Another Concrete Quantum Cryptanalysis of Binary Elliptic Curves

Dedy Septono Catur Putranto[1,2], Rini Wisnu Wardhani[1,2], Harashta Tatimma Larasati[1,3] and Howon Kim[1]

[1] Pusan National University, Busan, Republic of Korea
[2] National Cyber and Crypto Agency, Indonesia
[3] Institut Teknologi Bandung, Indonesia
{dedy.septono,rini.wisnu,harashta,howonkim}@pusan.ac.kr

Abstract. This paper presents concrete quantum cryptanalysis for binary elliptic curves for a time-efficient implementation perspective (i.e., reducing the circuit depth), complementing the previous research by Banegas et al., that focuses on the space-efficiency perspective (i.e., reducing the circuit width). To achieve the depth optimization, we propose an improvement to the existing circuit implementation of the Karatsuba multiplier and FLT-based inversion, then construct and analyze the resource in Qiskit quantum computer simulator. The proposed multiplier architecture, improving the quantum Karatsuba multiplier by Van Hoof et al., reduces the depth and yields lower number of CNOT gates that bounds to $O(n^{log_2(3)})$ while maintaining a similar number of Toffoli gates and qubits. Furthermore, our improved FLT-based inversion reduces CNOT count and overall depth, with a tradeoff of higher qubit size. Finally, we employ the proposed multiplier and FLT-based inversion for performing quantum cryptanalysis of binary point addition as well as the complete Shor's algorithm for elliptic curve discrete logarithm problem (ECDLP). As a result, apart from depth reduction, we are also able to reduce up to 90% of the Toffoli gates required in a single-step point addition compared to prior work, leading to significant improvements and give a new insights on quantum cryptanalysis for a depth-optimized implementation.

Keywords: Quantum Cryptanalysis · Elliptic Curves · Point Addition · Quantum Computing · Quantum Gates · Shor's Algorithm · Quantum Resource Estimation

1 Introduction

The field of quantum computing is evolving rapidly, especially since Shor's algorithm proved the apparent value of quantum phenomena [Sho94, Sho99]. Furthermore, significant research in quantum computing and hardware contributes to the advancement of knowledge. In recent years, scientists have been on the way to developing secure, the fastest, most applicable, or most cryptanalytic-resistant algorithms in the cryptography field due to the fact that quantum computer may, in the future, be able to crack the existing widely-used cryptosystem. The number of years left for asymmetric cryptographic algorithms, i.e, ECC and RSA, is determined by quantum computing progress and improvements in the Shor's algorithm optimization [BBvHL21] along with the progress in the quantum hardware itself and on the quantum error correction. In another case, quantum arithmetic circuit should be adequately optimized to compute and recursively process a quantum algorithm. For instance, in the field arithmetic, operations such as multiplications and inversion should be optimized for certain needs (e.g., for the purpose of minimizing width or minimizing depth), because they take a significant amount of resources and

time. In particular, for complex quantum circuits such as *point addition*, i.e., the underlying circuit of the *double scalar multiplication* (the most expensive part of the circuit for cracking the existing Elliptic Curve-based Cryptosystems (ECC), using Shor's algorithm for Elliptic Curve Discrete Logarithm Problem (ECDLP)), these two operations should be of top concerns since they constitute the largest resource for the whole quantum point addition circuit. Therefore, it is beneficial to explore the improvements on these aspects. Further, their application on the point addition circuit can further provide insights of the estimation of resource for cracking the current ECC-based cryptosystem.

1.1 Why Another Quantum Cryptanalysis on ECC?

Elliptic-curve cryptography (ECC), one of the most prominent Diffie-Hellman versions, has a wide range of applications. Because of its smaller key sizes, ECC, developed by Koblitz and Miller in 1985 [KMV00], has been recognized as the key component in the security protocol of those technologies among asymmetric cryptographic algorithms. Curve25519 [Ber06] and Curve448 [Ham15] were suggested by The Internet Research Task Force (IRTF) [LHT16] for a greater extent of realistic security, with 128-bit and 224-bit security levels, and integration in Transport Layer Security (TLS) standard 1.3 [Res18]. Following that, the National Institute of Standards and Technology (NIST) [CMRR19] adopted these curves as part of their standard.

Similar to the extensive usage of ECC, efficient construction of various arithmetic operations in quantum arithmetic circuits become increasingly crucial as hardware and quantum computing research evolves. Classical arithmetic computation techniques and many variant modifications of it are applied to construct ECC not only in the hardware area but also in the quantum area. Because of the widespread use of high-performance ECC Processors, some research is being conducted to meet the requirement for a high-security and efficient ECC processor architecture, and one clear example is Awaludin et.al. [APWK22]. Awaludin et al. claimed to have developed a high-performance curve 448 ECC processor based on a novel variant Karatsuba architecture [KO62]. The traditional Karatsuba technique has a $O(n^{1.59})$ time complexity and a $O(n)$ spatial complexity; some variant has been presented to minimize computation complexity i.e. [PRM17] [Gid19].

A quantum arithmetic circuit is a reference model that can implement quantum computation activities. Quantum arithmetic circuits become more critical and powerful as the underlying, essential component to performing algorithm computation. Furthermore, quantum arithmetic circuits are becoming increasingly important since they must acquire resources estimated over the arithmetic operation in quantum processing, cryptanalysis, and actual implementation, either in hardware implementation or quantum hardware processing. Notably, some research attempts to implement an algorithm and analyze the performances, but just a few in cryptanalysis or attack with a quantum approach. Even though quantum computers are now relatively small compared to classical computers, they will pose a threat to computer security at some point in the future [BBvHL21]. Curve 448 is a conservatively built elliptic curve that has resulted in ECC building challenges and is perceived breakthroughs in strong cryptanalysis and classical attack [Ham15]. The challenge is not only to elaborate on ECC in the scope of the need for optimum computation but also high requirement to suit security aspect, i.e., cryptanalysis in ECC.

Research quantum cryptanalysis observes more extensive use of the arithmetic variant design and follow-up necessary computation in the quantum computing field. In the quantum computing field, Proos and Zalka describe how to implement Shor's efficient quantum technique for discrete logarithms for the particular elliptic curve groups in their elliptic curves over $GF(p)$ implementation paper gives a comprehensive explanation of the elliptic curve in the quantum case [PZ03]. A quantum computer has been forecasted to break a 160-bit elliptic curve cryptographic key with approximately 1000 qubits, whereas factoring the 1024-bit RSA modulus, which is more secure, would take over 2000 qubits [PZ03].

Rötteler, Naehrig, Svore, and Lauter conducted remarkable quantum cryptanalysis of elliptic curve encryption over prime fields [RNSL17], which is then improved in Haner et al [HJN+20].

Recently, Banegas et al., present concrete resource estimation to perform Shor's ECDLP on binary elliptic curves. The focus of their work include optimizing multiplication and division circuits, two essential arithmetic circuits which is expensive compared to the more elementary operations such as addition. They propose to employ a space-efficient Karatsuba multiplication from Van Hoof [VH19]. Additionally, the authors also present inversion circuits (a component in the division circuit) in two types, i.e., based on greatest common divisor (GCD) and based on Fermat's Little Theorem (FLT), then compared their performance to be them employed in the calculation of point addition's resource cost estimation. The focus of their research is to reduce the number of qubits [BBvHL21] along with the number of Toffoli gates.

However, there has not been a quantum cryptanalysis in binary elliptic curve that focuses on reducing the depth of the circuit. In fact, [BBvHL21] can be considered the only one which have shown a concrete estimate of performing Shor's ECDLP on Binary ECC. To complement previous research, we propose to perform analysis from the time-efficiency perspective (i.e., to achive optimized depth). Considering that multiplier and the inversion are the most computationally intensive step in the field operation, we propose to employ our modification to the existing implementation, which gives improvement in tems of depth. In detail, we propose improving the existing Karatsuba multiplier in Hoof's study and modified step in Banegas et al.'s FLT-based inversion to achieve depth optimization, then create and analyze the resource in the Qiskit quantum computer simulator. Also, we compare our result to previous works, i.e., [MMCP09] [KS15] [VH19] for multipliers, and other works such as [BBvHL21] [LPWK22] to FLT-based inversion. Furthermore, referring to [BBvHL21]'s concrete implementation, we incorporate our improvement to the analysis of point addition such as in [BBvHL21], in the quantum cryptanalysis of Shor's algorithm for elliptic curve discrete logarithm problem (ECDLP).

1.2 Contributions

The contributions of this paper can be summarized as follows:

1. This study presents a concrete quantum cryptanalysis for binary elliptic curves. Different from the previous work by Banegas et al., [BBvHL21] which focuses on optimizing the space-efficiency (i.e., width or number of qubits), we offer the analysis from the time-efficient implementation standpoint (i.e., overall depth).

2. We improve the existing quantum Karatsuba multiplier variant by Van Hoof et al., [VH19]. In particular, we modify the implementation steps in [VH19], build it in Qiskit quantum computer simulator, obtain the resource analysis, and compare with the result by [VH19]. The improved multiplier reaches better optimization in terms of depth and number of CNOT gates which bounds to $O(n^{log_2(3)})$ while preserving the similar number of Toffoli gates and qubits as the previous work.

3. Furthermore, for quantum inversion, we propose to utilize our Karatsuba as the underlying multiplier block in the low-depth FLT-based inversion variant by Larasati et al., [LPWK22], which in our simulation shows a lower CNOT count and lower overall depth than previous works, with the tradeoff of a larger qubit size.

