Computer Security Symposium 2022
24 - 27 October 2022

量子FLT逆元計算アルゴリズムの改良

田口 廉[1,a)]　高安 敦[1]

概要：楕円曲線上の離散対数問題（elliptic curve discrete logarithm problem, ECDLP）は Shor の量子アルゴリズムによって多項式時間で解けることが知られており，実装に係るリソースの見積りと削減は重要な研究課題である．本研究はバイナリ楕円曲線上の ECDLP を対象とし，特に最も支配的な $\mathbb{F}_{2^n}^*$ 上の逆元計算を行う方法の一つである量子 FLT 逆元計算に注目する．Banegas らと Putranto らはそれぞれ量子ビット数と深さの最適化を目指した量子 FLT 逆元計算アルゴリズムを提案した．本論文で，我々は 2 つの改良アルゴリズムを提案し，NIST の推奨する次数 n において既存研究とリソース数を比較する．1 つ目の提案アルゴリズムは，全ての n において Putranto らのアルゴリズムの量子ゲート数・深さを犠牲にすることなく量子ビット数を削減しており，特に $n = 409, 571$ のときには量子ゲート数をも削減する．2 つ目の提案アルゴリズムは，全ての n において Banegas らのアルゴリズムの量子ビット数・量子ゲート数を犠牲にすることなく深さを削減し，特に $n = 409, 571$ のときには量子ビット数・量子ゲート数をも削減する．提案アルゴリズムは，既存研究がいずれも Itoh-Tsujii 古典 FLT 逆元計算を基にしていたのに対して，より一般的な加法連鎖列による FLT 逆元計算を基にしている．そして，古典計算では計算コストに影響を与えなかった加法連鎖列の性質が量子計算ではリソース数を変化させることを明らかにし，その事実に基づいて最適化を行うことで前述の改良を達成している．

キーワード：ECDLP，量子アルゴリズム，FLT 逆元計算，リソース評価，加法連鎖

Improved Quantum FLT-based Inversion Algorithm

Ren Taguchi[1,a)]　Atsushi Takayasu[1]

Abstract: Shor's quantum algorithm solves the elliptic elliptic curve discrete logarithm problem (ECDLP) in polynomial time. FLT-based inversion is a method for computing an inverse over $\mathbb{F}_{2^n}$, where the computation is a dominant part of Shor's algorithm. Banegas et al.'s algorithm and Putranto et al.'s algorithm are designed to optimize the number of qubits and the number of quantum gates, depth of circuits, respectively. In this paper, we propose two improved quantum FLT-based inversion algorithms. The first algorithm reduces the number of qubits of Putranto et al.'s algorithm, while the second algorithm reduces the depth of Banegas et al.'s algorithm. In particular, when $n = 409, 571$, the first and second algorithms also reduce the number of qubits and the gates.

Keywords: ECDLP, quantum algorithm, FLT-based inversion, resource estimation,addition chain

1. はじめに

1.1 背景

RSA 暗号・楕円曲線暗号は，実用的に最も広く利用されている公開鍵暗号方式である．楕円曲線暗号においては素体 $\mathbb{F}_q$ 上の楕円曲線と $\mathbb{F}_{2^n}$ 上のバイナリ楕円曲線を利用することが NIST によって推奨されており [3]，バイナリ楕円曲線では次数 $n = 163, 233, 283, 409, 571$ がパラメータとして設定されている．これまでの研究で，素因数分解と楕円曲線上の離散対数問題（elliptic curve discrete logarithm problem, ECDLP）を多項式時間で解く（古典）アルゴリ

[1] 東京大学情報理工学系研究科
Graduate School of Information Science and Technology,The University of Tokyo
[a)] rtaguchi-495@g.ecc.u-tokyo.ac.jp

ズムは知られておらず，そのため前述の方式は安全であると考えられている．

1994 年に，Shor は素因数分解と ECDLP を多項式時間で解く量子アルゴリズム [13] を提案した．そのため，量子攻撃に対しても耐性のある耐量子計算機暗号への移行が近年活発に議論されている．ただし，RSA 暗号・楕円曲線暗号が実用的に破られる時期については意見が分かれる．事実，量子コンピュータを物理的に実現するためにはまだ多くの技術的課題が残されており，暗号で用いるような大きなパラメータの問題を解く規模を実現するのは容易ではない．これまで，Shor のアルゴリズム実装に関する数多くの研究が発表されているが，15 や 21 の素因数分解が主で ECDLP を解く実装はまだ報告されていない．そのため，耐量子計算機暗号への移行時期を正しく見積もるために，Shor のアルゴリズムを実行するための量子回路のリソース数の評価や削減を目指す研究は重要な研究課題である．特に，リソースとして主に重要と考えられているのは，量子ビット数・量子ゲート数・回路の深さであり，量子ゲート数の中でも特に Toffoli ゲート数が重要視されている．

ECDLP を解くためのリソース評価の研究は，Roetteler ら [12] によって素体 $\mathbb{F}_q$ 上の楕円曲線における ECDLP がそれまでの予想よりずっと効率的に解けることが指摘され，大きな注目を集めた．本論文は，Banegas ら [2] や Putranto ら [10] によるバイナリ楕円曲線における ECDLP に注目する．Shor のアルゴリズム実行の際に支配的となるのは $\mathbb{F}_{2^n}$ 上の乗法逆元の計算であり，有力な計算法として GCD 逆元計算と FLT 逆元計算[*1]がある．Banegas らは，量子ビット数の最適化を目指し，この二つの計算法の量子アルゴリズムを提案した．これらを比較すると，量子 GCD 逆元計算アルゴリズムはより量子ビット数が少ないという点で Banegas らの目指していたものとなったが，量子 FLT 逆元計算アルゴリズムは Toffoli ゲート数がより少なく深さがより小さい．Putranto らは深さの最適化を目指し，Banegas らのものを簡略化した量子 FLT 逆元計算アルゴリズムを提案した．より正確には，Putranto らのアルゴリズムは Banegas らのアルゴリズムにおいて，ガベージと呼ばれる計算過程で生じる不要な量子ビットを削除する計算を省略したものとなっており，量子ビット数は増えるが Toffoli ゲート数を変えずに深さを小さくすることに成功している．

