ペアリング暗号に適した楕円曲線の探索

照屋唯紀
産業技術総合研究所

1. はじめに

楕円曲線上に構成できるペアリングと呼ばれる演算を用いることで、有用な機能と高い性能を合わせ持つ暗号方式が構成できることが示された [14, 12]。その後、ペアリングを用いた ID ベース暗号や短かい電子署名、集約署名、効率的な放送型暗号、属性ベース暗号など、数多くの暗号方式が提案されている [6]。これらペアリングを用いた暗号技術を総称して**ペアリング暗号**と言う。

安全なペアリング暗号を実現するためには、安全性の根拠となる有限体離散対数問題および楕円曲線離散対数問題の困難性を十分に持つ楕円曲線を使用する必要がある。つまり、現時点で判明している解読法である数体篩法や関数体篩法、Pollard の ρ 法などに対し、十分な耐性を持つ楕円曲線を使用しなければならない。そして実用的なペアリング暗号を実現するためには、安全な楕円曲線の中から高速な暗号方式の実装を可能とする楕円曲線を使用することが望ましい。

本課題では、安全性や効率性の要件を考慮したペアリング暗号に適した楕円曲線を大規模に探索するプログラムを作成する。そして、実際に探索実験を行うことで、最良のペアリング暗号に適した楕円曲線の選定に貢献することを目標とする。

本稿の構成： 第 2 節ではペアリング暗号に関する数学的基礎について解説する。第 3 節では、ペアリング暗号の効率的な計算に適した楕円曲線の探索方法や要件について解説し、これを踏まえて、第 4 節で本課題で開発した探索プログラムについて述べ、第 5 節で開発したプログラムを用いて実施した実験とその結果について解説する。最後に第 6 節で本課題をまとめる。

2. 数学的準備

ここでは、探索の対象となるペアリング暗号に適した楕円曲線とその周辺の数学的背景について解説する。ここでは、ペアリング演算については紹介しない。詳細は [6] を参照願いたい。

諸定義： 正整数 n に対して、0 から $n-1$ までの整数の集合を $\langle n\rangle := \{0,\dots,n-1\}$ と書く。非ゼロ整数 z の符号付き 2 進数展開を $z = \sum_{i=0}^{m} s_i 2^i$ （$i = \langle m+1\rangle$ について $s_i \in \{-1,0,1\}$）となるような有限個の数値列 $s_0, s_1, \dots, s_m$ とする。ここで、非ゼロで m が最大の s_m を z の最上位（非ゼロ）ビット（most siginificant bit）、m を最上位（非ゼロ）ビット位置（most siginificant bit position）と呼び、$\mathrm{hw}(z) := \#\{s_i \mid s_i \neq 0\}$ をハミング重み（Hamming weight）と呼ぶ。また、符号付き 2 進数展開 $s_0,\dots,s_m$ を、対応する整数 $z = \sum_{i=0}^{m} s_i 2^i$ へと変換する写像を $\mathsf{SB2Int}$ と書く。

楕円曲線： p を 3 より大きい素数とし、$\mathbb{F}_p$ を体位数が p の有限体とする。$\mathbb{F}_p$ 上定義される楕円曲線 $E/\mathbb{F}_p$（または単に E）は、X と Y を変数とする方程式 $Y^2 = X^3 + aX + b$ で定義される。ここで、$a, b \in \mathbb{F}_p$ かつ $4a^3 + 27b^2 \neq 0$ とする。ℓ を正の整数とし、$\mathbb{F}_{p^\ell}$ を $\mathbb{F}_p$ の ℓ 次拡大体とする。E の $\mathbb{F}_{p^\ell}$-有理点群 $E(\mathbb{F}_{p^\ell})$ とは、$E(\mathbb{F}_{p^\ell}) := \{(x,y) \in \mathbb{F}_{p^\ell} \times \mathbb{F}_{p^\ell} \mid y^2 = x^3 + ax + b\} \cup \{\infty\}$ である。ここで、∞ は無限遠点である。

$E(\mathbb{F}_{p^\ell})$ には加法を定義可能であることがわかっており、これを $+$ で表す。そして、$E(\mathbb{F}_{p^\ell})$ は単位元を ∞、点 $P = (x,y) \in E(\mathbb{F}_{p^\ell})$ の逆元 $-P$ を $-P := (x,-y)$ とする群（アーベル群）を成す。整数 α について、$[\alpha]P$ は、$\alpha > 0$ ならば、$[\alpha]P := \sum_{i=1}^{\alpha} P$、$\alpha < 0$ ならば、$[\alpha]P := [-\alpha](-P)$、

