Concrete Quantum Cryptanalysis of Binary Elliptic Curves via Addition Chain*

Ren Taguchi† Atsushi Takayasu‡

August 3, 2023

Abstract

Thus far, several papers reported concrete resource estimates of Shor's quantum algorithm for solving the elliptic curve discrete logarithm problem (ECDLP). In this paper, we study quantum FLT-based inversion algorithms over binary elliptic curves. There are two major algorithms proposed by Banegas et al. and Putranto et al., where the former and latter algorithms achieve fewer numbers of qubits and smaller depths of circuits, respectively. We propose two quantum FLT-based inversion algorithms that essentially outperform previous FLT-based algorithms and compare the performance for NIST curves of the degree n. Specifically, for all n, our first algorithm achieves fewer qubits than Putranto et al.'s one without sacrificing the number of Toffoli gates and the depth of circuits, while our second algorithm achieves smaller depths of circuits without sacrificing the number of qubits and Toffoli gates. For example, when $n = 571$, the number of qubits of our first algorithm is 74 % of that of Putranto et al.'s one, while the depth of our second algorithm is 83 % of that of Banegas et al.'s one. The improvements stem from the fact that FLT-based inversions can be performed with arbitrary sequences of addition chains for $n - 1$ although both Banegas et al. and Putranto et al. follow fixed sequences that were introduced by Itoh and Tsujii's classical FLT-based inversion. In particular, we analyze how several properties of addition chains, which do not affect the computational resources of classical FLT-based inversions, affect the computational resources of quantum FLT-based inversions and find appropriate sequences.

*This is the full version of [TT23]. This research was in part conducted under a contract of "Research and Development for Expansion of Radio Wave Resources (JPJ000254)" the Ministry of Internal Affairs and Communications, Japan, and JSPS KAKENHI Grant Numbers JP19K20267 and JP21H03440, Japan.

†Graduate School of Information Science and Technology, the University of Tokyo, Japan. rtaguchi-495@g.ecc.u-tokyo.ac.jp

‡Graduate School of Information Science and Technology, the University of Tokyo, Japan, and National Institute of Advanced Industrial Science and Technology, Japan. takayasu-a@g.ecc.u-tokyo.ac.jp

Contents

1 Introduction

1.1 Background

RSA [RSA78] and elliptic-curve cryptography (ECC) [Kob87, Mil85] are public-key cryptosystems that are the most widely used in practice. RSA and ECC are believed to be secure since there are no known polynomial time algorithms for solving the factorization problem and elliptic curve discrete logarithm problem (ECDLP). NIST [CP13] recommends elliptic curves for ECC over a prime field $\mathbb{F}_q$ and a binary field $\mathbb{F}_{2^n}$. Specifically, degrees $n = 163, 233, 283$, and 571 are recommended for binary elliptic curves. However, Shor [Sho94] proposed a quantum algorithm that solves the factorization problem and ECDLP in polynomial time. Then, designing post-quantum public key cryptosystems (PQC) has been paid much attention and the timing of the transition to PQC has been actively discussed.

Despite the theoretical effectiveness, Shor's algorithm is currently not efficient in practice. For example, there are several reports of the quantum algorithm to solve the factorization problem [ASK19, DLQ+20, LBC+12, LBYP07, MLLL+12, LWL+07, MNM+16, PMO09, SSV13, VSB+01]; however, the target composite integers are mainly 15 and 21, while the classical factorization of 795-bit composite integers has been reported [BGG+20]. The situation stems from the fact that physical realizations of large-scale quantum computers have a lot of technical barriers. Thus, there are several papers [GE21, GS21, HLH22, VBE96, Zal98, Bea03, FMMC12, HRS17, TK06, Kun05] that estimate the concrete resource estimates of quantum factoring and its improvements in terms of the number of qubits, the number of quantum gates, and depth of circuits.

Compared with the situation of quantum factoring, the quantum resource estimates of the ECDLP were not studied until recently. Although the first attempt was given by Proos and Zalka [PZ03], their analysis lacks the implementation of elliptic curve additions that are the most dominant step to run Shor's quantum algorithm. Roetteler et al. [RNSL17] showed the first concrete resource estimates of ECDLP over a prime field $\mathbb{F}_q$ by indicating how to perform elliptic curve additions quantumly. Subsequently, Banegas et al. [BBvHL20] gave the alternative results for a binary field $\mathbb{F}_{2^n}$ and the work was followed by Putranto et al. [PWLK22].

In this paper, we focus on binary elliptic curves. We especially study an inversion in $\mathbb{F}_{2^n}$, where the computation is the most dominant operation to realize elliptic curve additions. For this purpose, Banegas et al. [BBvHL20] proposed two quantum methods for inversion in $\mathbb{F}_{2^n}$, i.e., an extended GCD-based inversion and FLT-based inversion[1] inspired by Bernstein and Yang's inversion [BY19] and Itoh and Tsujii's inversion [IT88], respectively. Their results indicate that the extended GCD-based inversion requires fewer qubits, while the FLT-based inversion requires fewer Toffoli gates and a smaller depth of circuits. Although Banegas et al. [BBvHL20] tried to minimize the required number of qubits, Putranto et al. [PWLK22] revisited the analysis to minimize the depth of circuits. Then, Putranto et al. proposed a quantum FLT-based inversion algorithm that works with a smaller depth of circuits and larger qubits than Banegas et al.'s FLT-based inversion algorithm, while the numbers of Toffoli gates are unchanged.

1.2 Our Contribution

In this paper, we propose two quantum FLT-based inversion algorithms. We concretely analyze quantum resources for the algorithms over NIST-recommended curves. Then, we show that our proposed algorithms improve previous FLT-based inversion algorithms by Banegas et al. [BBvHL20] and Putranto et al. [PWLK22] for all degrees $n = 163, 233, 283$, and 571. Briefly speaking, our first and second algorithms are based on FLT-based inversion algorithms by Putranto et al. and Banegas et al., respectively. Intuitively, our algorithms successfully overcome the disadvantages of previous FLT-based inversion algorithms. Indeed, for all degrees n, our first and second algorithms require fewer qubits and smaller depth of circuits than Putranto et al. and Banegas et al., respectively. Moreover, we want to claim two further benefits of our algorithms. At first, our algorithms do not sacrifice the advantages of previous FLT-based inversion algorithms in the sense that the number of qubits, number of Toffoli gates, and depth of circuits of our first and second algorithms do not exceed those of Putranto et al. and Banegas et al., respectively. Next, our algorithms successfully reduce the number of Toffoli gates of previous FLT-based inversion algorithms for $n = 571$. In other words, our algorithms improve all three factors of previous FLT-based inversion algorithms for $n = 571$. For example, our first (resp. second) algorithm for $n = 571$ requires 74%, 93%, and 97% (resp. 93%, 93%, and 79%)

[1] FLT is the abbreviation of Fermat's little theorem

of qubits, Toffoli gates, and depth of Putranto et al.'s algorithm (resp. Banegas et al.'s algorithm). We also apply windowing to our algorithms. Windowing is a way for reducing Toffoli gates by using quantum read-only memory (QROM). Both Banegas et al. [BBvHL20] and Putranto et al. [PWLK22] also estimated the number of Toffoli gates when windowing is applied.

Difference from Preliminary Version. In the preliminary version [TT23], we use quantum multiplication by Hoof [Igg19] to estimate quantum resources. Recently, Kim et al. [KKKH22] proposed a new quantum multiplication that has an advantage over Hoof's one in terms of the number of Toffoli gates and depth. Therefore, in this version, we use Kim et al.'s quantum multiplication to estimate quantum resources.

1.3 Technical Overview

Both previous quantum FLT-based inversion algorithms by Banegas et al. [BBvHL20] and Putranto et al. [PWLK22] are modifications of Itoh and Tsujii's classical FLT-based inversion algorithm [IT88]. Given $f \in \mathbb{F}_{2^n}^*$, both classical and quantum FLT-based inversion algorithms compute $f^{-1} \in \mathbb{F}_{2^n}^*$ based on the fact that $f^{2^n-2} = f^{-1}$. Itoh and Tsujii's inversion finally computes f^{-1} by $\left(f^{2^{n-1}-1}\right)^2 = f^{2^n-2}$ and the main step of the algorithm is a computation of $f^{2^{n-1}-1}$. Here, we describe how to compute $f^{2^{n-1}-1} = f^{2^{162}-1}$ when $n = 163$. Observe that 162 has Hamming weight three in binary, where $162 = 128 + 32 + 2 = 2^7 + 2^5 + 2^1$. We start from $f = f^{2^{2^0}-1}$ and compute each $f^{2^{2^1}-1}, f^{2^{2^2}-1}, \ldots, f^{2^{2^7}-1}$. Specifically, given $f^{2^{2^{k-1}}-1}$ for $k = 1, 2, \ldots, 7 = \lfloor \log 162 \rfloor$, we can compute $f^{2^{2^k}-1}$ by

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

with seven field multiplications. Then, we compute $f^{2^{2^7+2^5}-1}$ and $f^{2^{2^7+2^5+2^1}-1} = f^{2^{162}-1}$ by

$$\left(f^{2^{2^7}-1}\right)^{2^{2^5}} \times f^{2^{2^5}-1} = f^{2^{2^7+2^5}-2^{2^5}} \times f^{2^{2^5}-1} = f^{2^{2^7+2^5}-1},$$
$$\left(f^{2^{2^7+2^5}-1}\right)^{2^{2^1}} \times f^{2^{2^1}-1} = f^{2^{2^7+2^5+2^1}-2^{2^1}} \times f^{2^{2^1}-1} = f^{2^{2^7+2^5+2^1}-1},$$

with two field multiplications. Thus, nine field multiplications in total are required for computing $f^{2^{162}-1}$. In general, Itoh and Tsujii's inversion requires $\lfloor \log(n-1) \rfloor + t - 1$ field multiplications, where t denotes the Hamming weight of $n-1$ in binary.

Next, we explain how to perform FLT-based inversion quantumly. Putranto et al.'s algorithm [PWLK22] is simpler than Banegas et al.'s algorithm [BBvHL20] since Banegas et al.'s algorithm can be viewed as a modification of Putranto et al.'s algorithm by clearing garbages and reduces the required number of qubits. Therefore, we use Putranto et al.'s algorithm to explain an overview of quantum FLT-based inversion. For simplicity, we focus on the number of qubits to perform Putranto et al.'s algorithm. At first, we describe how to compute compute each $f^{2^{2^1}-1}, f^{2^{2^2}-1}, \ldots, f^{2^{2^7}-1}$. A point to note is that when given $f^{2^{2^{k-1}}-1}$ as a quantum superposition in i-th register, we cannot efficiently compute $f^{2^{2^k}-1}$ in the next register. In turn, we apply CNOT gates and copy $f^{2^{2^{k-1}}-1}$ in an $(i+1)$-th register. Then, we apply CNOT gates to the i-th register and obtain $\left(f^{2^{2^{k-1}}-1}\right)^{2^{k-1}} = f^{2^{2^k}-2^{2^{k-1}}}$ in the i-th register. Finally, we apply Toffoli gates to the i-th and $(i+1)$-th registers and obtain $f^{2^{2^{k-1}}-1} \times f^{2^{2^k}-2^{2^{k-1}}} = f^{2^{2^k}-1}$ in the $(i+2)$-th register. Thus, when given $f = f^{2^{2^0}-1}$ in the first register, $2\lfloor \log 162 \rfloor + 1 = 15$ registers, i.e., $15n$ qubits, are required so far. Next, we explain how to compute $f^{2^{2^7+2^5}-1}$ and $f^{2^{2^7+2^5+2^1}-1} = f^{2^{162}-1}$. When given $f^{2^{2^7}-1}$ in i-th register and $f^{2^{2^5}-1}$ in j-th register, we apply CNOT gates to the i-th register and obtain $\left(f^{2^{2^7}-1}\right)^{2^{2^5}} = f^{2^{2^7+2^5}-2^{2^5}}$ in the i-th register. Then, we apply Toffoli gates to the i-th and j-th registers and obtain $f^{2^{2^7+2^5}-2^{2^5}} \times f^{2^{2^5}-1} = f^{2^{2^7+2^5}-1}$ in the 16-th register. Similarly, we can compute

$f^{2^{2^7+2^5+2^1}-1} = f^{2^{162}-1}$ to the 17-th register. Finally, we apply CNOT gates to the 17-th register and obtain $= f^{2^{163}-2}$ in the 17-th register. Therefore, 17 registers, i.e., $17n$ qubits, are required in total. In general, Putranto et al.'s quantum FLT-based inversion algorithm requires $(2\lfloor \log(n-1) \rfloor + t)n$ qubits.