1.2 成果

我々は，既存の量子 FLT 逆元計算アルゴリズムで必要となる量子ビット数・Toffoli ゲート数と次数 n の関係を詳細に解析することで改良アルゴリズムを提案する．Banegas らのアルゴリズム [2] と Putranto らのアルゴリズム [10] の量子ビット数・Toffoli ゲート数は，次数 n の値と $n-1$ を 2 進数表記したときのハミング重みに依存している．FLT 逆元計算は，与えられた $\mathbb{F}_{2^n}$ の元に対して n によって定められた乗算を繰り返すことで逆元を計算する方法だが，Banegas らと Putranto らはいずれも Itoh と Tsujii の古典 FLT 逆元計算 [9] で提案された計算手順を利用している．我々は，この計算手順を加法連鎖の観点からより一般的に捉え直す．ここで，Itoh と Tsujii による計算手順は，n に対して特定の加法連鎖列を割り当てることで決定されるものと捉えられることに注意されたい．古典アルゴリズムの文脈では，我々と同様に加法連鎖による FLT 逆元計算アルゴリズムが多くの論文 [1], [4], [5], [6], [11] で議論されてきたが，計算コストに影響するのが加法連鎖の長さのみであることから大きな注目を集めてこなかった．ところが興味深いことに，量子アルゴリズムの文脈では，加法連鎖列の長さのみならず各要素の計算方法など様々な要因が量子ビット数・Toffoli ゲート数に影響することを示す．そして，任意の加法連鎖列に対して量子 FLT 逆元計算アルゴリズムが存在することを示し，リソース数の評価指針を与える．より正確には，我々は 2 つの量子アルゴリズムを示す．1 つ目の基本アルゴリズムは，Itoh と Tsujii の加法連鎖列に従っていた Putranto らのアルゴリズムを一般の加法連鎖に対応させたものであり，2 つ目の拡張アルゴリズムは，Banegas らと同様に我々の基本アルゴリズムにガベージ処理を加えて量子ビット数を削減させるものである．

上記の結果に基づき，NIST の推奨パラメータ $n = 163, 233, 283, 409, 571$ に対して，Toffoli ゲート数・量子ビット数・深さの順に最適化を目指し加法連鎖列を探索することで量子 FLT 逆元計算アルゴリズムを得る．前述の通り，提案アルゴリズムは Itoh と Tsujii の加法連鎖列を一般の加法連鎖列に置き換えるものであり，より良い加法連鎖列が存在しない場合には既存アルゴリズムを改良することはできない．ところが興味深いことに，提案手法は全てのパラメータに対して改良に成功する．まず，$n = 409, 571$ のときは Itoh と Tsujii の加法連鎖列より短い加法連鎖列を見つけることができ，提案アルゴリズムは量子ビット数・Toffoli ゲート数・深さ全てにおいて Putranto らのアルゴリズムと Banegas らのアルゴリズムを改良する．$n = 163, 233, 283$ のときには Itoh と Tsujii と同じ長さの加法連鎖列しか見つけることができず，既存アルゴリズムと Toffoli ゲート数は変わらず，また既存研究において最適化を目指した Putranto らのアルゴリズムの深さと Banegas らのアルゴリズムの量子ビット数を改良することはできない．ただし，我々は長さ以外のリソースに影響する加法連鎖列の情報を得ることで，同じ長さでもより良い加法連鎖列を見つけることができる．これによって，Putranto らのアルゴリズムの量子ビット数と Banegas らのアルゴリズムの深さを改良する．

[*1] FLT はフェルマーの小定理（Falmat's little theorem）の略である．

1.3 本論文の構成

本論文では，第 2 章でバイナリ楕円曲線上の離散対数問題及び $\mathbb{F}_{2^n}$ 上の量子計算を説明する．第 3 章で古典・量子双方の FLT 逆元計算アルゴリズムを示す．第 4 章では我々の提案する量子 FLT 逆元計算アルゴリズムを示し，第 5 章で提案手法と既存手法との比較を行う．

2. 準備

本章では，第 2.1 節においてバイナリ楕円曲線とその上での離散対数問題の定義を与える．第 2.2 節で $\mathbb{F}_{2^n}$ 上の量子計算に関する基本的な内容を説明する．

2.1 バイナリ楕円曲線上の離散対数問題

n を自然数とするとき，バイナリ楕円曲線は，$a \in \mathbb{F}_{2^n}, b \in \mathbb{F}_{2^n}^*$ を用いて $y^2 + xy = x^3 + ax^2 + b$ と表される．一般に楕円曲線上の有理点からなる集合は可換群をなすことが知られている．バイナリ楕円曲線では，$P = (x_1, y_1), Q = (x_2, y_2)$ の加算結果 $R = (x_3, y_3)$ は，$Q \neq \pm P$ のとき

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, \quad y_3 = (x_2 + x_3)\lambda + x_3 + y_2$$

と書ける．なお，$\lambda = (y_1 + y_2)/(x_1 + x_2)$ である．すなわち，$\mathbb{F}_{2^n}$ 上の四則演算によりバイナリ楕円曲線上の加算は計算できる．また，P を上の加算の意味で k 倍した点を $[k]P$ と書く．このとき，バイナリ楕円曲線上の離散対数問題は，有理点 $P, [k]P$ の座標から k を求める問題である．

2.2 $\mathbb{F}_{2^n}$ 上の量子計算

古典計算では，0 または 1 を表すビットを利用して計算を行う．それに対して量子計算では，$|0\rangle$ と $|1\rangle$，その重ね合わせ状態をとる量子ビットを用いて計算を行う．そこでまず，次数 n のバイナリ楕円曲線上の加算を行うための $\mathbb{F}_{2^n}$ 上の演算を量子的に行うため，$\mathbb{F}_{2^n}$ の元を量子ビットを用いて表現する．表現方法は複数あるが，ここでは n 次の $\mathbb{F}_2[x]$ の既約多項式 $m(x)$ について $\mathbb{F}_{2^n} \simeq \mathbb{F}_2[x]/(m(x))$ となることを利用する．このとき，$\mathbb{F}_2[x]/(m(x))$ の元は各係数が 0 か 1 である高々 $n-1$ 次の多項式として書かれるため，n 個の量子ビットを用意してそのそれぞれの状態 $|0\rangle, |1\rangle$ を多項式の係数と対応させることで自然に $\mathbb{F}_{2^n}$ の元を表現できる．以降では $\mathbb{F}_{2^n}$ の元を表す n 量子ビットのことをレジスタと呼ぶ．

量子計算では，古典計算での NOT や AND・OR などに対応する量子ゲートと呼ばれる計算ゲートを用いる．本論文では CNOT ゲート，Toffoli ゲート，swap ゲートのみを扱う．なお，swap ゲートは CNOT ゲート 3 つで構成でき，Toffoli ゲートは CNOT ゲートや swap ゲートと比べて計算コストが大きいことに注意されたい．

