiksoftware SAGE (siehe [27]) programmiert. Empirische Ergebnisse dazu folgen in Kapitel 5, die die polynomielle Laufzeit der Primzahltests demonstrieren. Die Arbeit schließt mit einer Diskussion der Ergebnisse und einem Ausblick im letzten Kapitel. Der Code für SAGE in der Programmiersprache Cython befindet sich im Anhang der Arbeit.

2 Der Primzahltest nach Agrawal-Kayal-Saxena

Die vorliegende Arbeit behandelt Primzahltests mit polynomieller Laufzeit. Der Algorithmus von Goldwasser-Kilian und dessen Erweiterung von Atkin-Morain stehen dabei im Vordergrund. Als Einführung in die Thematik werden die Tests von Miller-Rabin (polynomielle Laufzeit unter der Annahme der verallgemeinerten Riemann'schen Vermutung) und der erste Primzahltest mit bewiesener polynomieller Laufzeit nach Agrawal-Kayal-Saxena vorgestellt. Diese Einführung folgt Kapitel 11 in [9].

2.1 Theorem (Miller-Rabin)**:** *Sei ein ungerades $n \geq 3$ gegeben. Wähle $t \in \mathbb{N}_0$ entsprechend der Darstellung $n-1 = 2^t m$ mit $2 \nmid m$. Dann ist n genau dann prim, falls für alle $0 < a < n$ mit größtem gemeinsamen Teiler $\gcd(a,n) = 1$ entweder*

$$a^m \equiv 1 \pmod{n} \quad \text{oder} \quad a^{2^s m} \equiv 1 \pmod{n} \text{ für ein } s \in \{0, 1, \ldots, t-1\}$$

gilt.

Ist n keine Primzahl, so erfüllen höchstens ein Viertel der a's eine der Bedingungen.

Beweis. Siehe Satz 11.14 in [9]. □

2.2 Bemerkung: *Der Miller-Rabin Test wird hauptsächlich als probabilistischer Primzahltest benutzt. Sind nämlich k Basen a gefunden, die eine der Bedingungen des Satzes erfüllen, so muss n mit Wahrscheinlichkeit $1-(1/4)^k$ prim sein. Eine Basis a, die keine der Bedingungen erfüllt, ist ein Zeuge für die Nicht-Primalität von n.*

Mit dem Satz von Miller-Rabin kann natürlich auch die Primalität von n bewiesen werden. Da aber dafür nach dem Satz alle $0 < a < n$ getestet werden müssen, hat diese Variante exponentielle Laufzeit.

Unter der Annahme der verallgemeinerten Riemann'schen Vermutung reicht es aus, alle $0 < a \leq 2\log^2 n$ zu testen. Damit würde der Miller-Rabin Test zu einem deterministischen polynomiellen Algorithmus.

Ohne Annahme einer Vermutung konnte zuerst für den deterministischen Test nach Agrawal-Kayal-Saxena (im Folgenden auch kurz als AKS-Test bezeichnet) die polynomielle Laufzeit gezeigt werden. Dieser beruht auf dem folgenden Satz.

2.3 Theorem: *Ein $n \in \mathbb{N}$ ist genau dann prim, falls*

$$(X+a)^n \equiv X^n + a \pmod{n}$$

in $\mathbb{Z}[X]$ für alle $a \in \mathbb{Z}$ gilt.

Beweis. Siehe Satz 11.17 in [9]. □

Beim AKS-Test wird die Bedingung aus dem Satz nicht nur in $\mathbb{Z}/n\mathbb{Z}[X]$ getestet, sondern zudem modulo eines Polynoms $x^r - 1$. Nach [20] lässt sich dies wie folgt in einen sehr technischen Primzahltest übersetzen:

2.4 Theorem (AKS nach M. Agrawal, N. Kayal, N. Saxena, H. W. Lenstra Jr.)**:** *Seien n, r und v positive natürliche Zahlen. Sei $S \subset \mathbb{N}$ eine endliche Menge.*

Angenommen, $\gcd(n, r) = 1$*, n besitzt Ordnung v modulo r,* $\gcd(n, b - b') = 1$ *gilt für alle verschiedenen $b, b' \in S$,* $\binom{\#S + \phi(r) - 1}{\#S} \geq n^{2d\lfloor\sqrt{\phi(r)/d}\rfloor}$ *für alle positiven $d|(\phi(r)/v)$ und $(x + b)^n \equiv x^n + b$ im Ring $\mathbb{Z}/n\mathbb{Z}[X]/(x^r - 1)$ für alle $b \in S$.*

Dann ist n eine Primzahlpotenz.

Beweis. Siehe Theorem 2.3 in [20]. □

2.5 Bemerkung: *Ob n eine Primzahlpotenz ist, lässt sich leicht in polynomieller Laufzeit bestimmen. Die höchste Potenz, für die n Primzahlpotenz sein kann, ist sicherlich* $\log_2(n)$*. Daher muss lediglich in $O(\log n)$ für alle $2 \leq k \leq \log_2(n)$ numerisch $w = \lfloor \sqrt[k]{n} \rfloor$ bestimmt und auf $w^k = n$ getestet werden.*

Satz 2.4 liefert einen polynomiellen Algorithmus wie folgt (siehe Kapitel 3 in [20]):

1. Teste, ob n ein Primzahlpotenz ist. Falls zutreffend, so ist n nicht prim.
2. Berechne $N := 2n(n-1)(n^2-1)(n^3-1)\cdots(n^{4\lceil\log n\rceil^2}-1)$. Ermittle die kleinste Primzahl r, die N nicht teilt (siehe [20] für eine Abschätzung, dass r polynomiell beschränkt ist).
3. Teste, ob n gleich einer Primzahl kleiner r ist. Falls zutreffend, ist n prim.
4. Teste $(x+b)^n \equiv x^n + b$ in $\mathbb{Z}/n\mathbb{Z}[x]/(x^r - 1)$ für alle $b \in S$, wobei $S := \{1, 2, \ldots, r\}$. Falls die Bedingung für ein b nicht gilt, so ist n nach Satz 2.3 nicht prim.
5. Nach Satz 2.4 ist n prim.

Alle obigen Operationen können in polynomieller Zeit durchgeführt werden. Die ursprüngliche Version von Agrawal-Kayal-Saxena hat eine Laufzeit von $O(\log^{12} n)$. Diverse Verbesserungen findet man in [20], durch die die Laufzeit bis auf $O(\log^{6+\epsilon} n)$ und sogar $O(\log^{3+\epsilon} n)$ verringert werden kann.

Die technische Gestalt des AKS-Tests ist der Grund dafür, dass in der Praxis ein einfacherer und leicht implementierbarer Primzahltest verwendet wird. Dies ist der Test von Goldwasser-Kilian (siehe nächstes Kapitel 3) bzw. dessen Verbesserung von Atkin-Morain zum aktuellen (2010) Standard „ECPP“ (siehe Kapitel 4).

3 Der Goldwasser-Kilian Primzahltest

Der Test nach Goldwasser-Kilian zertifiziert Primzahlen durch eine Reihe von Operationen auf sogenannten elliptischen Kurven. Die Idee der Zertifikate und die Theorie der elliptischen Kurven wird nun eingeführt.

3.1 Zertifizierung von Primzahlen

Soll bei einem klassischen Primzahltest, beispielsweise beim Sieb des Eratosthenes, ein Ergebnis verifiziert werden, so muss der Test komplett neu durchgeführt werden. Es besteht somit kein Unterschied darin, ob eine Zahl n das erste Mal auf Primalität getestet wird oder ob ein bestehendes Ergebnis (ein vorheriger Durchlauf lieferte bereits, dass n prim bzw. nicht prim ist) überprüft wird. Kann ein Algorithmus berechnete Zwischenwerte so ausgeben, dass mit diesen alleine das Ergebnis verifiziert werden kann, so spricht man von einem (Primzahl-) Zertifikat. Diese Idee geht auf Vaughan Pratt (1975, siehe [12] und [26]) zurück. Veranschaulicht wird die Idee am Lucas-Test:

3.1 Theorem (Lucas-Test)**:** *Sei $n \in \mathbb{N}$ gegeben. Falls ein $x \in \mathbb{N}$, $\gcd(x, n) = 1$, existiert mit*

1. $x^{n-1} \equiv 1 \pmod{n}$,
2. *Für jeden Primfaktor p von $n-1$ gilt $x^{(n-1)/p} \not\equiv 1 \pmod{n}$,*

dann ist n prim.

Beweis. Die beiden Bedingungen bedeuten, dass x die volle Ordnung $n-1$ in $(\mathbb{Z}/n\mathbb{Z})^\times$ (der Einheitengruppe von $\mathbb{Z}/n\mathbb{Z}$) besitzt. Da $\mathrm{ord}(x)$ die Gruppenordnung teilt, folgt $n-1|\phi(n)$, wobei $\phi(n) = \#\{1 \leq a \leq n : \gcd(a, n) = 1\}$ die Euler-ϕ-Funktion beschreibt. Wegen $\phi(n) < n$ folgt $\phi(n) = n-1$ und somit ist n prim nach Definition von $\phi(n)$. □

3.2 Bemerkung: *Ist nun eine Zahl n und eine Faktorisierung $n-1 = p_1 \cdots p_m$ gegeben, so muss zum Primalitätsbeweis von n nur noch ein x gefunden werden, dass die Bedingungen des Lucas-Tests erfüllt. Da der Test aber alle Primfaktoren von $n-1$ voraussetzt, muss ein vollständiger Beweis für n auch die Primeigenschaft aller p_i beinhalten. Dazu kann der Lucas-Test erneut angewendet werden, und zwar sobald die Faktorisierungen aller $p_i - 1$ bekannt sind. Diese Vorgehensweise liefert ein rekursives Zertifikat für n, bestehend aus n, x und der Faktorisierung von $n-1 = p_1 \cdots p_m$ in Primfaktoren, wobei die Primalität aller $p_i < n$ erneut per Lucas-Test gezeigt wird. Die Rekursion endet, sobald die p_i eine gewisse Schranke wie beispielsweise 10^6 unterschreiten und direkt per Probedivision oder Primzahltabelle getestet werden können. Dies ist das Pratt-Zertifikat.*

Der Vorteil eines solchen Zertifikates gegenüber klassischen Tests (Probedivision, Sieb des Eratosthenes, etc.) ist die Möglichkeit einer schnellen Verifizierung. Ist nämlich ein

vollständiges Zertifikat berechnet, so muss lediglich Schritt 2 des Lucas-Tests in $O(1)$ überprüft werden. Die Faktorisierungen und die Suche nach den x entfällt. Bei klassischen Tests gibt es keinen Unterschied zwischen einer Erstberechnung oder einer späteren Verifizierung.

Das einzige Problem dieses Tests liegt in der notwendigen Faktorisierung von $n-1$, da momentan (Stand 2010) keine Faktorisierungen in polynomieller Zeit möglich sind. Dies bedeutet, dass auch das Pratt-Zertifikat nicht in polynomieller Zeit berechnet werden kann.

3.2 Elliptische Kurven

Es folgt eine Einführung in die notwendige Theorie über elliptische Kurven, die dem Goldwasser-Kilian Test zugrunde liegt. In allen folgenden Kapiteln sei K der Grundkörper, über dem die elliptischen Kurven betrachtet werden.

3.3 Definition: *Der affine Raum $\mathbb{A}_K^n$ und der projektive Raum $\mathbb{P}_K^n$ über einem Körper K sind definiert als*

$$\begin{aligned} \mathbb{A}^n &:= \mathbb{A}_K^n := \{(a_1,\ldots,a_n) : a_i \in K\}, \\ \mathbb{P}^n &:= \mathbb{P}_K^n := \{(a_0 : a_1 : \ldots : a_n) : a_i \in K \text{ und nicht alle } 0\}. \end{aligned}$$

Eine affine algebraische Menge $V \subseteq \mathbb{A}^n$ ist die Menge aller Lösungen eines Systems von Polynomgleichungen in den Variablen $x_1,\ldots,x_n$, d.h.

$$V : \begin{cases} f_1(x_1,\ldots,x_n) = 0 \\ \vdots \\ f_m(x_1,\ldots,x_n) = 0 \end{cases}$$

Eine projektive algebraische Menge $V \subseteq \mathbb{P}^n$ ist die Menge aller Lösungen eines Systems von homogenen Polynomgleichungen in den Variablen $x_0,\ldots,x_n$.

3.4 Definition:

1. *Eine projektive Kurve ist eine unendliche, irreduzible algebraische Menge $C \subseteq \mathbb{P}^n$ mit der Eigenschaft, dass jede echte algebraische Teilmenge $Y \subsetneq C$ endlich ist.*
2. *Eine affine Kurve $C = \{f_1(x_1,\ldots,x_n) = 0,\ldots,f_m(x_1,\ldots,x_n) = 0\}$ ist nicht-singulär (oder glatt) am Punkt $P = (a_1,\ldots,a_n)$, falls die Matrix $A = \left(\frac{\partial f_i}{\partial x_j}\right)_{i,j}(P)$ den Rang $n-1$ besitzt. Ansonsten ist C singulär.*

   Für eine Kurve $C : f(x,y) = 0$, bestimmt durch f in zwei Variablen, ist Singularität an $P = (a,b)$ äquivalent zu $\frac{\partial f}{\partial x}(P) = \frac{\partial f}{\partial y}(P) = 0$.
3. *C ist nicht-singulär, falls C an jedem ihrer Punkte nicht-singulär ist.*

3.5 Bemerkung: *Die Menge der rationalen Funktionen* $f = \frac{g(x_0:\ldots:x_n)}{h(x_0:\ldots:x_n)}$ *auf* $\mathbb{P}^n = \mathbb{P}^n_K$ *über einem Körper* K *bildet selbst einen Körper* $k(\mathbb{P}^n)$*. Ist* $C \subseteq \mathbb{P}^n$ *eine irreduzible Kurve,* $f = \frac{g}{h} \in k(\mathbb{P}^n)$ *mit* $h \neq 0$ *auf* C*, so erhält man durch die Einschränkung*

$$f : C \setminus \{\text{endliche Menge}\} \to k$$

eine rationale Funktion auf C *(nicht definiert für die endliche Nullstellenmenge* $h = 0$*). Diese bildet einen Körper* $k(C)$*. Für eine irreduzible Kurve* $C \subseteq \mathbb{P}^2$*, definiert durch* $f(x_0 : x_1 : x_2) = 0$*, ist* $k(C)$ *der Quotientenkörper* $k(C) = Q\left(\frac{k[x,y]}{(f)}\right)$*.*

Mit der Definition einer Kurve können direkt auch Abbildungen zwischen Kurven eingeführt werden.

3.6 Definition:

1. *Seien* $C \subseteq \mathbb{P}^n$ *und* $D \subseteq \mathbb{P}^m$ *zwei Kurven. Eine rationale Funktion* $\phi : C \to D$ *ist gegeben durch rationale Funktionen*

   $$\phi(P) = (f_0(P) : \ldots : f_m(P)) \in D,$$

   wobei $P \in C$ *ist und nicht alle* $f_i(P) \in k(C)$ *Null sind.*

2. *Eine nicht-konstante Funktion* $\phi : C \to D$ *induziert einen sogenannten pullback* $\phi^* : k(D) \to k(C)$ *durch* $f \to \phi^*(f) = f \circ \phi$ *(da* $C \xrightarrow{\phi} D \xrightarrow{f} k$*).*

3. *Der Grad von* ϕ *wird dann definiert als der Grad der Körpererweiterung des pullbacks,*

   $$\deg(\phi) := [k(C) : \phi^* k(D)]\,.$$

Die obigen Definitionen werden an einem Beispiel veranschaulicht.

3.7 Beispiel: *Der Kreis* $C : f(x, y) = x^2 + y^2 - 1 = 0$ *(über* $\mathbb{R}$*) wird auf eine Gerade* $D : y = 0$ *abgebildet. Die Bedingung* $\frac{\partial f}{\partial x}(P) = \frac{\partial f}{\partial y}(P) = 0$ *ist nur für* $(0, 0) \notin C$ *erfüllt, daher ist* C *glatt. Rechts notiert ist jeweils der Funktionenkörper.*

$$\begin{array}{ll} C : x^2 + y^2 = 1 & k(C) \cong k(x, \sqrt{1 - x^2}) \\ \downarrow \phi(x, y) = (x, 0) & \uparrow \phi^*(x) = x \circ \phi = x \circ (x, 0) = x \\ D : y = 0 & k(D) \cong k(x) \end{array}$$

Damit folgt sofort, dass $\deg(\phi) = \left[k(x, \sqrt{1 - x^2}) : k(x)\right] = 2$*. Dieses Ergebnis bekommt man auch, indem man den pullback* ϕ^* *von* D *auf* C *betrachtet. Jedes* $x \in D$ *hat zwei Urbilder* $(x, \pm\sqrt{1 - x^2})$ *in* C*, dies entspricht* $\deg(\phi) = 2$*.*

Mit dem Begriff des Geschlechtes einer Kurve kann nun auch die Definition einer elliptischen Kurve eingeführt werden.

3.8 Definition:

1. *Für eine nicht-singuläre projektive Kurve* $C \subseteq \mathbb{P}^2$, $C : f(x_0 : x_1 : x_2) = 0$ *mit* $d = \deg(f)$, *ist das Geschlecht (genus)* g *definiert als*

$$g := \frac{(d-1)(d-2)}{2}.$$

   Ist $g = 1$, *so ist* C *eine Kubik.*

2. *Wird eine projektive Kurve der Einfachheit halber als* $f(x, y) = 0$ *angegeben, so ist implizit deren Homogenisierung* $f(x : y : z) = 0$ *gemeint.*

3. *Eine* ***elliptische Kurve*** *über einem Körper* K *ist eine nicht-singuläre projektive Kurve* E/K *vom Geschlecht* $g = 1$ *mit einem ausgezeichneten Punkt* O, *notiert als* (E, O).

4. *Die Menge der projektiven Punkte, die auf* E *liegen und Koordinaten aus dem Körper* K *besitzen, wird mit* $E(K)$ *bezeichnet.*

Elliptische Kurven werden standardmäßig mit ihrer sogenannten Weierstraß-Gleichung angegeben.

3.9 Definition: *In beliebiger Charakteristik definiert*

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

eine elliptische Kurve in generalisierter Weierstraß-Form ($a_i \in K$). *Für Charakteristik* $\neq 2, 3$ *kann diese Gleichung durch Ergänzung des Quadrats auf der linken und der Kubik auf der rechten Seite umgeformt werden zur vereinfachten Weierstraß-Gleichung*

$$y^2 = x^3 + ax + b.$$

Da eine elliptische Kurve per Definition projektiv ist, ist mit $y^2 = x^3 + ax + b$ *eigentlich die Homogenisierung* $y^2z = x^3 + axz^2 + bz^3$ *gemeint. Alle Punkte* $(x : y : z) \in \mathbb{P}^2$ *mit* $z \neq 0$ *entsprechen damit der Klasse* $(x/z : y/z : 1)$. *Für* $z = 0$ *erhält man nach Einsetzen in die homogene Weierstraß-Gleichung* $0 = x^3$, *dies entspricht also der Klasse* $(0 : y : 0) = (0 : 1 : 0) \in \mathbb{P}^2$. *Dieser Punkt ist erst durch die Homogenisierung zur Kurve hinzugekommen und wird der ausgezeichnete „Punkt im Unendlichen“ genannt, notiert als* $O = (0 : 1 : 0)$. *Die Menge* K*-rationaler Punkte auf* E *ist damit*

$$E(K) = \{(x, y) \in K^2 : y^2 = x^3 + ax + b\} \cup \{O\}.$$

In vereinfachter Weierstraß-Form wird E *auch als* $E(a, b)$ *notiert und der Körper* K *implizit vorausgesetzt.*

Die Definitionen der elliptischen Kurve per Definition 3.8 und per Weierstraß-Gleichung sind konsistent, wie der folgende Satz zeigt:

3.10 Theorem: *Jede elliptische Kurve ist isomorph zu einer Kurve in Weierstraß-Form.*

Beweis. Siehe [10], Kapitel 2. □

Bevor elliptische Kurven über endlichen Körpern betrachtet werden, folgt die Isomorphie zwischen elliptischen Kurven.

3.11 Theorem: *Zwei elliptische Kurven $E(a,b)$ und $E(a',b')$ sind isomorph über $\bar{K}$ genau dann, wenn ein $c \in \bar{K}^\times$ existiert mit $a' = c^4 a$, $b' = c^6 b$, wobei der Isomorphismus durch die Abbildung $(x,y) \to (c^2 x, c^3 y)$ gegeben ist.*

Beweis. Siehe [5], III, Proposition 3.1. □

3.12 Definition: *Für eine elliptische Kurve $E(a,b)$ in vereinfachter Weierstraß-Form werden*

$$\Delta(E) := -16(4a^3 + 27b^2), \;\; j(E) := \frac{1728(-4a)^3}{\Delta(E)}$$

die Diskriminante und die j-Invariante von E genannt.

Die j-Invariante ist die Invariante der Isomorphieklasse von elliptischen Kurven, wie der folgende Satz zeigt.

3.13 Theorem: *Für zwei elliptische Kurven $E(a,b)$, $E'(a',b')$ gilt über $\bar{K}$:*

$$E \cong E' \Leftrightarrow j(E) = j(E').$$

Beweis. Nach Satz 3.11 ist $E \cong E'$ genau dann, wenn ein $c \in \bar{K}^\times$ existiert mit $a' = c^4 a$ und $b' = c^6 b$. Ziehen der vierten bzw. sechsten Wurzel liefert die äquivalente Bedingung $\sqrt[4]{\frac{a'}{a}} = c = \sqrt[6]{\frac{b'}{b}}$. Anschließendes Potenzieren der Gleichung zum Exponenten 12 liefert $\left(\frac{a'}{a}\right)^3 = \left(\frac{b'}{b}\right)^2$. Umstellen ergibt

$$\frac{4a^3 + 27b^2}{a^3} = \frac{4a'^3 + 27b'^2}{a'^3},$$

dass nach Invertierung und Multiplikation mit $\frac{1728(-4)^3}{-16}$ genau $j(E) = j(E')$ entspricht. □

Dieser Satz ermöglicht es, die elliptischen Kurven nur durch die Größe j zu klassifizieren. Im nächsten Abschnitt wird eine Struktur auf $E(K)$ eingeführt.

3.3 Das Gruppengesetz auf einer elliptischen Kurve

Die Menge der Punkte $E(K)$ einer elliptischen Kurve (E, O) mit Koordinaten in K besitzt eine Gruppenstruktur mit neutralem Element O, gegeben durch die folgende Regel:

3.14 Definition (Additionsregel)**:** *Seien P und Q Punkte auf E und sei L die Gerade durch P und Q (bzw. die Tangente im Fall $P = Q$). Diese Gerade L schneidet die Kurve in einem dritten Punkt R (dies folgt aus der Schnittzahl, siehe Kapitel 1 in [11]).*

Man definiert dann $P + Q := -R$, wobei $-R$ der zweite Schnittpunkt von $E(K)$ mit einer vertikalen Gerade G durch R ist (und $-R = R$ falls G eine Tangente an E ist).

Explizite Formeln sind im nächsten Satz zusammengefasst.

3.15 Theorem: *Für eine elliptische Kurve (E, O) gilt eine Gruppenstruktur $(E, +)$ wie folgt:*

1. *Das neutrale Element ist O.*
2. *Die Inverse von $P = (x, y)$ ist $-P = (x, -y)$.*
3. *Die Addition von $P = (x_1, y_1)$ und $Q = (x_2, y_2) \neq -P$ ist gegeben durch*

$$P + Q = (x, -\kappa(x - x_1) - y_1),$$

wobei $x = \kappa^2 - x_1 - x_2$ und $\kappa = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & P = Q \end{cases}$

4. *Es gilt $(P + Q) + R = P + (Q + R)$ für $P, Q, R \in E$.*

Beweis. (1) und (2) folgen aus der obigen Regel. Seien nun $P = (x_1, y_1)$ und $Q = (x_2, y_2)$ gegeben. Der Punkt $R = (x, y)$ muss auf der Geraden durch P und Q liegen, d.h. es gilt (a) $y = \kappa(x - x_1) + y_1$ mit Steigung $\kappa = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & P = Q \end{cases}$.