Summarizing the above discussion, given $f = f^{2^{2^0}-1}$ and the previous FLT-based inversion algorithms for $n = 163$ computes $f^{2^{2^1}-1}, f^{2^{2^2}-1}, \ldots, f^{2^{2^7}-1}$, $f^{2^{2^7+2^5}-1}$, and $f^{2^{2^7+2^5+2^1}-1} = f^{2^{162}-1}$. The first key observation of our improvement is that the exponents of 2 during the calculation, i.e.,

$$\{2^0 = 1, 2^1, 2^2, \ldots, 2^7, 2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162\},$$

is an addition chain for $n-1 = 162$. In general, an addition chain for N is a sequence $p_0 = 1, p_1, \ldots, p_\ell = N$, where $p_s = p_i + p_j$ holds for some $0 \leq i, j < s$. Here, ℓ is called a length of an addition chain. We show that $f^{2^{n-1}-1}$ can be computed with an arbitrary addition chain for $n-1$ by following the similar steps of Putranto et al.'s algorithm. For example, there is another addition chain

$$\{1, 2, 4, 8, 16, 32, 33, 65, 97, 162\}$$

for 162. Keen readers may think that the observation is not interesting since the relation between FLT-based inversion and addition chain has been already discussed in the context of classical computation [RHCCS05, GP02, CKA21, AJD12, HGWC15]. These papers mentioned that the computational cost of FLT-based inversion relates to the length of addition chains in the sense that the number of field multiplications $\lfloor \log(n-1) \rfloor + t - 1$ is the same as the length of addition chains. Similarly, the computational cost of quantum FLT-based inversion relates to the length of addition chains in the sense that the number of Toffoli gates is determined by the length of addition chains. Here, the length of an addition chain $\{1, 2, 4, 8, 16, 32, 33, 65, 97, 162\}$ is nine which is the same as that of previous addition chain $\{2^0 = 1, 2^1, 2^2, \ldots, 2^7, 2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162\}$.

However, we show that the computational cost of quantum FLT-based inversion also depends on other properties of addition chains. Hereafter, for an addition chain $\{p_s\}_{s=0}^{\ell}$, we call p_s a doubled term if it is computed by $p_s = p_i + p_i$ for some $0 \leq i < s$ and an added term otherwise. In the above example for $n = 163$, $2^1, 2^2, \ldots, 2^7$ are doubled terms and $2^7 + 2^5, 2^7 + 2^5 + 2^1$ are added terms for $\{2^0 = 1, 2^1, 2^2, \ldots, 2^7, 2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162\}$ whereas $\{2, 4, 8, 16, 32\}$ are doubled terms and $\{33, 65, 97, 162\}$ are added terms for $\{1, 2, 4, 8, 16, 32, 33, 65, 97, 162\}$. For an addition chain $\{p_s\}_{s=0}^{\ell}$, let d and m denote the number of doubled terms and added terms, where $\ell = d + m$. Then, we show that the number of qubits $(2\lfloor \log(n-1) \rfloor + t)n$ for Putranto et al.'s algorithm is essentially described by $(2d + m + 1)n$. In other words, even if the lengths of addition chains are the same, the computational costs of the quantum FLT-based inversion algorithm may not be the same depending on other properties of addition chains. Indeed, an addition chain $\{2^0 = 1, 2^1, 2^2, \ldots, 2^7, 2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162\}$ has seven doubled terms and two added terms whereas $\{1, 2, 4, 8, 16, 32, 33, 65, 97, 162\}$ has five doubled terms $\{2, 4, 8, 16, 32\}$ and four added terms $\{33, 65, 97, 162\}$. Therefore, quantum FLT-based inversion based on the latter addition chain requires fewer qubits than that on the former. Based on the discussion and more, we find more appropriate addition chains for all $n = 163, 233, 283, 571$ and obtain our improvements.

1.4 Organization

In Section 3, we review previous FLT-based inversion algorithms. In Section 4, we propose quantum FLT-based inversion algorithms. In Section 5, we compare our proposed algorithms and previous quantum algorithms. In Section 6, we apply windowing to our algorithms.

2 Preliminaries

In Section 2.1, we review binary elliptic curves and the binary elliptic curve discrete logarithm problem (ECDLP). Then, we briefly explain Shor's algorithm for binary ECDLP in Section 2.2. We also describe an overview of quantum computing on the field $\mathbb{F}_{2^n}$ in Section 2.3.

2.1 Elliptic Curve Discrete Logarithm Problem

Let n be a positive integer. A binary elliptic curve of degree n is given by $y^2 + xy = x^3 + ax^2 + b$, where $a \in \mathbb{F}_{2^n}$ and $b \in \mathbb{F}_{2^n}^*$. In general, the set of rational points on an elliptic curve along with a special point O called a point at infinity forms a group under point addition, where O is a neutral element. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ denote points on a binary elliptic curve. When $P \neq Q$, a point addition $P + Q = (x_3, y_3)$ is given by

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

with $\lambda = (y_1 + y_2)/(x_1 + x_2)$. Let $[k]P$ denote $P + \cdots + P$ that is a sum of k P's under point addition. Then, $[2]P = (x_3, y_3)$ is given by

$$x_3 = \lambda^2 + \lambda + a, \quad y_3 = x_1^2 + (\lambda + 1)x_3$$

with $\lambda = x_1 + y_1/x_1$. It is known that only basic arithmetic in $\mathbb{F}_{2^n}$ is sufficient for computing point addition on a binary elliptic curve. Then, the task of the binary ECDLP is computing k from P and $[k]P$.

2.2 Shor's Algorithm for Binary ECDLP

Shor's algorithm for the binary ECDLP of degree n consists of two parts, i.e., the point addition part and Quantum Fourier Transform part. The point addition part requires $2n + 2$ times point additions with $O(n^3)$ gates, while the Quantum Fourier Transform part requires $O(n^2)$ gates. Therefore, the point addition part is dominant in Shor's algorithm. As we mentioned in Section 2.1, an inversion in $\mathbb{F}_{2^n}$, i.e., computation of λ, is required for performing point addition $P + Q$. Moreover, several works [RNSL17, HJN+20, PWLK22, BBvHL20] indicate that the inversion computation requires the largest quantum resources in point addition. Therefore, the efficiency of quantum inversion computations greatly affects the total quantum resources for Shor's algorithm.

2.3 Quantum Computation in $\mathbb{F}_{2^n}$

In quantum computation, we use a "qubit" represented by $|0\rangle, |1\rangle$ and their superposition. We represent an element of $\mathbb{F}_{2^n}$ by n qubits. Here, we use the fact that for $m(x) \in \mathbb{F}_2[x]$ which is an irreducible polynomial of degree n, it holds that $\mathbb{F}_{2^n} \simeq \mathbb{F}_2[x]/(m(x))$. Thus, we can express an element of $\mathbb{F}_{2^n}$ as a polynomial of degree at most $n - 1$ with its coefficients 0 or 1.

In quantum circuits, we use some quantum gates that are similar to NOT, AND, and OR in classical circuits. In this paper, we consider only CNOT gates, Toffoli (TOF) gates, and swap gates. Let a, b, and c denote quantum states of one-qubit. Then, CNOT, TOF, and swap operations are given by

$$\begin{aligned} \mathrm{CNOT}(a, b) &= (a, a \oplus b), \qquad \mathrm{TOF}(a, b, c) = (a, b, c \oplus (a \cdot b)), \\ \mathrm{swap}(a, b) &= (b, a), \end{aligned}$$

respectively. The swap gate consists of three CNOT gates, while the TOF gate is more expensive than a CNOT and swap gate.

We summarize known quantum algorithms which we will use for performing basic arithmetic in $\mathbb{F}_{2^n}$. Let `ADD` and `SQUARE` denote Banegas et al.'s algorithms [BBvHL20] for addition and squaring, respectively, while `MODMULT` denote Kim et al.'s algorithm [KKKH22] for multiplication. Let f, g, and h be quantum states of elements in $\mathbb{F}_{2^n}$. Then, the algorithms are described as follows:

$$\begin{aligned} \texttt{ADD}(f, g) &= (f, f + g), \qquad \texttt{SQUARE}(f) = f^2, \\ \texttt{MODMULT}(f, g, h) &= (f, g, h + fg). \end{aligned}$$

Similarly, we also use a $\texttt{SQUARE}^{-1}$ operation given by

$$\texttt{SQUARE}^{-1}(f^2) = f.$$

Here, ADD, SQUARE, and SQUARE^{-1} are based on only CNOT gates. Specifically, the number of CNOT gates are n for ADD, and at most $n^2 - n$ for SQUARE or SQUARE^{-1}. In contrast, MODMULT requires not only CNOT gates but also TOF gates. Throughout the paper, ADD and MODMULT may take only specific inputs. Let $\mathbf{0}$ denote a quantum state of a zero element in $\mathbb{F}_{2^n}$. Then, when we set $g = \mathbf{0}$ as the input of ADD, given f and $\texttt{ADD}(f, \mathbf{0}) = (f, f)$ copy f to a new n-qubit register. Similarly, when we set $h = \mathbf{0}$ as the input of MODMULT, given f, g and $\texttt{MODMULT}(f, g, \mathbf{0}) = (f, g, fg)$ writes fg in a new n-qubit register.

3 FLT-based Inversion

In this section, we review previous FLT-based inversion algorithms. In Section 3.1, we briefly explain Itoh and Tsujii's classical FLT-based inversion [IT88]. Then, in Sections 3.2 and 3.3, we review Putranto et al.'s [PWLK22] and Banegas et al.'s [BBvHL20] quantum FLT-based inversion algorithm.

3.1 Classical FLT-based Inversion

Let f be an element of $\mathbb{F}_{2^n}^*$. For simplicity, we use a notation

$$\langle \alpha \rangle \coloneqq f^\alpha$$

hereafter. The task of inversion is computing $\langle -1 \rangle$ from $\langle 1 \rangle$. Based on the extended Fermat's little theorem, the FLT-based inversion method performs inversion by computing $\langle 2^n - 2 \rangle = \langle -1 \rangle$. For this purpose, we use the following three relations:

$$\langle 2^{2^{k-1}} - 1 \rangle \times \langle 2^{2^{k-1}} - 1 \rangle^{2^{2^{k-1}}} = \langle 2^{2^k} - 1 \rangle, \tag{1}$$

$$\langle 2^\alpha - 1 \rangle^{2^\beta} \times \langle 2^\beta - 1 \rangle = \langle 2^{\alpha+\beta} - 1 \rangle, \tag{2}$$

$$\langle 2^{n-1} - 1 \rangle^2 = \langle 2^n - 2 \rangle. \tag{3}$$

Let t denote the Hamming weight of $n-1$ in binary. Then, we have $n-1 = \sum_{s=1}^{t} 2^{k_s}$ with $k_1 = \lfloor \log_2(n-1) \rfloor > k_2 > \cdots > k_t \geq 0$. The FLT-based inversion consists of three steps as follows.

First Step: The step computes $\langle 2^{2^1} - 1 \rangle, \langle 2^{2^2} - 1 \rangle, \ldots, \langle 2^{2^{k_1}} - 1 \rangle$ from $\langle 2^{2^0} - 1 \rangle = \langle 1 \rangle$. For this purpose, we apply (1) to $\langle 2^{2^{i-1}} - 1 \rangle$ and obtain $\langle 2^{2^i} - 1 \rangle$ for $i = 1, 2, \ldots, k_1$ sequentially.

Second Step: The step computes $\langle 2^{n-1} - 1 \rangle$ from $\langle 2^{2^{k_1}} - 1 \rangle, \langle 2^{2^{k_2}} - 1 \rangle, \ldots, \langle 2^{2^{k_t}} - 1 \rangle$ which were computed in the first step. For this purpose, we apply (2) to $\langle 2^{2^{k_{i+1}}} - 1 \rangle$ and $\langle 2^{\sum_{s=1}^{i} 2^{k_s}} - 1 \rangle$, and obtain $\langle 2^{2^{k_{i+1}}} - 1 \rangle \times \langle 2^{\sum_{s=1}^{i} 2^{k_s}} - 1 \rangle^{2^{2^{k_{i+1}}}} = \langle 2^{\sum_{s=1}^{i+1} 2^{k_s}} - 1 \rangle$ for $i = 1, 2, \ldots, t-1$ sequentially, where the last output is $\langle 2^{\sum_{s=1}^{t} 2^{k_s}} - 1 \rangle = \langle 2^{n-1} - 1 \rangle$.

