Reducing the Depth of Quantum FLT-Based Inversion Circuit

Harashta Tatimma Larasati, Dedy Septono Catur Putranto, Rini Wisnu Wardhani, Howon Kim
{harashta, dedy.septono, rini.wisnu, howonkim}@pusan.ac.kr

***Abstract*—Works on quantum computing and cryptanalysis has increased significantly in the past few years. Various constructions of quantum arithmetic circuits, as one of the essential components in the field, has also been proposed. However, there has only been a few studies on finite field inversion despite its essential use in realizing quantum algorithms, such as in Shor's algorithm for Elliptic Curve Discrete Logarith Problem (ECDLP). In this study, we propose to reduce the depth of the existing quantum Fermat's Little Theorem (FLT)-based inversion circuit for binary finite field. In particular, we propose follow a complete waterfall approach to translate the Itoh-Tsujii's variant of FLT to the corresponding quantum circuit and remove the inverse squaring operations employed in the previous work by Banegas et al., lowering the number of CNOT gates (CNOT count), which contributes to reduced overall depth and gate count. Furthermore, compare the cost by firstly constructing our method and previous work's in Qiskit quantum computer simulator and perform the resource analysis. Our approach can serve as an alternative for a time-efficient implementation.**

***Index Terms*—quantum cryptanalysis, inversion, binary field, simulation.**

I. Introduction

The study on quantum computing has been emerging, particularly after Peter Shor demonstrated the apparent advantage of using quantum phenomena to speed up computation and crack classically intractable problems underlying current public-key cryptosystems, later known as the Shor's algorithm [1, 2]. In addition, research in quantum hardware has seen significant advancement in the past few years, continually achieving an increased number of physical qubits every few years [3, 4]. This accelerates the potential use of quantum computers in Noisy Intermediate-Scale Quantum (NISQ) era, and later in the Fault Tolerant Quantum Computation (FTQC). Inevitably, the construction of efficient quantum arithmetic circuits, i.e., circuits to perform arithmetic operations in a quantum computer, becomes essential.

To date, numerous research efforts have proposed the quantum circuit implementations of arithmetic operations. Driven by the interest in Shor's algorithm for factoring —the variant to crack Rivest–Shamir-Adleman (RSA)-based cryptosystems, early research [5, 6] had focused on designing an explicit construction of quantum circuits for performing *modular exponentiation*, one of the essential components comprised in the algorithm. The circuits include specialized circuits and their underlying components, such as adder, modular adder, and modular multiplier by a constant. They are further improved in the later research customized to certain properties of modular arithmetic operations to obtain additional optimizations, e.g., [7–10].

Apart from for factoring, the other variant of Shor's algorithm as the basis for cracking Elliptic Curve Cryptography (ECC)-based cryptosystem has also been studied, calling for concrete construction of its underlying circuit as well, e.g., [11–14], though not as flourished. For this case, the circuit building blocks pose differences from the previous algorithm variant. Instead of a *modular exponentiation*, Shor's algorithm for Elliptic Curve Discrete Logarithm Problem (ECDLP) requires a *scalar multiplication*, which is further can be broken down into a series of *point addition* circuit. One of the essential component of the point addition circuit is the *inversion* block, used together with multiplication block to perform a *division* in the finite field case. In terms of the type of finite field and its respective elliptic curve, the works in quantum cryptanalysis of Shor's ECDLP can be classified into two main areas: for prime elliptic curves, such as [11–13] and for binary elliptic curves [14–16].

In terms of inversion operation, there has only been a few methods proposed for the quantum case, some of which are based on the classical extended Euclidean algorithm (EEA) [11] and its variant [12, 13], extended greatest common divisor (GCD) [14], and Fermat's Little Theorem (FLT) [14]. As for the binary elliptic curves, the latter two are the most recent with an explicit quantum circuit construction. Compared to other arithmetic operations such as addition and multiplication, the amount of proposals to improve inversion circuit is very minimal despite its significance in the quantum cryptanalysis and quantum arithmetic in general. To meet different needs, e.g., space-efficient implementation, time-efficient implementation, or combination of both), an alternative to the existing approach will be very beneficial.

In this paper, we propose to reduce the depth of the FLT-based quantum inversion circuit for binary elliptic curves. We depart from the previous work by Banegas et al. [14], which interprets the second variant of classical FLT-based Itoh-Tsujii algorithm to the corresponding quantum circuit, adjusting their algorithm for a more time-efficient implementation. In particular, we employ a complete waterfall approach and alter the first and second stage of their algorithm, relocating the uncomputation to the end, thus eliminating the intermediate uncomputation (i.e., repeated inverse squarings) used in the previous work and minimizing the number of CNOT gates in the circuit. As a result, an overall lower depth and gate count can be achieved. Additionally, for verification and comparison, we build the code for both our proposed inversion method and the previous work in Qiskit quantum computer simulator, then perform the resource analysis for the inversion bit size n of 8, 16, 127, 163, and for the use case of binary elliptic curve standard: B-233, B-283, B-409, and B-571 as, for the context of quantum cryptanalysis. Note that our proposed method comes with a tradeoff of higher ancilla qubits. Nevertheless, by the fact that the depth corresponds to the time complexity of a quantum circuit [17] and considering the potential benefit later discussed in Section IV-B, this work is advantageous for a time-efficient implementation.

