3.

有限体上の楕円曲線に関連した計算問題

安田 雅哉（九州大学マス・フォア・インダストリ研究所）

3.1 はじめに

本稿では，有限体上の楕円曲線の基本的な性質を紹介したのち，有限体上の楕円曲線の位数計算である Schoof アルゴリズム [Sch85, Sch95] を紹介する．現在までに，Schoof アルゴリズムは Elkies や Atkin らによって高速化改良され，SEA（Schoof-Elkies-Atkin）アルゴリズムとして楕円曲線暗号で利用するための楕円曲線パラメータ選択時などで利用されてきた（楕円曲線暗号については [BSS99, Coh05] などを参照）．本稿では，Schoof アルゴリズムの処理概要を紹介するとともに，そのアルゴリズムを数式処理 PARI [PARI]（version 2.9.2）で実装し，その実装ソースコードも付録として示しておく（C 言語のライブラリとして利用できる PARI ライブラリをインストールしたのち，付録の実装ソースコードをコンパイルして利用して頂ければ幸いです）．

3.2 数学的準備

本節では，有限体 $\mathbb{F}_p$ 上の楕円曲線 E の群位数 $\#E(\mathbb{F}_p)$ を計算する上で必要となる数学的な基本性質を紹介する．本稿では，簡単のため 5 以上の素数 p に対する有限体 $\mathbb{F}_p$ のみを扱う．

3.2.1 Hasse の定理と Frobenius 写像

素数 $p \geq 5$ を固定し，次の Weierstrass 方程式で定義される有限体 $\mathbb{F}_p$ 上の楕円曲線 E を考える：

$$E : y^2 = x^3 + ax + b \ (a, b \in \mathbb{F}_p, \Delta = -16(4a^3 + 27b^2) \neq 0). \tag{3.1}$$

定理 3.2.1 (Hasse). 有限体 $\mathbb{F}_p$ 上定義された楕円曲線 E の $\mathbb{F}_p$-有理点全体がなす群 $E(\mathbb{F}_p)$ の位数 $\#E(\mathbb{F}_p)$ について, 不等式

$$|\#E(\mathbb{F}_p) - p - 1| \leq 2\sqrt{p}$$

が成り立つ.

本節では，上記の Hasse の定理の証明を与えることを目指す．まず Hasse の定理の証明において中心的な役割を果たす Frobenius 写像を紹介する；有限体 $\mathbb{F}_p$ 上の楕円曲線 E 上の Frobenius 写像 φ を

$$\varphi : E(\bar{\mathbb{F}}_p) \longrightarrow E(\bar{\mathbb{F}}_p), \ (x, y) \mapsto (x^p, y^p), \ \mathcal{O} \mapsto \mathcal{O}$$

と定義する ($(y^p)^2 = (x^3 + ax + b)^p = (x^p)^3 + ax^p + b$ が成り立つので，$(x^p, y^p) \in E$ となることに注意)．ただし，$\mathcal{O} \in E(\bar{\mathbb{F}}_p)$ を無限遠点とする (無限遠点 $\mathcal{O}$ は群 $E(\bar{\mathbb{F}}_p)$ の零元).

楕円曲線上の自己準同型に関する基本性質

ここでは，Hasse の定理の証明で必要な基本性質をまとめておく．具体的には，有限体 $\mathbb{F}_p$ 上定義された楕円曲線 E 上の自己準同型環 $\mathrm{End}(E)$(以下で定義) に関する基本性質を紹介する．

定義 3.2.2. 群 $E(\bar{\mathbb{F}}_p)$ 上の自己準同型 $\phi : E(\bar{\mathbb{F}}_p) \longrightarrow E(\bar{\mathbb{F}}_p)$ の元全体を $\mathrm{End}(E)$ で表す．特に，E 上の m-倍写像 $[m]$ と Frobenius 写像 φ は $\mathrm{End}(E)$ の元とみなすことができる．任意の 2 つの自己準同型 $\phi, \psi \in \mathrm{End}(E)$ に対して，$(\phi + \psi)(P) := \phi(P) + \psi(P)$ と $(\phi\psi)(P) := \phi(\psi(P))$ の演算を定めることで，集合 $\mathrm{End}(E)$ に環の構造を定義することができる．自己準同型環 $\mathrm{End}(E)$ は零因子を持たない標数 0 の環であるため [Sil86, Proposition 4.2(c) in Chapter III]，楕円曲線 E 上のスカラー倍写像から定まる環準同型写像

$$[\cdot] : \mathbb{Z} \longrightarrow \mathrm{End}(E), \ m \mapsto [m] \tag{3.2}$$

は単射である．

さらに，有限体 $\mathbb{F}_p$ 上定義された楕円曲線 E 上の有理関数全体がなす体を $\bar{\mathbb{F}}_p(E)$ とする．0 でない自己準同型 $\phi \in \mathrm{End}(E)$ は，体の有限次拡大

$$\phi^* : \bar{\mathbb{F}}_p(E) \longrightarrow \bar{\mathbb{F}}_p(E) \tag{3.3}$$

を導く．その体の拡大次数を自己準同型 ϕ の次数と定める：$\deg(\phi) := \left[\bar{\mathbb{F}}_p(E) : \phi^*(\bar{\mathbb{F}}_p(E))\right]$ (便宜上，0-自己準同型に対しては $\deg[0] = 0$ と定める)．さらに，ϕ^* で得られる体の拡大が分離的であるとき，自己準同型 ϕ は分離的であるという．

例 3.2.3. 方程式 (3.1) で定義される楕円曲線 E 上の有理関数体 $\bar{\mathbb{F}}_p(E)$ は $\bar{\mathbb{F}}_p(x,y)/(y^2 - x^3 - ax - b)$ と表せる．楕円曲線 E 上の Frobenius 写像 φ から，体の有限次拡大

$$\begin{aligned}\varphi^* : \bar{\mathbb{F}}_p(x,y)/(y^2 - x^3 - ax + b) &\longrightarrow \bar{\mathbb{F}}_p(x,y)/(y^2 - x^3 - ax - b),\\ x \mapsto x^p, \quad y &\mapsto y^p\end{aligned}$$

が導かれる．ここで，

$$\varphi^*\left(\bar{\mathbb{F}}_p(E)\right) = \bar{\mathbb{F}}_p(x^p, y^p)/(y^2 - x^3 - ax - b) \subset \bar{\mathbb{F}}_p(x,y)/(y^2 - x^3 - ax - b) \tag{3.4}$$

に注意する．また，$y \cdot y^p = (y^2)^{\frac{p+1}{2}} = \left(x^3 + ax + b\right)^{\frac{p+1}{2}}$ に注意すると，

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

が成り立つので，体の拡大 (3.4) は x のみに依存することが分かる．体の拡大 (3.4) は次数 p の方程式 $X^p - x^p = (X - x)^p = 0$ で定まるので，$\deg\varphi = p$ であることが分かる．さらに体の拡大 (3.4) は明らかに分離的でないので，Frobenius 写像 φ は分離的でないことが分かる．

定義 3.2.4. 楕円曲線 E に対して，$\bar{\mathbb{F}}_p(E)$-ベクトル空間

$$\Omega_E = \left\langle df \mid f \in \bar{\mathbb{F}}_p(E)\right\rangle_{\bar{\mathbb{F}}_p(E)}$$

を E 上の**有理微分形式のなす空間** (the space of meromorphic differential forms on E) と呼ぶ．ただし，微分形式 $df \in \Omega_E$ は形式的に次の 3 条件を満たす：(i) 任意の $f, g \in \bar{\mathbb{F}}_p(E)$ に対し，$d(f+g) = df + dg$. (ii) 任意の $f, g \in \bar{\mathbb{F}}_p(E)$ に対し，$d(fg) = fdg + gdf$. (iii) 任意の $a \in \bar{\mathbb{F}}_p$ に対し，$da = 0$.

各自己準同型 $\phi \in \mathrm{End}(E)$ は (3.3) のように体の拡大 ϕ^* を導き， さらに有理微分形式のなす空間上の $\bar{\mathbb{F}}_p(E)$-線型写像

$$\phi^* : \Omega_E \longrightarrow \Omega_E,\ \phi^*\left(\sum f_i dg_i\right) = \sum \left(\phi^* f_i\right) d\left(\phi^* g_i\right)$$

を導く．特に，自己準同型 ϕ が分離的であることと，Ω_E 上の ϕ^* が 0-写像でないことが同値である [Sil86, Proposition 4.2(c) in Chapter II].

