初期化付き量子Karatsuba乗算と並列逆元計算によるバイナリ楕円曲線の量子楕円曲線加算回路の効率化

肖 懐遠[1,a)]　高安 敦[1,2]

概要：バイナリ楕円曲線上の離散対数問題を解く量子アルゴリズムにおいて，量子加算回路のリソースが支配的である．さらに，既存研究では楕円曲線加算のリソースは 2 回の有限体上の逆元計算が支配的であり，逆元計算は有限体上の乗算を繰り返す．Jang ら (CHES'25) は，計算過程を消去し切らない量子 Karatsuba 乗算を活用し，従来より圧倒的に多数の量子ビット数を必要とするが，大幅に深さを削減する量子加算回路を提案した．本回路は量子ビット数と深さの積が既存研究より小さいという点で，他の既存研究より良いトレードオフを達成している．本研究で我々は，Jang らの量子加算回路の量子ビット数と深さを共に削減する量子加算回路を提案する．まず，既存手法を組み合わせて，最も効率的な量子逆元計算アルゴリズムを構成する．特に，計算過程を全て初期化する量子 Karatsuba 乗算回路を用いて，量子ビット数と深さの両方を削減する．さらに，量子楕円曲線加算における 2 回の逆元計算を並列に実行することで，深さを削減する．結果として，これらの提案手法によって，Jang らと比べて $n = 571$ のときに Shor のアルゴリズムの量子ビット数と深さをそれぞれ 79.2% と 61.6% 削減し，それらの積を 92.0% 削減する．

Improving Quantum Circuits of Shor's Algorithm for Binary Elliptic Curves

Kaien Sho[1,a)]　Atsushi Takayasu[1,2]

Abstract: In the quantum algorithm for solving the discrete logarithm problem on binary elliptic curves, the resource cost of the quantum point addition circuit is dominant. Existing work shows that point addition is bottlenecked by two finite-field inversions, each implemented via repeated multiplications. Jang et al. (CHES'25) introduced Karatsuba multiplication that omits full uncomputation of intermediates, trading substantially more qubits for dramatically reduced depth and yielding a better qubit-depth product than prior methods. We propose a quantum point addition circuit that reduces both qubit count and depth compared to Jang et al.'s circuit. First, we employ Karatsuba multiplication with full uncomputation of intermediates, significantly cutting total qubits. Although uncomputation adds depth, we execute the two inversions in point addition in parallel to offset this. For degree $n = 571$, our circuit achieves a 79.2% qubit-count reduction and a 61.6% depth reduction in Shor's algorithm relative to Jang et al.'s implementation.

1. はじめに

1.1 背景

楕円曲線暗号 [Mil86], [Kob87] は RSA 暗号 [RSA78] と並んで，最も広く利用されている公開鍵暗号方式の一つである．NIST によって素体上の楕円曲線と有限体 $\mathbb{F}_{2^n}$ 上のバイナリ楕円曲線の使用が推奨され，さらに，後者では拡大次数 $n = 163, 233, 283, 571$ が推奨されている．楕円曲線暗号の安全性を保証するためには楕円曲線上の離散対数問題（Elliptic Curve Discrete Logarithm Problem, ECDLP）が計算量的に困難である必要があり，古典アルゴリズムでは多項式時間では解けないと信じられている．ただし，Shor の量子アルゴリズム [Sho94] は多項式時間で ECDLP を解くことができる．耐量子計算機暗号への

[1] 東京大学
The University of Tokyo
[2] 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
[a)] sho-kaien1023@g.ecc.u-tokyo.ac.jp

移行の必要性を確認するために，ECDLP を解く量子アルゴリズムの効率的な量子回路の設計やそのリソース解析が近年活発に研究されている．特に，本論文では素体上の ECDLP [RNSL17], [HJN+20] ではなく，より活発に研究されているバイナリ楕円曲線を対象とする [BBvHL21], [PWLK22], [KH23], [TT23], [TT24], [JSB+25].

Banegas ら [BBvHL21] は, バイナリ ECDLP を解く Shor のアルゴリズムの量子回路を初めて設計した．そのリソースは $2n$ 回計算する楕円曲線加算が支配的であり，楕円曲線加算のリソースは 2 回の逆元計算が支配的である．実際，これまで Banegas らとは異なる量子楕円曲線加算を用いた改良量子回路が提案されている [TT24], [JSB+25] が，いずれにおいても量子楕円曲線加算の深さが量子逆元計算の深さの 2 倍程度であることは同じで，改良の本質は量子逆元計算の改良であった．Banegas らは GCD (greatest common divisor) 法と FLT (Fermat's little theorem) 法に基づく量子逆元計算アルゴリズムを提案していた．その後，GCD 法の改良 [KH23] も提案されてはいるが，FLT 法に基づく様々な改良が提案されてきた [PWLK22], [TT23], [TT24]．FLT 法は $\mathbb{F}_{2^n}$ 上の乗算と 2 冪乗算を繰り返す逆元計算法だが，これらの研究では補助量子ビットを必要としない量子省メモリ Karatsuba 乗算 [BBvHL21], [PWLK22], [KKKH24] を用いている．ここで述べた研究では，GCD 法と FLT 法いずれにおいても数千量子ビット程度で Shor のアルゴリズムを実行する量子回路を構成している．

Jang ら [JSB+25] は，FLT 法に基づき前述の研究とは一線を画す量子回路を提案した．Jang らの量子回路は，数万から数十万の量子ビットを用いるが，圧倒的に回路の深さを削減し，量子ビット数 M と深さ D の積 MD が既存研究で最小になるという点で改良を達成している．実際，ECDLP より研究の進んでいる素因数分解の量子回路設計でも，最先端の研究 [GE21] は MD の最小化を目指しており，Jang ら [JSB+25] の改良は実用上意味のある指標に基づいていると言える．従来のバイナリ ECDLP の量子回路 [BBvHL21], [PWLK22], [KH23], [TT23], [TT24] は基本的に量子ビット数の最小化を目指す研究が多かったが，Jang ら [JSB+25] は深さの最小化を目指した設計であった．そのため, Banegas ら [BBvHL21] と比べたとき，Jang ら [JSB+25] は従来の改良 [PWLK22], [TT23], [TT24] とは異なる方針に従っており，補助量子ビットを用いて一部の計算を並列化することで深さを削減した．さらに，計算過程で用いた Jang らの量子 Karatsuba 乗算 [JKL+23] も深さの削減に大きく寄与している．この乗算は，量子省メモリ Karatsuba 乗算 [BBvHL21], [PWLK22], [KKKH24] と違い補助量子ビットを必要とする．さらに，乗算後に全ての補助量子ビットを初期化はせず，深さを犠牲にしない一部の補助量子ビットのみを初期化することで可能な限り深さを削減している．

1.2 成果

本研究で, 我々は量子ビット数 M と深さ D の積 MD が既存研究 [BBvHL21], [PWLK22], [KH23], [TT23], [TT24], [JSB+25] と比べて最小となるバイナリ ECDLP の量子回路を提案する．これまで MD が最小だった Jang らの量子回路 [JSB+25] は，従来研究より M は大きくなったが，D を削減していた．一方で，我々は Jang らの量子回路の M と D の両方を削減することで MD を最小化する．