Außerdem liegt $R = (x, y)$ auf E und erfüllt (b) $y^2 = x^3 + ax + b$. Lösen der zwei Gleichungen (a) und (b) nach x und y liefert $x = \kappa^2 - x_1 - x_2$, $y = \kappa(x - x_1) + y_1$. Schließlich ist $P + Q = -R = (x, -y)$ nach Definition. Die Assoziativität kann durch explizites Nachrechnen mit Hilfe der Additionsformel verifiziert werden. □

Nach Einführung der Addition von Punkten auf einer elliptischen Kurve wird eine Multiplikationsabbildung einfach durch wiederholte Addition definiert.

3.16 Definition: *Für $P \in E$ und ein $m \in \mathbb{Z}$ definiert man eine Multiplikationsabbildung $[m] : E \to E$ durch*

$$\begin{array}{lll} [0]P := O & & \\ [m]P := P + \ldots + P & (m \text{ Mal}) & m > 0 \\ [m]P := (-P) + \ldots + (-P) & (|m| \text{ Mal}) & m < 0. \end{array}$$

Die Ordnung $\mathrm{ord}(P)$ *eines Punktes $P \in E(K)$ ist die kleinste Zahl $n \in \mathbb{N}$ mit $[n]P = O$.*

Da $(E(K), +)$ nach 3.15 eine Gruppe ist, gilt nach dem Satz von Lagrange auch

$$[\mathrm{ord}(E)]\, P = O,$$

wobei $\mathrm{ord}(E) = \#E(K)$.

3.17 Bemerkung: *$[m]$ ist nur für $m = 0$ konstant (siehe [5], III, Proposition 4.2).*

Abschließend folgt ein Beispiel zu elliptischen Kurven.

3.18 Beispiel: *In vereinfachter Weierstraß-Gleichung ist $E : y^2 = x^3 - 25x$ über $\mathbb{Q}$ eine elliptische Kurve. Wegen $\Delta(E) = 4(-25)^3 + 27(0)^2 \neq 0$ ist E glatt. Eine Computersuche zeigt, dass $(0, 0)$ und $(-\frac{5}{9}, \frac{100}{27})$ Punkte auf E sind.*

Auch $E' : y^2 = x^3 + 1$ über $\mathbb{Q}$ ist eine elliptische Kurve und $P = (2, 3) = (2 : 3 : 1) \in E'$. Mit 3.15 folgt $P + P = (0 : 1 : 1)$ und durch wiederholte Anwendung erstmalig $6P = O$, d.h. $\mathrm{ord}(P) = 6$.

Die Ordnung von Punkten auf einer elliptischen Kurve wird eine direkte Rolle im Test von Goldwasser-Kilian besitzen. Der nächste Schritt in Richtung des Satzes von Goldwasser-Kilian ist der Satz von Hasse, zu dessen Vorbereitung zunächst der Begriff der Isogenien eingeführt wird.

3.4 Isogenien

Ein weiteres Hilfsmittel für den Umgang mit elliptischen Kurven sind sogenannte Isogenien, zu denen einige Ergebnisse nun folgen.

3.19 Definition: *Seien (E, O_E), $(E', O_{E'})$ zwei elliptische Kurven. Eine Isogenie ist eine nicht-konstante rationale Abbildung $E \to E'$, die O_E auf $O_{E'}$ abbildet.*

Ist $\phi : E \to E'$ eine Isogenie und sind $k(E)$, $k(E')$ die Funktionenkörper zu E, E', so induziert ϕ eine Abbildung $\phi^* : k(E') \to k(E)$ durch $f \to f \circ \phi$.

Der Grad von ϕ ist dann $\deg(\phi) := [k(E) : \phi^* k(E')]$. *$\phi$ heißt separabel, falls die Körpererweiterung $k(E)/\phi^* k(E')$ separabel ist.*

3.20 Bemerkung: *Sei $\phi : E \to E'$ eine Isogenie vom Grad $\deg(\phi) = m \neq 0$. Dann existiert genau eine sogenannte duale Isogenie $\hat{\phi} : E' \to E$ mit*

$$\hat{\phi}\phi = [m] = \deg(\phi)$$

auf E (siehe [5], III, Theorem 6.1).

Endomorphismen $\phi \in \mathrm{End}_K(E)$ sind natürlich auch Isogenien.

Für das charakteristische Polynom von Endomorphismen folgen zwei Ergebnisse im nächsten Satz.

3.21 Theorem: *Sei E/K eine elliptische Kurve, $\phi \in \mathrm{End}_K(E)$ und $m = \deg(\phi)$. Dann gilt:*

1. *Es existiert ein $a_\phi \in \mathbb{Z}$, sodass $f_\phi(\phi) = 0$ für $f_\phi(T) = T^2 - a_\phi T + m$ gilt.*
2. *f_ϕ faktorisiert als $f_\phi(T) = (T - \alpha)(T - \bar{\alpha})$ mit $\alpha \in \mathbb{C}$, $|\alpha| = \sqrt{m}$.*

Beweis. Nach Definition der dualen Isogenie gilt $\deg(\phi) = \hat{\phi}\phi = m \in \mathbb{Z}$. Analog ist

$$\deg(1 - \phi) = \widehat{(1 - \phi)}(1 - \phi) = (1 - \hat{\phi})(1 - \phi) = 1 - (\hat{\phi} + \phi) + m \in \mathbb{Z}.$$

Wegen $\deg(1 - \phi) \in \mathbb{Z}$ und $m \in \mathbb{Z}$ folgt $\hat{\phi} + \phi \in \mathbb{Z}$. Mit $a_\phi := \hat{\phi} + \phi \in \mathbb{Z}$ folgt für $f_\phi(T) = T^2 - a_\phi T + m$ und $m = \deg(\phi) = \hat{\phi}\phi$ nach Ausmultiplizieren

$$f_\phi(\phi) = \phi^2 - (\hat{\phi} + \phi)\phi + \hat{\phi}\phi = 0.$$

Für den zweiten Teil ist zu zeigen, dass $f_\phi(T) = T^2 - a_\phi T + m$ eine Diskriminante $\Delta = a_\phi^2 - 4m < 0$ besitzt. Einsetzen von $\frac{b}{c} \in \mathbb{Q}$ in f_ϕ ergibt

$$f_\phi\left(\frac{b}{c}\right) = \frac{b^2}{c^2} - a_\phi \frac{b}{c} + m = \frac{1}{c^2}\left(b^2 - a_\phi bc + c^2 m\right) = \frac{1}{c^2}(b - c\hat{\phi})(b - c\phi) = \frac{1}{c^2}\deg(b - c\phi) > 0$$

für alle $\frac{b}{c} \in \mathbb{Q}$, da $c^2 > 0$ und der Grad positiv ist. Da $\mathbb{Q}$ dicht in $\mathbb{R}$ liegt, folgt $f_\phi(x) > 0$ $\forall x \in \mathbb{R}$ und damit $\Delta < 0$. f_ϕ faktorisiert daher als $f_\phi(T) = (T - \alpha)(T - \bar{\alpha})$ mit $\alpha \in \mathbb{C}$. Ein Koeffizientenvergleich des konstanten Terms von f_ϕ ergibt $m = \alpha\bar{\alpha} = |\alpha|^2$. □

3.5 Elliptische Kurven über $\mathbb{F}_p$ und der Satz von Hasse

Im folgenden werden elliptische Kurven über endlichen Körpern $\mathbb{F}_p$, p prim, betrachtet. Über die Anzahl der Punkte $(x, y) \in \mathbb{F}_p^2$ in $E(\mathbb{F}_p)$ sind zwei wichtige Ergebnisse bekannt. Zum einen kann $\#E(\mathbb{F}_p)$ nur in einem bestimmten Intervall, dem sogenannten Hasse-Intervall, liegen. Diese Abschätzung durch den Satz von Hasse ist die letzte Vorbereitung auf den Satz von Goldwasser-Kilian. Zum zweiten ist es möglich, die Anzahl der Punkte

für eine konkrete Kurve $E(a,b)$ über $\mathbb{F}_p$ exakt zu bestimmen. Eine effiziente Methode ist der Algorithmus von Schoof (siehe [13]), der als gegeben vorausgesetzt wird. Mit dem Schoof-Algorithmus wird die Anzahl der Punkte von $E(\mathbb{F}_p)$ modulo kleinen Primzahlen solange bestimmt, bis das Primzahlprodukt die obere Hasse-Schranke übersteigt und die genaue Anzahl anschließend per Chinesischem Restsatz ermittelt.

3.22 Definition: *Sei $E : y^2 = x^3 + ax + b$ eine elliptische Kurve, insbesondere ist $\Delta(E) \neq 0$. Falls $\Delta(E) \neq 0 \pmod p$ für eine Primzahl p ist, so definiert die reduzierte Kurve $\tilde{E} : y^2 = x^3 + ax + b \pmod p$ auch eine elliptische Kurve über $\mathbb{F}_p$ und man sagt, E hat „gute Reduktion“ bei p.*

3.23 Definition:

1. *Für eine elliptische Kurve $E : y^2 = x^3 + ax + b$ (genauso für E in generalisierter Weierstraß-Form) bezeichne $E^{(p)}$ diejenige Kurve, die aus E durch Potenzieren aller Koeffizienten zur p-ten Potenz hervorgeht.*
2. *Der Frobeniusmorphismus $F_p : E \to E^{(p)}$ ist definiert als*
$$F_p : (x, y) \to (x^p, y^p).$$

3.24 Bemerkung:

1. *Einsetzen der Koeffizienten von $E^{(p)}$ zeigt, dass*
$$\Delta(E^{(p)}) = 4\,(a^p)^3 + 27\,(b^p)^2 = 4\left(a^3\right)^p + 27\left(b^2\right)^p = \left(4a^3 + 27b^2\right)^p = \Delta(E)^p$$
in $\mathbb{F}_p$ gilt. Somit ist $E^{(p)}$ genau dann singulär über $\mathbb{F}_p$, wenn auch E singulär ist. Der Grad und damit das Geschlecht sowie der ausgezeichnete Punkt O bleiben unverändert. Damit ist $E^{(p)}$ ebenfalls eine elliptische Kurve.
2. *Nach dem kleinen Satz von Fermat ist $a^p = a$ in $\mathbb{F}_p$ für alle $a \in \mathbb{F}_p$. Ist E daher über $\mathbb{F}_p$ definiert, so ist der Frobeniusmorphismus die Identität.*
3. *Der Grad des Frobeniusmorphismus ist $\deg(F_p) = p$. Dies kann leicht mit Hilfe des pullback berechnet werden: Für $C := E$, $C' := E^{(p)}$, $\phi := F_p : C \to C'$ und einen Punkt $Q = (a, b) \in C'$ mit Bewertung v_Q gilt für den Verzweigungsindex*
$$e_Q = v_Q\phi^*(x - a^p) = v_Q(x^p - a^p) = v_Q\left((x - a)^p\right) = p \cdot v_Q(x - a) = p.$$

Es sei nun $K := \mathbb{F}_p$ endlicher Körper für eine Primzahl p. Elliptische Kurven werden über $\mathbb{P}^n_K$ betrachtet. Da K endlich ist, muss auch $\#E(K) \leq 2 \cdot \#K$ endlich sein.

3.25 Definition (Zeta-Funktion)**:** *Sei $E/\mathbb{F}_p$ eine elliptische Kurve und $\#E(\mathbb{F}_{p^n})$ die Anzahl der Punkte von E über $\mathbb{F}_{p^n}$, p prim, $n \in \mathbb{N}$. Die Zeta-Funktion ist definiert als*
$$Z_{E/\mathbb{F}_p}(T) := \exp\left(\sum_{n=1}^{\infty} \frac{\#E(\mathbb{F}_{p^n})}{n} \cdot T^n\right).$$

Der eigentliche Hauptsatz von Hasse über die Zeta-Funktion für elliptische Kurven und die daraus abgeleitete Abschätzung für die Anzahl der Punkte folgen schließlich aus den Ergebnissen über Isogenien.

3.26 Theorem (Hasse)**:** *Für eine elliptische Kurve $E/\mathbb{F}_p$ gilt*

$$Z_{E/\mathbb{F}_p}(T) = \frac{(1-\alpha T)(1-\bar{\alpha}T)}{(1-T)(1-pT)} = \frac{1-aT+pT^2}{(1-T)(1-pT)},$$

wobei $\alpha \in \mathbb{C}$ den Betrag $|\alpha| = \sqrt{p}$ besitzt und $a = p+1-\#E(\mathbb{F}_p)$ ist.

Beweis. Um die Definition der Zeta-Funktion anzuwenden, bleibt einzig die explizite Berechnung von $\#E(\mathbb{F}_{p^n})$.

Über $\mathbb{F}_p$ ist $E^{(p)} = E$ und damit der Frobeniusmorphismus $\phi := F_p : E \to E^{(p)}$ ein Endomorphismus, d.h. $\phi \in \mathrm{End}(E)$ und damit eine Isogenie.

Zudem ist aus Satz 3.21 bekannt, dass ϕ ein charakteristisches Polynom der Form $f_\phi(T) = T^2 - aT + p$ mit $a = \phi + \hat{\phi}$ (siehe Beweis von Satz 3.21) und $m = \deg(\phi) = \hat{\phi}\phi = p$ (siehe Bemerkung 3.24 über $\deg(F_p) = p$) besitzt. f_ϕ faktorisiert wegen negativer Diskriminante als

$$f_\phi(T) = T^2 - aT + p = (T-\alpha)(T-\bar{\alpha})$$

mit $\alpha \in \mathbb{C}$, $a = \alpha + \bar{\alpha}$ und $p = \alpha\bar{\alpha}$.

Punkte aus $E(\mathbb{F}_p)$ sind genau die Fixpunkte unter $\phi : E \to E$, daher ist

$$E(\mathbb{F}_p) = \ker(1-\phi).$$

Aus [5], Kapitel III 4.10, III 5.5 und Kapitel V 1.1 ist bekannt, dass $1-\phi$ separabel und damit

$$\#\ker(1-\phi) = \deg(1-\phi)$$

ist. Eingesetzt folgt daraus zusammen mit der Definition von $p = \deg(\phi) = \hat{\phi}\phi$ und $a = \phi + \hat{\phi}$ wie oben, dass

$$\#E(\mathbb{F}_p) = \#\ker(1-\phi) = \deg(1-\phi) = (1-\hat{\phi})(1-\phi) = 1-a+p = 1-\alpha-\bar{\alpha}+p.$$

Insbesondere ist $a = p+1-\#E(\mathbb{F}_p)$. Auf die gleiche Weise gilt auch für $n \in \mathbb{N}$, dass

$$\#E(\mathbb{F}_{p^n}) = \#\ker(1-\phi^n) = \deg(1-\phi^n) = (1-\hat{\phi^n})(1-\phi^n) = 1-\alpha^n-\bar{\alpha}^n+p^n.$$

Einsetzen von $\#E(\mathbb{F}_{p^n})$ in die Definition der Zeta-Funktion und Vereinfachen ergibt das gewünschte Ergebnis. □

Der Satz von Hasse erlaubt die exakte Berechnung der Ordnung einer elliptischen Kurve über $\mathbb{F}_{p^n}$ für beliebige $n \in \mathbb{N}$ und zudem eine Abschätzung für die Anzahl an Punkten $\#E(\mathbb{F}_p)$, wie das nächste Korollar zeigt.

3.27 Korollar:

1. *$\#E(\mathbb{F}_p)$ bestimmt $\#E(\mathbb{F}_{p^n})$ für alle $n > 1$.*
2. *Für die Anzahl der Punkte gilt die Abschätzung*

$$|\#E(\mathbb{F}_{p^n}) - p^n - 1| \leq 2\sqrt{p^n}.$$

Insbesondere gilt für $n = 1$, dass $|\#E(\mathbb{F}_p) - p - 1| \leq 2\sqrt{p}$. Dies bedeutet, dass die Ordnung von $E(\mathbb{F}_p)$ im sogenannten Hasse-Intervall

$$[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$$

liegt.

Beweis. $\#E(\mathbb{F}_p) = 1 - \alpha - \bar{\alpha} + p$ und $p = \alpha\bar{\alpha} = |\alpha|^2$ bestimmen $\alpha \in \mathbb{C}$ bei bekanntem p und $\#E(\mathbb{F}_p)$ eindeutig. Daraus lässt sich sofort die Ordnung von $E(\mathbb{F}_{p^n})$ per Formel $\#E(\mathbb{F}_{p^n}) = 1 - \alpha^n - \bar{\alpha}^n + p^n$ für alle $n \in \mathbb{N}$ berechnen.

Mit $|\alpha| = \sqrt{p}$ folgt aus $\#E(\mathbb{F}_{p^n}) = 1 - \alpha^n - \bar{\alpha}^n + p^n$ die Abschätzung

$$|\#E(\mathbb{F}_{p^n}) - p^n - 1| = |\alpha^n + \bar{\alpha}^n| \leq |\alpha|^n + |\bar{\alpha}|^n = 2\sqrt{p^n}.$$

□

Auch zum Satz von Hasse folgt ein abschließendes Beispiel.

3.28 Beispiel: *Betrachte die elliptische Kurve $E : y^2 = x^3 + 1$ aus Beispiel 3.18 nun über dem Körper $\mathbb{F}_5$, d.h. $p = 5$. Aus dem Kapitel über Isogenien ist bekannt, dass der Frobeniusendomorphismus $\phi = F_5$ ein charakteristisches Polynom $f_\phi(T) = T^2 - aT + p$ mit $a = p + 1 - \#E(\mathbb{F}_p)$ besitzt, wobei nach dem Satz von Hasse $\alpha \in \mathbb{C}$ durch $a = \alpha + \bar{\alpha}$ und $\alpha\bar{\alpha} = p$ bestimmt ist. Probieren aller Punkte in $\mathbb{F}_5$ liefert $\#E(\mathbb{F}_5) = 6$. Es folgt $a = 0$ und $\alpha = \sqrt{-5}$.*

Zum einen liegt $\#E(\mathbb{F}_5) = 6$ im Hasse-Intervall $\left[5 + 1 - 2\sqrt{5}, 5 + 1 + 2\sqrt{5}\right]$ und zum anderen folgt nach Einsetzen in die Formel $\#E(\mathbb{F}_{p^n}) = 1 - \alpha^n - \bar{\alpha}^n + p^n$, dass für $n = 3$ beispielsweise $\#E(\mathbb{F}_{5^3}) = 1 - (\sqrt{-5})^3 - (-\sqrt{-5})^3 + 5^3 = 126$ ist.

3.6 Der Satz von Goldwasser-Kilian

Sind $P, Q \in E(a, b)$ zwei Punkte auf einer elliptischen Kurve über $\mathbb{F}_p$, p prim, so können die Additionsformeln aus Satz 3.15 ohne Probleme auf den Körper $\mathbb{F}_p$ übertragen werden. Auch für $\mathbb{Z}/n\mathbb{Z}$ mit zusammengesetztem n ist die Addition definiert, sofern die Inverse im Nenner existieren. Sind $R, S \in E(\mathbb{Z}/n\mathbb{Z})$ zwei Punkte, für die $R + S$ wegen fehlender Inverse nicht existiert, so ist automatisch ein echter Faktor von n gefunden.

Sei E über $\mathbb{Z}/n\mathbb{Z}$ mit zusammengesetztem n gegeben, wobei $\Delta(E) \neq 0 \pmod{n}$. Weiter sei p ein Primfaktor von n.

3.29 Lemma: *Seien n zusammengesetzt, $p|n$ ein Primfaktor und $P, Q \in E(\mathbb{Z}/n\mathbb{Z})$. Ist $P+Q$ auf $E(\mathbb{Z}/n\mathbb{Z})$ definiert, so gilt $(R+S)_p = R_p + S_p$ in $E(\mathbb{F}_p)$.*

Beweis. Dies folgt leicht aus einer Fallunterscheidung bei Betrachtung der Additionsformel modulo p. Siehe Lemma 1 in [2] für eine ausführliche Version. □

Mit der Hasse-Abschätzung kann nun der Satz von Goldwasser-Kilian formuliert werden. Dieser ermöglicht eine Reduktion der Primalität von n auf die Primalität eines $q_1 < n$. Erneute Anwendung des Satzes auf die Folge der q_i liefert eine absteigende Rekursionsfolge (sogenannter *Downrun*),

$$n \text{ prim} \Leftarrow q_1 \text{ prim} \Leftarrow \ldots \Leftarrow q_l \text{ prim},$$

die dann abbricht, wenn ein q_l so klein ist, dass es mit einem weiteren Primzahlbeweis (zum Beispiel dem Lucas-Test, meist jedoch einfach per Probedivision) eindeutig als Primzahl identifiziert werden kann. Nach dem Satz von Goldwasser-Kilian überträgt die letzte Zahl q_l somit nacheinander die Primalität auf alle aufsteigenden q_i inklusive n selbst.

3.30 Theorem (Goldwasser-Kilian)**:** *Sei $n \in \mathbb{N}$ gegeben mit $\gcd(n, 6) = 1$. Falls eine elliptische Kurve $E(a, b)$ über $\mathbb{Z}/n\mathbb{Z}$ (insbesondere $\gcd(n, 4a^3+27b^2) = 1$) und ein Punkt $P \in E(\mathbb{Z}/n\mathbb{Z})$, $P \neq O$, existieren mit $[q]P = O$, q prim und $q > (\sqrt[4]{n}+1)^2$, dann ist n eine Primzahl.*

Beweis. Angenommen, n wäre nicht prim, dann gäbe es einen Primteiler $p|n$ mit $p \leq \sqrt{n}$.

Da $[q]P = O$ ist, folgt mit Lemma 3.29 über $\mathbb{F}_p$ sofort $[q]P_p = O_p$. Daher gilt $\text{ord}(P_p)|q$ und wegen q prim kann die Ordnung von P_p in $E(\mathbb{F}_p)$ nur 1 oder q sein. Nach Voraussetzung ist aber $P \neq O$ und daher auch $P_p \neq O_p$. (Denn $P \neq O$ in $E(\mathbb{Z}/n\mathbb{Z})$ bedeutet, dass insbesondere die letzte Koordinate von P teilerfremd zu n ist. Somit ist diese auch teilerfremd zu p, insbesondere von 0 verschieden und daher $P_p \neq O_p$.) Das zeigt $\text{ord}(P_p) = q$ in $E(\mathbb{F}_p)$.

Mit $q > (\sqrt[4]{n}+1)^2$ und $p \leq \sqrt{n}$ sowie der Abschätzung von Hasse folgt

$$\#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p} < \sqrt{n} + 1 + 2\sqrt[4]{n} = (\sqrt[4]{n}+1)^2 < q.$$

Dies ist ein Widerspruch, denn $\text{ord}(P_p) = q$ teilt die Ordnung $\#E(\mathbb{F}_p) < q$. □

Dieser Satz liefert sofort einen Algorithmus für ein gegebenes $n \in \mathbb{N}$. Es wird vorausgesetzt, dass $\gcd(n, 6) = 1$ wie in Satz 3.30 ist und dass n bereits einen probabilistischen Primzahltest bestanden hat.

Goldwasser-Kilian

1. Wähle eine elliptische Kurve $E(a,b)$ über $\mathbb{Z}/n\mathbb{Z}$ durch zufällige $a,b \in \mathbb{Z}/n\mathbb{Z}$ mit $\gcd(n, 4a^3+27b^2) = 1$. Tritt ein nicht-trivialer gcd auf, so ist n nicht prim.

2. Berechne $m := \#E(a,b)$ über $\mathbb{Z}/n\mathbb{Z}$ mit dem Algorithmus von Schoof unter der Annahme, dass n prim ist. Falls Schoof einen Fehler meldet oder m nicht im Hasse-Intervall liegt, so ist n nicht prim.

3. Teste m auf Gestalt $m = kq$ mit $q > (\sqrt[4]{n}+1)^2$ Pseudoprimzahl. Dauert die Faktorisierung zu lange oder hat m nicht die benötigte Gestalt, so wiederhole ab Schritt 1.

