On the Untapped Potential of the Quantum FLT-based Inversion*

Ren Taguchi† Atsushi Takayasu‡

February 14, 2024

Abstract

Thus far, several papers estimated concrete quantum resources of Shor's algorithm for solving a binary elliptic curve discrete logarithm problem. In particular, the complexity of computing quantum inversions over a binary field $\mathbb{F}_{2^n}$ is dominant when running the algorithm, where n is a degree of a binary elliptic curve. There are two major methods for quantum inversion, i.e., the quantum GCD-based inversion and the quantum FLT-based inversion. Among them, the latter method is known to require more qubits; however, the latter one is valuable since it requires much fewer Toffoli gates and less depth. When $n = 571$, Kim-Hong's quantum GCD-based inversion algorithm (Quantum Information Processing 2023) and Taguchi-Takayasu's quantum FLT-based inversion algorithm (CT-RSA 2023) require $3,473$ qubits and $8,566$ qubits, respectively. In contrast, for the same $n = 571$, the latter algorithm requires only 2.3% of Toffoli gates and 84% of depth compared to the former one. In this paper, we modify Taguchi-Takayasu's quantum FLT-based inversion algorithm to reduce the required qubits. While Taguchi-Takayasu's FLT-based inversion algorithm takes an addition chain for $n - 1$ as input and computes a sequence whose number is the same as the length of the chain, our proposed algorithm employs an uncomputation step and stores a shorter one. As a result, our proposed algorithm requires only $3,998$ qubits for $n = 571$, which is only 15% more than Kim-Hong's GCD-based inversion algorithm. Furthermore, our proposed algorithm preserves the advantage of FLT-based inversion since it requires only 3.7% of Toffoli gates and 77% of depth compared to Kim-Hong's GCD-based inversion algorithm for $n = 571$.

*This is the full version of [TT24]. This research was in part conducted under a contract of JSPS KAKENHI Grant Numbers 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 the most widely used public-key cryptosystems in practice. The security of RSA and ECC relates to the computational complexity of the factorization problem and the elliptic curve discrete logarithm problem (ECDLP). Since there are no algorithms that solve the factorization problem/ECDLP in polynomial time, RSA and ECC are believed to be secure. In 1994, Shor [Sho94] proposed a quantum polynomial time algorithm for solving the problems. Thus, quantum resource estimates and optimized quantum circuits of the algorithm has been actively studied.

In this paper, we focus on the ECDLP over a binary elliptic curve called the binary ECDLP. Banegas et al. [BBvHL20] presented the first concrete quantum circuits for solving the problem. For this purpose, they proposed a quantum elliptic curve point addition algorithm and *two* quantum inversion algorithms over $\mathbb{F}_{2^n}$, where n is called a degree of a binary field and $n = 163, 233, 283, 571$ are recommended by NIST [CP13]. Banegas et al. estimated the concrete quantum resource, where they regarded required qubits as the main optimization target. The number of Toffoli gates is their secondary one since the gates are much more expensive than CNOT gates. Since the depth of circuits is also known to be cared as mentioned in [RNSL17], we collectively call the required qubits, Toffoli gates, and the depth the main quantum resource throughout the paper. Banegas et al.'s analysis indicates that the quantum resource varies greatly depending on which of their two quantum inversion algorithms is used. They concluded that their *GCD-based* inversion algorithm is better than their *FLT-based* one[1]since the former requires fewer qubits, while the latter requires much fewer Toffoli gates and less depth. When $n = 571$, their GCD-based and FLT-based inversion algorithms require $4,015$ and $9,137$ qubits, respectively, while the latter requires only 2.4% of Toffoli gates and 94%[2] of depth to run Shor's algorithm. A point to note is that the depth of a circuit is not an exact value but an upper bound. In general, it is technically hard to analyze fully parallel quantum computation towards minimizing the depth.

Afterward, there have been several subsequent works that updated the quantum resource estimate by presenting improved quantum inversion algorithms. Kim and Hong [KH23] proposed a GCD-based inversion algorithm that reduces all main quantum resources of Banegas et al.'s GCD-based algorithm. Although Putranto et al. [PWLK22] proposed an FLT-based inversion algorithm that reduces the depth of Banegas et al.'s FLT-based algorithm, it requires more qubits. Taguchi and Takayasu [TT23] proposed FLT-based inversion algorithms that reduce the depth (resp. required qubits) of Banegas et al.'s (resp. Putranto et al.'s) FLT-based algorithms. On the other hand, these works do not change the relationship between GCD-based and FLT-based inversion algorithms. When $n = 571$, Kim-Hong's GCD-based and Taguchi-Takayasu's FLT-based inversion algorithms require $3,473$ and $8,566$ qubits, respectively, while the latter requires only 2.3% of Toffoli gates and 84% of depth to run Shor's algorithm. Therefore, it is desirable to develop GCD-based (resp. FLT-based) inversion algorithms that drastically reduce required Toffoli gates and depth (resp. required qubits) of Kim-Hong's GCD-based (resp. Taguchi-Takayasu's FLT-based) algorithms.

1.2 Our Contribution

In this paper, we break the relationship between GCD-based and FLT-based inversion algorithms by presenting an FLT-based method that requires much fewer qubits. When $n = 571$, our method requires $3,998$ qubits to run Shor's algorithm and reduces all main quantum resources of Banegas et al.'s GCD-based algorithm [BBvHL20]. Although the required qubits are still more than Kim-Hong's GCD-based algorithm, they are competitive since ours are just 15% more than Kim-Hong. Furthermore, our method preserves the advantage of FLT-based inversion since it requires only 3.7% of Toffoli gates and 77% of depth to run Shor's algorithm compared to Kim-Hong's GCD-based inversion algorithm.

We briefly explain three technical ingredients to obtain the result.

Register-Bounded Addition Chain. Taguchi-Takayasu's FLT-based inversion algorithm takes an addition chain as input and computes a sequence whose number is the same as the length of the chain. Briefly

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

[2]Although Banegas et al. [BBvHL20] used Hoof's quantum multiplication algorithm [vH19], we replace it with more efficient Kim et al.'s quantum multiplication algorithm [KKKH22] and update their analysis. We use the more efficient algorithm throughout the paper.

speaking, the addition chain represents the sequence of computation. Unfortunately, this procedure wastes the number of ancillary registers since there are several terms that are stored until the end of the computation, while they are used only at an early step of the computation. If we delete such terms, we can save the required qubits; however, an addition chain does not indicate which terms should be deleted and when. For this purpose, we introduce a *register-bounded addition chain*. A register-bounded addition chain is a longer sequence than an addition chain and represents the sequence of computation/uncomputation. We find register-bounded addition chains for NIST recommended degrees $n = 163, 233, 283, 571$ and reduce the required qubits for inversions.

Modified Elliptic Curve Point Addition Algorithm. Although a register-bounded addition chain enables us to reduce required qubits, the resulting inversion algorithm requires slightly more qubits than Banegas et al.'s GCD-based inversion algorithm. Since our final target is not an inversion itself but Shor's algorithm, we modify Banegas et al.'s point addition algorithm [BBvHL20] and further reduce the required qubits for running Shor's algorithm. Interestingly, our proposed point addition algorithm itself does not reduce the required qubits; however, it becomes effective when combined with our inversion algorithm. Specifically, we design our point addition algorithm so that the proposed inversion algorithm and the point addition algorithm share the same ancillary registers.

Depth Reduction of Quantum Multiple Squaring Circuits. The above two ingredients enable us to run Shor's algorithm with $3,998$ qubits for $n = 571$. However, the algorithm lost the advantage of FLT-based inversion since it requires more depth than GCD-based inversion algorithms. To preserve the advantage, we find how to perform parallel quantum computation during FLT-based inversion and reduce the depth. Since FLT-based inversion is inherently required to compute 2^k-th powers many times for large k, previous FLT-based inversion algorithms applied a circuit for computing squaring k times for computing 2^k-th power. In contrast, we analyze quantum circuits for computing 2^k-th powers directly and find that much less depth is sufficient for any n. The circuits are effective for all FLT-based inversion algorithms and enable our algorithm to preserve the advantage of FLT-based inversion.

Organization. In Section 3, we present our FLT-based method. In Section 4, we analyze the quantum resource and compare it with previous ones.

Difference from Preliminary Version. In [TT24], the depth of $\texttt{spSQUARE}^k$ described in Section 3.3 which computes 2^k-th power is roughly estimated. In this version, we compute the exact depth for $\texttt{spSQUARE}^k$ and estimate quantum resources for Shor's algorithm.

2 Preliminaries

In Section 2.1, we explain binary elliptic curves and a binary elliptic curve discrete logarithm problem (binary ECDLP). In Section 2.2, we explain quantum computations and quantum basic arithmetics over $\mathbb{F}_{2^n}$. In Section 2.3, we describe Shor's algorithm for solving the binary ECDLP.

2.1 Binary Elliptic Curve Discrete Logarithm Problem

Let n be a non-negative 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 $b \in \mathbb{F}_{2^n}^*$. The set of rational points on an elliptic curve and a special point O form an abelian group under point addition, where O is the identity element called a point at infinity. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ denote rational points on a binary elliptic curve. If $P \neq Q$, $P + Q = (x_3, y_3)$ is given by

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

where $\lambda = (y_1 + y_2)/(x_1 + x_2)$. Otherwise, $P + P = (x_3, y_3)$ is given by

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

where $\lambda = x_1 + y_1/x_1$. As the above formulas imply, we compute an inversion when we compute a point addition. Hereafter, $[k]P$ denotes a sum of k P's under point addition. The above two formulas indicate that we can compute $[k]P$ from P and k in polynomial time. However, there is no known polynomial time algorithm that computes k from P and $[k]P$. This problem over a binary field is called the binary elliptic curve discrete logarithm problem (binary ECDLP).

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

In classical computation, we use a "bit" represented by 0 or 1. In contrast, in quantum computation, we use a "qubit" represented by $|0\rangle, |1\rangle$ and their superposition. Let $m(x)$ be an irreducible polynomial in $\mathbb{F}_2[x]$ of degree n and $(m(x))$ be an ideal generated by $m(x)$ over $\mathbb{F}_2[x]$. To represent an element in $f \in \mathbb{F}_{2^n}$ by qubits, we use a polynomial representation based on a relation $\mathbb{F}_{2^n} \simeq \mathbb{F}_2[x]/(m(x))$. Since f is represented by a polynomial of degree less than $n-1$, we represent it by n qubits and corresponding coefficients of the polynomial as the quantum state of $|0\rangle$ or $|1\rangle$. Hereafter, we call the n qubits representing an element in $\mathbb{F}_{2^n}$ a *register*.

We employs quantum circuits to describe quantum computations, where X gates, CNOT gates, Toffoli (TOF) gates, and SWAP gates are basic quantum gates. An X gate exchanges the coefficients of $|0\rangle$ and $|1\rangle$. Let a, b, and c denote $|0\rangle$ or $|1\rangle$. Then, CNOT, TOF, and SWAP operations are given by $\mathrm{CNOT}(a, b) = (a, a \oplus b)$, $\mathrm{TOF}(a, b, c) = (a, b, c \oplus (a \cdot b))$, and $\mathrm{SWAP}(a, b) = (b, a)$, respectively. A TOF gate is believed to be much more expensive than a CNOT gate. To explain our method in Section 3, we may use a SWAP gate; however, we do not use the gate actually by designing subsequent circuits appropriately.