次に，Banegas ら [2] の $\mathbb{F}_{2^n}$ の元の加算・二倍算を行う量子アルゴリズム，Hoof [8] の $\mathbb{F}_{2^n}^*$ の元の乗算を行う量子アルゴリズムを概説する．$f, g, h \in \mathbb{F}_{2^n}$ とする．加算を行う量子アルゴリズム ADD は $\mathrm{ADD}(f, g) = (f+g, f)$ なる入出力を持つ．なお本論文では，f を別レジスタにもう一つ格納するための，$g = 0$ の ADD 演算のみ用いる．すなわち，f と別に $|0\rangle$ で初期化されたレジスタが必要な演算であり，新たに n 量子ビットを要する．また，二乗算を行う量子アルゴリズム SQUARE は $\mathrm{SQUARE}(f) = f^2$ なる入出力を持つ．さらに，量子ゲートの可逆性から $\mathrm{SQUARE}^{-1}(f^2) = f$ なる入出力を持つ SQUARE^{-1} も構成できる．ここで ADD, SQAURE・SQUARE^{-1} は全て CNOT ゲートだけで構成でき，その必要個数は ADD で n 個，SQUARE・SQUARE^{-1} で最大 $n^2 - n$ 個である．乗算を行うアルゴリズム MODMULT_Imp は $\mathrm{MODMULT\_Imp}(h, f, g) = (h+fg, f, g)$ なる入出力を持ち，この計算には CNOT ゲートだけでなく Toffoli ゲートを利用する．本論文では，$h = 0$ である場合の MODMULT_Imp しか用いず，その際 $|0\rangle$ で初期化されたレジスタ h が必要になるから，MODMULT_Imp は新たに n 量子ビットを必要とする計算である．

3. FLT 逆元計算

本章では，第 3.1 節に Itoh と Tsujii [9] による古典 FLT 逆元計算を示し，その後第 3.2 節でより簡潔な Putranto らの量子 FLT 逆元計算アルゴリズム [10] を，第 3.3 節でガベージ処理を伴う Banegas らの量子 FLT 逆元計算アルゴリズム [2] を示す．

3.1 FLT 逆元計算

FLT 逆元計算の原理は，$f \in \mathbb{F}_{2^n}^*$ について，フェルマーの小定理の一般化であるオイラーの定理から導かれる関係式 $f^{2^n-1} = 1$ から $f^{-1} = f^{2^n-2}$ のように逆元計算ができることである．そこで，以下に示す関係式

$$\left(f^{2^{2^{k-1}}-1}\right)^{2^{2^{k-1}}} \times f^{2^{2^{k-1}}-1} = f^{2^{2^k}-1}, \tag{1}$$

$$\left(f^{2^\alpha-1}\right)^{2^\beta} \times f^{2^\beta-1} = f^{2^{\alpha+\beta}-1}, \tag{2}$$

を利用して f^{2^n-2} を計算することを考える．まず $n-1$ の二進表記を考え，そのハミング重みを t とする．最下位ビットを 0 ビット目と数えることにすれば，$n-1$ の二進表記で 1 の現れる位置を k_1 ビット目，k_2 ビット目，$\cdots$，k_t ビット目と書くことができる．このとき，$k_1 > k_2 > \cdots k_t \geq 0$ であることに注意されたい．今定義した記号を用いれば，$n-1 = \sum_{s=1}^{t} 2^{k_s}$ と書ける．この上で，関係式 (1) で $k=1$ としたものを使えば f から $f^{2^{2^1}-1}$ が計算でき，同じ関係式で $k=2$ とすれば今の結果から $f^{2^{2^2}-1}$ が計算できる．これを繰り返し行うことにより，$f^{2^{2^{k_1}}-1}$ まで計算ができる．このステップの途中において，$1 \leq s \leq t$ なる任意の自然数 s について $f^{2^{2^{k_s}}-1}$ が計算できていることに注意する．次に (2) の関係式と前のステップで得られている結果を利用

して順次 $f^{2^{2^{k_1}+2^{k_2}}-1}, \cdots, f^{2^{2^{k_1}+\cdots+2^{k_t}}-1}$ を計算する．ここで最後に得られたもの指数部分に現れる $2^{k_1}+\cdots+2^{k_t}$ は $n-1$ と等しいから，最後の式は $f^{2^{n-1}-1}$ と等しいので，$\left(f^{2^{n-1}-1}\right)^2 = f^{2^n-2}$ により f^{2^n-2} を計算できる．

アルゴリズム 1 Putranto らの量子 FLT 逆元計算アルゴリズム

入力: n 次既約多項式 $m(x) \in \mathbb{F}_2[x]$，$n-1$ の二進表記による $k_1, \cdots, k_t$，$k_p = 2k_1 + t - 1$, 多項式 $f_0 = f \in \mathbb{F}_{2^n}^*$，全てのビットが $|0\rangle$ で初期化された多項式 $f_1, \cdots, f_{k_p}$
出力: $f_{k_p} = f^{-1}$
1: **for** $i = 1, \ldots, k_1$ **do**
2: ADD($f_{2(i-1)+1}, f_{2(i-1)}$)
3: **for** $j = 1, \ldots, j = 2^{i-1}$ **do**
4: SQUARE($f_{2(i-1)+1}$)
5: MODMULT_Imp($f_{2(i-1)+2}, f_{2(i-1)+1}, f_{2(i-1)}$)
6: **for** $i = 1, \ldots, t-1$ **do**
7: **for** $j = 1, \ldots, 2^{k_{i+1}}$ **do**
8: SQUARE(f_{2k_1+i-1})
9: MODMULT_Imp($f_{2k_1+i}, f_{2k_{i+1}}, f_{2k_1+i-1}$)
10: **if** $t = 1$ **then**
11: swap(f_{k_1}, f_{k_p})
12: SQUARE(f_{k_p})

3.2 Putranto らの量子 FLT 逆元計算

Putranto らの提案した量子 FLT 逆元計算アルゴリズム [10] を**アルゴリズム 1** に示す．これは Itoh と Tsujii の FLT 逆元計算 [9] をそのまま量子アルゴリズムで記述したものであるが，現在知られる $\mathbb{F}_{2^n}$ における量子逆元計算アルゴリズムの中では最も Toffoli ゲート数・深さが少ない．特に，Toffoli ゲートを使う MODMULT_Imp の使用を極力減らし，代わりに Toffoli ゲートを使わない SQUARE を多用することが Toffoli ゲートを削減している．ここで，アルゴリズム 1 の 2 つの大ループを詳細に説明する．

1 行目から 5 行目までのループ：ここでは式 (1) による計算を行っており，$1 \le i \le k_1$ について i 番目のループでは $f^{2^{i-1}-1}$ を入力として f^{2^i-1} を出力する．具体的には，$1 \le i \le k_1$ について，ADD を用いて $f^{2^{2^{i-1}}-1}$ を新たなレジスタに格納する．次いで新たなレジスタにある $f^{2^{2^{i-1}}-1}$ に SQUARE を 2^{i-1} 回適用して $(f^{2^{2^{i-1}}-1})^{2^{2^{i-1}}}$ を得る．この結果と $f^{2^{2^{i-1}}-1}$ を MODMULT_Imp により掛け合わせることで，式 (1) の左辺の形を得るので $f^{2^{2^i}-1}$ が新たなレジスタに格納される．従ってこのループでは乗算を k_1 回要し，新たに $2k_1$ レジスタ，すなわち $2k_1n$ 量子ビットを要する．