4. Wähle einen zufälligen Punkt $P_E \in E(\mathbb{Z}/n\mathbb{Z})$. (Wähle dazu zufällige $x \in \mathbb{Z}/n\mathbb{Z}$ und teste jeweils, ob $Z := x^3+ax+b$ das Legendre-Symbol $\left(\frac{Z}{n}\right) \neq -1$ besitzt. Ermittle dann y aus $y^2 = Z$ per Algorithmus von Shanks-Tonelli. Falls dabei ein Fehler auftritt oder am Ende $y^2 \not\equiv Z \pmod{n}$ ist, so ist n nicht prim.)

5. Berechne $U := [m/q]P_E$ für $P_E = (x,y)$. Ist $U = O$, so wiederhole ab Schritt 4. Ist $[q]U \neq O$ oder treten undefinierte Additionen in $E(\mathbb{Z}/n\mathbb{Z})$ auf, so ist n nicht prim.

6. Ausgabe ist das Zertifikat (n,a,b,m,q,U). Ist q prim, so auch n.

3.31 Bemerkung (Korrektheit)**:** *Die wesentlichen Stellen des Algorithmus sind die Bestimmung von q und eines Punktes auf E mit Ordnung q. Da $q|\#E(a,b)$ wählt man den umgekehrten Weg und ermittelt zuerst $m = \#E(a,b)$ per Schoof-Algorithmus sowie q danach als Primfaktor von m. Schritt 5 des Algorithmus stellt sicher, dass $U = [m/q]P_E$ die Ordnung q besitzt und $U \neq O$ ist. Somit nimmt U die Rolle von P in Satz 3.30 ein.*

Die ermittelte Kurve E mit Punkt U und $q > (\sqrt[4]{n}+1)^2$ erfüllen genau den Satz von Goldwasser-Kilian und beweisen die Primalität von n sofern auch q prim ist. Um die Primalität von $q < n$ zu beweisen, kann Goldwasser-Kilian erneut aufgerufen werden und liefert Zertifikate

$$(q_0 = n, a_0, b_0, m_0, q_1, U_0),\ (q_1, a_1, b_1, m_1, q_2, U_1),\ \ldots$$

Ist nach genügender Rekursion ein q_j so klein, dass dessen Primalität direkt bewiesen werden kann (meist per Probedivison), so impliziert dies die Primalität von q_{j-1} per Satz 3.30 und durch wiederholte Anwendung von Satz 3.30 rückwirkend für alle q_i, $0 \leq i < j$, bis hin zu $n = q_0$. Stellt sich beim Abstieg der q_i heraus, dass ein q_k nicht prim ist, so muss der Downrun für q_{k-1} neu gestartet werden.

3.32 Bemerkung (Vorteil)**:** *Das Zertifikat von Goldwasser-Kilian besitzt einen klaren Vorteil gegenüber dem von Pratt: obwohl bei beiden Algorithmen faktorisiert werden muss ($m = kq$ bei Goldwasser-Kilian und $n-1 = p_1 \cdots p_m$ bei Pratt), ist die Faktorisierung*

von $n-1$ beim deterministischen Pratt-Zertifikat zwingend erforderlich. Ist $n-1$ schwer faktorisierbar, so kann Pratt nicht fortgeführt werden. Lässt sich hingegen die Ordnung m der Kurve E bei Goldwasser-Kilian in Schritt 3 nicht faktorisieren, so geht man einfach zurück zu Schritt 1 und wählt solange neue Kurven, bis eine der Kurvenordnungen leicht faktorisierbar ist. Zudem arbeitet Goldwasser-Kilian nur mit einer linearen Rekursion für das q, während Pratt m rekursive Aufrufe für alle p_i aus $n-1 = p_1 \cdots p_m$ benötigt.

3.33 Bemerkung (Probabilistischer Primzahltest)**:** *Es ist völlig legitim in Schritt 3 einen probabilistischen Primzahltest zu benutzen, obwohl Goldwasser-Kilian mit vollständigem Zertifikat am Ende ein Primzahlbeweis ist. Denn äquivalent könnte man in Schritt 3 direkt einen rekursiven Aufruf für q starten, sobald man annimmt, dass $m = kq$ mit q prim ist. Der probabilistische Test für q wird nur verwendet, um ungeeignete Kandidaten m ohne Struktur $m = kq$ auszusortieren. Erst mit dem rekursiven Beweis von q prim ist die Suche nach einer geeigneten Ordnung m abgeschlossen. Stellt sich heraus, dass q in Schritt 3 nicht prim ist, so hat m nicht die geforderte Gestalt $m = kq$ und es muss ab Schritt 1 wiederholt werden. In der Formulierung oben wird die Berechnung unter der Annahme q prim weitergeführt und die Primalität von q erst danach bewiesen.*

3.34 Bemerkung (Originalalgorithmus)**:** *Der Originalalgorithmus von Goldwasser und Kilian (siehe [1]) sucht nur nach geraden Ordnungen $m = 2q$ mit $q > (\sqrt[4]{n}+1)^2$ Pseudoprimzahl. Eine Verallgemeinerung auf $m = kq$ ist aber ohne Probleme möglich, da in Satz 3.30 nur q eine Rolle spielt. Beide Varianten haben die gleiche asymptotische Laufzeit (siehe Herleitung der polynomiellen Laufzeit im nächsten Abschnitt 3.7).*

Vor der Laufzeitabschätzung folgt ein beispielhafter Primzahlbeweis per Goldwasser-Kilian, der mit dem Programm zur Arbeit (siehe Anhang) berechnet wurde.

3.35 Beispiel: *Die Primalität von $n = 10^{20} + 39$ soll per Goldwasser-Kilian bewiesen werden. Eine Suche nach zufälligen $a, b \in \mathbb{Z}/n\mathbb{Z}$ liefert zwei Kurvenparameter*

$$a = 21201453409687609344,\ b = 38818188629438832640,$$

die eine elliptische Kurve $E(a,b)$ über $\mathbb{Z}/n\mathbb{Z}$ definieren. Die Ordnung von $E(a,b)$ wird per Algorithmus von Schoof zu

$$m = 99999999997868912271 = 3 \cdot 11 \cdot 3030303030238451887$$

bestimmt, das leicht in die Gestalt $m = kq$ mit Anteil $k = 33$ und einer Pseudoprimzahl $q = 3030303030238451887 > (\sqrt[4]{n}+1)^2$ faktorisiert werden kann. Der Zufallspunkt

$$P_E = (6323098999013255168 : 78509094751890658929 : 1)$$

auf E erfüllt $U := [m/q]P_E \neq O$ und $[q]U = O$. Somit erfüllen E, q und $U = [m/q]P_E$ den Satz von Goldwasser-Kilian unter der Annahme, dass q prim ist. Rekursive Anwendung des Algorithmus (bzw. Satz 3.30) auf $q < n$ zeigt schließlich, dass q in der Tat eine Primzahl ist. Nach Goldwasser-Kilian ist mit q auch n prim.

3.7 Herleitung der polynomiellen Laufzeit

Bisher unterscheiden sich das Pratt-Zertifikat und das Goldwasser-Kilian-Zertifikat nur durch eine verschiedene Berechnungsweise. Unter einer plausiblen Annahme kann jedoch das Goldwasser-Kilian-Zertifikat im Gegensatz zum Pratt-Zertifikat in erwarteter polynomieller Zeit berechnet werden. Dies wird nun hergeleitet.

Der Algorithmus von Goldwasser-Kilian terminiert in erwarteter polynomieller Zeit für alle Eingaben, wenn die folgende Vermutung wahr ist.

3.36 Vermutung: *Es existieren Konstanten $c_1, c_2 > 0$, sodass*

$$\pi(x + \sqrt{x}) - \pi(x) \geq \frac{c_2\sqrt{x}}{\log^{c_1} x}$$

für alle genügend großen x gilt.

Diese Vermutung ist ein Spezialfall der allgemeineren Vermutung von Cramér (siehe [26]), die besagt, dass der Abstand von n-ter und $(n+1)$-ter Primzahl in der Ordnung $p_{n+1} - p_n = O(\log^2 p_n)$ liegt. Die Cramér-Vermutung impliziert Vermutung 3.36 mit $c_1 = 2$. Dies sieht man wie folgt: Die Anzahl an Primzahlen $\pi(x+\sqrt{x})-\pi(x)$ im Intervall $[x, x+\sqrt{x}]$ lässt sich nach unten abschätzen durch die Intervalllänge $\sqrt{x}$ geteilt durch den Primzahlabstand $O(\log^2 x)$. Mit Cramér ist $\pi(x + \sqrt{x}) - \pi(x) = \frac{\sqrt{x}}{O(\log^2 x)} = o\left(\frac{\sqrt{x}}{\log^2 x}\right)$. In o-Notation bedeutet dies $\pi(x + \sqrt{x}) - \pi(x) \geq \frac{c_2\sqrt{x}}{\log^{c_1} x}$ für ein $c_2 > 0$ und $c_1 = 2$.

Da die Vermutung von Cramér für plausibel gehalten wird, spricht dies für die Wahrheit von Vermutung 3.36.

3.37 Bemerkung (Geometrische Verteilung)**:** *Tritt ein Ereignis mit Wahrscheinlichkeit p ein und wird auf das erste Eintreffen im n-ten Schritt gewartet, so entspricht dies dem sogenannten „Warten auf Erfolg“, beschrieben durch die geometrische Verteilung $\mathbb{P}(X = n) = p(1-p)^{n-1}$. Der Erwartungswert von X ist $\mathbb{E}(X) = \frac{1}{p}$. Ist p konstant, so muss eine erwartet konstante Anzahl $\mathbb{E}(X) = \frac{1}{p}$ an Versuchen bis zum ersten Ereignis gewartet werden.*

Als nächstes werden alle Schritte des Goldwasser-Kilian-Algorithmus einzeln auf ihre erwartete Laufzeit analysiert. Die erwartete Gesamtlaufzeit ist dann die Summe der Laufzeiten aller Schritte multipliziert mit der Anzahl an Wiederholungen des *Downrun*, d.h. mit der Anzahl an Primzahlen, die für das Zertifikat erzeugt werden.

Es sei k die Anzahl an Bits von n.

Alle Variablen außer m, die im Algorithmus auftreten, sind aus $\mathbb{Z}/n\mathbb{Z}$ und daher $O(k)$ Bits lang. Wie später noch gezeigt wird, ist auch $\log(m) = O(k)$.

Die folgende Laufzeitabschätzung orientiert sich an dem Originalalgorithmus von Goldwasser und Kilian, der nach Ordnungen $m = 2q$ sucht (siehe Bemerkung 3.34). Die Erweiterung zu $m = kq$ hat gleiche erwartete Laufzeit, wenn die Faktorisierung bis zu einer gewissen Schranke $k \in [2, \ldots, k_{max}]$ in polynomieller Zeit abläuft. Dies ist möglich, falls beispielsweise das Produkt der ersten 1000 Primzahlen gespeichert ist und alle kleinen Primfaktoren von m durch eine einzige ggT-Berechnung abgespalten werden.

Schritt 1

Bei der Wahl zufälliger $a, b \in \mathbb{Z}/n\mathbb{Z}$ mit $\gcd(n, 4a^3 + 27b^2) = 1$ wird im ungünstigen Fall entweder ein Faktor gefunden und der Algorithmus bricht ab, oder $b = \pm\sqrt{-4a^3/27} \pmod{n}$ und es muss ein neues Paar gewählt werden. Letzteres tritt in $\frac{2}{n}$ der Fälle ein. Dies ist weniger als $1/2$ für $n > 5$, somit ist die Erfolgswahrscheinlichkeit mindestens $1/2$ und es muss nur eine erwartet konstante Anzahl (also $O(1)$) an Parameterpaaren probiert werden. Die Berechnung von $4a^3 + 27b^2$ für Zahlen mit $\log(a),\ \log(b) \in O(k)$ dauert $O(k^2)$ wegen der Multiplikationen und nochmals $O(k^3)$ für die ggT-Berechnung.

Schritt 2

Die Berechnung der Ordnung $m = \#E(a, b)$ per Algorithmus von Schoof dauert $O(k^9)$ mit dem Originalalgorithmus und $O(k^6)$ mit der Variante Schoof-Elkies-Atkin. Wie auch in der Originalarbeit [1] wird hier $O(k^9)$ angenommen.

Schritt 3

Der Test von m auf Gestalt $m = 2q$ dauert $O(1)$ für die Abspaltung des Faktors 2 und $O(k^4)$ für einen probabilistischen Primzahltest von q (beispielsweise per Miller-Rabin). Dabei ist $\log(m) = O(k)$, wie folgende Überlegung zeigt: die Ordnung m ist durch die obere Intervallgrenze $n + 1 + 2\sqrt{n}$ nach dem Satz von Hasse beschränkt. Ist $\log(n) = O(k)$, so ist $\log(\sqrt{n}) = O(k/2)$. Da $\sqrt{n}$ nur halbe Bitlänge von n besitzt, folgt daraus die Größenordnung $\log(m) = O(k) + O(1) + O(k/2) = O(k)$.

Schritte 1-3

Die Laufzeit für die ersten drei Schritte wird von $O(k^9)$ dominiert, multipliziert mit der erwarteten Anzahl T an Kurven, die probiert werden müssen, bevor eine mit geeigneter Ordnung $m = 2q$ gefunden wird: insgesamt $T \cdot O(k^9)$. Die Größenordnung von T wird am Ende bestimmt.

Schritt 4

Nach erfolgreichem Schritt 3 mit rekursivem Primzahlbeweis für q wird jetzt $m = 2q$ mit q prim vorausgesetzt. Die zufällige Wahl von x in Schritt 4 laufe in $O(1)$, die Berechnung von $Z = x^3 + ax + b$ kostet $O(k^2)$ und die Berechnung des Legendre-Symbols inklusive der Lösung $y^2 \equiv Z$ per Shanks-Tonelli (siehe [8], Algorithmus 2.3.8) insgesamt $O(k^4)$. Da die Hälfte der Elemente $x \in \mathbb{Z}/n\mathbb{Z}$ quadratische Reste sind, ist die Erfolgswahrscheinlichkeit auf ein x demnach $1/2$. Daher müssen erwartet nur zwei x-Werte getestet werden (d.h. $O(1)$). Die Laufzeit für Schritt 4 ist daher $O(k^4)$.

Schritt 5

Die Addition zweier Punkte auf der elliptischen Kurve dauert $O(k^2)$ wegen der Multiplikationen. Mit wiederholten Verdoppelungen benötigen dann die Berechnungen von $U = [m/q]P_E$ und $[q]U$ nochmals $O(k)$ Additionen (d.h. $O(k^3)$). Der Test auf $U = O$ bzw. $[q]U \neq O$ läuft in $O(1)$. Dies ergibt insgesamt $O(k^3)$.

Schritte 4-5

Schritte 4 und 5 (dominiert von $O(k^4)$) müssen solange wiederholt werden, bis ein Punkt $U = [m/q]P_E$ der Ordnung q gefunden ist. $E(a, b)$ über $\mathbb{Z}/n\mathbb{Z}$ ist eine endliche abelsche Gruppe und somit Produkt von zyklischen additiven Gruppen $(\mathbb{Z}/m_1\mathbb{Z}) \times (\mathbb{Z}/m_2\mathbb{Z})$ mit $m_1|m_2$. Wegen $\#E(a, b) = m = 2q$ und $m_1|m_2$ ist $m_1 = 1$ und $m_2 = 2q$. Somit folgt $E(a, b) \cong \mathbb{Z}/(2q)\mathbb{Z}$. Es ist bekannt, dass $\mathbb{Z}/(2q)\mathbb{Z}$ insgesamt $q - 1$ Punkte der Ordnung q besitzt, und so auch $E(a, b)$. Man beachte, dass diese Punkte gepaart auftreten, denn mit $[q](x, y) = O$ gilt auch für dessen Inverse $[q](x, -y) = [q](-(x, y)) = [-q](x, y) = -([q](x, y)) = O$. Die Wahl eines zufälligen x und einer festen Wurzel $y = \sqrt{x^3 + ax + b}$ liefert daher im günstigen Fall einen der $(q-1)/2$ Punkte der Ordnung q. Mit bekannter Anzahl $m = 2q$ an Punkten ist die Erfolgswahrscheinlichkeit also $((q-1)/2)/(2q) = O(1)$ und daher die erwartete Laufzeit zur Suche eines Punktes der Ordnung q konstant. Schritte 4 und 5 laufen also in erwarteter Zeit $O(k^4)$.

Schritte 1-6

Die erwartete Laufzeit $T \cdot O(k^9)$ der ersten drei Schritte dominiert eindeutig die der letzten drei Schritte (die Schritte 4-5 haben eine Laufzeit von $O(k^4)$, die Ausgabe des Zertifikats in Schritt 6 läuft in $O(1)$). Daher ist die erwartete Laufzeit pro einem *Downrun* ebenfalls $T \cdot O(k^9)$. T ist die erwartete Anzahl an Kurven, die durchprobiert werden muss, um eine mit Ordnung $m = 2q$ zu finden. Die Größenordnung von T wird nun bestimmt.

Bestimmung von T

Zur Bestimmung von T muss bekannt sein, wie viele elliptische Kurven $E(a, b)$ über $\mathbb{Z}/n\mathbb{Z}$ mit Ordnung $m = 2q$ (Primzahl q) im Hasse-Intervall $[n + 1 - 2\sqrt{n}, n + 1 + 2\sqrt{n}]$ existieren. Zur Abschätzung wird ein Theorem von Lenstra (siehe [2], Seite 458, oder auch [21]) über die Verteilung von Kurvenordnungen in Intervallen benutzt.

3.38 Theorem (Lenstra)**:** *Sei $p > 5$ prim und*

$$S \subseteq [p + 1 - \lfloor\sqrt{p}\rfloor, p + 1 + \lfloor\sqrt{p}\rfloor] .$$

Wird eine elliptische Kurve $E(a, b)$ über $\mathbb{F}_p$ gleichverteilt gewählt, so ist die Wahrscheinlichkeit für $\#E(a, b) \in S$ mindestens

$$\mathbb{P}(\#E(a, b) \in S) > \frac{c}{\log p} \cdot \frac{|S| - 2}{2\lfloor\sqrt{p}\rfloor + 1},$$

wobei c eine feste Konstante ist (Beweis: Siehe [21]).

3.39 Bemerkung: *Um das Theorem von Lenstra anzuwenden, muss S definiert werden. Dazu wird die Frage nach der Anzahl der Kurvenordnungen modifiziert: Die Anzahl der Ordnungen $m = 2q$ im Intervall $[n+1-2\sqrt{n}, n+1+2\sqrt{n}]$ ist gleich der Anzahl der Primzahlen im Intervall $[(n+1-2\sqrt{n})/2, (n+1+2\sqrt{n})/2]$.*

Es fällt auf, dass in Satz 3.38 im Gegensatz zum Hasse-Intervall kein Faktor 2 vor der Wurzel steht. Um Lenstra anzuwenden, wird daher nun angenommen, dass Kurvenordnungen nur im kleineren Intervall $[n+1-\lfloor\sqrt{n}\rfloor, n+1+\lfloor\sqrt{n}\rfloor]$ auftreten. Wird bereits im kleineren Intervall eine genügende Anzahl an Ordnungen gefunden (d.h. in der Größenordnung $O(k)$), so auch im größeren.

Setze $x := (n+1-\lfloor\sqrt{n}\rfloor)/2$, $y := (n+1+\lfloor\sqrt{n}\rfloor)/2$ und definiere

$$S := \{q \in [x,y] : \ q \text{ prim}\}.$$

Die Kardinalität von S gibt die Anzahl der Primzahlen im Intervall

$$[x,y] = \left[(n+1-\lfloor\sqrt{n}\rfloor)/2, (n+1+\lfloor\sqrt{n}\rfloor)/2\right]$$

an, die gleichzeitig auch die Anzahl der Ordnungen $m = 2q$ im abgeschwächten Intervall

$$\left[n+1-\lfloor\sqrt{n}\rfloor, n+1+\lfloor\sqrt{n}\rfloor\right] \subseteq \left[n+1-2\sqrt{n}, n+1+2\sqrt{n}\right]$$

(enthalten im Hasse-Intervall) ist.

Nun muss noch die Kardinalität von S bestimmt werden. Dazu hilft die Vermutung 3.36 von Goldwasser-Kilian. Per Induktion für $n > 37$ lässt sich leicht zeigen, dass $y > x + \sqrt{x}$ ist. Mit Vermutung 3.36 ist

$$\pi(y) - \pi(x) > \pi(x+\sqrt{x}) - \pi(x) = o\left(\frac{\sqrt{x}}{\log^{c_1} x}\right).$$

Die weiteren Ergebnisse sollen direkt wieder in der Bitlänge k von n ausgedrückt werden. $\log(n)$ ist von der Ordnung $O(k)$, $\log(\sqrt{n}) = O(k/2)$ und somit auch $\log(x) = O(k)$, $\log(\sqrt{x}) = O(k/2)$. Daher werden nun $\sqrt{n}$ und $\sqrt{x}$ als $\exp(k/2)$ in der o-Notation angegeben. Da $|S|$ die Anzahl an Primzahlen im Intervall $[x,y]$ beschreibt, ist

$$|S| = o\left(\frac{\sqrt{x}}{\log^{c_1} x}\right) = o\left(\frac{\exp(k/2)}{k^{c_1}}\right).$$

Mit dem Theorem von Lenstra beträgt damit die Wahrscheinlichkeit auf eine Kurvenordnung $m = 2q$ mit Primzahl q über $\mathbb{Z}/n\mathbb{Z}$:

$$\mathbb{P}\left(\#E(a,b) = 2q\right) > \frac{c}{\log n} \cdot \frac{|S|-2}{2\lfloor\sqrt{n}\rfloor + 1} = o\left(\frac{1}{k} \cdot \frac{\exp(k/2)}{k^{c_1} \cdot \exp(k/2)}\right) = o\left(\frac{1}{k^{c_1+1}}\right).$$

Dies zeigt, dass die Wahrscheinlichkeit für eine Kurve der Ordnung $m = 2q$ bei $o\left(\frac{1}{k^{c_1+1}}\right)$ liegt. Der Erwartungswert der geometrischen Verteilung ist reziprok zur Wahrscheinlichkeit und ergibt die erwartete Anzahl T an Kurven, die getestet werden müssen, bis eine der Ordung $m = 2q$ gefunden wird. Somit ist $T = O(k^{c_1+1})$.

Laufzeit eines *Downrun*-Schrittes

Nach vorheriger Betrachtung ist die Laufzeit der Schritte 1-6 des Algorithmus $T \cdot O(k^9)$. Mit $T = O(k^{c_1+1})$ beträgt die erwartete Laufzeit eines *Downrun*-Schrittes $O(k^{10+c_1})$ zur Reduktion des Primalitätproblems von n auf ein kleineres q.

Laufzeit aller *Downrun*-Schritte

Die erwartete Gesamtlaufzeit für den Primalitätsbeweis von n ist die erwartete Laufzeit pro *Downrun*-Schritt multipliziert mit der Anzahl an *Downrun*-Schritten, d.h. der Anzahl an Primzahlen q_i, die für das Zertifikat erzeugt werden.

Dazu wird lediglich die maximale Anzahl an Primzahlen q_i im Zertifikat abgeschätzt. Diese Anzahl ist maximal, wenn beim Beweis eines q_i das jeweils nächste q_{i+1} größtmöglich ist. Für jedes q_i liegt die Ordnung m der elliptischen Kurve, die im Beweis des q_i benutzt wird, im Hasse-Intervall

$$[q_i + 1 - 2\sqrt{q_i}, q_i + 1 + 2\sqrt{q_i}]$$

und bestimmt q_{i+1} per $m = 2q_{i+1}$. Wegen $m \leq q_i + 1 + 2\sqrt{q_i}$ ist q_{i+1} maximal

$$q_{i+1} \leq \frac{q_i + 1 + 2\sqrt{q_i}}{2}.$$

Da $\sqrt{q_i}$ nur halbe Bitlänge von q_i besitzt, ist $q_{i+1} = q_i/2 + o(q_i)$. Das zeigt, dass $O(\log n) = O(k)$ Reduktionsschritte nötig sind, bis die Primalität von $n = q_0$ auf ein trivial zu testendes q_l reduziert ist. Für die Suche nach Ordnungen $m = kq$ mit $k \geq 2$ trifft diese Betrachtung natürlich weiterhin zu.