Next, we explain quantum basic arithmetics. Let f, g, and h denote quantum states of elements in $\mathbb{F}_{2^n}$. We use `ADD` (resp. `SQUARE` and `spSQUARE`) to denote Banegas et al.'s algorithm [BBvHL20] for addition (resp. squaring) over $\mathbb{F}_{2^n}$, where $\texttt{ADD}\,(f, g) = (f, f+g)$, $\texttt{SQUARE}\,(f) = f^2$, and $\texttt{spSQUARE}\,(f, g) = (f, f^2+g)$. We can use `ADD` to compute a copy of a given element by $\texttt{ADD}(f, 0) = (f, f)$. We use $\texttt{SQUARE}^{-1}$ and $\texttt{spSQUARE}^{-1}$ to denote inverse operations of `SQUARE` and `spSQUARE`, respectively. Banegas et al.'s algorithms [BBvHL20] for computing the operations are based only on CNOT gates, where `ADD`, `SQUARE`, and `spSQUARE` require n, at most $n^2 - n$, and at most n^2 CNOT gates, respectively. Circuits for computing $\texttt{SQUARE}^{-1}$ and $\texttt{spSQUARE}^{-1}$ are reversed circuits for computing `SQUARE` and `spSQUARE`, respectively. We use `MODMULT` to denote Kim et al.'s multiplication algorithm over $\mathbb{F}_{2^n}$ [KKKH22], where $\texttt{MODMULT}\,(f, g, h) = (f, g, f \cdot g + h)$ which requires TOF gates as well as CNOT gates. Indeed, we can compute multiplication of given two elements by $\texttt{MODMULT}\,(f, g, 0) = (f, g, f \cdot g)$. Since we consider the arithmetics over $\mathbb{F}_{2^n}$, it holds that $\texttt{ADD}\,(f, f) = (f, 0)$ and $\texttt{MODMULT}\,(f, g, f \cdot g) = (f, g, 0)$.

Finally, we describe `INV` which denotes the inversion computation over $\mathbb{F}_{2^n}$, where $\texttt{INV}\,(f, [0, \ldots, 0], 0) = (f, [r_1, \ldots, r_m], f^{-1})$. Observe that `INV` requires $m+2$ registers whose first one stores $f \in \mathbb{F}_{2^n}$. The other $m+1$ registers are ancillary registers that include the last one to store f^{-1}. We call the register for output and the m registers enclosed by [ ] *inversion ancillary registers*. Moreover, we call an inversion ancillary register a *dirty ancillary register* if the output r_i is non-zero. We use $\texttt{INV}^{-1}$ to denote an inverse operation of `INV`, where $\texttt{INV}^{-1}(f, [r_1, \ldots, r_m], f^{-1}) = (f, [0, \ldots, 0], 0)$. We use $\texttt{INV}^{-1}$ only when the input $[r_1, \ldots, r_m]$ in the inversion ancillary registers is the same as the output of `INV` in the same registers.

In this paper, the above quantum computations also take registers as input, e.g., $\texttt{ADD}(g_1, g_2)$, where g_1 and g_2 is a register which stores $f \in \mathbb{F}_{2^n}$ and 0, respectively. Then, $\texttt{ADD}(g_1, g_2)$ describes $\texttt{ADD}(f, 0) = (f, f)$.

2.3 Shor's Algorithm for Solving the Binary ECDLP

Shor's algorithm mainly consists of a point addition part and a quantum Fourier transform. Since the former and the latter require $O(n^3)$ and $O(n^2)$ quantum gates, respectively, the point addition part is relatively expensive. Banegas et al.'s point addition algorithm [BBvHL20] consists of quantum arithmetics over $\mathbb{F}_{2^n}$ denoted by `MODMULT`, `INV`, $\texttt{INV}^{-1}$, `spSQUARE`, `const_ADD`, `ctrl_ADD`, and `ctrl_const_ADD`. Although we do not explain in detail, `const_ADD`, `ctrl_ADD`, and `ctrl_const_ADD` operate addition, where they require at most n X gates, at most n TOF gates, and at most n CNOT gates, respectively. In this paper, we count the numbers of TOF and CNOT gates and ignore X gates by following previous works [BBvHL20]. Banegas et al.'s point addition algorithm requires $3n+1$ qubits except inversion ancillary registers. More precisely, they require $2n+1$ qubits for input and n qubits for an ancillary register of point addition which we call a *point addition ancillary register*.

3 Our Method

In Section 3.1, we explain register-bounded addition chain. In Section 3.2, we propose a quantum point addition algorithm. In Section 3.3, we describe the depth reduction of squaring. In Section 3.4, we show our

quantum FLT-based inversion algorithm.

3.1 Register-Bounded Addition Chain

Hereafter, we use a notation $\langle\alpha\rangle \coloneqq f^{\alpha}$ for simplicity for $f \in \mathbb{F}_{2^n}^*$. Then, the FLT-based inversion computes $\langle -1\rangle = \langle 2^n - 2\rangle$. We focus on the computation of $\langle 2^{n-1} - 1\rangle$ hereafter since we can compute $\langle 2^n - 2\rangle$ by applying squaring to $\langle 2^{n-1} - 1\rangle$.

Taguchi-Takayasu's FLT-based Algorithm. At first, we summarize overvie- ws of Taguchi-Takayasu's quantum FLT-based inversion algorithm [TT23]. To be precise, Taguchi and Takayasu proposed two algorithms, i.e., Basic algorithm and Extended algorithm. Hereafter, we only describe thier Extended algorithm since their Extended algorithm requires fewer qubits than their Basic algorithm. Therefore, we call their Extended algorithm simply Taguchi-Takayasu's FLT-based algorithm.

We review an addition chain that is Taguchi-Takayasu's FLT-based essential ingredient to improve previous FLT-based algorithms.

Definition 1 (Addition chain)**.** *Let ℓ and N denote non-negative integers. An addition chain for N of length ℓ is a sequence $p_0 = 1, p_1, p_2, \ldots, p_\ell = N$ which satisfies the following condition:*

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

We call each term p_s of an addition chain a doubled term or an added term. In particular, if there are no i and j which satisfy $0 \leq i, j < s, p_s = p_i + p_j$, and $p_i \neq p_j$, and an added term otherwise. Taguchi-Takayasu's FLT-based algorithm takes $\langle 2^{p_0} - 1\rangle = f \in \mathbb{F}_{2^n}^*$ and an addition chain $\{p_s\}_{s=0}^{\ell}$ for $n-1$ of length ℓ as inputs and computes $\langle 2^{p_1} - 1\rangle, \langle 2^{p_2} - 1\rangle, \ldots, \langle 2^{p_\ell} - 1\rangle = \langle 2^{n-1} - 1\rangle$ sequentially by the relation

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

Taguchi-Takayasu's FLT-based algorithm computes $\langle 2^{p_s} - 1\rangle$ in two distinct ways for all $1 \leq s \leq \ell$ depending on whether p_s is an added term or a doubled term. If p_s is an added term, we compute $\langle 2^{p_s} - 1\rangle = \langle 2^{p_i+p_j} - 1\rangle$ from $\langle 2^{p_i} - 1\rangle$ and $\langle 2^{p_j} - 1\rangle$ which have been stored distinct registers. In particular, we first apply `SQUARE` p_i times to $\langle 2^{p_j} - 1\rangle$ and obtain $\langle 2^{p_i+p_j} - 2^{p_i}\rangle$. After that, we apply `MODMULT` to $\langle 2^{p_i+p_j} - 2^{p_i}\rangle$ and $\langle 2^{p_i} - 1\rangle$ and obtain $\langle 2^{p_i+p_j} - 1\rangle = \langle 2^{p_s} - 1\rangle$. On the other hand, if p_s is a doubled term, we first compute a copy of $\langle 2^{p_i} - 1\rangle$ in another ancillary register by using `ADD`. Then, we apply `SQUARE` p_i times to the copy and obtain $\langle 2^{p_i+p_i} - 2^{p_i}\rangle$. Finally, we apply `MODMULT` to $\langle 2^{p_i} - 1\rangle$ and $\langle 2^{p_i+p_i} - 2^{p_i}\rangle$ and obtain $\langle 2^{p_i+p_i} - 1\rangle = \langle 2^{p_s} - 1\rangle$. To reduce the qubits, we uncompute the copy of $\langle 2^{p_i} - 1\rangle$ by `ADD`. Theorem 1 describes the quantum resources for Taguchi-Takayasu's FLT-based algorithm.

Theorem 1 ([TT23], Theorem 2)**.** *Let f be an element in $\mathbb{F}_{2^n}^*$ and $\{p_s\}_{s=0}^{\ell}$ be an addition chain for $n-1$ of length ℓ with p_ℓ is an added term. Taguchi-Takayasu's FLT-based algorithm takes $f = \langle 1\rangle$ and $\{p_s\}_{s=0}^{\ell}$ as input and outputs $\langle -1\rangle = \langle 2^n - 2\rangle$ with ℓ ancillary registers and ℓ multiplications.*

Taguchi-Takayasu's FLT-based algorithm requires ℓ ancillary registers to store $\langle 2^{p_s} - 1\rangle$ for all $s = 1, 2, \ldots, \ell$. Furthermore, every term of an addition chain for Taguchi-Takayasu's algorithm appears only once.

Our Proposed Algorithm. Now, we reduce even more qubits than Taguchi-Takayasu's FLT-based algorithm. Keen readers may notice that we can further reduce required qubits by uncomputing not only copied $\langle 2^{p_s} - 1\rangle$ but also original $\langle 2^{p_s} - 1\rangle$ itself. For an example of Taguchi-Takayasu's FLT-based algorithm with an addition chain $\{p_s\}_{s=0}^{9} = \{1, 2, 3, 6, 9, 18, 27, 54, 108, 162\}$, observe that $\{1, 2, 3, 6, 9, 18\}$ will not be used again after computing 27. In other words, after we compute 27, we can uncompute $\{2, 3, 6, 9, 18\}$ if possible. However, while an addition chain tells us a sequence of computation, it does not tell us which terms can be uncomputed and when. Therefore, we need another method to analyze our proposed algorithm. For this purpose, we introduce register-bounded addition chains.

Definition 2 (Register-Bounded Addition Chain)**.** *Let $\tilde{\ell}$ and N denote non-negative integers. A register-bounded addition chain for N of length $\tilde{\ell}$ is a sequence $\tilde{p} \coloneqq \{\tilde{p}_s\}_{s=0}^{\tilde{\ell}} = \tilde{p}_0 = 1, \tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_{\tilde{\ell}}$ which satisfies following conditions:*

- *For all $s = 1, \ldots, \tilde{\ell}$, there exist i and j which satisfy $\tilde{p}_i + \tilde{p}_j = \tilde{p}_s$ and $\tilde{p}_i \in S(\tilde{p}, s-1), \tilde{p}_j \in S(\tilde{p}, s-1)$, where $S(\tilde{p}, t) \coloneqq \{\tilde{p}_s \mid 0 \leq s \leq t$, there exists no s' such that $0 \leq s' \leq t, s \neq s'$, and $\tilde{p}_s = \tilde{p}_{s'}\}$.*
- *There exists ω which satisfies $\tilde{p}_\omega = N$.*
- *Every term appears once or twice.*

Due to the first condition, a register-bounded addition chain $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ is an addition chain. Therefore, we can also define doubled terms and added terms for a register-bounded addition chain. Furthermore, a sequence of different terms of $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ is also an addition chain. A register-bounded addition chain explains both computations and uncomputations. Specifically, the first and second time each term $\tilde{p}_s$ appear, we compute and uncompute $f^{2^{\tilde{p}_s}-1}$, respectively. Briefly speaking, $S(\tilde{p}, t)$ is a set of $\tilde{p}_0, \tilde{p}_1, \ldots, \tilde{p}_t$ that appear only once. Thus, when we compute or uncompute $\tilde{p}_s$ for all $1 \leq s \leq \tilde{\ell}$, we choose former terms that appear once in $\tilde{p}_0, \tilde{p}_1, \ldots, \tilde{p}_{s-1}$, while there is no condition for an addition chain. Then, we define a function $C(\tilde{p}, t)$ by $C(\tilde{p}, t) \coloneqq 1$ when $\tilde{p}_t$ is a doubled term and $C(\tilde{p}, t) \coloneqq 0$ otherwise. We also define $r(\tilde{p}, t)$ which we call the *register counting function* given by $r(\tilde{p}, t) \coloneqq \#S(\tilde{p}, t) + C(\tilde{p}, t) - 1$. Intuitively, $r(\tilde{p}, t)$ denotes the number of required ancillary registers when we compute or uncompute $\langle 2^{\tilde{p}_t} - 1\rangle$. Moreover, we use the notation $R(\tilde{p}) \coloneqq \max_{1 \leq t \leq \tilde{\ell}} r(\tilde{p}, t)$ hereafter. Thus, $R(\tilde{p})$ describes the number of required ancillary registers for a whole inversion computation. We explain quantum resources for a quantum FLT-based inversion algorithm which we compute and uncompute based on a register-bounded addition chain by Theorem 2.