方程式 (3.1) で定義される有限体 $\mathbb{F}_p$ 上の楕円曲線 E において，

$$\omega = \frac{dx}{y} \in \Omega_E$$

を不変微分 (invariant differential) と呼ぶ．不変微分 ω は楕円曲線 E 上のすべての点で正則で，任意の m-倍写像 $[m] \in \mathrm{End}(E)$ に対して，$[m]^*\omega = m\omega$ を満たす [Sil86, Corollary 5.3 in Chapter III].

命題 3.2.5. $m, n \in \mathbb{Z}$ とする．方程式 (3.1) で定義される有限体 $\mathbb{F}_p$ 上の楕円曲線 E 上の自己準同型 $\phi := m + n\varphi \in \mathrm{End}(E)$ は，$p \nmid m$ ならば分離的である．特に，$1 - \varphi \in \mathrm{End}(E)$ は分離的な自己準同型である．さらに，分離的な自己準同型 ϕ に対して，$\#\ker(\phi) = \deg(\phi)$ が成り立つ．

証明. 自己準同型 ϕ が分離的であることを示すには，$\phi^*\omega \neq 0 \in \Omega_E$ であることを確かめれば良い．任意の 2 つの自己準同型 $\psi_1, \psi_2 \in \mathrm{End}(E)$ に対して，$(\psi_1 + \psi_2)^*\omega = \psi_1^*\omega + \psi_2^*\omega$ が成り立つので [Sil86, Theorem 5.2 in Chapter III]，$\phi^*\omega = (m + n\varphi)^*\omega = [m]^*\omega + \varphi^*\left([n]^*\omega\right) = m\omega + n\varphi^*(\omega)$ が成り立つ．また

$$\varphi^*\omega = \frac{dx^p}{y^p} = \frac{px^{p-1}dx}{y^p} = 0$$

より，$\phi^*\omega = m\omega$ となる．よって，$p \nmid m$ のとき $\phi^*\omega \neq 0$ より，自己準同型 ϕ は分離的であることが分かる．一方，分離的な自己準同型 ϕ に対して，ほとんどすべての点 $Q \in E(\bar{\mathbb{F}}_p)$ に対し $\#\phi^{-1}(Q) = \deg\phi$ が成り立つ [Sil86, Proposition 2.6 (b) in Chapter II]．ここで，$\#\phi^{-1}(Q) = \deg\phi$ を満たす点 $Q \in E(\bar{\mathbb{F}}_p)$ を 1 つ固定する．自己準同型 ϕ は 0-写像ではないので ϕ は全射であり [Sil86, Theorem 2.3 in Chapter II]，$\phi(R) = Q$ となる点 $R \in E(\bar{\mathbb{F}}_p)$ が存在する．さらに，自己準同型 ϕ は $E(\bar{\mathbb{F}}_p)$ 上の準同型写像なので，集合としての写像 $\phi^{-1}(\mathcal{O}) \longrightarrow \phi^{-1}(Q),\ P \mapsto P + R$ が定義でき，構成の仕方から全単射であることは明らか．よって，$\#\ker(\phi) = \#\phi^{-1}(\mathcal{O}) = \#\phi^{-1}(Q) = \deg\phi$ が成り立つ． □

定義 3.2.6. アーベル群 A 上の関数 $d : A \to \mathbb{R}$ が，以下の 2 条件を満たすとき d は 2 次形式 (quadratic form) と呼ぶ：

(i) 任意の元 $\alpha \in A$ に対して，$d(\alpha) = d(-\alpha)$ が成り立つ．
(ii) $A \times A \to \mathbb{R}, (\alpha, \beta) \mapsto d(\alpha + \beta) - d(\alpha) - d(\beta)$ が双線型写像となる．

さらに，以下の 2 条件を満たすとき，2 次形式 d は**正定値** (positive definite) であるという：

(iii) 任意の元 $\alpha \in A$ に対して，$d(\alpha) \geq 0$.
(iv) $d(\alpha) = 0 \Longleftrightarrow \alpha = 0$.

命題 3.2.7. 楕円曲線 E の自己準同型環 $\mathrm{End}(E)$ 上の次数写像 $\deg : \mathrm{End}(E) \longrightarrow \mathbb{Z}$ は正定値 2 次形式である．

証明. 自己準同型環 $\mathrm{End}(E)$ 上のペアリング $\langle \phi, \psi \rangle := \deg(\phi+\psi) - \deg(\phi) - \deg(\psi)$ が双線型であることを示せばよい．次数 $m = \deg(\phi)$ を持つ任意の自己準同型 $\phi \in \mathrm{End}(E)$ に対して，$\phi \circ \hat{\phi} = \hat{\phi} \circ \phi = [m] \in \mathrm{End}(E)$ を満たす双対自己準同型 $\hat{\phi}$ が唯一つ存在する [Sil86, Theorems 6.1 and 6.2 in Chapter III]．また任意の 2 つの自己準同型 $\phi, \psi \in \mathrm{End}(E)$ に対して，$\widehat{\phi+\psi} = \hat{\phi} + \hat{\psi}$ が成り立つ [Sil86, Theorem 6.2 (c) in Chapter III]．上記で定義したペアリングに単射準同型写像 (3.2) を適用すると，

$$\begin{aligned}[\langle \phi, \psi \rangle] &= [\deg(\phi+\psi)] - [\deg(\phi)] - [\deg(\psi)] \\ &= (\widehat{\phi+\psi}) \circ (\phi+\psi) - \hat{\phi} \circ \phi - \hat{\psi} \circ \psi = \hat{\phi} \circ \psi + \hat{\psi} \circ \phi \end{aligned} \tag{3.5}$$

が成り立つ．(3.5) は ϕ と ψ の両方において線型であるため，ペアリングは双線型写像であることが分かる． □

Hasse の定理の証明

前節で定理 3.2.1 を証明する準備がほぼ整った．定理 3.2.1 の証明の前に，下記の補題を示しておく：

補題 3.2.8. A をアーベル群とし，$d : A \longrightarrow \mathbb{Z}$ を正定値 2 次形式とする．このとき，任意の元 $\psi, \phi \in A$ に対して，

$$|d(\psi-\phi) - d(\phi) - d(\psi)| \leq 2\sqrt{d(\phi)d(\psi)} \tag{3.6}$$

が成り立つ．

証明. 元 $\psi, \phi \in A$ に対して，$L(\psi, \phi) = d(\psi-\phi) - d(\phi) - d(\psi)$ とおく．定義 3.2.6 の (i) と (ii) により L は双線型写像となり，任意の $m, n \in \mathbb{Z}$ に対して

$$mnL(\psi, \phi) = L(m\psi, n\phi) = d(m\psi - n\phi) - d(m\psi) - d(n\phi) \tag{3.7}$$

が成り立つ．さらに d が正定値であることに注意すると，

$$-2d(m\psi) = L(m\psi, m\psi) = m^2 L(\psi, \psi) = -2m^2 d(\psi)$$

より，$d(m\psi) = m^2(\psi)$ が成り立つ．同様に $d(n\phi) = n^2 d(\phi)$ が成り立つので，等式 (3.7) より $0 \leq d(m\psi - n\phi) = m^2 d(\psi) + mnL(\psi, \phi) + n^2 d(\phi)$ が成り立つ (左辺の不等号は定義 3.2.6(iii) より)．ここで $m = -L(\psi, \phi)$, $n = 2d(\psi)$ ととると，上記の不等式から $d(\psi)\left(4d(\psi)d(\phi) - L(\psi, \phi)^2\right) \geq 0$ が成立する．これより，$d(\psi) \neq 0$ の場合 (つまり $\psi \neq 0$ の場合)，$L(\psi, \phi)^2 \leq 4d(\psi)d(\phi)$ が成り立つので，不等式 (3.6) が成立する．一方，$\psi = 0$ の場合は，不等式 (3.6) は明らかに成立する． □

以下で Hasse の定理の証明を与える：

Proof of 定理 *3.2.1.* 楕円曲線 E 上の任意の点 P に対して，$P \in E(\mathbb{F}_p) \Longleftrightarrow \phi(P) = P$ が成り立つので, $E(\mathbb{F}_p) = \ker(1-\phi)$ が成立する．さらに, 命題 3.2.5 から $1-\phi \in \mathrm{End}(E)$ は分離的なので，$\#E(\mathbb{F}_p) = \#\ker(1-\phi) = \deg(1-\phi)$ が成り立つ．また命題 3.2.7 から次数写像 $\deg : \mathrm{End}(E) \longrightarrow \mathbb{Z}$ は正定値 2 次形式で，補題 3.2.8 より $|\#E(\mathbb{F}_p) - \deg(\phi) - \deg(1)| \leq 2\sqrt{\deg(\phi)\deg(1)}$ が成立する．ここで，$\deg(\phi) = p, \deg(1) = 1$ に注意すれば，Hasse の定理が成立することが分かる． □