Da die q_i im *Downrun* kleiner als n sind, kann die erwartete Laufzeit jedes Reduktionsschrittes einfach mit dem Aufwand $O(k^{10+c_1})$ für n selbst abgeschätzt werden. Mit $O(k)$ *Downrun*-Schritten beweisen die vorherigen Überlegungen den folgenden Satz.

3.40 Theorem (Laufzeit)**:** *Der vollständige Primalitätsbeweis von n per Goldwasser-Kilian besitzt eine erwartete Gesamtlaufzeit von $O(k^{11+c_1})$ (vgl. [14], C.9.10, Seite 269).*

In manchen Referenzen wird eine erwartete Laufzeit von $O(k^{12})$ angegeben. Diese Abschätzung beruht auf der *zusätzlichen* Annahme, dass der Primzahlsatz $\pi(x) \approx \frac{x}{\log x}$ für genügend große x auch in kleinen Intervallen gilt. Wird nämlich angenommen, dass der Primzahlsatz auch auf Vermutung 3.36 zutrifft, so folgt für genügend große x

$$\pi(x + \sqrt{x}) - \pi(x) \approx \frac{x + \sqrt{x}}{\log(x + \sqrt{x})} - \frac{x}{\log x} \approx \frac{x + \sqrt{x}}{\log x} - \frac{x}{\log x} = \frac{\sqrt{x}}{\log x},$$

wobei $\log(x+\sqrt{x}) \approx \log x$, da $\sqrt{x}$ nur halb so viele Stellen wie x besitzt und daher $x+\sqrt{x}$ und x beide $O(k)$ Bits lang sind. Ein Vergleich mit Vermutung 3.36 motiviert die Wahl $c_1 = 1$ und daher die Gesamtlaufzeit $O(k^{12})$.

3.41 Bemerkung: *Die Originalarbeit [1] von Goldwasser und Kilian enthält eine weitere Abschätzung. Verzichtet man auf die unbewiesene Vermutung 3.36, so lässt sich zeigen, dass der Algorithmus in erwarteter Laufzeit $O(k^{12})$ für mindestens $1 - O\left(2^{-k^{\frac{1}{\log\log k}}}\right)$ aller Eingaben terminiert (siehe Theorem 3 in [1]).*

3.42 Bemerkung: *Abschließend soll noch bemerkt werden, dass Vermutung 3.36 nur die Laufzeitanalyse betrifft. Die Korrektheit des Algorithmus beruht auf dem bewiesenen Satz 3.30 von Goldwasser-Kilian. Kann also ein Zertifikat für n berechnet werden, so ist n mit Sicherheit prim. Damit ist der Goldwasser-Kilian Test ein sogenannter Las-Vegas Algorithmus (immer korrekt, probabilistische Laufzeit).*

3.8 Verifikation eines Zertifikats

Ist ein Zertifikat der Form $(q_i, a_i, b_i, m_i, q_{i+1}, U_i)$ für $n = q_0$ und *alle* weiteren Primzahlen q_i im *Downrun* erstellt, so kann die Primalität von n schneller als in $O(k^{12})$ verifiziert werden. Pro Reduktionsschritt müssen nämlich nur die Voraussetzungen des Satzes von Goldwasser-Kilian geprüft werden. Diese sind pro q_i:

1. Es gilt $\gcd(q_i, 6) = 1$ und $\gcd(q_i, 4a_i^3 + 27b_i^2) = 1$. Dies definiert die elliptische Kurve $E(a_i, b_i)$ über $\mathbb{Z}/q_i\mathbb{Z}$.
2. Der Punkt U_i liegt auf $E(a_i, b_i)$ und erfüllt $U_i \neq O$, $[q_i]U_i = O$.
3. $q_{i+1} > (\sqrt[4]{q_i} + 1)^2$.

Zur Laufzeit: Punkt 1 erfordert Multiplikationen in $O(k^2)$ und ggT-Berechnungen in $O(k^3)$. Zur Probe auf $U_i \in E(a_i, b_i)$ in Punkt 2 muss getestet werden, ob $U_i = (x_i, y_i)$ die Gleichung $E_i : y^2 = x^3 + a_i x + b_i$ erfüllt. Dies dauert $O(k^2)$. Der Test auf $U_i \neq O$ läuft in $O(1)$ und auf $[q_i]U_i = O$ in $O(k^3)$ mit Hilfe von wiederholten Verdoppelungen. Punkt 3 läuft in $O(k^2 \log k)$ unter Verwendung einer Newton-Iteration zur Wurzelberechnung (siehe Algorithmus 9.2.11 in [8]).

Die Laufzeit zur Verifikation eines Reduktionsschrittes ist damit $O(k^3)$. Die Länge des Zertifikates hat die Größenordnung $O(k)$, da nach dem vorherigem Abschnitt $O(k)$ Primzahlen für den *Downrun* erzeugt werden. Damit ist gezeigt:

3.43 Theorem: *Die Gesamtlaufzeit zur Verifikation des Primalitätsbeweises ist $O(k^4)$.*

4 Die Verbesserung von Atkin-Morain zu ECPP

An der Laufzeitabschätzung 3.7 aus dem letzten Kapitel wurde bereits ein großer Nachteil des Algorithmus von Goldwasser-Kilian sichtbar: der verwendete Algorithmus von Schoof besitzt mit $O(k^9)$ (k ist die Anzahl an Bits von n) bzw. $O(k^6)$ für die Variante Schoof-Elkies-Atkin den größten Anteil der Laufzeit. Zudem ist es ineffizient zuerst zufällige Kurven zu raten, deren Ordnung m aufwändig per Schoof zu zählen und erst danach m zu analysieren. In vielen Fällen wird die Kurve nämlich wegen falscher Gestalt von m wieder verworfen.

Genau dieser Punkt wird durch die Verbesserung von Atkin-Morain zu ECPP („Elliptic Curve Primality Proving“) mit Hilfe der Theorie der „Komplexen Multiplikation“ oder kurz CM („Complex Multiplication“) behoben: diese Variante erlaubt es, schnell Kurven mit bekannten Ordnungen zu erzeugen, und zwar ohne dass die Punktanzahl per Schoof gezählt werden muss. Damit kann in ECPP auf den Algorithmus von Schoof komplett verzichtet werden. Zusätzlich ermöglicht CM, die Kurvenordnung m bereits vor der Kurve zu berechnen. Damit können Ordnungen durchprobiert werden, ohne eine einzige explizite Kurve zu den Ordnungen zu kennen. Erst wenn eine passende Ordnung mit Struktur $m = kq$ gefunden wurde, wird die elliptische Kurve zu m berechnet.

Die Erweiterung von Atkin-Morain bezieht sich im Wesentlichen nur auf die Punkte 1 und 2 des Algorithmus von Goldwasser-Kilian aus dem Kapitel 3.6, d.h. auf das Finden der elliptischen Kurve und die Berechnung der Ordnung m per Algorithmus von Schoof. Der Rest des Algorithmus bleibt bestehen. Insbesondere beruht die Korrektheit von ECPP weiterhin auf dem Satz von Goldwasser-Kilian und ECPP liefert das gleiche Zertifikat.

4.1 Die Weierstraß $\wp$-Funktion

Für die Methode der Komplexen Multiplikation werden elliptische Kurven aus dem letzten Kapitel über $\mathbb{C}$ betrachtet. Dabei stellt sich ein interessanter Zusammenhang zu einer bestimmten Funktion, der Weierstraß $\wp$-Funktion, heraus.

4.1 Definition (Gitter)**:** *Ein Gitter (lattice) $L \subseteq \mathbb{C}$ ist eine diskrete Untergruppe vom Rang 2, $L := \mathbb{Z}\, w_1 + \mathbb{Z}\, w_2$. Dabei sind $w_1, w_2 \in \mathbb{C}$ linear unabhängig über $\mathbb{R}$. Es bezeichne $L := [w_1, w_2]$ das von w_1, w_2 erzeugte Gitter.*

4.2 Definition (Weierstraß $\wp$-Funktion)**:** *Über einem Gitter L ist die Weierstraß $\wp$-Funktion definiert als*

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

Aus [17] (Kapitel 3.8 und 3.9) ist bekannt, dass die Weierstraß $\wp(z, L)$-Funktion meromorph in $\mathbb{C}$ ist und eine doppelte Polstelle an jedem Gitterpunkt $w \in L$ besitzt. Die $\wp$-Funktion ist ein wichtiges Beispiel einer sogenannten elliptischen Funktion (d.h. definiert über $\mathbb{C}$ und doppelt periodisch) und erfüllt die folgende Differentialgleichung.

4.3 Theorem (Differentialgleichung)**:** *Sei* $G_{2k}(L) := \sum_{0 \neq w \in L} w^{-2k}$ *für* $k \geq 2$. *Setze* $g_2 := g_2(L) := 60G_4(L)$ *und* $g_3 := g_3(L) := 140G_6(L)$. *Dann erfüllt* $\wp(z)$ *die Differentialgleichung*

$$\wp'(z)^2 = 4\wp(z)^3 - g_2\wp(z) - g_3.$$

Beweis. Berechnen der Ableitung, Ausschreiben aller Terme und die Subtraktion der linken von der rechten Seite ergibt eine elliptische, holomorphe Funktion mit Funktionswert 0 an $z = 0$, die nach dem Satz von Liouville somit konstant 0 ist (siehe auch [5], VI, Theorem 3.5). □

Wichtige Erkenntnis über die $\wp$-Funktion ist nun, dass jede elliptische Kurve über einem Gitter L in $\wp$ und $\wp'$ ausgedrückt werden kann:

4.4 Theorem:

$$\begin{array}{rcl} \mathbb{C}/L & \stackrel{\cong}{\to} & E_L \\ z & \to & (\wp(z) : \wp'(z) : 1). \end{array}$$

Beweis. Siehe [5], VI, Korollar 5.1.1. □

Es folgt ein Beispiel zu dieser Isomorphie.

4.5 Beispiel (Siehe [10], Aufgabe 24)**:** *Sei* $E : y^2 = x^3 - x = (x-0)(x+1)(x-1)$ *über* $\mathbb{C}$ *gegeben. Nach [5], VI, Proposition 5.2, kann die Basis* w_1, w_2 *zur elliptischen Kurve* E *aus Integralen der Form* $\int_{P_1}^{P_2} dx/y \pmod{L}$ *gewonnen werden. Hier ist* $y = \sqrt{x^3 - x}$.

Eine Berechnung zeigt $w_1 = \int_0^1 \frac{dx}{\sqrt{x^3-x}} = (-i)w_2$, *wobei* $w_2 = \int_1^\infty \frac{dx}{\sqrt{x^3-x}}$. *Nach Definition 4.1 und Satz 4.4 kommt* E *von einem Gitter* $L = \mathbb{Z}\, w_1 + \mathbb{Z}\, w_2 = w_1(\mathbb{Z} + i\mathbb{Z})$. *Damit ist* $E \cong \mathbb{C}/w_1(\mathbb{Z} + i\mathbb{Z}) = \mathbb{C}/(\mathbb{Z} + i\mathbb{Z})$ *wegen* $w_1 \in \mathbb{C}$.

Umgekehrt kann aus $L := \mathbb{Z} + i\mathbb{Z}$ *die elliptische Kurve* E *per Basis* $\wp(z, L)$, $\wp'(z, L)$ *rekonstruiert werden.*

Ähnlich zur Definition der j-Invariante einer elliptischen Kurve in Definition 3.12 motiviert dies auch die Definition der j-Invariante für das Gitter L.

4.6 Definition (j-Invariante)**:** *Assoziiert mit dem Gitter L ist dessen j-Invariante*

$$j(L) := 1728\frac{g_2(L)^3}{g_2(L)^3 - 27g_3(L)^2} = 1728\frac{g_2(L)^2}{\Delta(L)},$$

wobei $\Delta(L) := g_2(L)^3 - 27g_3(L)^2$. Sei $\tau \in \mathbb{C}$ gegeben und sei $L_\tau := [1, \tau]$ das dazugehörige Gitter. Dann bezeichne

$$j(\tau) := j(L_\tau) := j([1, \tau]) := 1728\frac{g_2(L_\tau)^2}{\Delta(L_\tau)}.$$

4.7 Bemerkung (Ähnlichkeit)**:** *Man beachte die ähnliche Gestalt der Differentialgleichung in Theorem 4.3 und der Gleichung einer elliptischen Kurve $E : y^2 = x^3 + ax + b$:*

In der Tat folgt nach Division der Differentialgleichung in Theorem 4.3 durch 4, dass $\left(\frac{\wp'(z)}{2}\right)^2 = \wp(z)^3 - \frac{g_2}{4}\wp(z) - \frac{g_3}{4}$. Eine Substitution $y := \wp'(z)/2$, $x := \wp(z)$, $a := -g_2/4$ und $b := -g_3/4$ liefert schließlich wegen $\Delta(E) = -16(4a^3 + 27b^2) = g_2^3 - 27g_3^2 = \Delta(L) \neq 0$ eine elliptische Kurve mit vereinfachter Weierstraß-Gleichung $y^2 = x^3 + ax + b$ und der j-Invariante $j(E) = j(L)$.

4.2 Quadratische Formen

4.8 Definition (Quadratische Form)**:**

1. *Eine ganzzahlige quadratische Form $F := [a, b, c]$ ist ein homogenes quadratisches Polynom $f(x, y) := ax^2 + bxy + cy^2$ mit $a, b, c \in \mathbb{Z}$.*
2. *Die Diskriminante $D(F)$ einer quadratischen Form ist als $D(F) := b^2 - 4ac$ definiert.*
3. *Sei die Matrix $M_F := \begin{pmatrix} a & b/2 \\ b/2 & c \end{pmatrix}$ einer Form $F = [a, b, c]$ zugeordnet. Dann werden die quadratischen Formen F und F' äquivalent genannt ($F \sim F'$), falls eine Matrix $A \in SL_2(\mathbb{Z}) = \{M \in \mathbb{Z}^{2\times 2} : \det(M) = 1\}$ existiert mit $M_{F'} = A^{-1}M_F A$.*
4. *$F = [a, b, c]$ heißt primitiv, falls $\gcd(a, b, c) = 1$. F heißt reduziert, falls zudem gilt: $|b| \leq a \leq c$ sowie $b \geq 0$ wenn $|b| = a$ oder $a = c$.*

Die j-Invariante aus dem letzten Abschnitt und eine reduzierte quadratische Form F sowie dessen Diskriminante $D(F)$ hängen wie folgt zusammen.

4.9 Bemerkung: *Sei $F(x, y) = ax^2 + bxy + cy^2$ eine reduzierte quadratische Form mit Diskriminante D. Da F reduziert ist, ist $D < 0$. Es bezeichne daher τ die Nullstelle von $F(x, 1)$ in der oberen Halbebene, d.h.*

$$\tau = \frac{-b + \sqrt{D}}{2a}.$$

Definiere dann $j([a,b,c]) := j(L_\tau) = j(\tau) = j\left(\frac{-b+\sqrt{D}}{2a}\right)$. *Die* j*-Invariante der quadratischen Form ist somit die* j*-Invariante des durch die Nullstelle* $\frac{-b+\sqrt{D}}{2a}$ *aufgespannten Gitters.*

Die Menge der reduzierten quadratischen Formen zur Diskrimante D bildet eine Gruppe $Cl(D)$ (Klassengruppe). Deren Ordnung wird mit $h(D)$ (Klassenzahl) bezeichnet.

4.3 Quadratische Zahlkörper

Aus der algebraischen Zahlentheorie (siehe Kapitel 16 in [9]) sind folgende Definitionen und Zusammenhänge bekannt:

4.10 Definition (Quadratischer Zahlkörper)**:** *Ein Zahlkörper* K *ist ein* $\mathbb{Q} \subseteq K \subset \mathbb{C}$ *mit endlicher Dimension* $[K:\mathbb{Q}]$ *als* $\mathbb{Q}$*-Vektorraum. Ein Zahlkörper der Form* $K = \mathbb{Q}(\sqrt{d})$, $d \in \mathbb{Z}$ *kein Quadrat, wird quadratischer Zahlkörper genannt. Der Ring der ganzen Zahlen* $\mathcal{O}_K$ *ist der Ring aller Zahlen aus* K*, die eine Ganzheitsgleichung über* $\mathbb{Z}$ *erfüllen.*

4.11 Bemerkung:

1. *Für* $K = \mathbb{Q}(\sqrt{d})$ *existieren zwei Körperautomorphismen, nämlich* $id : K \to K$ *und* $\sigma : K \to K$, $a + b\sqrt{d} \to a - b\sqrt{d}$.
2. *Für* $x = a + b\sqrt{d} \in K$ *definiert man dessen Norm als* $N(x) := x \cdot \sigma(x)$ *und dessen Spur als* $Sp(x) := x + \sigma(x)$. *Eine einfache Rechnung zeigt* $Sp(x) = 2a \in \mathbb{Z}$ *und* $N(x) = a^2 - db^2 \in \mathbb{Z}$.
3. *Für* $x \in K \setminus \mathbb{Q}$ *hat das Minimalpolynom offensichtlich nicht Grad 1. Einsetzen zeigt, dass* x *jedoch* $x^2 - Sp(x) + N(x) = 0$ *genügt, das wegen* $N(x),\ Sp(x) \in \mathbb{Z}$ *dessen Minimalpolynom sein muss.*

Der Ring $\mathcal{O}_K$ für $K = \mathbb{Q}(\sqrt{d})$ kann leicht charakterisiert werden (siehe [9], Satz 16.20 und dessen Beweis):

4.12 Theorem: *Ist* $K = \mathbb{Q}(\sqrt{d})$, $d \in \mathbb{Z}$ *quadratfrei, so gilt für den Ring der ganzen Zahlen*

$$\mathcal{O}_K = \begin{cases} \mathbb{Z}\left[\sqrt{d}\right] & d \equiv 2,3 \pmod 4 \\ \mathbb{Z}\left[\frac{1+\sqrt{d}}{2}\right] & d \equiv 1 \pmod 4. \end{cases}$$

Beweis. Als Nullstelle des Polynoms $x^2 - d \in \mathbb{Z}[x]$ ist $\sqrt{d}$ immer ganz. Auch für $d \equiv 1 \pmod 4$ ist $(d-1)/4 \in \mathbb{Z}$ und $w := \frac{1+\sqrt{d}}{2}$ daher als Nullstelle von $x^2 - x - (d-1)/4 \in \mathbb{Z}[x]$ ganz. Da die ganzen Zahlen einen Ring bilden, folgt $\mathbb{Z}[\sqrt{d}] \subseteq \mathcal{O}_K$ bzw. $\mathbb{Z}[w] \subseteq \mathcal{O}_K$.

Sei umgekehrt $x = a + b\sqrt{d}$ mit $a, b \in \mathbb{Q}$ ganz, d.h. Spur $Sp(x) = 2a \in \mathbb{Z}$ und Norm $N(x) = a^2 - db^2 \in \mathbb{Z}$. Wegen $2a \in \mathbb{Z}$ folgt aus $4N(x) = (2a)^2 - d(2b)^2 \in \mathbb{Z}$ auch $d(2b)^2 \in \mathbb{Z}$. d quadratfrei impliziert dann $2b \in \mathbb{Z}$. Ist zudem $a \in \mathbb{Z}$, so impliziert $N(x) \in \mathbb{Z}$ auch $db^2 \in \mathbb{Z}$ und somit $b \in \mathbb{Z}$ für quadratfreies d.

Ist $a \notin \mathbb{Z}$, so bleibt wegen $2a \in \mathbb{Z}$ nur der Fall $a = c/2$ mit ungeradem $c \in \mathbb{Z}$ übrig. Einsetzen ergibt $N(x) = (c^2 - d(2b)^2)/4 \in \mathbb{Z}$, d.h. $c^2 - d(2b)^2 \equiv 0 \pmod 4$. Wegen $c \equiv 1 \pmod 2$ ist $1 \equiv c^2 \equiv d(2b)^2 \pmod 4$. Quadrate können mod 4 nur 0 oder 1 sein, daher ist $(2b)^2 \equiv 1 \pmod 4$ und $2b \equiv 1 \pmod 2$. Das zeigt, dass auch b von der Form $b = e/2$ mit ungeradem $e \in \mathbb{Z}$ ist ($e \in \mathbb{Z}$ wegen $2b \in \mathbb{Z}$). Einsetzen von $(2b)^2 \equiv 1 \pmod 4$ in $1 \equiv d(2b)^2 \equiv d \pmod 4$ zeigt $d \equiv 1 \pmod 4$. □

4.13 Bemerkung: *Alle Elemente aus $\mathcal{O}_K$ für $K = \mathbb{Q}(\sqrt{d})$, $d \in \mathbb{Z}$ quadratfrei, lassen sich mit $a, b \in \mathbb{Z}$ als*

$$\frac{a + b\sqrt{d}}{2}$$

schreiben. Für $d \equiv 2, 3 \pmod 4$ ist dies mit geraden a, b klar und für $d \equiv 1 \pmod 4$ folgt für $x \in \mathbb{Z}\left[\frac{1+\sqrt{d}}{2}\right]$, dass

$$x = r \cdot 1 + s \cdot \frac{1 + \sqrt{d}}{2} = \frac{(2r + s) + s\sqrt{d}}{2}.$$

4.14 Definition: *Die Diskriminante eines Zahlkörpers $K = \mathbb{Q}(\sqrt{d})$, $d \in \mathbb{Z}$ quadratfrei, zur Basis $B = (b_1, b_2)$ ist definiert als*

$$\Delta(B) := \det \begin{pmatrix} b_1 & \sigma(b_1) \\ b_2 & \sigma(b_2) \end{pmatrix}^2,$$

wobei id und σ die Körperautomorphismen aus Bemerkung 4.11 sind.

4.4 Die Fundamentaldiskriminante

Die Methode der Komplexen Multiplikation arbeitet mit solchen elliptischen Kurven, die $\mathrm{End}_E(K) = \mathcal{O}_K$ für einen imaginär-quadratischen Zahlkörper $K = \mathbb{Q}(\sqrt{D})$ und $D < 0$ erfüllen. Dazu wird zunächst die sogenannte Fundamentaldiskriminante eingeführt.

4.15 Definition (Fundamentaldiskriminante)**:** *Eine negative ganze Zahl $D \in \mathbb{Z}_{<0}$ wird Fundamentaldiskriminante genannt, falls D entweder $D \equiv 1 \pmod 4$ und quadratfrei ist, oder $D \equiv 0 \pmod 4$, $D/4$ quadratfrei und $D/4 \equiv 2, 3 \pmod 4$ erfüllt.*

Eine alternative Definition nach [8] ist:

4.16 Definition (Fundamentaldiskriminante)**:** *Eine negative ganze Zahl $D \in \mathbb{Z}_{<0}$ wird Fundamentaldiskriminante genannt, falls der ungerade Teil d von $D = 2^e \cdot d$ quadratfrei ist und $|D| \equiv 3, 4, 7, 8, 11$ oder $15 \pmod{16}$.*

Wird die Fundamentaldiskriminante D nach Definition 4.15 nun in d geschrieben als

$$D = \begin{cases} 4d & \text{für } D \equiv 0 \pmod 4 \text{ mit } d = D/4 \equiv 2, 3 \pmod 4 \\ d & \text{für } D \equiv 1 \pmod 4 \text{ mit } d = D, \end{cases}$$

so stimmt D nach dem folgenden Satz (siehe Lemma 16.30 in [9]) mit der Diskriminante Δ von $K = \mathbb{Q}(\sqrt{d})$ (Basis von $\mathcal{O}_K$ aus Theorem 4.12) überein.