Theorem 2. *Let f be an element in $\mathbb{F}_{2^n}^*$, $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ be a register-bounded addition chain for $n-1$ of length $\tilde{\ell}$, and ℓ denote the length of an addition chain which consists of different terms of $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$. There exists a quantum algorithm that takes $f = \langle 1\rangle$ and $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ as input and outputs $\langle -1\rangle = \langle 2^n - 2\rangle$ with $R(\tilde{p})$ ancillary registers, $\tilde{\ell}$ multiplications, and $2\ell - \tilde{\ell}$ dirty ancillary registers at the end of the algorithm.*

Proof. We compute or uncompute $\langle 2^{\tilde{p}_s} - 1\rangle$ in the s-th procedure for all $s = 1, \ldots, \tilde{\ell}$. More precisely, we compute $\langle 2^{\tilde{p}_s} - 1\rangle$ if $\tilde{p}_s$ appears for the first time in $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ and uncompute $\langle 2^{\tilde{p}_s} - 1\rangle$ if it is the second time to appear. By the second condition of Definition 2, we compute $\langle 2^{n-1} - 1\rangle$ in the ω-th procedure, where $0 \leq \omega \leq \tilde{\ell}$ in the same way as Taguchi-Takayasu's FLT-based algorithm. Then, we explain uncomputations of $\langle 2^{\tilde{p}_s} - 1\rangle$. We only describe the case that $\tilde{p}_s$ is a doubled term.

Uncomputation of $\langle 2^{\tilde{p}_s} - 1\rangle$: We have $\langle 2^{\tilde{p}_i} - 1\rangle$ stored in the register g_{k_1}, 0 stored in the register g_{k_2}, and $\langle 2^{\tilde{p}_s} - 1\rangle$ stored in the register g_{k_3}, where $\tilde{p}_s = \tilde{p}_i + \tilde{p}_i$ and $i \in S(\tilde{p}, s-1)$. At first, we apply $\texttt{ADD}\,(g_{k_1}, g_{k_2})$ and obtain $\langle 2^{\tilde{p}_i} - 1\rangle$ in the g_{k_2}. Next, we apply $\texttt{SQUARE}$ $\tilde{p}_i$ times to the g_{k_2} and obtain $\langle 2^{\tilde{p}_i} - 1\rangle^{2^{\tilde{p}_i}} = \langle 2^{\tilde{p}_i + \tilde{p}_i} - 2^{\tilde{p}_i}\rangle$. Then, we apply $\texttt{MODMULT}\,(g_{k_1}, g_{k_2}, g_{k_3})$ and obtain $\langle 2^{\tilde{p}_i+\tilde{p}_i} - 2^{\tilde{p}_i}\rangle \times \langle 2^{\tilde{p}_i} - 1\rangle + \langle 2^{\tilde{p}_s} - 1\rangle = \langle 2^{\tilde{p}_i+\tilde{p}_i} - 1\rangle + \langle 2^{\tilde{p}_s} - 1\rangle = \langle 2^{\tilde{p}_s} - 1\rangle + \langle 2^{\tilde{p}_s} - 1\rangle = 0$ in the g_{k_3} by (1). By the same procedure as the uncomputation of copy in Taguchi-Takayasu's FLT-based algorithm, we uncompute the g_{k_2}.

Then, $\langle 2^{\tilde{p}_s} - 1\rangle$ is stored after the t-th procedure if and only if $s \in S(\tilde{p}, t)$. Therefore, we can always compute or uncompute $\langle 2^{\tilde{p}_s} - 1\rangle$ since there exist $\langle 2^{\tilde{p}_i} - 1\rangle$ and $\langle 2^{\tilde{p}_j} - 1\rangle$ in some registers such that $0 \leq i, j < s, \tilde{p}_s = \tilde{p}_i + \tilde{p}_j$ by the first condition of Definition 2. Furthermore, $\#S(\tilde{p}, t)$ describes the number of registers that store non-zero terms including input after the t-th procedure. However, we also require another register to copy $\langle 2^{\tilde{p}_t} - 1\rangle$ when $\tilde{p}_t$ is a doubled term. In other words, we use $\#S(\tilde{p}, t)+1$ registers when $\tilde{p}_t$ is a doubled term and $\#S(\tilde{p}, t)$ registers when $\tilde{p}_t$ is an added term for t-th procedure. Then, the number of required ancillary registers for t-th procedure is $\#S(\tilde{p}, t) + C(\tilde{p}, t) - 1 = r(\tilde{p}, t)$. Therefore, we require $\max_{0 \leq t \leq \tilde{\ell}} r(\tilde{p}, t) = R(\tilde{p})$ ancillary registers. Moreover, each procedure requires a multiplication, in other words, we require $\tilde{\ell}$ multiplications in total. The third condition of Definition 2 ensures that we compute ℓ different terms only once, in other words, we do not compute $\langle 2^{\tilde{p}_s} - 1\rangle$ after we uncompute $\langle 2^{\tilde{p}_s} - 1\rangle$. We require ℓ registers to store them. However, there are $r(\tilde{p}, \tilde{\ell})$ non-zero ancillary registers at the end of the algorithm. Therefore, we uncompute $\ell - r(\tilde{p}, \tilde{\ell})$ times. Then, it holds that $\tilde{\ell} = \ell + (\ell - r(\tilde{p}, \tilde{\ell})) = 2\ell - r(\tilde{p}, \tilde{\ell})$. By this relation, it holds $r(\tilde{p}, \tilde{\ell}) = 2\ell - \tilde{\ell}$. □

By Theorem 2, we use $R(\tilde{p})n$ qubits except for g_0, and $\tilde{\ell} = 2\ell - r(\tilde{p}, \tilde{\ell})$ multiplications for our proposed inversion algorithm. Then, when we fix ℓ and $R(\tilde{p})$, larger $r(\tilde{p}, \tilde{\ell})$ is desired to reduce multiplications. On the other hand, it holds that $r(\tilde{p}, \tilde{\ell}) \leq R(\tilde{p})$ by the definition of $R(\tilde{p})$.

Algorithm 1 Proposed quantum point addition algorithm

Input: An irreducible polynomial $m(x) \in \mathbb{F}_2[x]$ of degree n, a coefficient of an binary elliptic curve a, single qubit q, an elliptic curve point $P_1 = (x_1, y_1)$ stored in x, y, a fixed elliptic curve point $P_2 = (x_2, y_2)$, a non-negative integer R, registers $g_1, g_2, \ldots, g_{R-1}, g_R = \lambda$ initialized to an all-$|0\rangle$ state

Output: $(x, y) = P_1 + P_2 = P_3(x_3, y_3)$ if $q = 1$
$(x, y) = P_1 = (x_1, x_2)$ if $q = 0$

1: `const_ADD` (x_2, x)
2: `ctrl_const_ADD`$_q(y_2, y)$ // $\lambda = 0$
3: `INV` $(x, [g_2, \ldots, g_{R-1}, \lambda], g_1)$
4: `MODMULT` (g_1, y, λ)
5: `MODMULT` (x, λ, y)
6: `SWAP` (y, λ) // $\lambda = 0$
7: `INV`$^{-1}(x, [g_2, \ldots, g_{R-1}, \lambda], g_1)$
8: `SWAP` (y, λ)
9: `spSQUARE` (λ, y)
10: `ctrl_const_ADD`$_q(a + x_2, x)$
11: `ctrl_ADD`$_q(\lambda, x)$
12: `ctrl_ADD`$_q(y, x)$
13: `spSQUARE` (λ, y)
14: `SWAP` (y, λ) // $\lambda = 0$
15: `INV` $(x, [g_2, \ldots, g_{R-1}, \lambda], g_1)$
16: `SWAP` (y, λ)
17: `MODMULT` (x, λ, y)
18: `MODMULT` (g_1, y, λ) // $\lambda = 0$
19: `INV`$^{-1}(x, [g_2, \ldots, g_{R-1}, \lambda], g_1)$
20: `const_ADD` (x_2, x)
21: `ctrl_ADD`$_q(x, y)$
22: `ctrl_const_ADD`$_q(y_2, y)$

3.2 Modified Quantum Point Addition Algorithm

As we explained in Section 2.3, there are two types of ancillary registers, i.e., inversion ancillary registers and a point addition ancillary register to run Shor's algorithm. We modify Banegas et al.'s quantum point addition algorithm [BBvHL20] described as Algorithm 1 to reduce required qubits by combining with our FLT-based inversion algorithm in Section 3.1, where we use $R := R(\tilde{p})$ to describe the number of inversion ancillary registers for simplicity. Intuitively, we delete the point addition ancillary register and perform point addition by using an inversion ancillary register. Briefly speaking, Algorithm 1 is the same as Banegas et al.'s algorithm by deleting `SWAP` operations in lines 6 and 16, exchanging line 5 and line 7, and exchanging line 15 and line 17. The modification changes the role of a register λ which is a point addition ancillary register in Banegas et al.'s algorithm, while it is both a point addition ancillary register and an inversion ancillary register in Algorithm 1. In other words, all R inversion ancillary registers are divided into the registers for only inversion computation, i.e., $g_1, \ldots, g_{R-1}$, and the register for both inversion computation and point addition computation, i.e., λ. Then, the number of qubits for Shor's algorithm with Algorithm 1 is $(2 + R)n + 1$ qubits, while $(3 + R)n + 1$ qubits with Banegas et al.'s point addition algorithm. We note that Algorithm 1 itself does not purely improve Banegas et al.'s point addition algorithm since Algorithm 1 requires some conditions. Concretely, Algorithm 1 requires that `INV` and `INV`$^{-1}$ satisfy two conditions, i.e., (i) λ store 0 at the end of `INV` and (ii) x at the beginning of `INV` (`INV`$^{-1}$) and x at the end of `INV` (`INV`$^{-1}$) must be the same state. Quantum FLT-based inversion algorithms always satisfy (ii). However, previous FLT-based inversion algorithms do not satisfy (i) since they fully use all registers at the end of algorihtm. Our proposed FLT-based inversion algorithm can prepare a clear register at the end of algorithm by choosing $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ properly. We explain the detail in Section 3.4. On the other hand, previous quantum

GCD-based inversion algorithms satisfy (i), while they do not satisfy (ii). GCD-based inversion algorithms apply Euclidean algorithm to x and m, where m is an irreducible polynomial in Section 2.2. In Euclidean algorithm, we compute $x \leftarrow x \mod m$ or $m \leftarrow m \mod x$ until it holds $x = 1$ or $m = 1$. Thus, x at the end of quantum GCD-based inverion algorithms is different state to x at the beginning.

3.3 Depth Reduction of Quantum Multiple Squaring Circuits

We explain how to reduce the depth of quantum circuits for computing 2^k-th powers. Let $f = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$ be a polynomial which represents an element in $\mathbb{F}_{2^n}$ with coefficients $a_i \in \mathbb{F}_2$. For an irreducible polynomial $m(x) \in \mathbb{F}_2[x]$ of degree n, we have $f^2 = a_0 + a_1 x^2 + \cdots + a_{n-1}x^{2n-2} \mod m(x) = a'_0 + a'_1 x + \cdots + a'_{n-1}x^{n-1} \mod m(x)$. Since each a'_i is a sum of $a_0, a_1, \ldots, a_{n-1}$, there exists a matrix $T_n = (t_{i,j}) \in GL_n(\mathbb{F}_2)$ which satisfies

$$[a'_0, a'_1, a'_2, \ldots, a'_{n-1}]^\top = T_n [a_0, a_1, a_2, \ldots, a_{n-1}]^\top, \tag{2}$$

where $t_{i,j} \in \mathbb{F}_2$ for all $1 \leq i, j \leq n$. The matrix T_n is uniquely determined for $m(x)$; in other words, the relation (2) holds for any f.

Banegas et al.'s Estimate. We explain how Banegas et al. [BBvHL20] constructed a quantum circuit of `SQUARE` and `spSQUARE` by using the above matrix T_n. We also review their quantum resource estimation of `SQUARE` and `spSQUARE`.