$\alpha = 0$ ならば、$[\alpha]P := \infty$ とする。$E(\mathbb{F}_{p^\ell})$ の要素数を群位数と呼び、$\#E(\mathbb{F}_{p^\ell})$ と書く。n を正の整数とする。E の n-ねじれ点群 $E[n]$ とは、$E[n] := \{(x,y) \in E \mid [n](x,y) = \infty\}$ である。また、$E/\mathbb{F}_p$ の p^ℓ-フロベニウス準同型写像 π_{p^ℓ} とは、$\pi_{p^\ell}((x,y)) := (x^{p^\ell}, y^{p^\ell})$ である。$E/\mathbb{F}_p$ について、$\#E(\mathbb{F}_p) = p + 1 - t$ となるような整数 t が存在する。この t は（フロベニウスの）トレースと呼ばれ、$|t| \leq 2\sqrt{p}$ を満たす。

ある $E/\mathbb{F}_p$ について、ある非平方数を D、ある整数を f として、$Df^2 = 4p - t^2$ という方程式を構成できる。これは Complex Multiplication（CM）方程式と呼ばれ、D は E の CM 判別式と呼ばれる。逆に CM 方程式が存在する場合、これに対応するような楕円曲線 E が存在する。CM 方程式から E を求める方法は CM 法 [13] と呼ばれる。

ペアリング： r を素数とし、$\gcd(r,p) = 1$ および $r \mid \#E(\mathbb{F}_p)$ および $r^2 \nmid \#E(\mathbb{F}_p)$ を満たすものとする。$r \mid (p^k - 1)$ となるような最小の正整数 k を、$E/\mathbb{F}_p$ の位数 r に関する埋め込み次数と呼ぶ。μ_r を 1 の r 乗根の集合とする。定義より、$\mu_r \subseteq \mathbb{F}_{p^k}$ である。$\mathbb{G}_1 := E[r] \cap \ker(\pi_p - [1])$ および $\mathbb{G}_2 := E[r] \cap \ker(\pi_p - [p])$ とする。この時、非退化かつ双線形性を持ち、効率的に計算可能な写像 $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mu_r$ が存在する [11]。この e を**ペアリング**と呼ぶ。

$P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ とする。Q より定まる正規化された E 上の次数 r のある有理関数 $f_{r,Q}$ を用いて、ペアリングの計算は $f_{r,Q}(P)^{\frac{p^k-1}{r}}$ と記述できる。次数 T の有理関数の評価値 $f_{T,Q}(P)$ と $[T]Q$ を求めるアルゴリズムは Miller のアルゴリズムと呼ばれる。また、$(p^k - 1)/r$ 乗算は最終べき乗算と呼ばれる。どちらも符号付きバイナリ法（または signed double-and-add 法とも呼ばれる）による反復計算で値を求める。Miller のアルゴリズムにおいては有理関数の次数 T が反復計算のパラメータであり、$O(\log T)$ 回程度の有限体演算で計算できる。最終べき乗算も同じく符号付きバイナリ法により計算できる。しかし、r や p^k が非常に巨大であるため、ペアリングの計算コストは大きい。高速にペアリングを計算可能な楕円曲線を選ぶことが、実用的なペアリング暗号の実装において重要な課題となる。

離散対数問題： 位数が素数 r の巡回群 $\mathbb{G}$ について、$\mathbb{G}$ の群演算をここでは乗法 $\cdot$ で記述する。$g, h \in \mathbb{G}$ を入力に取り、$g^x = h$ となるような整数 x を求める問題を**離散対数問題**（*Discrete Logarithm Problem, DLP*）と呼ぶ。$\mathbb{G}$ が有限体の乗法群の部分群ならば、この問題は**有限体離散対数問題**（*Finite Field DLP, FFDLP*）と呼ばれ、楕円曲線の有理点群の部分群ならば、**楕円曲線離散対数問題**（*Elliptic Curve DLP, ECDLP*）と呼ばれる。ペアリング $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mu_r$ を使用するペアリング暗号では、その安全性を保証するためには、使用する楕円曲線 $E/\mathbb{F}_p$ より定義される $\mathbb{G}_1$, $\mathbb{G}_2$, μ_r における全ての DLP が計算困難であることが要求される。一般的な FFDLP の最速の求解法は数体篩法または関数体篩法であり、求解にかかる時間は $\log p^k$ の準指数オーダーである。一般的な ECDLP の最速の求解法は Pollard の ρ 法であり、求解にかかる時間は $\log r$ の指数オーダーである。よって、十分に大きい r と p および埋め込み次数 k を持つ楕円曲線を使用することが、安全なペアリング暗号を実現するために重要である。