4.17 Theorem: *Für einen Zahlkörper $K = \mathbb{Q}(\sqrt{d})$, $d \in \mathbb{Z}$ quadratfrei, ist die Diskriminante*

$$\Delta = \begin{cases} 4d & d \equiv 2, 3 \pmod 4 \\ d & d \equiv 1 \pmod 4. \end{cases}$$

Beweis. Für den Fall $d \equiv 2, 3 \pmod 4$ ist $\mathcal{O}_K = \mathbb{Z}[\sqrt{d}]$ nach Theorem 4.12 mit Basis $(1, \sqrt{d})$ und Automorphismen id, σ aus 4.11. Einsetzen in die Definition 4.14 der Diskriminante liefert

$$\Delta = \det \begin{pmatrix} 1 & 1 \\ \sqrt{d} & -\sqrt{d} \end{pmatrix}^2 = 4d,$$

sowie für $d \equiv 1 \pmod 4$, $\mathcal{O}_K = \mathbb{Z}[\frac{1+\sqrt{d}}{2}]$ mit Basis $(1, \frac{1+\sqrt{d}}{2})$ analog

$$\Delta = \det \begin{pmatrix} 1 & 1 \\ \frac{1+\sqrt{d}}{2} & \frac{1-\sqrt{d}}{2} \end{pmatrix}^2 = d.$$

□

Die Diskriminante $\Delta = D$ liefert eine einheitliche Darstellung von K und $\mathcal{O}_K$ (siehe Korollar 16.31 in [9]).

4.18 Theorem: *Für $K = \mathbb{Q}[\sqrt{d}]$ mit Diskriminante $\Delta = D$ ist*

$$K = \mathbb{Q}[\sqrt{D}] \quad \text{und} \quad \mathcal{O}_K = \mathbb{Z}\left[\frac{D + \sqrt{D}}{2}\right].$$

Beweis. Wegen $\sqrt{D} = \sqrt{4d} = 2\sqrt{d}$ ist $K = \mathbb{Q}[\sqrt{d}] = \mathbb{Q}[\sqrt{D}]$. Zudem gilt

$$\frac{D + \sqrt{D}}{2} = \begin{cases} \frac{4d+\sqrt{4d}}{2} = 2d + \sqrt{d} & d \equiv 2, 3 \pmod 4 \\ \frac{d+\sqrt{d}}{2} = \frac{d-1}{2} + \frac{1+\sqrt{d}}{2} & d \equiv 1 \pmod 4. \end{cases}$$

Wegen $2d \in \mathbb{Z}$, $\frac{d-1}{2} \in \mathbb{Z}$ entspricht dies $\mathbb{Z}\left[\sqrt{d}\right]$ bzw. $\mathbb{Z}\left[\frac{1+\sqrt{d}}{2}\right]$ aus Theorem 4.12. □

Sei nun eine Fundamentaldiskriminante D gegeben, $K = \mathbb{Q}(\sqrt{D})$ und $\mathcal{O}_K = \mathbb{Z}\left[\frac{D+\sqrt{D}}{2}\right]$. Für die Einheitengruppe $\mathcal{O}_K^\times$ gilt dann der folgende Satz (siehe [9], Satz 17.3).

4.19 Theorem (Einheiten)**:**

1. *Für $D = -3$, $K = \mathbb{Q}(\sqrt{-3})$, ist $\mathcal{O}_K^\times = \{\pm 1, \pm w, \pm w^2\}$ mit $w = e^{2\pi i/3} = \frac{-1+\sqrt{-3}}{2}$.*
2. *Für $D = -4$, $K = \mathbb{Q}(\sqrt{-4})$, ist $\mathcal{O}_K^\times = \{\pm 1, \pm i\}$.*
3. *Für alle $D < -4$ ist $\mathcal{O}_K^\times = \{\pm 1\}$.*

Beweis. Einheiten in $\mathcal{O}_K$ sind alle $x \in \mathcal{O}_K$ mit Norm $N(x) = \pm 1$ (siehe [9], Lemma 17.1). Schreibe $x = a + b\sqrt{D}$ mit $a, b \in \frac{1}{2}\mathbb{Z}$ nach Satz 4.12 und Bemerkung 4.13. Der Fall $N(x) = a^2 - Db^2 < 0$ ist wegen $D < 0$ nicht möglich. Gesucht sind daher Lösungen von $1 = a^2 + (-D)b^2$.

Zwei sofortige Lösungen sind $(a, b) = (\pm 1, 0)$ für alle D. Da $-D > 0$ ist, gibt es für $|a| > 1$ keine Lösungen. Für $a = 0$ vereinfacht sich das Problem zu $1 = (-D)b^2$, wobei 1 und b^2 Quadrate sind. Allerdings ist $D = -4$ die einzige Fundamentaldiskriminante, die selbst ein Quadrat ist, und es folgt $b = \pm\frac{1}{2}$, d.h. $x = 0 \pm \frac{1}{2}\sqrt{-4} = \pm i$.

Es bleibt der Fall $D \equiv 1 \pmod 4$. Mit $A := 2a \in \mathbb{Z}$, $B := 2b \in \mathbb{Z}$ werden Lösungen von $4 = A^2 + (-D)B^2$ gesucht. Ist $-D \geq 5$, so folgt sofort $A = \pm 2$ und $B = 0$, d.h. erneut $x = \pm 1$. Ist $-D < 5$, so folgt $D = -3$ wegen $D \equiv 1 \pmod 4$, d.h. $4 = A^2 + 3B^2$. Der Fall $B = 0$ lieferte $x = \pm 1$, nun liefert $B = \pm 1$ sofort $A = \pm 1$. Für $|B| \geq 2$ gibt es offensichtlich keine Lösung mehr. Zusammengefasst sind $\{\pm 1, \frac{\pm 1 \pm \sqrt{-3}}{2}\}$ Lösungen für den Fall $D = -3$, die genau der Menge $\{\pm 1, \pm w, \pm w^2\}$ mit $w = \frac{-1+\sqrt{-3}}{2}$ entsprechen. □

Die Einheiten werden wichtig sein, sobald zu einer vorgegebenen Ordnung die elliptische Kurve aus mehreren verfügbaren Möglichkeiten gewählt werden soll.

4.5 Das Hilbert'sche Klassenpolynom

Nach der Einführung von quadratischen Zahlkörpern, deren Diskriminanten und Ganzheitsringen, soll nun wieder zur Herleitung der Komplexen Multiplikation zurückgekehrt werden. Dazu zeigt der folgende Satz (siehe Theorem 3.1 aus [3]) die Verbindung der Fundamentaldiskriminante D und der j-Invariante aus dem ersten Abschnitt.

4.20 Theorem (Das Hilbert'sche Klassenpolynom)**:** *Sei $K = \mathbb{Q}(\sqrt{D})$ ein quadratischer Zahlkörper und D eine Fundamentaldiskriminante.*

Der Hilbert'sche Klassenkörper von K (die maximale unverzweigte abelsche Erweiterung von K) ist gegeben durch Adjunktion einer beliebigen j-Invariante $j([a, b, c])$ an K, wobei $[a, b, c] \in Cl(D)$ eine beliebige reduzierte quadratische Form zur Diskriminante D ist.

Das Minimalpolynom $H_D(X)$ der $j([a,b,c])$'s zur Diskriminante D ist aus $\mathbb{Z}[X]$ und kann dargestellt werden als

$$H_D(X) = \prod_{[a,b,c] \in Cl(D)} \left(X - j\left(\frac{-b + \sqrt{D}}{2a} \right) \right) \in \mathbb{Z}[X].$$

4.21 Bemerkung: *Die explizite Berechnung des Klassenpolynoms $H_D(X)$ ist sehr aufwändig und wird in einigen Arbeiten separat behandelt. Siehe beispielsweise [3], [7] oder [18] für gängige Methoden. In dieser Arbeit, besonders in der Implementierung, wird das Klassenpolynom als gegeben betrachtet.*

Zum Schluss dieses Abschnitts wird als Vorbereitung auf die Komplexe Multiplikation das Verhalten von Primzahlen im Klassenkörper H betrachtet (siehe [3], Theoreme 2.3 und 3.3).

4.22 Theorem: *Sei $K = \mathbb{Q}(\sqrt{D})$, D eine Fundamentaldiskriminante und H der Hilbert'sche Klassenkörper von K. Dann sind für eine Primzahl p äquivalent:*

1. *p ist eine Norm in K.*
2. *(p) spaltet vollständig in H.*
3. *p spaltet in zwei verschiedene Elemente in $\mathcal{O}_K$.*
4. *$H_D(X) \pmod p$ faktorisiert vollständig in lineare Faktoren mit Nullstellen in $\mathbb{F}_p$.*
5. *$4p = u^2 + |D|v^2$ hat eine Lösung mit $u, v \in \mathbb{N}$.*

4.23 Bemerkung: *Der Satz hat für ECPP eine entscheidende Bedeutung: Anhand der letzten Bedingung kann leicht getestet werden, ob p in $\mathcal{O}_K$ als Produkt zweier Elemente zerfällt (und zwar die Form $p = \alpha\bar{\alpha}$ hat), denn Elemente aus $\mathcal{O}_K$ haben nach Satz 4.12 und Bemerkung 4.13 die Gestalt $\alpha = \frac{u+v\sqrt{D}}{2}$ mit $u, v \in \mathbb{Z}$. Somit gilt*

$$p = \alpha\bar{\alpha} = \frac{u^2 - Dv^2}{4} = \frac{u^2 + |D|v^2}{4} \;\Leftrightarrow\; 4p = u^2 + |D|v^2$$

wegen $D < 0$. Für den Test auf $4p = u^2 + |D|v^2$ existiert ein effizienter Algorithmus von Cornacchia (siehe [8], Algorithmus 2.3.13). Die genaue Vorgehensweise wird im Abschnitt über ECPP behandelt.

4.6 Komplexe Multiplikation

In Kapitel 3.5 wurde eine elliptische Kurve E über $\mathbb{F}_p$ betrachtet. Genauso kann E auch über einem Zahlkörper K mit einem Primideal $\mathfrak{p}$ über p betrachtet werden. Dieser Abschnitt orientiert sich an [7].

Sei dazu $E : y^2 = x^3 + ax + b$ mit $a, b \in K$ gegeben und $\mathbb{F}_{\mathfrak{p}} = \mathcal{O}_K/\mathfrak{p}$. Es bezeichne $\tilde{a}, \tilde{b}$ die Bilder von a und b in $\mathbb{F}_{\mathfrak{p}}$. Falls $\widetilde{\Delta} := -16(4\tilde{a}^2 + 27\tilde{b}^3) \neq 0$ ist, so definiert $\widetilde{E} : y^2 = x^3 + \tilde{a}x + \tilde{b}$ eine elliptische Kurve über $\mathbb{F}_{\mathfrak{p}}$. Analog zu Definition 3.22 in Kapitel 3.5 wird $\mathfrak{p}$ dann Primzahl mit guter Reduktion genannt.

Der folgende Satz von Deuring (siehe [19], Paragraph 5, Theorem 14) zeigt eine Verbindung zwischen K und $\mathbb{F}_p$.

4.24 Theorem (Deuring Lift)**:** *Sei eine elliptische Kurve E über $\mathbb{F}_p$ definiert und habe einen nicht-trivialen Endomorphismus ϕ. Dann existiert eine elliptische Kurve E' über einem Zahlkörper K, ein Endomorphismus ϕ' von E' und ein $\mathfrak{p} \in \mathcal{O}_K$ über p mit guter Reduktion, sodass E zu $\widetilde{E'} := E' \pmod{\mathfrak{p}}$ isomorph ist und unter dem Isomorphismus $\widetilde{\phi'}$ mit ϕ übereinstimmt.*

Der Ausdruck „Komplexe Multiplikation“ bezieht sich auf den Endomorphismenring $\mathrm{End}(E)$ der elliptischen Kurve. Aus Kapitel 3.3 über das Gruppengesetz von elliptischen Kurven ist bekannt, dass für alle $n \in \mathbb{Z}$ eine Multiplikationsabbildung $[n] : E \to E$ auf E existiert (siehe Definition 3.16). Diese wurde einfach durch n-fache Wiederholung der Addition $[n]P = P + \ldots + P$ für $n \geq 0$ und $[n]P = (-P) + \ldots + (-P)$ für $n < 0$ definiert. Da die Multiplikationsabbildung ein Endomorphismus für alle $n \in \mathbb{Z}$ ist, erhält man sofort eine Injektion $\mathbb{Z} \hookrightarrow \mathrm{End}(E)$. Für die meisten Kurven gibt es keine weiteren Endomorphismen, d.h. $\mathrm{End}(E) \cong \mathbb{Z}$. Ist aber $\mathrm{End}(E) = \mathcal{O}_E \supsetneq \mathbb{Z}$, so sagt man, dass E komplexe Multiplikation mit $\mathcal{O}_E$ besitzt.

4.25 Definition: *E hat komplexe Multiplikation (mit $\mathcal{O}_E$), falls* $\mathrm{End}(E) = \mathcal{O}_E \supsetneq \mathbb{Z}$.

Sei nun E eine elliptische Kurve über $\mathbb{C}$ und $K = \mathbb{Q}(\sqrt{D})$ ein quadratischer Zahlkörper mit Ganzheitsring $\mathcal{O}_K =: \mathcal{O}_D$. Weiter habe E komplexe Multiplikation mit $\mathcal{O}_D$, d.h. $\mathrm{End}_{\mathbb{C}}(E) \cong \mathcal{O}_D$. Da der Frobeniusmorphismus $F_p \in \mathrm{End}_{\mathbb{F}_p}(\tilde{E})$ (siehe Definition 3.23) ein Endomorphismus von E über $\mathbb{F}_p$ ist, existiert nach dem Satz 4.24 von Deuring eine Kurve E' und ein $\pi \in \mathrm{End}_{\mathbb{C}}(E)$, dass modulo p mit F_p übereinstimmt.

Da F_p und π unter dem Isomorphismus aus Satz 4.24 übereinstimmen, ist insbesondere auch deren Grad erhalten. Dabei ist $\deg(F_p) = p$ nach Bemerkung 3.24, d.h. auch $\deg(\pi) = p$. Der Grad eines Elementes $\pi \in \mathbb{C}$ ist einfach dessen Norm, somit folgt $N(\pi) = \pi\bar{\pi} = \deg(\pi) = p$ in $\mathcal{O}_D$.

Analog zur Vorgehensweise im Beweis von Satz 3.26 von Hasse ist

$$\#E(\mathbb{F}_p) = \#\ker(1 - F_p) = \deg(1 - F_p) = \deg(1 - \pi) = (1 - \pi)(1 - \bar{\pi}) = 1 + p - t,$$

wobei $t := \pi + \bar{\pi}$ gesetzt wurde, $p = \pi\bar{\pi}$ aus der obigen Normbetrachtung folgt und $\deg(1 - F_p) = \deg(1 - \pi)$ wegen Erhaltung des Grades gilt.

Diese Betrachtung ist noch nicht vollständig. Die nächste Bemerkung ist essentiell für ECPP.

4.26 Bemerkung: *Bei der vorherigen Vorgehensweise gibt es noch ein Problem. Denn ist $u \in \mathcal{O}_K$ eine Einheit ($N(u) = 1$), so gilt zwar ebenfalls $N(u\pi) = N(u)N(\pi) = p$, aber im allgemeinen $u\pi + \bar{u}\bar{\pi} \neq \pi + \bar{\pi}$. Je nach Einheit erhält man demnach einen verschiedenen Wert für $\#E(\mathbb{F}_p)$.*

Für alle $D < -4$ wurde im Satz 4.19 gezeigt, dass $\mathcal{O}_K^\times = \{\pm 1\}$ ist. Daher kommen für $D < -4$ nur $u = 1$ und $u = -1$ in Frage. Für die zwei Sonderfälle $D = -3$ und $D = -4$ mit 6 bzw. 4 Einheiten ist die Situation schwieriger. Da aber die Methode der Komplexen Multiplikation für alle $D < -4$ angewendet werden kann, soll der Einfachheit halber auf $D = -3$ und $D = -4$ verzichtet werden. Der ECPP-Algorithmus kommt problemlos ohne die beiden Sonderfälle aus, wenn von Anfang an nur Diskriminanten $D < -4$ betrachtet werden.

Ab diesem Punkt gelte für eine Fundamentaldiskriminante D immer $D < -4$. Siehe [8], Kapitel 7.5.3 für die Behandlung der Sonderfälle.

Seien $p = N(\pi)$ und $t := \pi + \bar{\pi}$ wie oben. Die Einheiten sind $u \in \mathcal{O}_K^\times = \{\pm 1\}$, daher kommen für $u\pi$ nur π oder $-\pi$ in Frage. Je nachdem, ob π oder $-\pi$ gewählt wird, ist dann $\pi + \bar{\pi} = t$ oder $-\pi - \bar{\pi} = -t$. In beiden Fällen bleibt aber $p = N(\pi) = N(-\pi)$. Die beiden Ordnungen, die man demnach erhält, sind

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

mit $t := \pi + \bar{\pi}$.

Zusammengefasst wird das obige Ergebnis im nächsten Satz.

4.27 Theorem: *Sei E eine elliptische Kurve über $\mathbb{C}$ und $K = \mathbb{Q}(\sqrt{D})$ ein quadratischer Zahlkörper mit Fundamentaldiskriminante $D < -4$ und Ganzheitsring $\mathcal{O}_K =: \mathcal{O}_D$. Weiter habe E komplexe Multiplikation mit $\mathcal{O}_D$. Ist p eine Primzahl, die über $\mathcal{O}_D$ in $p = \pi\bar{\pi}$ zerfällt, so ist*

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

mit $t := \pi + \bar{\pi}$.

4.28 Bemerkung: *Für eine elliptische Kurve $E/\mathbb{F}_p$ kann gezeigt werden, dass der Frobeniusendomorphismus $F_p \notin \mathbb{Z}$ ist (d.h. $F_p \neq [n]\ \forall n \in \mathbb{Z}$). $\mathrm{End}(E) \supseteq \mathbb{Z}$ gilt immer wegen der Multiplikationsabbildung (siehe Erklärung vor Definition 4.25). Wegen $F_p \notin \mathbb{Z}$ folgt somit $\mathrm{End}(E) \supsetneq \mathbb{Z}$. Daher hat E über $\mathbb{F}_p$ immer komplexe Multiplikation.*

4.7 Elliptic Curve Primality Proving

Vor der Beschreibung des eigentlichen Kerns von ECPP soll die Aufgabenstellung wiederholt werden:

Der Algorithmus von Goldwasser-Kilian aus Kapitel 3 liefert einen Primalitätsbeweis in erwarteter Zeit $O(k^{12})$, bei dem ineffizient Kurven geraten und deren Punkte per Schoof-Algorithmus in $O(k^9)$ gezählt werden. Um diese Vorgehensweise zu verbessern, sollen elliptische Kurven mit bekannter Ordnung konstruiert werden. Dadurch kann auf den Schoof-Algorithmus verzichtet und die Laufzeit verringert werden. Abgesehen von der neuen Konstruktion elliptischer Kurven mit bekannter Ordnung ist der restliche Goldwasser-Kilian Algorithmus unverändert. Die folgende Methode nach Atkin und Morain erlaubt es zudem, die möglichen zwei Ordnung $\#E(\mathbb{F}_p) = p + 1 \pm t$ (siehe zuvor) vor der konkreten elliptischen Kurve zu berechnen. Damit kann die Kurvenberechnung solange aufgeschoben werden, bis eine passende Ordnung m der Form $m = kq$ gefunden ist.

Der nächste Satz (siehe [3], Kapitel 8.6.2) gibt eine einfache Formel zur Konstruktion einer elliptischen Kurve parametrisiert durch die j-Invariante.

4.29 Theorem (Konstruktion)**:** *Sei eine beliebige j-Invariante j_0 gegeben. Dann gilt:*

1. *Ist $j_0 = 1728$, so hat $E : y^2 = x^3 + ax$ für alle $a \neq 0$ die j-Invariante j_0.*
2. *Ist $j_0 = 0$, so hat $E : y^2 = x^3 + b$ für alle $b \neq 0$ die j-Invariante j_0.*
3. *Ist $j_0 \neq 0, 1728$, so setze $c := j_0/(1728 - j_0)$. Dann hat*
$$E : y^2 = x^3 + 3cg^2x + 2cg^3$$
für alle $g \neq 0$ die j-Invariante j_0.

Beweis. Der Beweis folgt leicht durch explizites Nachrechnen von $j = \frac{1728(-4a)^3}{4a^3+27b^2}$ für die angegeben Kurven $E : y^2 = x^3 + ax + b$.

Für $j_0 = 1728$ hat $E : y^2 = x^3 + ax$ für alle $a \neq 0$ die Invariante
$$j = \frac{1728(-4a)^3}{-16(4a^3 + 27 \cdot 0)} = 1728.$$

Für $j_0 = 0$ hat $E : y^2 = x^3 + b$ für alle $b \neq 0$ die Invariante
$$j = \frac{1728(-4 \cdot 0)^3}{-16(4 \cdot 0^3 + 27b^2)} = 0.$$

Für alle $j_0 \neq 0, 1728$ hat $E : y^2 = x^3 + 3cg^2x + 2cg^3$ mit $c := j_0/(1728 - j_0)$ die Invariante
$$j = \frac{1728(-4 \cdot 3g^2 j_0/(1728 - j_0))^3}{-16(4 \cdot (3g^2 j_0/(1728 - j_0))^3 + 27 \cdot (2g^3 j_0/(1728 - j_0))^2)} = j_0.$$

□

4.30 Bemerkung (Konstruktion über $\mathbb{F}_p$)**:** *Über $\mathbb{F}_p$ bzw. $\mathbb{Z}/n\mathbb{Z}$ im Falle der zu testenden Zahl n wird die j-Invariante modulo n ermittelt. Entsprechend werden die Formeln aus Satz 4.29 auch modulo n berechnet, insbesondere ist $c := j_0 \cdot (1728 - j_0)^{-1} \in \mathbb{Z}/n\mathbb{Z}$.*

Mit diesen Herleitungen kann die allgemeine Vorgehensweise von ECPP abgeschlossen werden. Wie zuvor sei n diejenige natürliche Zahl, deren Primalität bewiesen werden soll. Insbesondere wird vorausgesetzt, dass n starke Pseudoprimzahl zu mehreren Basen ist. In den Sätzen zuvor nimmt n deshalb nun die Rolle von p ein.

In Satz 4.27 wurde bereits die allgemeine Vorgehensweise angedeutet:

1. Gesucht ist eine Fundamentaldiskriminante $D < -4$ und deren quadratischer Zahlkörper $K = \mathbb{Q}(\sqrt{D})$, sodass n in $n = \pi\bar{\pi}$ über $\mathcal{O}_K$ zerfällt. Ein $\pi \in \mathcal{O}_K$ hat nach Satz 4.12 und Bemerkung 4.13 die Gestalt $\pi = \frac{u+v\sqrt{D}}{2}$ mit $u, v \in \mathbb{Z}$. Somit gilt

$$n = \pi\bar{\pi} = \frac{u^2 - Dv^2}{4} = \frac{u^2 + |D|v^2}{4} \;\Leftrightarrow\; 4n = u^2 + |D|v^2$$

   wegen $D < 0$. Dies ist gerade die Aussage von Satz 4.22. Der Test auf eine Darstellung $4n = u^2 + |D|v^2$ geschieht per Algorithmus von Cornacchia (siehe [8], Algorithmus 2.3.13). Nicht jedes D erlaubt so eine Darstellung. Daher müssen gewöhnlich mehrere D ausprobiert werden. Der Algorithmus von Cornacchia liefert jedoch immer die Lösung (u, v), falls diese existiert.