4. Finally, we incorporate our improved multiplier and inversion to the case for quantum cryptanalysis of binary elliptic curve, yielding a depth estimate of $7n+6n\log_2(3)$, qubit count of $4n + 7n\log(n) + 7$, and Toffoli count of $4n^3 + 3n\log((3) + 1) + 25n^2\log(n) + 2n^2 + O(n^{\log((3)+1)})$. The result also shows that our lower depth implementation also reduce the number of Toffoli gates, with the tradeoff of higher

number of qubits and CNOT gates. Also, the result reduces 90% of the Toffoli gates needed in a single-step point addition besides optimizing the depth.

1.3 Organization of the Paper

Section 2 consists of preliminaries of this paper, such as basic finite field arithmetic (addition, multiplication, and inversion), a brief explanation of binary elliptic curves, quantum background, Shor's algorithm, and covers detailed in binary elliptic curve quantum cryptanalysis. The Multiplication in Section 3 provides previous work on multiplication that inspired this research and our improved multiplier architecture. Section 4 gives FLT-based inversion and our modification FLT-based inversion. In Section 5, besides describing the evaluation method and setup, we put the improvement together to achieve a details resource count and analysis. Section 5.2.3 introduces the incorporation of the improved multiplication and FLT-based inversion in binary elliptic curves quantum cryptanalysis addressing point addition operation. Finally, Section 6 draws a conclusion. Along with the paper writing, related work to this subject will be discussed.

2 Preliminaries

In general, this section briefly describes essential finite field representative and binary elliptic curve arithmetic in quantum fundamentals related to this research. Subsection 2.1 provides basic finite field arithmetic, such as finite field arithmetic in a binary elliptic curve, SHORs, and ECDLP in subsection 2.2. Subsection 2.4 gives a brief explanation of the Quantum background. In addition, Subsection 2.5 will discuss addition, multiplication, squaring, and inversion (in a division operation) in quantum implementation, which will be used in further quantum cryptanalysis point addition. The following section (sections 3 and 4) will explain multiplication and inversion correlated to this research in more detail.

2.1 Finite Field Arithmetic

Arithmetic functions play a fundamental role in quantum computation. The primary computing operation and its variety on a bit system consider a wealth of implementation in the hardware and quantum computing field. The performance of such systems that utilize the binary field (F_{2^n}) and prime field (F_p) is heavily influenced by the optimum implementation of the underlying field operations, such as modular addition (add), subtraction (sub), multiplication (mult), squaring (squ), and inversion (inv).

Finite fields F(p) or Galois field GF(p) characterized by $0, 1, \ldots, p-1$ finite elements, are commonly employed in cryptographic techniques and have applications in various domains. Notably, binary fields are more commonly employed than prime fields in hardware or software implementation due to their carry-free characteristic and easier hardware implementation. [WH11] presents a complete comparison research analysis of prime and binary fields. The operations listed below are of significant interest: Addition is perfoming $\alpha+\beta$ from given $\alpha, \beta \in F_{2^n}$. Multiplication conducts product of $\alpha.\beta$ from given $\alpha, \beta \in F_{2^n}$. Squaring operation determine α^2, for a given $\alpha \in F_{2^n}$. Lastly, inversion operation, given $\alpha \in F_{2^n}^*$, find $\alpha^{-1} \in F_{2^n}$.

Traditional arithmetic computation techniques, as well as various variants of them, are used to build ECC not only in the hardware but also in the quantum realm, for further information on elliptic curves, see [CFA+05]. Figure 1 gives clear implementation hierarchy in ECC.

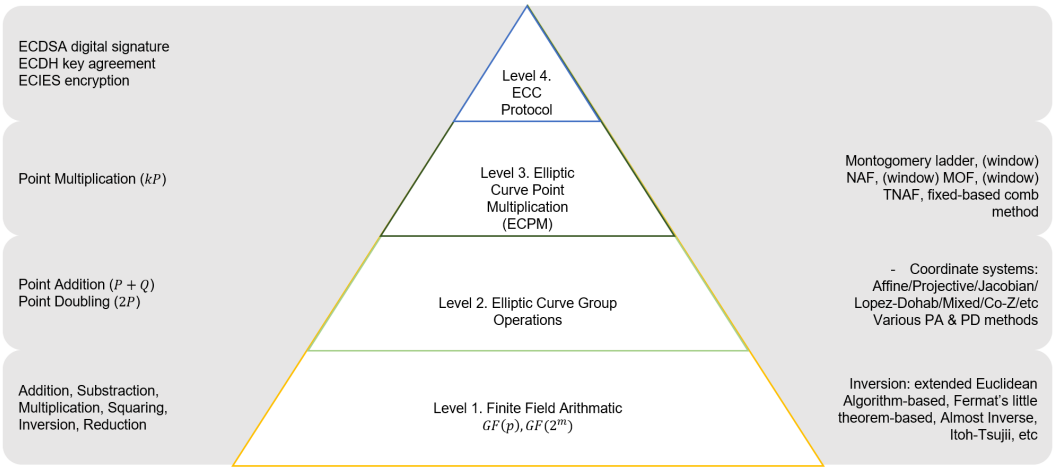

Figure 1: Elliptic Curve-based cryptography (ECC) implementation hierarchy [LK21]

In curve over binary fields standardized in [KG13], an elliptic curve E, over F_{2^n} is specified by the coefficient $a, b \in F_{2^n}$ of its defining equation $y^2 + xy = x^3 + ax^2 + b$, where $a \in F_2$ and $b \in F_{2^n}^*$. The elements are represented as polynomials with coefficients in F_2 and a degree less than n in the polynomial representation for F_{2^n}. Computation use that $F_{2^n} \cong F_2[z]/m(z)$, where $m(z) \in F_2[z]$ is an irreducible polynomial of degree n, i.e., in all computations are done modulo $m(z)$. Tabel 2 shows a list of irreducible polynomials for binary finite fields.

Table 1: List of irreducible polynomials for binary fields F_{2^n} used in this paper

Degree	Irreducible polynomial	Source
8	$x^8 + x^4 + x^3 + x + 1$	[CFA+05]
16	$x^{16} + x^5 + x^3 + x + 1$	[CFA+05]
127	$x^{127} + x + 1$	[CFA+05]
163	$z^{163} + z^7 + z^6 + x^3 + 1$	[KG13]
233	$z^{233} + z^{74} + 1$	[KG13]
283	$z^{283} + z^{12} + z^7 + z^5 + 1$	[KG13]
571	$z^{571} + z^{10} + z^5 + z^2 + 1$	[KG13]

Points on binary elliptic curve are tuples $P = (x, y) \in F_{2^n}^*$ satisfying the curve operation along with a special point O called the "point at infinity" or the netral element. The negative of a point $P_1 = (x_1, y_1)$ is $-P_1 = (x_1, y_1 + x_1)$, so that $P_1 + (-P_1) = O$. Two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2) \neq \pm P_1$ are added to produce $P_1 + P_2 = P_3 = (x_3, y_3)$, as $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$, $y_3 = (x_2 + x_3)\lambda + x_3 + y_2$ with $\lambda = \frac{y_1 + y_2}{x_1 + x_2}$ and $P_1 \neq -P_1$ is doubled to produce $[2]P_1 = (x_3, y_3)$ as $x_3 = \lambda^2 + \lambda + a, y_3 = x_1^2 + (\lambda + 1)x_3$ with $\lambda = x_1 + \frac{y_1}{x_1}$.

2.2 Elliptic Curve Discrete Logarithm Problem

The fundamental problem that enables the establishment of elliptic curve-based cryptography protocols and algorithms is the Elliptic Curve Discrete Logarithm Problem (ECDLP). Examples include the elliptic Curve Diffie-Hellman (ECDH) key exchange, the Elliptic Curve Digital Signature Algorithm (ECDSA), and the El Gamal encryption method. The basic concept of ECDLP [Yan15] Let E be an elliptic curve over the Finite Field F_p, say, in the short Weierstrass equation $E : y^2 = x^3 + ax + b$ (mod p), with S and T the two

points in the elliptic curve group $E(F_p)$. The ECDLP is to find the integer $k = log_T S \in Z$, or $k = log_T S$ (mod p) such that $S = kT \in E(F_p)$, or $S = kT$ (mod p), assuming k exists. Recovery of k without knowing S and T is traditionally regarded as a problematic issue, with the fastest classical approach still running in exponential time. However, since Peter Shor demonstrated that k could be acquired using a quantum computer in polynomial time, contemporary public-key cryptosystems may be deemed insecure.

2.3 Shor's Algorithm

Peter Shor addressed two challenges in his major work from 1994: factoring a big integer into prime factors and finding discrete logarithms over finite groups [Sho94]. The first is the issue that underpins the RSA cryptosystem's security. Simultaneously, the latter is the foundation of Diffie-Hellman and digital signature algorithm methods, while its enlargement to the elliptic curve is the security foundation of ECC. Despite its importance, Shor's algorithm for the integer factorization problem, also known as the Quantum Factoring Algorithm, appears to have received more notice than Shor's algorithm for the ECDLP. Shor's example described how to break RSA using a quantum computer to factor integers in polynomial time, as proposed by Peter Shor in 1994 to utilize quantum computers to break traditional asymmetric encryption. Some research also demonstrated how to apply his method to any discrete logarithm issue; for instance, in [PZ03] [RNSL17] [BBvHL21]. This study refer to [RNSL17] [BBvHL21] to utilize Shor's methodologies in quantum cryptanalysis elliptic curves Point addition, Figure 2 and Figure 3 show Shor's algorithm to calculate the discrete logarithm elliptic curve [RNSL17].