我々は, まず従来研究と同様, 量子 FLT 逆元計算を改良する．Banegas ら [BBvHL21] の量子 FLT 逆元計算を基本とし, 前述のように Jang ら [JSB+25] は他の改良 [PWLK22], [TT23], [TT24] とは異なる方針に従っていたが, 我々は Jang らの改良 [JSB+25] と Taguchi と Takayasu の改良 [TT23], [TT24] が相補的であることを確認し，これらを組み合わせて量子 FLT 逆元計算を提案する．さらに，我々はここで Jang らの量子 Karatsuba 乗算 [JKL+23] を用いず，初期化付き量子 Karatsuba 乗算を提案する．提案乗算は，Jang ら [JKL+23] と違い乗算後に全ての補助量子ビットを初期化するため，乗算 1 回の比較では M と D の両方が大きくなってしまう．ただし，FLT 逆元計算においては乗算と 2 冪乗算を複数回繰り返すことに着目し，一部の乗算では補助量子ビットを初期化しても深さが悪化しないことを確認する．結果として，逆元計算の提案手法は Jang ら [JSB+25] より M と D の両方を削減する．

提案量子 FLT 逆元計算によって Jang ら [JSB+25] より M と D の両方を削減するバイナリ ECDLP の量子回路を構成可能だが，我々はさらに量子楕円曲線加算を改良する．従来の量子楕円曲線加算 [BBvHL21], [TT24], [JSB+25] のリソースは 2 回の逆元計算が支配的で，楕円曲線加算と逆元計算の量子ビット数はほぼ同じで，楕円曲線加算の深さは逆元計算の 2 倍程度であった．これは，バイナリ ECDLP の既存の全ての量子回路 [BBvHL21], [PWLK22], [KH23], [TT23], [TT24], [JSB+25] に共通する特徴であった．本研究では，2 回の逆元計算を並列に実行することで楕円曲線加算の深さを削減する．より詳細には，逆元計算は補助量子ビットへの計算過程の書き込みとその初期化の 2 段階だが，楕円曲線加算において 1 回目の逆元計算の補助量子ビットの初期化と 2 回目の逆元計算の補助量子ビットへの書き込みを並列に実行する．これによって，深さが逆元計算の 2 倍以下になる量子楕円曲線加算を初めて提案し，バイナリ ECDLP の改良量子回路を得る．例として，$n = 571$ のとき，提案手法は Jang らの手法 [JSB+25] に基づく Shor のアルゴリズムの MD を 92.0% 削減する．

1.3 構成

第 2 章で，本論文を読むための基礎事項を説明する．第 3 章で逆元計算，第 4 章で初期化付き量子 Karatsuba 乗算を提案する．さらに，第 5 章で並列逆元計算について説明

し，第 6 章でバイナリ ECDLP の量子回路を既存研究と比較する．

2. 準備

本章では，本論文に必要となる背景知識について述べる．

2.1 バイナリ楕円曲線上の離散対数問題

自然数 n に対して，有限体 $\mathbb{F}_{2^n}$ は $\mathbb{F}_2$ 上の n 次既約多項式 $m(x)$ を用いて，$\mathbb{F}_2[x]/m(x)$ と表すことができる標数 2 の拡大体である．この有限体 $\mathbb{F}_{2^n}$ 上で定義されるバイナリ楕円曲線 $B_{a,b}$ は，Weierstrass 標準形

$$B_{a,b} : y^2 + xy = x^3 + ax^2 + b$$

を満たす座標のペア $(x, y) \in \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ と，無限遠点 $\mathcal{O}$ から構成される．ここで，$a, b \in \mathbb{F}_{2^n}$ であり，$b \neq 0$ である．

$B_{a,b}(\mathbb{F}_{2^n})$ 上の点の間には無限遠点 $\mathcal{O}$ を単位元とする加法が定義され，$B_{a,b}(\mathbb{F}_{2^n})$ はアーベル群となる．具体的には，$Q \neq \pm P$ を満たす 2 つの点 $P = (x_1, y_1), Q = (x_2, y_2) \in B_{a,b}(\mathbb{F}_{2^n})$ の楕円曲線加算 $R = P + Q = (x_3, y_3)$ は，$\lambda = (y_1+y_2)/(x_1+x_2)$ を用いて $x_3 = \lambda^2+\lambda+x_1+x_2+a$, $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$ となる．k 個の P の和を $[k]P$ と書くことにする．楕円曲線離散対数問題 (Elliptic Curve Discrete Logarithm Problem, ECDLP) とは，基点 $P \in B_{a,b}(\mathbb{F}_{2^n}) \setminus \{\mathcal{O}\}$ と $Q \in \langle P \rangle$ が与えられたとき，$Q = [k]P$ を満たす整数 $k \in [0, n-1]$ を求める問題である．

2.2 量子アルゴリズム

Shor のアルゴリズム [Sho94] は，ECDLP を多項式時間で解くことができる．量子計算の基本単位である量子ビットは，$|0\rangle$ と $|1\rangle$ の状態の重ね合わせとして $|\alpha|^2 + |\beta|^2 = 1$ を満たす $\alpha, \beta \in \mathbb{C}$ を用いて $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ と記述される．量子回路では，量子ゲートと呼ばれる可逆な演算子を用いて量子ビットの状態を変化させ計算をする．代表的な量子ゲートにはそれぞれ図 1 と図 2 で示した CNOT ゲートと Toffoli ゲートがある．Jang ら [JSB+25] に従い，我々は量子回路のリソースを量子ビット数 M と深さ D で評価する．ただし，Toffoli ゲートはさらに基本的なゲートに分解し，深さを 8 とする [AMMR13].

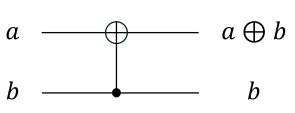

図 1 CNOT ゲート

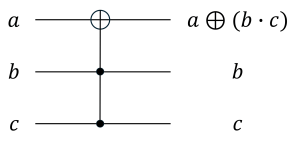

図 2 Toffoli ゲート

2.3 $\mathbb{F}_{2^n}$ 上の量子演算

Shor のアルゴリズム実装のためには量子楕円曲線加算を計算する必要があり，そのためには，$\mathbb{F}_{2^n}$ 上の量子演算を計算する必要がある．例として，Jang らの量子楕円曲線加算 [JSB+25] を Algorithm 1 に示す．

Algorithm 1 Jang らの量子楕円曲線加算アルゴリズム

Input: 量子状態: 制御ビット q，点 $P_1(x_1, y_1)$，初期化された補助量子ビット
古典状態: 楕円曲線のパラメータ a，固定点 $P_2(x_2, y_2)$

Output: $q = 1$ の場合 $P_1 + P_2 = P_3(x_3, y_3)$，$q = 0$ の場合 $P_1(x_1, y_1)$，初期化された補助量子ビット