3. ペアリング暗号の効率的計算に適した楕円曲線

ここでは、ペアリング暗号の効率的な計算に適した楕円曲線の生成方法や、その要件を解説する。

曲線族： CM 法 [13] を利用することで、素数群位数 r や体位数 p および埋め込み次数 k を事前に定め、対応する楕円曲線 E を生成できる。しかし、暗号技術として利用可能な適切な条件を満

たす r, p, k および E の組みを求めることは一般的に難しい。この問題を解決する方法として、曲線族を用いて効率的に適切な楕円曲線を探索する方法が提案されている [7]。曲線族を利用した探索方法では、k と CM 判別式 D を固定し、r、p、トレース t を自由に設定するのではなく、CM 方程式など必要な関係を満たす有理数係数多項式 $r(X)$, $p(X)$, $t(X)$ で表現されるものとして抽象化し、そして $r(z)$ と $p(z)$ が適切な楕円曲線 E を与える整数 z を探索する。この k, D, $r(X)$, $p(X)$, $t(X)$ の 5 つ組みを曲線族と呼び、代表的なものとして、Barreto–Lynn–Scott（BLS）曲線族や Kachisa–Schaefer–Scott（KSS）曲線族がある。例えば、埋め込み次数が 16、CM 判別式が 1 の KSS 曲線族 KSS16 は次の多項式により定義される:

$$\begin{aligned} t(X) &= (2X^5 + 41X + 35)/35, \qquad r(X) = (X^8 + 48X^4 + 625)/61250, \\ p(X) &= (X^{10} + 2X^9 + 5X^8 + 48X^6 + 152X^5 + 240X^4 + 625X^2 + 2398X + 3125)/980. \end{aligned} \tag{1}$$

なお、Guillevic [8, 9] は、いくつかの曲線族について、128 bit 安全性を持つ楕円曲線を生成するために必要な、曲線族に与える整数 z のビット長を見積もった。この見積もりによって得られる楕円曲線に関する r と p のビット長を、一部の曲線族について表 1 に示す。

効率的なペアリング計算法： Miller のアルゴリズムと最終べき乗算の両方に共通する効率化の基本的なアイデアは、素体 $\mathbb{F}_p$ の拡大体で高速に計算することができる p べき乗算を利用することである。Miller のアルゴリズムでは $f_{r,Q}(P)$ と $[r]Q$ を計算する。この時、r を p 進数展開 $r = \sum_{i=0}^{d} h_i p^i$ し、反復計算のパラメータが係数列 $h_0, \ldots, h_d$ となるようにアルゴリズムを変形できる。例えば、式 (1) に示した KSS16 については、ペアリング計算の効率的なアルゴリズムを次のような式で表現できる [15]:

$$e(P, Q) \coloneqq \left(\left(f_{x,Q}(P) \cdot \ell_{[x]Q,[p]Q}(P) \right)^{p^3} \cdot \ell_{Q,Q}(P) \right)^{\frac{p^{16}-1}{r}}. \tag{2}$$

ここで、$\ell_{A,B}$ は、直観的には楕円曲線 E 上の点 A と B を通る直線である（厳密な定義は割愛する）。最終べき乗算においても、同様に $(p^{16} - 1)/r$ の p 進数展開を考えることで高速化が可能である（展開式が複雑なため、ここでは割愛する）。

また、Miller のアルゴリズムと最終べき乗算、それぞれの反復計算パラメータの p 進数展開は、式 (2) に示すように、曲線族の多項式表現の形式で得ることが可能である。これにより、曲線族を用いて楕円曲線を生成する前に、整数 z を与えた際のペアリングの計算コストをある程度推定できる。必ずしも最速のペアリング計算が可能となる保証は無いが、いくつかの BLS 曲線族や KSS 曲線族、Guillevic [8] による見積もりの報告に挙げられている曲線族に共通する性質として、**曲線族に与える整数 z の符号付き *2* 進数展開について、そのハミング重みが小さければ、その楕円曲線におけるペアリングの計算コストも小さい傾向がある。**