II. Preliminaries

A. Binary Finite Fields in Classical & Quantum Computing

Finite fields, or fields with a finite number of elements, are commonly employed in cryptographic applications such as symmetric and asymmetric key cryptosystems and have applications in various other domains such as network coding and error control theory [18]. The binary field GF(2^m) —here we refer to as GF(2^n) —and prime field GF(p) can be considered as the most extensively utilized finite fields. For a detailed comparison of prime and binary fields, readers can refer to [19].

Regarding binary fields, for classical computer implementation, a simple logical exclusive-OR (XOR) gate is used to perform addition, whereas a logical AND gate is often used to conduct multiplication. The binary fields are more easily employed than prime fields due to their carry-free characteristic and more staightforward hardware implementation.

Translating to its quantum counterpart, performing quantum arithmetic operation in binary field is also relatively simpler and more cost-efficient than in prime field since addition can be performed by just a CNOT gate for each bit, whereas multiplication can be done directly by a series of Toffoli gates (plus reduction by a series of swap gates and small amount of CNOT gates).

In a quantum computer environment, the depth of a quantum circuit refers to the number of time steps (time complexity) required for the quantum operations making up the circuit to run [17], which is important since maintaining a long coherence in a quantum computer is still a very challenging problem.

B. Related Works on Quantum Inversion Circuit

Despite the importance of inversion circuit in the quantum cryptanalysis, there has not been many works that extends beyond the specific implementation and optimization for small bits, such as for Advanced Encryption Standard (AES) inversion which is fixed on 8-bit inversion, e.g., [20]. In terms of a more general technique or for larger numbers, such as for use in Shor's ECDLP for ECC quantum cryptanalysis, there has only been a few to date.

For prime fields (i.e., $GF(p)$), the pioneering work is in 2003 by Proos and Zalka [11], which started the discussion on using extended Euclidean algorithm (EEA) for performing the inversion. Fourteen years later, quantum computer simulators began to rise in development, enabling a more fine-grained, gate-level optimization of quantum circuit. That time, Roetteler et al. [12] proposed another approach for performing inversion, i.e,, using the Kaliski's almost inverse (also known as the Kaliski's binary GCD) algorithm, since they found out that constructing an efficient EEA-based quantum inversion circuit is far from trivial.

Regarding the work in the binary elliptic curves, early research in the past decade (e.g., [15, 21, 22]) focused on employing the projective coordinate to eliminate the need of inversion circuit. However, as pointed out in the more recent literature, the division/inversion problem can not be completely removed since unlike in the affine coordinates, projective coordinates introduce new challenge of *non-unique* representation of points, which is required in Shor's algorithm [13]. To re-obtain a unique representation, one will still need to perform division by Z coordinate, which is also expensive [13].

In terms of the more recent works for quantum inversion in binary elliptic curves, two different approach are recently proposed by Haner et al, [13]: extended GCD algorithm variant adopted from the classical inversion by [23], and a Fermat's Little Theorem (FLT)-based inversion adapted from classical Itoh-Tsujii inversion [24]. Readers interested in a more detailed research landscape of the works in Shor's ECDLP may refer to [25].

C. FLT-based Inversion in Quantum Computing

The use of Fermat's Little Theorem (FLT)-based inversion for quantum computing and cryptanalysis has first been discussed by Banegas et al [14]. The theorem itself, introduced by Pierre de Fermat in 1640, states that for a prime number p and any number not divisible by p, say f, their relation can be described as Equation 1.

$$f^{p-1} \equiv 1 \pmod{p} \tag{1}$$

If both sides are multiplied with a^{-1}, the result is as stated in Equation 2.

$$f^{-1} \equiv f^{p-2} \pmod{p} \tag{2}$$

That is, an inversion f^{-1} can be performed by utilizing exponentiation. This also applies to binary finite fields, in which the equation can be slightly rewritten as in Equation 3.

$$f^{-1} \equiv f^{2^n-2} \pmod{m(x)} \tag{3}$$

In particular, the inversion can be achieved using n multiplications and $n-1$ squarings [14], as shown in Equation 4.

$$f^{2^n-2} = f^{2^1}.f^{2^2}.f^{2^3}...f^{2^{n-1}}. \tag{4}$$

Classically, inversion using the Fermat's Little Theorem (FLT) approach is generally less preferable than the extended Euclidean counterpart because to perform exponentiation, a large number of multiplications need to be employed, hence considered more expensive. Nevertheless, several papers have shown that using the right combination and approach, FLT-based inversion can also contribute to higher speed, such as in [26]. Furthermore, there exists improved variants of FLT algorithm, with the most popular are the ones proposed by Itoh and Tsujii [24], which reduces the number of multiplications via a smart arrangement of equations.