`SQUARE`. Let $T_n = L_n U_n P_n$ be an LUP decomposition, where L_n and U_n are lower and upper triangular matrices, respectively, and P_n is a permutation matrix. The multiplication by matrices U_n and L_n (resp. P_n) can be performed by CNOT (resp. SWAP) gates. In particular, Banegas et al. showed that the numbers of CNOT gates are the number of ones in L_n and U_n except their diagonal entries; thus, the circuits require at most $n(n-1)/2$ CNOT gates and the depth is at most $n(n-1)/2$. In total, $\texttt{SQUARE}(f) = f^2$ requires at most $n^2 - n$ CNOT gates and the depth is at most $n^2 - n$. Since we can compute the concrete number of CNOT gates of `SQUARE` for every irreducible polynomial $m(x)$, we use SQ_n to denote the number, where the depth is at most SQ_n.

`spSQUARE`. For the above matrix T_n determined by an irreducible polynomial $m(x)$, let $spSQ_n \geq n$ denote the number of ones in T_n including diagonal entries. Let a_i and b_i be coefficients of f and g for x^i, respectively. Then, we can describe a computation $\texttt{spSQUARE}\,(f, g) = (f, f^2 + g)$ by

$$\begin{bmatrix} I_n & O_n \\ T_n & I_n \end{bmatrix} [a_0, a_1, \ldots, a_{n-1}, b_0, b_1, \ldots, b_{n-1}]^\top, \tag{3}$$

where I_n and O_n are an identity matrix and a zero matrix, respectively. As `SQUARE`, we can compute `spSQUARE` with $spSQ_n$ CNOT gates and the upper bound of depth of the circuit is $spSQ_n$.

Depth Reduction of `spSQUARE`. Observe that $spSQ_n \geq n$ holds due to $T_n \in GL_n(\mathbb{F}_2)$. However, we show that a smaller depth is sufficient for computing `spSQUARE` with the following stronger claim.

Theorem 3. *Let $R_i(A)$ and $C_i(A)$ denote a number of ones in i-th row of A and i-th column of A, respectively, where $A \in M_n(\mathbb{F}_2)$ and $i = 1, \ldots, n$. Let $L(A) \coloneqq \max(R_1(A), \ldots, R_n(A), C_1(A), \ldots, C_n(A))$. For a matrix $H_n = (h_{i,j}) \in M_n(\mathbb{F}_2)$, there exists a quantum circuit for computing a multiplication by a matrix $\begin{bmatrix} I_n & O_n \\ H_n & I_n \end{bmatrix}$ with depth $L(H_n)$.*

Difference from Preliminary Version. In [TT24], we estimate the *upper bound* of the depth for the quantum computation described in Theorem 3 is n. On the other hand, we estimate the *exact* depth in this version.

Before providing a proof, we show an example for $H_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, where $spSQ_3 = 7$. In this case, we want to compute

$$\begin{bmatrix} I_3 & O_3 \\ H_3 & I_3 \end{bmatrix} [a_0, a_1, \ldots, a_{n-1}, b_0, b_1, \ldots, b_{n-1}]^\top$$
$$= [a_0, a_1, a_2, b_0 + a_0 + a_1, b_1 + a_0 + a_2, b_2 + a_0 + a_1 + a_2]^\top.$$

It is easy to check that $spSQ_3 = 7$ CNOT gates are sufficient for the purpose by adding a_0 to the fourth, fifth, and sixth bits, a_1 to the fourth and sixth bits, and a_2 to the fifth and sixth bits. We can design a circuit with depth $spSQ_3 = 7$ by applying the CNOT gates one by one. On the other hand, we find that the depth $n = 3$ is sufficient by applying several CNOT gates simultaneously. In particular, the following design of a circuit works with the claimed depth, while distinct CNOT gates do not share their working bits at the same time:

- Add a_0 and a_2 to the fourth and sixth bits, respectively.
- Add a_0, a_1, and a_2 to the sixth, fourth, and fifth bits, respectively.
- Add a_0 and a_1 to the fifth and sixth bits, respectively.

We express the design by matrices

$$\Gamma_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \Gamma_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \qquad \Gamma_3 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

such that $H_3 = \Gamma_1 + \Gamma_2 + \Gamma_3$ and every rows and columns have at most one 1. The three columns of the matrices correspond to the first, second, and third bits, while the three rows correspond to the fourth, fifth, and sixth bits. The condition $H_3 = \Gamma_1 + \Gamma_2 + \Gamma_3$ ensures that matrices Γ_1, Γ_2, and Γ_3 represent the computation by H_3, while the other condition ensures that distinct CNOT gates do not share their working bits at the same time. We show how to decompose H_n to at most $L(H_n)$ Γ_i's in general and provide a proof of Theorem 3.

Proof. We construct a bipartite graph $G = (V, E)$ as follows:

- Let $U = \{u_1, \ldots, u_n\}, W = \{w_1, \ldots, w_n\}$ and $V = U \cup W$,
- Let $E = \{(u_i, w_j) \mid h_{ij} = 1\}$.

Then, the degree of u_i which is called $d(u_i)$ equals $R_i(H_n)$ and $d(w_i) = C_i(H_n)$, where $i = 1, \ldots, n$. Thus, the maximum degree of G called ΔG equals $L(H_n)$. Now, we give an edge coloring to G. Since G is a bipartite graph, the edge chromatic number of G is determined by Theorem 4.

Theorem 4 ([Kön16])**.** *The edge chromatic number of any bipartite graph equals its maximum vertex degree.*

Therefore, the edge chromatic number of G is $\Delta G = L(H_n)$.

We consider the graph G which is colored with $L(H_n)$ colors. Then, we construct matrices $\Gamma_1, \ldots, \Gamma_{L(H_n)}$ as follows:

- Let $E_s = \{e \in E \mid e \text{ is colored by } s\text{-th color}\}$, where $s = 1, \ldots, L(H_n)$,
- Let $\Gamma_s = (\iota_{i,j})$, where $\iota_{i,j} = 1$ if and only if $G_s \coloneqq (V, E_s)$ contains an edge (u_i, w_j).

By the definition of edge coloring, it holds that $R_i(\Gamma_s) = 1, C_i(\Gamma_s) = 1$ for all $i = 1, \ldots, n, s = 1, \ldots, L(H_n)$ and $\Gamma_1 + \cdots + \Gamma_{L(H_n)} = H_n$. □

By Theorem 3, we reduce the depth for `spSQUARE` from SQ_n to n. However, this is only a small contribution when we estimate the resources for Shor's algorithm since $H_n = T_n$ is a sparse matrix and SQ_n is sufficiently close to n for all NIST-recommended n. On the other hand, if H_n is not sparse, we can drastically reduce the depth.

We consider quantum FLT-based inversion algorithms. Let k be a non-negative integer. When we compute a doubled term, i.e., $\langle 2^{2k} - 1 \rangle$ by using a register g_1 which stores $\langle 2^k - 1 \rangle$ and g_2 which stores 0, we employ a quantum computation called `ADD-SQUARE`^{k} given by

1: `ADD` (g_1, g_2)

2: **for** $s = 1, \ldots, k$ **do**

3: $\texttt{SQUARE}\,(g_2)$

Previous works estimated the depth for $\texttt{ADD-SQUARE}^k$ is $1 + kSQ_n$. We give a tighter upper bound for $\texttt{ADD-SQUARE}^k$.

Let f in $\mathbb{F}_{2^n}$ and denote $f = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$, where $a_i \in \mathbb{F}_2$. By observing (2), T_n^k satisfies

$$[a_0^{(k)}, a_1^{(k)}, a_2^{(k)}, \ldots, a_{n-1}^{(k)}]^\top = T_n^k[a_0, a_1, a_2, \ldots, a_{n-1}]^\top, \tag{4}$$

where $a_i^{(k)}$ is a coefficient of x^i for f^{2^k} for $i = 0, 1, \ldots, n-1$. Then, a quantum computation called $\texttt{spSQUARE}^k$ given by

$$\begin{bmatrix} I_n & O_n \\ T_n^k & I_n \end{bmatrix} [a_0, a_1, \ldots, a_{n-1}, 0, 0, \ldots, 0]^\top$$

also describes $\texttt{ADD-SQUARE}^k$. Thus, let $H_n = T_n^k$ in Theorem 3, the depth for $\texttt{spSQUARE}^k$ is at most n. This is a significantly large contribution since T_n^k contains about $n^2/2$ ones for almost all k for all n. In almost all k, the upper bound of the depth for $\texttt{spSQUARE}^k$ is much smaller than the upper bound of the depth for $\texttt{ADD-SQUARE}^k$ for all n, however, we choose the lesser way when we apply this in FLT-based inversion algorithms. We note that we use an inverse of $\texttt{spSQUARE}^k$ or $\texttt{ADD-SQUARE}^k$ written by $(\texttt{spSQUARE}^k)^{-1}$ or $(\texttt{ADD-SQUARE}^k)^{-1}$ when we uncompute $\langle 2^{2k} - 1\rangle$. Then, we use a reversed circuit of $\texttt{spSQUARE}^k$ or $\texttt{ADD-SQUARE}^k$. We repeatedly claim that we can apply the above depth reduction to all quantum FLT-based inversion algorithms.

When we compute or uncompute an added term, we can use $\texttt{SQUARE}^k$ which is given by applying LUP decomposition to T_n^k. Let $SQ_n^{(k)}$ denote the upper bound of the CNOT gates and the depth for $\texttt{SQUARE}^k$. In FLT-based inversion algorithms, we compare the depth of applying $\texttt{SQUARE}$ k times, i.e., kSQ_n, and the depth of $\texttt{SQUARE}^k$, i.e., $SQ_n^{(k)}$ and choose the lesser way. Figure 1 compares kSQ_n (blue line) and $SQ_n^{(k)}$ (orange line) for all n. By Figure 1, we apply $\texttt{SQUARE}$ k times when k is smaller than a threshold and apply $\texttt{SQUARE}^k$ when k is larger than that.