zk-SNARK に適した楕円曲線： ペアリングに基づくいくつかの暗号方式は、その高速計算のために、使用する楕円曲線に対して特別な条件を要求するものがある。代表的なものがブロックチェーンにおけるプライバシ保護技術として利用されている**ゼロ知識簡潔非対話証明** *zk-SNARK (Zero-Knowledge Succinct Non-interactive ARgument of Knowledge)* [4] である。ここで、ペアリングの入出力となる 3 つの群 $\mathbb{G}_1$, $\mathbb{G}_2$, μ_r の群位数 r をある正の整数 κ とある奇数 u により $r - 1 = 2^\kappa u$ と表現する時、$\mathrm{adic}_2(r) \coloneqq \kappa$ とする。zk-SNARK 方式の高速計算を適用するためには $\mathrm{adic}_2(r)$ がある程度大きい必要があり、$\mathrm{adic}_2(r) \geq 30$ が望ましいとされている [4]。

第 1 表: Guillevic [9] が見積もった 128 bit 安全性を持つ楕円曲線の候補の一部。d はツイスト次数。ビット長が小さいほど性能が良い。

曲線族	r (bit)	p (bit)	$p^{k/d}$ (bit)	p^k (bit)
BN	446	446	892	5343
BLS12	299	446	892	5352

曲線族	r (bit)	p (bit)	$p^{k/d}$ (bit)	p^k (bit)
FM17	296	447	894	5356
KSS16	256	330	1320	5268

4. 探索プログラムの開発

ここでは、本課題で行ったプログラムの開発について述べる。

探索方針： 第 3 節で行った解説から、次のような楕円曲線の探索方針を定める。

方針 1 (zk-SNARK 方式の高速計算に適した楕円曲線)**.** 曲線族 $k, D, p(X), t(X), r(X)$ を使用して zk-SNARK 方式に適した高速にペアリングが計算可能な楕円曲線を得るためには、次のような条件を満たす整数 z を探索する:

1. $p(z)$ と $r(z)$ が同時に素数となり、そのビット長は十分に大きい。
2. $\mathrm{hw}(z)$ が小さい。
3. $\mathrm{adic}_2\big(r(z)\big)$ が大きい。特に $\mathrm{adic}_2\big(r(z)\big) \geq 30$ が望ましい。

表 1 に挙げている曲線族のうち、BN, BLS12（他の埋め込み次数の BLS 曲線族も含む）, FM17 の曲線族に対して探索方針 1 に基づき楕円曲線を探索することは容易である。なぜならば、それら曲線族の $r(X)$ の定数項は 1 であり、有理数係数多項式 $g(X)$ によって $r(X) - 1 = g(X) \cdot X$ と分解が可能なため、$\mathrm{adic}_2\big(r(z)\big)$ が大きくなるように z を設定することが比較的容易となる可能性が高いからである。しかし KSS16 の $r(X)$ の定数項は有理数であり、このような構造を持たない。よって、ハミング重みが小さく、$\mathrm{adic}_2\big(r(z)\big)$ が大きい楕円曲線を発見することは困難であると予想される。

本課題の目的は、上記のように、困難な探索方針と曲線族についても所望の楕円曲線が発見できるように、大規模に楕円曲線の探索を行うプログラムを開発することである。そして、実際に実験を行い、その結果を観察することで、最良の楕円曲線の選択に貢献することを目標とする。

探索アルゴリズム： 探索方針 1 に基づく探索を、次の手続きを並列に処理することで行う:

- 最上位ビット位置 m とハミング重み w を入力に取り、そのような符号付き 2 進数展開 $s_0, \ldots, s_m$ を生成し、さらにこれを対応する整数 $z = \sum_{i=0}^{m} s_i 2^i$ に変換する。
- 曲線族 $k, D, p(X), t(X), r(X)$ と整数 z を入力に取り、$p(z)$ と $r(z)$ が同時に素数となるか判定する。

ここで、符号付き 2 進数展開 $s_0, \ldots, s_m$ について、$s_0, \ldots, s_m$ の値のうち、非ゼロの個数が w となる展開全ての集合を $\mathsf{SB}_{m,w}$ と書く。なお、$\#\mathsf{SB}_{m,w} = 2^w \cdot \binom{m}{w-1}$ である。