Third Step: The step applies (3) to $\langle 2^{n-1} - 1 \rangle$ and obtain $\langle 2^n - 2 \rangle = \langle -1 \rangle$.

Since the procedure may be complicated at the first glance, we describe the above procedure in a case of $n = 163$. In this case, it holds that $n - 1 = 162 = 2^7 + 2^5 + 2^1$, where $t = 3$ and $k_1 = 7, k_2 = 5, k_3 = 1$. In the first step, we compute $\langle 2^{2^1} - 1 \rangle, \langle 2^{2^2} - 1 \rangle, \ldots, \langle 2^{2^7} - 1 \rangle$ from $\langle 2^{2^0} - 1 \rangle = \langle 1 \rangle$. For this purpose, we apply (1) to $\langle 2^{2^0} - 1 \rangle, \langle 2^{2^1} - 1 \rangle, \ldots, \langle 2^{2^6} - 1 \rangle$ and obtain $\langle 2^{2^1} - 1 \rangle, \langle 2^{2^2} - 1 \rangle, \ldots, \langle 2^{2^7} - 1 \rangle$, respectively. In the second step, we compute $\langle 2^{2^7+2^5} - 1 \rangle$ and $\langle 2^{2^7+2^5+2^1} - 1 \rangle = \langle 2^{162} - 1 \rangle$ from $\langle 2^{2^7} - 1 \rangle, \langle 2^{2^5} - 1 \rangle, \langle 2^{2^1} - 1 \rangle$. For this purpose, we first apply (2) to $\langle 2^{2^7} - 1 \rangle$ and $\langle 2^{2^5} - 1 \rangle$, and obtain $\langle 2^{2^7} - 1 \rangle^{2^{2^5}} \times \langle 2^{2^5} - 1 \rangle = \langle 2^{2^7+2^5} - 1 \rangle$. Then, we apply (2) to $\langle 2^{2^7+2^5} - 1 \rangle$ and $\langle 2^{2^1} - 1 \rangle$, and obtain $\langle 2^{2^7+2^5} - 1 \rangle^{2^{2^1}} \times \langle 2^{2^1} - 1 \rangle = \langle 2^{2^7+2^5+2^1} - 1 \rangle = \langle 2^{162} - 1 \rangle$. Finally, in the third step, we apply (3) to $\langle 2^{162} - 1 \rangle$ and obtain $\langle 2^{163} - 2 \rangle = \langle -1 \rangle$.

3.2 Putranto et al.'s Quantum FLT-based Inversion Algorithm

We explain Putranto et al.'s quantum FLT-based inversion algorithm [PWLK22] that is a simple quantum translation of Itoh and Tsujii's classical FLT-based inversion [IT88]. Putranto et al.'s algorithm is given in Algorithm 1. The algorithm saves the number of TOF gates by using SQUARE which uses only CNOT gates. Here, we explain the main parts of Algorithm 1, i.e., the loop from line 1 to 5 and from line 6 to 9.

Algorithm 1 Putranto et al.'s quantum FLT-based inversion algorithm

Input: An irreducible polynomial $m(x) \in \mathbb{F}_{2^n}^*$ of degree n, $k_1, \ldots, k_t$ as explained in Section 3.1, $k_p = 2k_1 + t - 1$, a polynomial $f_0 = f \in \mathbb{F}_{2^n}^*$ of degree up to $n-1$, polynomials $f_1, \cdots, f_{k_p}$ initialized to an all-$|0\rangle$ state.

Output: $f_{k_p} = f^{-1}$

1: **for** $i = 1, \ldots, k_1$ **do**
2: ADD($f_{2(i-1)}, f_{2(i-1)+1}$)
3: **for** $j = 1, \ldots, 2^{i-1}$ **do**
4: SQUARE($f_{2(i-1)+1}$)
5: MODMULT($f_{2(i-1)}, f_{2(i-1)+1}, f_{2(i-1)+2}$)
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($f_{2k_{i+1}}, f_{2k_1+i-1}, f_{2k_1+i}$)
10: **if** $t = 1$ **then**
11: swap(f_{k_1}, f_{k_p})
12: SQUARE(f_{k_p})

Loop from line 1 to 5: The loop performs the first step of Itoh and Tsujii's FLT-based inversion. Specifically, for $i = 1, 2, \ldots, k_1$, the i-th loop takes $f_{2(i-1)} = \langle 2^{2^{i-1}} - 1\rangle$ as input and outputs $\langle 2^{2^i} - 1\rangle$ by applying (1). For this purpose, we first apply `ADD` to copy $f_{2(i-1)} = \langle 2^{2^{i-1}} - 1\rangle$ in a new register $f_{2(i-1)+1}$. Then, we apply the `SQUARE` operation 2^{i-1} times to $f_{2(i-1)+1} = \langle 2^{2^{i-1}} - 1\rangle$ and obtain $\langle 2^{2^{i-1}} - 1\rangle^{2^{2^{i-1}}}$ in the same register. Finally, we apply `MODMULT` to $f_{2(i-1)} = \langle 2^{2^{i-1}} - 1\rangle$ and $f_{2(i-1)+1} = \langle 2^{2^{i-1}} - 1\rangle^{2^{2^{i-1}}}$, and obtain $\langle 2^{2^i} - 1\rangle$ in a new register $f_{2(i-1)+2}$. Therefore, we use the `MODMULT` operation k_1 times and new $2k_1$ registers, i.e., $2k_1 n$ qubits.

Loop from line 6 to 9: The loop performs the second step of Itoh and Tsujii's FLT-based inversion. Specifically, for $i = 1, 2, \ldots, t-1$, the i-th loop takes $f_{2k_{i+1}} = \langle 2^{2^{k_{i+1}}} - 1\rangle$ and $f_{2k_1+i-1} = \langle 2^{\sum_{s=1}^{i} 2^{k_s}} - 1\rangle$ as input, and outputs $\langle 2^{\sum_{s=1}^{i+1} 2^{k_s}} - 1\rangle$ by applying (2). For this purpose, we first apply the `SQUARE` operation $2^{k_{i+1}}$ times to $f_{2k_1+i-1} = \langle 2^{\sum_{s=1}^{i} 2^{k_s}} - 1\rangle$ and obtain $\langle 2^{\sum_{s=1}^{i} 2^{k_s}} - 1\rangle^{2^{2^{k_{i+1}}}}$ in the same register. Then, we apply `MODMULT` to $f_{2k_{i+1}} = \langle 2^{2^{k_{i+1}}} - 1\rangle$ and $f_{2k_1+i-1} = \langle 2^{\sum_{s=1}^{i} 2^{k_s}} - 1\rangle^{2^{2^{k_{i+1}}}}$, and obtain $\langle 2^{\sum_{s=1}^{i+1} 2^{k_s}} - 1\rangle$ in a new register f_{2k_1+i}. Therefore, we use `MODMULT` operation $t-1$ times and new $t-1$ registers, i.e., $(t-1)n$ qubits. We note that the last output of the loop is $f_{k_p} = \langle 2^{\sum_{s=1}^{t} 2^{k_s}} - 1\rangle = \langle 2^{n-1} - 1\rangle$.

Although we omit the detail, the line 12 performs the third step of Itoh and Tsujii's FLT-based inversion. To sum up, Algorithm 1 applies the `MODMULT` operation $k_1 + t - 1$ times and uses new $(2k_1 + t - 1)n = k_p n$ qubits.

We note that we use Algorithm 1 two times for an inversion computation each. The second operation uncomputes the ancillary qubits.

3.3 Banegas et al.'s Quantum FLT-based Inversion Algorithm

We explain Banegas et al.'s quantum FLT-based inversion algorithm [BBvHL20] that is a fewer-qubit variant of Putranto et al.'s algorithm. Banegas et al.'s algorithm is given in Algorithm 2 by clearing garbages. Algorithm 2 is similar to Algorithm 1 except the additional step in from line 6 to 8. To demonstrate the effectiveness of the step, we again focus on Algorithm 1. From line 1 to 5, for $i = 1, 2, \ldots, k_1$, the i-th loop takes $f_{2(i-1)} = \langle 2^{2^{i-1}} - 1\rangle$ as input and outputs $f_{2(i-1)} = \langle 2^{2^i} - 1\rangle$. During the computation, we also use a register $f_{2(i-1)+1}$ that results in $f_{2(i-1)+1} = \langle 2^{2^{i-1}} - 1\rangle^{2^{2^{i-1}}}$. A point to note is that the register $f_{2(i-1)+1}$ is used only for the computation and remains as it is. Therefore, Algorithm 2 initializes the register and

Algorithm 2 Banegas et al.'s quantum FLT-based inversion algorithm

Input: An irreducible polynomial $m(x) \in \mathbb{F}_{2^n}^*$ of degree n, $k_1, \ldots, k_t$ as explained in Section 3.1, $k_b = \max(k_1 + t - 1, k_1 + 1)$, a polynomial $f_0 = f \in \mathbb{F}_{2^n}^*$ of degree up to $n-1$, polynomials $f_1, \cdots, f_{k_b}$ initialized to an all-$|0\rangle$ state.

Output: $f_{k_b} = f^{-1}$

1: **for** $i = 1, \ldots, k_1$ **do**
2: ADD(f_{i-1}, f_{k_b})
3: **for** $j = 1, \ldots, 2^{i-1}$ **do**
4: SQUARE(f_{k_b})
5: MODMULT(f_{i-1}, f_{k_b}, f_i)
6: **for** $j = 1, \ldots, 2^{i-1}$ **do**
7: SQUARE$^{-1}(f_{k_b})$
8: ADD(f_{i-1}, f_{k_b})
9: **for** $i = 1, \ldots, t-1$ **do**
10: **for** $j = 1, \ldots, 2^{k_{i+1}}$ **do**
11: SQUARE(f_{k_1+i-1})
12: MODMULT$(f_{k_{i+1}}, f_{k_1+i-1}, f_{k_1+i})$
13: **if** $t = 1$ **then**
14: swap(f_{k_1}, f_{k_b})
15: SQUARE(f_{k_b})

successfully reduce the qubits by applying $\texttt{SQUARE}^{-1}$. On the other hand, due to the additional procedure, Algorithm 2 requires larger depth and more CNOT gates than Algorithm 1. We explain the loop from line 1 to line 8 in Algorithm 2 below.

Loop from line 1 to 8: The loop performs the same step of the loop from line 1 to 5 in Algorithm 1. In particular, $f_{k_{i-1}}, f_{k_b}$, and f_i in Algorithm 2 play the same role as $f_{k_{2(i-1)}}, f_{2(i-1)+1}$, and $f_{2(i-1)+2}$ in Algorithm 1, respectively. Thus, the loop takes $f_{i-1} = \langle 2^{2^{i-1}} - 1\rangle$ as input and results in $f_{i-1} = \langle 2^{2^{i-1}} - 1\rangle$, $f_{k_b} = \langle 2^{2^{i-1}} - 1\rangle^{2^{2^{i-1}}}$, and $f_i = \langle 2^{2^i} - 1\rangle$ by line 5. Then, we apply the $\texttt{SQUARE}^{-1}$ operation 2^{i-1} times to $f_{k_b} = \langle 2^{2^{i-1}} - 1\rangle^{2^{2^{i-1}}}$ and obtain $\langle 2^{2^{i-1}} - 1\rangle$ in the same register. Finally, we apply ADD to $f_{i-1} = \langle 2^{2^{i-1}} - 1\rangle$ and $f_{k_b} = \langle 2^{2^{i-1}} - 1\rangle$, and initialize f_{k_b}. Since f_{k_b} in Algorithm 2 plays the same role as $f_{2(i-1)+1}$ in Algorithm 1 for all $i = 1, 2, \ldots, k_1$, Algorithm 2 reduces $k_1 - 1$ registers, i.e., $(k_1 - 1)n$ qubits. Therefore, we use the MODMULT operation k_1 times and new $k_1 + 1$ registers, i.e., $(k_1 + 1)n$ qubits.

Although we omit the detail, f_{k_b} is also used to store the outputs of second and third steps. Thus, Algorithm 2 reduces one more register, i.e., n qubits. To sum up, Algorithm 2 applies the MODMULT operation $k_1 + t - 1$ times and use new $(k_1 + t - 1)n = k_b n$ qubits.

We repeatedly claim that we use Algorithm 2 two times in each inversion computation.

4 Our Method

In this section, we propose quantum FLT-based inversion algorithms. In Section 4.1, we review the notion of addition chain which is a core tool of our improvement. In Sections 4.2 and 4.3, we propose our basic algorithm and extended algorithm that are improvements of Putranto et al.'s algorithm [PWLK22] and Banegas et al.'s algorithm [BBvHL20], respectively.