1: $\mathrm{ADD}(x_1, x_2) = (x_1 + x_2, x_2)$
2: $\mathrm{CTRL\_ADD}(y_1, y_2, q) = (y_1 + q \cdot y_2, y_2, q)$
3: $\mathrm{INV\_CAL}(x_1 + x_2, 0, 0) = (x_1 + x_2, (x_1 + x_2)^{-1}, *)$
4: $\mathrm{MUL}((x_1 + x_2)^{-1}, y_1 + q \cdot y_2, 0)$
$= ((x_1 + x_2)^{-1}, y_1 + q \cdot y_2, \lambda(= (y_1 + q \cdot y_2)/(x_1 + x_2)))$
5: $\mathrm{INV\_INI}(x_1 + x_2, (x_1 + x_2)^{-1}, *) = (x_1 + x_2, 0, 0)$
6: $\mathrm{MUL}(x_1 + x_2, \lambda, y_1 + q \cdot y_2) = (x_1 + x_2, \lambda, 0)$
7: $\mathrm{SQR}_1(\lambda, 0) = (\lambda, \lambda^2)$
8: $\mathrm{ADD}(0, \lambda^2 + \lambda + a + x_2) = (\lambda^2 + \lambda + a + x_2, \lambda^2 + \lambda + a + x_2)$
9: $\mathrm{CTRL\_ADD}(x_1 + x_2, \lambda^2 + \lambda + a + x_2, q)$
$= (x(= x_1 + x_2 + q \cdot (\lambda^2 + \lambda + a + x_2)), \lambda^2 + \lambda + a + x_2, q)$
10: $\mathrm{ADD}(\lambda^2 + \lambda + a + x_2, \lambda^2 + \lambda + a + x_2) = (0, \lambda^2 + \lambda + a + x_2)$
11: $\mathrm{SQR}_1^{\dagger}(\lambda, \lambda^2) = (\lambda, 0)$
12: $\mathrm{MUL}(x, \lambda, 0) = (x, \lambda, x \cdot \lambda)$
13: $\mathrm{INV\_CAL}(x, 0, 0) = (x, x^{-1}, *)$
14: $\mathrm{MUL}(x \cdot \lambda, x^{-1}, \lambda) = (x \cdot \lambda, x^{-1}, 0)$
15: $\mathrm{INV\_INI}(x, x^{-1}, *) = (x, 0, 0)$
16: $\mathrm{ADD}(x, x_2) = (x_1 + q \cdot (x_1 + x_3), x_2)$
17: $\mathrm{CTRL\_ADD}(x \cdot \lambda, y_2 + x_1 + q \cdot (x_1 + x_3), q)$
$= (y_1 + q \cdot (y_1 + y_3), y_2 + x_1 + q \cdot (x_1 + x_3), q)$

$\mathbb{F}_{2^n}$ 上の量子演算を説明する．$\mathbb{F}_{2^n}$ の元は，n 量子ビットの多項式で表すことができる．$\mathbb{F}_{2^n}$ 上の加算 $\mathrm{ADD}(f, g) = (f+g, g)$ は n 個の CNOT ゲートを用いて深さ 1 で実装でき，制御加算 $\mathrm{CTRL\_ADD}(f, g, q) = (f+q \cdot g, g, q)$ は n 個の Toffoli ゲートを用いて深さ $8n$ で実装できる．詳細は省略するが，2^α 乗算は最大 n^2 個の CNOT ゲートで深さ最大 n で実装可能である [TT24]．1 量子ビットの乗算 $\mathrm{CTRL\_ADD}(f, g, q) = (f + q \cdot g, g, q)$ は 1 つの Toffoli ゲートで実装できる．一方，$\mathbb{F}_{2^n}$ 上の乗算 $\mathrm{MUL}(f, g, h) = (f, g, h + f \cdot g)$ は複雑であり，第 4 章で説明する．また，$\mathbb{F}_{2^n}$ 上の逆元計算は量子楕円曲線加算において最も支配的な計算であり，第 3 章で説明する．

3. 逆元計算

本章で，量子 FLT 逆元計算アルゴリズムを提案する．提案アルゴリズムは，Jang ら [JSB+25] と Taguchi と Takayasu [TT23], [TT24] の量子 FLT 逆元計算アルゴリズムを組み合わせた構成であるため，技術的には自然であり本論文の改良の本質は第 4 章以降の内容である．

提案手法の計算過程は，Taguchi と Takayasu の加法連鎖を用いた FLT 逆元計算 [TT23] に基づくが，特に $n = 163, 233, 283, 571$ ににおいては Taguchi と Takayasu が用いた加法連鎖列 [TT24] を用いることで量子ビット数と深さを削減する．長さ ℓ の加法連鎖列 $\alpha_0 = 1, \alpha_1, \ldots, \alpha_\ell$ とは，全ての $1 \leq k \leq \ell$ を満たす k に対して $0 \leq i, j < k$ を

Algorithm 2 $n = 8$ における逆元計算アルゴリズム

Input: 入力 $f \in \mathbb{F}_{2^n}$, 出力用量子ビット 0, 初期化された補助量子ビット

Output: $f, f^{-1} = f^{2^8-2}$, 計算過程が書き込まれた補助量子ビット

1: $\mathrm{SQR}_1(f^{2^1-1}, 0) = (f^{2^1-1}, f^{2^2-2^1})$
2: $\mathrm{MUL\_CAL}(f^{2^1-1}, f^{2^2-2^1}, 0, 0) = (f^{2^1-1}, f^{2^2-2^1}, f^{2^2-1}, *)$
3: $\mathrm{MUL\_INI}(f^{2^1-1}, f^{2^2-2^1}, *) = (f^{2^1-1}, f^{2^2-2^1}, 0)$
4: $\mathrm{SQR}_1(f^{2^2-1}, 0) = (f^{2^2-1}, f^{2^3-2^1})$
5: $\mathrm{MUL\_CAL}(f^{2^1-1}, f^{2^3-2^1}, 0, 0) = (f^{2^1-1}, f^{2^3-2^1}, f^{2^3-1}, *)$
6: $\mathrm{MUL\_INI}(f^{2^1-1}, f^{2^3-2^1}, *) = (f^{2^1-1}, f^{2^3-2^1}, 0)$
7: $\mathrm{SQR}_2(f^{2^2-1}, 0) = (f^{2^2-1}, f^{2^4-2^2})$
8: $\mathrm{MUL\_CAL}(f^{2^2-1}, f^{2^4-2^2}, 0, 0) = (f^{2^2-1}, f^{2^4-2^2}, f^{2^4-1}, *)$
9: $\mathrm{SQR}_1^\dagger(f^{2^1-1}, f^{2^2-2^1}) = (f^{2^1-1}, 0)$
10: $\mathrm{MUL\_INI}(f^{2^2-1}, f^{2^4-2^2}, 0) = (f^{2^2-1}, f^{2^4-2^2}, *)$
11: $\mathrm{SQR}_4(f^{2^3-1}, 0) = (f^{2^3-1}, f^{2^7-2^4})$
12: $\mathrm{MUL\_CAL}(f^{2^4-1}, f^{2^7-2^4}, 0, 0) = (f^{2^4-1}, f^{2^7-2^4}, f^{2^7-1}, *)$
13: $\mathrm{SQR}_1^\dagger(f^{2^2-1}, f^{2^3-2^1}) = (f^{2^2-1}, 0)$
14: $\mathrm{MUL\_INI}(f^{2^4-1}, f^{2^7-2^4}, 0) = (f^{2^4-1}, f^{2^7-2^4}, *)$
15: $\mathrm{SQR}_2^\dagger(f^{2^2-1}, f^{2^4-2^2}) = (f^{2^2-1}, 0)$
16: $\mathrm{SQR}_4^\dagger(f^{2^3-1}, f^{2^7-2^4}) = (f^{2^3-1}, 0)$
17: $\mathrm{SQR}_1(f^{2^7-1}, 0) = (f^{2^7-1}, f^{2^8-2})$