6 行目から 9 行目までのループ：ここでは式 (2) による計算を行い，$f^{2^{2^{k_1}}-1}, \ldots, f^{2^{2^{k_t}}-1}$ から $f^{2^{2^{k_1}+\cdots+2^{k_t}}-1}$ を得る．まずは $f^{2^{2^{k_1}}-1}$ に SQUARE を 2^{k_2} 回適用して $(f^{2^{2^{k_1}}-1})^{2^{2^{k_2}}}$ を得る．これと $f^{2^{2^{k_2}}-1}$ を MODMULT_Imp により掛け合わせることで式 (2) で $\alpha = 2^{k_1}, \beta = 2^{k_2}$ とした形を得るので $f^{2^{2^{k_1}+2^{k_2}}-1}$ が新たなレジスタに格納される．得られた結果と $f^{2^{2^{k_3}}}$ について同様のことを行い，これを $f^{2^{2^{k_t}}-1}$ を使うまで続けることで $f^{2^{2^{k_1}+\cdots+2^{k_t}}-1}$ を得る．従ってこのループでは乗算を $t-1$ 回要し，新たに $t-1$ レジスタ，すなわち $(t-1)n$ 量子ビットを要する．

以上をまとめると，アルゴリズム 1 全体では乗算を k_1+t-1 回，また新たに $(2k_1+t-1)n = k_pn$ 量子ビットを要する．

3.3 Banegas らの量子 FLT 逆元計算

次に Banegas らの提案した量子 FLT 逆元計算アルゴリズム [2] を**アルゴリズム 2** に示す．このアルゴリズムは，アルゴリズム 1 の 1 行目から 5 行目のループで発生するガベージを SQUARE^{-1} を用いて消去することで，量子ビット数を削減する．この際，ガベージ消去に伴って深さ・CNOT ゲート数はアルゴリズム 1 と比べ増加する．まず，1 行目から 8 行目のループにおいて，ADD 演算に必要な 0 に初期化されたレジスタを使いまわしていることで，アルゴリズム 1 の 1 行目から 5 行目のループと比べて k_1-1 レジスタ，すなわち $(k_1-1)n$ 量子ビットを削減する．さらに，最終結果である f^{2^n-2} も前述の ADD 演算に使われたレジスタに格納するため，アルゴリズム 1 と比べて 1 レジスタ，つまり n 量子ビットを削減している．以上から，アルゴリズム 2 全体では乗算を k_1+t-1 回，また新たに必要な量子ビットはアルゴリズム 1 と比べて k_1n ビット減少し，$(k_1+t-1) = k_bn$ ビットとなる．なお，依然として GCD 逆元計算 [2] よりは多くの量子ビットを使用する．

4. 提案手法

本章では，まず第 4.1 節において加法連鎖に関する基本的事項を説明し，第 4.2 節で任意の加法連鎖列を入力にもつ量子 FLT 逆元計算アルゴリズムである基本アルゴリズムを示す．続く第 4.3 節ではアルゴリズム 2 で使われているガベージ処理を提案手法に対して適用し，量子ビット数と深さのトレードオフを達成する拡張アルゴリズムを示す．

4.1 加法連鎖

加法連鎖とは，1 から始めて加法を繰り返すことで目的の数を得る数列であり，冪乗計算の効率化の文脈でよく用いられる．具体的には，自然数 N, ℓ に対して，数列 $p_0 = 1, p_1, p_2, \ldots, p_\ell = N$ が，条件

- $\ell = 1, 2, \ldots, \ell$ において $0 \le i, j < s$ なる i, j があって $p_s = p_i + p_j$ が成り立つ

を満たすとき，N の長さ ℓ の加法連鎖列であるという．一般に，長さ ℓ が小さいほど効率的なので，加法連鎖列には同じ数は出てこず，$i \ne j$ ならば $p_i \ne p_j$ が成り立つとする．

アルゴリズム 2 Banegas らの量子 FLT 逆元計算アルゴリズム

入力: n 次既約多項式 $m(x) \in \mathbb{F}_2[x]$, $n-1$ の二進表記による $k_1, \cdots, k_t$, $k_b = \max(k_1+t-1, k_1+1)$, 多項式 $f_0 = f \in \mathbb{F}_{2^n}^*$, 全てのビットが $|0\rangle$ で初期化された多項式 $f_1, \cdots, f_{k_b}$

出力: $f_{k_b} = f^{-1}$

```
for i = 1, ..., k_1 do
    ADD(f_{k_b}, f_{i-1})
    for j = 1, ..., 2^{i-1} do
        SQUARE(f_{k_b})
    MODMULT_Imp(f_i, f_{k_b}, f_{i-1})
    for j = 1, ..., 2^{i-1} do
        SQUARE^{-1}(f_{k_b})
    ADD(f_{k_b}, f_{i-1})
for i = 1, ..., t-1 do
    for j = 1, ..., 2^{k_{i+1}} do
        SQUARE(f_{k_1+i-1})
    MODMULT_Imp(f_{k_1+i}, f_{k_{i+1}}, f_{k_1+i-1})
if t = 1 then
    swap(f_{k_1}, f_{k_b})
SQUARE(f_{k_b})
```

4.2 基本アルゴリズム

第3節の Itoh と Tsujii の FLT 逆元計算の説明で用いた記号 t や $k_1, k_2, \ldots, k_t$ を用いて, $n-1$ の長さ $\ell = k_1+t-1$[*2] の加法連鎖列 $\{p_s\}_{s=0}^{\ell}$ を考える.

$$p_s = \begin{cases} 1 & s = 0 \\ p_{s-1} + p_{s-1} & 1 \le s \le k_1 \\ p_{s-1} + p_{k_{s-k_1+1}} & k_1 + 1 \le s \le \ell \end{cases}$$

Itoh と Tsujii の FLT 逆元計算は加法連鎖列 $\{p_s\}_{s=0}^{\ell}$ と関連があり, $f^{2^{p_0}-1} = f$ から始めて, 逐次 $f^{2^{p_1}-1}, f^{2^{p_2}-1}, \ldots, f^{2^{p_\ell}-1}$ を計算している. 実際, $1 \le s \le k_1$ の場合は式 (1) によって $\left(f^{2^{p_{s-1}}-1}\right)^{2^{p_{s-1}}} \times f^{2^{p_{s-1}}-1} = f^{2^{p_{s-1}+p_{s-1}}-1} = f^{2^{p_s}-1}$ が成り立ち, $k_1+1 \le s \le \ell$ の場合も, 式 (2) によって $\left(f^{2^{p_{s-1}}-1}\right)^{2^{p_{k_{s-k_1+1}}}} \times f^{2^{p_{k_{s-k_1+1}}}-1} = f^{2^{p_{s-1}+p_{k_{s-k_1+1}}}-1} = f^{2^{p_s}-1}$ が成り立つ. あとは, $s = \ell$ のとき $f^{2^{p_\ell}-1} = f^{2^{n-1}-1}$ を得るので, 最後に二乗を1回行えば f^{2^n-2} が得られる. 式 (1), (2) の計算には, 二乗算ではない乗算をそれぞれ1回ずつ要する. 従って, 加法連鎖列の長さ ℓ は, Itoh と Tsujii の FLT 逆元計算で行われる乗算回数と一致している. このことは, Itoh と Tsujii の手法を量子アルゴリズムで記述しているアルゴリズム 1, 2 にも言える. すなわち, 加法連鎖列の長さ ℓ は, アルゴリズム 1, 2 において MODMULT_Imp を使用する回数となる.