4.1 Addition Chain

Let N and ℓ be non-negative integers. An addition chain for N of length ℓ is given by $p_0 = 1, p_1, p_2, \ldots, p_\ell = N$ with the following property:

- for all $s = 1, 2, \ldots, \ell$, there exist i and j which satisfy $0 \leq i, j < s$ and $p_s = p_i + p_j$.

If there are no i and j such that $i \neq j$ satisfying $p_s = p_i + p_j$, p_s should be computed by $p_s = 2p_i$ for some $0 \leq i < s$. We call such p_s a doubled term. Otherwise, we call p_s including p_0 an added term. For an addition chain $\{p_s\}_{s=0}^{\ell}$, we define two sets

$$D \coloneqq \{s \in \{1, 2, \ldots, \ell\} \mid p_s \text{ is a doubled term}\},$$
$$M \coloneqq \{s \in \{1, 2, \ldots, \ell\} \mid p_s \text{ is an added term}\},$$

such that $D \cap M = \emptyset$. We also introduce two sequences $\{a_s\}_{s=1}^{\ell}$ and $\{b_s\}_{s=1}^{\ell}$ that satisfy $p_s = p_{a_s} + p_{b_s}$ for all $1 \leq s \leq \ell$. Intuitively, the sequences indicate how each term p_s is computed. We note that the sequences may not be unique for an addition chain $\{p_s\}_{s=0}^{\ell}$.

Aw we explained in Section 1.3, there is relation between the FLT-based inversion and addition chains. In the first and second steps of Algorithms 1 and 2, we start from $\langle 2^{2^0} - 1 \rangle$ and compute $\langle 2^{2^1} - 1 \rangle, \langle 2^{2^2} - 1 \rangle, \ldots, \langle 2^{2^7} - 1 \rangle, \langle 2^{2^7+2^5} - 1 \rangle$, and $\langle 2^{2^7+2^5+2^1} - 1 \rangle = \langle 2^{2^{162}} - 1 \rangle$ when $n = 163$. Here, we focus on the exponents of 2, i.e,.,

$$\{2^0 = 1, 2^1, 2^2, \ldots, 2^7, 2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162\}.$$

We find that the sequence of numbers is an addition chain for 162. Moreover, $2^1, 2^2, \ldots, 2^7$ are doubled terms and $2^7 + 2^5, 2^7 + 2^5 + 2^1 = 162$ are added terms. In general, Algorithms 1 and 2 are based on the same addition chain for $n - 1$ following Itoh and Tsujii's FLT-based inversion. Moreover, the first $\lfloor \log_2(n-1) \rfloor$ elements excluding $2^0 = 1$ are always doubled terms and the last $t - 1$ elements are always added terms. Hereafter, we call the sequence Itoh and Tsujii's addition chain.

4.2 Basic Algorithm

We find that previous quantum FLT-based inversion algorithms [PWLK22, BBvHL20] are based on Itoh and Tsujii's addition chains that are automatically determined by the value $n - 1$. Here, we show that Putranto et al.'s algorithm [PWLK22] can use arbitrary addition chains and does not necessarily have to be specific to Itoh and Tsujii's addition chains.

At first, we introduce some properties that arbitrary addition chains inherently satisfy. These properties enable us to prove the main theorem later.

Lemma 1. *For an arbitrary addition chain $\{p'_s\}_{s=0}^{\ell}$ for N of length ℓ, there exists an addition chain $\{p_s\}_{s=0}^{\ell}$ for the same N and ℓ so that the latter addition chain satisfies following properties.*

(i) Both $\{p_s\}_{s=0}^{\ell}$ and $\{p'_s\}_{s=0}^{\ell}$ consist of the same elements although the order may not be the same. In other words, for all $0 < s < \ell$, there exists $0 < s' < \ell$ such that $p_s = p'_{s'}$. Specifically, $p_0 = p'_0 = 1$ and $p_\ell = p'_\ell = N$ hold.

(ii) A sequence consisting of only added terms of $\{p_s\}_{s=0}^{\ell}$ are monotonically increasing. In other words, for all $i, j \in M$ such that $i < j$, it holds that $p_i < p_j$.

(iii) An element for computing a doubled term appear just before the doubled term. In other words, for all $i \in D$, it holds that $p_i = 2p_{i-1}$.

Proof. It is clear that for an arbitrary addition chain $\{p'_s\}_{s=0}^{\ell}$ for N of length ℓ, there is a unique sequence $\{p_s\}_{s=0}^{\ell}$ that satisfy all properties (i)–(iii). What we have to show is that $\{p_s\}_{s=0}^{\ell}$ is an addition chain for N of length ℓ. Due to the property (i), $p_0 = 1$ and $p_\ell = N$ hold. We complete the proof by showing that for all $s = 1, 2, \ldots, \ell$, there exist i and j which satisfy $0 \leq i \leq j < s$ and $p_s = p_i + p_j$. If $s \in D$, it holds that $p_s = 2p_{s-1} = p_{s-1} + p_{s-1}$ due to the property (iii).

Hereafter, we consider the case of $s \in M$ such that $p_s = p_i + p_j$. To prove the claim, we show that for all $1 \leq s < v \leq \ell$, it holds that $p_s < p_v$. If the statement holds, there exist i and j which satisfy $0 \leq i \leq j < s$ and $p_s = p_i + p_j$ since $p_i < p_s$ and $p_j < p_s$ hold. If $v \in M$ holds, then it holds that $p_s < p_v$ due to the property (ii). If $v \in D$, then there exists an index $v' \in M$ such that $s \leq v' < v$ and $p_v = 2^{v-v'} p_{v'}$. Due to the property (ii), it holds that $p_s \leq p_{v'} < 2^{v-v'} p_{v'} = p_v$. Thus, we complete the proof. □

We are ready for providing the existence of quantum an FLT-based inversion algorithm that uses an arbitrary addition chain.

Theorem 1. *Let f be an element of $\mathbb{F}_{2^n}^*$ and $\{p_s\}_{s=0}^{\ell}$ be an addition chain for $n-1$ of length ℓ satisfying the properties (i)–(iii) of Lemma 1. Let d and m denote the numbers of doubled terms and added terms in $\{p_s\}_{s=0}^{\ell}$, respectively. There exists a quantum algorithm that takes $f = \langle 1 \rangle$ and $\{p_s\}_{s=0}^{\ell}$ as input and outputs $\langle 2^{n-1} - 1 \rangle$ with new $(2d + m + 1)n = (\ell + d + 1)n$ qubits and* `MODMULT` *operations ℓ times.*

We note that an algorithm given in Theorem 1 is an extension of Putranto et al.'s algorithm [PWLK22] for an arbitrary addition chain. In other words, when the algorithm takes Itoh and Tsujii's addition chain as input, then the efficiency is the same as Putranto et al.'s algorithm since it holds that $d = \lfloor \log_2(n-1) \rfloor$ and $m = t - 1$ for Itoh and Tsujii's addition chain.

Proof. In this proof, we assume $p_{a_s} \leq p_{b_s}$, where $\{a_s\}_{s=1}^{\ell}$ and $\{b_s\}_{s=1}^{\ell}$ are sequences that satisfy $p_s = p_{a_s} + p_{b_s}$ for all $1 \leq s \leq \ell$ as we introduced in Section 4.1. Hereafter, we are given $\langle 2^{p_0} - 1 \rangle = f$ and compute $\langle 2^{p_1} - 1 \rangle, \ldots, \langle 2^{p_\ell} - 1 \rangle$ sequentially. We show the proof by mathematical induction. Specifically, we show how to compute $\langle 2^{p_u} - 1 \rangle$ for $1 \leq u \leq \ell$ by assuming that $\langle 2^{p_1} - 1 \rangle, \ldots, \langle 2^{p_{u-1}} - 1 \rangle$ have been computed.

At first, we discuss the simplest case. In particular, we show how to compute $\langle 2^{p_u} - 1 \rangle$ by assuming that $\langle 2^{p_{a_u}} - 1 \rangle$ and $\langle 2^{p_{b_u}} - 1 \rangle$ are stored as they are. We divide the situation into two cases, i.e., $u \in D$ and $u \in M$, and explain separately.

Case of $u \in D$: We can compute $\langle 2^{p_u} - 1 \rangle$ in essentially the same way as in the loop from line 1 to 5 in Algorithm 1. Let $\langle 2^{p_{a_u}} - 1 \rangle$ be stored in i-th register. We first apply `ADD` to copy $\langle 2^{p_{a_u}} - 1 \rangle$ in a new j-th register. Then, we apply the `SQUARE` operation $2^{p_{a_u}}$ times to j-th register and obtain $\langle 2^{2p_{a_u}} - 2^{p_{a_u}} \rangle$ in the same register. Finally, we apply `MODMULT` to $\langle 2^{p_{a_u}} - 1 \rangle$ in the i-th register and $\langle 2^{2p_{a_u}} - 2^{p_{a_u}} \rangle$ in the j-th register, and obtain $\langle 2^{2^{2p_{a_u}}} - 1 \rangle$ in a new k-th register. Due to $u \in D$, it holds that $p_u = p_{a_u} + p_{a_u} = 2p_{a_u}$, i.e., $\langle 2^{2^{2p_{a_u}}} - 1 \rangle = \langle 2^{p_u} - 1 \rangle$. Here, we use the `MODMULT` operation once and new two registers (j-th and k-th register), i.e., $2n$ qubits.

Case of $u \in M$: We can compute $\langle 2^{p_u} - 1 \rangle$ in essentially the same way as in the loop from line 6 to 9 in Algorithm 1. Let $\langle 2^{p_{a_u}} - 1 \rangle$ and $\langle 2^{p_{b_u}} - 1 \rangle$ be stored in i-th register and j-th register, respectively. We first apply the `SQUARE` operation $2^{p_{b_u}}$ times to $\langle 2^{p_{a_u}} - 1 \rangle$ in i-th register and obtain $\langle 2^{p_{a_u}+p_{b_u}} - 2^{p_{b_u}} \rangle$ in the same register. Then, we apply `MODMULT` to $\langle 2^{p_{a_u}+p_{b_u}} - 2^{p_{b_u}} \rangle$ in the i-th register and $\langle 2^{p_{b_u}} - 1 \rangle$ in the j-th register, and obtain $\langle 2^{p_{a_u}+p_{b_u}} - 1 \rangle = \langle 2^{p_u} - 1 \rangle$ in a new k-th register. Here, we use the `MODMULT` operation once and new one register (k-th register), i.e., n qubits.

After the computation, $\langle 2^{p_{a_u}} - 1 \rangle$ is still stored as it is if $u \in D$; however, $\langle 2^{p_{a_u}} - 1 \rangle$ becomes $\langle 2^{p_{a_u}} - 1 \rangle^{2^{p_{b_u}}} = \langle 2^{p_{a_u}+p_{b_u}} - 2^{p_{b_u}} \rangle$ if $u \in M$. In other words, an assumption that $\langle 2^{p_{a_u}} - 1 \rangle$ and $\langle 2^{p_{b_u}} - 1 \rangle$ are stored as they are does not always hold. We note that the assumption always hold if $u \in D$ since $a_u = u - 1$ due to the property (iii) of Lemma 1.

Next, we show how to compute $\langle 2^{p_u} - 1 \rangle$ for $u \in M$ in general. Let c_u and d_u be non-negative integers. Then, we show how to compute $\langle 2^{p_u} - 1 \rangle$ from $\langle 2^{p_{a_u}+p_{c_u}} - 2^{p_{c_u}} \rangle$ and $\langle 2^{p_{b_u}+p_{d_u}} - 2^{p_{d_u}} \rangle$. We should consider three cases, i.e., the case of $(c_u, d_u) = (0, 0)$, the case of $c_u > 0 \wedge d_u = 0$, and the case of $d_u > 0$. When $(c_u, d_u) = (0, 0)$, we can compute $\langle 2^{p_u} - 1 \rangle$ as explained above since $\langle 2^{p_{a_u}} - 1 \rangle$ and $\langle 2^{p_{b_u}} - 1 \rangle$ are stored as they are. Hereafter, we show how to compute $\langle 2^{p_u} - 1 \rangle$ if $c_u > 0 \wedge d_u = 0$ by following the same way as the case of $(c_u, d_u) = (0, 0)$. Moreover, we show that the case of $d_u > 0$ never happens.