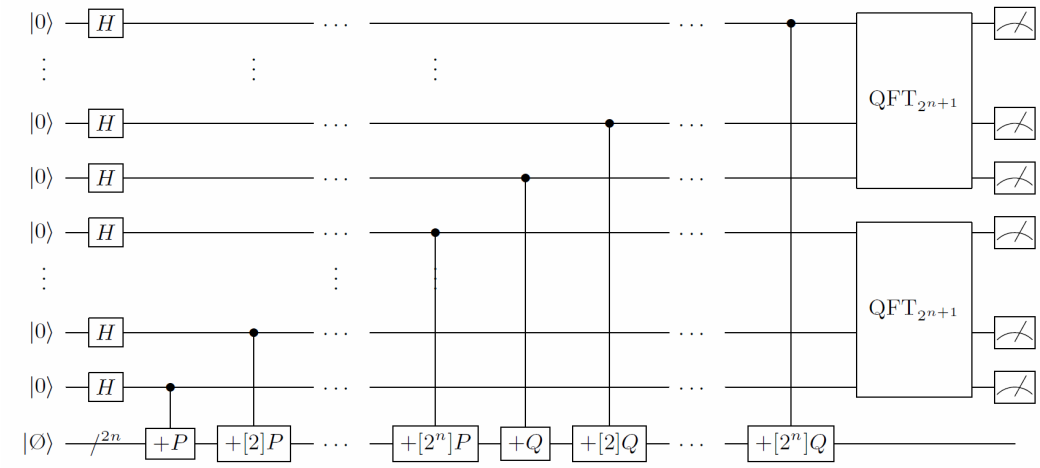

Figure 2: Shor's algorithm to calculate the discrete logarithm elliptic curve generated by a point P [RNSL17]

2.4 Theoretical Background

The use of arithmetic operations in the Quantum field ranges from ordinary quantum algorithms to quantum cryptanalysis; hence, fast iImplementation elliptic curve and Shor's method for discrete logarithm problems (Shor's ECDLP) are two examples of thriving research in quantum computing. To conduct a variety of quantum algorithms, the quantum circuit requires reversible gates that correspond to arithmetic operations like add, subtract, multiply, divide, or invert. Quantum computing, which employs qubits as the

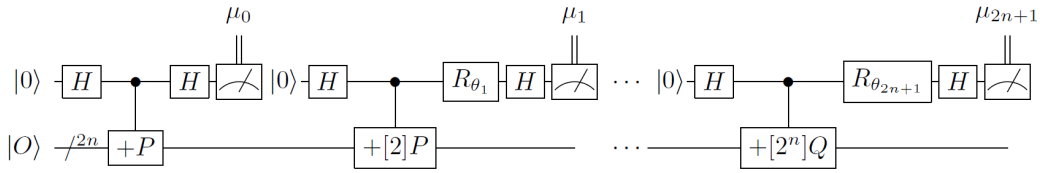

Figure 3: Shor's algorithm for finding elliptic curve logarithm with semiclassical Fourier transform [RNSL17]

information unit, necessitates the creation of a new component since it differs physically from conventional frequency-based computing, which uses bits as the minimal information unit. A bit and a qubit are distinguished because a bit may be either 0 or 1, but a qubit can be in a state of superposition. A qubit is allowed to exist in both states 0 and 1, meaning it can be in two states simultaneously. Reversible gates are bijective in quantum computing. Unlike typical gates, every input state corresponds to a single output state, requiring an equal number of input and output qubits.

The basic states of a qubit are expressed in the ket notation $|0\rangle$ and $|1\rangle$. The superposition is a weighted sum of these two base states, where $alpha|0\rangle + beta|1\rangle > i$ and $|\alpha^2| + |\beta^2| = 1$. The probability of seeing 0 in the measurement is equal to $|\alpha^2|$. In uniform superposition, a qubit with $|\alpha| = |\beta|$, such as $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$ has an equal probability of being measured as 0 or 1. The superposition of n states is achieved by combining n qubits : $\sum_{i=0}^{2^n-1} \alpha_i|(q_{n-1,i}, q_{n-2,i} \ldots, q_{1,i}, q_{0,i})_2\rangle$ with $\sum_{i=0}^{2^n-1} |\alpha^2| = 1$, where $i = (q_{n-1,i}, q_{n-2,i} \ldots, q_{1,i}, q_{0,i})_2$ is the presentation of i in base 2. Measuring outputs i with probability $|\alpha_i^2|$. For simplicity we write $\sum_{i=0}^{2^n-1} |\alpha^2| = 1$ in the following.

The NOT, CNOT, and Swap gates are required for quantum circuit construction, whereas Shor's circuit requires the Hadamard gate (H), phase shift gate (R), and measurement. The following gates are necessary for elliptic-curve computations and Shor's circuit: First, the NOT Gate has a single, qubit either input or output. For input $|0\rangle$, the output is $|1\rangle$, and vice versa. In fact, it is the polar opposite. The Second, the CNOT, or Pauli−X, is referred to as the XOR or F_2- addition gates. The gate involves two input qubits a and b and computes the addition of one input to the other qubit input. Note that, outputs is replacing one input qubit: $(a, b) \rightarrow (a \bigoplus b, b)$. Written as $a \leftarrow CNOT(a, b)$, It has its inverse: $(a \bigoplus b \bigoplus b, b) = (a, b)$ if applied twice. Stated as $c \leftarrow TOF(a, b, c)$ in algorithms, the Toffoli (TOF) gate, the third gate, is identical to AND or $F2-$multiplication. $(a, b, c) \rightarrow (a, b, c \bigoplus (a.b))$ takes three qubits as inputs, adds the result of the first two qubits' multiplication to the third qubit, and outputs the remaining qubits as themselves. It is also the polar opposite of itself. Fourth, the swap operation, switches two quantum bits a and b; after the swap, qubit a becomes b, and qubit b becomes a. Lastly, is Hadamard and Phase gate. The following operation specifies the Hadamard transformation: $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. A phase gate, often known as a phase shift, is a device that considers the following unitary operation: $|0\rangle \rightarrow |0\rangle, |1\rangle \rightarrow exp(i\phi)|1\rangle$. As a result, $|0\rangle$ remains unaltered while $exp(i\phi)$ alters the phase of $|1\rangle$.

Quantum algorithms are made up of operations on qubit registers. Input qubits, output qubits, and auxiliary qubits are the three types of qubits. The three types of qubits are input qubits, output qubits, and ancillary qubits. The input and output qubits hold the input and will have the output when the algorithm is completed, but the ancilla qubits are employed by the algorithm but do not hold the input or output. To preserve reversibility, inversion circuits, for example, require a high number of ancilla qubits. Following that, ancilla qubits should be left uncomputed, lowering the cost (i.e., depth, total gate).

2.5 Quantum Arithmetic Operations

As the research in quantum computers faces rapid advancements, methods to realize various quantum arithmetic operations become increasingly important, including performing those elliptic curve related operations. This section introduces numerous quantum arithmetic operations relevant to the subject; further study related to implementation see i.e. [VH19], [BBvHL21].

2.5.1 Addition and Binary Shift

Fast addition formulae for points on an elliptic curve over a finite binary field F_{2^n} aim at reducing the number of (expensive) F_{2^n}-operations. In the quantum field, for additional research instance [CDKM04, Tak09, TMCK21], prior to the present time, study for a faster addition is still underway.

To achieve $\mid \alpha\rangle \mid \beta\rangle \mid 0\rangle \mapsto \mid \alpha\rangle \mid \beta\rangle \mid \alpha+\beta\rangle$, where the sum is saved in a different register, we can first add $\mid \alpha\rangle$ to $\mid 0\rangle$, then add $\mid \beta\rangle$, i.e., $2n$ CNOT gates and depth 2 sufficient. The Addition of two polynomials in F_2 of degree at most $n-1$ uses n CNOT gates with depth 1 with operation without ancillary qubits, and the addition result replaces either of the inputs since each Addition. Addition for polynomials over F_2 is the same as Addition for the element of the field F_{2^n} since it is coefficient-wise. For polynomials in $F_2[z]$, multiplication by z is a shift of the coefficient vector. This requires no quantum computation by doing a series of swaps. To multiply a polynomial $g(z)$ of degree at most $n-1$ by z, a finite field needs to be followed by a modular reduction by a fixed irreducible weight-ω degree-n polynomial $m(z)$. ω will always be 3 or 5 for our needs.

2.5.2 Multiplication

The classical way of multiplication is known as the schoolbook algorithm, and it has $O(n^2)$ space and $O(n^{1.59})$ time complexity. The conventional Karatsuba algorithm for multiplying polynomial and multi-precision integers, for example, has seen significant improvements. The Karatsuba algorithm has an $O(n^{1.59})$ time complexity and an $O(n)$ spatial complexity. Multiplication related to this study will be described in more detail in section 3.

2.5.3 Schoolbook Multiplication

The simplest technique is to multiply in a Schoolbook. The first and second polynomials that take n^2 Toffoli gates for two polynomials of degree at most $n-1$ are used to calculate the number of pairs of qubits. The result must be stored independently from the input in $2n-1$ qubits; unlike the previous circuits, we cannot replace any of the inputs with the result since the Toffoli gate demands a separate output. Let's say we wish to use a weight-k and degree-n odd polynomial to conduct modular reduction steps. In such a situation, we may use $(n-1)(k-2)$ CNOT gates with no auxiliary qubits (by using the modular shift approach after every n multiplication). To store the result, n qubits are utilized.