満たす i, j が存在し，$\alpha_k = \alpha_i + \alpha_j$ が成り立つ数列である．$\alpha_\ell = n-1$ を満たす加法連鎖列を用いる $\mathbb{F}_{2^n}$ 上の FLT 逆元計算は，$f = f^{2^{\alpha_0}-1}$ を入力に，2 冪乗算 $\left(f^{2^{\alpha_i}-1}\right)^{2^{\alpha_j}} = f^{2^{\alpha_i+\alpha_j}-2^{\alpha_j}} = f^{2^{\alpha_k}-2^{\alpha_j}}$ と乗算 $f^{2^{\alpha_k}-2^{\alpha_j}} \cdot f^{2^{\alpha_j}-1} = f^{2^{\alpha_k}-1}$ を繰り返して $f^{2^{\alpha_1}-1}, \ldots, f^{2^{\alpha_\ell}-1} = f^{2^{n-1}-1}$ を順次計算し，最終的には Fermat の小定理より逆元 $\left(f^{2^{n-1}-1}\right)^2 = f^{2^n-2} = f^{-1}$ を得る．逆元計算時には補助量子ビットに $f^{2^{\alpha_1}-1}, \ldots, f^{2^{\alpha_\ell}-1}$ が保存されているが，Jang ら [JSB+25] と同様，他に補助量子ビットが $2n$ ビットあり，$f^{2^{\alpha_k}-1}$ を得る乗算で使用した 2 冪乗算結果 $f^{2^{\alpha_k}-2^{\alpha_j}}$ の消去と次の $f^{2^{\alpha_{k+1}}-1}$ を得る乗算で使用する $f^{2^{\alpha_{k+1}}-2^{\alpha_{j'}}}$ の 2 冪乗算を並列に計算することで深さを削減する．

提案量子逆元計算を示すための基本演算を説明する．乗算 MUL_CAL は，$\mathrm{MUL\_CAL}(f, g, h, 0) = (f, g, h + f \cdot g, *)$ なる入出力を持つ．ただし，最後の引数は乗算用の補助量子ビットを表し，$*$ は明示しないが乗算の計算過程が記録されている．乗算初期化 MUL_INI は，$\mathrm{MUL\_INI}(f, g, *) = (f, g, 0)$ なる入出力を持つ．第 4 章で詳しく述べるが，MUL_INI は乗算結果 $h + f \cdot g$ を必要としないことに注意されたい．また，MUL_CAL と MUL_INI は第 4 章で提案する量子 Karatsuba 乗算を用いる．2 冪乗算 SQR_α は $\mathrm{SQR}_\alpha(f, 0) = (f, f^{2^\alpha})$ なる入出力を持ち，2 冪乗算初期化 $\mathrm{SQR}_\alpha^\dagger$ は $\mathrm{SQR}_\alpha^\dagger(f, f^{2^\alpha}) = (f, 0)$ なる入出力を持つ．SQR_α と $\mathrm{SQR}_\alpha^\dagger$ は Taguchi と Takayasu の量子 2 冪乗算 [TT24] を用いる．これらを用いて，簡単のため $n = 8$ において $\mathrm{INV\_CAL}(f, 0, 0) = (f, f^{-1}, *)$ なる入出力を持つ提案逆元計算アルゴリズムを Algorithm 2 に示す．ただし，最後の引数は逆元計算用の補助量子ビットを表し，$*$ は明示しないが逆元計算の計算過程が記録されている．詳細は省略するが，$\mathrm{INV\_INI}(f, f^{-1}, *) = (f, 0, 0)$ なる入出力を持つ逆元計算初期化は，INV_CAL とほぼ同様に計算可能である．

表 1 提案手法と Jang ら [JSB+25] の量子除算アルゴリズムのリソース比較

n	アルゴリズム	量子ビット数	深さ
163	[JSB+25]	53,133	2,801
	提案手法	15,606	2,180
233	[JSB+25]	82,898	3,476
	提案手法	22,697	2,496
283	[JSB+25]	144,671	5,412
	提案手法	35,913	3,424
571	[JSB+25]	500,450	11,723
	提案手法	104,362	5,885

逆元計算は楕円曲線加算において除算の計算に用いる．簡潔に言えば，除算は入力が (f, g, h) で出力が $(f, g, h + f^{-1} \cdot g)$ となる演算で，Jang らの量子楕円曲線加算 [JSB+25] の Algorithm 1 の 3–5 行目と 13–15 行目のように INV_CAL で f^{-1} を計算し，MUL で $f^{-1} \cdot g$ を足し，INV_INI で逆元計算過程の補助量子ビットを初期化する．提案手法の性能の参考として，表 1 において $n = 163, 233, 283, 571$ の量子除算アルゴリズムのリソースを Jang ら [JSB+25] と比較している．提案手法では，量子ビットと深さの両方を改良している．紙面の都合で他の量子逆元計算 [BBvHL21], [PWLK22], [KH23], [TT23], [TT24] とはここで量子除算を比較していないが，これらの研究とは本質的に重要な Shor のアルゴリズムのリソースを第 6 章の表 3 で比較する．

4. 初期化付き量子 Karatsuba 乗算

本章で，初期化付き量子 Karatsuba 乗算アルゴリズムを提案する．第 4.1 節で Jang らの量子 Karatsuba 乗算アルゴリズム [JKL+23] を説明し，第 4.2 節で提案手法を説明し，それぞれのリソースを比較する．

4.1 Jang らの量子 Karatsuba 乗算

まずは，Karatsuba 乗算の概要を説明する．ここで，$\mathbb{F}_{2^n}$ 上の多項式は n ビットで表すことができることに注意されたい．多項式 $f, g \in \mathbb{F}_{2^n}$ を入力とし，$h = f \cdot g \in \mathbb{F}_{2^n}$ を出力する乗算を考える．素朴な乗算は計算量 $O(n^2)$ だが，Karatsuba 乗算は多項式を分割してサイズの小さな多項式の乗算とその結果の加算によって $h = f \cdot g \in \mathbb{F}_{2^n}$ を計算することで計算量を削減する．具体的には，$f, g \in \mathbb{F}_{2^n}$ を

$$f = f_1 x^{n/2} + f_0, \qquad g = g_1 x^{n/2} + g_0$$

と分割すると，f_1, f_0, g_1, g_0 は $n/2$ ビットで表すことのできる多項式となる．このとき，

$$\begin{aligned} & f \cdot g \\ &= (f_1 \cdot g_1) x^n \\ &\quad + ((f_0 + f_1) \cdot (g_0 + g_1) + f_0 \cdot g_0 + f_1 \cdot g_1)\, x^{n/2} \\ &\quad + f_0 \cdot g_0 \end{aligned} \tag{1}$$