2. Ist eine Fundamentaldiskriminante D und $\pi = \frac{u+v\sqrt{D}}{2}$ gefunden, so kommen nur die zwei Ordnungen $m_\pm = \#E(\mathbb{Z}/n\mathbb{Z}) = n + 1 \pm t$ in Frage, wobei

$$t = \pi + \bar{\pi} = \frac{u + v\sqrt{D}}{2} + \frac{u - v\sqrt{D}}{2} = u.$$

   Daher kann direkt getestet werden, ob eine der beiden Ordnungen

$$m_\pm = n + 1 \pm u$$

   die Gestalt $m = kq$ mit $q > (\sqrt[4]{n}+1)^2$ Pseudoprimzahl besitzt. Falls beide Ordnungen nicht faktorisiert werden können und/oder die falsche Gestalt haben, so muss die nächst kleinere Diskriminante ab Schritt 1 probiert werden. Wurde ein geeignetes $m_\pm$ gefunden, wird anschließend die elliptische Kurve E zu $m_\pm$ konstruiert.

3. Die Kurve wird über $\mathbb{C}$ anhand ihrer j-Invariante charakterisiert. Mögliche j-Invarianten, die zu Kurven mit Komplexer Multiplikation mit $\mathcal{O}_K$ gehören, sind im Hilbert'schen Klassenpolynom $H_D(X)$ kodiert. Nach Satz 4.22 sind die Existenz der Lösung $4n = u^2 + |D|v^2$ und die Faktorisierung von $H_D(X) \pmod n$ in Linearfaktoren äquivalent. Da nach Wahl von D und π eine Lösung $4n = u^2 + |D|v^2$ existiert, faktorisiert $H_D(X) \pmod n$. Die Nullstellen von $H_D(X) \pmod n$ sind die j-Invarianten

$$j([a, b, c]) := j(L_\tau) = j(\tau) = j\left(\frac{-b + \sqrt{D}}{2a}\right)$$

von quadratischen Formen (siehe Bemerkung 4.9).

4. Es gilt (siehe [7], Kapitel 3.4)

$$j(E_\tau) = 1728 g_2(L_\tau)^3/(g_2(L_\tau)^3 - 27 g_3(L_\tau)^2) = j(\tau),$$

daher kann eine beliebige Nullstelle $j(\tau)$ von $H_D(X) \pmod n$ als j-Invariante der elliptischen Kurve mit Weierstraß-Gleichung $y^2 = x^3 - g_2(L_\tau)x - g_3(L_\tau)$ benutzt werden.

5. Ist eine Nullstelle $j_0 \pmod n$ von $H_D(X) \pmod n$ berechnet (beispielsweise mit dem Algorithmus von Berlekamp, siehe [22]), so erhält man die expliziten Kurvenparameter zu j_0 mit Satz 4.29. Allerdings gibt es noch einen wichtigen Punkt zu beachten: Satz 4.29 besagt, dass mit $c := j_0/(1728 - j_0)$ alle Kurven

$$E : y^2 = x^3 + 3cg^2x + 2cg^3$$

für alle $g \neq 0$ die j-Invariante j_0 besitzen. Wählt man jedoch zwei verschiedene g, dann haben die beiden resultierenden elliptischen Kurven $E_1 : y^2 = x^3 + ax + b$ und $E_2 : y^2 = x^3 + a'x + b'$ nicht automatisch die beiden Ordungen $m_\pm$. Denn falls die beiden g's zwei isomorphe elliptische Kurven beschreiben, so haben beide natürlich die gleiche Ordnung. Nach Satz 3.11 ist bekannt, dass zwei Kurven (über $\mathbb{C}$) isomorph sind, falls $a' = \kappa^4 a$, $b' = \kappa^6 b$ für ein $\kappa \in \mathbb{C}^\times$, d.h. $\kappa \neq 0$. Dieser Fall muss daher noch ausgeschlossen werden. Der Einfachheit halber wählt man die erste Kurve durch $g = 1$ bereits fest aus und behält den Parameter g in der zweiten Kurvengleichung bei, d.h.

$$\begin{aligned} E_1 :&\quad y^2 = x^3 + 3cx + 2c, \\ E_2 :&\quad y^2 = x^3 + 3cg^2x + 2cg^3. \end{aligned}$$

Für die zweite Kurve wird g so gewählt, dass E_1 und E_2 nicht isomorph sind: $a' \neq \kappa^4 a$ ergibt $3cg^2 \neq \kappa^4 3c$ und somit $g^2 \neq \kappa^4$ ($c \neq 0$). Vereinfachen ergibt $g \neq \kappa^2$. Analog folgt für die zweite Bedingung $b' = \kappa^6 b$ eingesetzt $2cg^3 \neq \kappa^6 2c$ und somit $g^3 \neq \kappa^6$ oder $g \neq \kappa^2$. Dies bedeutet, dass die Kurven genau dann nicht isomorph sind, wenn g ein quadratischer Nichtrest ist.

In der Praxis wählt man deshalb einen quadratischen Nichtrest $g \pmod n$ und erhält mit

$$\begin{aligned} E_1 :&\quad y^2 = x^3 + 3cx + 2c \\ E_2 :&\quad y^2 = x^3 + 3cg^2x + 2cg^3 \end{aligned}$$

zwei Kurven zur j-Invariante j_0, die beide verschiedene Ordnungen besitzen. Eine der Kurven muss daher $m_+ = n + 1 + u$ und die andere $m_- = n + 1 - u$ Punkte besitzen.

6. Schließlich ist es einfach, die Ordnungen m_+ und m_- den Kurven E_1 und E_2 zuzuordnen. Dazu wählt man eine Kurve fest, beispielsweise E_1, und bestimmt einen Zufallspunkt $P \in E_1$ analog zu Schritt 4 im Algorithmus von Goldwasser-Kilian (Kapitel 3.6). Ist dann $[m_+]P = O$, aber $[m_-]P \neq O$, so gehört die Ordnung m_+ zu E_1 und m_- zu E_2. Ist $[m_+]P \neq O$, aber $[m_-]P = O$, so gehört m_- zu E_1 und m_+ zu E_2. Im Fall $[m_+]P = [m_-]P = O$ ist keine Entscheidung möglich und es muss ein neuer Punkt gewählt werden (dies tritt auf, falls die Ordnung von P ein Teiler von $\gcd(m_+, m_-)$ ist). Der Fall $[m_+]P \neq O$ und $[m_-]P \neq O$ kann bei fehlerfreier Berechnung nicht eintreten.

Der ECPP-Algorithmus

Die zuvor erklärte Vorgehensweise wird nun als vollständiger Primzahltest zusammengefasst. Wie in Kapitel 3.6 sei dazu ein $n \in \mathbb{N}$ gegeben, dass $\gcd(n, 6) = 1$ wie in Satz 3.30 erfüllt und bereits einen probabilistischen Primzahltest bestanden hat.

Atkin-Morain ECPP

1. Wähle eine Fundamentaldiskriminante $D < -4$, sodass zu $4n = u^2 + |D|v^2$ eine Lösung (u, v) per Algorithmus von Cornacchia ([8], Algorithmus 2.3.13) berechnet werden kann.

2. Teste beide $m_\pm = n + 1 \pm u$ auf Gestalt $m = kq$ mit $q > (\sqrt[4]{n} + 1)^2$ Pseudoprimzahl. Dauern die Faktorisierungen zu lange oder hat keines der beiden $m_\pm$ die benötigte Gestalt, so wiederhole ab Schritt 1.

3. Konstruiere die elliptische Kurve zu $m_\pm$. Berechne dazu zunächst eine beliebige Nullstelle j_0 des Hilbert'schen Klassenpolynoms $H_D(X) \pmod n$.

   Mit $c := j_0 \cdot (1728 - j_0)^{-1} \in \mathbb{Z}/n\mathbb{Z}$ und quadratischem Nichtrest $g \in \mathbb{Z}/n\mathbb{Z}$ kommen die beiden Kurven

   $$\begin{aligned} E_1 : &\quad y^2 = x^3 + 3cx + 2c \\ E_2 : &\quad y^2 = x^3 + 3cg^2x + 2cg^3 \end{aligned}$$

   in Betracht. Eine der Kurven hat die Ordnung m_+, die andere Ordnung m_-.

4. Wähle einen zufälligen Punkt $P \in E_1$ (wähle dazu zufällige $x \in \mathbb{Z}/n\mathbb{Z}$ und teste jeweils, ob $Z := x^3 + 3cx + 2c$ das Legendre-Symbol $\left(\frac{Z}{n}\right) \neq -1$ besitzt. Ermittle dann y aus $y^2 = Z$ per Algorithmus von Shanks-Tonelli. Falls dabei ein Fehler auftritt oder am Ende $y^2 \not\equiv Z \pmod n$ ist, so ist n nicht prim).

   Teste verschiedene $P = (x, y) \in E_1$ wie oben solange, bis $[m_+]P = O$ aber $[m_-]P \neq O$ gilt (dann ist $\mathrm{ord}(E_1) = m_+$ und $\mathrm{ord}(E_2) = m_-$) bzw. $[m_+]P \neq O$ aber $[m_-]P = O$ (dann ist $\mathrm{ord}(E_1) = m_-$ und $\mathrm{ord}(E_2) = m_+$).

   Hatte m_+ die geforderte Gestalt in Schritt 2, so setze $m := m_+$ und bezeichne die Kurve zu m_+ mit $E(a, b)$ (d.h. $(a, b) = (3c, 2c)$ bzw. $(a, b) = (3cg^2, 2cg^3)$). Analog für m_-, falls m_- die geforderte Gestalt in Schritt 2 hatte.

   Nun ist m die Ordnung von $E(a, b)$ und erfüllt $m = kq$ mit $q > (\sqrt[4]{n} + 1)^2$ Pseudoprimzahl. Berechne einen Punkt $P_E \in E$ wie zuvor (bzw. benutze den bereits berechneten Punkt $P_E := P$ von oben falls $E = E_1$).

5. Berechne $U := [m/q]P_E$. Ist $U = O$ so wiederhole ab Schritt 4. Ist $[q]U \neq O$ oder treten undefinierte Additionen in $E(\mathbb{Z}/n\mathbb{Z})$ auf, so ist n nicht prim.

6. Ausgabe ist das Zertifikat (n, a, b, m, q, U). Ist q prim, so auch n.

Außer der neuen Berechnungsweise von E, m und P per Komplexer Multiplikation ist die Rolle der elliptischen Kurve $E(a, b)$, der Primzahl q und des Punktes $U \in E$ gleich geblieben. Nach dem Satz 3.30 von Goldwasser-Kilian ist n prim, sobald q prim ist. Eine rekursive Anwendung auf die q_i analog zu Goldwasser-Kilian liefert ein Zertifikat für n, sobald die Primalität des letzten q_l per klassischem Primzahltest bewiesen ist.

Der Algorithmus von Cornacchia kann dem Quellcode im Anhang entnommen werden. Zum Abschluss folgt ein beispielhafter Primalitätsbeweis der Zahl $n = 10^{20} + 39$ per ECPP, die bereits in Beispiel 3.35 per Goldwasser-Kilian untersucht wurde. Das Beispiel wurde mit dem Programm zur Arbeit (siehe Anhang) berechnet.

4.31 Beispiel: *Die Primalität von* $n = 10^{20} + 39$ *soll bewiesen werden. Dazu wird zuerst nach einer Diskriminante* D *gesucht, für die* $4n = u^2 + |D|v^2$ *per Algorithmus von Cornacchia eine Lösung besitzt. Eine Suche über Fundamentaldiskriminanten* $D < -4$ *ergibt erstmalig für* $D = -15$ *eine Lösung*

$$4 \cdot (10^{20} + 39) = 19543688104^2 + 15 \cdot 1096790934^2$$

mit $u = 19543688104$*, die außerdem eine faktorisierbare Ordnung*

$$\begin{aligned} m_+ &= n + 1 + u = 100000000019543688144, \\ m_- &= n + 1 - u = 99999999980456311936 = 2^7 \cdot 19 \cdot 199 \cdot 4561 \cdot 45302615957, \end{aligned}$$

liefert (m_+ *besitzt einen relativ großen Primfaktor* 792691*). Dabei wurden alle Faktoren bis* 4561 *schnell per Probedivision gefunden. Die Pseudoprimzahl* $q = 45302615957$ *erfüllt* $q > (\sqrt[4]{n} + 1)^2$*.*

Erst nach der erfolgreichen Suche von m *wird die elliptische Kurve konstruiert. Die ermittelte Lösung von* $4n = u^2 + |D|v^2$ *impliziert die Faktorisierung von*

$$H_{-15}(X) = x^2 + 191025x - 121287375$$

in Linearfaktoren, wobei

$$j_0 = 96298930280091632533 \in \mathbb{Z}/n\mathbb{Z}$$

eine Nullstelle von $H_D(X) \pmod n$ *ist. Mit*

$$c := j_0 \cdot (1728 - j_0)^{-1} = 3491474911106435715 \in \mathbb{Z}/n\mathbb{Z}$$

und zufällig gewähltem quadratischem Nichtrest $g = 6737762177825767424 \in \mathbb{Z}/n\mathbb{Z}$ *kommen die beiden Kurven*

$$\begin{aligned} E_1: &\quad y^2 = x^3 + 3cx + 2c \\ E_2: &\quad y^2 = x^3 + 3cg^2x + 2cg^3 \end{aligned}$$

in Frage. Explizite Kurven über $\mathbb{Z}/n\mathbb{Z}$ *erhält man nach Einsetzen von* c *und* g *als*

$$\begin{aligned} E_1: &\quad y^2 = x^3 + 10474424733319307145x + 6982949822212871430, \\ E_2: &\quad y^2 = x^3 + 8509368889355718600x + 91336069641761520881. \end{aligned}$$

Der Zufallspunkt

$$P = (65783483021156220928 : 40505189531508570830 : 1)$$

liegt auf E_1 und erfüllt $[m_+]P \neq O$, $[m_-]P = O$. Damit steht die Zuordnung von $\mathrm{ord}(E_1) = m_-$ *und* $\mathrm{ord}(E_2) = m_+$ *fest.*

Für den Primalitätsbeweis ist die Ordnung m_- und damit deren dazugehörige Kurve E_1 geeignet. Der Punkt $P_E := P$ liegt bereits auf E_1 und erfüllt außerdem $U := [m/q]P_E \neq O$ und $[q]U = O$. Somit erfüllen $E := E_1$, q und $U = [m/q]P_E$ den Satz von Goldwasser-Kilian unter der Annahme, dass q prim ist. Rekursive Anwendung des Algorithmus auf $q < n$ zeigt schließlich, dass q in der Tat eine Primzahl ist. Nach Goldwasser-Kilian/ ECPP ist mit q auch n prim.

4.8 Laufzeit

Die Laufzeit der ECPP-Methode kann nicht genauso anschaulich hergeleitet werden wie die von Goldwasser-Kilian in Kapitel 3.7. Nach [8], Kapitel 7.6.2, zeigen heuristische Abschätzungen eine Laufzeit von $O(k^{4+\epsilon})$, wobei der Zusatzaufwand ϵ aus der unbekannten Verteilung der Kurvenordnungen $m = kq$ stammt, die bei der Komplexen Multiplkation verwendet werden. In der Praxis liefert die ECPP-Methode aber deutlich schnellere Zertifikate als der Originalalgorithmus von Goldwasser-Kilian. Der Grund liegt im fehlenden Schoof-Algorithmus. Zwar muss bei beiden Methoden probiert werden (elliptische Kurven bei Goldwasser-Kilian und Diskriminanten bei ECPP), allerdings ist es deutlich aufwändiger, die Ordnung jedes Mal in $O(k^9)$ bei Goldwasser-Kilian zu ermitteln, statt Lösungen von $4n = u^2 + |D|v^2$ per Cornacchia zu finden. Bei beiden Methoden muss die gefundene Ordnung m anschließend auf eine Faktorisierung $m = kq$ geprüft werden.

4.9 Konstruktion von elliptischen Kurven mit vorgegebener Ordnung

Im zuvor beschriebenen ECPP-Algorithmus wurden elliptische Kurven mit bekannten Ordnungen hergeleitet. Die Ordnungen $m_\pm = p + 1 \pm u$ selbst ergaben sich aber aus der Lösung von $4p = u^2 + |D|v^2$ (hier sei $p = n$ prim) und wurden nicht von außen vorgegeben. Mit Hilfe der Komplexen Multiplikation kann jedoch auch eine elliptische Kurve zu vorgegebener Ordnung M über einem vorgegebenen Körper $\mathbb{F}_p$ mit p prim berechnet werden. Dabei muss M natürlich im Hasse-Intervall $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$ liegen.

Dazu muss der obige Algorithmus nur minimal verändert werden. Ohne Beschränkung wird dazu nun die Ordnung $m_- = p + 1 - u$ betrachtet. Vorgegeben sind die Punktzahl $M < p + 1$ und die Primzahl p, d.h.

$$M \stackrel{!}{=} m_- = p + 1 - u \;\Rightarrow\; u = p + 1 - M.$$

Wie zuvor muss p zur Anwendung der Komplexen Multiplikation in $p = \pi\bar{\pi}$ über $\mathcal{O}_K$, $K = \mathbb{Q}(\sqrt{D})$, nach Satz 4.27 faktorisieren. Nach Satz 4.22 ist dies erneut äquivalent

zu

$$p = \pi\bar{\pi} \in \mathcal{O}_K \;\Leftrightarrow\; 4p = u^2 + |D|v^2 = u^2 - Dv^2$$

da $D < 0$. Allerdings wird der Algorithmus von Cornacchia nicht mehr zur Suche von (u, v) benötigt, da u bereits per $u = p + 1 - M$ gegeben ist. Stattdessen werden nun eine Diskriminante D und ein v gesucht, sodass $4p = u^2 - Dv^2$ eine Lösung für gegebene p und u besitzt. Umstellen nach D ergibt

$$D = \frac{u^2 - 4p}{v^2}.$$

Man sucht daher nach einem v, sodass $v^2|(u^2 - 4p)$, und setzt $D := (u^2 - 4p)/v^2$.

Ist die Lösung von $4p = u^2 + |D|v^2$ für ein D gefunden, so faktorisiert $H_D(X) \pmod p$ nach Satz 4.22 wie zuvor. Abschließend wird eine Nullstelle j_0 von $H_D(X) \pmod p$ berechnet und daraus mit Satz 4.29 die zwei möglichen elliptischen Kurven E_1, E_2 ermittelt, wobei wie im ECPP-Algorithmus durch Punkte $P \in E_1$ geprüft werden muss, welche der beiden Kurven E_1, E_2 wirklich Ordnung $M = p + 1 - u$ besitzt. Die jeweils andere Kurve besitzt Ordnung $p + 1 + u$.

4.32 Beispiel: *Für die Primzahl $p = 10^{10} + 32751$ soll eine Kurve mit exakt $M = 10^{10}$ Punkten über $\mathbb{F}_p$ konstruiert werden. Dafür wird zunächst u wie oben berechnet:*

$$u = p + 1 - M = 32752.$$

Als nächstes sind v und D gesucht, sodass $v^2|(u^2 - 4p)$. Eine Computersuche (bzw. Faktorisierung von $u^2 - 4p$) ergibt eine Lösung von $4p = u^2 - Dv^2$ mit

$$v = 1750,\; D = -12711.$$

Das Hilbertpolynom $H_D(X)$ faktorisiert nun modulo p, sodass leicht eine Nullstelle

$$j_0 = 9752490586 \in \mathbb{F}_p$$

gefunden wird. Mit

$$c := j_0 \cdot (1728 - j_0)^{-1} = 6644248527 \in \mathbb{F}_p$$

und zufällig gewähltem quadratischem Nichtrest $g = 3980614158 \in \mathbb{F}_p$ kommen die beiden Kurven (Formeln analog Beispiel 4.31)

$$\begin{aligned} E_1 :&\quad y^2 = x^3 + 9932712830x + 3288464303, \\ E_2 :&\quad y^2 = x^3 + 6637802908x + 8297672347. \end{aligned}$$

in Frage. Der Zufallspunkt

$$P = (2839844080 : 5474549347 : 1)$$

liegt auf E_1 *und erfüllt* $[M]P \neq O$, $[p+1+u]P = O$. *Damit steht die Zuordnung von* $\mathrm{ord}(E_1) = p+1+u$ *und* $\mathrm{ord}(E_2) = M$ *fest. Eine Berechnung der Ordnung per Schoof-Algorithmus zur Kontrolle bestätigt*

$$\#E_2(\mathbb{F}_p) = 10^{10}.$$

Verwendung in ECPP

Diese Variante zur Konstruktion von Kurven kann auch im ECPP-Algorithmus benutzt werden, wobei ausgenutzt wird, dass die Ordnung der noch nicht konstruierten Kurve später im Hasse-Intervall $[n+1-2\sqrt{n}, n+1+2\sqrt{n}]$ liegen wird. Mit einem Siebverfahren können viele Ordnungen im Hasse-Intervall simultan auf eine Faktorisierung der Form $M = kq$ mit Pseudoprimzahl $q > (\sqrt[4]{n}+1)^2$ untersucht werden. Ist eine geeignete Ordnung M gefunden, so wird wie oben beschrieben eine elliptische Kurve über $\mathbb{Z}/n\mathbb{Z}$ mit Ordnung M konstruiert. Anschließend bleibt nur noch, analog zum Algorithmus von Goldwasser-Kilian zu E mit Ordnung $M = kq$ einen Punkt $U \neq O$ mit $[q]U = O$ zu finden.

5 Empirische Ergebnisse einer Implementierung in SAGE

Im Rahmen der Masterarbeit wurden die beiden Primzahltests von Goldwasser-Kilian und Atkin-Morain (ECPP) sowie eine Funktion zur Verifikation von bereits erstellten Zertifikaten in der Programmiersprache Cython für die Mathematiksoftware SAGE (siehe [27]) programmiert. Der Quellcode der Programme befindet sich im Anhang der Arbeit. Empirische Ergebnisse zu den Programmen folgen in den nächsten Abschnitten.

5.1 Goldwasser-Kilian

Der Goldwasser-Kilian Test funktioniert exakt wie im Abschnitt 3.6 dargestellt. Das Programm unterstützt die Suche nach Ordnungen $m = 2q$ wie im Originalalgorithmus von Goldwasser und Kilian sowie die Suche nach Ordnungen der Form $m = kq$, wobei k als das Produkt aller Primzahlen kleiner 10^3 gewählt wurde, die m teilen.

Als Beispiel für ein Zertifikat wird $n = 10^{20} + 39$ auf Primalität getestet. Das Ergebnis des Aufrufs

```
goldwasser_kilian(10**20+39)
```

ist das Zertifikat

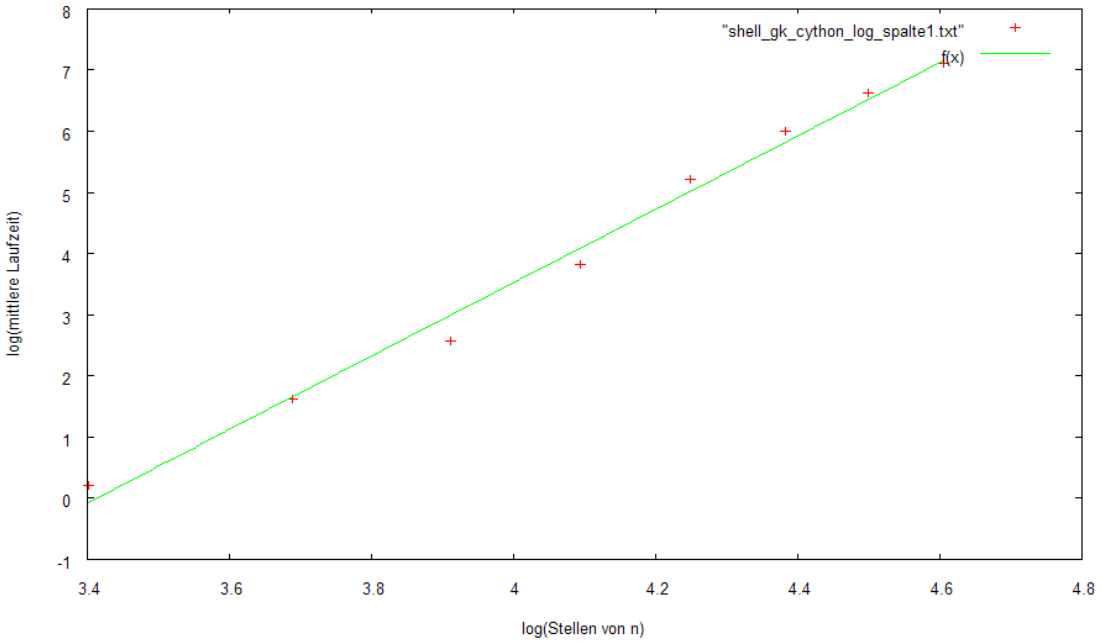
$$
\begin{aligned}
&[10000000000000000039, 31484432173069852672, 39553474583282556928,\\
&10000000014867206541, 539348143913549,\\
&(39164891430400385024 : 86449249723524901718 : 1),\\
&539348143913549, 155238070886498, 37712508792888,\\
&539348162953818, 13178619043, (450430088614431 : 340614625612821 : 1),\\
&13178619043, 8257850338, 2005441708, 13178568208,\\
&63358501, (4691694285 : 11267068726 : 1),\\
&63358501, 36280326, 60341751, 63363972, 754333, (40950989 : 55530902 : 1)],
\end{aligned}
$$

das aus aneinandergehängten 6-Tupeln $[q_i, a_i, b_i, m_i, q_{i+1}, U_i]$ besteht. Dabei ist q_i die aktuelle Zahl im *Downrun*, (a_i, b_i) sind die Kurvenparameter einer elliptischen Kurve $E(a_i, b_i)$ mit Punkt U_i, m_i ist die Ordnung von E und q_{i+1} die Zahl, auf die die Primalität von q_i reduziert wurde. Es ist gut zu sehen, dass alle q_i ab $q_0 = n = 10^{20} + 39$ monoton fallen bis das letzte $q_3 = 754333$ so klein ist, dass dessen Primalität leicht per klassischem Test gezeigt werden kann.