2.5.4 Karatsuba Multiplication

Required to multiply two polynomials in a finite field, the multiplication method has a wealth of variations. A for classical computers, often based around Karatsuba's multiplication method, finding preceded by Gauss. Gauss showed that multiplying two distinct variables $(a+bi)$ and $(c+di)$ in R using just three multiplications as follows:

$$(a+bi)(c+di) = ac - bd + ((a+b)(c+d) - ac - bd)i \tag{1}$$

Karatsuba and Ofman construct long integer multiplication as demonstrated in equation 2, building on a previous idea by Gauss equation (equation 1). Karatsuba divides each input polynomial into high- and low-degree portions for polynomials f, g in $R[x]$ with degrees less than $2k$: $f = f_0 + f_1 x^k$, $g = g_0 + g_1 x^k$, where f_0, f_1, g_0, g_1 are polynomials in $R[x]$ of degree less than k.The three intermediate products are calculated as follows: $\alpha = f_0.g_0, \beta = f_0.f_1, \gamma = (f_0.f_1).(g_0.g_1)$. Finally, these products are combined to produce the Karatsuba multiplier, which is the product of f and g [**?**]:

$$f.g = \alpha + (\gamma - \alpha - \beta)x^k + \beta x^{2k} \tag{2}$$

2.5.5 Squaring

Squaring in F_{2^n} based on following equation :

$$\left(\sum_{i=0}^{n-1} a_i z^i)\right)^2 = \sum_{i=0}^{n-1} a_i z^{2i} \mod m(z) \tag{3}$$

This would be no cost operation if the mod operation were not considered, as we only need to shuffle zeroes across our registers. In F_{2^n}, there are two approaches: a circuit that saves the result of squaring a polynomial of degree at most $n-1$ in n independent qubits, or a circuit that substitutes the input with the result. Because squaring is bijective, the second technique is only practicable for finite fields with $2n$ members. We exploit the fact that squaring is a linear map, which we can write as a n by n matrix to square and replace the input. We produce a lower triangular, upper triangular, and permutation matrix using an LUP decomposition, which can be translated into a circuit with at most $n^2 - n$ CNOT gates and several swaps.

2.5.6 Inversion

As a crucial process in a quantum circuit, the division stage is the most computationally intensive. Hence, it must be thoroughly investigated as part of finite field division. Some approaches have been proven to solve inversion and division in binary fields, such as extended Greatest Common Divisor (extended GCD), Fermat's Little Theorem (FLT), Euclid's algorithm, and Kaliski's binary inversion algorithm. Inversion operations can be performed using a variety of approaches; GCD and FLT are two of the most prevalent. Banegas et al. approach division by a field element as multiplication by the inverse of that element, which is similar to Itoh-approach Tsujii's to the FLT method. The FLT-based inverses described in section 4 are based on altering numerous stages in Banegas et al. [BBvHL21] to minimize the depth, which we compare in the last section. The Modification step attempts to give an alternate and effective method of reducing gate quantum circuit consumption while addressing the lowest time complexity.

3 Karatsuba Multiplication

This part introduces multiplication as it relates to the research, the preceding subsection briefly describes Schoolbook (sec 2.5.3) and Karatsuba (sec 2.5.4) multiplication. Following that, in sections 3.1 and 3.2, we introduce multiplication in a quantum circuit corresponding to Karatsuba derived by Van Hoof Study and Banegas et al. [BBvHL21]. Section 3.2 gives derived multiplication as a modification from [VH19]. Finally, Section 5.2.1 will present a comparison of all multiplication operations.

3.1 Related Work on Space-efficient Karatsuba Multiplication

Classic Karatsuba in-place multiplication in binary polynomial rings is describe below. For given input polynomials $f(x)$ size up to n, $g(x)$ size up to n and $h(x)$ in $2n$, output is $h + f.g$. For k, such that $\frac{n}{2} \leq k < n$, we can split each polynomial as follows: $f = f_0 + f_1 x^k, g = g_0 + g_1 x^k$ and $h = h_0 + h_1 x^k + h_2 x^2 k + h_3 x^{3k}$ for chosen $k = \frac{n}{2}$. Compute $\alpha = f_0.g_0, \beta = f_1.g1$ and $\gamma = (f_0 + f_1)$ to add those in Karatsuba multiplication $h + f.g = h + \alpha + (\gamma + \alpha + \beta)x^k + \beta^{2k}$. α, β, γ can be separated as f and g to get non overlap result, which is useful for checking correctness: $h + f.g = (h_0 + \alpha_0) + (h_1 + \alpha_0 + \alpha_1 + \beta_0 + \gamma_0)x^k + (h_2 + \alpha_1 + \beta_0 + \beta_1 + \gamma_1)x^{2k} + (h_3 + \beta_1)x^{3k}$. Following Equation 4 is rewrite the proving equation.

$$h + f.g = h + (1 + x^k)\alpha + x^k\gamma + x^k(1 + x^k)\beta \tag{4}$$

In quantum computing, integer multiplication is required to conduct the Shors method for factoring integers, and it is at the core of the process [PRM17]. They were yielding an asymptotic reduction of the amount of space required from $O(n^{1.585})$ to $O(n^{1.427})$; parent et al. improved reversible and quantum circuits for Karatsuba-based integer multiplication in their research. In [Gid19], the space Karatsuba multiplication on a quantum computer complexity improves by Gidney's research to obtain $O(n^{log3})$ gate complexity. Futhermore, on Karatsuba in quantum, Roche suggested multiplication of polynomials, an improved technique with the same $O(n^{1.59})$ but a substantially lower O(log n) space complexity [Roc09].

Besides Roche in [Roc09], Hoof's study [VH19] claimed as space-efficient and ran the algorithm recursively to the KMULT algorithm rather than a modular multiplier. Being space-efficient variants of karatsuba multiplication methods, Hoof's research split Karatsuba into two parts. First part is for given $f(x), g(x), h(x)$ calculate $h + f.g$. Second, given $k, f(x), g(x), h(x)$ with $k >$ max $(deg(f), deg(g))$ compute $h + (1 + x^k)f.g$. Deriving from Karatsuba calculation, Hoof's steps modular multiplication approach reaches fewer Toffoli-count and fulfills in space required.

Rather than using schoolbook multiplication, approaches such as Karatsuba multiplication can be used to speed up large number multiplication. Algorithm modular multiplication from [VH19] is a sub-quadratic Toffoli gate count multiplication technique for binary polynomials in finite fields that yield only $O(n^{log_2(3)})$ Toffoli Gate usage. Hoof's study uses $O(n^2)$ CNOT Gates, $O(n^{log_2(3)})$ Toffoli Gates, and $3n$ total qubits. Both $2n$ qubits for the input, f, g and n separate qubits for the output,h. Hoof clearly uses the Karatsuba approach to building base algorithm, such as KMULT$_n$ (algorithm 3), KMULT ((algorithm 4), CONSTMODMULT (algorithm 5), MODSHIFT (algorithm 6) [VH19]. As the basis of this research, the algorithm in Hoof's study is provided in the appendix. Hoof's research implements it in space-efficient quantum polynomial multiplication for binary finite fields with sub-quadratic Toffoli gate count. Banegas et al. utilizes space-efficient variants of Karatsuba introduced by Van hoof which use CNOT gates $O(n^2)$, $On^{log_2(3)}$ Toffoli Gates and $3n$ total qubit [BBvHL21]. Still, field multiplications, in particular, should be thoroughly optimized because they consume a considerable amount of resources and time.

3.2 Proposed Improvement on Space-efficient Karatsuba Multiplication

This study complements considered prior research and compares the quantum multiplication inspired by prior work (such as [Roc09, RNSL17, VH19, HJN+20, BBvHL21]). This subsection shows the improvement quantum circuit from previous Karatsuba research, utilizing space-efficient multiplication form by Van Hoof. The space-efficient multiplier is a base for Banegas et al.'s research in their notable concrete implementation.

Table 2: Improved Modular Multiplier (Algorithm ModMult_Imp)

Line	H in ModMultImp
1	α
2-3	$(x^k)^{-1}\alpha \mod m(x)$
4	$((x^k)^{-1}\alpha + \beta) \mod m(x)$
5	$((x^k)^{-1}\alpha + \beta)(1 + x^k) \mod m(x)$
6-10	$((x^k)^{-1}\alpha(1 + x^k) + \beta(1 + x^k) + \gamma \mod m(x)$
11	$(((x^k)^{-1}\alpha(1 + x^k) + \beta(1 + x^k) + \gamma)(x^k) \mod m(x)$
12	$\alpha(1 + x^k) + x^k\beta(1 + x^k) + x^k\gamma \mod m$