3.2.2 楕円曲線に付随する等分多項式

ここでは，次節以降で紹介する Schoof アルゴリズムで利用する楕円曲線に付随する等分多項式を紹介する．方程式 (3.1) で定義される有限体 $\mathbb{F}_p$ 上の楕円曲線 E に付随する 2 変数多項式 $\psi_m = \psi_m(x,y) \in \mathbb{F}_p[x,y]$ を次のように定める [Sil86, Exercise 3.7 in Chapter III]：

$$\begin{cases} \psi_0 = 0,\ \psi_1 = 1,\ \psi_2 = 2y, \\ \psi_3 = 3x^4 + 6ax^2 + 12bx - a^2, \\ \psi_4 = 4y\left(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3\right), \\ \psi_{2m+1} = \psi_{m+2}\psi_m^3 - \psi_{m-1}\psi_{m+1}^3 \quad (m \geq 2), \\ \psi_{2m} = \dfrac{\psi_m}{2y}\left(\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2\right) \quad (m \geq 3). \end{cases}$$

$m \geq 3$ に対し ψ_{2m} の分子は y で割れるので，ψ_{2m} は $\mathbb{F}_p[x,y]$ の元となることに注意する．このとき，整数 $m \geq 2$ と楕円曲線 E 上の点 $P = (x,y) \in E(\overline{\mathbb{F}}_p) \setminus E[m]$ に対して,

$$[m]P = \left(x - \frac{\psi_{m-1}\psi_{m+1}}{\psi_m^2}, \frac{\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2}{4y\psi_m^3}\right) \tag{3.8}$$

が成り立つ．さらに，$\mathcal{O} \neq P = (x_P, y_P) \in E(\overline{\mathbb{F}}_p)$ と $m \geq 1$ に対して，$P \in E[m] \Longleftrightarrow \psi_m(x_P, y_P) = 0$ が成立する．

上記で定めた 2 変数多項式 $\psi_m(x,y) \in \mathbb{F}_p[x,y]$ に対し，m-等分多項式 f_m を次のように定める：$f_1 = \psi_1 = 1$ とし，$m \geq 2$ に対しては

$$f_m = \begin{cases} \psi_m & (m : \text{奇数}), \\ \psi_m/\psi_2 & (m : \text{偶数}). \end{cases}$$

すると，f_m は x に関する 1 変数多項式として表現できる．実際，$f_m = f_m(x) \in \mathbb{F}_p[x]$ は次のように帰納的に計算できる (f_3, f_4 が $\mathbb{F}_p[x]$ の元なので,

任意の f_m が $\mathbb{F}_p[x]$ の元となることが分かる)：

$$\begin{aligned} f_0 = 0,\ f_1 = 1,\ f_2 = 1,\ f_3 = \psi_3,\ f_4 = \psi_4/\psi_2, \\ f_{2m+1} = \begin{cases} f_{m+2}f_m^3 - F^2 f_{m-1}f_{m+1}^3 & (m \geq 3 : \text{奇数}), \\ F^2 f_{m+2}f_m^3 - f_{m-1}f_{m+1}^3 & (m \geq 2 : \text{偶数}), \end{cases} \\ f_{2m} = (f_{m+2}f_{m-1}^2 - f_{m-2}f_{m+1}^2)f_m \quad (m \geq 3). \end{aligned}$$

ただし, $F = 4(x^3+ax+b) \in \mathbb{F}_p[x]$ とする．このとき, 任意の $P = (x_P, y_P) \in E(\overline{\mathbb{F}}_p) \setminus E[2]$ と $m \geq 3$ に対して，$P \in E[m] \iff f_m(x_P) = 0$ が成り立つ．m-等分多項式 $f_m(x)$ は PARI/GP [PARI] の elldivpol のコマンドで求めることが可能である (ただし elldivpol コマンドでは，偶数 m に対し $\psi_2\psi_m$，奇数 m に対し ψ_m に対応する $\mathbb{F}_p[x]$ の多項式が出力される).

3.3 Schoof アルゴリズムの紹介

本節では，方程式 (3.1) で定義される有限体 $\mathbb{F}_p$ 上の楕円曲線 E が与えられたとき，E 上の $\mathbb{F}_p$-有理点の個数 $\#E(\mathbb{F}_p)$ を効率的に計算可能とする Schoof アルゴリズム [Sch85, Sch95] と簡単な数値例を紹介する．

3.3.1 計算戦略と全体処理概要（Algorithm 1）

有限体 $\mathbb{F}_p$ 上の楕円曲線 E に対して，$t = p + 1 - \#E(\mathbb{F}_p)$ とおく ($t \in \mathbb{Z}$ はトレースと呼ばれる)．Hasse の定理により $|t| \leq 2\sqrt{p}$ であることは分かるが，具体的な t の値は得られない．具体的な t の値を計算するために，まず Schoof アルゴリズムでは

$$M := \prod_i \ell_i \geq 4\sqrt{p} \tag{3.9}$$

を満たす複数の小さな素数 $\ell_i \neq p$ に対して $t \bmod \ell_i \in \mathbb{Z}/\ell_i\mathbb{Z}$ を求める．中国人剰余定理 (Chinese Remaider Theorem) により $t \bmod M \in \mathbb{Z}/M\mathbb{Z}$ が得られ，条件 (3.9) から正しい $t \in \mathbb{Z}$ の値が得られる (Hasse の定理より $|t| \leq 2\sqrt{p}$ を満たすので，$-2\sqrt{p} \leq t_0 \leq 2\sqrt{p}$ かつ $t_0 \equiv t \bmod M$ を満たす整数 t_0 がトレース t と一致)．固定した素数 $\ell \neq p$ に対して，$t \bmod \ell$ を求める計算戦略は以下である：有限体 $\mathbb{F}_p$ 上の楕円曲線 E 上の Frobenius 写像 $\varphi \in \mathrm{End}(E)$ は，自己準同型環 $\mathrm{End}(E)$ 上で $\varphi^2 - t\varphi + p = 0$ を満たす [Sil86, Chapter 5]．特に，無限遠点 $\mathcal{O}$ と異なる任意の ℓ-等分点 $P = (x, y) \in E[\ell]$ に対して，

$$\begin{aligned} &\varphi^2(P) - [t]\varphi(P) + [p]P = \mathcal{O} \\ \iff &(x^{p^2}, y^{p^2}) - [t](x^p, y^p) + [p](x, y) = \mathcal{O} \end{aligned} \tag{3.10}$$

を満たす．ここで，楕円曲線 E 上の点 $\varphi(P)$ は ℓ-等分点であるので $[\ell]\varphi(P) = \mathcal{O}$ を満たすことに注意する．そこで，$0 \leq t_\ell, p_\ell \leq \ell - 1$ かつ $t_\ell \equiv t \bmod \ell$,

Algorithm 1 Schoof アルゴリズムの処理全体概要

Input: 有限体 $\mathbb{F}_p$ 上の楕円曲線 $E : y^2 = f(x) = x^3 + ax + b$ ($p \geq 5$ と仮定)

Output: 楕円曲線 E 上の $\mathbb{F}_p$-有理点の個数 $\#E(\mathbb{F}_p)$

1: $M \leftarrow 1$, $\ell \leftarrow 2$, $A \leftarrow \emptyset$
2: **while** $M < 4\sqrt{p}$ **do**
3: $p_\ell \leftarrow p \bmod \ell$ (p_ℓ は $0 \leq p_\ell \leq \ell - 1$ を満たす整数)
4: **for** $n = 0, 1, 2, \ldots, \ell - 1$ **do**
5: (3.12) で定義される環 R_ℓ 上で関係式 (3.11) が成り立つか確認 (成り立つ場合は for ループから抜ける)
6: **end for**
7: $t_\ell \leftarrow n$, $A \leftarrow A \cup \{(t_\ell, \ell)\}$
8: $M \leftarrow M \times \ell$, $\ell \leftarrow \texttt{nextprime}(\ell + 1)$ /* 次の素数 ℓ を選択 */
9: **end while**
10: 集合 $A = \{(t_\ell, \ell)\}_{\ell \geq 2}$ 上で中国人剰余定理を適用し，$-2\sqrt{p} \leq t_0 \leq 2\sqrt{p}$ かつ，すべての ℓ に対して $t_0 \equiv t_\ell \bmod \ell$ を満たす整数 t_0 を計算
11: $p + 1 - t_0$ を出力 /* 位数 $\#E(\mathbb{F}_p)$ に一致 */