Die folgende Tabelle zeigt die durchschnittliche Laufzeit und die durchschnittliche Zertifikatlänge für jeweils zehn Durchläufe pro Testzahl n:

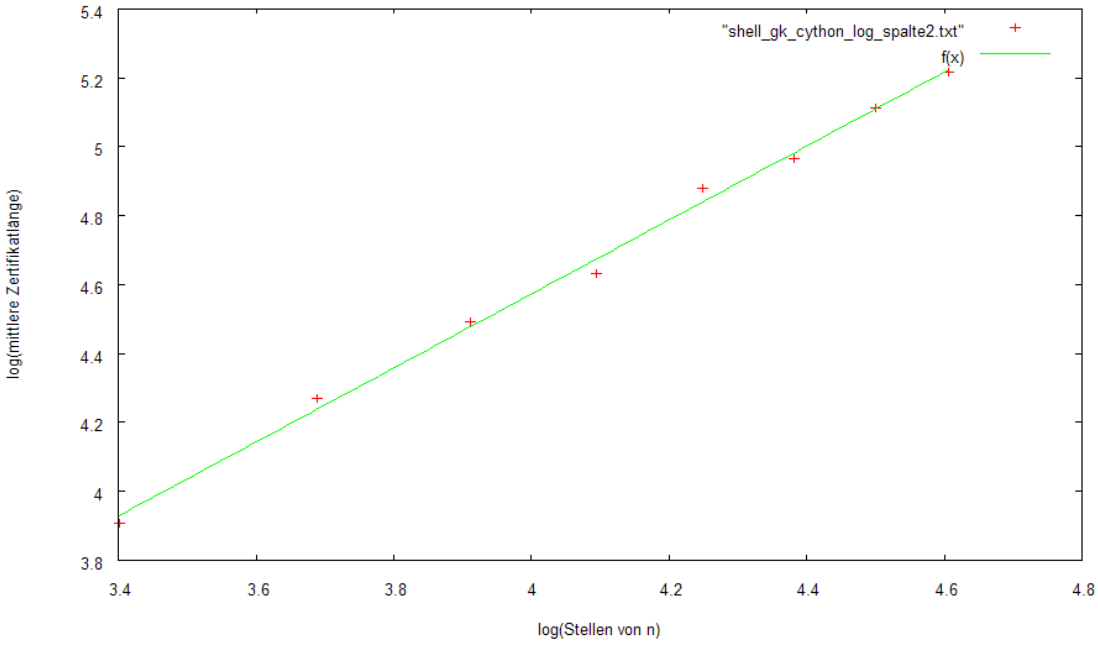
Stellen von n	mittlere Laufzeit in Sekunden	mittlere Zertifikatlänge
30	1.238	49.8
40	5.010	71.4
50	12.975	89.4
60	45.988	102.6
70	183.494	131.4
80	400.091	143.4
90	755.513	166.2
100	1230.028	184.5

Ein logarithmischer Plot der Laufzeit zu den Stellen von n zeigt einen linearen Zusammenhang, der die polynomielle Laufzeit bestätigt:



Die Steigung der Fitgeraden entspricht dem Exponenten der polynomiellen Laufzeit und beträgt für die obige Gerade 6.00398 ± 0.2406. Die empirische Laufzeit liegt damit unter der vorhergesagten Gesamtlaufzeit von $O(k^9)$ (In Kapitel 3.7, Satz 3.40, wurde eine Gesamtlaufzeit von $O(k^{11+c_1})$ hergeleitet, wobei c_1 aus der Vermutung 3.36 von Goldwasser-Kilian stammt. Die am Ende von Kapitel 3.7 motivierte Wahl von $c_1 = 1$ lieferte eine Laufzeit von $O(k^{12})$. Die im Programm verwendete Schoof-Elkies-Atkin Variante ist mit $O(k^6)$ deutlich schneller als der ursprüngliche Schoof-Algorithmus mit $O(k^9)$ und reduziert die Gesamtlaufzeit auf $O(k^9)$.)

Auch die in Kapitel 3.7 hergeleitete polynomielle Länge des Zertifikats kann auf die gleiche Weise mit Hilfe eines logarithmischen Plots von Zertifikatlänge zu Stellen von n überprüft werden:



Der Aufruf `goldwasser_kilian(n)` in SAGE testet die übergebene Zahl n auf Primalität und gibt ein vollständiges Zertifikat für n zurück, falls n prim ist, bzw. 0 sonst.

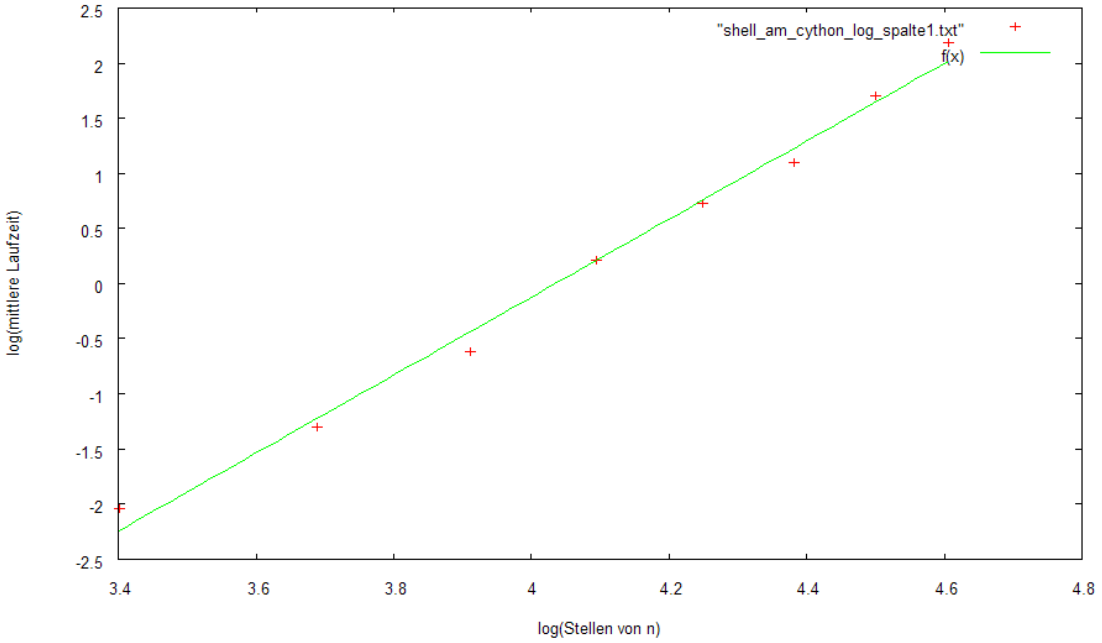
5.2 Atkin-Morain (ECPP)

Analog zu Goldwasser-Kilian kann auch der Algorithmus von Atkin-Morain (ECPP) in SAGE verwendet werden. Die folgende Tabelle zeigt erneut mittlere Laufzeiten für jeweils zehn Eingaben n verschiedener Größenordnungen und die dazugehörige Zertifikatlänge.

Stellen von n	mittlere Laufzeit in Sekunden	mittlere Zertifikatlänge
30	0.129	45.6
40	0.273	61.8
50	0.539	81.6
60	1.234	100.2
70	2.071	118.8
80	3.003	131.4
90	5.484	151.2
100	8.916	168.0

Der Geschwindigkeitsvorteil von ECPP gegenüber Goldwasser-Kilian ist anhand der experimentellen Werte gut zu beobachten: Für eine 100-stellige Zahl benötigt Goldwasser-Kilian durchschnittlich 1230 Sekunden, ECPP durchschnittlich nur 9 Sekunden.

Ein logarithmischer Plot der Laufzeit zur Stellenanzahl von n zeigt auch hier die polynomielle Laufzeit von ECPP:



Die Steigung der Fitgeraden beträgt hier 3.53607 ± 0.1319 und liegt nahe an der erwarteten Laufzeit von $O(k^{4+\epsilon})$ aus Kapitel 4.8.

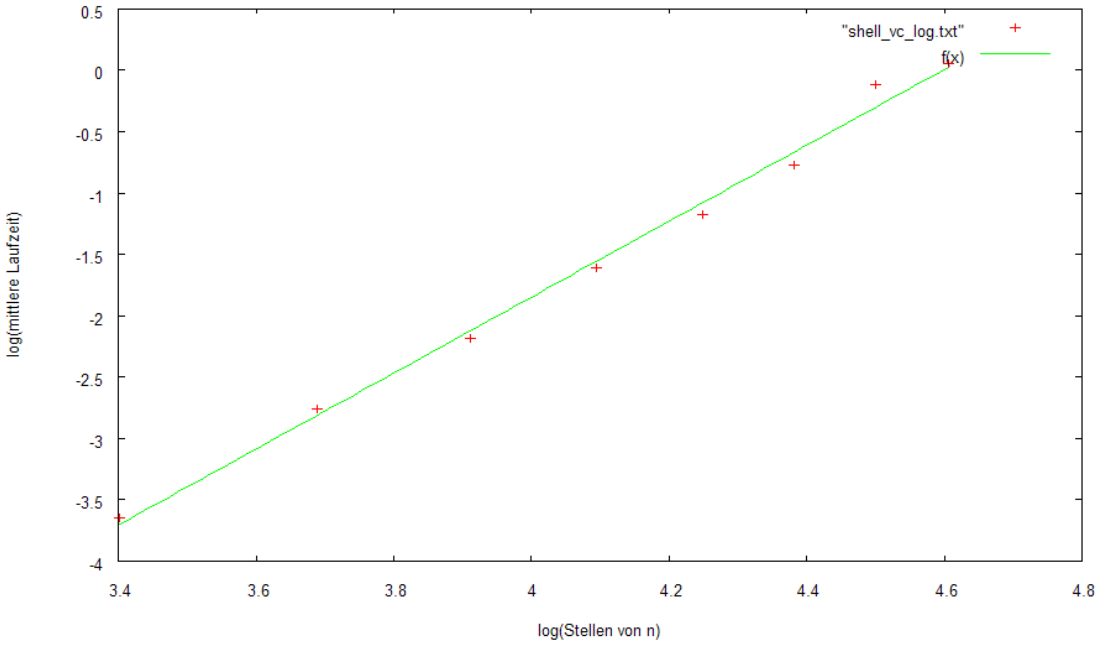
Der Aufruf `atkin_morain(n)` bzw. `ecpp(n)` (`ecpp(n)` ist keine neue Funktion, sondern nur ein kurzes Synonym für `atkin_morain(n)`) in SAGE testet die übergebene Zahl n auf Primalität und gibt ein vollständiges Zertifikat für n zurück, falls n prim ist, bzw. 0 sonst.

5.3 Verifikation eines Zertifikates

Die folgende Tabelle zeigt gemittelte Laufzeiten zur Überprüfung eines Zertifikats von jeweils zehn Eingaben n verschiedener Größenordnungen. Jedes Zertifikat wurde zuvor per Aufruf `ecpp(n)` erstellt.

Stellen von n	mittlere Laufzeit in Sekunden
30	0.026
40	0.063
50	0.113
60	0.200
70	0.308
80	0.460
90	0.894
100	1.055

Anhand der Tabelle ist zu sehen, dass ein Primalitätszertifikat deutlich schneller geprüft als erstellt werden kann. Der entsprechende logarithmische Plot von Stellenanzahl zu mittlerer Laufzeit bestätigt die hergeleitete polynomielle Laufzeit zur Verifizierung eines Zertifikats.



Die Fitgerade für die Verifikation eines Zertifikats hat die Steigung 3.09547 ± 0.09813. Dies entspricht näherungsweise der Laufzeit von $O(k^4)$, die in Satz 3.43 hergeleitet wurde.

Der Aufruf `verify_certificate(c)` in SAGE testet das übergebene Zertifikat c für n, das zuvor per `c=goldwasser_kilian(n)`, `c=atkin_morain(n)` oder `c=ecpp(n)` erstellt wurde.

Die Funktion `verify_certificate(c)` gibt 1 zurück, falls das Zertifikat c erfolgreich verifiziert werden konnte (und daher n mit gültigem Zertifikat prim ist), oder 0, falls c ungültig ist.

5.4 Implementierte Unterprogramme

Außer den vier bereits separat vorgestellten Hauptprogrammen `goldwasser_kilian(n)`, `atkin_morain(n)`, `ecpp(n)` und `verify_certificate(n)` beinhaltet die SAGE Implementierung weitere nützliche Unterprogramme:

1. `findq1(m,n)`: testet die Ordnung m auf Gestalt $m = 2q$ mit Pseudoprimzahl $q > (\sqrt[4]{n}+1)^2$
2. `findq2(m,n,bound)`: testet die Ordnung m auf Gestalt $m = kq$ mit Pseudoprimzahl $q > (\sqrt[4]{n}+1)^2$, wobei alle kleinen Primfaktoren von m bis zur Schranke `bound` in k abgespalten werden
3. `get_point(a,b,n)`: gibt einen nicht-trivialen Zufallspunkt $P = (x, y)$ auf der elliptischen Kurve $E(a, b)$ über $\mathbb{Z}/n\mathbb{Z}$ zurück
4. `goldwasser_kilian_downrun_step(n)`: reduziert die Primalität von n per Algorithmus von Goldwasser-Kilian auf ein kleineres q und gibt das Teilzertifikat zurück (dabei wird der bereits in SAGE programmierte Schoof-Elkies-Atkin Algorithmus zur Bestimmung der Ordnung in $O(k^6)$ statt $O(k^9)$ bei Schoof verwendet)
5. `cornacchia_smith(D,p)`: berechnet eine Lösung (u, v) von $4p = u^2 + |D|v^2$ mit dem Algorithmus von Cornacchia-Smith
6. `is_cube(x,p)`: testet, ob $a^3 \equiv x \pmod{p}$ eine Lösung mit $a \in \mathbb{F}_p$ besitzt
7. `curve_parameters(D,p,order)`: ist eine faktorisierbare Ordnung `order` unter den beiden Kandidaten $m_\pm = p + 1 \pm u$ gefunden (u per Algorithmus von Cornacchia-Smith), so berechnet diese Funktion die Kurvenparameter (a, b) einer elliptischen Kurve $E(a, b)$ mit Ordnung `order` über $\mathbb{F}_p$ per Methode der Komplexen Multiplikation
8. `oddpart(n)`: liefert den ungeraden Teil von n zurück (d.h. d aus der Darstellung $n = 2^e \cdot d$ mit $2 \nmid d$)
9. `squarefree(n)`: testet, ob n quadratfrei ist
10. `getD(p)`: sucht eine Fundamentaldiskriminante D zu p so, dass zu (D, p) eine Cornacchia-Smith Lösung (u, v) existiert und eine der resultierenden Ordnungen $m_\pm = p + 1 \pm u$ die Gestalt $m = kq$ mit Pseudoprimzahl $q > (\sqrt[4]{p}+1)^2$ besitzt
11. `atkin_morain_downrun_step(n)`: reduziert die Primalität von n per Algorithmus von Atkin-Morain auf ein kleineres q und gibt das Teilzertifikat zurück

6 Diskussion der Ergebnisse und Ausblick

Die vorliegende Arbeit gibt eine Einführung in die Theorie polynomieller Primzahltests anhand des Tests von Agrawal-Kayal-Saxena und behandelt speziell die polynomiellen Primzahlbeweise nach Goldwasser-Kilian und Atkin-Morain. Dabei werden elliptische Kurven, der Satz von Hasse und das Primzahlkriterium von Goldwasser-Kilian eingeführt und bewiesen. Der Beweis des Satzes von Goldwasser-Kilian ergibt unmittelbar einen effizienten Algorithmus zur Überprüfung der Primalität einer Eingabezahl. Die effiziente polynomielle Laufzeit wird detailliert hergeleitet und auf zugrundeliegende Annahmen hingewiesen.

Die Erweiterung zu ECPP nach Atkin-Morain verzichtet auf die Bestimmung der Ordnung per Schoof-Algorithmus und konstruiert elliptische Kurven mit bekannten Ordnungen per Methode der Komplexen Multiplikation. Diese Methode wird in der Arbeit zusammen mit dem verbesserten Algorithmus hergeleitet. Außerdem wird dargestellt, wie die Komplexe Multiplikation dazu verwendet werden kann, Kurven mit vorgegebenen Ordnungen zu berechnen.

Die theoretischen Ergebnisse werden im Kapitel zur Implementierung in SAGE verifiziert. Dazu gehören logarithmische Plots des Aufwands, die die polynomielle Laufzeit aller Funktionen (Goldwasser-Kilian, ECPP, Verifikation eines Zertifikates) sowie die polynomielle Zertifikatlänge bestätigen. Außerdem wird ersichtlich, dass ECPP Primalitätsbeweise in der Praxis deutlich schneller berechnet. Die Implementierung im Anhang liefert zudem schnelle Funktionen zur Bestimmung von Zufallspunkten auf elliptischen Kurven, zur Suche geeigneter Fundamentaldiskriminanten mit faktorisierbaren Kurvenordnungen und zur Berechnung der Parameter einer elliptischen Kurve mit vorgegebener Ordnung per Methode der Komplexen Multiplikation.

Mittlerweile (Stand 2010) wurde auch ECPP weiter verbessert. Ein ungelöstes Problem des Goldwasser-Kilian/ECPP Algorithmus besteht darin, dass diese Tests eine polynomielle Laufzeit nur für fast alle Eingaben n bis auf einen exponentiell kleinen Anteil besitzen (siehe Bemerkung 3.41). Die Verbesserung nach Adleman-Huang (siehe [23] und [24]) führt daher für eine beliebige Eingabe n zuerst einen Reduktionsschritt der Primalität auf ein zufällig verteiltes n' durch, dass auch größer als n sein kann. Damit verlagert diese Variante das Problem auf eine zufällige andere Zahl n', die mit hoher Wahrscheinlichkeit zu dem Anteil der effizient beweisbaren Eingaben von ECPP gehört. Andernfalls reduziert man erneut auf ein weiteres zufälliges n', und wendet ECPP auf n' an. Die Primalität von n' impliziert dann auch die Primalität von n. Für diese Variante ist ein echter erwarteter polynomieller Aufwand für *alle* Eingaben bewiesen worden. Auf diese Weise konnten bereits Zertifikate für Primzahlen von mehr als 2000 Stellen berechnet werden. Ein weiteres Verfahren nach P. Mihailescu (siehe [25]), das nicht auf elliptischen Kurven, sondern auf zyklotomischen Erweiterungen der Ringe $(\mathbb{Z}/s\mathbb{Z})$ aufbaut, ist in diesen Größenordnungen mittlerweile sogar schneller als ECPP. Allerdings ist die Verifikation eines solchen Zertifikates nach Mihailescu im Gegensatz zum Goldwasser-Kilian/ECPP Zertifikat kaum schneller als dessen Generierung (siehe [2]).

Eigenständigkeitserklärung

Ich erkläre, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Datum Unterschrift

Literatur

[1] S. Goldwasser, J. Kilian, *Almost all primes can be quickly certified*, Proceedings of the 18th Annual ACM Symposium on Theory of Computing (Berkeley, Calif., May 28-30), ACM, New York, pp. 316-329.

[2] S. Goldwasser, J. Kilian, *Primality Testing Using Elliptic Curves*, Journal of the ACM, Vol. 46, No. 4, July 1999, pp. 450-472.

[3] A. Atkin, F. Morain, *Elliptic Curves and Primality Proving*, Mathematics of Computation, Volume 61, Number 203, July 1993, pp. 29-68.

[4] F. Morain, *Implementation of the Atkin-Goldwasser-Kilian Primality Testing Algorithm*, INRIA Research Report, No. 911, October 1988.

[5] J. Silverman, *The Arithmetic of Elliptic Curves (Graduate Texts in Mathematics)*, Springer, 1986, ISBN 3-540-96203-4.

[6] J. Silverman, *Advanced Topics in the Arithmetic of Elliptic Curves (Graduate Texts in Mathematics)*, Springer, 1994, ISBN 0-387-94325-0.

[7] B. Holm, *Constructing Elliptic Curves with a Given Number of Points*, Part III Essay, 2005, *http://www.cl.cam.ac.uk/∼bh288/publications/Holm_2005_partiii-essay.pdf*.

[8] R. Crandall, C. Pomerance, *Prime Numbers - A Computational Perspective*, Springer, 2005, ISBN 0-387-25282-7.

[9] S. Müller-Stach, J. Piontkowski, *Elementare und algebraische Zahlentheorie*, Vieweg, 2006, ISBN 978-3-8348-0211-8.

[10] T. Dokchitser (*http://www.dpmms.cam.ac.uk/∼td278/*), *Elliptic Curves*, Vorlesungsskript.

[11] J. Milne, *Elliptic Curves*, Booksurge Llc, 2006, ISBN 1-419-65257-5.

[12] V. Pratt, *Every Prime has a Succinct Certificate*, SIAM J. of Comp., 1975, pp. 214-220.

[13] R. Schoof, *Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p*, Mathematics of Computation, Vol. 44, No. 170. (1985), pp. 483-494.

[14] S. Goldwasser (*http://people.csail.mit.edu/shafi/*), M. Bellare, *Lecture Notes on Cryptography*, 2008, *http://cseweb.ucsd.edu/∼mihir/papers/gb.pdf*.

[15] G. Cornacchia, *Su di un metodo per la risoluzione in numeri interi dell' equazione* $\sum_{h=0}^{n} C_h x^{n-h} y^h = P$, Giornale di Matematiche di Battaglini 46 (1908), pp. 33-90.

[16] F. Morain, J. Nicolas, *On Cornacchia's algorithm for solving the diophantine equation* $u^2 + dv^2 = m$, March 1990 (nicht publiziert).

[17] G. A. Jones, D. Singerman, *Complex Functions: An algebraic and geometric viewpoint*, Lectures in Mathematics, Cambridge University Press, 1987.

[18] R. Bröker, P. Stevenhagen, *Elliptic Curves with a Given Number of Points*, Algorithmic Number Theory, pp. 117-131, Lecture Notes in Comput. Sci., 3076, Springer, 2004.

[19] S. Lang, *Elliptic functions*, Graduate texts in mathematics 112, Springer, New York/London, 2nd edition, 1987.

[20] D. Bernstein, *Proving Primality after Agrawal-Kayal-Saxena*, Draft 1991.

[21] H. Lenstra, Jr., *Factoring integers with elliptic curves*, Ann. of Math., 2:649-673, 1987.