ここまでの議論で, Itoh と Tsujii の FLT 逆元計算は, $n-1$ のある加法連鎖列を与え, それに基づいて計算を進めていると考えることができ, これは量子 FLT 逆元計算においても同様である. そこで我々は, 任意の $n-1$ の加法連鎖列 $\{p_s\}_{s=0}^{\ell}$ に対してそれに基づいて計算を進める量子 FLT 逆元計算アルゴリズムが存在することを示す. また, $p_s = p_i + p_j$ で計算される p_s を, $i = j, i \neq j$ のときそれぞれ二倍算で得られる項・加算で得られる項と呼ぶことにする. ただし, 便宜的に初項 p_0 は加算で得られる項とする. 既存の量子 FLT 逆元計算アルゴリズムで説明した通り, 二倍算で得られる項では $f^{2^{p_s}-1}$ の計算で補助レジスタが必要になり効率が悪い. そのため, 各項 p_s は複数通りの計算が可能な場合があるが, なるべく効率的なアルゴリズムを得るために, 各項 p_s が二倍算ではない加算で得られる場合には加算で得られる項とする.

実際に一般の加法連鎖列に対する証明を行うのは複雑になるので, まず以下のように証明しやすい加法連鎖列に変形できることを示す.

補題 1. 任意の加法連鎖列 $\{p'_s\}_{s=0}^{\ell}$ に対して, 各項の計算法を変えずに順番を並び替えた加法連鎖列 $\{p_s\}_{s=0}^{\ell}$ で, 以下の条件 $(i), (ii)$ を満たすようなものがある.

(i) $\{p_s\}_{s=0}^{\ell}$ のうち加算で得られる項を取り出してそのまま並べた数列は単調増加数列である.

(ii) p_s が二倍算で得られるとき, $p_{s-1} = p_s/2$ が成り立つ.

証明. $\{p'_s\}_{s=0}^{\ell}$ から加算で得られた項だけ取り出し, それらを単調増加になるように並び替えた $\{p'_s\}_{s=0}^{\ell}$ の部分数列を作る. この部分数列に対し, 二倍算で得られる項 p'_s を小さい順に $p'_s/2$ の直後に挿入する. この結果得られた数列を $\{p_s\}_{s=0}^{\ell}$ とおけば, $\{p_s\}_{s=0}^{\ell}$ は条件 $(i), (ii)$ を満たす. また, $\{p_s\}_{s=0}^{\ell}$ が加法連鎖列であることは自明である. □

補題 1 で得られる加法連鎖列に対して, 以下が成り立つ.

定理 1. $f \in \mathbb{F}_{2^n}^*$ と $n-1$ の長さ ℓ 及び二倍算 d 回, 加算 m 回で得られる補題 *1* の加法連鎖列 $\{p_s\}_{s=0}^{\ell}$ を入力として, 新たに $(2d+m+1)n$ 量子ビットを利用することで $f^{2^{n-1}-1}$ を計算する量子アルゴリズムが存在する.

証明. 集合 D, M をそれぞれ $D = \{s \in \{1, 2, \ldots, \ell\} \mid p_s$ は二倍算で得られる項$\}, M = \{s \in \{1, 2, \ldots, \ell\} \mid p_s$ は加算で得られる項$\}$ とおく. また, $1 \le s \le \ell$ について p_s は $p_s = p_{a_s} + p_{b_s}$ で得られるとして数列 $\{a_s\}_{s=1}^{\ell}, \{b_s\}_{s=1}^{\ell}$ を定める. ただし, $a_s \le b_s$ とする. このとき, $f^{2^{p_0}-1} = f$ から始めて $f^{2^{p_1}-1}, \ldots, f^{2^{p_{n-1}}-1}$ を順に計算できることを示す. 今, ある $1 \le u \le \ell$ に対して $f^{2^{p_{u-1}}-1}$ を計算したとし, 次に $f^{2^{p_u}-1}$ を計算する.

まず, $f^{2^{p_{a_u}}-1}, f^{2^{p_{b_u}}-1}$ が保持されているとき, $u \in D, M$ のときそれぞれ以下のように $f^{2^{p_u}-1}$ を計算できる. ただし, 補題 1 の加法連鎖列の並べ方より, $u \in D$ のときには必ず $f^{2^{p_{a_u}}-1} = f^{2^{p_{b_u}}-1} = f^{2^{p_{u-1}}-1}$ が保持されていることに注意されたい.

$u \in D$ のとき：アルゴリズム 1 の 1 行目から 5 行目のルー

---

*2 簡単のため, 以下では $t \neq 1$ として議論する. 実際, NIST の推奨パラメータでは常にこの条件が成り立つ.

プ 1 回分と同じ手順により $f^{2^{p_u}-1}$ を計算する．つまり，$f^{2^{p_{a_u}}-1}$ に ADD を適用することでもう 1 つ用意し，片方に SQUARE を p_{a_u} 回適用して $\left(f^{2^{p_{a_u}}-1}\right)^{2^{p_{a_u}}}$ とし，これと $f^{2^{p_{a_u}}-1}$ に MODMULT_Imp を適用すると，$p_u = p_{a_u} + p_{b_u} = p_{a_u} + p_{a_u} = 2p_{a_u}$ より $\left(f^{2^{p_{a_u}}-1}\right)^{2^{p_{a_u}}} \times f^{2^{p_{a_u}}-1} = f^{2^{p_u}-1}$ を得る．このとき新たに 2 レジスタ，すなわち $2n$ 量子ビットが必要となる．