開発および実験環境： 本課題では Oakbridge-CX [2] を使用した。その主な理由は、Python 処理系を拡張することで実装されている数式処理系 SageMath [3] の上でプログラムを開発し、開発工数を削減するためである。また、並列化を行うために mpi4py [1]（Python 用の MPI ラッパー）を使用した。

素数判定： 整数 z を代入した $p(z)$ と $r(z)$ が素数か否かを判定するために、SageMath が提供している `is_pseudoprime` 関数と `is_prime` 関数を使用した。前者は確率的な素数判定を行うため

高速に判定できる。そこで、最初に `is_pseudoprime` 関数を使用して素数判定を行い、これが真になった場合に続けて `is_prime` 関数を実行し、素数であるか否かを最終的に判定するような実装を行った。なお、$p(z)$ と $r(z)$ のどちらかが有理数となった場合や、判定の結果が偽となった場合には、処理を打ち切っている（early abort）。このように実装することで、$p(z)$ と $r(z)$ の素数判定の効率化を図った。

整数の並列生成： 探索を並列に行うためには、符号付き 2 進数展開 $s \in \mathsf{SB}_{m,w}$ を素直に 1 つずつ直列に生成するのではなく、並列に生成する必要がある。そのために、整数と $\mathsf{SB}_{m,w}$ の 1 対 1 対応を与える rank/unrank 関数（完全ハッシュ関数とその逆関数）を構成する。展開 $s \in \mathsf{SB}_{m,w}$ は、最大の非ゼロビット位置が m である w 個の非ゼロビットの位置とその各ビットの符号により表現可能である。つまり、$\langle m \rangle$ から $w-1$ 個の整数を選ぶ組み合わせにより展開の最上位ビット以外の非ゼロビット位置を生成し、次にその各ビットの符号の列 $\{-1,1\}^w$ を割り当てることにより、$\mathsf{SB}_{m,w}$ の要素を生成できる。そこで、m 個の要素から $w-1$ 個選ぶ組み合わせ（総数 $\binom{m}{w-1}$）と $\{-1,1\}^w$（総数 2^w）の 2 つの対象それぞれの rank/unrank 関数を利用し、$\langle \#\mathsf{SB}_{m,w} \rangle = \{0,\ldots,2^w \cdot \binom{m}{w-1} - 1\}$ と $\mathsf{SB}_{m,w}$ の要素の rank/unrank 関数を構成した。この unrank : $\langle \#\mathsf{SB}_{m,w} \rangle \to \mathsf{SB}_{m,w}$ を使用することで、任意の展開 $s = (s_0,\ldots,s_m) \in \mathsf{SB}_{m,w}$ と対応する整数 $z = \sum_{i=0}^{m} s_i 2^i$ を並列に生成できるようになる。

注意 1. 符号付き 2 進数展開による表現は冗長である。すなわち、1 つの整数 z に、$z = \sum_{i=0}^{m} s_i 2^i$（$i = \langle m+1 \rangle$ について $s_i \in \{-1,0,1\}$）となるような数値列 $s_0, s_1, \ldots, s_m$ が唯一とは限らない。そのため、上記の unrank 関数を用いて生成した展開 $s \in \mathsf{SB}_{m,w}$ より得られた整数 $z = \mathsf{SB2Int}(s)$ を求め、さらにその絶対値について符号付き 2 進数展開の 1 つである NAF（Non-adjacent form） [10] を求め、その結果得られた符号付き 2 進数展開を元々の z の正負に応じて各非ゼロの符号を反転させた $s'_0, s'_1, \ldots, s'_{m'}$ を z の符号付き 2 進数展開として使用した。よって、プログラム実行時に指定した最上位ビット位置 m とハミング重み w とは異なる最上位ビット位置 m' とハミング重み w' を持つ z が得られる場合がある。なお、NAF への変換を実行した結果得られた符号付き 2 進数展開について、最上位ビット位置 m から数えて 3 つのビットが $s_{m-2} = \mp 1, s_{m-1} = 0, s_m = \pm 1$ であった場合、これを $s_{m-2} = \pm 1, s_{m-1} = \pm 1, s_m = 0$ と、最上位ビット位置が 1 小さい表現に変更することができる。なお、この場合は $m' = m-1, w' = w$ となる。Miller のアルゴリズムや最終べき乗算は z のビット長が短い程（最上位ビット位置が小さい程）、計算コストが小さいため、このような変更を行うと有利となる可能性がある。