が成り立ち，n ビット多項式の乗算 $f \cdot g$ は $n/2$ ビット多

項式の 3 つの乗算 $f_1 \cdot g_1, (f_0 + f_1) \cdot (g_0 + g_1), f_0 \cdot g_0$ とそれらの和で計算できる．加算は乗算より効率的なので，この分割でほぼ 3/4 倍のコストで計算できる．さらに，$n/2$ ビット多項式の乗算は 3 つの $n/4$ ビット多項式の乗算とそれらの和で計算できるため，1 ビット多項式の乗算になるまで分割し続けると，計算量は $O(n^{\log_2 3})$ となる．

図 3 で $n = 4$ のときの Jang らの量子 Karatsuba 乗算 [JKL+23] の概要を示す．白色以外の同色のビットは同一の量子ビットであることを意味し，枠で囲われたビットは新しく割り当てられた補助量子ビットであることを表す．Karatsuba 法により，2 つの 4 ビット多項式 f, g の 1 回の乗算を 6 つの 2 ビット多項式 $f_0, f_1, g_0, g_1, f_0 + f_1, g_0 + g_1$ に分割するが，f_0, f_1, g_0, g_1 への分割は説明の都合で実際は f, g から変化がなく，$f_0 + f_1, g_0 + g_1$ の計 4 ビットを新しい補助量子ビットに書き込む．これら 6 つの 2 ビット多項式による 3 回の乗算をさらに 18 個の 1 ビット多項式に分割し，計 6 ビットを新しい補助量子ビットに書き込む．ここで，9 回の 1 ビット乗算を実行し，結果を 9 つの新しい補助量子ビットに書き込み，計 19 ビットの補助量子ビットを用いる．その後，乗算結果を式 (1) に従い足し合わせることで 7 ビット多項式 $f \times g$ を得る．最後に剰余演算によって 4 ビット多項式 $f \times g \mod m(x)$ を得る．Jang らの乗算では，1 ビット乗算の前の補助量子ビットのみを初期化する．Jang らは 1 ビット乗算後にはそれ以前に割り当てた補助量子ビットを用いないことに着目し，乗算結果の足し合わせや剰余演算をこれらの補助量子ビットの初期化と並列に実行する．そのため，$n = 4$ では 1 ビット乗算の結果を書き込んだ 9 ビットのうち出力の 4 ビットを除く 5 ビットは初期化されずに乗算アルゴリズムが終了する．

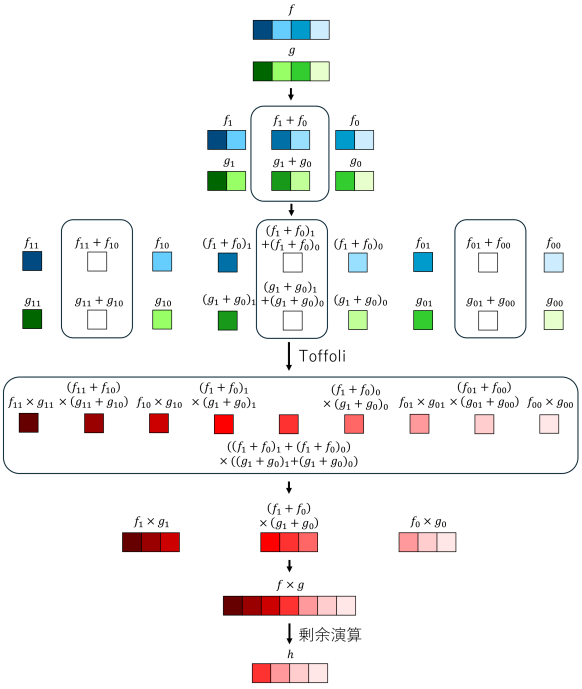

図 3 $n = 4$ での Jang らの量子 Karatsuba 乗算の概要

4.2 提案手法

提案乗算アルゴリズムと Jang らの量子 Karatsuba 乗算との違いは，$2n-1$ ビットの乗算結果 $f \times g$ を n ビット多項式 $f \times g \mod m(x)$ とする剰余演算である．図 4 は，図 3 で示した Jang らの量子 Karatsuba 乗算の後半をより詳細に記述しており，提案手法における同処理を図 5 に示す．Jang らの量子 Karatsuba 乗算の図 4 では，剰余演算の入出力で同じ量子ビットを用い，$2n-1$ ビットの乗算結果 $f \times g$ を同じ量子ビットにおいて n ビット多項式 $f \times g \mod m(x)$ に変更している．つまり，$a_1, a_2, c_1 + b_1 + a_1 + a_3, c_2 + b_2 + a_2$ の 4 ビットに $c_1 + c_3 + b_3 + a_3, c_2, c_3$ を足して剰余結果を求めそのまま出力する．一方で，提案手法の図 5 では，補助量子ビットを増やし，剰余演算の出力を入力とは違う量子ビットに割り当てる．つまり，CNOT ゲートを用いて $a_1, a_2, c_1 + b_1 + a_1 + a_3, c_2 + b_2 + a_2$ を新たな 4 ビットの補助量子ビットに書き込み，$c_1 + c_3 + b_3 + a_3, c_2, c_3$ を足して剰余を求め出力する．これが第 3 章における MUL_CAL である．このとき，図 4 と違い図 5 では元々の $a_1, a_2, c_1 + b_1 + a_1 + a_3, c_2 + b_2 + a_2$ が記録されているため，乗算結果の出力後に消去すべき補助量子ビットを出力結果を使わずに初期化可能である．これが第 3 章における MUL_INI である．

表 2 で，提案手法と既存の量子乗算アルゴリズム [vH20], [PWLK22], [JKL+23], [KKKH24] に必要な量子リソースを比較する．提案手法の深さは，MUL_CAL と MUL_INI を合わせて補助量子ビットを初期化したたものである．ただし，Jang らの量子 Karatsuba 乗算 [JKL+23] では初期化されない補助量子ビットがあることに注意されたい．そのため，乗算 1 回の量子ビット数の他に，入出力を除いて初期化されずに残る量子ビット数も併せて記載する．

Jang ら [JKL+23] 以外の既存研究は補助量子ビットを用いず量子ビット数が小さいが，深さが大きい．一方で，Jang ら [JKL+23] は量子ビット数と深さいずれも提案手法より小さい．そのため，一見すると提案手法は Jang

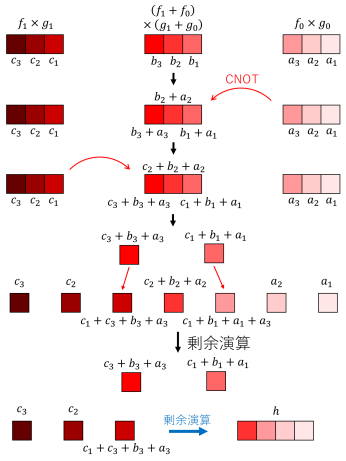

図 4 $n = 4$ での Jang らの量子 Karatsuba 乗算における剰余演算

表 2　乗算の量子リソースの比較

n	アルゴリズム	量子ビット数	初期化されない量子ビット数	深さ
163	[vH20]	489	0	17,906
	[PWLK22]	489	0	13,814
	[JKL+23]	13,161	4,224	52
	[KKKH24]	489	0	10,210
	提案手法	13,324	0	84
233	[vH20]	699	0	29,530
	[PWLK22]	699	0	19,294
	[JKL+23]	18,969	6,090	42
	[KKKH24]	699	0	16,383
	提案手法	19,202	0	82
283	[vH20]	849	0	41,548
	[PWLK22]	849	0	31,894
	[JKL+23]	30,819	9,990	56
	[KKKH24]	849	0	22,050
	提案手法	31,102	0	94
571	[vH20]	1,713	0	121,821
	[PWLK22]	1,713	0	95,863
	[JKL+23]	93,513	30,600	60
	[KKKH24]	1,713	0	61,771
	提案手法	94,084	0	104