$u \in M$ のとき：アルゴリズム 1 の 6 行目から 9 行目のループ 1 回分と同じ手順により $f^{2^{p_u}-1}$ を計算する．つまり，$f^{2^{p_{a_u}}-1}$ に SQUARE を p_{b_u} 回適用して $\left(f^{2^{p_{a_u}}-1}\right)^{2^{p_{b_u}}}$ とし，これと $f^{2^{p_{b_u}}-1}$ に MODMULT_Imp を適用すると，$p_u = p_{a_u} + p_{b_u}$ より $\left(f^{2^{p_{a_u}}-1}\right)^{2^{p_{b_u}}} \times f^{2^{p_{b_u}}-1} = f^{2^{p_u}-1}$ を得る．この計算において，新たに必要となる量子ビット数は，MODMULT_Imp を使用する際に $f^{2^{p_u}-1}$ を格納するための n ビットだけである．

この計算において，$u \in M$ のときには $f^{2^{p_{a_u}}-1}$ が $\left(f^{2^{p_{a_u}}-1}\right)^{2^{p_{b_u}}} = f^{2^{p_{a_u}+p_{b_u}}-2^{p_{b_u}}}$ と変化することに注意されたい．$u \in D$ のときには，新たなレジスタを用いることで $f^{2^{p_{a_u}}-1}$ を二つ用意し，片方のみを $\left(f^{2^{p_{a_u}}-1}\right)^{2^{p_{a_u}}} = f^{2^{2p_{a_u}}-2^{p_{a_u}}}$ とするので，依然 $f^{2^{p_{a_u}}-1}$ を保持している．

次に，$f^{2^{p_{a_u}}-1}, f^{2^{p_{b_u}}-1}$ が保持されておらず，計算過程である非負整数 c_u, d_u に対して $f^{2^{p_{a_u}+p_{c_u}}-2^{p_{c_u}}}, f^{2^{p_{b_u}+p_{d_u}}-2^{p_{d_u}}}$ と変化した場合を考える．このとき，$c_u \neq 0$ かつ $d_u = 0$ のとき上記 $(c_u, d_u) = (0, 0)$ の場合の計算手順と同様にして $f^{2^{p_u}-1}$ が計算でき，$d_u \neq 0$ のとはならないことを示す．

$c_u \neq 0$ かつ $d_u = 0$ のとき：$f^{2^{p_{a_u}}-1}$ は $f^{2^{p_{a_u}+p_{c_u}}-2^{p_{c_u}}} \times f^{2^{p_{c_u}}-1} = f^{2^{p_{a_u}+p_{c_u}}-1}$ なる計算を行うために $f^{2^{p_{a_u}+p_{c_u}}-2^{p_{c_u}}}$ に変化しており，補題 1 の加法連鎖列の並べ方より $p_{a_u} + p_{c_u} < p_u = p_{a_u} + p_{b_u}$，つまり $c_u < b_u$ である．そのため，SQUARE を $p_{b_u} - p_{c_u}$ 回適用して $\left(f^{2^{p_{a_u}+p_{c_u}}-2^{p_{c_u}}}\right)^{2^{p_{b_u}-p_{c_u}}} = \left(f^{2^{p_{a_u}}-1}\right)^{2^{p_{b_u}}}$ とし，上記 $(c_u, d_u) = (0, 0)$ の場合の計算手順と同様にして $f^{2^{p_u}-1}$ が計算できる．

$d_u \neq 0$ のとき：$f^{2^{p_{b_u}}-1}$ は $f^{2^{p_{b_u}+p_{d_u}}-2^{p_{c_u}}} \times f^{2^{p_{d_u}}-1} = f^{2^{p_{b_u}+p_{d_u}}-1}$ なる計算を行うために $f^{2^{p_{b_u}+p_{d_u}}-2^{p_{d_u}}}$ に変化している．$p_{u'} = p_{b_u} + p_{d_u}$ とすると，計算手順より $u' < u$，つまり $p_{u'} < p_u$ が成り立つ．このとき，$a_{u'} = b_u, b_{u'} = d_u$ なので $b_u < d_u$ が成り立つが，$a_u < b_u$ より $p_{u'} = b_u + d_u > a_u + b_u = p_u$ となり矛盾．よって，このような状況は起こらない．

よって，任意の加法連鎖列に対して量子 FLT 逆元計算が可能であることがわかった．

また $s \in D$ のときは 1 回の乗算と 1 回の加算を用い，$s \in M$ のときは 1 回の乗算を用いているから，入力の n ビットと合わせて必要となる量子ビット数は $(2d+m+1)n$ であり，合計乗算回数は $d + m = \ell$ 回である．□

アルゴリズム 3 基本アルゴリズム

入力: n 次既約多項式 $m(x) \in \mathbb{F}_2[x]$，$n-1$ の加法連鎖列 p_s(二倍算 d 回，加算 m 回) とそれに伴う数列 a_s, b_s, Q_s，多項式 $g_0 = f \in \mathbb{F}_{2^n}^*$，全てのビットが $|0\rangle$ で初期化された多項式 $g_1, \cdots, g_{d+m}, h_0, \cdots, h_{d-1}$

出力: $g_{d+m} = f^{2^n-2}$

1: $dcount \leftarrow 0$
2: **for** $s = 1, \ldots, d+m$ **do**
3:     **if** $s \in D$ **then**
4:         ADD(h_{dcount}, g_{a_s})
5:         **for** $i = 1, \ldots, Q_s$ **do**
6:             SQUARE(h_{dcount})
7:         MODMULT_Imp(g_s, g_{a_s}, h_{dcount})
8:         $dcount \leftarrow dcount + 1$
9:     **else** $\{s \in M\}$
10:         **for** $i = 1, \ldots, Q_s$ **do**
11:             SQUARE(g_{a_s})
12:         MODMULT_Imp(g_s, g_{a_s}, g_{b_s})
13: SQUARE(g_{d+m})

定理 1 によって構成される基本アルゴリズムを**アルゴリズム 3** に示す．上の証明の内容からは，二乗を繰り返す過程において，何回の二乗計算が必要であるかが不明瞭であるので，$f^{2^{p_{a_s}}-1}$ を二乗する回数を定めた一意に定まる数列 $\{Q_s\}_{s=1}^{\ell}$ を用意してアルゴリズムを記述している．