並列探索： 整数列 $\langle \#\mathsf{SB}_{m,w} \rangle$ を分割し、それらを MPI により管理されるプロセスに割り当て、unrank 関数を使用することで並列に探索を行うことができる。しかし、ランダムに取り出した展開 $s \in \mathsf{SB}_{m,w}$ について、$p\big(\mathsf{SB2Int}(s)\big)$ と $r\big(\mathsf{SB2Int}(s)\big)$ が同時に素数となる事象を考えた時、その分布は不明である。つまり、$\mathsf{SB}_{m,w}$ を MPI が管理するプロセス数に等分割し、これを各プロセスに割り当てて並列に探索を行った場合、作業量が一定となる保証が無いため、最も作業量が大きいプロセスが終了するまで探索処理が終了しない。本課題では、分布の特定ではなく、producer-consumer スタイルによる並列探索を行うことで効率化を図る。各プロセスには、producer と consumer どちらかの役割が与えられる。また、consumer の役割を与えるプロセスの数を多く取るようにする。そして、次に述べるような処理を並列に行う。$\mathsf{SB}_{m,w}$ を適切な大きさに分割する。Producer 役のプロセスは、分割された $\mathsf{SB}_{m,w}$ を管理し、consumer 役のプロセスに未探索の分割された整数列を割り当てる。Consumer 役のプロセスは、割り当てられた整数列を

第 2 表: 探索範囲 S_1 に対する探索結果

整数 z	$\mathrm{adic}_2(z)$	$\mathrm{hw}(z)$	r (bit)	p (bit)
$-2^{34}+2^{27}-2^{23}+2^{20}-2^{11}+2^{0}$ （[9] より）	4	6	257	330
$-2^{34}-2^{29}+2^{24}-2^{21}-2^{19}-2^{17}-2^{15}-2^{11}-2^{9}-2^{7}+2^{2}-2^{0}$	20	12	257	331
$2^{34}+2^{30}+2^{26}-2^{24}+2^{21}-2^{19}-2^{17}-2^{15}-2^{11}-2^{9}-2^{7}+2^{2}-2^{0}$	20	13	257	331
$-2^{34}-2^{32}-2^{30}-2^{28}-2^{25}-2^{19}+2^{17}+2^{15}+2^{11}+2^{9}+2^{7}-2^{2}+2^{0}$	20	13	260	335
$-2^{35}+2^{31}+2^{26}+2^{22}-2^{19}+2^{17}+2^{15}+2^{11}+2^{9}+2^{7}-2^{2}+2^{0}$	20	12	264	340
$2^{34}+2^{29}-2^{27}-2^{25}-2^{23}-2^{21}+2^{17}-2^{15}-2^{11}-2^{9}-2^{7}+2^{2}-2^{0}$	21	13	257	331
$2^{34}+2^{33}-2^{29}+2^{27}+2^{25}-2^{23}+2^{17}-2^{15}-2^{11}-2^{9}-2^{7}+2^{2}-2^{0}$	21	13	261	336
$-2^{35}-2^{31}-2^{25}+2^{21}-2^{19}+2^{17}-2^{15}-2^{11}-2^{9}-2^{7}+2^{2}-2^{0}$	22	12	265	341

第 3 表: 探索範囲 S_2 に対する探索結果

整数 z	$\mathrm{adic}_2(r)$	$\mathrm{hw}(z)$	r (bit)	p (bit)
$2^{78}-2^{76}-2^{28}+2^{14}+2^{7}+2^{0}$ （[8] より）	4	6	605	766
$2^{78}+2^{65}+2^{46}-2^{11}-2^{9}-2^{7}+2^{2}-2^{0}$	17	8	609	771

unrank 関数を用いて曲線族の多項式に代入すべき整数 z の展開に変換し、$p(z)$ と $r(z)$ が同時に素数になる整数 z を保存する。そして、割り当てられた整数列に対する全ての処理が終わった際には、次に探索すべき整数列を producer 役のプロセスに対して要求する。このような方法で作業量を均等化し、並列探索の性能向上を図った。