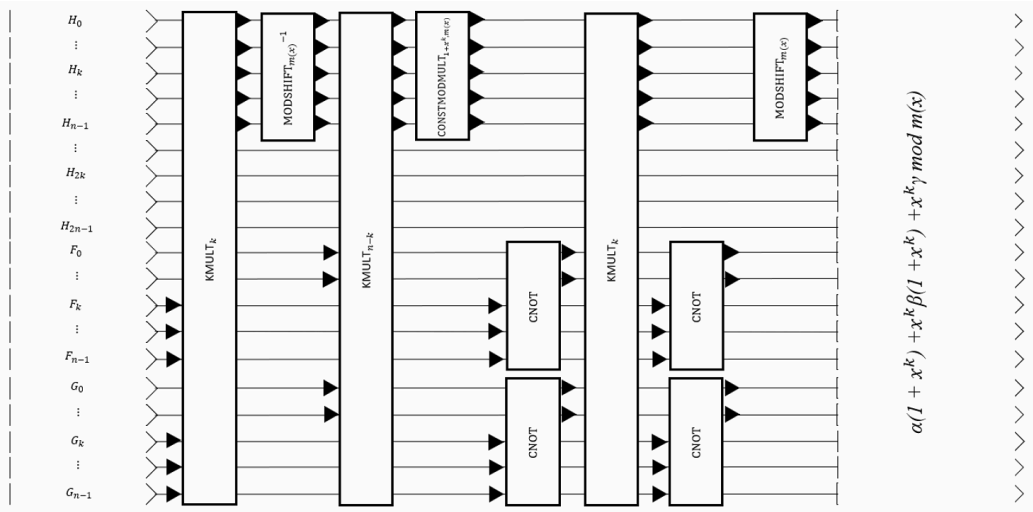

Figure 4: Improvement Quantum Circuit Karatsuba Multiplication

Furthermore, provide a new extension for improving quantum circuit by refining the implementation of multiplication in quantum circuit. As shown in table 2, this study offers modified steps in modular multiplier over 12 steps from Hoof [VH19] or 13 steps Banegas et al. [BBvHL21] with analysis the needs of gate of each function on algorithm with consideration of space efficient with reversible algorithm. As shown in tabel 2 the final result for modular multiplication is $\alpha(1 + x^k) + x^k\beta(1 + x^k) + x^k\gamma \mod m$. With similar basis algorithm in reversible quantum circuit [BBvHL21], such as KMULT$_n$, KMULT$_x k, k$, CONSTMODMULT$_f(x), m(x)$, MODSHIFT$_m(x)$, the improved modular multiplication algorithm works as follows: The algorithm step chases the α value in the first line of array H. Instead of computing $(1 + x^k)\beta \mod m$, the improved multiplier algorithm's in $2 - 3$ lines calculate $(x^k)^{-1}\alpha \mod m(x)$. Next, $((x^k)^{-1}\alpha + \beta)(1 + x^k) \mod m(x)$ will be processed and added to the result in stages 5. The step 6-10 is to get multiply $(1 + x^k)$ as a polynomial constant to α, β, yielding $\alpha(1 + x^k) + x^k\beta(1 + x^k) + x^k\gamma \mod m$. In contrast, we do not recursively call CONSTMODMULT$_f(x), m(x)$ inverse as needed in Tabel hoof, which seeks to erase the constant $(1+x^k)$. To maintain the quantum circuit space-efficient, this strategy reduces CNOT used and quantum circuit depth without modifying the best Toffoli result from previous work. For improvement of the Hoof Karatsuba to the better optimization, we consider overall the Toffoli, CNOT gate, and Depth, quantum circuit yielded modular multiplication is based on the following Algorithm 1 in Table 2.

Algorithm 1 MODMULT_Imp. As inspired from [VH19] [BBvHL21], we propose a reversible algorithm for multiplying two polynomials in $\mathbb{F}_2[x]/m(x)$ with an irreducible polynomial $m(x)$.

Fixed input : A constant integer n to indicate field size, $k = \lfloor \frac{n}{2} \rfloor$. $m(x)$ of degree n as a field polynomial. The LUP-decomposition precomputed for multiplication by $1 + x^k$ modulo $m(x)$.
Quantum input : Three binary polynomials $f(x), g(x), h(x)$ of degree up to $n-1$ store in array F, G, H respectively of size n.
Result: F and G as input, H as $h + f \cdot g \bmod m$.
1: $H[0..n-1] \leftarrow \text{KMULT}_k(F[0..k-1], G[0..k-1], H[0..n-1])$
2: **for** $i = 0, ..., k-1$**do**
3: $\quad H[0..n-1] \leftarrow \text{MODSHIFT}^{-1}_{m(x)}(H[0..n-1])$
4: $H[0..n-1] \leftarrow \text{KMULT}_{n-k}(F[k..n-1], G[k..n-1], H[0..n-1])$
5: $H[0..n-1] \leftarrow \text{CONSTMODMULT}_{1+x^k, m(x)}(H[0..n-1])$
6: $F[0..n-k-1] \leftarrow \text{CNOT}(F[0..n-k-1], F[k..n-1])$
7: $G[0..n-k-1] \leftarrow \text{CNOT}(G[0..n-k-1], G[k..n-1])$
8: $H[0..n-1] \leftarrow \text{KMULT}_k(F[0..k-1], G[0..k-1], H[0..n-1])$
9: $G[0..n-k-1] \leftarrow \text{CNOT}(G[0..n-k-1], G[k..n-1])$
10: $F[0..n-k-1] \leftarrow \text{CNOT}(F[0..n-k-1], F[k..n-1])$
11: **for** $i = 0, , k-1$ **do**
12: $\quad H[0..n-1] \leftarrow \text{MODSHIFT}_{m(x)}(H[0..n-1])$

4 FLT-based Inversion

In an elliptic curve point addition operation, inversion circuit is essential due to its use as the component to perform a division operation [HJN+20]. The division itself, for the use of the finite field, will consist of multiplication and an inversion and circuit (as opposed to the case for the standard case, which generally comprises additions and its variants, as in [TMCVH19,TVMC16]), which can be considered as the most computationally intensive operation [RNSL17]. Therefore, it should be thoroughly investigated to obtain improvements, which can reduce the overall cost of point addition circuit. Several approaches have been proposed for quantum inversion in general: standard Euclidean algorithm [PZ03], (Kaliski's) binary GCD algorithm [RNSL17] and its improvement [HJN+20], all of which are proposed for the prime field use. For the binary field, extended GCD and Fermat's Little Theorem (FLT) has recently been proposed by [BBvHL21]. In this section, we focus on discussing the previous work on quantum FLT-based inversion, then present our improvement. In particular, we employ the approach of FLT-based inversion proposed in [LPWK22], the incorporate our improved Karatsuba multiplication instead of the standard Schoolbook or the other Karatsuba in [BBvHL21], which in our simulation, yields lower depth and width compared to the previous works.

4.1 Related Work on FLT-based Inversion

Proposing and comparing two different inversion algorithms (i.e., extended GCD and FLT), previous work [BBvHL21] have discussed Itoh-Tsujii's approach to the FLT algorithm, which is well-known for cryptography in classical computing to reduce the number of multiplications in the FLT inversion. For a prime number p and integer x, the basic FLT in modular arithmetic asserts that $xp = x \mod p$. The Fermat primality test is based on FLT, first stated in 1640, as one of the most fundamental findings of basic number theory. For binary fields, FLT may be extended to $f^{2n-2} = f^{-1} \mod m(x)$, where $m(x)$ have n as the degree. Then the inversion can be performed by n multiplications

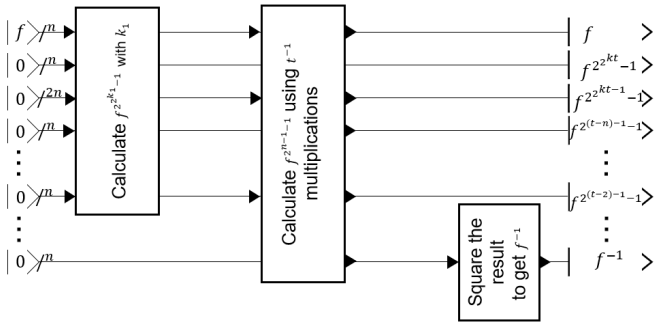

Figure 5: High-level scheme of a three-stage inversion inferred from [BBvHL21]; algorithm modifications and register placement follows [LPWK22].

and $n-1$ squarings by equation $f^{2^n-2} = f^{2^1} f^{2^2} f^{2^3} \dots f^{2^{n-1}}$. For the quantum circuit implementation, Banegas et al [BBvHL21] have described the derivation by adhering to the Itoh-Tsujii [IT88]'s two observations as shown in Equations 5 and 6, to reduce the cost below $2\log(n)$ multiplications and to $n-1$ squarings.

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

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

By the condition that: $n-1$ is denoted 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 $k_1 = \lfloor \log(n-1) \rfloor$, the FLT-based quantum inversion circuit can be constructed in three steps as follows [BBvHL21]:

1. Calculate $f^{2^{2^{k_1}-1}}$ with k_1 multiplications by utilizing 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 $\left\{ \dots \left\{ (f^{2^{2^{k_1}-1}})^{2^{2^{k_2}}} (f^{2^{2^{k_2}}-1}) \right\}^{2^{2^{k_3}}} \dots \right\}^{2^{2^{k_t}}}$ $(f^{2^{2^{k_t}-1}})\}$ using $t-1$ multiplications.
3. Square output of the above steps to obtain the inverse, i.e., f^{-1}.

Using the above steps, Banegas et al [BBvHL21] derived an inversion circuit comprising a series of squarings, multiplications, and inverse squarings to accomplish an inversion operation. For a detailed description of their FLT-based inversion, their algorithm and quantum circuit example can be found on Section 6.2 of their paper: on Algorithm 2 (line 1-16) and Circuit 6, respectively.

Subsequently, Larasati et al [LPWK22] proposed to modify the steps in from [BBvHL21] to minimize the overall depth of the circuit. Specifically, they propose to remove the inverse squaring to create a more streamlined construction in a waterfall approach [LPWK22], and postponing the uncomputation to the end. This results in the reduction of the CNOT count without increasing the depth of the T-gate (i.e., T-depth), which contributes to a lower overall depth. Their approach comes with a tradeoff of a larger qubit size (i.e., circuit width), but overall depth is minimized, which is advantageous for time-efficient implementation, as discussed in the Section IV-B of their paper. Additionally, they also compared their result with [BBvHL21] via simulation in Qiskit. Note that since they focus on evaluating the inversion algorithm, their Qiskit simulation utilize a standard Schoolbook as the multiplication method for both scenarios (i.e., theirs and Banegas et al's).