Case of $c_u > 0 \wedge d_u = 0$: Let $\langle 2^{p_{a_u}+p_{c_u}} - 2^{p_{c_u}} \rangle$ and $\langle 2^{p_{b_u}} - 1 \rangle$ be stored in i-th register and j-th register, respectively. We first apply the `SQUARE` operation $2^{p_{b_u}-p_{c_u}}$ times to $\langle 2^{p_{a_u}+p_{c_u}} - 2^{p_{c_u}} \rangle$ in the i-th register and obtain $\langle 2^{p_{a_u}+p_{b_u}} - 2^{p_{b_u}} \rangle$ in the same register. Then, we apply `MODMULT` to $\langle 2^{p_{a_u}+p_{b_u}} - 2^{p_{b_u}} \rangle$ in the i-th register and $\langle 2^{p_{b_u}} - 1 \rangle$ in the j-th register, and obtain $\langle 2^{p_{a_u}+p_{b_u}} - 1 \rangle = \langle 2^{p_u} - 1 \rangle$ in a new k-th register. Here, we use the `MODMULT` operation once and new one register (k-th register), i.e., n qubits.

Here, we should check that $p_{b_u} - p_{c_u} > 0$ holds. As we have described so far, $\langle 2^{p_{a_u}} - 1 \rangle$ becomes $\langle 2^{p_{a_u}+p_{c_u}} - 2^{p_{c_u}} \rangle$ when we compute $\langle 2^{p_{a_u}+p_{c_u}} - 1 \rangle$. If $p_{a_u} + p_{c_u}$ is a doubled term and $p_{a_u} = p_{c_u}$ holds, $\langle 2^{p_{a_u}} - 1 \rangle$ is still stored as they are; in other words, $c_u = 0$ holds. Thus, $p_{a_u} + p_{c_u}$ is an added term. In this case, since $\langle 2^{p_{a_u}+p_{c_u}} - 1 \rangle$ was already computed, it holds that $p_{a_u} + p_{c_u} < p_{b_u} + p_{c_u}$ due to the property (ii) of Lemma 1.

Algorithm 3 Basic algorithm

Input: An irreducible polynomial $m(x) \in \mathbb{F}_{2^n}^*$ of degree n, an addition chain $\{p_s\}_{s=0}^{\ell}$ for $n-1$ of length ℓ (composed of d doubled terms and m added terms) and related $\{a_s\}_{s=1}^{\ell}, \{b_s\}_{s=1}^{\ell}, \{Q_s\}_{s=1}^{\ell}$, a polynomial $g_0 = f \in \mathbb{F}_{2^n}^*$ of degree up to $n-1$, polynomials $g_1, \ldots, g_{d+m}$ initialized to an all-$|0\rangle$ state.
Output: $g_{d+m} = f^{2^n-2}$

```
dcount ← 0
for s = 1, ..., d + m do
  if s ∈ D then
    ADD(g_{a_s}, h_{dcount})
    for i = 1, ..., Q_s do
      SQUARE(h_{dcount})
    MODMULT(g_{a_s}, h_{dcount}, g_s)
    dcount ← dcount + 1
  else {s ∈ M}
    for i = 1, ..., Q_s do
      SQUARE(g_{a_s})
    MODMULT(g_{a_s}, g_{b_s}, g_s)
SQUARE(g_{d+m})
```

Case of $d_u > 0$: As we have described so far, $\langle 2^{p_{b_u}} - 1\rangle$ becomes $\langle 2^{p_{b_u}+p_{d_u}} - 2^{p_{d_u}}\rangle$ when we compute $\langle 2^{p_{b_u}+p_{d_u}} - 1\rangle$. Let u' be an index such that $p_{u'} = p_{b_u} + p_{d_u}$. Then, it hold that $a_{u'} = b_u$ and $b_{u'} = d_u$. Since $\langle 2^{p_{b_u}+p_{d_u}} - 1\rangle$ was already computed, it holds that $p_{b_u} + p_{d_u} < p_{a_u} + p_{b_u} \Leftrightarrow p_{d_u} < p_{a_u}$ due to the property (ii) of Lemma 1. Moreover, as we mentioned at the beginning of this proof, $p_{a_s} \leq p_{b_s}$ holds for all s. Thus, it hold that $p_{a_u} \leq p_{b_u} = p_{a_{u'}} \leq d_u = p_{b_{u'}}$. This is the contradiction. Thus, $d_u > 0$ never happens.

To sum up, when we compute $\langle 2^{p_u} - 1\rangle$, we always apply `MODMULT` once and use $2n$ and n new qubits if $u \in D$ and $u \in M$, respectively. Therefore, we apply `MODMULT` operation $d + m = \ell$ times and use new $(2d + m + 1)n$ qubits. □

We describe our basic algorithm based on Theorem 1 in Algorithm 3. We note that Algorithm 3 takes not only an addition chain $\{p_s\}_{s=0}^{\ell}$ but also $\{a_s\}_{s=1}^{\ell}$, $\{b_s\}_{s=1}^{\ell}$, and $\{Q_s\}_{s=1}^{\ell}$ as input. Here, we explain the roles of the additional inputs. We proved Theorem 1 by assuming $p_{a_s} < p_{b_s}$; however, the algorithm becomes less efficient since we apply `SQUARE` operation $2^{p_{b_s}}$ times to $\langle 2^{p_{a_s}} - 1\rangle$ and obtain $\langle 2^{p_{a_s}+p_{b_s}} - 2^{p_{b_s}}\rangle$ for computing $\langle 2^{p_{a_s}+p_{b_s}} - 1\rangle$ from $\langle 2^{p_{a_s}+p_{b_s}} - 2^{p_{b_s}}\rangle$ and $\langle 2^{p_{b_s}} - 1\rangle$. In other words, we can save the number of `SQUARE` if we apply the operation $2^{p_{a_s}}$ times to $\langle 2^{p_{b_s}} - 1\rangle$ and obtain $\langle 2^{p_{a_s}+p_{b_s}} - 2^{p_{a_s}}\rangle$ for computing $\langle 2^{p_{a_s}+p_{b_s}} - 1\rangle$ from $\langle 2^{p_{a_s}+p_{b_s}} - 2^{p_{a_s}}\rangle$ and $\langle 2^{p_{a_s}} - 1\rangle$. Therefore, the restriction $p_{a_s} < p_{b_s}$ results in more CNOT gates and larger depth. However, the restriction is required for proving the existence of a quantum algorithm for arbitrary addition chains. In contrast, we focus on specific binary curves recommended by NIST. Thus, Algorithm 3 takes $\{a_s\}_{s=1}^{\ell}$ and $\{b_s\}_{s=1}^{\ell}$ as input, where it is interesting that $p_{a_s} \geq p_{b_s}$ hold for most s. The last input $\{Q_s\}_{s=1}^{\ell}$ describes the numbers of `SQUARE` to be applied in each step.

4.3 Extended Algorithm

As we explained in Section 3.3, Banegas et al. [BBvHL20] reduced the required qubits from Putranto et al.'s algorithm [PWLK22] by clearing garbages and sacrificing the number of CNOT gates and the depth. In the same way, we can reduce required qubits of our Algorithm 3 as described in Algorithm 4. What is more, we introduce a trade-off parameter L, where Algorithm 4 with the larger L requires fewer qubits, more CNOT, and larger depth. We can further save one register, i.e, n qubits, to store the output $\langle 2^n - 2\rangle$ if the last element $n-1$ of an addition chain is an added term, where we can find such an addition chain for NIST recommended curves for all n. The performance of Algorithm 4 is described as follows.

Algorithm 4 Extended algorithm

Input: An irreducible polynomial $m(x) \in \mathbb{F}_{2^n}^*$ of degree n, an addition chain $\{p_s\}_{s=0}^{\ell}$ for $n-1$ of length ℓ (composed of d doubled terms and m added terms) and related $\{a_s\}_{s=1}^{\ell}, \{b_s\}_{s=1}^{\ell}, \{Q_s\}_{s=1}^{\ell}, \{c\ell_t\}_{t=0}^{d}$, a polynomial $g_0 = f \in \mathbb{F}_{2^n}^*$ of degree up to $n-1$, polynomials $g_1, \ldots, g_{d+m-1}, h_0, \ldots, h_{d-L-1}$ initialized to an all-$|0\rangle$ state, an array `pl` that members are initialized to -1.

Output: $h_{\overline{d}} = f^{2^n-2}$

1: $dcount \leftarrow 0$
2: **for** $s = 1, \ldots, d+m$ **do**
3: **if** $s \in D$ **then**
4: **if** $\mathrm{pl}[\overline{dcount}] \neq -1$ **then**
5: GARBAGECLEAR$(c\ell_{dcount}, \mathrm{pl}[\overline{dcount}], \overline{dcount})$
6: ADD$(g_{a_s}, h_{\overline{dcount}})$
7: **for** $i = 1, \ldots, Q_s$ **do**
8: SQUARE$(h_{\overline{dcount}})$
9: MODMULT$(g_{a_s}, h_{\overline{dcount}}, g_s)$
10: $\mathrm{pl}[\overline{dcount}] \leftarrow a_s$
11: $dcount \leftarrow dcount + 1$
12: **else** $\{s \in M\}$
13: **for** $i = 1, \ldots, Q_s$ **do**
14: SQUARE(g_{a_s})
15: MODMULT(g_{a_s}, g_{b_s}, g_s)
16: **if** $\mathrm{pl}[\overline{d}] \neq -1$ **then**
17: GARBAGECLEAR$(c\ell_d, \mathrm{pl}[\overline{d}], \overline{d})$
18: **for** $i = 1, \ldots, Q_{d+m}$ **do**
19: SQUARE$(g_{a_{d+m}})$
20: MODMULT$(g_{a_{d+m}}, g_{b_{d+m}}, h_{\overline{d}})$
21: SQUARE$(h_{\overline{d}})$

Theorem 2. *Let f be an element of $\mathbb{F}_{2^n}^*$ and $\{p_s\}_{s=0}^{\ell}$ be an addition chain for $n-1$ of length ℓ satisfying the properties (i)–(iii) of Lemma 1 and $\ell \in M$. Let d and m denote the numbers of doubled terms and added terms in $\{p_s\}_{s=0}^{\ell}$, respectively. There exists a quantum algorithm that takes $f = \langle 1 \rangle$, $\{p_s\}_{s=0}^{\ell}$, and $L \in \{0, 1, \ldots, d-1\}$ as input and outputs $\langle 2^{n-1} - 1 \rangle$ with new $(2d + m - L)n = (\ell + d - L)n$ qubits and* `MODMULT` *operations ℓ times.*

Algorithm 4 takes `pl` and $\{c\ell_t\}_{t=0}^{d}$ as addition input. An array `pl` has $d-L$ members, and stores indices of the polynomials g which are used for `ADD` to clear garbages. The sequence $\{c\ell_t\}_{s=0}^{t}$ describe the number of times to applying `SQUARE` or $\texttt{SQUARE}^{-1}$ for clearing garbages. More precisely, we apply `SQUARE` $c\ell_t$ times if $c\ell_t > 0$ and $\texttt{SQUARE}^{-1}$ $-c\ell_t$ times if $c\ell_t < 0$. We set $c\ell_0 = 0$ and $\overline{x} \coloneqq x \mod (d-L)$. Garbages are stored in $h_0, \ldots, h_{d-L-1}$ in turn and clearing is performed by initializing them to 0 from h_0 to h_{d-L-1} in this order. We describe the algorithm for clearing garbages in Algorithm 5. We note that the case of $L = 0$ is different from basic algorithm since clearing to store $\langle 2^{n-1} - 1 \rangle$ is still performed. When $L = d-1$, we only prepare a polynomial h_0 for garbages, however, initializing is performed whenever we compute $\langle 2^{p_s} - 1 \rangle$, where $s \in D$. In general, each time L increases by 1, we apply an additional clearing, that implicates the trade-off between the number of qubits and the number of CNOT gates, and the depth.

Algorithm 3 and Algorithm 4 are also applied two times for an inversion computation each. We uncompute the ancillary qubits by the second operation.