For the use in the quantum binary finite fields, deriving from classical FLT-based inversion algorithm, Banegas et al. [14] has described the construction of the corresponding quantum circuit. In particular, they follow the well-known Itoh-Tsujii's derivation of the FLT algorithm which requires a smaller number of operations than the standard derivation. Using the second variant of Itoh-Tsujii, the inversion cost can be reduced to below $2\log(n)$ multiplications and with the same amount of squarings as the original FLT [14]. In detail, lowering the amount of multiplications can be achieved by considering the following two distributive equations:

$$f^{2^n-2} = (f^{2^{n-1}-1})^2 \tag{5}$$

$$(f^{2^{2^t}-1} = f^{2^{2^t}-1})^{2^{2^{t-1}}} (f^{2^{2^t}-1}) \tag{6}$$

Let $n-1$ be written as $k_1 \dots k_t$ with $\sum_{s=1}^{t} 2^{k_s} = n-1$ and $k_1 > k_2 > k_3 > \dots k_t \geq 0$, with t the Hamming weight of $n-1$ in binary, $t \leq \lfloor \log{(n-1)} \rfloor + 1$, and $k_1 = \lfloor \log{(n-1)} \rfloor$. Using Equations 5 and 6, inversion via exponentiation can be achieved through the following three stages [14]:

1) Calculate $f^{2^{2^{k_1}-1}}$ with k_1 multiplications using Equation 6, save the intermediate result $f^{2^{2^{k_t}-1}}$, $f^{2^{2^{k_{t-1}}-1}}$, $\dots$, $f^{2^{2^{k_1}-1}}$.
2) Calculate $\{ \dots \{(f^{2^{2^{k_1}-1}})^{2^{2^{k_2}}} (f^{2^{2^{k_2}-1}})\}^{2^{2^{k_3}}} \dots \}^{2^{2^{k_t}}} (f^{2^{2^{k_t}-1}})\}$ using $t-1$ multiplications.
3) Square 1) and 2) to obtain the inverse, f^{-1}.

In their paper, Banegas et al. [14] describe the quantum cryptanalysis of binary elliptic curve. Specifically, the describe the implementation of Shor's ECDLP, focusing on the point addition operations and particularly the underlying components, including the division operation which can be done by an inversion followed by a multiplication in the respective registers. Their target of optimization is the circuit width (i.e., qubit size). For their FLT-based inversion, their translation of Itoh-Tsujii's algorithm to the quantum circuit are presented at Algorithm 2 in the Section 6.2 of their paper [14]. In summary, to perform the inversion, they employ a series of squarings (i.e, K in their paper), (modular) multiplications M, along with the inverse squarings K^{-1}. The example of their construction is illustrated on Circuit 6 in the Section 6.2 of their paper. In addition, there has been several works that focuses on reducing depth rather than width, such as by Rahman et al., [27], which constructs a combinatorial circuit to analyze KATAN block cipher by optimizing the depth and number of gates [28], and depth analysis in [29].

III. Proposed Method

In this section, we describe our proposed quantum FLT-based inversion operation for binary elliptic curve $GF(2^n)$ to achieve a lower depth. In particular, we propose a different approach from the previous work by Banegas et al. [14]; that is, here we employ a complete waterfall approach to translate the Itoh-Tsujii algorithm to the corresponding quantum circuit. We modify the steps in previous work [14] to minimize in the number of CNOT gates. As a result, the overall depth and gate count can be lowered whereas the same T-depth as the previous work can be maintained.

A. Proposed Variant of FLT-based Inversion

In this paper, to aid the explanation of our method, we start by elaborating the quantum circuit construction used in Banegas et al [14], then introduce our differences along the way.

In general, the quantum FLT-based inversion consists of three stages [14], as discussed in the Section II-C and illustrated in Fig. 1. At the first stage, $f^{2^{2^{k_1}}-1}$ is calculated. This can be performed by a series of squarings and (modular) multiplications. At the second stage, $f^{2^{(n-1)}-1)}$ is calculated in a similar fashion. At the third stage, both are squared to obtain the inversion result, f^{-1}.

In the method implemented in the previous work [14], an inverse squaring operation (namely, K^{-1}, or equivalently $SQUARE^{-1}$) is required in each iteration of the first stage. That is, after finishing a squaring, the value is uncomputed by the $SQUARE^{-1}$ so that the qubit can be reused for the subsequent squaring. Unfortunately, this introduces more CNOT gates to the circuit since $SQUARE^{-1}$ practically consists of CNOT gates, which is bounded by $O(n^2)$. Furthermore, qubit reuse also requires additional series of CNOT gates of at least n per squaring for uncomputation, yielding larger overall depth and gate count.

In this paper, a more straightforward way of using a waterfall approach to perform inversion is presented. We keep the standard method of using three stages of calculation. However, the method of calculation is different, giving different construction and operation placement, as presented in Algorithm 1. Furthermore, using a sequential arrangement of values, uncomputation is performed after the calculation finishes; hence $SQUARE^{-1}$ and additional CNOTs can be removed. As a result, a circuit with a lower overall depth and smaller gate count can be obtained.

Regarding the explanation of Algorithm 1, the algorithm is for performing a time-efficient FLT-based quantum inversion circuit for a binary elliptic curve using constant field polynomial $m(x)$ of degree $n > 0$ as a fixed input. For simplicity and ease of comparison, we closely follow the notation and description of [14]. k_1 is a register that saves result multiplication final in stage 1 as a predecessor value for the next stage, while s is the iteration variable. Note that $k_1 > k_2 > \dots k_t \geq 0$ such that $\sum_{s=1}^{t} 2^{k_s} = n-1$. In this case, maximum register number is $k = 2k_1 + t$, in quantum circuit we define this value as number of qubits.