4.2 Proposed Improvement on FLT-based Inversion

For achieving a lower depth implementation, the FLT-based inversion algorithm in [LPWK22] can be taken into consideration. However, it comes with a tradeoff of a larger qubit size, as shown in their simulation. In this paper, we propose to incorporate our improved multiplication circuit MODMULT_Imp to the FLT−based inversion algorithm in [LPWK22]. Instead of employing a standard Schoolbook multiplication or the Karatsuba multiplication in [BBvHL21], we substitute the multiplier with our MODMULT_Imp. Considering the fact that multiplication is used repeatedly and, in fact, is one of the core components of the inversion circuit, the depth reduction using our method is manyfold. The method of our approach is presented in Algorithm 2.

Algorithm 2 INV_Imp. Our proposed algorithm for FLT-based inversion, which modifies [BBvHL21, LPWK22]. For simplicity, we follow the notation style of [BBvHL21]

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

1: **for** $i = 1, , k_1$ **do** //stage 1
2: CNOT($f_{2*(i-1)+1}, f_{2*(i-1)}$)
3: **for** $i = 1, , t-1$ **do**
4: SQUARE($f_{2*(i-1)+1}$)
5: MODMULT_Imp($f_{2*(i-1)+2}, f_{2*(i-1)+1, f_{2*(i-1)}}$)
6: **for** $s = 1, , t-1$ **do** //stage2
8: **for** $k = 1, , 2^{k_{s+1}}$ **do**
9: SQUARE($f_{2*(i-1)+1}$)
10: MODMULT_Imp($f_{2*(i-1)+2}, f_{2*(i-1)+1, f_{2*(i-1)}}$)
11: **if** $t = 1$ **then**
12: swap (f_{k_1}, f_k)
13: SQUARE(f_k) //stage 3

5 Evaluation

This section provides the information about our evaluation method and setup, along with the rationale behind our evaluation approach, then presents the result.

5.1 Evaluation Method & Setup

To evaluate our proposed Karatsuba multiplication and FLT-based inversion, we build each circuit in Qiskit based on our algorithms (i.e., Algorithms 1 and 2), then obtain the resource analysis. Note that due to a large number of qubits, we can not perform full simulation, i.e., we do not perform simulation on the quantum hardware or the Qiskit simulator. Rather, we construct the corresponding circuits and then utilize the built-in function in Qiskit to count the resource requirement. Then, to further evaluate our method, we compare our result with the previous works.

For multiplication, we compare our technique (Table 3) to the quantum Karatsuba variation presented by Van Hoof et al., [VH19], which may be regarded as the most recent

space-efficient quantum multiplier for the binary fields. That multiplication method is the base for the underlying multiplier in Banegas et al., [BBvHL21], the most recent work on quantum cryptanalysis for binary elliptic curves. In detail, we refer to the result of Van Hoof's multiplier directly from the result presented in Table 5 of their paper [VH19], whereas our method's result is derived from our Qiskit implementation. Additionally, for Table 3, we employ the exactly same performance metrics as in [VH19]. Due to limited time and resources, we run the quantum circuit for n of 8, 16, 127, 163, 233, 283, and 571 with their respective irreducible polynomials follow [BBvHL21], whereas the rest of the values are derived from interpolation. Additionally, Schoolbook multiplication is also taken from [BBvHL21] as a comparison to the baseline multiplication method. Furthermore, we then compare our result with two other works on quantum binary multiplications, i.e., [KS15] and [MMCP09] following the previous comparison in [VH19], presented on Table 4.

In terms of inversion, we also build our variant of the FLT-based algorithm circuit in Qiskit, then compare our inversion algorithm with two of the latest FLT-based inversion for binary fields, i.e., Banegas et al., [BBvHL21] and Larasati et al., [LPWK22]. In [BBvHL21], their presented result (i.e., Table 2 of their paper) is already for division instead of inversion; thus, we can not directly use the result. So in this paper, for inversion evaluation, we make a slightly different approach from the multiplication evaluation explained above. Particularly, we recreate the Banegas' FLT algorithms (Algorithm 2 lines 1-16 of [BBvHL21]) in Qiskit, still we replace their original multiplication (i.e., Van Hoof's Karatsuba [VH19]) to our proposed multiplication, in which at the result in Table 5 denoted as BAN*. Nevertheless, for [LPWK22], we can rebuild their circuit in Qiskit and run it according to their original setting (i.e., using Schoolbook multiplication)), referred in the table as LAR*. Instead of directly acquiring the result from their paper, we need to rerun to obtain their Toffoli resource count, which is not provided in their result.

Lastly, we incorporate our result for the quantum cryptanalysis in binary elliptic curves and discuss the relevant aspects of our study.

5.2 Evaluation Result

5.2.1 Result & Comparison of Karatsuba Multiplications

As shown in Table 3, our proposed improvement of Karatsuba multiplication yields a lower CNOT than that of Van Hoof's space-efficient Karatsuba algorithm [VH19] while maintaining the same number of Toffoli gates. Our construction also results in a lower overall depth. Further looking into Table 4, Van Hoof's [VH19] bounds quadratically to $O(n^2)$ CNOT gates, $O(n^{log_2(3)})$ Toffoli gates and $3n$ total qubits. With our improved multiplication, we reach better optimization for CNOT count, which bounds to $O(n^{log_2(3)})$, while maintaining the similar Toffoli count and qubit count. In other words, the result shows that our construction enhances the current quantum space-efficient Karatsuba multiplier for the aforementioned evaluation metrics. Note that there are instances where our Toffoli count is even slightly lower for some degrees, i.e., for n of 127, 163, 233, 283, 571, which we save 2, 32, 16, 32, and 42 Toffoli gates, respectively. We are uncertain about the cause of this difference, but it might have a relation to the different environments that we use (ours in Qiskit while Van Hoof's in on Microsoft QDK). Additionally, compared to the schoolbook multiplication, both Karatsuba multipliers enormously reduce the number of Toffoli gates.

Further, presented in Table 4 is the additional comparison with other quantum binary multipliers: those of Kepley and Steinwandt [KS15] and Maslov et al., [MMCP09] taken example for several n. In terms of the gate count (Toffoli and CNOT counts), our work and [MMCP09] bounds to $O(n^{log_2(3)})$, but we obtain lower qubit count of $3n$ as opposed to $O(n^{log_2(3)})$. Additionally, among others, the work of [MMCP09] gives the lowest cost

Table 3: Detailed Multiplication Comparison with Previous Work. For several instances of our proposed algorithm, TOF gate count for Schoolbook multiplication, and Karatsuba version of hoof's paper, we used CNOT, TOF gate, and Depth upper limits with field polynomials in Table 1, and the lowest CNOT count irreducible polynomial.

Degree	Schoolbook	This Work			Previous Work [VH19]		
	TOF	TOF	CNOT	Depth	TOF	CNOT	Depth
2	4	3	10	3	3	11	10
4	16	9	32	9	9	49	36
8	64	27	102	82	27	220	139
16	256	81	376	226	81	725	396
32	1,024	243	1,194	246	243	2,371	1,204
64	4,096	729	3,973	754	729	7,160	3,312
127	16,129	2,183	12,659	3,151	2,185	21,028	9,063
128	16,384	2,187	13,223	3,311	2,187	21,898	9,586
163	26,569	4,355	22,080	6,976	4,387	38,143	18,647
233	54,289	6,307	36,238	25,225	6,323	66,974	32,505
256	65,536	6,561	36,005	17,080	6,561	66,107	27,756
283	80,089	10,241	54,099	23,225	10,273	91,737	43,249
571	326,041	31,139	166,982	45,979	31,171	274,967	124,999
1024	1,048,576	59,049	487,345	66,470	59,049	600,089	240,678

Table 4: Multiplication Comparison of Resource Analysis with Several Literatures. In both Toffoli and CNOT gates, and qubit count, this study is comparable with those of Kepley and Steinwandt [KS15], and Maslov et al. [MMCP09].

n	Toffoli Count				CNOT Count				Qubit Count			
	Ours	[VH19]	[KS15]	[MMCP09]	Ours	[VH19]	[KS15]	[MMCP09]	Ours	[VH19]	[KS15]	[MMCP09]
4	9	9	9	16	32	49	22	3	12	12	17	12
16	81	81	81	256	334	725	376	45	48	48	113	48
127	2183	2185	2185	16129	12597	21028	13046	126	381	381	2433	381
256	6561	6561	6561	65536	44005	66107	57008	765	768	768	7073	768
n	$O(n^{log_2 3})$	$O(n^{log_2 3})$	$O(n^{log_2 3})$	n^2	$O(n^{log_2 3})$	$O(n^2)$	$O(n^{log_2 3})$	$O(n)$	$3n$	$3n$	$O(n^{log_2 3})$	$3n$

for CNOT, still, it comes with a quadratically bounded Toffoli count —, which is only as efficient as schoolbook multiplication [PRM17] —while others are bound to $O(n^{log_2(3)})$. By this information, our proposed multiplier is considerably beneficial for use in quantum arithmetic operations, and quantum cryptanalysis.