5. 実験

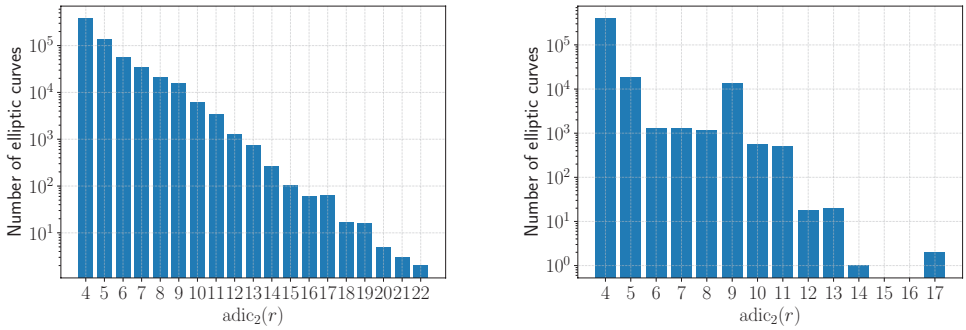
ここでは、開発したプログラムを用いて実施した探索実験について述べる。

探索対象の設定： 曲線族として KSS16 を用いて探索を行った。探索対象となる整数の集合 $\mathsf{SB}_{m,w}$ の最上位ビット位置 m は、Guillevic [8] による見積もりを参考に、128 bit 安全性と 192 bit 安全性の達成に必要な最上位ビット位置に近い値とした。具体的には $m \in \{33, 34, 35, 77, 78\}$ と設定した。ハミング重みについては、$w = 1$ から順に 1 つずつ増やしながら設定し、探索を行った。

探索結果の概要： 最上位ビット位置 $m \in \{33, 34\}$ についてはハミング重み $w \in \{1, \ldots, 13\}$ まで、$m = 35$ については $w \in \{1, \ldots, 12\}$ まで探索を行った。この探索が完了した範囲を S_1 と書く。S_1 に対する探索について、重複を排除せずに数え上げて 12 105 138 309 966 個（$\approx 2^{44}$ 個）の整数を列挙し、657 578 個（$\approx 2^{20}$）の楕円曲線を発見した。しかし、発見した楕円曲線のうち、最大の $\mathrm{adic}_2(r)$ は 22 であった。

最上位ビット位置 $m \in \{77, 78\}$ についてはハミング重み $w \in \{1, \ldots, 8\}$ まで探索を行った。この探索が完了した範囲を S_2 と書く。S_2 に対する探索について、重複を排除せずに数え上げて 2 137 041 827 694 個（$\approx 2^{41}$ 個）の整数を列挙し、434 731 個（$\approx 2^{19}$）の楕円曲線を発見した。発見した楕円曲線のうち、最大の $\mathrm{adic}_2(r)$ は 17 であった。

探索範囲 S_1 に対する探索により発見した楕円曲線のうち、$\mathrm{adic}_2(r) \geq 20$ であり、群位数が 256 bit 以上の楕円曲線を与える整数 z を表 2 に示す。また、S_2 に対する探索により発見した楕円曲線のうち、$\mathrm{adic}_2(r) \geq 16$ であり、体位数が 766 bit 以上の楕円曲線を与える整数 z を表 3 に示す。なお、表 2 と表 3 それぞれについて、比較のためにハミング重みが小さい整数 z も示す。S_1 と S_2 それぞれについて、発見した楕円曲線の $\mathrm{adic}_2(r)$ の頻度分布を図 1 に示す。



(a) S_1 の探索結果　　(b) S_2 の探索結果

第 1 図: 発見した楕円曲線の $\mathrm{adic}_2(r)$ の頻度分布

考察： 本課題では $\mathrm{adic}_2(r) \geq 30$ となるような楕円曲線を発見できなかった。そこで、実験結果を元に発見するために必要な計算量を見積もる。探索範囲 S_1, S_2 について、発見した楕円曲線の個数の $\log_2$ の値と最大の $\mathrm{adic}_2(r)$ の値が比較的近いこと、そして、図 1a に示す S_1 の探索結果の対数頻度分布が線形に減少していることから、楕円曲線の群位数 r を符号無し 2 進数展開 $r = \sum_{i=0}^{m} b_i 2^i$ （ここで $i = \langle m+1 \rangle$ について $b_i \in \{0, 1\}$）を考えた時、探索により発見される楕円曲線の r の下位のビット $b_0, b_1, \ldots$ の分布は独立一様であると仮定する。このような仮定を置いた場合、$\mathrm{adic}_2(r) = \gamma$ となるような楕円曲線を発見するためには 2^γ 個の楕円曲線を発見する必要があると考えられる。目標は $\gamma \geq 30$ であり、今回実施した実験では、探索範囲 S_1 と S_2 それぞれについて $2^{30-20} = 1024$、$2^{30-19} = 2048$ となるため、今回使用した 17 280 ノード時間の 2048 倍の 35 389 440 ノード時間の計算量を投じれば、$\mathrm{adic}_2(r) \geq 30$ となる楕円曲線を発見できるという見積もりとなる。この見積もりは、探索方針 1 に示すような $\mathrm{hw}(z)$ が小さく $\mathrm{adic}_2\big(r(z)\big)$ が大きい楕円曲線を本課題で実装したアルゴリズムで探索することは難しく、さらなる高速化やアルゴリズムの改良が必要であることを示唆している。