For the operations (functions), lower case represents the standard gate operation (in this case, CNOT and swap), while the all-capital case (in this case, *SQUARE* and *MULT*) represents a block of operation which can be rolled out according to one's implementation preference. Generally, for binary finite field quantum circuit, squaring (*SQUARE*) operation is constructed via LUP Decomposition, a decomposition method for linear mapping also used in [14, 20]. As for the (modular) multiplication operation, the straightforward Schoolbook multiplication (followed by reduction) or other approach such as Karatsuba or Toom-Cook can also be employed. In this study, since our focus is on the inversion algorithm itself, we utilize the simple-yet-straightforward Schoolbook multiplication for the underlying multiplication block.

Algorithm 1 Our proposed variant of FLT. For simplicity and ease of comparison, notation and description follow [14].

```
Fixed input : A constant field polynomial $m(x)$ of
        degree $n > 0.k_1 > k_2 > \dots k_t \geq 0$ such
        that $\sum_{s=1}^{3} 2^{k_s} = n - 1.k_{max} = 2 * k_1 + t$.
Quantum input :
- A non-zero binary polynomials of degree up to
  $n - 1$ stored in array (register) $f_0$ of size $n$ to invert.
- $k$ zero arrays of size $n$ initialized to an all-$|0\rangle$
  state: $f_1, \dots, f_k$.
Result : inverse of the input, stored in $f_k$
for $i = 1, \dots, k_1$ do                                        //stage 1
    CNOT$(f_{2*(i-1)+1}, f_{2*(i-1)})$
    for $i = 1, \dots, t-1$ do
        SQUARE$(f_{2*(i-1)+1})$
        MULT$(f_{2*(i-1)+2}, f_{2*(i-1)+1, f_{2*(i-1)}})$
for $s = 1, \dots, t-1$ do                                        //stage2
    for $k = 1, \dots, 2^{k_{s+1}}$ do
        SQUARE$(f_{2*(i-1)+1})$
        MULT$(f_{2*(i-1)+2}, f_{2*(i-1)+1, f_{2*(i-1)}})$
if $t = 1$ then
    swap $(f_{k_1}, f_k)$
SQUARE$(f_k)$                                                     //stage 3
```

Note that more ancilla qubits are required compared to the previous work, but they similarly are to be uncomputed at the end. In addition, considering the development of quantum hardware which has seen a rapid acceleration in the increasing number of qubits, this approach does pose a competitive advantage.

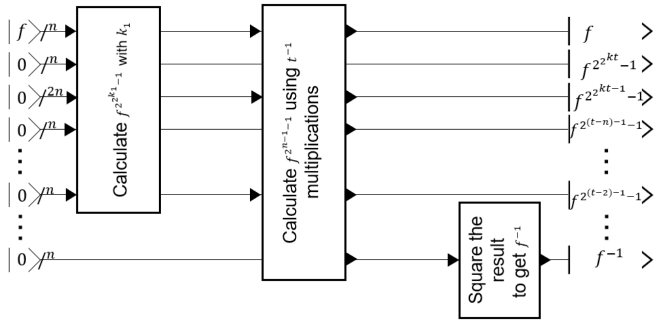

Fig. 1. A three-stage FLT inversion steps, inferred from [14]

B. Experiment Setup for Evaluation

To compare the performance of our method and previous work, we construct both the FLT inversion by Banegas et al., [14] and ours in the Qiskit quantum computer simulator

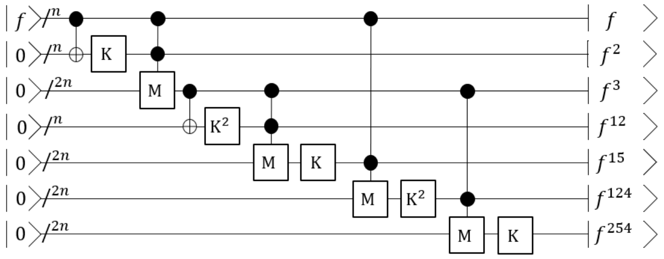

Fig. 2. Illustration of Modified FLT Algorithm, Example for $n = 8$

platform and run the resource analysis. For the previous work's construction, we follow the algorithm presented on Algorithm 2 in their paper. In our experiment, since our scope is on the FLT inversion evaluation and not the lower-level component (e.g., the multiplication block), we employ the exact same setting for the underlying circuits (e.g., same version of squaring and multiplication blocks) for both Banegas et al's and ours. Additionally, for constructing a modular multiplication, we utilize the Schoolbook multiplication followed by reduction for both scenarios due to its simplicity. Employing other multiplication methods (e.g., quantum Karatsuba such as [14, 30] or Toom Cook multiplication such as [31, 32]) can be done accordingly by simply changing the underlying blocks since it is supported by Qiskit. Following previous works on quantum cryptanalysis [13, 14, 28], swap operations can be considered as free since it can be done via qubit relabeling [33, 34].

Furthermore, since the most prominent use of this quantum inversion circuit is on the quantum cryptanalysis of binary elliptic curve using Shor's ECDLP, we evaluate our work with the parameters for several NIST's standardized curves, i.e., B-233, B-283 B-409 and B-571 with their respective irreducible polynomials, along with the smaller curve with the irreducible polynomial listed in [14]. In terms of the metric used, we currently analyze the number of CNOT Gates (i.e., CNOT count), circuit depth, and qubit size (i.e., circuit width).