Algorithm 5 GARBAGECLEAR(c, k, ℓ)

Input: Integers c, k, ℓ.
1: **if** $c > 0$ **then**
2: **for** $i = 1, \ldots, c$ **do**
3: SQUARE(h_ℓ)
4: **if** $c < 0$ **then**
5: **for** $i = 1, \ldots, -c$ **do**
6: SQUARE$^{-1}(h_\ell)$
7: ADD(g_k, h_ℓ)

Table 1: d, m, ℓ of Itoh and Tsujii's addition chains

n	163	233	283	571
d	7	7	8	9
m	2	3	3	4
ℓ	9	10	11	13

Table 2: d, m, ℓ of our choice of addition chains

n	163	233	283	571
d	5	4	3	4
m	4	6	8	8
ℓ	9	10	11	12

Table 3: Our choice of addition chains $\{p_s\}_{s=0}^{\ell}$ with the sequences $\{a_s\}_{s=1}^{\ell}$, $\{b_s\}_{s=1}^{\ell}$, and $\{Q_s\}_{s=1}^{\ell}$

n	sequences		
163	p_s	:	$1, 2, 4, 8, 16, 32, 33, 65, 97, 162$
	a_s	:	$0, 1, 2, 3, 4, 5, 5, 7, 8$
	b_s	:	$0, 1, 2, 3, 4, 0, 6, 5, 7$
	Q_s	:	$1, 2, 4, 8, 16, 1, 32, 32, 65$
233	p_s	:	$1, 2, 4, 8, 16, 24, 40, 56, 96, 136, 232$
	a_s	:	$0, 1, 2, 3, 4, 4, 4, 7, 8, 8$
	b_s	:	$0, 1, 2, 3, 3, 5, 6, 6, 6, 9$
	Q_s	:	$1, 2, 4, 8, 8, 16, 16, 40, 40, 96$
283	p_s	:	$1, 2, 4, 6, 12, 18, 30, 48, 78, 126, 204, 282$
	a_s	:	$0, 1, 2, 3, 4, 4, 6, 6, 8, 8, 8$
	b_s	:	$0, 1, 1, 3, 3, 5, 5, 7, 7, 9, 10$
	Q_s	:	$1, 2, 2, 6, 6, 12, 18, 30, 48, 78, 78$
571	p_s	:	$1, 2, 4, 8, 16, 18, 34, 50, 84, 134, 218, 352, 570$
	a_s	:	$0, 1, 2, 3, 4, 4, 4, 7, 7, 9, 9, 11$
	b_s	:	$0, 1, 2, 3, 1, 5, 6, 6, 8, 8, 10, 10$
	Q_s	:	$1, 2, 4, 8, 2, 16, 16, 34, 50, 84, 134, 218$

5 Comparison

In this section, we compare our proposed quantum FLT-based inversion algorithms with previous ones [PWLK22, BBvHL20]. In Section 5.1, we find addition chains for our algorithms. In Section 5.2, we compare the quantum resources for computing inversion. In Section 5.3, we show the effectiveness of the trade-off parameter L of our extended algorithm. In Section 5.4, we compare the quantum resources for point addition and Shor's algorithm.

Difference from Preliminary Version. As mentioned in Section1.2, we use quantum multiplication by Hoof [Igg19] in [TT23], however, we use one by Kim et al. [KKKH22] in this version. Therefore, we update the number of quantum resources in Tables and Figures by Kim et al.'s multiplication.

Table 4: Comparison of the number of TOF gates, qubits, and CNOT gates and the depth in an inversion between ours and prior work

n	basic algorithm				extended algorithm			
	TOF	qubits	CNOT	depth	TOF	qubits	CNOT	depth
163	18,848	2,771	1,557,528	300,920	18,848	1,956	1,579,944	310,830
233	30,261	3,961	3,345,540	434,995	30,261	3,029	3,353,750	437,747
283	41,032	4,811	5,489,296	837,096	41,032	3,962	5,502,090	840,612
571	95,325	10,849	23,458,648	3,433,263	95,325	8,565	23,514,068	3,456,469
n	PWLK22-FLT				BBHL21-FLT			
	TOF	qubits	CNOT	depth	TOF	qubits	CNOT	depth
163	18,848	3,097	1,558,180	300,924	18,848	1,956	1,601,716	342,516
233	30,261	4,660	3,346,938	435,001	30,261	3,029	3,374,430	459,709
283	41,032	6,226	5,492,126	837,106	41,032	3,962	5,644,678	985,710
571	102,951	14,275	25,189,566	3,556,815	102,951	9,136	26,043,772	4,401,901
n	BBHL21-GCD							
	TOF	qubits	CNOT	depth				
163	438,766	1,156	414,586	510,628				
233	823,095	1,646	834,256	992,766				
283	1,194,498	1,997	1,222,600	1,449,098				
571	4,434,315	4,014	4,857,244	5,602,181				

5.1 Our Choice of Addition Chains

As we showed in Theorems 1 and 2, the quantum resource of FLT-based inversion depends on d, m, ℓ of addition chain. Table 1 summarizes d, m, ℓ Itoh and Tsujii's addition chain for all n recommended by NIST. We find addition chains for all n in order of priority the number of TOF and qubits. In other words, we first find addition chains with the minimum length ℓ, then find the one with minimum doubled terms d among them. Table 2 summarizes d, m, ℓ our choice of addition chains and Table 3 summarizes the concrete addition chains $\{p_s\}_{s=0}^{\ell}$ with the sequences $\{a_s\}_{s=1}^{\ell}$, $\{b_s\}_{s=1}^{\ell}$, and $\{Q_s\}_{s=1}^{\ell}$ which are input of our algorithms. We can find addition chains with shorter length ℓ for $n = 571$. Moreover, we can find addition chains with fewer doubled terms d for all n. Our choice of addition chains work well with our algorithms. Indeed, we can save CNOT gates since $p_{a_s} \geq p_{b_s}$ holds for most s as we discussed at the end of Section 4.2. Similarly, we can save one register for Algorithm 4 since $n-1$ is an added term as we discussed in Section 4.3.

5.2 Comparison in a Quantum Inversion Computation

Table 4 compares quantum resources among the following algorithms:

- basic algorithm: our proposed Algorithm 3
- extended algorithm: our proposed Algorithm 4 for $L = d - 1$
- PWLK22-FLT: Putranto et al.'s FLT-based algorithm
- BBHL21-FLT: Banegas et al.'s FLT-based algorithm
- BBHL21-GCD: Banegas et al.'s GCD-based algorithm

in terms of the number of TOF, qubits, CNOT, and depth.

We compare the quantum resources for computing $h + gf^{-1}$ from f, g, h with two inversions and one modular multiplication. Here, the depth of `ADD` is 1. We calculate the number of CNOT gates and the upper bound of the depth of `SQUARE` by using LUP decomposition which Banegas et al.'s [BBvHL20] used. The number of TOF gates and CNOT gates and the upper bound of the depth of `MODMULT` are given by

Table 5: Quantum resources of extended algorithm in each L

(a) $n = 163$

		qubits	CNOT	depth
basic		2,771	1,557,528	300,920
L	0	2,608	1,558,514	300,920
	1	2,445	1,560,486	301,584
	2	2,282	1,563,452	302,906
	3	2,119	1,569,058	305,548
	4	1,956	1,579,944	310,830

(b) $n = 233$

		qubits	CNOT	depth
basic		3,961	3,345,540	434,995
L	0	3,728	3,346,398	434,995
	1	3,495	3,348,114	435,391
	2	3,262	3,350,148	436,177
	3	3,029	3,353,750	437,747

(c) $n = 283$

		qubits	CNOT	depth
basic		4,811	5,489,296	837,096
L	0	4,528	5,491,032	837,096
	1	4,245	5,494,504	838,270
	2	3,962	5,502,090	840,612

(d) $n = 571$

		qubits	CNOT	depth
basic		10,849	23,458,648	3,433,263
L	0	10,278	23,463,104	3,433,263
	1	9,707	23,472,016	3,436,581
	2	9,136	23,486,414	3,443,211
	3	8,565	23,514,068	3,456,469

Hoof [Igg19]. We also calculate the depth considering parallel computation by ourselves, although we do not describe it in detail. However, since paralleling is not complete, the depth is upper bound in each case.

As we described in Sections 4.2 and 4.3, our algorithms achieve the same performance when we use Itoh and Tsujii's addition chain. However, we find better addition chains with smaller ℓ and/or d for all n as we claimed in Section 5.1. Thus, our basic and extended algorithms are strictly better than PWLK22-FLT and BBHL21-FLT, respectively. Indeed, Algorithm 3 and Algorithm 4 successfully reduce all quantum resources of PWLK22-FLT and BBHL21-FLT, respectively. Moreover, our extended algorithm achieves smaller depth than PWLK22-FLT when $n = 571$. Compared with BBHL21-GCD, although BBHL21-GCD achieves fewer qubits than our algorithms by two, our algorithms achieve much fewer TOF than BBHL21-GCD by ten.

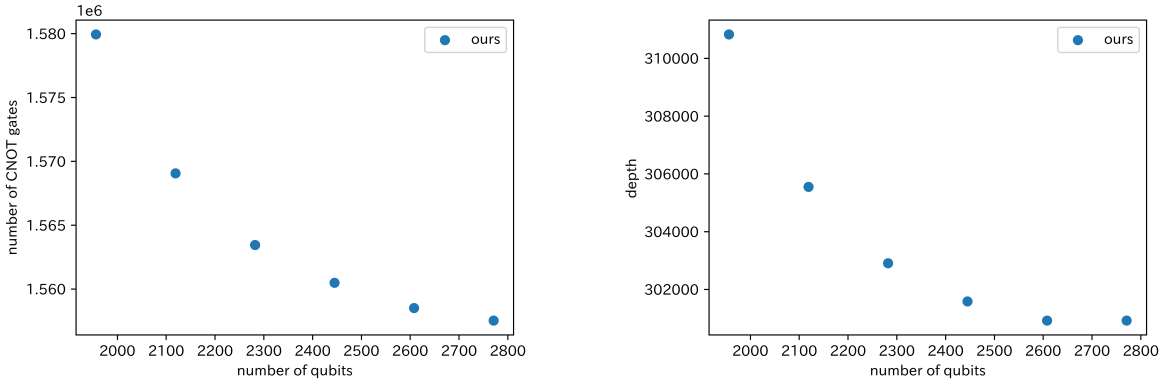
Remark 1. *In the preliminary version [TT23], addition chains given in Table 3 are different from the ones which are used for quantum resource estimation. In this version, we correctly describe addition chains used for estimation in Table 3.*

Remark 2. *After the publication of the preliminary version [TT23], Kim and Hong proposed a quantum GCD-based inversion algorithm [KH23] which achieves slightly fewer qubits and fewer TOF gates than Banegas et al.'s GCD-based inversion algorithm. However, we do not list the algorithm in Table 4 since Kim and Hong did not estimate the number of CNOT gates and the depth and the analysis of their GCD-based algorithm is out of scope of this paper. We note that Kim and Hong's GCD-based algorithm does not violate the advantage of FLT-based algorithms since the number of TOF gates of the former algorithm is close to that of Banegas et al.'s GCD-based inversion algorithm and much larger than those of FLT-based ones.*

5.3 Quantum Resources Trade-off in Extended Algorithm

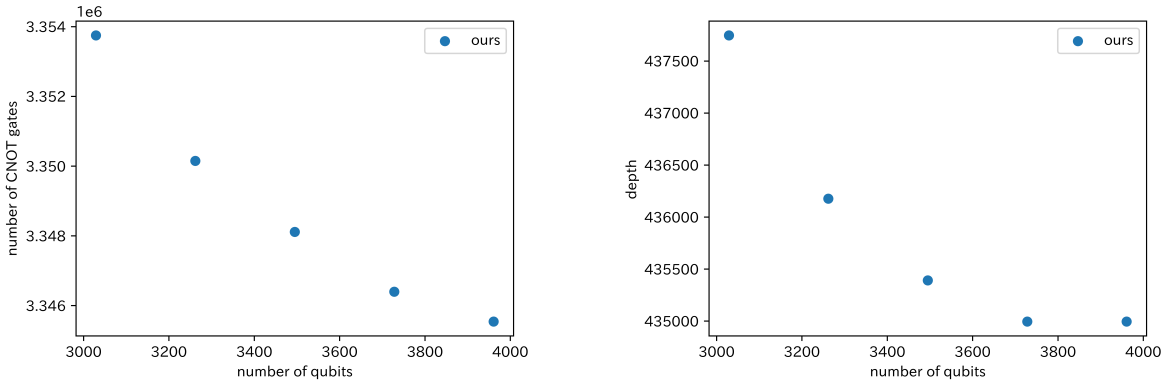
We describe the quantum resources of Algorithm 4 (extended algorithm) for all possible trade-off parameters L. As we discussed in Section 4.3, the extended algorithm for $L = 0$ is not the case of basic algorithm, but the case that only n qubits for storing the computation results are reduced. Figures 1–8 illustrate the trade-off with respect to L. Throughout the comparisons, we do not consider the number of TOF since L does not affect it. In all Figures 1–8, the round points which are placed on the rightmost represent basic algorithm, then $L = 0, 1, 2, \ldots$ from the right to the left. We can see that the number of qubits decreases and the number of CNOT gates and the depth increase for the larger L. However, we can see the same depth in the case of basic algorithm and $L = 0$ although the numbers of CNOT gates are not the same. The reason

is that we can completely parallelize clearing garbage for storing $\langle 2^{n-1}-1\rangle$. Although we may be able to parallelize other clearing procedures and will get better upper bounds of the depth, we leave it as a future work.