$p_\ell \equiv p \bmod \ell$ を満たす整数を t_ℓ, p_ℓ とする．すると，関係式 (3.10) は

$$(x^{p^2}, y^{p^2}) - [t_\ell](x^p, y^p) + [p_\ell](x, y) = \mathcal{O}$$

と書き直すことができる．そこで具体的な $t_\ell \in \mathbb{Z}$ を求めるために，Schoof アルゴリズムでは適当な ℓ-等分点 $P = (x, y) \in E[\ell] \setminus \{\mathcal{O}\}$ を選び，$n = 0, 1, \ldots, \ell - 1$ の順に

$$(x^{p^2}, y^{p^2}) + [p_\ell](x, y) = [n](x^p, y^p) \tag{3.11}$$

の関係式が成立するか計算していく (p_ℓ は p と ℓ から計算可能)．関係式 (3.11) が成立する n が t_ℓ と一致する．特に，任意の ℓ-等分点 $P = (x, y) \in E[\ell] \setminus \{\mathcal{O}\}$ は $f_\ell(x) = 0$ かつ $y^2 = f(x) = x^3 + ax + b$ を満たすので，関係式 (3.11) の計算として

$$R_\ell = \mathbb{F}_p[x, y] / \left(f_\ell(x), y^2 - f(x)\right) \tag{3.12}$$

の環上で楕円曲線 E の群演算を行えば良い．

Algorithm 1 に，Schoof アルゴリズム [Sch85, Sch95] の全体処理概要を示しておく．Algorithm 1 の Step 5 について，$\ell = 2$ の場合は $\mathbb{F}_p$-有理な 2-等分点 $P \neq \mathcal{O}$ が存在すれば，$t \equiv 0 \bmod 2$ であることが分かる．さらに，奇素数 $\ell \geq 3$ に対する関係式 (3.11) の成立確認において，等分多項式を利用することで楕円曲線点の x-座標が一致するか効率的に確認することができる．

3.3.2 Schoof アルゴリズムの問題点と改良

Schoof アルゴリズムでは，Algorithm 1 の Step 5 の計算を (3.12) で定義される環 R_ℓ 上ですべて行う必要があるため，計算コストが非常に重いという問題点がある．その計算コストを下げるため，ℓ-等分多項式 $f_\ell(x)$ のある因子多項式 $F_\ell(x)$ を見つけ，R_ℓ より小さな環

$$\mathbb{F}_p[x,y]/(F_\ell(x), y^2 - f(x))$$

上で Step 5 の計算を行う改良方法が知られている．具体的には，奇素数 $\ell \neq p$ に対する等分多項式 $f_\ell(x)$ の次数が $\frac{\ell^2-1}{2}$ に対して，因子多項式 $F_\ell(x)$ の次数は $\frac{\ell-1}{2}$ であるため，大きな素数 ℓ で非常に計算効率が向上する．以下では，因子多項式 $F_\ell(x)$ の構成とその構成アルゴリズムを紹介する．

同種写像における Vélu の公式と因子多項式 $F_\ell(x)$ の定義

有限体 $\mathbb{F}_p$ 上の楕円曲線を E とする．奇素数 $\ell \neq p$ に対し，群 $E(\overline{\mathbb{F}}_p)$ の位数 ℓ の部分群を F とする．Vélu [Vél71] は，部分群 F から定まる同種写像 $\phi : E \longrightarrow \widetilde{E} = E/F$ を次のように明示的に表現した；まず，$F = \{\mathcal{O}\} \cup F^+ \cup F^-$ かつ $F^- = \{-P : P \in F^+\}$ を満たすように，集合 $F^* := F \setminus \{\mathcal{O}\}$ を 2 つの集合 F^+ と F^- に分割しておく．また，各 $P = (x_P, y_P) \in F^+$ に対して，

$$g_P^x = 3x_P^2 + a,\ g_P^y = -2y_P,\ v_p = 2g_P^x \text{ and } u_P = (g_P^y)^2 .$$

とおく．このとき，任意の点 $(x,y) \in E \setminus F$ に対する $\widetilde{E}$ の点 $\phi(x,y)$ は

$$\left(x + \sum_{P \in F^+} \frac{v_P}{x - x_P} - \frac{u_P}{(x-x_P)^2}, y - \sum_{P \in F^+} \frac{2u_P y}{(x-x_P)^3} + v_P \frac{y - y_P - g_P^x g_P^y}{(x-x_P)^2} \right)$$

となる．さらに，$v = \sum_{P \in F^+} v_P$, $w = \sum_{P \in F^+} u_P + x_P v_P$ とおくと，楕円曲線 $\widetilde{E}$ の Weierstrass 方程式は $y^2 = x^3 + (a - 5v)x + (b - 7w)$ である．不変微分に関する条件 $\phi^*(\omega_{\widetilde{E}}) = \omega_E$ を満たすとき，同種写像 $\phi : E \longrightarrow \widetilde{E}$ が正規化されているという．正規化されている同種写像 ϕ に関して，

$$\phi(x,y) = \left(\frac{N_\ell(x)}{D_\ell(x)}, y \left(\frac{N_\ell(x)}{D_\ell(x)} \right)' \right)$$

が成り立つ．ただし，多項式 $D_\ell(x)$ は

$$D_\ell(x) = \prod_{Q \in F^*} (x - x_Q) = x^{\ell-1} - s_1 x^{\ell-2} + s_2 x^{\ell-3} - \cdots + s_{\ell-1}$$

であり，$N_\ell(x)$ は

$$\frac{N_\ell(x)}{D_\ell(x)} = \ell x - s_1 - (3x^2 + a)\frac{D_\ell'(x)}{D_\ell(x)} - 2(x^3 + ax + b)\left(\frac{D_\ell'(x)}{D_\ell(x)} \right)'$$

を満たす多項式である．このとき，$d = \frac{\ell-1}{2}$ とおき，

$$F_\ell(x) = \prod_{Q \in F^+} (x - x_Q) = x^d + t_1 x^{d-1} + \cdots + t_d$$

と定める（明らかに，$D_\ell(x) = F_\ell(x)^2$ が成り立つ）．

因子多項式 $F_\ell(x)$ の構成アルゴリズム (Algorithm 2)

有限体 $\mathbb{F}_p$ 上の楕円曲線 E の j-不変量を j とする．奇素数 $\ell \neq p$ に関するモジュラー多項式 $\Phi_\ell(X, Y) \in \mathbb{Z}[X, Y]$ に対して [Sil94, Exercise 2.18 in Chapter II]，$\Phi_\ell(X, j) \in \mathbb{F}_p[X]$ が一次式に分解可能のとき，ℓ を Elkies 素数と呼ぶ（詳しくは，[BSS99, Chapter VII] や [Coh05, Chapter IV] を参照）．Algorithm 2 に，Elkies 素数 ℓ における ℓ-等分多項式 $f_\ell(x)$ の次数 $d = \frac{\ell-1}{2}$ の因子多項式 $F_\ell(x)$ の構成アルゴリズムを示す．また，以下でその構成アルゴリズムに関する具体的な数値例を示しておく．

■数値例 1　$p = 131$ に対して, 有限体 $\mathbb{F}_p$ 上の楕円曲線 $E : y^2 = x^3 + x + 23$ を考える．ここでは, $\ell = 5$ をとり $d = \frac{\ell-1}{2} = 2$ とする．このとき, Algorithm 2 の Steps 1, 2, 3 において, $j = j(E) = 78$, $\overline{E}_4(q) = 83$, $\overline{E}_6(q) = 91$, $j' = 66$ が計算できる (ここで示す計算結果はすべて mod$p$ における値)．また, Step 4 において,

$$\Phi_5(x, j) \equiv x^6 + x^5 + 67x^4 + 106x^3 + 16x^2 + 33x + 41 \bmod p$$

より,

$$\mathrm{GCD}(\Phi_5(x, j), x^p - x) = (x - 17)(x - 26) \in \mathbb{F}_p[x]$$

が得られる．これにより, 方程式 $\Phi_5(x, j) \equiv 0 \bmod p$ は $\mathbb{F}_p$ 上の根 $x = 17$ または 26 を持つため, $\ell = 5$ は $\mathbb{F}_p$ 上の楕円曲線 E に対する Elkies 素数であることが分かる．ここで $\tilde{j} = 17 \in \mathbb{F}_p$ と選択し, Step 5 から $\tilde{j}' = 48$ と計算できる．また, Steps 6 と 7 から $\tilde{a} = 62$, $\tilde{b} = 20$, $\overline{E}_4(q^\ell) = 37$, $\overline{E}_6(q^\ell) = 119$ が得られる．さらに, Step 8 で

$$\frac{j''}{j'} - \ell \frac{\tilde{j}''}{\tilde{j}'} = 2$$

と計算されるので, Step 9 で $p_1 = 42 \in \mathbb{F}_p$ と計算できる．Steps 14 と 15 で得られる $\mathbb{F}_p[[w]]$ の元は

$$\begin{cases} A(w) = 1 + 110w + 113w^2 + O(w^3) \\ C(w) = 26w + 109w^2 + O(w^3) \end{cases}$$

と計算され, Step 18 で $F_{5,2} = 1$, $F_{5,1} = 110$, $F_{5,0} = 61 \in \mathbb{F}_p$ が得られる．よって, $f_5(x)$ の次数 $d = 2$ の因子 $F_5(x) = x^2 + 110x + 61 \in \mathbb{F}_p[x]$ が出力さ

れる. 実際, $\ell = 5$ に対する等分多項式は

$$f_5(x) = 5x^{12} + 62x^{10} + 94x^9 + 26x^8 + 18x^7 + 72x^6 + 105x^5 + 100x^4 + 49x^3 + 60x^2 + 80x + 122 \in \mathbb{F}_p[x]$$

であり, 上記で求めた多項式 $F_5(x)$ で割り切れることが分かる.