IV. Result and Discussions

A. Evaluation Result

To verify our proposed method, we implement both versions of FLT-based algorithm (i.e., ours and Banegas et al.,'s [14]) on Qiskit platform. Then, we perform the resource analysis on the decomposed (i.e., transpile function in Qiskit) version of the quantum circuits and compare the result. We ran our experiment on Python 3.9.7, Qiskit version 0.30.0 locally in an x64-based desktop PC (Intel i7-8700, 6 cores processor), on Windows 10 Pro OS with 64 GB of RAM. For the transpile function, the basis gates follow the decomposition described in Q-Crypton, a quantum-security evaluation platform for cryptography developed by ETRI, a Korean government research institute. In particular, the allowed basis gates are ['id', 'h','t', 'tdg', 's', 'sdg', 'rz', 'x', 'y', 'z', 'cx'], without utilizing optimization function (i.e., set the optimization level to 0).

Algorithm 1 is translated to quantum circuit construction following the pseudocode given in the Appendix. In the algorithm, the placement of the operations (e.g., squarings, multiplications, etc) on the quantum registers are shown. Additionally, *anc* in the pseudocode refers to the list of ancillary registers, namely *anc1*, *anc2*, and so on. The corresponding quantum circuits for degree-8 inversions for the previous work and this work are shown in Figures 3 and 4. For the higher bits, due to a very long circuit, the pictures are not displayed in this paper.

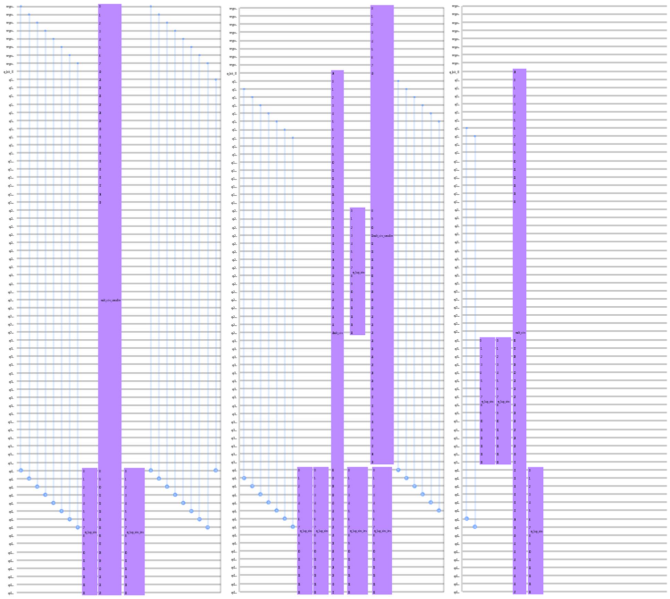

Fig. 3. Our Qiskit's Circuit Construction of Previous Work's FLT-based Inversion [14]

The result of the experiment is presented in Table I. In all cases, our evaluations decomposed higher-level operations to CNOT, T, $T^{\dagger}$, and Hadamard gates only, all with equal gate count except CNOT gates for both the previous work and ours. As shown, the number of CNOT gates in our proposed method are lower than the previous work. This also results in a lower overall depth of the circuit. Additionally, it can be inferred that the larger the inversion bit size (which is same as the degree of $m(x)$), the larger the savings obtained from our inversion, as shown in Figures 5 and 6. Note that our method comes with a tradeoff of higher circuit width (i.e., qubit size).

B. Discussions

This proposed work focus on minimizing the depth of the circuit as the main metric. In other words, this work wishes to achieve a time-efficient circuit. This is achieved by reducing the number of CNOT gates in the circuit by employing a complete waterfall approach to eliminate the use of the inverse squaring circuits ($SQUARE^{-1}$) used in the previous work, i.e., [14]. Their removal contributes to an overall lower circuit depth.

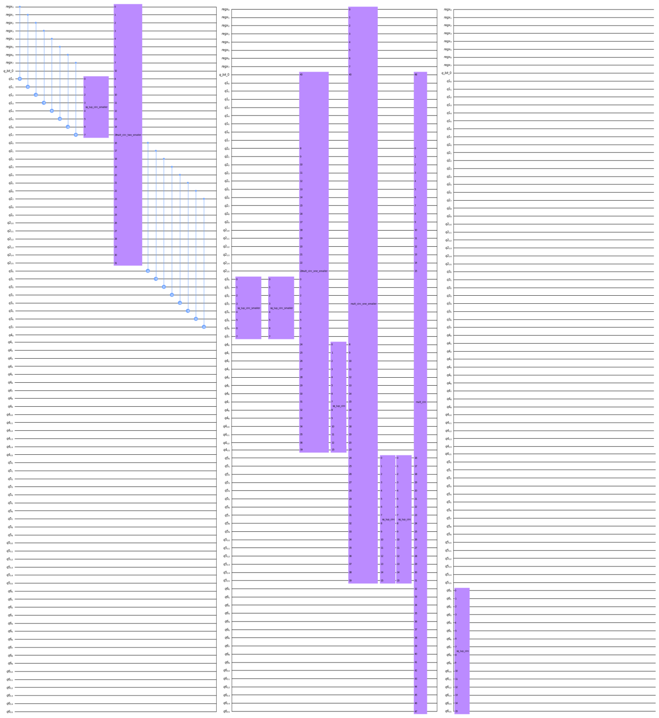

Fig. 4. Our Qiskit's Circuit Construction of Our Method's FLT-based Inversion.pdf