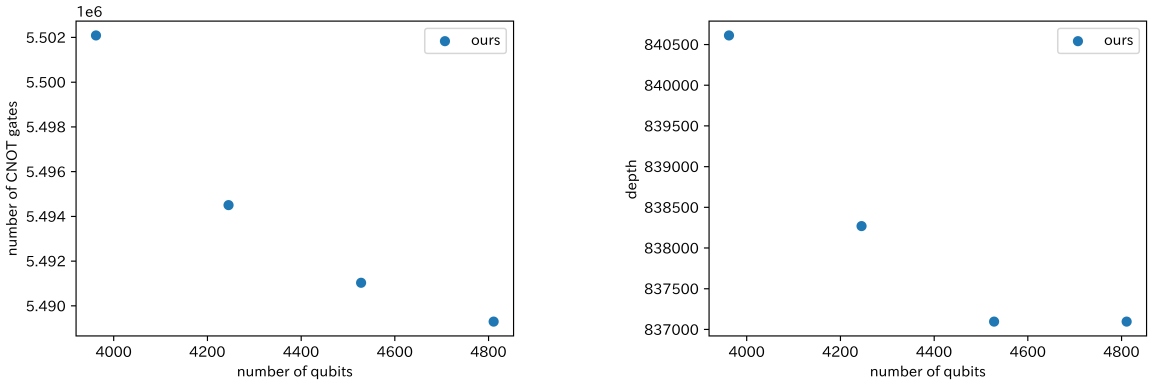
(a) qubit-CNOT **(b)** qubit-depth

Figure 1: Quantum resources trade-off in extended algorithm where $n = 163$.



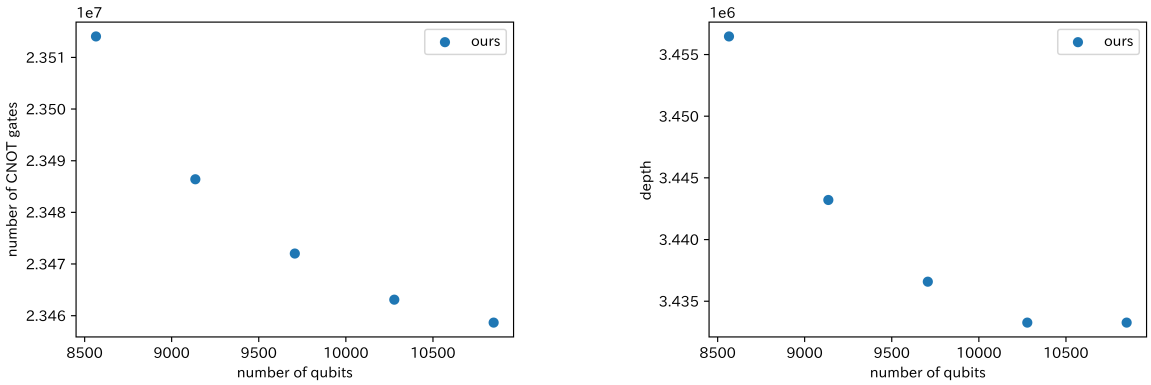
(a) qubit-CNOT **(b)** qubit-depth

Figure 2: Quantum resources trade-off in extended algorithm where $n = 233$



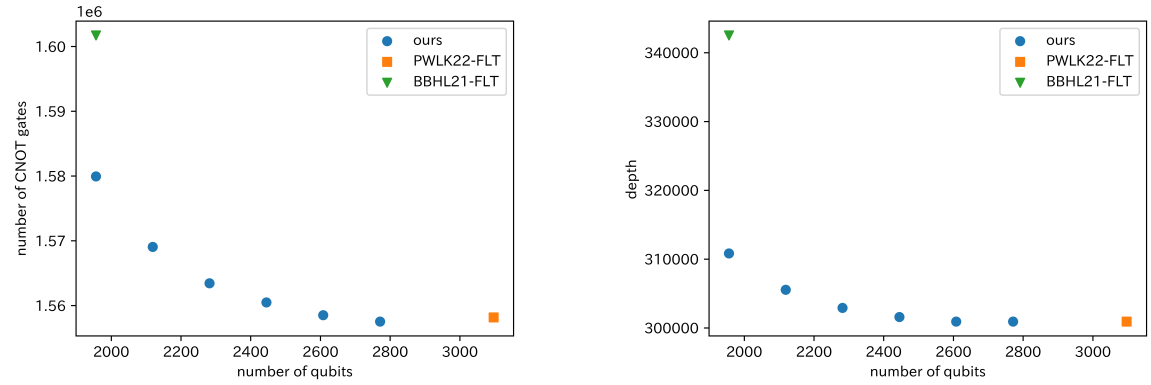
(a) qubit-CNOT **(b)** qubit-depth

Figure 3: Quantum resources trade-off in extended algorithm where $n = 283$



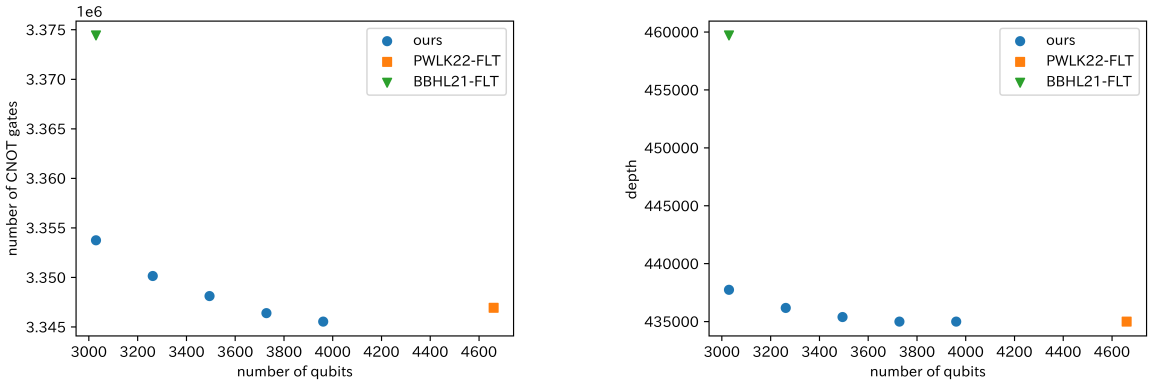
(a) qubit-CNOT **(b)** qubit-depth

Figure 4: Quantum resources trade-off in extended algorithm for $n = 571$



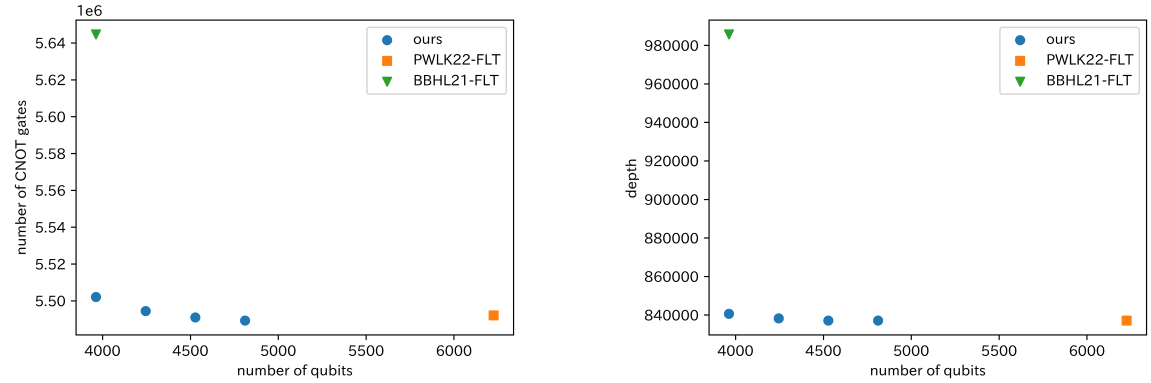
(a) qubit-CNOT **(b)** qubit-depth

Figure 5: Quantum resources trade-off in FLT-based inversion algorithms where $n = 163$



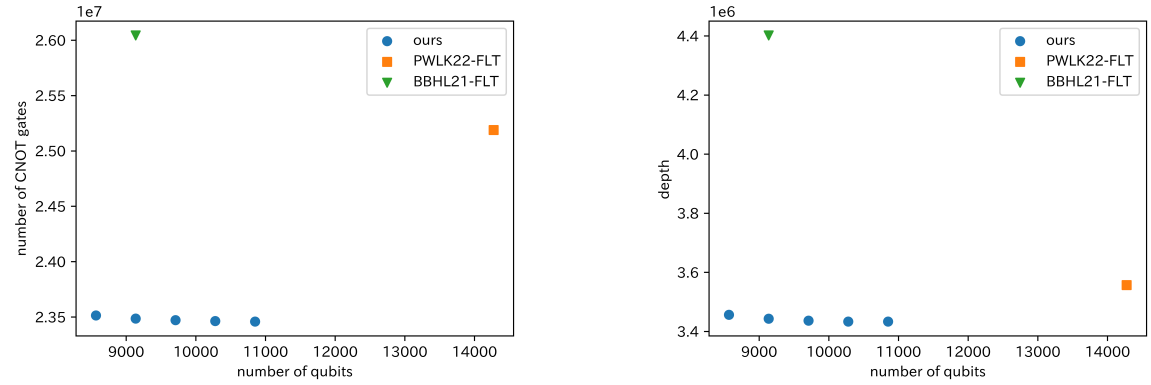
(a) qubit-CNOT **(b)** qubit-depth

Figure 6: Quantum resources trade-off in FLT-based inversion algorithms where $n = 233$



(a) qubit-CNOT **(b)** qubit-depth

Figure 7: Quantum resources trade-off in FLT-based inversion algorithms where $n = 283$



(a) qubit-CNOT **(b)** qubit-depth

Figure 8: Quantum resources trade-off in FLT-based inversion algorithms for $n = 571$

Table 6: Comparison of the number of TOF gates, qubits, and CNOT gates and the depth in Shor's algorithm between ours and prior works

n	basic algorithm			
	TOF	qubits	CNOT	depth
163	13,175,432	2,772	1,072,118,184	204,448,960
233	30,000,204	3,962	3,276,928,512	423,198,828
283	49,121,208	4,812	6,491,648,712	977,034,976
571	228,787,416	10,850	55,651,292,840	8,000,884,320
n	extended algorithm			
	TOF	qubits	CNOT	depth
163	13,175,432	1,957	1,086,823,080	210,949,920
233	30,000,204	3,030	3,284,613,072	425,774,700
283	49,121,208	3,963	6,506,182,696	981,029,152
571	228,787,416	8,566	55,778,093,800	8,053,979,648
n	PWLK22-FLT			
	TOF	qubits	CNOT	depth
163	13,175,432	3,098	1,072,545,896	204,451,584
233	30,000,204	4,661	3,278,237,040	423,204,444
283	49,121,208	6,227	6,494,863,592	977,046,336
571	246,235,704	14,276	59,611,633,224	8,283,571,296
n	BBHL21-FLT			
	TOF	qubits	CNOT	depth
163	13,175,432	1,957	1,101,105,512	231,735,936
233	30,000,204	3,030	3,303,969,552	446,331,132
283	49,121,208	3,963	6,668,162,664	1,145,860,480
571	246,235,704	9,137	61,566,056,552	10,217,128,064
n	BBHL21-GCD			
	TOF	qubits	CNOT	depth
163	288,641,640	1,157	322,348,232	342,017,408
233	772,092,828	1,647	926,366,688	945,272,484
283	1,359,458,584	1,998	1,644,682,056	1,672,269,248
571	10,156,396,536	4,015	13,091,280,488	12,963,368,704

5.4 Comparison in Shor's Algorithm

Table 4 compares quantum resources among Shor's algorithm based on our proposed FLT-based inversion algorithms and previous inversion algorithms as in Table 4 in terms of the number of TOF, qubits, CNOT, and depth. To perform $2n+2$ point additions, we use Banegas et al.'s point addition algorithm [BBvHL20]. A point addition computation contains two quantum inversion computations. We simply add the numbers in Table 4 for counting the quantum resources. Banegas et al.'s point addition algorithm contains some computations which we do not summarize. We refer to the paper [BBvHL20] for counting the number of TOF gates and CNOT gates for those computations. We consider parallel quantum computing and calculate the depth of them by ourselves. Since we use semiclassical Fourier transform [GN96] in a part of Shor's algorithm, we use only another control qubit to point additions, therefore the whole number of qubits increases by 1 from the number of qubits used in a single inversion. Table 6 shows the number of quantum resources in Shor's algorithm. Our two algorithms still perform better like a comparison in an inversion algorithm, since inversion computations occupy the largest part of a point addition computation in a view of the number of qubits and quantum gates. However, Banegas et al.'s point addition algorithm initializes λ, and this leads us to compute two inversions. If we prepare other n qubits for λ in each point addition, we can save up an inversion and the number of TOF gates and CNOT gates and the depth will be about a half of the values summarized in Table 6. Then, the number of qubits increases by $(2n+1)n$.