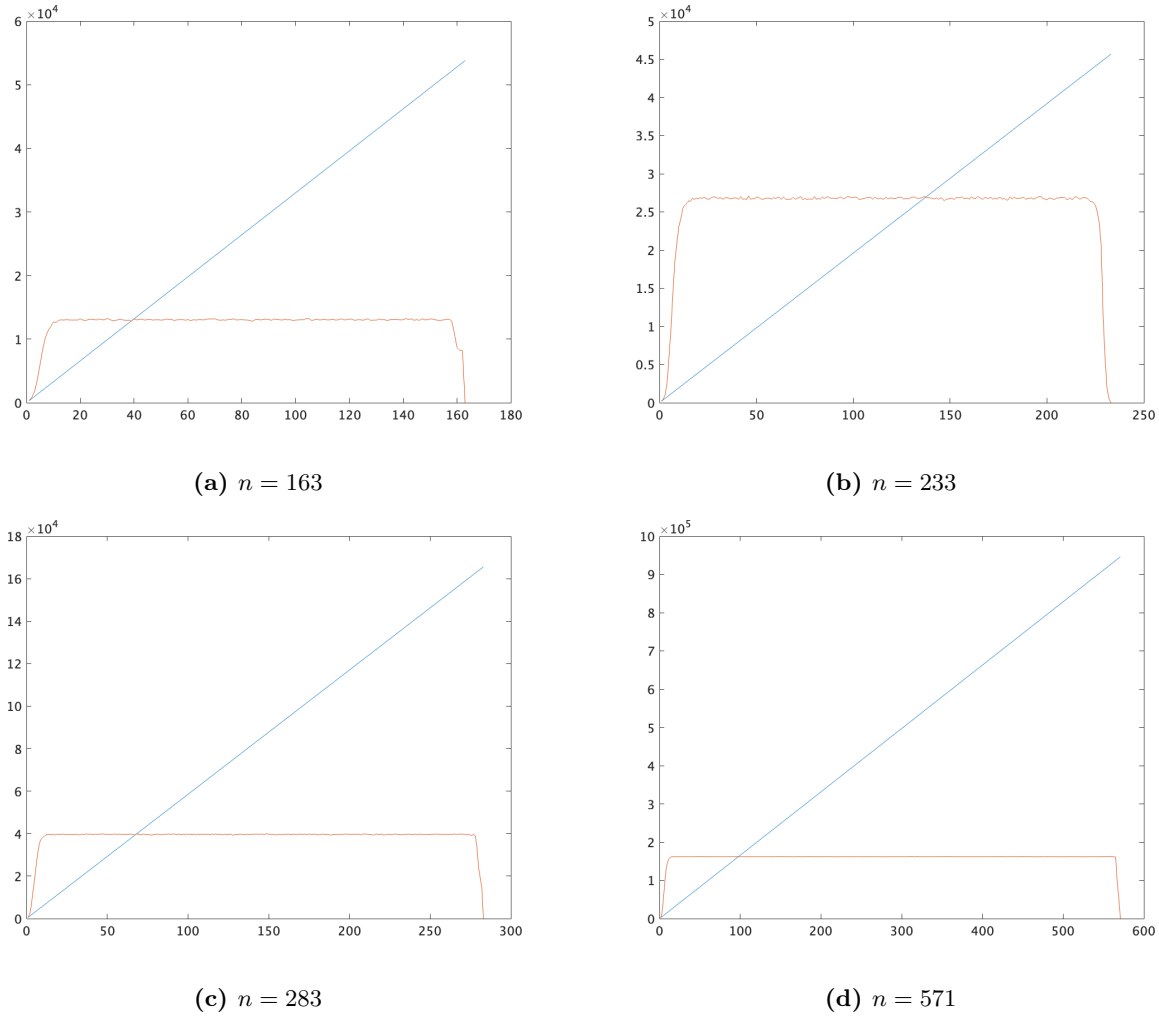
3.4 Proposed Inversion Algorithm

In this section, we construct our proposed quantum inversion algorithm that is based on the idea in Section 3.1. As we described in Section 3.1, larger $r(\tilde{p}, \tilde{\ell})$ is desired to reduce multiplications and it follows $r(\tilde{p}, \tilde{\ell}) \leq R(\tilde{p})$. However, to apply our proposed quantum point addition algorithm in Section 3.2, conditions (i) and (ii) must be satisfied. Our proposed inversion algorithm always satisfies (ii). On the other hand, (i) is satisfied if and only if $r(\tilde{p}, \tilde{\ell}) < R(\tilde{p})$. For this reason, we consider the case of $r(\tilde{p}, \tilde{\ell}) = R(\tilde{p}) - 1$ hereafter. We apply the depth reduction in squaring described in Section 3.3. Then, we prepare several sequences that describe our proposed inversion algorithm. We define two sequences $\{\tilde{a}_s\}_{s=1}^{\tilde{\ell}}, \{\tilde{b}_s\}_{s=1}^{\tilde{\ell}}$ that satisfy $\tilde{p}_{\tilde{a}_s} \in S(\tilde{p}, s-1)$ and $\tilde{p}_{\tilde{b}_s} \in S(\tilde{p}, s-1)$, and $\tilde{p}_s = \tilde{p}_{\tilde{a}_s} + \tilde{p}_{\tilde{b}_s}$ for all $1 \leq s \leq \tilde{\ell}$, where $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ is a register-bounded addition chain for $n-1$. We assume that $\tilde{a}_s = \tilde{b}_s$ if and only if $\tilde{p}_s$ is a doubled term of $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$. For the register-bounded addition chain $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$, we define two sets

$$D \coloneqq \{s \in \{1, 2, \ldots, \tilde{\ell}\} \mid \tilde{a}_s = \tilde{b}_s\},$$
$$M \coloneqq \{s \in \{1, 2, \ldots, \tilde{\ell}\} \mid \tilde{a}_s \neq \tilde{b}_s\}.$$

Now, we consider the general case of computation or uncomputation of $\langle 2^{\tilde{p}_s} - 1\rangle$ by using $\langle 2^{\tilde{p}_{\tilde{a}_s}} - 1\rangle$ and $\langle 2^{\tilde{p}_{\tilde{b}_s}} - 1\rangle$ for all $1 \leq s \leq \tilde{\ell}$ since we explained only a simple case. More precisely, we compute or uncompute $\langle 2^{\tilde{p}_s} - 1\rangle^{2^{\gamma_s}}$ in the h_3-th register by using $\langle 2^{\tilde{p}_{\tilde{a}_s}} - 1\rangle^{2^{\alpha_s}}$ in the h_1-th register and $\langle 2^{\tilde{p}_{\tilde{b}_s}} - 1\rangle^{2^{\beta_s}}$ in the h_2-th register, where $\alpha_s, \beta_s, \gamma_s$ are integers for all $s = 1, \ldots, \tilde{\ell}$. We decide that $\gamma_s = 0$ when we compute $\langle 2^{\tilde{p}_s} - 1\rangle$ and $\alpha_s = \beta_s$ when $\tilde{p}_s$ is a doubled term. Then, we define the sequences $\{\tilde{Q}_s^{(a)}\}_{s=1}^{\tilde{\ell}}, \{\tilde{Q}_s^{(b)}\}_{s=1}^{\tilde{\ell}}, \{\tilde{Q}_s\}_{s=1}^{\tilde{\ell}}$ such that $\tilde{Q}_s^{(a)}, \tilde{Q}_s^{(b)}, \tilde{Q}_s$ describe the times to apply squaring or its inverse to the h_1-th register, the h_2-th register, h_3-th register in the s-th procedure, respectively. In this case, it holds that $\tilde{Q}_s^{(a)} = -\alpha_s, \tilde{Q}_s^{(b)} = \tilde{p}_{\tilde{a}_s} - \beta_s$, and $\tilde{Q}_s = -\gamma_s$ by observing

$$\left(\langle 2^{\tilde{p}_{\tilde{a}_s}} - 1\rangle^{2^{\alpha_s}}\right)^{2^{-\alpha_s}} \times \left(\langle 2^{\tilde{p}_{\tilde{b}_s}} - 1\rangle^{2^{\beta_s}}\right)^{2^{\tilde{p}_{\tilde{a}_s} - \beta_s}} = \langle 2^{\tilde{p}_{\tilde{a}_s} + \tilde{p}_{\tilde{b}_s}} - 1\rangle = \langle 2^{\tilde{p}_s} - 1\rangle,$$
$$\left(\langle 2^{\tilde{p}_s} - 1\rangle^{2^{\gamma_s}}\right)^{2^{-\gamma_s}} = \langle 2^{\tilde{p}_s} - 1\rangle.$$



(a) $n = 163$ **(b)** $n = 233$

(c) $n = 283$ **(d)** $n = 571$

Figure 1: The upper bound of the depth for computing 2^k-th power

As we described in a proof of Theorem 2, we can construct a quantum algorithm that computes or uncomputes by the above two relations based on a register-bounded addition chain $\{\tilde{p}_s\}_{s=0}^{\ell}$.

We describe our proposed algorithm in Algorithm 2 which takes a register-bounded addition chain $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ for $n-1$ of length $\tilde{\ell}$ and sequences $\{\tilde{a}_s\}_{s=1}^{\tilde{\ell}}$,$\{\tilde{b}_s\}_{s=1}^{\tilde{\ell}}$,$\{\tilde{Q}_s^{(a)}\}_{s=1}^{\tilde{\ell}}$,$\{\tilde{Q}_s^{(b)}\}_{s=1}^{\tilde{\ell}}$,$\{\tilde{Q}_s\}_{s=1}^{\tilde{\ell}}$ as input. $\texttt{caseOPTSQUARE}(g,v)$ applies $\texttt{ca-}$ $\texttt{seSQUARE}(g,v)$ if $|v|SQ_n < SQ_n^{(|v|)}$ and applies $\texttt{caseSQUARE}^v(g)$ otherwise, where $\texttt{caseSQUARE}\,(g,v)$ applies $\texttt{SQUARE}$ v times to g when $v>0$, applies $\texttt{SQUARE}^{-1}$ $-v$ times to g when $v<0$, and do nothing when $v=0$ and $\texttt{caseSQUARE}^v(g)$ applies $\texttt{SQUARE}^v(g)$ when $v>0$, applies $(\texttt{SQUARE}^{-v})^{-1}(g)$ when $v<0$, and do nothing when $v=0$. $\texttt{caseOPTspSQUARE}(g_1,g_2,v)$ applies $\texttt{ADD}(g_1,g_2)$ and $\texttt{caseSQUARE}(g_2,v)$ if $1+|v|SQ_n < n$ and applies $\texttt{casespSQUARE}^v(g_1,g_2)$ otherwise, where $\texttt{casespSQUARE}^v(g_1,g_2)$ applies $\texttt{spSQUARE}^v(g_1,g_2)$ when $v>0$, applies $(\texttt{spSQUARE}^{-v})^{-1}(g_1,g_2)$ when $v<0$, and applies $\texttt{ADD}(g_1,g_2)$ when $v=0$. We note that a $(\texttt{SQUARE}^v)^{-1}$ circuit, a $(\texttt{spSQUARE}^v)^{-1}$ circuit, and a $\texttt{caseOPTspSQUARE}^{-1}$ circuit are a reversed circuit of $\texttt{SQUARE}^v$, a reversed circuit of $\texttt{spSQUARE}^v$, and a reversed circuit of $\texttt{caseOPTspSQUARE}^v$, respectively. pl$[s]$ stores the register number which stores $f^{2^{\tilde{p}_s}-1}$ for all $1 \le s \le \tilde{\ell}$. pld$[s]$ stores the register number which stores the copy of $f^{2^{\tilde{p}_{\tilde{a}_s}}-1}$ for all $s \in D$. The size of pld, i.e., $\tilde{d}$ equals $\#D$. We note that $\tilde{p}_{\tilde{\ell}}$ does not always equal $n-1$. In other words, $g_{\mathrm{pl}[\tilde{\ell}]}$ does not always store $\langle 2^{n-1}-1\rangle$. Then, we define the non-negative integer ω such that $g_{\mathrm{pl}[\omega]}$ stores $\langle 2^{n-1}-1\rangle$ at the end of the loop from line 2 to line 16. By $\texttt{SWAP}$ procedure in line 18, $\langle 2^n-2\rangle = \langle -1\rangle$ is always stored in g_1. However, this procedure can be abbreviated because we can

Algorithm 2 Proposed inversion algorithm

Input: An irreducible polynomial $m(x) \in \mathbb{F}_2[x]$ of degree n, a register-bounded addition chain $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$, sequences $\{\tilde{a}_s\}_{s=1}^{\tilde{\ell}}, \{\tilde{b}_s\}_{s=1}^{\tilde{\ell}}, \{\tilde{Q}_s^{(a)}\}_{s=1}^{\tilde{\ell}}, \{\tilde{Q}_s^{(b)}\}_{s=1}^{\tilde{\ell}}, \{\tilde{Q}_s\}_{s=1}^{\tilde{\ell}}$, a register g_0 which stores a polynomial $f \in \mathbb{F}_{2^n}^*$ of degree up to $n-1$, registers $g_1, \ldots, g_{R(\tilde{p})}$ initialized to an all-$|0\rangle$ state, arrays $\mathrm{pl}[\tilde{\ell}]$, $\mathrm{pld}[\tilde{d}]$, a non-negative integer ω which satisfies $\tilde{p}_\omega = n-1$

Output: $g_1 = f^{2^n-2}$

$dcount \leftarrow 0$
for $s = 1, \ldots, \tilde{\ell}$ **do**
  if $s \in D$ **then**
    caseOPTspSQUARE $(g_{\mathrm{pl}[\tilde{a}_s]}, g_{\mathrm{pld}[dcount]}, \tilde{Q}_s^{(b)})$
    caseOPTSQUARE $(g_{\mathrm{pl}[\tilde{a}_s]}, \tilde{Q}_s^{(a)})$
    caseOPTSQUARE $(g_{\mathrm{pl}[s]}, \tilde{Q}_s)$
    MODMULT $(g_{\mathrm{pl}[\tilde{a}_s]}, g_{\mathrm{pld}[dcount]}, g_{\mathrm{pl}[s]})$
    caseOPTspSQUARE^{-1} $(g_{\mathrm{pl}[\tilde{a}_s]}, g_{\mathrm{pld}[dcount]}, \tilde{p}_{\tilde{a}_s})$
    $dcount \leftarrow dcount + 1$
  else // $s \in M$
    caseOPTSQUARE $(g_{\mathrm{pl}[\tilde{a}_s]}, \tilde{Q}_s^{(a)})$
    caseOPTSQUARE $(g_{\mathrm{pl}[\tilde{b}_s]}, \tilde{Q}_s^{(b)})$
    caseOPTSQUARE $(g_{\mathrm{pl}[s]}, \tilde{Q}_s)$
    MODMULT $(g_{\mathrm{pl}[\tilde{a}_s]}, g_{\mathrm{pl}[\tilde{b}_s]}, g_{\mathrm{pl}[s]})$
SQUARE $(g_{\mathrm{pl}[\omega]})$
SWAP $(g_{\mathrm{pl}[\omega]}, g_1)$

change the registers in advance such that $\mathrm{pl}[\omega] = 1$. In Section 4, we explain our choices of ℓ and $R(\tilde{p})$ and show $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ for all n.