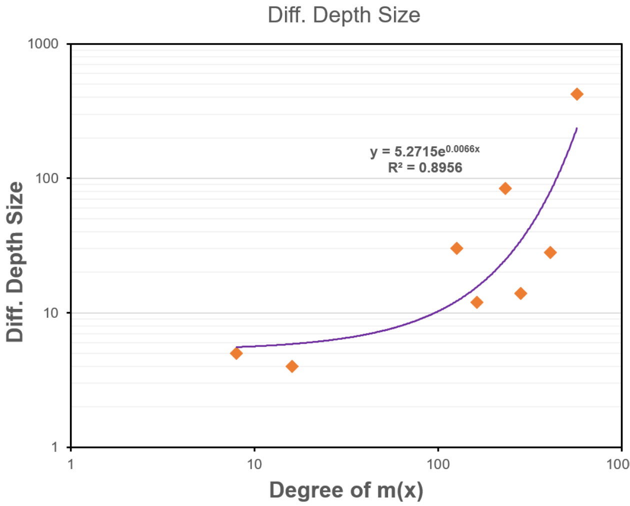

Fig. 5. Depth size difference

In this subsection, we note several considerations that may arise from our proposed method. Firstly, one may think that the depth reduction is not very large. However, an inversion circuit is rarely employed as a standalone circuit. Rather, it is usually incorporated to a larger circuit performing a more complex operation. For instance, one of the prominent use of quantum inversion circuit is in the Shor's ECDLP for cracking ECC —currently for quantum cryptanalysis and resource estimation, and possibly for the real cracking in the future of fault-tolerant quantum computation —specifically, as one of the components in the *point addition* operation. Further, the point addition is used iteratively (repeatedly) in the *scalar multiplication* operation. In the long run, more depth savings can be obtained.

Secondly, one might question whether the overall depth, as the primary metric used in this paper, is of importance. Indeed, focusing on T depth (and T-count) reduction is very

TABLE I
COMPARISON OF FLT-BASED INVERSION OF THE PREVIOUS WORK [14] AND THIS WORK BASED ON QISKIT SIMULATION RESULT

n	**Qubit Count**		**CNOT Count**		**Overall Depth**	
	Previous Work [14]	**This Work**	**Previous Work [14]**	**This Work**	**Previous Work [14]**	**This Work**
8	73	89	1856	1804	810	805
16	209	257	10014	9966	2565	2561
127	2922	3684	1074196	1073434	31267	31237
163	3098	4239	1484366	1483225	53730	53718
233	4894	6525	3306274	3304643	58999	58915
283	6510	8774	5431582	5429318	161071	161057
409	9408	12680	11148086	11144814	111151	111123
571	15418	20557	25778322	25773183	200641	200217

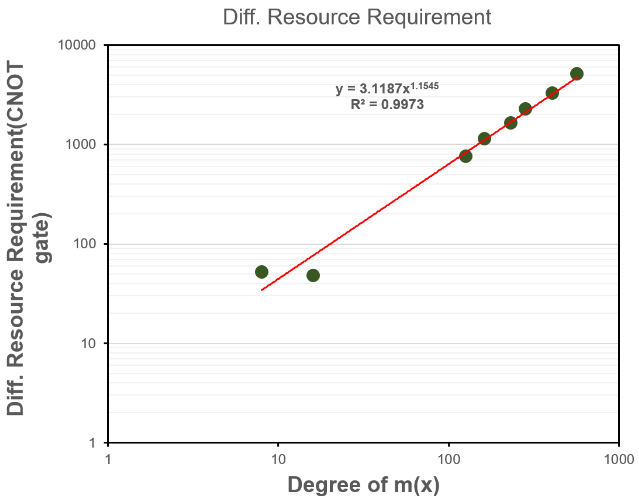

Fig. 6. CNOT size difference

beneficial [29] due to the T-gate's criticality in the Fault-Tolerant Quantum Computation (FTQC) [35], thus has been celebrated as the focus of the recent research efforts in quantum computing and cryptanalysis. Nevertheless, as stated in [29], the improvements in T depth and T-count are frequently accompanied by an increase in other resources, such as the number of controlled-NOT gates (CNOTs) or the overall depth of the circuit. This extra cost is not insignificant and can have an impact on a quantum algorithm's final outputs [29]. Furthermore, as discussed in [36], the cost of a quantum circuit should not be reduced to the T-cost. Contrarily, it is critical to reduce secondary resources to a minimum while maintaining the T-cost unchanged [29], like the purpose of our work. Another supporting argument come from a recent study on quantum cryptanalysis of Grover's algorithm on a block cipher titled KATAN, which states that NIST has not specified any limits on circuit width or the use of ancilla qubits whereas maximum depth of a circuit has been discussed [27]. This strengthens the importance of depth, the parameter that relates to decoherence of aquantum systems and corresponds to time complexity of a quantum computer.

Another consideration that may arise is about the number of qubits. Compared to the previous work, our variant of quantum FLT-based inversion does employ larger number of qubits. However, it comes from the ancilla registers, which are also present in the FLT inversion. On quantum computers, the ancilla registers will need to be uncomputed at the end of the computation. That is, the ancilla registers will be returned to state $|0\rangle$ by applying the inverse gates in reverse order. This uncomputation results in twice the depth of the original circuit, as in [30, 31], whereas the ancilla qubits will be cleared up. Thus, in certain cases, reducing the depth in the large-scale future quantum computer can be more important. Additionally, going back to the discussion of our approach, the saving on depth will be doubled.