ら [JKL+23] に劣っている印象を受ける．ただし，表 2 は乗算 1 回での比較であり，第 3 章の逆元計算で用いる際に提案手法が Jang ら [JKL+23] の乗算に劣るわけではない．つまり，逆元計算においては乗算と 2 冪乗算を交互に繰り返すことに注意する必要がある．Jang らの乗算 [JKL+23] の量子ビット数は提案手法より小さいが，初期化されない補助量子ビットがあるため，乗算回数に依存して初期化されない補助量子ビットが増えていく．そのため，逆元計算のように複数の乗算を行う際には提案手法の方が量子ビット数が小さい．さらに，第 3 章の逆元計算のように

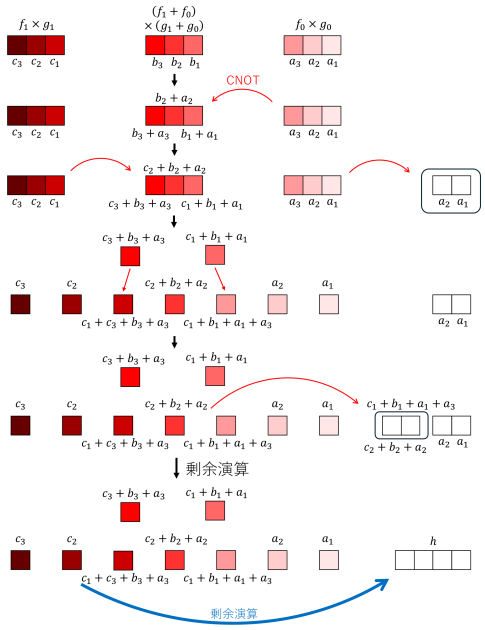

図 5　$n = 4$ での初期化付き量子 Karatsuba 乗算における剰余演算

2 冪乗算と乗算を交互に行う場合，例えば乗算の後に 2 冪乗算を行う場合，必ずしも MUL_CAL と MUL_INI の後に 2 冪乗算を行う必要はなく，MUL_CAL の後に MUL_INI と 2 冪乗算を並列に実行可能である．これによって，逆元計算においては表 2 の記載より小さな深さで提案乗算を実行可能である．例えば，Algorithm 2 の 3 行目と 4 行目，6 行目と 7 行目，8 行目と 9 行目，10 行目と 11 行目，12 行目と 13 行目，14 行目から 17 行目はそれぞれ同じ量子ビットを使わないため並列に実行可能である．つまり，$n = 4$ では全ての MUL_INI と SQR_α を並列に実行可能である．$n = 163, 233, 283, 571$ では，ほとんどの 2 冪乗算は乗算の初期化 MUL_INI より深さが大きくなり，逆元計算において MUL_INI の深さの影響が少ない．表 2 で示したように，提案手法の MUL_CAL と MUL_INI の深さの和は Jang らの乗算 [JKL+23] の深さより大きい．ただし，MUL_CAL のみなら $n = 163, 233, 283, 571$ における深さはそれぞれ $46, 43, 50, 55$ であり，$n = 233$ 以外では Jang らの乗算 [JKL+23] の深さより小さい．そのため，逆元計算における多くの乗算では Jang ら [JKL+23] より提案手法の方が深さが小さいと言える．まとめると，提案手法は乗算 1 回では Jang ら [JKL+23] に明らかに劣るが，逆元計算に特化した性質を持っていると言うことができ，結果的に表 1 で示した改良を可能にしている．

5. 並列逆元計算

本章では，楕円曲線加算アルゴリズムを提案する．楕円曲線加算は 2 回の逆元計算が支配的であり，例えば Algorithm 1 では 3 行目と 13 行目で逆元計算を行い，5 行目と 15 行目で逆元計算の際に利用した補助量子ビットを初期化している．提案楕円曲線加算アルゴリズムは，1 回目の逆元計算の初期化と 2 回目の逆元計算を並列に実行することで深さを削減する．提案楕円曲線加算アルゴリズムを Algorithm 3 に示す．ここで，11 行目の INV_CAL と 13 行目の INV_INI は並列に実行可能である．第 3 章で示したように，逆元計算は計算過程を記録するために徐々に使用する補助量子ビット数が増加していき，初期化では徐々に使用する補助量子ビット数が減少していく．よって，提案手法では逆元計算を並列に実行するために計算過程を記録する補助量子ビット数を 2 倍にする必要はなく，11 行目で消去した補助量子ビットに 13 行目の逆元計算の計算過程を記録していく．また第 4 章で述べたように，乗算初期化 MUL_INI の深さは 2 冪乗算の深さよりも小さい場合があり，逆元計算において乗算用の補助量子ビットが使用されていない時間がある．その間に逆元計算の初期化での乗算を実行することで，乗算用の補助量子ビット数を増やさずに逆元計算と初期化を並列に実行できる．そのため，並列逆元計算により量子ビット数をほとんど増やすことなく楕円曲線加算の深さを削減できる．Jang らの楕円曲線

Algorithm 3 並列逆元計算を用いた楕円曲線加算

Input: 量子状態: 制御ビット q，点 $P_1(x_1, y_1)$，初期化された補助量子ビット
古典状態: 楕円曲線のパラメータ a，固定点 $P_2(x_2, y_2)$

Output: $q = 1$ の場合 $P_1 + P_2 = P_3(x_3, y_3)$，$q = 0$ の場合 $P_1(x_1, y_1)$，初期化された補助量子ビット