Finally, we describe the number of qubits of our method and Banegas et al.'s method for Shor's algorithm. Banegas et al. [BBvHL20] showed the number of qubits for Shor's algorithm using their quantum point addition algorithm with their quantum GCD-based inversion algorithm is $7n + \lfloor \log n \rfloor + 9$ for all n. Then, we show the number of qubits for our method for Shor's algorithm in Theorem 5

Theorem 5. *The number of qubits for using Algorithm 1 as a point addition with Algorithm 2 as an inversion algorithm which takes $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ as an input is given by $(2 + R(\tilde{p}))n + 1$.*

Therefore, if we find a register-bounded addition chain with $R(\tilde{p}) \leq 5$, our method achieves fewer qubits than Banegas et al.'s GCD-based method.

4 Comparison

In Section 4.1, we explain our choice of register-bounded addition chains and compare the number of qubits for an inversion. In Section 4.2, we describe the trade-off for our proposed inversion algorithm. In Section 4.3, we compare the quantum resources in a whole Shor's algorithm between our proposed method and previous methods.

Difference from Preliminary Version. As mentioned in Section 1.2, we use the exact value of depth for spSQUARE^k while we use an upper bound in [TT24]. Thus, we update the Tables and Figures in this Section.

4.1 Our Choice of Register-Bounded Addition chains

As we showed in Theorem 2, the number of ancillary registers and the number of multiplications for our proposed inversion algorithm depends on $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ for $n-1$. In particular, the number of ancillary registers

Table 1: Our choice of register-bounded addition chains $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$

n	Register-bounded addition chains
163	$\{\tilde{p}_s\}_{s=0}^{14} = \{1, 2, 3, 6, 9, 6, 3, 2, 18, 27, 54, 27, 18, 108, 162\}$
233	$\{\tilde{p}_s\}_{s=0}^{16} = \{1, 2, 3, 4, 7, 4, 3, 2, 14, 28, 29, 28, 14, 58, 116, 58, 232\}$
283	$\{\tilde{p}_s\}_{s=0}^{18} = \{1, 2, 3, 6, 9, 15, 9, 6, 3, 30, 45, 47, 45, 30, 2, 94, 141, 94, 282\}$
571	$\{\tilde{p}_s\}_{s=0}^{20} = \{1, 2, 3, 4, 7, 4, 3, 2, 14, 28, 29, 57, 29, 28, 14, 114, 171, 285, 171, 114, 570\}$

Table 2: Comparison of ℓ, R and the number of qubits for an inversion between ours and prior works

n	Proposed algorithm			BBHL21-GCD			KH23-GCD			TT23-FLT		
	ℓ	R	qubits	ℓ	R	qubits	ℓ	R	qubits	ℓ	R	qubits
163	9	5	978	-	-	830	-	-	690	9	9	1,630
233	10	5	1,398	-	-	1,180	-	-	970	10	10	2,563
283	11	5	1,698	-	-	1,431	-	-	1,174	11	11	3,396
571	12	5	3,426	-	-	2,872	-	-	2,330	12	12	7,423

equals $R(\tilde{p})$, and the number of multiplications equal $\tilde{\ell}$, where ℓ is the length of an addition chain $\{p_s\}_{s=0}^{\ell}$ for $n-1$ which consists of the different terms in $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$. As we described in Section 3.1, we consider the case of $r(\tilde{p}, \tilde{\ell}) = 2\ell - \tilde{\ell} = R(\tilde{p}) - 1$. In this situation, we reduce the number of qubits as much as possible, in other words, we find register-bounded addition chains $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ with as small $R(\tilde{p})$ as possible. For this purpose, we find the shortest addition chains for $n-1$ at first. After that, we add some terms to the shortest addition chains and get register-bounded addition chains. Thus, we find some register-bounded addition chains with as small $R(\tilde{p})$ as possible.

In Table 1, we show our register-bounded addition chains $\{\tilde{p}_s\}_{s=0}^{\tilde{\ell}}$ for NIST-recommended degrees $n = 163, 233, 283$, and 571. In Table 2, we show ℓ, R and the number of qubits for our proposed inversion algorithm and previous quantum inversion algorithms, i.e., Banegas et al.'s quantum GCD-based inversion algorithm which we call BBHL21-GCD, Kim-Hong's quantum GCD-based inversion algorithm which we call KH23-GCD, and Taguchi-Takayasu's quantum FLT-based inversion algorithm which we call TT23-FLT for all n. R for our proposed algorithm is minimum $R(\tilde{p})$. We do not compare quantum FLT-based inversion algorithms proposed by Putranto et al. [PWLK22] and Banegas et al [BBvHL20] since Taguchi-Takayasu's FLT-based Basic and Extended algorithm reduce all quantum resources compared to them. We also do not compare Taguchi-Takayasu's Basic algorithm since their Extended algorithm requires fewer qubits than Basic algorithm. The number of qubits in Table 2 includes an input $f \in \mathbb{F}_{2^n}^*$. Table 2 indicates the minimum $R = R(\tilde{p})$ of register-bounded addition chains for $n-1$ is 5 for all NIST-recommended n when we use shortest addition chains for $n-1$. Then, our proposed algorithm achieves the fewest qubits compared to the previous quantum FLT-based inversion algorithms, however, it is still larger than the number of both GCD-based algorithms for all cases.

4.2 Quantum Resources Trade-off in Our Proposed Inversion Algorithm

In Section 4.1, we showed register-bounded addition chains with $R(\tilde{p}) = 5$ for all n, where $R(\tilde{p})$ describes the number of ancillary registers. On the other hand, TT23-FLT requires ℓ ancillary registers, where ℓ is the length of shortest addition chains for $n-1$. As we described in Table 2, $\ell = 9, 10, 11, 12$ when $n = 163, 233, 283, 571$, respectively. Then, we also consider all possible cases, i.e., $R(\tilde{p}) = 5, 6, \ldots, \ell$ for our proposed inversion algorithm for all n and estimate the quantum resources. We note that $R(\tilde{p}) = \ell$ is not the case of TT23-FLT since $r(\tilde{p}, \tilde{\ell}) = R(\tilde{p}) - 1 = \ell - 1$ for our proposed algorithm. In other words, our proposed algorithm has a clear ancillary register at the end of the algorithm, while TT23-FLT has no clear ancillary register at the end of the algorithm. In Section 4.3, we show which $R(\tilde{p})$ is preferable in some parameters.

4.3 Comparison with Previous Methods in Shor's Algorithm

In this section, we compare the quantum resources of our method for Shor's algorithm, i.e., our proposed quantum inversion algorithm in Section 3.3 with our proposed quantum point addition algorithm described in Algorithm 1 and previous methods, i.e., BBHL21-GCD, KH23-GCD, and TT23-FLT with Banegas et al.'s quantum point addition algorithm, since previous three algorithms do not satisfy the conditions (i) and (ii) in Section 3.2.

Table 3: Quantum resources our proposed method for Shor's algorithm in each $R(\tilde{p})$

$n = 163$

		qubits	TOF	depth	CNOT
$R(\tilde{p})$	5	1,142	19,682,952	231,367,264	1,672,852,808
	6	1,305	18,381,448	213,817,952	1,537,254,984
	7	1,468	17,079,944	199,044,832	1,431,709,832
	8	1,631	15,778,440	185,195,360	1,330,132,168
	9	1,794	14,476,936	171,799,840	1,230,076,424

$n = 233$

		qubits	TOF	depth	CNOT
$R(\tilde{p})$	5	1,632	46,185,516	530,966,124	5,557,595,472
	6	1,865	43,487,964	498,462,588	5,265,803,088
	7	2,098	40,790,412	466,936,236	4,886,412,336
	8	2,331	38,092,860	435,739,356	4,495,044,528
	9	2,564	35,395,308	405,070,380	4,205,086,704
	10	2,797	32,697,756	374,240,412	3,897,577,008

$n = 283$

		qubits	TOF	depth	CNOT
$R(\tilde{p})$	5	1,982	77,493,944	1,163,334,432	10,840,880,376
	6	2,265	73,440,696	1,050,009,344	10,086,914,904
	7	2,548	69,387,448	979,225,184	9,376,194,680
	8	2,831	65,334,200	915,836,384	8,853,416,568
	9	3,114	61,280,952	865,461,600	8,324,617,656
	10	3,397	57,227,704	814,003,072	7,810,516,312
	11	3,680	53,174,456	761,247,232	7,298,371,160

$n = 571$

		qubits	TOF	depth	CNOT
$R(\tilde{p})$	5	3,998	368,373,720	9,725,226,368	95,224,517,960
	6	4,569	350,925,432	9,007,389,248	89,353,935,528
	7	5,140	333,477,144	7,551,521,120	84,233,455,592
	8	5,711	316,028,856	7,258,300,192	78,814,533,512
	9	6,282	298,580,568	6,972,789,824	74,298,854,344
	10	6,853	281,132,280	6,470,235,200	70,131,687,912
	11	7,424	263,683,992	6,149,658,944	66,146,499,848
	12	7,995	246,235,704	5,866,994,848	62,199,223,944

Here, we concretely estimate the quantum resources, i.e., the number of qubits, TOF gates, and depth of our method and previous methods for Shor's algorithm. We also compute the number of CNOT gates, however, we note that a CNOT gate is much cheaper than a TOF gate. We note that Shor's algorithm requires $2n + 2$ point additions. As Roetteler et al. [RNSL17] mentioned, we can ignore the special cases of point addition since it does not affect quantum Fourier transform. Moreover, we apply semiclassical Fourier

Table 4: Comparison of the number of qubits, TOF gates, depth, and CNOT gates for Shor's algorithm between ours and prior works

n	Proposed method			
	qubits	TOF	depth	CNOT
163	1,142	19,682,952	231,367,264	1,672,852,808
233	1,632	46,185,516	530,966,124	5,557,595,472
283	1,982	77,493,944	1,163,334,432	10,840,880,376
571	3,998	368,373,720	9,725,226,368	95,224,517,960
n	BBHL21-GCD method			
	qubits	TOF	depth	CNOT
163	1,157	288,641,640	341,963,616	322,348,232
233	1,647	772,092,828	945,129,276	926,188,848
283	1,998	1,359,458,584	1,672,107,936	1,644,678,648
571	4,015	10,156,396,536	12,962,714,336	13,091,280,488
n	KH23-GCD method			
	qubits	TOF	depth	CNOT
163	1,017	243,048,328	319,284,384	391,632,328
233	1,437	694,262,556	898,421,004	1,128,567,024
283	1,741	1,237,627,128	1,594,550,944	2,006,665,048
571	3,473	9,942,884,952	12,608,046,880	16,064,737,832
n	TT23-FLT method			
	qubits	TOF	depth	CNOT
163	1,957	13,175,432	182,158,080	1,121,173,864
233	3,030	30,000,204	421,008,588	3,302,850,096
283	3,963	49,121,208	940,573,920	6,556,415,480
571	8,566	228,787,416	6,723,013,440	55,292,822,728

transform [GN96] in Shor's algorithm since it requires only 1 qubit. Our proposed inversion algorithm uses the register-bounded addition chains of Table 1. For estimating the resources of TT23-FLT, we use addition chains that Taguchi and Takayasu [TT23] used. Values of the depth are upper bounds because we do not completely consider parallel quantum computation. Moreover, we compute the concrete number of CNOT gates of `SQUARE`, $\texttt{SQUARE}^{-1}$, and `spSQUARE`, and assume that `const_ADD` requires $n/2$ X gates on average, `ctrl_ADD` requires $n/2$ TOF gates on average, and `ctrl_const_ADD` requires $n/2$ CNOT gates on average. We estimate the upper bound of the depth of `SQUARE` as the number of CNOT gates for `SQUARE`, while we estimate the depth of `spSQUARE` as described in Section 3.3. We also apply the depth reduction for `SQUARE` and `spSQUARE` described in Section 3.3 to the TT23-FLT method and estimate the quantum resources. When we estimate the upper bound of the depth for Shor's algorithm, we simply add the upper bound of the depth for each distinct quantum computation. Banegas et al. [BBvHL20] applied windowing that reduces the number of TOF gates by using some lookups from a QROM and estimated the number of TOF gates. We note that we can also apply windowing our proposed method while we do not estimate the quantum resources. We provide a python code [Tag23] for computing quantum resources.