Furthermore, looking at the progress of quantum hardware development, particularly in the most popular one, i.e., superconducting qubit, the physical qubit size has seen a significant increase in the past few years —and is quite promising to increase much more rapidly in the future —whereas progress in the coherence time (relates to depth or time complexity) is not as fast. Combined with the progress in the quantum error correction (QEC), achieving a similar increase in logical qubit may be viable in the future.

To conclude, bearing in mind that this proposed inversion circuit does not incur larger cost in both T count and T-depth, this construction can be employed as an alternative to the existing quantum inversion circuit when time complexity is the main consideration.

V. CONCLUSION

In this paper, we have presented our approach to construct FLT-based quantum inversion circuit with lower depth for use in binary elliptic curves. We offer an alternative variant from the prior work [14] to perform inversion, a crucial subcircuit of division circuit and the most expensive sub-part of computing the scalar multiplication in the quantum binary elliptic curve. In detail, the proposed design employs different approaches at the first and second stages of Banegas et al., [14]'s elaboration of Itoh-Tsujii's FLT inversion algorithm. We utilize a different arrangement of the quantum circuit and move the uncomputation of squarings to the end to eliminate the intermediate uncomputation, minimizing the number of CNOT gates in the circuit. As a result, a lower overall gate count and circuit depth can be achieved. For verification, we implemented both our method and the previous work's FLT approach in IBM Qiskit quantum computer simulator and compared the resource requirement for B-233, B-283, B-409, and B-571 as the standard for binary elliptic curve, which confirms that a lower overall gate count and circuit depth can be achieved when our approach is used in the perspective of quantum cryptanalysis of ECC. Additionally, our approach maintains the same T-depth as the previous work, which is of additional advantage. Note that our approach does incur a higher number of auxiliary qubits. Nevertheless, the fact that uncomputation must also be performed in any case, meaning that the auxiliary qubits will still need to be uncomputed at the end, gives our approach a leverage when one wants to pursue a lower depth implementation.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*, Ieee, 1994, pp. 124–134.

[2] ——, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[3] A. Cho, *Ibm promises 1000-qubit quantum computer—a milestone—by 2023*, Sep. 2020. [Online]. Available: https://www.science.org/content/article/ibm-promises-1000-qubit-quantum-computer-milestone-2023.

[4] P. Ball, *First quantum computer to pack 100 qubits enters crowded race*, Nov. 2021. [Online]. Available: https://www.nature.com/articles/d41586-021-03476-5.

[5] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Physical Review A*, vol. 54, no. 1, p. 147, 1996.

[6] D. Beckman, A. N. Chari, S. Devabhaktuni, and J. Preskill, "Efficient networks for quantum factoring," *Physical Review A*, vol. 54, no. 2, p. 1034, 1996.

[7] S. Beauregard, "Circuit for shor's algorithm using 2n+ 3 qubits," *arXiv preprint quant-ph/0205095*, 2002.

[8] I. L. Markov and M. Saeedi, "Constant-optimized quantum circuits for modular multiplication and exponentiation," *arXiv preprint arXiv:1202.6614*, 2012.

[9] R. Van Meter and K. M. Itoh, "Fast quantum modular exponentiation," *Physical Review A*, vol. 71, no. 5, p. 052 320, 2005.

[10] R. Rines and I. Chuang, "High performance quantum modular multipliers," *arXiv preprint arXiv:1801.01081*, 2018.

[11] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *arXiv preprint quant-ph/0301141*, 2003.

[12] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, "Quantum resource estimates for computing elliptic curve discrete logarithms," in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2017, pp. 241–270.

[13] T. Häner, S. Jaques, M. Naehrig, M. Roetteler, and M. Soeken, "Improved quantum circuits for elliptic curve discrete logarithms," in *International Conference on Post-Quantum Cryptography*, Springer, 2020, pp. 425–444.

[14] G. Banegas, D. J. Bernstein, I. van Hoof, and T. Lange, "Concrete quantum cryptanalysis of binary elliptic curves," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 451–472, 2021.

[15] M. Rötteler and R. Steinwandt, "A quantum circuit to find discrete logarithms on ordinary binary elliptic curves in depth o (log^ 2 n)," *arXiv preprint arXiv:1306.1161*, 2013.

[16] P. Budhathoki and R. Steinwandt, "Automatic synthesis of quantum circuits for point addition on ordinary binary elliptic curves," *Quantum Information Processing*, vol. 14, no. 1, pp. 201–216, 2015.

[17] L. Gyongyosi and S. Imre, "Circuit depth reduction for gate-model quantum computers," *Scientific Reports*, vol. 10, no. 1, pp. 1–17, 2020.

[18] P. K. G. Nadikuda and L. Boppana, "An area-efficient architecture for finite field inversion over gf(2m) using polynomial basis," *Microprocessors and Microsystems*, vol. 89, p. 104 439, 2022, ISSN: 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2022.104439. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014193312200014X.

[19] E. Wenger and M. Hutter, "Exploring the design space of prime field vs. binary field ecc-hardware implementations," in *Nordic Conference on Secure IT Systems*, Springer, 2011, pp. 256–271.