5.2.2 Result & Comparison of FLT-based Inversions

Table 5 presents the relevant FLT-based inversion result. Compared to BAN* (i.e., FLT algorithm by Banegas et al., [BBvHL21] but with multiplier changed to ours), we obtain lower CNOT count and lower overall depth, with the tradeoff of larger qubit size. Therefore, for a time-efficient implementation, our method is more suitable. Additionally, in comparison to Larasati et al., [LPWK22] (denoted as LAR in the table), the utilization of our proposed multiplier greatly reduces the high Toffoli count and qubit count that exist in [LPWK22] due to their use of schoolbook multiplication. Therefore, our approach can be thought of as an alternative solution to obtain the advantage of breaking down multiplication (using the Karatsuba approach) to give a lower Toffoli count while maintaining the lower depth inherent in [LPWK22].

Inversion is a generic arithmetic operation that may be employed repeatedly in a calculation, such as in the well-known Shor's algorithm, notably in its variation for the

Table 5: Comparison of Resource Analysis with Previous Work FLT Based inversion. In both Toffoli and CNOT gates, qubit and Depth, This Work Based on Qiskit Simulation Result. Note that, BAN* (FLT algorithm by Banegas et al. with multiplier changed to ours) while LAR* (FLT algorithm by ours, multiplier schoolbook and result rerun with transpile method).

n	Toffoli Count			CNOT Count			Qubit Count			Overall Depth		
	Ours	LAR*	BAN*	Ours	LAR*	BAN*	Ours	LAR*	BAN*	Ours	LAR*	BAN*
8	108	256	108	1012	268	1064	57	89	41	13	13	12
16	486	1536	486	6534	750	6792	161	257	113	24	24	23
127	24013	177419	24013	663541	8920	668272	2287	3684	1525	143	143	142
163	39195	239121	39195	946681	48499	984652	2772	4339	1631	178	178	270
233	63070	542890	63070	2010283	47303	2034266	4195	6525	2564	249	249	270
283	112651	880979	112651	3343751	143444	3473260	5661	8774	3397	301	301	528
571	404807	4238533	404807	15138845	341985	15445474	9408	20557	7995	592	592	1042

ECDLP. The bigger the circuit, the more significant depth savings are possible. It is very beneficial since lowering the overall depth, not only the T-depth, is very important to reduce computational time in a quantum computer. In addition, considering that maintaining long decoherence has still been a crucial issue, whereas expanding qubit size has seen significant advancement in the past years, it will be advantageous to reduce the depth and gate count of an inverse operation. Note that in both the prior work (FLT inversion by Banegas et al.) and our method, the uncomputation of ancilla registers is to be done. As a result, both the ancilla qubits/registers and the depth will be emptied at the conclusion. In our instance, the cost of saving on depth is likely to be more prominent as described in Table 5.

In particular, both the earlier work (FLT inversion by Banegas et al.) and our technique should be used to uncomputation ancilla registers. As a result, the ancilla qubits/registers will be cleared at the conclusion, but the depth will be doubled. In our instance, the cost of saving on depth is likely to be more significant. Inversion is a generic arithmetic operation that may be employed repeatedly in a calculation, such as in the well-known Shor's algorithm, notably in its variation for the ECDLP. The bigger the circuit, the more significant depth savings are possible.

5.2.3 Result of Binary ECC Point Addition with Improved Multipliers and Inversions

Now that we have obtained the resource count for our improved Karatsuba multiplications and FLT-based inversions, we can estimate the cost of a point addition in the binary case via numerical calculation. To do this, we refer to the point addition steps described in Algorithm 3 of [BBvHL21] and evaluate the exact same metrics. The result of a single point addition is as presented in Table 6. In summary, our approach yields significantly lower depth compared to Banegas et al., [BBvHL21]. For instance, for the highest degree (i.e., $n = 571$), our depth is below a hundred thousand, whereas the previous work is already above ten million. Furthermore, it is worth noting that our implementation also significantly reduces the total number of Toffoli gates, up to tenfold for the highest degree. Nevertheless, as stated in [BBvHL21] that their Toffoli gate count is based on their utilization of GCD-based rather than their FLT-based inversion due to the lower qubit size in their GCD. Additionally, note that our approach comes with a tradeoff of higher qubit size and CNOT gates. Similarly to [BBvHL21], we then analyze our result of point addition with a windowing approach, with the evaluation as shown in Table 7. With windowing, the depth is minimized even further. Take the highest degree for example; the previously one billion Toffolis required in a standard point addition, with windowing, it can be reduced to around 125 million (In [BBvHL21], windowing technique reduces their Toffoli gates from ten billion to approximately one million). Due to our advantages in terms

Table 6: Comparison of Resource Analysis of Single Step Point Addition with Previous Work. In both Toffoli and CNOT gates, qubit and Depth, This Work Based on Qiskit Simulation Result. Note that, BAN (Point Addition algorithm by Banegas et al. [BBvHL21]).

n	Ours	BAN	Ours			BAN			Ours	BAN
	qubits	qubits	Toffoli	CNOT	depth	Toffoli	CNOT	depth	Toffoli	Toffoli
8	74	68	348	2,734	210	7,360	3,522	8,562	6,264	132,480
16	194	125	1,344	15,924	623	21,016	11,686	25,205	45,696	714,544
127	2,320	904	52,773	1,352,497	7,010	559,141	497,957	776,234	13,509,888	143,140,096
163	2,805	1,157	96,299	2,137,063	14,632	893,585	827,623	1,262,280	31,586,072	293,095,880
233	4,228	1,647	152,067	4,480,745	46,702	1,669,299	1,615,287	2,406,230	71,167,356	781,231,932
283	5,694	1,998	267,115	7,376,571	34,792	2,427,369	2,359,187	3,503,964	151,721,320	1,378,745,592
571	13,167	4,015	935,883	32,888,178	93,142	8,987,401	9,081,061	13,238,554	1,070,650,152	10,281,586,744

of depth (and Toffoli count as well), our approach can be an option for a reduced-depth implementation that gives a significant advantage.

Table 7: Our TOF estimates for various field sizes using $2(2^l - 1)$ TOF gates per lookup. l is optimized for this. Field polynomials from Tabel 1.

n	l	Toffoli gate	Lookups	Total Toffoli gates	pre-computed points
8	7	1,387	24	6,991	512
16	8	8,046	36	24,072	1,536
127	13	1,107,478	120	3,005,062	163,840
163	13	2,598,650	156	4,985,290	212,992
233	14	5,320,116	204	11,767,647	557,056
283	14	11,214,906	252	19,094,269	688,128
571	16	68,306,437	432	124,928,677	4,718,592

For a complete Shor's algorithm, the complexity is described as follows. As described in [BBvHL21], the required number of point additions (i.e., to perform a complete *double scalar multiplications*, is $2n + 2$, in which each step comprising two divisions, four (including the two in the division), and three controlled additions. By this requirement, the approximate number of Toffoli gates for our approach is

$$4n^3 + 3n\log((3) + 1) + 25n^2\log(n) + 2n^2 + O(n^{\log((3)+1)})$$

In terms of qubit, the approximate number of qubit required by utilizing our proposed Karatsuba and FLT-based inversion as the subcircuits is

$$4n + 7n\log(n) + 7$$

Regarding depth, we perform extrapolation from our result in Table 6, which approximates to

$$7n + 6n\log_2(3)$$

6 Conclusions

In this paper, we present a quantum cryptanalysis for binary elliptic curve. Different from previous work by Banegas et al., [BBvHL21] that focus on optimizing the number of qubits, our aim here is to reduce the overall depth of the circuit to obtain the resource estimates from a time-efficient implementation perspective. To achieve a reduced depth, we propose a new, lower-depth implementation variant of Karatsuba multiplication and FLT-based inversion, then build and analyze the resource cost in Qiskit. In summary, our Karatsuba multiplication achieves lower the CNOT count than the previous work [VH19], in which ours bounds to $O(n^{log_2(3)})$ while preserving the similar number of Toffoli gates and qubits, and results in a lower circuit depth. Furthermore, we then employ our proposed multiplier to the FLT-based inversion of [LPWK22], obtaining lower CNOT count and lower overall depth, with a larger qubit size tradeoff. Finally, we incorporate our result for quantum cryptanalysis of binary elliptic curve, yielding a depth estimate of $7n + 6n\log_2(3)$, qubit count of $4n + 7n\log(n) + 7$, and Toffoli count of $4n^3 + 3n\log((3) + 1) + 25n^2\log(n) + 2n^2 + O(n^{\log((3)+1)})$. The result also shows that our lower depth implementation also reduce the number of Toffoli gates, with the tradeoff of higher number of qubits and CNOT gates. In the future, we plan to investigate on the optimal time-space tradeoff to achieve a balanced implementation.

7 Appendix

These algorithms are exactly from [VH19], used as a subcircuit of our Karatsuba multiplication.

Algorithm 3 KMULT_n. Reversible algorithm for multiplication of 2 polynomials of degree up to $n-1$.