Table 7: Optimal window size w and the number of TOF gates for Shor's algorithm

n	basic algorithm		extended algorithm			
	w	TOF	w	TOF		
163	9	$1,781,025$	9	$1,781,025$		
233	9	$3,679,975$	9	$3,679,975$		
283	10	$5,765,145$	10	$5,765,145$		
571	11	$23,390,601$	11	$23,390,601$		
n	PWLK22-FLT		BBHL21-FLT		BBHL21-GCD	
	w	TOF	w	TOF	w	TOF
163	9	$1,781,025$	9	$1,781,025$	13	$26,303,013$
233	9	$3,679,975$	9	$3,679,975$	14	$64,402,483$
283	10	$5,765,145$	10	$5,765,145$	15	$108,252,597$
571	11	$24,976,809$	11	$24,976,809$	16	$704,590,641$

6 Windowing

We briefly explain the quantum read-only memory (QROM) in Section 6.1. Then we describe point addition using windowing by Häner et al. [HJN+20] and show the optimal window size and the number of TOF gates in each case in Section 6.2.

6.1 Quantum Read Only Memory

Quantum read-only memory (QROM) allows classical memory to be accessed by giving an index, which can be represented by superposition. Let A denote the number of data stored in QROM. We explain data as $|d_i\rangle$ for $i = 0, 1, \ldots, A-1$. Then, the QROM operation is given by

$$\mathrm{QROM}\left(\sum_{i=0}^{A-1} \alpha_i |i\rangle |S_i\rangle\right) = \sum_{i=0}^{A-1} \alpha_i |i\rangle |S_i + d_i\rangle , \tag{4}$$

where $|i\rangle$ is the index, $\alpha_i \in \mathbb{C}$ is the amplitude of $|i\rangle$, and $|S_i\rangle$ is the arbitrary quantum state. For constructing QROM, we require some quantum resources, including TOF gates. Babbush et al. [BGB+18] gave a T-depth-less QROM construction, and they made use of $2(A-1)$ TOF gates. We note that several ancillary qubits are also required for QROM, however, we do not count them because we only focus on the number of TOF gates in this section. Generally, QROM is used for skipping some quantum computations and saving the quantum gates. Therefore, we should carefully analyze the balance between the required TOF gates for QROM and the reduced TOF gates.

6.2 Point Addition Using Windowing

Quantum computation using QROM has been discussed. For example, Gidney [Gid19] explained several quantum basic arithmetics with QROM. Those ways of using QROM for looking up some data are called windowing. Häner et al. [HJN+20] indicated that point addition on elliptic curves using windowing is also possible, and Banegas et al. [BBvHL20] and Putranto et al. [PWLK22] made use of that method. We describe the outline below. Let w be an non-negative integer, and $A = 2^w$. Then, QROM stores $[i]U$ for $i = 0, 1, \ldots, 2^w - 1$, where U is a point on a binary elliptic curve. Point addition algorithm which uses `LOOKUP` to access the above QROM is explained by Banegas et al. [BBvHL20]. We can decrease the times of point addition from $2(n+1)$ to $2\lceil \frac{n+1}{w} \rceil + 1$[2], therefore the number of TOF gates decreases with increasing w. However, the number of TOF gates to construct a QROM is $2(2^w - 1)$.

Now we find an optimal w, which minimizes the number of TOF gates, about each n for each algorithm. Then, we calculate the total number of TOF gates and compare our algorithms to prior works. We show the result in Table 7. Our two algorithms and prior FLT-based algorithms bring the same results for

[2] A point addition for canceling is contained. See Banegas et al.'s paper [BBvHL20] for detailed information.

$n = 163, 233, 283$. For $n = 571$, we can see the advantage of our algorithms over PWLK22-FLT and BBHL21-FLT. However, the optimal w of BBHL21-GCD are larger than others. That is because BBHL21-GCD uses much more TOF gates than FLT-based algorithms, then windowing performs better.

7 Conclusion

In this paper, we reconsidered quantum FLT-based inversion algorithms from the viewpoint of addition chains. In purpose of analyzing the quantum resources for quantum computation, we described the number of TOF gates, qubits, and CNOT gates and the depth change depending on the addition chain. Also, we showed the existence of a quantum FLT-based inversion algorithm whose input contains an arbitrary addition chain. Then, we constructed two algorithms, basic algorithm corresponding to Putranto et al.'s algorithm and extended algorithm corresponding to Banegas et al.'s algorithm. Moreover, we reduce the number of TOF gates and the number of qubits preferentially in this order and optimized addition chains. As a result, basic algorithm and extended algorithm purely improve Putranto et al.'s algorithm and Banegas et al.'s algorithm, respectively. That stems from the existence of better addition chains, whose length is shorter, or d is smaller than Itoh and Tsujii's addition chains. We can say that our results gave a more precise estimation of quantum resources used to solve binary ECDLP with NIST recommending n.

We get some optimized addition chains that perform the same as addition chains in Table 3, therefore we can choose an addition chain that depth is also reduced the most. We have already chosen addition chains that achieve less depth, however, it is extremely hard to optimize the depth since that requests a complete analysis of parallel quantum computation. We leave it to future work. Also, there may be a better way to clear all qubits used in inversion algorithms.

References

[AJD12] R. Azarderakhsh, K. Järvinen, and V. Dimitrov. Fast inversion in gf(2^m) with normal basis using hybrid-double multipliers. *IEEE Trans. computers*, 63(4):1041–1047, 2012.

[ASK19] Mirko Amico, Zain H. Saleem, and Muir Kumph. Experimental study of shor's factoring algorithm using the ibm q experience. *Phys. Rev. A*, 100:012305, Jul 2019.

[BBvHL20] Gustavo Banegas, Daniel J. Bernstein, Iggy van Hoof, and Tanja Lange. Concrete quantum cryptanalysis of binary elliptic curves. *IACR Trans.CHES*, 2021(1):451–472, Dec. 2020.

[Bea03] S. Beauregard. Circuit for shor's algorithm using $2n + 3$ qubits. *Quantum Inf. Comput.*, 3:175–185, 2003.

[BGB+18] Ryan Babbush, Craig Gidney, Dominic W. Berry, Nathan Wiebe, Jarrod McClean, Alexandru Paler, Austin Fowler, and Hartmut Neven. Encoding electronic spectra in quantum circuits with linear t complexity. *Physical Review X*, 8(4), oct 2018.

[BGG+20] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann. Comparing the difficulty of factorization and discrete logarithm: A 240-digit experiment. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020*, volume 12171 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2020.

[BY19] Daniel J. Bernstein and Bo-Yin Yang. Fast constant-time gcd computation and modular inversion. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):340–398, 2019.

[CKA21] Alvaro Cintas Canto, Mehran Mozaffari Kermani, and Reza Azarderakhsh. Crc-based error detection constructions for flt and ita finite field inversions over GF(2^m). *IEEE Trans. VLSI Systems*, 29(5):1033–1037, 2021.

[CP13] F.Kerry Cameron and D.Gallagher Patrick. Fips pub 186-4 Digital Signature Standard (dss). In *NIST*, pages 92–101, 2013.

[DLQ+20] Zhao-Chen Duan, Jin-Peng Li, Jian Qin, Ying Yu, Yong-Heng Huo, Sven Ḧofling, Chao-Yang Lu, Nai-Le Liu, Kai Chen, and Jian-Wei Pan. Proof-of-principle demonstration of compiled Shor ' s algorithm using a quantum dot single-photon source. *Optics Express*, 28:18917–18930, 2020.

[FMMC12] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, 2012.

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

[Gid19] Craig Gidney. Windowed quantum arithmetic, 2019.

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

[GP02] Jorge Guajardo and Christof Paar. Itoh-tsujii inversion in standard basis and its application in cryptography and codes. *Designs, Codes and Cryptography*, 25(2):207–216, 2002.

[GS21] Élie Gouzien and Nicolas Sangouard. Factoring 2048-bit rsa integers in 177 days with 13 436 qubits and a multimode memory. *Phys. Rev. Lett.*, 127:140503, 2021.

[HGWC15] Jingwei Hu, Wei Guo, Jizeng Wei, and Ray CC Cheung. Fast and generic inversion architectures over GF(2^m) using modified itoh–tsujii algorithms. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(4):367–371, 2015.

[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*, pages 425–444, Cham, 2020. Springer International Publishing.

[HLH22] Jinyoung Ha, Jonghyun Lee, and Jun Heo. Resource analysis of quantum computing with noisy qubits for shor's factoring algorithms. *Quantum Inf. Process.*, 21(2):60, 2022.

[HRS17] Thomas Haener, Martin Roetteler, and Krysta M. Svore. Factoring using $2n + 2$ qubits with toffoli based modular multiplication. *Quantum Information and Computation*, 18(7-8):673–684, 2017.

[Igg19] van Hoof Iggy. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic toffoli gate count. *CoRR*, abs/1910.02849, 2019.

[IT88] Toshiya Itoh and Shigeo Tsujii. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. *Information and computation*, 78(3):171–177, 1988.

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

[KKKH22] SUNYEOP Kim, INSUNG Kim, Seonggyeom Kim, and Seokhie Hong. Toffoli gate count optimized space-efficient quantum circuit for binary field multiplication. Cryptology ePrint Archive, Paper 2022/1095, 2022.

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

[Kun05] Noboru Kunihiro. Exact analyses of computational time for factoring in quantum computers. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 88-A(1):105–111, 2005.

[LBC+12] E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O ' Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland, and J. M. Martinis. Computing prime factors with a Josephson phase qubit quantum processor. *Nature Physics*, 8:719–723s, 2012.

[LBYP07] Chao-Yang Lu, Daniel E. Browne, Tao Yang, and Jian-Wei Pan. Demonstration of a compiled version of Shor ' s quantum factoring algorithm using photonic qubits. *Phys. Rev. Lett.*, 99:250504, 2007.

[LWL+07] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Cilchrist, and A. G. White. Experimental demonstration of a compiled version of Shor's algorithm with quantum entanglement. *Phys. Rev. Lett.*, 99:250505, 2007.

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.

[MLLL+12] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien. Experimental realisation of Shor ' s quantum factoring algorithm using qubit recycling. *Nature Photon*, 6:773–776, 2012.

[MNM+16] T. Monz, D. Nigg, E. A. Martinez, M. F. Brandl, P. Schindler, R. Rines, S. X. Wang, I. L. Chuang, and R. Blatt. Realization of a scalable Shor algorithm. *Science*, 351:1068–1070, 2016.

[PMO09] A. Politi, J. C. F. Matthews, and J. L. O ' Brien. Shor ' s quantum factoring algorithm on a photonic chip. *Science*, 325:1221, 2009.

[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, Paper 2022/501, 2022.

[PZ03] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information & Computation*, 3(4), 2003.

[RHCCS05] F. Rodriguez-Henriquez, N. Cruz-Cortes, and N.A. Saqib. A fast implementation of multiplicative inversion over GF(2^m). In *ITCC'05*, volume 1, pages 574–579. IEEE, 2005.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M. Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *ASIACRYPT 2017*, pages 241–270, 2017.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sho94] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *FOCS 1994*, pages 124–134, 1994.

[SSV13] John A. Smolin, Graeme Smith, and Alexander Vargo. Oversimplifying quantum factoring. *Nature*, 499:163–165, 2013.

[TK06] Yasuhiro Takahashi and Noboru Kunihiro. A quantum circuit for shor's factoring algorithm using 2n + 2 qubits. *Quantum Inf. Comput.*, 6(2):184–192, 2006.

[TT23] Ren Taguchi and Atsushi Takayasu. Concrete quantum cryptanalysis of binary elliptic curves via addition chain. In *Topics in Cryptology – CT-RSA 2023*, pages 57–83, 2023.

[VBE96] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 54:147–153, 1996.

[VSB+01] L. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414:883–887, 2001.

[Zal98] Christof Zalka. Fast versions of shor's quantum factoring algorithm, 1998.