4.3 拡張アルゴリズム

基本アルゴリズムでは，3 行目から 8 行目の if 文の中の計算を行う際に，1 つ 1 つの多項式に対して別々の量子ビット，すなわち合計 dn ビットを用意して計算をおこなっていた．ところが，Banegas らのアルゴリズム 2 で用いられている SQUARE の逆演算 SQUARE^{-1} を用いると，CNOT と深さを多く使用することで，使用する量子ビット数を削減できる．我々はこの手法を一般化し，パラメータ $L \in [0, d-1]$ について加法連鎖列の二倍算に対応する $\mathbb{F}_{2^n}^*$ の計算に必要な量子ビットを Ln ビット削減する．加えて，加法連鎖列の最後に出てくる項に対応する多項式も二倍算計算に対応する計算で用いた量子ビットに格納することが可能で，これによりさらに n 量子ビットの節約が可能である．この削減法はアルゴリズム 2 でも行われているものだが，加法連鎖の最後の項に対応する多項式が二倍算によって得られている場合は適用できない．だが，NIST が推奨する各パラメータ n について，加法連鎖の最後の項が加算で得られるような最適な列が存在するため，問題は生じない．以上 2 つの量子ビット削減を行い，アルゴリズム 3 と比較して量子ビットを $(L+1)n$ ビット削減するアルゴリズムを**アルゴリズム 4** に示す．入力はアルゴリズム 3 と同様のものにパラメータ L と配列 pl 及び数列 $\{c\ell_t\}_{t=0}^{d}$ が加わる．pl は $d-L$ 個の要素を持つ配列で，ガベージ処理の際，ADD 演算に用いられる多項式 g の添字を格納す

アルゴリズム 4 拡張アルゴリズム

入力: n 次既約多項式 $m(x) \in \mathbb{F}_2[x]$，$n-1$ の加法連鎖列 p_s(二倍算 d 回，加算 m 回) とそれに伴う数列 $a_s, b_s, Q_s, c\ell_t$，多項式 $g_0 = f \in \mathbb{F}_{2^n}^*$，全てのビットが $|0\rangle$ で初期化された多項式 $g_1, \cdots, g_{d+m-1}, h_0, \cdots, h_{d-L-1}$，全要素が -1 で初期化された配列 $\mathrm{pl}[d-L]$

出力: $h_{\overline{d}} = f^{2^n-2}$

$dcount \leftarrow 0$
for $s = 1, \ldots, d+m$ **do**
  if $s \in D$ **then**
    if $\mathrm{pl}[\overline{dcount}] \neq -1$ **then**
      GARBAGECLEAR($c\ell_{dcount}, \mathrm{pl}[\overline{dcount}], \overline{dcount}$)
    ADD($h_{\overline{dcount}}, g_{a_s}$)
    for $i = 1, \ldots, Q_s$ **do**
      SQUARE($h_{\overline{dcount}}$)
    MODMULT_Imp($g_s, g_{a_s}, h_{\overline{dcount}}$)
    $\mathrm{pl}[\overline{dcount}] \leftarrow a_s$
    $dcount \leftarrow dcount + 1$
  else $\{s \in M\}$
    for $i = 1, \ldots, Q_s$ **do**
      SQUARE(g_{a_s})
    MODMULT_Imp(g_s, g_{a_s}, g_{b_s})
if $\mathrm{pl}[\overline{d}] \neq -1$ **then**
  GARBAGECLEAR($c\ell_d, \mathrm{pl}[\overline{d}], \overline{d}$)
for $i = 1, \ldots, Q_{d+m}$ **do**
  SQUARE($g_{a_{d+m}}$)
MODMULT_Imp($h_{\overline{d}}, g_{a_{d+m}}, g_{b_{d+m}}$)
SQUARE($h_{\overline{d}}$)

る．また，$0 \leq t \leq d$ について，$c\ell_t$ はガベージ処理においてSQUARE またはSQUARE^{-1} を何回行うかを表す．なお，$c\ell_0 = 0$ とし，$\overline{x} \coloneqq x \mod (d-L)$ とする．ガベージ処理は，二倍算に対応する計算の際に，h_0 から h_{d-L-1} に対して順番に多項式を格納していき，格納しきれなくなった場合は h_0 から順に 0 に初期化することで行う．ここで，GARBAGECLEAR は初期化の関数だが，ページ数の都合で詳細は省略する．

5. 既存手法との比較

本章では，第 5.1 節において，リソースの削減基準を設けた際，提案手法において最適な加法連鎖列と関連数列を示す．続く第 5.2 節では，第 5.1 節で与えた加法連鎖列を用いて NIST の推奨パラメータ $n = 163, 233, 283, 409, 571$ での 1 回の量子逆元計算を行う場合の既存アルゴリズムと提案アルゴリズムとのリソース数の比較結果を示す．

5.1 最適な加法連鎖列

定理 1 から，アルゴリズム 3 は与えられた加法連鎖列の d, m, ℓ に依存して使用する Toffoli ゲート・量子ビット・および深さが変化する．なお，Itoh と Tsujii の FLT 逆元計算による加法連鎖列では，$(n; d, m, \ell) = (163; 7, 2, 9), (233; 7, 3, 10), (283; 8, 3, 11), (409; 8, 3, 11),$

表 1 $n = 571$ における提案加法連鎖列 $\{p_s\}_{s=0}^{12}$ と，Itoh と Tsujii の FLT 逆元計算における加法連鎖列 $\{q_s\}_{s=0}^{13}$

$\{p_s\}_{s=0}^{12}$	$1, 2, 4, 8, 16, 18, 34, 50, 84, 134, 218, 352, 570$
$\{q_s\}_{s=0}^{13}$	$1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 544, 560, 568, 570$

$(571; 9, 4, 13)$ である．我々は，Toffoli ゲート，量子ビット数の順に優先的に削減することで，$(n; d, m, \ell) = (163; 5, 4, 9), (233; 4, 6, 10), (283; 3, 8, 11), (409; 7, 3, 10),$ $(571; 4, 8, 12)$ となる最適な加法連鎖列を得る．ただし，深さは考慮しない．この結果から，$n = 409, 571$ では加法連鎖列の長さ ℓ と二倍算回数 d が減少し，$n = 163, 233, 283$ では二倍算回数 d が減少していることがわかる．また，$n = 571$ のとき，最適化された加法連鎖列 $\{p_s\}_{s=0}^{12}$ と，Itoh と Tsujii の FLT 逆元計算における加法連鎖列 $\{q_s\}_{s=0}^{13}$ を**表 1** に示す．

5.2 1 回の量子逆元計算での比較

表 2 に，アルゴリズム 3（提案基本アルゴリズム），アルゴリズム 4（提案拡張アルゴリズム）による Toffoli ゲート・量子ビット数・深さを，Putranto らの FLT アルゴリズム (PWLK22-FLT) [10] と Banegas らの FLT アルゴリズム (BBHL21-FLT) [2]，及び Banegas らの GCD アルゴリズム (BBHL21-GCD) [2] と比較した結果を示す．深さはいずれも上界値である．比較は 1 回の逆元計算により行い，拡張アルゴリズムは $L = d - 1$ としている．ここで，SQUARE の深さは Banegas ら [2] を MODMULT_Imp の使用する Toffoli ゲート数・深さは Hoof [7], [8] を参考にしている．さらに，提案拡張アルゴリズムと BBHL21-FLT の深さは，最後に行われるガベージ処理は完全に並列計算できることを考慮した値である．ただし，BBHL21-GCD の $n = 409$ での深さは先行研究で計算されておらず，厳密な比較が困難なため空欄としている．