1: $\mathrm{ADD}(x_1, x_2) = (x_1 + x_2, x_2)$
2: $\mathrm{CTRL\_ADD}(y_1, y_2, q) = (y_1 + q \cdot y_2, y_2, q)$
3: $\mathrm{INV\_CAL}(x_1 + x_2, 0, 0) = (x_1 + x_2, (x_1 + x_2)^{-1}, *)$
4: $\mathrm{MUL\_CAL}((x_1 + x_2)^{-1}, y_1 + q \cdot y_2, 0, 0)$
$= ((x_1 + x_2)^{-1}, y_1 + q \cdot y_2, \lambda(= (y_1 + q \cdot y_2)/(x_1 + x_2)), *)$
5: $\mathrm{MUL\_INI}((x_1{+}x_2)^{-1}, y_1{+}q{\cdot}y_2, *) = ((x_1{+}x_2)^{-1}, y_1{+}q{\cdot}y_2, 0)$
6: $\mathrm{MUL\_CAL}(x_1 + x_2, \lambda, y_1 + q \cdot y_2, 0) = (x_1 + x_2, \lambda, 0, *)$
7: $\mathrm{MUL\_INI}(x_1 + x_2, \lambda, *) = (x_1 + x_2, \lambda, 0)$
8: $\mathrm{SQR}_1(\lambda, 0) = (\lambda, \lambda^2)$
9: $\mathrm{ADD}(0, \lambda^2 + \lambda + a + x_2) = (\lambda^2 + \lambda + a + x_2, \lambda^2 + \lambda + a + x_2)$
10: $\mathrm{CTRL\_ADD}(x_1 + x_2, \lambda^2 + \lambda + a + x_2, q)$
$= (x(= x_1 + x_2 + q \cdot (\lambda^2 + \lambda + a + x_2)), \lambda^2 + \lambda + a + x_2, q)$
11: $\mathrm{INV\_CAL}(x, 0, 0) = (x, x^{-1}, *)$
12: $\mathrm{CTRL\_ADD}(x, \lambda^2{+}\lambda{+}a{+}x_2, q) = (x_1{+}x_2, \lambda^2{+}\lambda{+}a{+}x_2, q)$
13: $\mathrm{INV\_INI}(x_1 + x_2, (x_1 + x_2)^{-1}, *) = (x_1 + x_2, 0, 0)$
14: $\mathrm{CTRL\_ADD}(x_1{+}x_2, \lambda^2{+}\lambda{+}a{+}x_2, q) = (x, \lambda^2{+}\lambda{+}a{+}x_2, q)$
15: $\mathrm{ADD}(\lambda^2 + \lambda + a + x_2, \lambda^2 + \lambda + a + x_2) = (0, \lambda^2 + \lambda + a + x_2)$
16: $\mathrm{SQR}_1^{\dagger}(\lambda, \lambda^2) = (\lambda, 0)$
17: $\mathrm{MUL\_CAL}(x, \lambda, 0, 0) = (x, \lambda, x \cdot \lambda, *)$
18: $\mathrm{MUL\_INI}(x, \lambda, *) = (x, \lambda, 0)$
19: $\mathrm{MUL\_CAL}(x \cdot \lambda, x^{-1}, \lambda, 0) = (x \cdot \lambda, x^{-1}, 0, *)$
20: $\mathrm{MUL\_INI}(x \cdot \lambda, *) = (x \cdot \lambda, x^{-1}, 0)$
21: $\mathrm{INV\_INI}(x, x^{-1}, *) = (x, 0, 0)$
22: $\mathrm{ADD}(x, x_2) = (x_1 + q \cdot (x_1 + x_3), x_2)$
23: $\mathrm{CTRL\_ADD}(x \cdot \lambda, y_2 + x_1 + q \cdot (x_1 + x_3), q)$
$= (y_1 + q \cdot (y_1 + y_3), y_2 + x_1 + q \cdot (x_1 + x_3), q)$

加算 [JSB+25] は，逆元計算と逆元計算の初期化が計 4 回あるため，初期化を含めた逆元計算の深さの約 2 倍であった．一方で，提案楕円曲線加算は，1 回目の逆元計算の初期化と 2 回目の逆元計算を並列化することにより，初期化を含めた逆元計算の深さの約 3/2 倍で十分である．

6. Shor のアルゴリズムの比較

本章で, NIST の推奨パラメータ $n = 163, 233, 283, 571$ において，Shor のアルゴリズムを実装する量子回路のリソースを提案手法と既存研究 [BBvHL21], [PWLK22], [KH23], [TT23], [TT24], [JSB+25] で比較する．これらの既存研究と同様，量子フーリエ変換には semi-classical フーリエ変換 [GN96] を用いる．

表 3 に量子ビット数 M と深さ D とその積 MD の比較をまとめる．Banegas ら [BBvHL21] は，GCD と FLT の逆元計算に基づく手法を提案したため，それぞれのリソースをまとめている．Taguchi と Takayasu [TT23], [TT24] は，パラメータに応じて様々なトレードオフを満たす量子回路を設計したため，MD を最小化したリソースをまとめている．Jang ら [JSB+25] は，より大きな M と小さな D の量子回路も設計したが，MD がより小さいリソースをまとめている．提案手法は，既存研究で最先端

表 3 Shor のアルゴリズムの量子リソースの比較

n	アルゴリズム	量子ビット数 M	深さ D	MD
163	FLT [BBvHL21]	$1.96 \cdot 10^3$	$2.31 \cdot 10^8$	$4.53 \cdot 10^{11}$
	GCD [BBvHL21]	$1.16 \cdot 10^3$	$3.42 \cdot 10^8$	$3.97 \cdot 10^{11}$
	[PWLK22]	$3.10 \cdot 10^3$	$2.04 \cdot 10^8$	$6.32 \cdot 10^{11}$
	[KH23]	$1.02 \cdot 10^3$	$3.19 \cdot 10^8$	$3.25 \cdot 10^{11}$
	[TT23]	$1.96 \cdot 10^3$	$1.82 \cdot 10^8$	$3.56 \cdot 10^{11}$
	[TT24]	$1.14 \cdot 10^3$	$2.31 \cdot 10^8$	$2.63 \cdot 10^{11}$
	[JSB+25]	$5.31 \cdot 10^4$	$1.85 \cdot 10^6$	$9.82 \cdot 10^{10}$
	提案手法	$1.56 \cdot 10^4$	$1.30 \cdot 10^6$	$2.03 \cdot 10^{10}$
233	FLT [BBvHL21]	$3.03 \cdot 10^3$	$4.46 \cdot 10^8$	$1.35 \cdot 10^{12}$
	GCD [BBvHL21]	$1.65 \cdot 10^3$	$9.45 \cdot 10^8$	$1.56 \cdot 10^{12}$
	[PWLK22]	$4.66 \cdot 10^3$	$4.23 \cdot 10^8$	$1.97 \cdot 10^{12}$
	[KH23]	$1.44 \cdot 10^3$	$8.98 \cdot 10^8$	$1.29 \cdot 10^{12}$
	[TT23]	$3.03 \cdot 10^3$	$4.21 \cdot 10^8$	$1.28 \cdot 10^{12}$
	[TT24]	$1.63 \cdot 10^3$	$5.30 \cdot 10^8$	$8.64 \cdot 10^{11}$
	[JSB+25]	$8.29 \cdot 10^4$	$3.30 \cdot 10^6$	$2.74 \cdot 10^{11}$
	提案手法	$2.27 \cdot 10^4$	$2.06 \cdot 10^6$	$4.68 \cdot 10^{10}$
283	FLT [BBvHL21]	$3.96 \cdot 10^3$	$1.15 \cdot 10^9$	$4.55 \cdot 10^{12}$
	GCD [BBvHL21]	$2.00 \cdot 10^3$	$1.67 \cdot 10^9$	$3.34 \cdot 10^{12}$
	[PWLK22]	$6.23 \cdot 10^3$	$9.77 \cdot 10^8$	$6.09 \cdot 10^{12}$
	[KH23]	$1.74 \cdot 10^3$	$1.59 \cdot 10^9$	$2.77 \cdot 10^{12}$
	[TT23]	$3.96 \cdot 10^3$	$9.40 \cdot 10^8$	$3.72 \cdot 10^{12}$
	[TT24]	$1.98 \cdot 10^3$	$1.16 \cdot 10^9$	$2.30 \cdot 10^{12}$
	[JSB+25]	$1.45 \cdot 10^5$	$6.20 \cdot 10^6$	$8.99 \cdot 10^{11}$
	提案手法	$3.59 \cdot 10^4$	$3.40 \cdot 10^6$	$1.22 \cdot 10^{11}$
571	FLT [BBvHL21]	$9.14 \cdot 10^3$	$1.02 \cdot 10^{10}$	$9.32 \cdot 10^{13}$
	GCD [BBvHL21]	$4.02 \cdot 10^3$	$1.30 \cdot 10^{10}$	$5.23 \cdot 10^{13}$
	[PWLK22]	$1.43 \cdot 10^4$	$8.28 \cdot 10^9$	$1.18 \cdot 10^{14}$
	[KH23]	$3.47 \cdot 10^3$	$1.26 \cdot 10^{10}$	$4.37 \cdot 10^{13}$
	[TT23]	$8.57 \cdot 10^3$	$6.72 \cdot 10^9$	$5.76 \cdot 10^{13}$
	[TT24]	$4.00 \cdot 10^3$	$9.73 \cdot 10^9$	$3.89 \cdot 10^{13}$
	[JSB+25]	$5.00 \cdot 10^5$	$2.76 \cdot 10^7$	$1.38 \cdot 10^{13}$
	提案手法	$1.04 \cdot 10^5$	$1.06 \cdot 10^7$	$1.10 \cdot 10^{12}$