■数値例 2　$p = 1009$ に対して, 有限体 $\mathbb{F}_p$ 上の楕円曲線 $E : y^2 = x^3 + 320x + 197$ を考える. ここで $\ell = 13$ をとり $d = \frac{\ell-1}{2} = 6$ とする. このとき, Algorithm 2 の Steps 1, 2, 3 において, $j = j(E) = 951$, $\overline{E}_4(q) = 784$, $\overline{E}_6(q) = 696$, $j' = 278$ が計算できる (ここで示す計算結果はすべて mod$p$ における値). また, Step 4 において,

$$\Phi_{13}(x, j) \equiv x^{14} + 497x^{13} + 173x^{12} + 922x^{11} + 892x^{10} + 308x^9 + 469x^8 + 424x^7 + 350x^6 + 740x^5 + 455x^4 + 974x^3 + 846x^2 + 47x + 603 \bmod p$$

より,

$$\mathrm{GCD}(\Phi_{13}(x, j), x^p - x) = (x - 225)(x - 518) \in \mathbb{F}_p[x]$$

が得られる. これにより, 方程式 $\Phi_{13}(x, j) \equiv 0 \bmod p$ は $\mathbb{F}_p$ 上の根 $x = 225$ または 518 を持ち, $\ell = 13$ は $\mathbb{F}_p$ 上の楕円曲線 E に対する Elkies 素数であることが分かる. ここで $\tilde{j} = 225 \in \mathbb{F}_p$ と選択し, Step 5 から $\tilde{j}' = 216$ と計算できる. また, Steps 6 と 7 から $\tilde{a} = 334$, $\tilde{b} = 973$, $\overline{E}_4(q^\ell) = 112$, $\overline{E}_6(q^\ell) = 175$ が得られる. さらに, Step 8 で

$$\frac{j''}{j'} - \ell\frac{\tilde{j}''}{\tilde{j}'} = 958$$

と計算されるので, Step 9 で $p_1 = 347 \in \mathbb{F}_p$ と計算できる. Steps 14 と 15 で得られる $\mathbb{F}_p[[w]]$ の元は

$$\begin{cases} A(w) = 1 + 331w + 869w^2 + 83w^3 + 628w^4 + 669w^5 + 225w^6 + O(w^7) \\ C(w) = 945w + 116w^2 + 20w^3 + 85w^4 + 62w^5 + 697w^6 + O(w^7) \end{cases}$$

と計算され, Step 19 で下記の $F_{13}(x) \in \mathbb{F}_p[x]$ が出力される:

$$F_{13}(x) = x^6 + 331x^5 + 244x^4 + 371x^3 + 253x^2 + 654x + 814.$$

Trace t に対する $t \bmod \ell$ の計算（Algorithm 3）

有限体 $\mathbb{F}_p$ 上の楕円曲線 $E : y^2 = x^3 + ax + b$ と Elkies 素数 $\ell \geq 3$ に対して, ℓ-等分多項式 $f_\ell(x)$ の次数 $d = \frac{\ell-1}{2}$ の因子 $F_\ell(x) \in \mathbb{F}_p[x]$ が与えられたとする (因子 $F_\ell(x)$ は Algorithm 2 で見つける). ここでは, 楕円曲線 E が持つ

trace t に対して, $t \bmod \ell$ を求める方法を考える. $C \subset E[\ell]$ を因子 $F_\ell(x)$ に対応する位数 ℓ の巡回群とする. 任意の $\mathcal{O} \neq P = (x, y) \in C$ に対して,

$$(x^p, y^p) = \varphi(P) = [k]P \tag{3.13}$$

を満たす $1 \leq k \leq \ell - 1$ が存在する (k は C のみに依存). Frobenius 写像 $\varphi \in \mathrm{End}(E)$ は $\varphi^2 - [t]\varphi + [p] = [0]$ を満たし, 点 $P \in E$ は位数 ℓ の元なので,

$$t \equiv k + \frac{p}{k} \bmod \ell$$

が成り立つ. 上記の議論から, (3.13) を満たす k を求めることで $t \bmod \ell$ が計算できる.

Algorithm 3 に, Elkies 素数 $\ell \geq 3$ に対して $t \bmod \ell$ を求めるアルゴリズムを示す. Steps 1 と 2 で, (3.13) を満たす $1 \leq k \leq \ell - 1$ を決定する. 具体的には, 関係式 (3.8) と (3.13) より, k は

$$\begin{aligned} & x^p = x - \frac{\psi_{k-1}\psi_{k+1}}{\psi_k^2} \\ \Longleftrightarrow\ & (x^p - x)\psi_k^2 + \psi_{k-1}\psi_{k+1} = 0 \ (\text{in } \mathbb{F}_p[x]/(F_\ell(x), y^2 - x^3 - ax - b)) \end{aligned}$$

を満たすので, 環 $\mathbb{F}_p[x]/(F_\ell(x))$ 上で

$$\begin{cases} 4(x^p - x)(x^3 + ax + b)f_k^2 + f_{k-1}f_{k+1} = 0 & (k\text{:奇数}) \\ (x^p - x)f_k^2 + 4(x^3 + ax + b)f_{k-1}f_{k+1} = 0 & (k\text{:偶数}) \end{cases}$$

を満たす $1 \leq k \leq \ell - 1$ を Step 1 で求める (下記で y-座標の比較を行うので, $1 \leq k \leq d$ で探索すれば十分). Step 1 では点 $P = (x, y)$ の x-座標比較のみで, k または $\ell - k$ の 2 つが候補として残る. Step 2 では, y-座標で (3.13) を満たす k を決定する. 具体的には, Step 1 で求めた k が 1 の場合,

$$y^p - y = 0 \Longleftrightarrow y\left(\left(x^3 + ax + b\right)^{(p-1)/2} - 1\right) = 0$$

から, 環 $\mathbb{F}_p[x]/(F_\ell(x))$ 上で $\left(x^3 + ax + b\right)^{(p-1)/2} = 1$ が成立するか確認する ($P = (x, y) \in C$ は $[2]P \neq \mathcal{O}$ より, $y \neq 0$ であることに注意する). もし成立しない場合は, $k \leftarrow \ell - k$ とすれば良い. Step 1 で求めた k が 2 以上の場合 関係式 (3.8) と (3.13) より,

$$4y^{(p+1)}\psi_k^3 = \psi_{k+2}\psi_{k-1}^2 - \psi_{k-2}\psi_{k+1}^2$$

が成立するか確認すれば良い. より具体的には, 関係式 $y^2 = x^3 + ax + b$ と $\psi_m(x, y)$ ではなく $f_m(x)$ を利用して, 環 $\mathbb{F}_p[x]/(F_\ell(x))$ 上で

$$\begin{cases} 16(x^3 + ax + b)^{(p+3)/2} f_k^3 = f_{k+2}f_{k-1}^2 - f_{k-2}f_{k+1}^2 & (k\text{:奇数}) \\ (x^3 + ax + b)^{(p-1)/2} f_k^3 = f_{k+2}f_{k-1}^2 - f_{k-2}f_{k+1}^2 & (k\text{:偶数}) \end{cases}$$

が成り立つか確認する. 成立しない場合は, $k \leftarrow \ell - k$ とすれば良い.