Fixed input : A constant integer n to indicate polynomial size and an integer $k < n < 2k$ with $k = [\frac{n}{2}]$ for $n > 1$ and $k = 0$ for $n = 1$, to indicate upper and lower half
Quantum input : Two binary polynomials $f.g$ of degree up to $n-1$ store in array F and G respectively of size n. A binary polynomial h degree up to $2n-2$ stored in array H of size $2n-1$
Result: F and G as input, H as $h + fg$

if $n > 1$ **then**
  $H[0..3k-2] \leftarrow \text{KMULT1xk}_{k,k}(F[0..k-1]G[0..k-1]H[0..3k-2])$
  $H[k..2n-2] \leftarrow \text{KMULT1xk}_{k,n-1}(F[k..n-1]G[k..n-1]H[k..2n-2])$
  $F[0..n-k-1] \leftarrow \text{CNOT}(F[0..n-k-1], F[k..n-1])$
  $G[0..n-k-1] \leftarrow \text{CNOT}(G[0..n-k-1], G[k..n-1])$
  $H[k..3k-2] \leftarrow \text{KMULT}_k(F[0..k-1]G[0..k-1]H[0..3k-2])$
  $G[0..n-k-1] \leftarrow \text{CNOT}(G[0..n-k-1], G[k..n-1])$
  $F[0..n-k-1] \leftarrow \text{CNOT}(F[0..n-k-1], F[k..n-1])$
else
  $H[0] \leftarrow \text{TOF}(F[0], G[0], H[0])$

Algorithm 4 $\text{KMULT1xk}_{k,n}$. Reversible algorithm for multiplication of 2 polynomials of degree up to $n-1$ by the polynomial $1 + x^k$.

Fixed input : A constant integer $k > 0$ to indicate part size as well as an integer $n \leq k$ to indicate polynomial size. $l = \max(0, 2n-1-k)$ is the size of h_2 and $(fg)_1$. In the case of Karatsuba we will have either $n = k$ or $n = k-1$.
Quantum input : Two binary polynomials $f(x).g(x)$ of degree up to $n-1$ store in array F and G respectively of size n. A binary polynomial h degree up to $k + 2n - 2$ stored in array $/h$ of size $2k+1$
Result: F and G as input, H as $h + (1 + x^k)fg$

if $n > 1$ **then**
  $H[k..k+l-1] \leftarrow \text{CNOT}(H[k..k+l-1], H[2k..2k+l-1])$
  $H[0..k-1] \leftarrow \text{CNOT}(H[0..k-1], G[k..n-1], H[k..2k-1])$
  $H[k..3k-2] \leftarrow \text{KMULT}_n(F[0..n-1], G[0..n-1], H[k..2k+l-1])$
  $H[0..k-1] \leftarrow \text{CNOT}(H[0..k-1], H[k..2k-1])$
  $H[0..k-1] \leftarrow \text{CNOT}(H[k..k+l-1], H[2k..2k+l-1])$
else
  $H[0] \leftarrow \text{CNOT}(H[0], H[k])$
  $H[0] \leftarrow \text{TOF}(F[0], G[0], H[0])$
  $H[0] \leftarrow \text{CNOT}(H[0], H[k])$

Algorithm 5 $\text{CONSTMODMULT}_{f(x),m(x)}$, from Amento et all. Reversible algorithm for in-place multiplication by a nonzero constant polynomials $\mathbb{F}_2[x]/m(x)$ with $m(x)$ an irreducible polynomial.

Fixed input : A binary LUP -decomposition L, U, P^{-1} for a binary n by n matrix that corresponds to multiplication by constant polynomial $f(x)$ in the field $\mathbb{F}_2[x]/m(x)$.
Quantum input : A polynomials $g(x)$ of degree up to $n-1$ store in array G
Result: G as $f.g$ in the field $\mathbb{F}_2[x]/m(x)$.

for $i = 0..n-1$
do
  for $j = i+1..n-1$ **do**
    if $U[i,j] = 1$ **then**
      $G[i] \leftarrow \text{CNOT}(G[i], G[j])$
for $i = n-1..0$
do
  for $j = i-1..0$ **do**
    if $L[i,j] = 1$ **then**
      $G[i] \leftarrow \text{CNOT}(G[i], G[j])$
for $i = 0..n$
do
  for $j = i+1..n-1$ **do**
    if $P^{-1}[i,j] = 1$ **then**
      SWAP$(G[i], G[j])$
      SWAP column i and j of P^{-1}

Algorithm 6 $\text{MODSHIFT}_{m(x)}$from[[BBvHL21]]. Reversible algorithm for in-place multiplication by x in $\mathbb{F}_2[x]/m(x)$ where $m(x)$ an irreducible polynomial.

Fixed input : A constant polynomial $m(x)$ of degree n, represented as an ordered list M of length ω that contains the degrees of the nonzero tems in descending order.
Quantum input : A binary polynomial $g(x)$ of degree up to $n-1$ store in array G
Result: G as $g.x$ in the field $\mathbb{F}_2[x]/m(x)$.

for $i = 1..n$ **do**
  SWAP$(G[n-i], G[n-i-1])$
for $i = 1..\omega - 2$ **do**
  $G[Mi] \leftarrow \text{CNOT}(G[Mi], G[0])$

References

[APWK22] Asep Muhamad Awaludin, Jonguk Park, Rini Wisnu Wardhani, and Howon Kim. A high-performance ecc processor over curve448 based on a novel variant of the karatsuba formula for asymmetric digit multiplier. Cryptology ePrint Archive, Report 2022/371, 2022. https://ia.cr/2022/371.

[BBvHL21] Gustavo Banegas, Daniel J Bernstein, Iggy van Hoof, and Tanja Lange. Concrete quantum cryptanalysis of binary elliptic curves. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 451–472, 2021.

[Ber06] Daniel J Bernstein. Curve25519: new diffie-hellman speed records. In *International Workshop on Public Key Cryptography*, pages 207–228. Springer, 2006.

[CDKM04] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004.

[CFA+05] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC press, 2005.

[CMRR19] Lily Chen, Dustin Moody, Andrew Regenscheid, and Karen Randall. Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters. Technical report, National Institute of Standards and Technology, 2019.

[Gid19] Craig Gidney. Asymptotically efficient quantum karatsuba multiplication. *arXiv preprint arXiv:1904.07356*, 2019.

[Ham15] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. *IACR Cryptol. ePrint Arch.*, 2015:625, 2015.

[HJN+20] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. Improved quantum circuits for elliptic curve discrete logarithms. In *International Conference on Post-Quantum Cryptography*, pages 425–444. Springer, 2020.

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

[KG13] Cameron F Kerry and Patrick D Gallagher. Digital signature standard (dss). *FIPS PUB*, pages 186–4, 2013.

[KMV00] Neal Koblitz, Alfred Menezes, and Scott Vanstone. The state of elliptic curve cryptography. *Designs, codes and cryptography*, 19(2):173–193, 2000.

[KO62] Anatolii Alekseevich Karatsuba and Yu P Ofman. Multiplication of many-digital numbers by automatic computers. In *Doklady Akademii Nauk*, volume 145, pages 293–294. Russian Academy of Sciences, 1962.

[KS15] Shane Kepley and Rainer Steinwandt. Quantum circuits for $f_{2^n}-$multiplication with subquadratic gate count. *Quantum Information Processing*, 14(7):2373–2386, 2015.

[LHT16] Adam Langley, Mike Hamburg, and Sean Turner. Rfc 7748: Elliptic curves for security. *Internet Research Task Force (IRTF)*, 2016.

[LK21] Harashta Tatimma Larasati and Howon Kim. Quantum cryptanalysis landscape of shors algorithm for elliptic curve discrete logarithm problem. In *International Conference on Information Security Applications*, pages 91–104. Springer, 2021.

[LPWK22] Harashta Tatimma Larasati, Dedy Septono Catur Putranto, Rini Wisnu Wardhani, and Howon Kim. Reducing the depth of quantum flt-based inversion circuit. *Cryptology ePrint Archive*, 2022. https://ia.cr/2022/463.

[MMCP09] Dmitri Maslov, Jimson Mathew, Donny Cheung, and Dhiraj K Pradhan. An o (m2)-depth quantum algorithm for the elliptic curve discrete logarithm problem over gf (2m) a. *Quantum Information & Computation*, 9(7):610–621, 2009.

[PRM17] Alex Parent, Martin Roetteler, and Michele Mosca. Improved reversible and quantum circuits for karatsuba-based integer multiplication. *arXiv preprint arXiv:1706.03419*, 2017.

[PZ03] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *arXiv preprint quant-ph/0301141*, 2003.

[Res18] E. Rescorla. The transport layer security (tls) protocol version 1.3. RFC 8446, RFC Editor, August 2018.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 241–270. Springer, 2017.

[Roc09] Daniel S Roche. Space-and time-efficient polynomial multiplication. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 295–302, 2009.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[Tak09] Yasuhiro Takahashi. Quantum arithmetic circuits: a survey. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 92(5):1276–1283, 2009.

[TMCK21] Himanshu Thapliyal, Edgard Muñoz-Coreas, and Vladislav Khalus. Quantum circuit designs of carry lookahead adder optimized for t-count t-depth and qubits. *Sustainable Computing: Informatics and Systems*, 29:100457, 2021.

[TMCVH19] Himanshu Thapliyal, Edgard Munoz-Coreas, TSS Varun, and Travis Humble. Quantum circuit designs of integer division optimizing t-count and t-depth. *IEEE Transactions on Emerging Topics in Computing*, 2019.

[TVMC16] Himanshu Thapliyal, TSS Varun, and Edgard Munoz-Coreas. Quantum circuit design of integer division optimizing ancillary qubits and t-count. *arXiv preprint arXiv:1609.01241*, 2016.

[VH19] Iggy Van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic toffoli gate count. *arXiv preprint arXiv:1910.02849*, 2019.

[WH11] Erich Wenger and Michael Hutter. Exploring the design space of prime field vs. binary field ecc-hardware implementations. In *Nordic Conference on Secure IT Systems*, pages 256–271. Springer, 2011.

[Yan15] Song Y Yan. Quantum computing for elliptic curve discrete logarithms. In *Quantum Computational Number Theory*, pages 173–228. Springer, 2015.