[22] E. Berlekamp, *Factoring Polynomials Over Finite Fields*, Bell System Technical Journal, Band 46, 1967, pp. 1853-1859.

[23] L. Adleman, M. Huang, *Recognizing primes in polynomial time*, Proceedings of the 19th Annual ACM Symposium on Theory of Computing (New York, N.Y., May 25-27), ACM, New York, 1987, pp. 462-471.

[24] L. Adleman, M. Huang, *Primality testing and Abelian varieties over finite fields*, Lecture Notes in Mathematics, vol. 1512, Springer-Verlag, New York, 1992.

[25] P. Mihailescu, *Cyclotomy primality proving - Recent developments*, Proceedings of the 3rd International Algorithmic Number Theory Symposium (ANTS), Lecture Notes in Computer Science, vol. 877, Springer-Verlag, New York, 1994, pp. 95-110.

[26] Wikipedia (Englisch), *http://en.wikipedia.org*, Artikel *Primality certificate* und *Cramér's conjecture*.

[27] SAGE (a free open-source mathematics software system), *http://sagemath.org/*.

Appendix

Die folgenden Seiten beinhalten den kompletten Code einer Implementierung der Tests nach Goldwasser-Kilian und Atkin-Morain sowie eine Funktion zur schnellen Verifikation von Zertifikaten in der Programmiersprache Cython für die Mathematiksoftware SAGE.

```
%cython


"""
Primality proving via Goldwasser-Kilian and Atkin-Morain (ECPP)

main functions:
    - goldwasser_kilian(n): computes a primality certificate for n
    via Goldwasser-Kilian, returns a certificate for n
    if n is prime, 0 if n is composite, -1 if an error occurred
    - atkin_morain(n): computes a primality certificate for n via
    Atkin-Morain (ECPP), returns a certificate for n if
    n is prime, 0 if n is composite, -1 if an error occurred
    - ecpp(n): alias for atkin_morain(n)
    - verify_certificate(c): verifies a previously computed
    certificate (all functions above are compatible), returns
    1 if verification was successful (i.e. n is prime) and 0 if
    certificate is invalid

AUTHORS:

- Georg Hahn (2010-11-27): initial version

EXAMPLES:

Compute a certificate for n=10**20+39 via goldwasser_kilian(n)::

    sage: goldwasser_kilian(10**20+39) #random
    [100000000000000000039, 314844321730698526 72L,
    395534745832825569 28L, 100000000014867206541, 539348143913549,
    (391648914304003850 24 : 864492497235249017 18 : 1),
    539348143913549, 155238070886498, 37712508792888,
    539348162953818, 13178619043,
    (450430088614431 : 340614625612821 : 1), 13178619043,
    8257850338, 2005441708,
    13178568208, 63358501, (4691694285 : 11267068726 : 1), 63358501,
    36280326, 60341751,63363972, 754333,
    (40950989 : 55530902 : 1)]

Compute a certificate for n=10**20+39 via atkin_morain(n)::

    sage: atkin_morain(10**20+39) #random
    [100000000000000000039, 0, 100000000000000000038,
    99999999982060129348, 964019216509,
    (19568017359532265472 : 20079688430965599468 : 1), 964019216509,
    0, 858481324253L, 964020818668, 241005204667,
    (799886189130 : 601302851255 : 1), 241005204667, 0, 11167880189,
    241006096857, 626011,
    (140197349256 : 38879868868 : 1)]

Compute a certificate for n=10**20+39 via ecpp(n)::

    sage: ecpp(10**20+39) #random
    [100000000000000000039, 0, 100000000000000000038,
    99999999982060129348, 964019216509,
    (19568017359532265472 : 20079688430965599468 : 1), 964019216509,
    0, 858481324253L, 964020818668, 241005204667,
```

```
    (799886189130 : 601302851255 : 1), 241005204667, 0, 11167880189,
    241006096857, 626011, (140197349256 : 38879868868 : 1)]

Verify a certificate computed by ecpp via verify_certificate(c)::

    sage: c = ecpp(10**20+39);
    sage: verify_certificate(c)
    1

"""

#*****************************************************************
#       Copyright (C) 2010 Georg Hahn <ghahn@cantab.net>
#
#  Distributed under the terms of the GNU General Public License
#  (GPL)
#                  http://www.gnu.org/licenses/
#*****************************************************************


from sage.all import gcd, xgcd
from sage.all import power_mod
from sage.all import random
from sage.all import EllipticCurve
from sage.all import GF
from sage.all import kronecker
from sage.all import is_prime, is_pseudoprime
from sage.all import Mod
from sage.all import isqrt
from sage.all import hilbert_class_polynomial
from sage.rings.finite_rings.integer_mod import
square_root_mod_prime


def findq1(m,n):
    """
    function tests whether a curve order m is of the form m=2q,
    q>(n^(1/4)+1)^2 pseudoprime

    INPUT:

     - the integer m to test, and an integer n used to calculate the
     bound (n^(1/4)+1)^2

    OUTPUT:

    integer -- function directly returns q>1 if the curve order m is
    of desired form and 0 otherwise

    EXAMPLES:

    A simple example::

        sage: findq1(1928038433018,10**20+39)
        964019216509
```

```
    NOTES:

    function works accordingly to the original paper by Goldwasser
    and Kilian

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    if(m%2==1):
        return 0;
    q = m>>1;
    if((q>(isqrt(isqrt(n)+1)+1+1)**2) and (is_pseudoprime(q)==1)):
        return q;
    return 0;


def findq2(m,n,bound):
    """
    function tests whether a curve order m is of the form m=kq,
    q>(n^(1/4)+1)^2 pseudoprime, where k is the product of
    all primes up to parameter "bound" that divide m

    INPUT:

     - the integer m to test, an integer parameter "bound" as upper
     bound for finding small prime factors of m by
     trial division and an integer n used to calculate the bound
     (n^(1/4)+1)^2

    OUTPUT:

    integer -- function directly returns q>1 if the curve order m is
    of desired form and 0 otherwise

    EXAMPLES:

    A simple example::

        sage: findq2(99999999982060129348,10**20+39,10**3)
        964019216509L

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    x = 3;
    lastx = 1;
    m = oddpart(m);
    while((x<=bound) and (m>1)):
        while(m%x==0):
            m = int(m/x);
            lastx = x;
        x = x+2;

    q = m;
```

```
    if(q==1):
        q = lastx;
    if((q>(isqrt(isqrt(n)+1)+1+1)**2) and (is_pseudoprime(q)==1)):
        return q;
    return 0;


def get_point(a,b,n):
    """
    function seeks a point on the curve E: y^2=x^3+ax+b modulo n,
    where n is prime

    INPUT:

     - the curve parameters a,b of E: y^2=x^3+ax+b (simplified
     Weierstrass form) as integers and a prime integer n

    OUTPUT:

    vector of integers -- the integer coordinates [x,y] of a point
    P=E([x,y]) on E

    EXAMPLES:

    Get a point on the curve
    E: y^2=x^3+9932712830x+3288464303 modulo n=10**10+32751::

        sage: get_point(9932712830,3288464303,10**10+32751) #random
        [2839844080, 5474549347]

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    # choose x in [0,n-1] such that Q=(x^3+ax+b) mod n satisfies
    # (Q|n)!=-1
    x = int(random()*n);
    Q = x*x*x+a*x+b;
    while(kronecker(Q,n)==-1):
        x = int(random()*n);
        Q = x*x*x+a*x+b;
    # compute y^2=Q mod n
    y = int(square_root_mod_prime(Mod(Q,n)));
    return [x,y];


def goldwasser_kilian_downrun_step(n):
    """
    function performs one downrun step of Goldwasser-Kilian (one
    reduction step)

    INPUT:

     - an integer n whose primality shall be reduced to the
     primality of a smaller integer q
```

```
OUTPUT:

vector -- returns a certificate [n,a,b,m,q,P] for one reduction
step or 0 if n composite

EXAMPLES:

Reduce the primality of n=10**20+39::

    sage: goldwasser_kilian_downrun_step(10**20+39) #random
    [100000000000000000039, 314844321730698526 72L,
    395534745832825569 28L, 100000000014867206541,
    539348143913549,
    (39164891430400385024 : 86449249723524901718 : 1)]

AUTHORS:

- Georg Hahn (2010-11-27)
"""
curve_found = 0;
while(curve_found==0):
    # choose a curve over Z_n, defined by random a,b in [0,n-1]
    # such that gcd(4a^3+27b^2,n)=1
    a = 0;
    b = 0;
    while(gcd(4*a*a*a+27*b*b,n)>1):
        a = int(random()*n);
        b = int(random()*n);

    # define elliptic curve and calculate #E(a,b)(Z_n) assuming
    # n is prime (SEA=Schoof-Elkies-Atkin)
    E = EllipticCurve(GF(n),[a,b]);
    m = E.cardinality(algorithm='sea');

    # attempt to factor m=kq with q>(n^(1/4)+1)^2 probable prime
    #q = findq1(m,n);
    q = findq2(m,n,10**3);
    if(q>0):
        curve_found = 1;

O = E([0,1,0]);
point_found = 0;
while(point_found==0):
    # get Point P on E(a,b) modulo n
    P = E(get_point(a,b,n));

    # compute [m/q]P where P=[x,y]
    U = int(m/q)*P;
    if(U!=O):
        V = int(q)*U;
        if(V!=O):
            return 0;
        point_found = 1;

# output certificate
return [n,a,b,m,q,P];
```

```
def goldwasser_kilian(n):
    """
    function performs a whole downrun of Goldwasser-Kilian until the
    last q<10^6 is proven prime via is_prime

    INPUT:

     - an integer n to prove prime

    OUTPUT:

    vector -- returns a whole certificate for n if n is prime (the
    certificate consists of concatenated certificates
    [n,a,b,m,q,P] for all reduction steps), 0 if n is composite,
    -1 if an error occurred

    EXAMPLES:

    Compute a certificate for n=10**20+39 via goldwasser_kilian(n)::

        sage: goldwasser_kilian(10**20+39) #random
        [100000000000000039, 314844321730698526 72L,
        395534745832825569 28L, 100000000014867206541,
        539348143913549,
        (391648914304003850 24 : 864492497235249017 18 : 1),
        539348143913549, 155238070886498, 37712508792888,
        539348162953818, 13178619043,
        (450430088614431 : 340614625612821 : 1), 13178619043,
        8257850338, 2005441708, 13178568208,63358501,
        (4691694285 : 11267068726 : 1), 63358501, 36280326,
        60341751,63363972, 754333, (40950989 : 55530902 : 1)]

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    # test n for pseudoprimality
    if(is_pseudoprime(n)==0):
        return 0;

    # bound=10^6 for proving the last q prime via is_prime
    bound = 10**6;

    # downrun
    certificate = [];
    done = 0;
    while(done==0):
        # next step in downrun (returned variable is next part of
        # certificate or 0 if n composite)
        step = goldwasser_kilian_downrun_step(n);
        if(step==0):
            return -1;
        certificate.extend(step);
        # test last q=:n
        n = int(certificate[len(certificate)-2]);
        #print n;
```

```
        if(n<bound):
            if(is_prime(n)): done=1;
            else: return -1;

    return certificate;


def cornacchia_smith(D,p):
    """
    function finds a representation 4p=x^2+|D|y^2 for -4p<D<0 and
    D=0,1 mod 4 via Cornacchia-Smith

    INPUT:

     - a fundamental discriminant D and a prime number p

    OUTPUT:

    vector -- functions returns a solution [u,v] or 0 if unsolvable

    EXAMPLES:

    Compute a representation of p=10**20+39 using
    discriminant D=-15::

        sage: cornacchia_smith(-15,10**20+39)
        [19543688104, 1096790934]

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    # case p=2
    if(p==2):
        w = isqrt(D+8);
        if(w*w==D+8): return [w,1];
        else: return 0;
    # test for solvability
    if(kronecker(D,p)<1):
        return 0;
    # initial square root
    x0 = int(square_root_mod_prime(Mod(D,p)));
    if((x0%2)!=(D%2)):
        x0 = p-x0;
    # initialize Euklid chain
    a = 2*p;
    b = x0;
    c = 2*isqrt(p);
    temp = 0;
    # Euklid chain
    while(b>c):
        temp = b;
        b = a%b;
        a = temp;
    # final report
    t = 4*p-b*b;
```

```
    if(t%(abs(D))!=0):
        return 0;
    w = isqrt(int(t/(abs(D))));
    if(w*w!=int(t/(abs(D)))):
        return 0;
    return [b,w];


def is_cube(x,p):
    """
    function tests whether x is a cube mod p or not

    INPUT:

     - an integer x to test and a prime number p

    OUTPUT:

    integer -- functions returns 1 if x is a cube mod p
    and 0 otherwise

    EXAMPLES:

    Test x=3 in F_p where p=23::

        sage: is_cube(3,23)
        1

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    a = 0;
    while(a<p):
        if((a*a*a)%p==x%p):
            return 1;
        a = a+1;
    return 0;


def curve_parameters(D,p,order):
    """
    function returns curve parameters [a,b] for an elliptic curve
    E:y^2=x^3+ax+b having a given order(D)=order

    INPUT:

     - a fundamental discriminant D, a prime number p and an integer
     curve order "order"

    OUTPUT:

    vector of integers -- functions returns the curve parameters
    [a,b] of E in simplified Weierstrass form
```

```
EXAMPLES:

For D=-15 and p=10**20+39, get a curve with order
99999999980456311936::

    sage:
    curve_parameters(-15,10**20+39,99999999980456311936) #random
    [28153013400593936 60, 567309247109926568 44]

NOTES:

it is assumed that (D|p)=1 and that there exists a Cornacchia-
Smith solution for (D,p)

AUTHORS:

- Georg Hahn (2010-11-27)
"""
#orders = [p+1+u, p+1-u, p+1+2*v, p+1-2*v];
#orders = [p+1+u, p+1-u, p+1+(u+3*v)/2, p+1+(u-3*v)/2,
#         p+1-(u+3*v)/2, p+1-(u-3*v)/2];
#orders = [p+1+u, p+1-u];

# find a quadratic nonresidue g (mod p)
found = 0;
while(found==0):
    g = int(random()*p);
    while(kronecker(g,p)!=-1):
        g = int(random()*p);
    found = 1;
    if((p%3==1) and (power_mod(g,int((p-1)/3),p)==1)):
        found = 0;
    #if(D==-3):
    #    if(is_cube(g,p)==1):
    #        found = 0;

# curve parameters
param = [];
if(D==-4):
    param = [p-1,0,(-g)%p,0,(-g*g)%p,0,(-g*g*g)%p,0];
if(D==-3):
    param = [0,p-1,0,(-g)%p,0,(-g*g)%p,0,(-g*g*g)%p,0,
             (-g*g*g*g)%p,0,(-g*g*g*g*g)%p];
if(D<-4):
    # compute Hilbert class polynomial
    S = hilbert_class_polynomial(D);
    # get one root in F_p (first one in list)
    j = int(S.roots(GF(p))[0][0]);
    #c = j*((j-1728).inverse_mod(p));
    c = j*xgcd(j-1728,p)[1];
    r = (-3*c)%p;
    s = (2*c)%p;
    param = [r,s,(r*g*g)%p,(s*g*g*g)%p];

# last step: determine which coefficients in "param" belong to
# the given order
```

```
    # select random points on all [a,b] curves and rule out curves
    # until only one remains
    while(len(param)>2):
        param_copy = [];
        for i in range(int(len(param)/2)):
            # the parameter pair [a,b] for each curve is now
            # param[2i] and param[2i+1]
            a = param[2*i];
            b = param[2*i+1];
            E = EllipticCurve(GF(p),[a,b]);
            O = E([0,1,0]);

            # get Point P on E(a,b) modulo p
            P = E(get_point(a,b,p));

            # rule curve out if possible, otherwise keep parameter
            # pair for next run
            if(int(order)*P==O):
                param_copy.extend([a,b]);

        # copy back
        param = [];
        param.extend(param_copy);

    return param;


def oddpart(n):
    """
    function returns the odd part of a number n

    INPUT:

     - an integer n

    OUTPUT:

    integer -- the odd part of n=2^e * d where d is odd

    EXAMPLES:

    Odd part of 6::

        sage: odd_part(6)
        3

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    while(n%2==0):
        n = n>>1;
    return n;
```

```
def squarefree(n):
    """
    function returns 1 if n is squarefree and 0 if it is not (n>0)

    INPUT:

     - an integer n>0

    OUTPUT:

    integer -- 1 if n is squarefree and 0 if it is not

    EXAMPLES:

    The integer 6 is squarefree::

        sage: squarefree(6)
        1

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    if(n%4==0):
        return 0;
    x = 3;
    w = isqrt(n);
    while(x<=w):
        if(n%(x*x)==0):
            return 0;
        x = x+2;
    return 1;


def getD(p):
    """
    function looks for a suitable fundamental discriminant D s.t.
    one of the possible curve orders factors as m=kq

    INPUT:

     - a prime p

    OUTPUT:

    vector of integers -- function returns [D,m,q] when D is found
    where m is the order of one of the corresponding
    curves to D which factors as m=kq, q>(n^(1/4)+1)^2 probable
    prime

    EXAMPLES:

    Get a fundamental discriminant for p=10**20+39::

        sage: getD(10**20+39)
        [-3, 99999999982060129348, 964019216509L]
```

```
    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    D = -3;
    done = 0;
    while(done==0):
        if((-D)%16 in [3,4,7,8,11,15]):
            if(squarefree(oddpart(-D))==1):
                # D is a fundamental discriminant
                # find 4n=u^2+|D|v^2 via Cornacchia-Smith
                cs = cornacchia_smith(D,p);
                if(cs!=0):
                    u = cs[0];
                    v = cs[1];
                    # get possible curve orders
                    orders = [];
                    if(D==-4):
                        orders = [p+1+u, p+1-u, p+1+2*v, p+1-2*v];
                    if(D==-3):
                        o1 = u+3*v;
                        o1 = o1>>1;
                        o2 = u-3*v;
                        o2 = o2>>1;
                        orders = [p+1+u, p+1-u, p+1+o1, p+1+o2,
                                  p+1-o1, p+1-o2];
                    if(D<-4):
                        orders = [p+1+u, p+1-u];
                    # factor orders and return [D,m,q] when
                    # successful where m=orders[i]
                    for i in range(len(orders)):
                        q = findq2(orders[i],p,10**3);
                        if(q>0):
                            return [D,orders[i],q];
        D = D-1;


def atkin_morain_downrun_step(n):
    """
    function performs one downrun step of Goldwasser-Kilian-Atkin-
    Morain (one reduction step)

    INPUT:

     - an integer n whose primality shall be reduced to the
     primality of a smaller integer q

    OUTPUT:

    vector -- returns a certificate [n,a,b,m,q,P] for one reduction
    step or 0 if n composite
```

```
    EXAMPLES:

    Reduce the primality of n=10**20+39::

        sage: atkin_morain_downrun_step(10**20+39) #random
        [10000000000000000039, 0, 10000000000000000038,
        99999999982060129348, 964019216509,
        (66730941426430877696 : 87321777101668542978 : 1)]

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    # find a suitable discriminant and the corresponding order m=kq,
    # q>(n^(1/4)+1)^2 probable prime
    [D,m,q] = getD(n);

    # get curve parameters and define elliptic curve
    [a,b] = curve_parameters(D,n,m);
    E = EllipticCurve(GF(n),[a,b]);
    O = E([0,1,0]);

    point_found = 0;
    while(point_found==0):
        # get Point P on E(a,b) modulo n
        P = E(get_point(a,b,n));

        # compute [m/q]P where P=[x,y]
        U = int(m/q)*P;
        if(U!=O):
            V = int(q)*U;
            if(V!=O):
                return 0;
            point_found = 1;

    # output certificate
    return [n,a,b,m,q,P];


def atkin_morain(n):
    """
    function performs a whole downrun of GK-Atkin-Morain until the
    last q<10^6 is proven prime via is_prime

    INPUT:

     - an integer n to prove prime

    OUTPUT:

    vector -- returns a whole certificate for n if n is prime (the
    certificate consists of concatenated certificates
    [n,a,b,m,q,P] for all reduction steps), 0 if n is composite, -1
    if an error occurred
```

```
EXAMPLES:

Compute a certificate for n=10**20+39 via atkin_morain(n)::

    sage: atkin_morain(10**20+39) #random
    [10000000000000000039, 0, 10000000000000000038,
    99999999982060129348, 964019216509,
    (722136l4964914421760 : 13917788256052916206 : 1),
    964019216509, 0, 591506100901L, 964020818668, 241005204667,
    (213575224777 : 500479927639 : 1), 241005204667, 0,
    44160005370, 241006096857, 626011,
    (120027315517 : 190726073430 : 1)]

AUTHORS:

- Georg Hahn (2010-11-27)
"""
# test n for pseudoprimality
if(is_pseudoprime(n)==0):
    return 0;

# bound=10^6 for proving the last q prime via is_prime
bound = 10**6;

# downrun
certificate = [];
done = 0;
while(done==0):
    # next step in downrun (returned variable is next part of
    # certificate or 0 if n composite)
    step = atkin_morain_downrun_step(n);
    if(step==0):
        return -1;
    certificate.extend(step);
    # test last q=:n
    n = int(certificate[len(certificate)-2]);
    #print n;
    if(n<bound):
        if(is_prime(n)): done=1;
        else: return -1;

return certificate;
```

```
def ecpp(n):
    """
    ECPP refers to the algorithm of Atkin-Morain (Goldwasser-Kilian-
    Atkin-Morain)

    INPUT:

     - an integer n to prove prime

    OUTPUT:

    vector -- returns a whole certificate for n if n is prime (the
    certificate consists of concatenated certificates
    [n,a,b,m,q,P] for all reduction steps), 0 if n is composite, -1
    if an error occurred

    EXAMPLES:

    Compute a certificate for n=10**20+39 via ecpp(n)::

        sage: ecpp(10**20+39) #random
        [100000000000000000039, 0, 100000000000000000038,
        99999999982060129348, 964019216509,
        (72213614964914421760 : 13917788256052916206 : 1),
        964019216509, 0, 591506100901L, 964020818668, 241005204667,
        (213575224777 : 500479927639 : 1), 241005204667, 0,
        44160005370, 241006096857, 626011,
        (120027315517 : 190726073430 : 1)]

    AUTHORS:

    - Georg Hahn (2010-11-27)
    """
    return atkin_morain(n);


def verify_certificate(c):
    """
    function verifies a certificate previously computed by
    goldwasser_kilian(n), atkin_morain(n) or ecpp(n)

    INPUT:

     - a certificate c (for a number n) previously computed by
     goldwasser_kilian(n), atkin_morain(n) or ecpp(n)

    OUTPUT:

    integer -- returns 1 if verification was successful (i.e. n is
    prime) and 0 if certificate is invalid
```

```
EXAMPLES:

Verify a certificate computed by ecpp::

    sage: c = ecpp(10**20+39);
    sage: verify_certificate(c)
    1

AUTHORS:

- Georg Hahn (2010-11-27)
"""

# certificate must be non-empty and length of certificate must
# be a multiple of 6
if((len(c)==0) or (len(c)%6>0)):
    return 0;

# initialization
q = c[0];

# check every step
for i in range(int(len(c)/6)):
    # certificate consists of blocks (n,a,b,m,q,P)
    n = c[6*i];
    # next n must be last q
    if(n!=q):
        return 0;
    a = c[6*i+1];
    b = c[6*i+2];
    m = c[6*i+3];
    q = c[6*i+4];
    P = c[6*i+5];

    # check gcds
    if(gcd(n,6)>1):
        return 0;
    if(gcd(4*a*a*a+27*b*b,n)>1):
        return 0;
    # check q>(isqrt(isqrt(n)+1)+1+1)**2
    if(q<=(isqrt(isqrt(n)+1)+1+1)**2):
        return 0;
    # compute U=[m/q]P and check U!=O and V=[q]U=O
    E = EllipticCurve(GF(n),[a,b]);
    O = E([0,1,0]);
    U = int(m/q)*P;
    if(U==O):
        return 0;
    V = int(q)*U;
    if(V!=O):
        return 0;

# primality of the last q completes the check
return int(is_prime(q));
```