Algorithm 2 ℓ-等分多項式 $f_\ell(x)$ の因子 $F_\ell(x)$ の構成 (ℓ は Elkies 素数)

Input: 有限体 $\mathbb{F}_p$ 上の楕円曲線 $E : y^2 = x^3 + ax + b$ と Elkies 素数 $\ell \geq 3$
(ただし, p は ℓ と異なる (十分大きな) 素数で, $j(E) \neq 0, 1728$ と仮定)

Output: ℓ-等分多項式 $f_\ell(x)$ の次数 $d = \frac{\ell-1}{2}$ の因子 $F_\ell(x) \in \mathbb{F}_p[x]$

1: $j = j(E) = 1728\dfrac{4a^3}{4a^3 + 27b^2}$ を計算
2: $\overline{E}_4(q) = -48a$ と $\overline{E}_6(q) = 864b$ を計算
3: $j' = -j\dfrac{\overline{E}_6(q)}{\overline{E}_4(q)}$ を計算
4: $\Phi_\ell(x, j) \in \mathbb{F}_p[x]$ の $\mathbb{F}_p$ 上の根 $\tilde{\jmath}$ を 1 つ選ぶ /* $\Phi_\ell(x, y) \in \mathbb{Z}[x, y]$ は ℓ に対するモジュラー多項式, $x^p - x \in \mathbb{F}_p[x]$ との GCD 計算により $\tilde{\jmath}$ を見つけることが可能 */
5: $\tilde{\jmath}' = -\dfrac{j'\Phi_x(j, \tilde{\jmath})}{\ell\Phi_y(j, \tilde{\jmath})}$ を計算
6: $\tilde{a} = -\dfrac{(\tilde{\jmath}')^2}{48\tilde{\jmath}(\tilde{\jmath} - 1728)}$ と $\tilde{b} = -\dfrac{(\tilde{\jmath}')^3}{864\tilde{\jmath}^2(\tilde{\jmath} - 1728)}$ を計算
7: $\overline{E}_4(q^\ell) = -48\tilde{a}$ と $\overline{E}_6(q^\ell) = 864\tilde{b}$ を計算
8: $\dfrac{j''}{j'} - \ell\dfrac{\tilde{\jmath}''}{\tilde{\jmath}'} = -\dfrac{j'^2\Phi_{xx}(j, \tilde{\jmath}) + 2\ell j'\tilde{\jmath}'\Phi_{xy}(j, \tilde{\jmath}) + \ell^2\tilde{\jmath}'^2\Phi_{yy}(j, \tilde{\jmath})}{j'\Phi_x(j, \tilde{\jmath})}$ を計算
9: $p_1 = \dfrac{\ell}{2}\left(\dfrac{j''}{j'} - \ell\dfrac{\tilde{\jmath}''}{\tilde{\jmath}'}\right) + \dfrac{\ell}{4}\left(\dfrac{\overline{E}_4^2(q)}{\overline{E}_6(q)} - \ell\dfrac{\overline{E}_4^2(q^\ell)}{\overline{E}_6(q^\ell)}\right) + \dfrac{\ell}{3}\left(\dfrac{\overline{E}_6(q)}{\overline{E}_4(q)} - \ell\dfrac{\overline{E}_6(q^\ell)}{\overline{E}_4(q^\ell)}\right)$ を計算
10: $c_1 = -\dfrac{a}{5}, c_2 = -\dfrac{b}{7}$ と $\tilde{c}_1 = -\dfrac{\ell^4\tilde{a}}{5}, \tilde{c}_2 = -\dfrac{\ell^6\tilde{b}}{7}$ とおく
11: **for** $k = 3, 4, \ldots, d$ **do**
12: $\quad c_k = \dfrac{3}{(k-2)(2k+3)}\displaystyle\sum_{h=1}^{k-2} c_h c_{k-1-h}$ を計算
13: $\quad \tilde{c}_k = \dfrac{3}{(k-2)(2k+3)}\displaystyle\sum_{h=1}^{k-2} \tilde{c}_h \tilde{c}_{k-1-h}$ を計算
14: **end for**
15: $A(w) = \exp\left(-\dfrac{1}{2}p_1 w - \displaystyle\sum_{k=1}^{d}\frac{\tilde{c}_k - \ell c_k}{(2k+1)(2k+2)}w^{k+1}\right) + O(w^{d+1}) \in \mathbb{F}_p[[w]]$ /* $O(w^{d+1})$-精度の w-進展開 */
16: $C(w) = \displaystyle\sum_{k=1}^{d} c_k w^k + O(w^{d+1}) \in \mathbb{F}_p[[w]]$ /* $O(w^{d+1})$-精度の w-進展開 */
17: $F_{\ell,d} = 1$ とおく
18: **for** $i = 1, 2, \ldots, d$ **do**
19: $\quad F_{\ell,d-i} = [A(w)]_i - \displaystyle\sum_{k=1}^{i}\left(\sum_{h=0}^{k}\binom{d-i+k}{k-h}[C(w)^{k-h}]_h\right)F_{\ell,d-i+k}$ を計算 /* $[B(w)]_i$ は $B(w) \in \mathbb{F}_p[[w]]$ の w^i-係数とする */
20: **end for**
21: $F_\ell(x) = x^d + \displaystyle\sum_{i=0}^{d-1} F_{\ell,i}x^i \in \mathbb{F}_p[x]$ を出力

Algorithm 3 Trace t に対する $t \bmod \ell$ の計算 (ℓ は Elkies 素数)

Input: 有限体 $\mathbb{F}_p$ 上の楕円曲線 $E : y^2 = x^3 + ax + b$, Elkies 素数 $\ell \geq 3$ と ℓ-等分多項式 $f_\ell(x)$ の次数 $d = \frac{\ell-1}{2}$ の因子 $F_\ell(x) \in \mathbb{F}_p[x]$

Output: $t \bmod \ell$

1: 環 $\mathbb{F}_p[x]/(F_\ell(x))$ 上で

$$\begin{cases} 4(x^p - x)(x^3 + ax + b)f_k^2 + f_{k-1}f_{k+1} = 0 & (k\text{:奇数}) \\ (x^p - x)f_k^2 + 4(x^3 + ax + b)f_{k-1}f_{k+1} = 0 & (k\text{:偶数}) \end{cases}$$

が成り立つ $1 \leq k \leq d$ を求める

2: **if** $k = 1$ **then**

3: 環 $\mathbb{F}_p[x]/(F_\ell(x))$ 上で $\left(x^3 + ax + b\right)^{(p-1)/2} = 1$ が成立するか確認. 成立しない場合, $k \leftarrow \ell - k$ とする

4: **else**

5: 環 $\mathbb{F}_p[x]/(F_\ell(x))$ 上で

$$\begin{cases} 16(x^3 + ax + b)^{(p+3)/2} f_k^3 = f_{k+2}f_{k-1}^2 - f_{k-2}f_{k+1}^2 & (k\text{:奇数}) \\ (x^3 + ax + b)^{(p-1)/2} f_k^3 = f_{k+2}f_{k-1}^2 - f_{k-2}f_{k+1}^2 & (k\text{:偶数}) \end{cases}$$

が成り立つか確認. 成立しない場合, $k \leftarrow \ell - k$ とする

6: **end if**

7: $k + \dfrac{p}{k} \bmod \ell$ を出力する

参考文献

[BSS99] Ian Blake, Gadiel Seroussi, and Nigel Smart, *Elliptic curves in cryptography*, Cambridge university press **265**, 1999.

[Coh05] Henri Cohen et al., *Handbook of elliptic and hyperelliptic curve cryptography* CRC press, 2005.

[PARI] The PARI Group, Bordeaux, *PARI/GP*, available at `https://pari.math.u-bordeaux.fr/`.

[Sch85] René Schoof, *Elliptic curves over finite fields and the computation of square root mod p*, Math. Comp. **44**, 483–494, 1985.

[Sch95] René Schoof, *Counting points on elliptic curves over finite fields*, J. Théor. Nombres Bordeaux **7**(1), 219–254, 1995.

[Sil86] Joseph H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics **106**, Springer-Verlag New York. 1986.

[Sil94] Joseph H. Silverman, *Advanced topics in the arithmetic of elliptic curves*, Graduate Texts in Mathematics **151**, Springer-Verlag New York, 1994.

[Vél71] Jacques Vélu, *Isogénies entre courbes elliptiques* Comptes-Rendus de l'Académie des Sciences **273**, 238–241, 1971.