Table 3 compares the number of qubits, TOF gates, depth and CNOT gates for our proposed method in each $R(\tilde{p})$ for all n. Table 4 compares the number of qubits, TOF gates, depth, and CNOT gates in all cases for all n. In Table 4, we show the quantum resources in the case of $R(\tilde{p}) = 5$ for our proposed method. We compare our proposed method with the previous GCD-based methods and the FLT-based method.

Comparison with the GCD-based Methods. The number of qubits for our proposed method is close to the GCD-based methods, i.e., the BBHL21-GCD method and the KH23-GCD method for all n. Especially, our proposed method achieves fewer qubits than the BBHL21-GCD method, while it does not for an inversion as shown in Table 2. As described in Section 3.2, our proposed method for Shor's algorithm requires $(2 + R(\tilde{p}))n + 1$ qubits. Furthermore, we found register-bounded addition chains with $R(\tilde{p}) = 5$ for all n.

Table 5: Comparison of QD = "the number of qubits" × "depth" in Shor's algorithm between ours and prior works

n	QD			
	Proposed method	BBHL21-GCD method	KH23-GCD method	TT23-FLT method
163	$2.64 \cdot 10^{11}$	$3.96 \cdot 10^{11}$	$3.25 \cdot 10^{11}$	$3.56 \cdot 10^{11}$
233	$8.67 \cdot 10^{11}$	$1.56 \cdot 10^{12}$	$1.29 \cdot 10^{12}$	$1.28 \cdot 10^{12}$
283	$2.31 \cdot 10^{12}$	$3.34 \cdot 10^{12}$	$2.78 \cdot 10^{12}$	$3.73 \cdot 10^{12}$
571	$3.89 \cdot 10^{13}$	$5.20 \cdot 10^{13}$	$4.38 \cdot 10^{13}$	$5.76 \cdot 10^{13}$

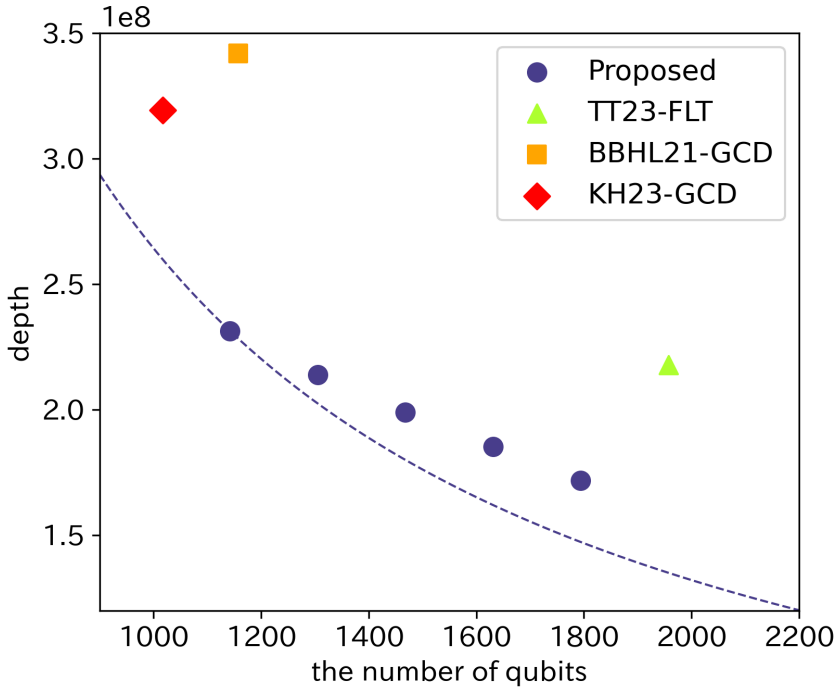
Table 6: Comparison of QT = "the number of qubits" × "the number of TOF gates" in Shor's algorithm between ours and prior works

n	QT			
	Proposed method	BBHL21-GCD method	KH23-GCD method	TT23-FLT method
163	$2.25 \cdot 10^{10}$	$3.34 \cdot 10^{11}$	$2.47 \cdot 10^{11}$	$2.58 \cdot 10^{10}$
233	$7.54 \cdot 10^{10}$	$1.27 \cdot 10^{12}$	$9.98 \cdot 10^{11}$	$9.09 \cdot 10^{10}$
283	$1.54 \cdot 10^{11}$	$2.72 \cdot 10^{12}$	$2.15 \cdot 10^{12}$	$1.95 \cdot 10^{11}$
571	$1.47 \cdot 10^{12}$	$4.08 \cdot 10^{13}$	$3.45 \cdot 10^{13}$	$1.96 \cdot 10^{12}$

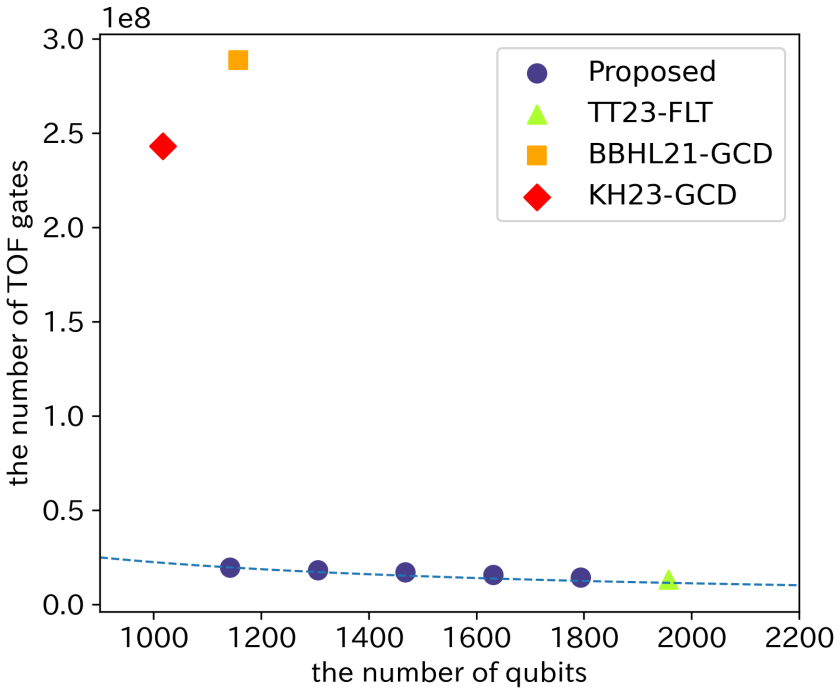
Then, the number of qubits is $7n + 1$ and it is smaller than the number of qubits for the BBHL21-GCD method, i.e., $7n + \lfloor \log n \rfloor + 9$. The KH23-GCD method requires fewer qubits than our proposed method, however, the difference is less than n. Precisely, the KH23-GCD method requires $6n + 4\lfloor \log n \rfloor + 11$ qubits and the difference to $7n + 1$ is $n - 4\lfloor \log n \rfloor - 10$. Furthermore, our proposed method still achieves much fewer TOF gates and less depth compared to the GCD-based methods while we halve the number of qubits from the TT23-FLT method. The number of TOF gates of our proposed method is from only 2% to 5% of the number of the GCD-based methods. As for the depth, the depth reduction of the squaring part in our algorithm in Section 3.3 contributes to keeping fewer than the GCD-based methods.

Comparison with the TT23-FLT Method. Our proposed method drastically reduces the number of qubits from the TT23-FLT method. Precisely, we halve the qubits for all n. Our proposed inversion algorithm applies additional procedures for uncomputations which require TOF gates, depth, and CNOT gates to TT23-FLT. When $n = 571$, our proposed inversion algorithm requires 8 additional procedures which is about 70% of the number of procedures for TT23-FLT. As you can see in Table 4, the number of TOF gates and CNOT gates for our proposed method is about 170% of the number for the TT23-FLT method.

By using the concrete number of quantum resources, we compute two values, i.e., "the number of qubits" × "depth" called QD and "the number of qubits" × "the number of TOF gates" called QT. QD is a same metric to "spacetime volume" by Gidney and Ekerå [GE21] and QT is a similar metric. Gidney and Ekerå used spacetime volume to evaluate Shor's algorithm for solving a factoring problem. Briefly speaking, QD and QT describe how a quantum algorithm works better on both the number of qubits and the number of TOF gates and both the number of qubits and depth, respectively. We show QD and QT for our proposed method and previous methods for all n in Table 5, 6, respectively. We also illustrate the relation between the number of qubits and depth and between the number of qubits and the number of TOF gates of our proposed method and the previous methods for all n in Figures 2, 3, 4, 5. $R(\tilde{p}) = 5, 6, \ldots$, from the left point in Figures 2, 3, 4, 5. Blue lines in Figures 2, 3, 4, 5 describe the points that QD = const. and QT = const. In Table 5, 6, $R(\tilde{p}) = 5, 5, 5, 7$ and $R(\tilde{p}) = 5, 5, 5, 5$ for our proposed method for $n = 163, 233, 283, 571$, respectively. Our proposed method achieves the fewest QD and QT compared to the previous method for all n. Thus, our proposed algorithm gives good trade-offs between the number of qubits and depth and between the number of qubits and the number of TOF gates.