[20] D. Chung, J. Lee, S. Lee, and D. Choi, "Towards optimizing quantum implementation of aes s-box.," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 941, 2020.

[21] D. Cheung, D. Maslov, J. Mathew, and D. K. Pradhan, "On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography," in *Workshop on Quantum Computation, Communication, and Cryptography*, Springer, 2008, pp. 96–104.

[22] D. Maslov, J. Mathew, D. Cheung, and D. K. Pradhan, "An o(m2)-depth quantum algorithm for the elliptic curve discrete logarithm problem over gf(2)a," *Quantum Info. Comput.*, vol. 9, no. 7, pp. 610–621, Jul. 2009, ISSN: 1533-7146. DOI: 10.5555/2011814.2011818.

[23] D. J. Bernstein and B.-Y. Yang, "Fast constant-time gcd computation and modular inversion," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 340–398, 2019.

[24] T. Itoh and S. Tsujii, "Structure of parallel multipliers for a class of fields gf (2m)," *Information and computation*, vol. 83, no. 1, pp. 21–40, 1989.

[25] H. T. Larasati and H. Kim, "Quantum cryptanalysis landscape of shor's algorithm for elliptic curve discrete logarithm problem," in *International Conference on Information Security Applications*, Springer, 2021, pp. 91–104.

[26] A. M. Awaludin, H. T. Larasati, and H. Kim, "High-speed and unified ecc processor for generic weierstrass curves over gf (p) on fpga," *Sensors*, vol. 21, no. 4, p. 1451, 2021.

[27] M. Rahman and G. Paul, "Grover on katan: Quantum resource estimation," *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–9, 2022.

[28] ——, "Grover on katan: Quantum resource estimation," *IEEE Transactions on Quantum Engineering*, 2022.

[29] T. G. de Brugière, M. Baboulin, B. Valiron, S. Martiel, and C. Allouche, "Reducing the depth of linear re-

versible quantum circuits," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–22, 2021.

[30] A. Parent, M. Roetteler, and M. Mosca, "Improved reversible and quantum circuits for karatsuba-based integer multiplication," *arXiv preprint arXiv:1706.03419*, 2017.

[31] S. Dutta, D. Bhattacharjee, and A. Chattopadhyay, "Quantum circuits for toom-cook multiplication," *Physical Review A*, vol. 98, no. 1, p. 012 311, 2018.

[32] H. T. Larasati, A. M. Awaludin, J. Ji, and H. Kim, "Quantum circuit design of toom 3-way multiplication," *Applied Sciences*, vol. 11, no. 9, p. 3752, 2021.

[33] K. Jang, H. Kim, S. Eum, and H. Seo, "Grover on gift," *Cryptology ePrint Archive*, 2020.

[34] K. Jang, G. Song, H. Kim, H. Kwon, H. Kim, and H. Seo, "Efficient implementation of present and gift on quantum computers," *Applied Sciences*, vol. 11, no. 11, p. 4776, 2021.

[35] V. Gheorghiu and M. Mosca, "Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes," *arXiv preprint arXiv:1902.02332*, 2019.

[36] D. Maslov, "Optimal and asymptotically optimal nct reversible circuits by the gate types," *arXiv preprint arXiv:1602.02627*, 2016.

Appendix: Python-style Pseudocode for Circuit Construction in Qiskit

```
########################################################################
###Pseudo code for the proposed work’s circuit construction in Qiskit##
########################################################################

#Condition
   if n == 8:
        ip = [8,4,3,1,0] # degree 8
        ks = [1,2]
        k1 = 2
    elif n == 16:
        ip = [16,5,3,1,0] #degree 16
        ks = [1,2,4]
        k1 = 3
    elif n == 127:
        ip = [127,1,0] #degree 127
        ks = [2,4,8,16,32]
        k1 = 6
    elif n == 163:
        ip = [163,7,6,3,0] #degree 163
        ks = [2,32]
        k1 = 7
    elif n == 233:
        ip = [233,74,0] #degree 233
        ks = [8,32,64]
        k1 = 7
    elif n == 283:
        ip = [283,12,7,5,0] #degree 283
        ks = [2,8,16]
        k1 = 8
    elif n == 409:
        ip = [409,87,0] #degree 409
        ks = [8,16,128]
        k1 = 8
    elif n == 571:
        ip = [571,10,5,2,0] #degree 571
        ks = [2,8,16,32]
        k1 = 9
    else:
        raise ValueError('n can only be 8, 16, 127, 163, 233, 283, 409, or 571')

    t= list(bin(n-1)[2:]).count('1') #Hamming Weight

#Stage 1
    for i in range(k1):
        for l in range(n):
            cnot(anc[2*i][l],anc[2*i+1][l]) #cnot(control, target)
        for j in range(2**i):
            SQUARE(anc[2*i+1])
        MULT(anc[2*i], anc[2*i+1], anc[2*i+2]]) #mult(input, input, output, control)

#Stage 2
    for s in range(len(ks)):
        for j in range(ks[s]):
            SQUARE(anc[2*k1+s])
        MULT(anc[2*s], anc[2*k1+s], anc[2*k1+s+1]) #mult(input, input, output, control)

    if t == 1:
        swap(anc[2*k1[0]],anc[2*k1+s+1])

    SQUARE(anc[2*k1+s+1])
```