注意 2. 上記の見積もりは根拠が不明瞭な経験則に基づくものであり、背後にある数学的な構造を明らかにしていないこと、作成したプログラムのプロファイリングに基づく見積もりではないため、実際には大きな誤差が生じる可能性や逸脱する可能性が否定できないことに注意。

また、表 2 および表 3 に示す整数 z の符号付き 2 進数展開による表現を観察すると、ビット位置 0 から 17 付近までに共通あるいは類似のパターンが表れていることがわかる。これは、探索空間にある整数のうち、適切な楕円曲線を与え、その $\mathrm{adic}_2(r)$ が高い整数の分布には何らかの構造が存在することを示唆している。これを明らかにし、整数の生成に活用することができれば、探索アルゴリズムの改良が可能となるかもしれない。

6. まとめ

本課題では、探索方針 1 に基づく並列に楕円曲線を探索するプログラムを作成した。これを用いて、KSS16 曲線族に対して探索を行った。ある程度の範囲について探索を行ったが、KSS16 について $\mathrm{adic}_2(r)$ が十分に大きい楕円曲線を発見することはできなかった。また、本課題で実装したアルゴリズムを用いて探索方針 1 に基づく探索、すなわち、$\mathrm{hw}(z)$ が小さく $\mathrm{adic}_2\big(r(z)\big)$ が大き

い適切な楕円曲線を与える整数 z を探索することは難しく、さらなる高速化あるいはアルゴリズムの改良が必要であると考えられる。また、実験により得られた整数には、一定のパターンがあることがわかった。これを解析し、アルゴリズムの改良に活かすことの検討が今後の課題である。今回は Cheon の攻撃法 [5] を考慮していない。この攻撃法の対象となる zk-SNARK 方式が存在するため、Cheon の攻撃法を考慮したパラメータや探索方針の調整を検討することも、今後の課題である。

参 考 文 献

[1] "mpi4py · pypi," https://pypi.org/project/mpi4py/.

[2] "Oakbridge-CX スーパーコンピュータシステム," https://www.cc.u-tokyo.ac.jp/supercomputer/obcx/service/.

[3] "SageMath – open-source mathematical software system," https://www.sagemath.org/.

[4] E. Ben-Sasson et al., "SNARKs for C: verifying program executions succinctly and in zero knowledge," in *CRYPTO 2013*, 2013, pp. 90–108.

[5] J. H. Cheon, "Discrete logarithm problems with auxiliary inputs," *J. Cryptol.*, vol. 23, no. 3, pp. 457–476, 2010.

[6] N. El Mrabet and M. Joye, Eds., *Guide to Pairing-Based Cryptography*. Chapman and Hall/CRC, 2016.

[7] D. Freeman et al., "A taxonomy of pairing-friendly elliptic curves," *J. Cryptol.*, vol. 23, no. 2, pp. 224–280, 2010.

[8] A. Guillevic, "A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level," in *PKC 2020*, 2020, pp. 535–564.

[9] ——, "Pairing-friendly curves," https://members.loria.fr/AGuillevic/pairing-friendly-curves/, 2021, (Accessed 2021/06/04).

[10] D. Hankerson et al., *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., 2004.

[11] F. Hess, "Pairing lattices," in *Pairing 2008*, 2008, pp. 18–38.

[12] A. Joux, "A one round protocol for tripartite Diffie-Hellman," *J. Cryptol.*, vol. 17, no. 4, pp. 263–276, 2004.

[13] K. Rubin and A. Silverberg, "Choosing the correct elliptic curve in the CM method," *Math. Comput.*, vol. 79, no. 269, pp. 545–561, 2010.

[14] R. Sakai et al., "Cryptosystems based on pairing," in *SCIS 2000*, January 2000, pp. 26–28, in Japanese.

[15] X. Zhang and D. Lin, "Analysis of optimum pairing products at high security levels," in *INDOCRYPT 2012*, 2012, pp. 412–430.