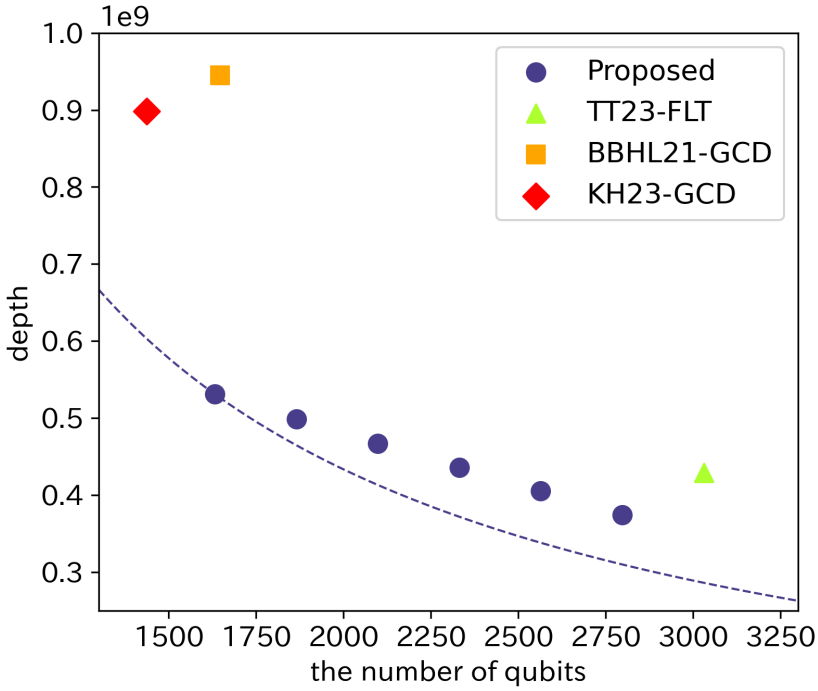


(a) qubits-depth

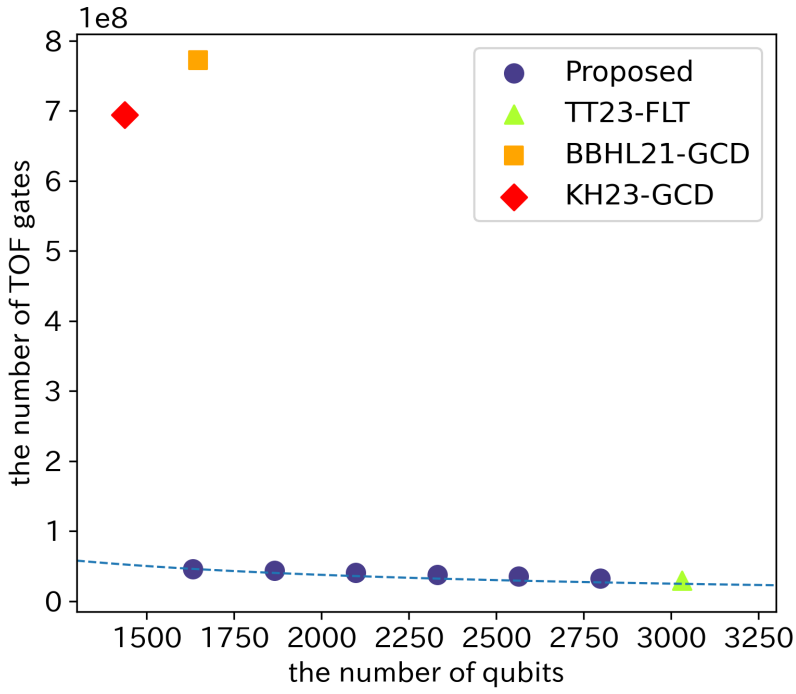


(b) qubits-TOF

Figure 2: Quantum resources trade-off in all methods for $n = 163$

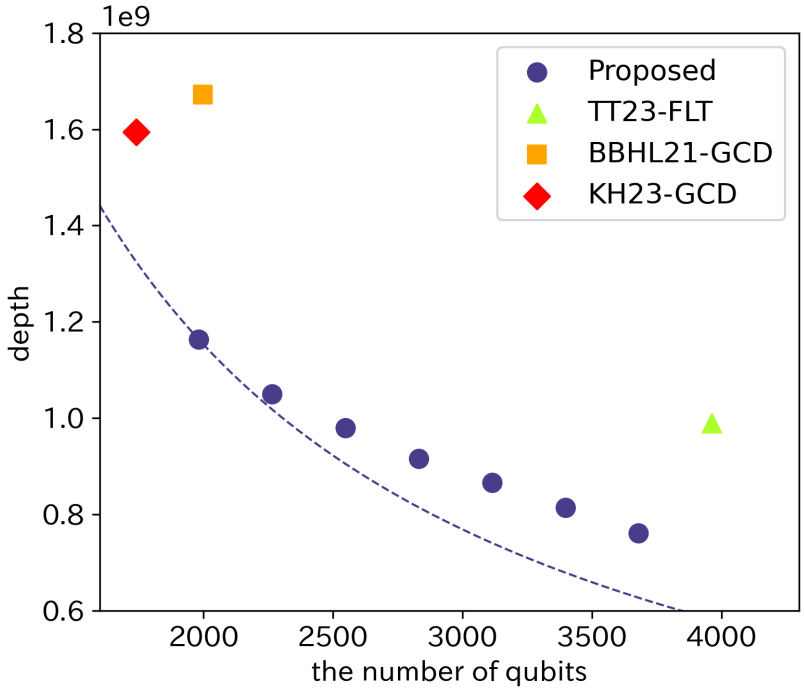


(a) qubits-depth

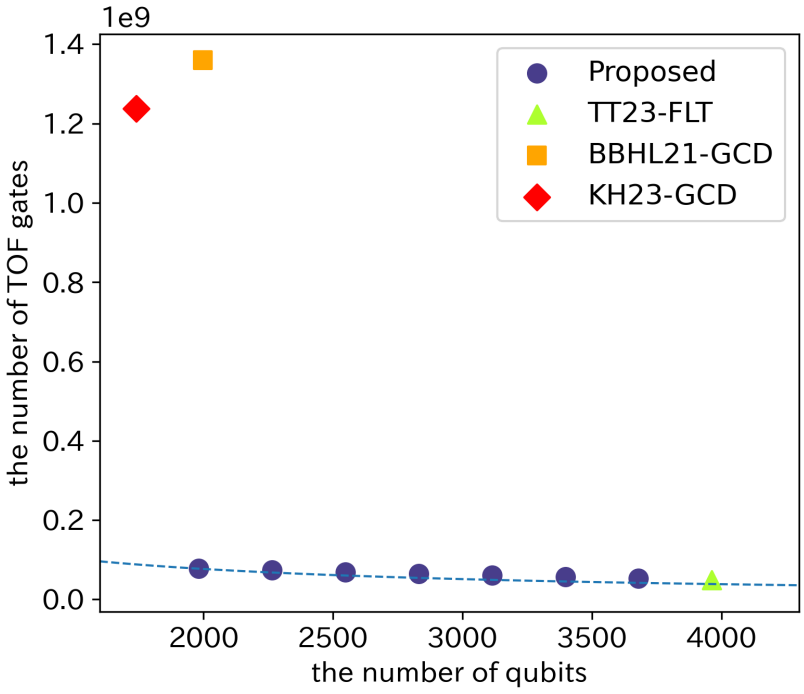


(b) qubits-TOF

Figure 3: Quantum resources trade-off in all methods for $n = 233$

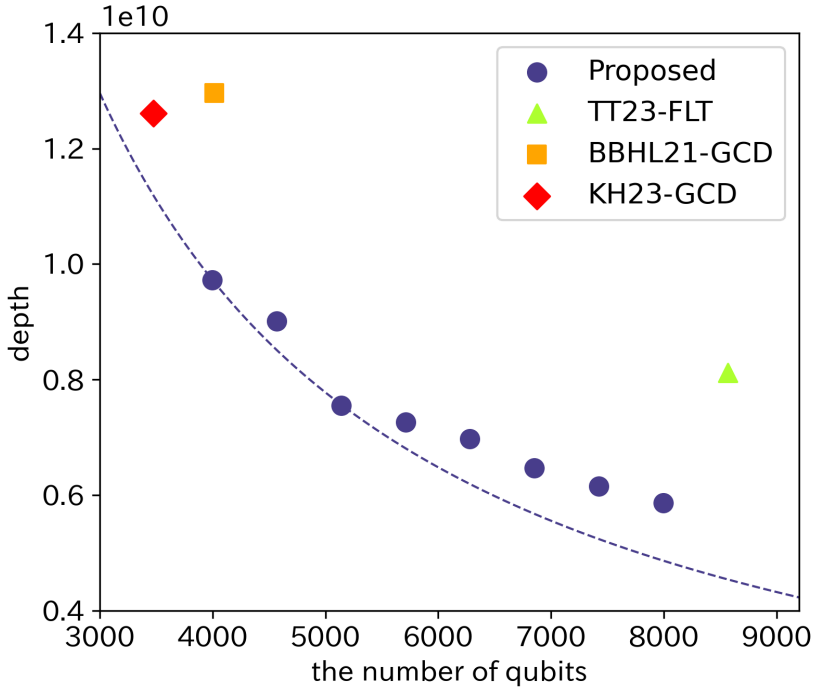


(a) qubits-depth

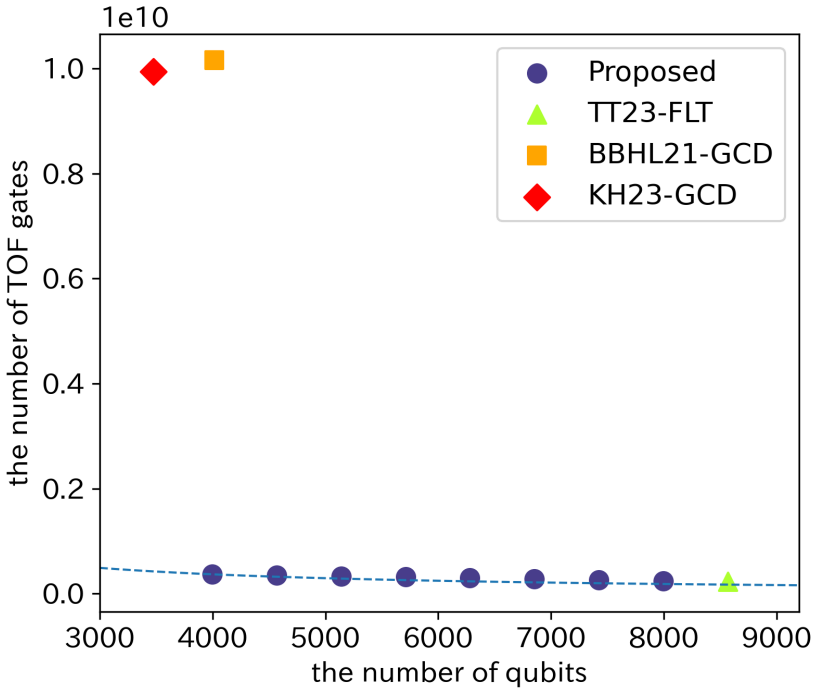


(b) qubits-TOF

Figure 4: Quantum resources trade-off in all methods for $n = 283$



(a) qubits-depth



(b) qubits-TOF

Figure 5: Quantum resources trade-off in all methods for $n = 571$

Comparison with the FLT-based methods without depth reduction of squaring.

Table 7 shows the upper bound of the depth and the number of CNOT gates of our proposed method and the TT23-FLT method for Shor's algorithm without depth reduction of squaring. Table 7 does not contain the number of qubits and TOF gates since it does not depend on whether we apply the depth reduction or not. By Table 4 and Table 7, our proposed squaring algorithms described in Section 3.3 reduce the total depth by $17 \sim 42\%$.

Table 7: The depth and the number of CNOT gates for Shor's algorithm of ours and the TT23-FLT method without depth reduction

n	Proposed method		TT23-FLT method	
	depth	CNOT	depth	CNOT
163	$329,024,672$	$1,632,333,000$	$245,155,072$	$1,114,099,560$
233	$732,314,700$	$5,145,976,368$	$505,737,180$	$3,363,231,456$
283	$1,884,653,536$	$10,624,599,608$	$1,189,712,352$	$6,709,448,312$
571	$16,669,416,192$	$93,673,235,656$	$10,548,613,504$	$58,217,280,264$

5 Windowing

Windowing is a way to skip several computations by using precomputed data. Häner et al. [HJN+20] indicated that quantum point addition on elliptic curves using windowing by QROM is also possible, and Banegas et al. [BBvHL20], Putranto et al. [PWLK22], and Taguchi-Takayasu [TT23] made use of that method. Although we omit the details, if we apply windowing to point addition with window size w, the number of point addition decreases from $2n+2$ to $2\lceil \frac{n+1}{w} \rceil + 1$. However, we require $2(2^w - 1)$ TOF gates to construct QROM. Thus, there is a w which minimizes the total number of TOF gates. We call this w an optimal window size. Table 8 shows an optimal window size and the number of TOF gates of our proposed method and the previous methods for Shor's algorithm for all n.

Table 8: Optimal window size w and the number of TOF gates for Shor's algorithm

n	Proposed method		TT23-FLT method	
	w	TOF	w	TOF
163	10	$2,501,073$	9	$1,781,025$
233	10	$5,391,359$	9	$3,679,975$
283	11	$8,464,129$	10	$5,765,145$
571	12	$35,787,477$	11	$23,375,349$
n	BBHL21-GCD method		KH23-GCD method	
	w	TOF	w	TOF
163	13	$26,303,013$	13	$22,549,905$
233	14	$64,402,483$	13	$58,401,627$
283	15	$108,252,597$	15	$99,887,409$
571	16	$704,590,641$	16	$690,966,213$

References

[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.

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

[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.

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

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

[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.

[Kön16] Dénes König. Über graphen und iher anwendung auf determinantentheorie und mengenlehre. *Math.Ann.*, 77, 1916.

[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, Cham, 1985. Springer.

[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.

[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.

[Tag23] Ren Taguchi. Quantum resource estimate for Shor's algorithm for solving binary ECDLP. Github, 2023.

[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.

[TT24] Ren Taguchi and Atsushi Takayasu. On the untapped potential of the quantum FLT-based inversion. In *Applied Cryptography and Network Security: 22th International Conference ACNS 2024*, 2024 (to appear).

[vH19] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count. Cryptology ePrint Archive, Paper 2019/1170, 2019.