まず，量子 FLT アルゴリズム同士を比較する．提案基本アルゴリズム・拡張アルゴリズムはそれぞれ PWLK22-FLT・BBHL21-FLT と比較すると提案手法の改良が確認しやすい．まず，$n = 409, 571$ のときは Itoh と Tsujii の加法連鎖列より長さ ℓ と二倍算回数 d が小さい加法連鎖列が得られたことで，いずれの場合も提案基本アルゴリズム・拡張アルゴリズムはそれぞれ PWLK22-FLT・BBHL21-FLT より Toffoli ゲート数・量子ビット数・深さが小さくなっており，完全な改良になっていると言える．特に，$n = 409$ のときは深さを犠牲にして量子ビット数を削減した提案拡張アルゴリズムが既存アルゴリズムの中では深さに強みがあった PWLK22-FLT よりも深さが小さくなっており，大幅な改良となっている．$n = 163, 233, 283$ のときは Itoh と Tsujii の加法連鎖列より二倍算回数 d のみが小さい加法連鎖列を得られたことで，提案基本アルゴリズム・拡張アルゴリズ

表 2 提案手法と，従来手法（PWLK22-FLT・BBHL21-FLT・BBHL21-GCD）の 1 回の量子逆元計算における Toffoli ゲート・量子ビット・深さによる比較

n	提案基本アルゴリズム			提案拡張アルゴリズム		
	Toffoli ゲート	量子ビット	深さ	Toffoli ゲート	量子ビット	深さ
163	39,483	2,445	447,144	39,483	1,630	473,544
233	63,230	3,495	711,082	63,230	2,563	713,826
283	113,003	4,245	1,285,550	113,003	3,396	1,341,710
409	171,010	7,362	2,022,644	171,010	4,499	2,069,080
571	374,052	9,707	4,934,513	374,052	7,423	5,315,623

n	PWLK22-FLT			BBHL21-FLT			BBHL21-GCD		
	Toffoli ゲート	量子ビット	深さ	Toffoli ゲート	量子ビット	深さ	Toffoli ゲート	量子ビット	深さ
163	39,483	2,771	447,148	39,483	1,630	488,740	437,774	1,156	594,425
233	63,230	4,194	711,088	63,230	2,563	735,796	821,654	1,646	1,143,203
283	113,003	5,660	1,285,560	113,003	3,396	1,434,164	1,192,714	1,997	1,667,315
409	188,111	8,180	2,196,082	188,111	4,908	2,258,834	2,342,338	2,879	−
571	405,223	13,133	5,178,165	405,223	7,994	6,023,251	4,430,502	4,014	6,372,485

ムは Toffoli ゲート数を変えずにそれぞれ PWLK22-FLT の量子ビット数・BBHL21-FLT の深さを改良している．ただし，提案拡張アルゴリズムと BBHL21-FLT の量子ビット数は等しいもの，提案基本アルゴリズムは非常にわずかだが PWLK22-FLT の深さを改良している．最後に，提案手法と BBHL21-GCD を比較すると，量子ビット数は最大 2 倍程度になるが，Toffoli ゲート数を約 10 分の 1 程度となっており，いずれの場合も深さを削減している．

6. 結論

本論文では，量子 FLT 逆元計算アルゴリズムを加法連鎖の観点からより一般的に捉えなおした．任意の加法連鎖列に基づいた量子 FLT 逆元計算アルゴリズムの存在を示し，加法連鎖列とその計算方法による Toffoli ゲート数・量子ビット数を導出した．提案アルゴリズムは Putranto ら [10] と Banegas ら [2] の量子 FLT 逆元計算アルゴリズムを参考にしているが，NIST が推奨する全ての次数 n において二つの既存アルゴリズムを改良している．

今回は逆元計算一回の比較を行ったが，複数回行うときには前の計算で使用したリソースを効率的に使いまわせる可能性がある．だが，これまでの研究では逆元計算アルゴリズムを素朴に複数回実行する方法しか知られておらず，Shor のアルゴリズムで実行するときのように，複数回実行することを考慮した場合の解析・改良が今後の課題となる．

謝辞 本研究は JSPS 科研費 19K20267，JP21H03440，JST CREST JPMJCR2113 の助成を受けたものです．

参考文献

[1] Azarderakhsh, R., Järvinen, K. and Dimitrov, V.: Fast Inversion in GF(2^m) with Normal Basis Using Hybrid-Double Multipliers, *IEEE Trans. computers*, Vol. 63, No. 4, pp. 1041–1047 (2012).

[2] Banegas, G., Bernstein, D. J., van Hoof, I. and Lange, T.: Concrete quantum cryptanalysis of binary elliptic curves, *IACR Trans. CHES*, Vol. 2021, No. 1, pp. 451–472 (2020).

[3] Cameron, F. and Patrick, D.: FIPS PUB 186-4 Digital Signature Standard (DSS), NIST, pp. 92–101 (2013).

[4] Canto, A. C., Kermani, M. M. and Azarderakhsh, R.: CRC-Based Error Detection Constructions for FLT and ITA Finite Field Inversions Over GF(2^m), *IEEE Trans. VLSI Systems*, Vol. 29, No. 5, pp. 1033–1037 (2021).

[5] Guajardo, J. and Paar, C.: Itoh-Tsujii inversion in standard basis and its application in cryptography and codes, *Designs, Codes and Cryptography*, Vol. 25, No. 2, pp. 207–216 (2002).

[6] Hu, J., Guo, W., Wei, J. and Cheung, R. C.: Fast and Generic Inversion Architectures Over GF(2^m) Using Modified Itoh–Tsujii Algorithms, *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 62, No. 4, pp. 367–371 (2015).

[7] Iggy, v. H.: Quantum modulo Karatsuba multiplier for binary polynomials, https://github.com/ikbenbeter/QMKMBP (2019).

[8] Iggy, v. H.: Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count, *CoRR*, Vol. abs/1910.02849 (2019).

[9] Itoh, T. and Tsujii, S.: A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases, *Information and computation*, Vol. 78, No. 3, pp. 171–177 (1988).

[10] Putranto, D. S. C., Wardhani, R. W., Larasati, H. T. and Kim, H.: Another Concrete Quantum Cryptanalysis of Binary Elliptic Curves, Cryptology ePrint Archive, Paper 2022/501 (2022).

[11] Rodriguez-Henriquez, F., Cruz-Cortes, N. and Saqib, N.: A fast implementation of multiplicative inversion over GF(2^m), ITCC'05, Vol. 1, IEEE, pp. 574–579 (2005).

[12] Roetteler, M., Naehrig, M., Svore, K. M. and Lauter, K.: Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms, ASIACRYPT 2017, pp. 241–270 (2017).

[13] Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring, *FOCS 1994*, pp. 124–134 (1994).