の Jang ら [JSB+25] より M と D の両方を削減しており，純粋に改良したと言える．例として $n = 571$ の場合，提案手法は Jang らの結果から，量子ビット数 M は 79.2%，深さ D は 61.6%，MD を 92.0% 削減している．他の既存研究 [BBvHL21], [PWLK22], [KH23], [TT23], [TT24] は，提案手法より M が小さく D が大きい．ただし，Jang ら [JSB+25] が主張したように，これらの研究は MD が Jang らより大きく，Jang ら [JSB+25] の方がより良いトレードオフを達成していると言える．Jang ら [JSB+25] を除いた既存研究の中では Taguchi と Takayasu [TT24] が最も小さな MD を達成しており，Jang ら [JSB+25] は他の既存研究より MD を削減したことを強く主張しているが，全ての $n = 163, 233, 283, 571$ で Taguchi と Takayasu [TT24] の MD を最大 68.3% 程度しか削減できていないことに注意されたい．つまり，提案手法が Jang ら [JSB+25] の MD を改良したギャップは，Jang ら [JSB+25] が他の既存研究 [BBvHL21], [PWLK22], [KH23], [TT23], [TT24] を改

良したギャップよりも大きい．

7. 結論

本論文では，バイナリ楕円曲線上の離散対数問題を解く Shor のアルゴリズムの量子回路を設計し，既存研究より量子リソースを大幅に削減した．まず我々は，第 3 章で Jang ら [JSB+25] と Taguchi と Takayasu [TT23], [TT24] を組み合わせた量子逆元計算アルゴリズムを提案した．さらに，第 4 章で提案逆元計算アルゴリズムで用いる初期化付き量子 Karatsuba 乗算を提案した．初期化付き量子 Karatsuba 乗算は Jang らの量子 Karatsuba 乗算 [JKL+23] より乗算 1 回あたりのリソースは大きいが，複数回の乗算を繰り返す量子逆元計算においては提案手法の方が効果的である．結果として Jang らの量子逆元計算 [JSB+25] より量子ビット数 M と深さ D の両方を改良した．また，既存の楕円曲線加算とは一線を画す並列逆元計算を用いた量子楕円曲線加算アルゴリズムを提案し，さらに深さを削減した．これによって，提案手法に基づく Shor のアルゴリズムは，$n = 571$ のとき Jang ら [JSB+25] より MD を 92.0% 削減している．今後の課題として，Jang ら [JSB+25] 以前の既存研究のように量子ビット数が数千程度の場合の改良と，素体上の楕円曲線における Shor のアルゴリズムの改良が挙げられる．

謝辞 本研究の一部は JST CREST Grant Number JPMJCR2113 と JSPS KAKENHI Grant Number 23K21667 の助成を受けたものです．

参考文献

[AMMR13] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Rötteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[BBvHL21] Gustavo Banegas, Daniel J. Bernstein, Iggy van Hoof, and Tanja Lange. Concrete quantum cryptanalysis of binary elliptic curves. *IACR TCHES*, 2021(1):451–472, 2021.

[GE21] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, April 2021.

[GN96] Robert B. Griffiths and Chi-Sheng Niu. Semiclassical fourier transform for quantum computation. *Physical Review Letters*, 76(17):3228–3231, 1996.

[HJN+20] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. Improved quantum circuits for elliptic curve discrete logarithms. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 425–444. Springer, Cham, 2020.

[JKL+23] Kyungbae Jang, Wonwoong Kim, Sejin Lim, Yeajun Kang, Yujin Yang, and Hwajeong Seo. Optimized implementation of quantum binary field multiplication with toffoli depth one. In *Proceedings of the 10th International Workshop on Information Security Applications (WISA 2022)*, volume 13720 of *Lecture Notes in Computer Science*, pages 251–264. Springer, Cham, 2023.

[JSB+25] Kyungbae Jang, Vikas Srivastava, Anubhab Baksi, Santanu Sarkar, and Hwajeong Seo. New quantum cryptanalysis of binary elliptic curves (extended version). In *Proceedings of the 27th Cryptographic Hardware and Embedded Systems (CHES 2025)*, Lecture Notes in Computer Science. Springer, Cham, 2025.

[KH23] Hyeonhak Kim and Seokhie Hong. New space-efficient quantum algorithm for binary elliptic curves using the optimized division algorithm. *Quantum Inf. Process.*, 22(6):237, 2023.

[KKKH24] Sunyeop Kim, Insung Kim, Seonggyeom Kim, and Seokhie Hong. Toffoli gate count optimized space-efficient quantum circuit for binary field multiplication. *Quantum Information Processing*, 23(10):330, 2024.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer, Berlin, Heidelberg, August 1986.

[PWLK22] Dedy Septono Catur Putranto, Rini Wisnu Wardhani, Harashta Tatimma Larasati, and Howon Kim. Another concrete quantum cryptanalysis of binary elliptic curves. Cryptology ePrint Archive, Report 2022/501, 2022.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin E. Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 241–270. Springer, Cham, December 2017.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[TT23] Ren Taguchi and Atsushi Takayasu. Concrete quantum cryptanalysis of binary elliptic curves via addition chain. In Mike Rosulek, editor, *CT-RSA 2023*, volume 13871 of *LNCS*, pages 57–83. Springer, Cham, April 2023.

[TT24] Ren Taguchi and Atsushi Takayasu. On the untapped potential of the quantum FLT-based inversion. In Christina Pöpper and Lejla Batina, editors, *ACNS 2024, Part II*, volume 14584 of *LNCS*, pages 79–100. Springer, Cham, March 2024.

[vH20] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic toffoli gate count. *Quantum Information Processing*, 20(9&10):721–735, 2020.