3.4 付録：PARI ライブラリ [PARI] による実装コード

今回，C 言語のライブラリとして利用可能な数式処理 PARI ライブラリ [PARI]（version 2.9.2）を利用して，Algorithm 2 と 3 を利用した Schoof アルゴリズム（Algorithm 1）を実装した．本節では，その実装ソースコードを示しておく．

```
#include <stdio.h>
#include <stdlib.h>
#include <pari/pari.h>
```

```
/*************************************************************
Construction of a factor of a 等分 polynomial
Input:
GEN p : characteristic of base finite field (prime number)
GEN E : elliptic curve E: y^2 = x^3 + ax + b over GF(p)
GEN ell : prime number (different from p)
Output:
GEN F : a factor of the ell-th 等分 polynomial
(if ell is an Elkies prime)
*************************************************************/
GEN Factor_of_等分Polynomial(GEN p, GEN E, GEN ell) {
int i, h, k, d, pp = itos(p);
long x=fetch_user_var("x"), y=fetch_user_var("y");
long w=fetch_user_var("w"); /* x. y. w: variables */
long ltop = avma, lbot; /* for gabage collection */
GEN a, b, j, jj, E4, E6;
GEN tilde_j, tilde_jj, tilde_a, tilde_b, tilde_E4, tilde_E6;
GEN q, p1, c, tilde_c;
GEN A, C, pol_F, F;
GEN tmp, tmp1, tmp2, tmp3, v, pol;
GEN Phi, Phi_x, Phi_y, Phi_xx, Phi_xy, Phi_yy;

/* Initialization */
a = compo(E, 4);
b = compo(E, 5);
v = cgetg(pp+2, t_VEC);
for (i=1; i<=pp+1; i++) {v[i] = (long) gen_0;}
v[2] = (long) gneg(gen_1); v[pp+1] = (long) gen_1;
pol = gtopolyrev(v, x); /* pol = x^p - x */

printf("\nStart: Factor_of_等分Polynomial\n");

/* Step 1 */
tmp = gmul(stoi(4), gpow(a, stoi(3), DEFAULTPREC)); /* 4a^3 */
tmp1 = gadd(tmp, gmul(stoi(27), gsqr(b))); /* 4a^3 + 27b^2 */
j = gmul(stoi(1728), gmul(tmp, ginv(tmp1)));  /* j = j(E) */
if (gcmp(lift(j), stoi(1728))==0 || gcmp(lift(j), stoi(0))==0)
```

```
{
printf("j(E) = 1728 or 0 in Factor_of_等分 Polynomial\n");
return gneg(gen_1); /* End of program for error */
}
pari_printf("j = %Ps\n", j);

/* Step 2 */
E4 = gmul(gneg(stoi(48)), a);  /* E4 = -48*a */
E6 = gmul(stoi(864), b);  /* E6 = 864*b */
pari_printf("E4 = %Ps, E6 = %Ps\n", E4, E6);

/* Step 3 */
jj = gneg(gmul(gmul(j, E6), ginv(E4))); /* jj = -(j*E6)/E4 */
pari_printf("jj = %Ps\n", jj);

/* Step 4 */
Phi = polmodular_ZXX(itos(ell), 0, x, y);/* modular poly. */
tmp = gsubst(Phi, y, j);   /* Phi(x, j) */
pari_printf("Phi(x, j) = %Ps\n", tmp);
tmp1 = ggcd(tmp, pol);
v = rootmod0(lift(tmp1), p, 0);
if (glength(v) == 0) {
printf("ell is an Atkin prime\n");
lbot = avma;
return gerepile(ltop, lbot, gen_0); /* End of program */
} else {
tilde_j = gcopy(compo(v, 1));
pari_printf("tilde_j = %Ps, ", tilde_j);
pari_printf("GCD = %Ps\n", v);
}

/* Step 5 */
Phi_x = deriv(Phi, x);
Phi_y = deriv(Phi, y);
tmp = gsubst(Phi_x, x, j); tmp = gsubst(tmp, y, tilde_j);
tmp1 = gsubst(Phi_y, x, j); tmp1 = gsubst(tmp1, y, tilde_j);
tilde_jj = gmul(gmul(jj, tmp), ginv(gmul(ell, tmp1)));
```

```
tilde_jj = gneg(tilde_jj);
pari_printf("tilde_jj = %Ps\n", tilde_jj);

/* Step 6 */
tmp = gmul(tilde_j, gsub(tilde_j, stoi(1728)));
tmp1 = gsqr(tilde_jj); /* tilde_jj^2 */
tilde_a = gneg(gmul(tmp1, ginv(gmul(stoi(48), tmp))));
tmp = gmul(gmul(stoi(864), tilde_j), tmp);
tmp1 = gmul(tmp1, tilde_jj);
tilde_b = gneg(gmul(tmp1, ginv(tmp)));
pari_printf("tilde_a = %Ps, tilde_b = %Ps\n", tilde_a, tilde_b);

/* Step 7 */
tilde_E4 = gneg(gmul(stoi(48), tilde_a));
tilde_E6 = gmul(stoi(864), tilde_b);
pari_printf("tilde_E4 = %Ps, tilde_E6 = %Ps\n", tilde_E4, tilde_E6);

/* Step 8 */
Phi_xx = deriv(Phi_x, x);
Phi_xy = deriv(Phi_x, y);
Phi_yy = deriv(Phi_y, y);
tmp = gsubst(Phi_xx, x, j); tmp = gsubst(tmp, y, tilde_j);
tmp = gmul(gsqr(jj), tmp); /* tmp = j^2*Phi_xx */
tmp1 = gsubst(Phi_xy, x, j); tmp1 = gsubst(tmp1, y, tilde_j);
tmp1 = gmul(gmul(jj, tilde_jj), tmp1);
tmp1 = gmul(gmul(stoi(2), ell), tmp1);
tmp2 = gsubst(Phi_yy, x, j); tmp2 = gsubst(tmp2, y, tilde_j);
tmp2 = gmul(gsqr(tilde_jj), tmp2);
tmp2 = gmul(gsqr(ell), tmp2); /* ell^2*tilde_jj^2*Phi_yy */
tmp3 = gsubst(Phi_x, x, j); tmp3 = gsubst(tmp3, y, tilde_j);
tmp3 = gmul(jj, tmp3); /* jj*Phi_x */
q = gadd(gadd(tmp, tmp1), tmp2);
q = gneg(gmul(q, ginv(tmp3)));
pari_printf("q = %Ps\n", q);

/* Step 9 */
tmp = gmul(gsqr(E4), ginv(E6));
```

```
tmp1 = gmul(gsqr(tilde_E4), ginv(tilde_E6));
tmp = gsub(tmp, gmul(ell, tmp1));
tmp = gmul(gmul(ell, ginv(stoi(4))), tmp);/*ell/4*(....)*/
tmp1 = gmul(E6, ginv(E4));
tmp2 = gmul(tilde_E6, ginv(tilde_E4));
tmp1 = gsub(tmp1, gmul(ell, tmp2));
tmp1 = gmul(gmul(ell, ginv(stoi(3))), tmp1);/*ell/3*(...)*/
p1=gadd(gadd(gmul(gmul(ell, ginv(stoi(2))),q),tmp),tmp1);
pari_printf("p1 = %Ps\n", p1);

/* Step 10 */
d = itos(ell) - 1; d = d/2;
c = cgetg(d+1, t_VEC); tilde_c = cgetg(d+1, t_VEC);
for (i=1; i<=d; i++) {
c[i] = (long) gen_0; tilde_c[i] = (long) gen_0;
}
c[1] = (long) gneg(gmul(a, ginv(stoi(5))));
tmp = gsqr(ell); tmp1 = gsqr(tmp); tmp2 = gmul(tmp, tmp1);
tilde_c[1]=(long)gmul(tmp1,gneg(gmul(tilde_a,ginv(stoi(5)))));
if (d >= 2) {
c[2] = (long) gneg(gmul(b, ginv(stoi(7))));
tilde_c[2]=(long)gmul(tmp2,gneg(gmul(tilde_b,ginv(stoi(7)))));
}

/* Steps 11, 12, 13 */
for (k=3; k<=d; k++) {
tmp = gen_0; tmp1 = gen_0;
for (h=1; h<=k-2; h++) {
tmp=gadd(tmp, gmul(compo(c, h), compo(c, k-1-h)));
tmp1=gadd(tmp1, gmul(compo(tilde_c, h), compo(tilde_c, k-1-h)));
}
tmp2 = gmodulo(stoi((k-2)*(2*k+3)), p);
tmp2 = gmul(stoi(3), ginv(tmp2));
c[k] = (long) gmul(tmp2, tmp);
tilde_c[k] = (long) gmul(tmp2, tmp1);
}
```

```
/* Step 14 */
v = cgetg(d+3, t_VEC);
for (k=1; k<=d+2; k++) {v[k] = (long) gen_0;}
v[2] = (long) gneg(gmul(p1, ginv(gmodulo(stoi(2), p))));
for (k=1; k<=d; k++) {
tmp = gsub(compo(tilde_c, k), gmul(ell, compo(c, k)));
tmp = gmul(tmp, ginv(gmodulo(stoi((2*k+1)*(2*k+2)), p)));
v[k+2] = (long) gneg(tmp);
}
pol = gtopolyrev(v, w);
pol = gadd(pol, zeroser(w, d+1));

A = gen_1; tmp = gen_1;
for (k=1; k<=d; k++) {
tmp = gmul(tmp, pol);
A = gadd(A, gmul(tmp, ginv(mpfact(k))));
}
pari_printf("A(w) = %Ps\n", A);

/* Step 15 */
v = cgetg(d+2, t_VEC);
v[1] = (long) gen_0;
for (i=2; i<=d+1; i++) {v[i] = (long) compo(c, i-1);}
C = gtopolyrev(v, w);
C = gadd(C, zeroser(w, d+1));
pari_printf("C(w) = %Ps\n", C);

/* Step 16 */
F = cgetg(d+2, t_VEC);
for (i=1; i<=d; i++) {F[i] = (long) gen_0;}
F[d+1] = (long) gmodulo(gen_1, p);

/* Steps 17, 18, 19 */
for (i=1; i<=d; i++) {
tmp2 = gen_0;
for (k=1; k<=i; k++) {
tmp1 = gen_0;
```

```
for (h=0; h<=k; h++) {
tmp = gpow(C, stoi(k-h), DEFAULTPREC);
tmp = polcoeff0(tmp, h, w);
tmp = gmul(binomial(stoi(d-i+k), k-h), tmp);
tmp1 = gadd(tmp1, tmp);
}
tmp2 = gadd(tmp2, gmul(tmp1, compo(F, d-i+k+1)));
}
F[d-i+1] = (long) gsub(polcoeff0(A, i, w), tmp2);
}

/* Step 19 */
pol_F = gtopolyrev(F, x);
pari_printf("pol_F = %Ps\n", lift(pol_F));
lbot = avma;

return gerepile(ltop, lbot, gcopy(pol_F));
}

/*******************************************************
Computation of an eigenvalue for an Elkies prime
Input:
GEN p : characteristic of base finite field (prime)
GEN E : elliptic curve E: y^2 = x^3 + ax + b over GF(p)
GEN ell : Elkies prime (different from p)
GEN F : a factor of the ell-th 等分 polynomial
Output:
GEN t : t modulo ell (t = trace of E mod p)
*******************************************************/
GEN Eigenvalue_Elkies(GEN p, GEN E, GEN ell, GEN F)
{
int d, k, s, r;
long x=fetch_user_var("x"), y=fetch_user_var("y");
long ltop = avma, lbot;
GEN pol_x, z, w;
GEN a=compo(E, 4), b=compo(E, 5); /* Elements of GF(p) */
GEN f, g, h, f1, f2, lambda;
```

```
GEN tmp, tmp1, tmp2, tmp3;

printf("\nStart: Eigenvalue_Elkies\n");
/* pol_x = x (mod F) */
pol_x = cgetg(4, t_POL);
setvarn(pol_x, x);
pol_x[2] = (long) gen_0;
pol_x[3] = (long) gen_1;
pol_x = gmodulo(pol_x, F);

/* z = x^p - x (mod F) */
z = gsub(gpow(pol_x, p, DEFAULTPREC), pol_x);

/* w = x^3+ax+b (mod F) */
tmp = gpow(pol_x, stoi(3), DEFAULTPREC);
w = gadd(tmp, gadd(gmul(lift(a), pol_x), lift(b)));

/* Step 1 */
d = itos(ell)-1; d = d/2;
f = elldivpol(E, 0, y);
g = elldivpol(E, 1, y);
h = elldivpol(E, 2, y);
for (k=1; k<=d; k++) {
if (k%2 == 1) {
tmp = gsubst(g, y, pol_x); tmp = gsqr(tmp);
tmp = gmul(z, tmp);
tmp = gmul(gmul(stoi(4), w), tmp);
tmp1 = gsubst(f, y, pol_x);
tmp2 = gsubst(h, y, pol_x);
tmp1 = gmul(tmp1, tmp2); /* f_k-1*f_k+1 */
tmp2 = gadd(tmp, tmp1);
tmp2 = lift(lift(tmp2));
tmp2 = lift(ggcd(tmp2, F));
if (poldegree(tmp2, x) >= 1) {
s = k; break;
}
} else {
```

```
tmp = gsubst(g, y, pol_x); tmp = gsqr(tmp);
tmp = gmul(tmp, z); /* (x^p-x)*f_k^2 */
tmp1 = gsubst(f, y, pol_x);
tmp2 = gsubst(h, y, pol_x);
tmp1 = gmul(tmp1, tmp2);
tmp1 = gmul(tmp1, gmul(stoi(4), w));
tmp2 = lift(lift(gadd(tmp, tmp1)));
tmp2 = lift(ggcd(tmp2, F));
if (poldegree(tmp2, x) >= 1) {
s = k; break;
}
}
/* Update of 等分 polynomials */
f = gcopy(g);
g = gcopy(h);
h = elldivpol(E, k+2, y);
}

/* Step 2 */
r = itos(p)-1; r = r/2;
tmp = gpow(w, stoi(r), DEFAULTPREC); /*(x^3+ax+b)^{(p-1)/2}*/
if (s == 1) {
tmp1 = ggcd(gsub(tmp, gen_1), F);
tmp1 = lift(tmp1);
if (poldegree(tmp1, x) >= 1) {k = 1;}
else {k = itos(ell)-1;}
} else {
f = elldivpol(E, s-2, y); f = gsubst(f, y, pol_x);
g = elldivpol(E, s-1, y); g = gsubst(g, y, pol_x);
h = elldivpol(E, s, y); h = gsubst(h, y, pol_x);
f1 = elldivpol(E, s+1, y); f1 = gsubst(f1, y, pol_x);
f2 = elldivpol(E, s+2, y); f2 = gsubst(f2, y, pol_x);

tmp1 = gsub(gmul(f2, gsqr(g)), gmul(f, gsqr(f1)));
tmp2 = gpow(h, stoi(3), DEFAULTPREC);
if (s%2==1) {
tmp2 = gmul(gmul(gmul(stoi(16), gsqr(w)), tmp), tmp2);
```

```
tmp3 = lift(ggcd(gsub(tmp2, tmp1), F));
} else {
tmp2 = gmul(tmp, tmp2);
tmp3 = lift(ggcd(gsub(tmp2, tmp1), F));
}
if (poldegree(tmp3, x) >= 1) {k = s;}
else {k = itos(ell) - s;}
}

/* Step 3 */
tmp = gmodulo(stoi(k), ell);
tmp1 = gmul(gadd(gsqr(tmp), p), ginv(tmp));
pari_printf("t (mod ell) = %Ps\n", tmp1);
lbot = avma;

return gerepile(ltop, lbot, gcopy(tmp1));
}

/*********************************************************
Computation of trace of an elliptic curve (Elkies primes)
Input:
GEN p : characteristic of base finite field (prime number)
GEN E : elliptic curve E: y^2 = x^3 + ax + b over GF(p)
Output:
GEN t : trace of E mod p
*********************************************************/
GEN Computation_Trace(GEN p, GEN E)
{
long x = fetch_user_var("x");
long ltop = avma, lbot;
GEN F, ell, M = gen_1, L, MAX;
GEN tmp, tmp1;

MAX = gmul(stoi(4), gsqrt(p, DEFAULTPREC)); /* 4p^{1/2} */
ell = gnextprime(stoi(3));
tmp = gmodulo(gen_0, gen_1);
while (gcmp(M, MAX) <= 0) {
```

```
  F = Factor_of_等分 Polynomial(p, E, ell);

if (poldegree(F, x) >= 1) {
/* Case where ell is an Elkies prime */
L = Eigenvalue_Elkies(p, E, ell, F);
tmp = chinese(tmp, L); /* CRT computation */
M = gmul(M, ell);
}
ell = gnextprime(gadd(ell, stoi(1)));
}

tmp = lift(tmp);
tmp1 = gmul(stoi(2), gsqrt(p, DEFAULTPREC));
if (gcmp(tmp, tmp1) > 0) {
/* Case where tmp > 2p^{1/2} */
tmp = gsub(tmp, M);
}
lbot = avma;

return gerepile(ltop, lbot, gcopy(tmp)); /* trace of E mod p */
}

/****************
Main for Tests
*****************/
int main(void)
{
pari_init(20000000000,2);

int i;
GEN p, a, b, ell;
GEN E, F;

/* Test for Schoof Algorithm (10 times) */
GEN trace, t;
p = gnextprime(stoi(1000000));
```

```
for (i=1; i<=10; i++) {
p = gnextprime(gadd(p, gen_1));
GEN a = genrand(p), b = genrand(p);
GEN E = cgetg(3, t_VEC);
E[1] = (long) gmodulo(a, p);
E[2] = (long) gmodulo(b, p);
E = ellinit(E, NULL, DEFAULTPREC);

trace = Computation_Trace(p, E);
pari_printf("trace = %Ps\n", trace);
t = ellap(E, NULL);
pari_printf("t = %Ps\n", t);

if (gcmp(trace, t) != 0) {
printf("Error of Computation_Trace\n");
break;
}
}

return 0;
}
```