박사학위논문

암호 알고리즘에 대한 고도화된 양자 공격 연구

ECC, AES, SHA-2/3에 대한 분석

2026년

한 성 대 학 교 대 학 원

정보컴퓨터공학과

정보시스템공학전공

장 경 배

박사학위논문
지도교수 서화정

암호 알고리즘에 대한 고도화된 양자 공격 연구

ECC, AES, SHA-2/3에 대한 분석

Improved Quantum Attacks on Cryptography

2025년 12월 일

한 성 대 학 교 대 학 원

정 보 컴 퓨 터 공 학 과

정보시스템공학전공

장 경 배

박사학위논문
지도교수 서화정

암호 알고리즘에 대한 고도화된 양자 공격 연구

ECC, AES, SHA-2/3에 대한 분석

Improved Quantum Attacks on Cryptography

위 논문을 정보컴퓨터공학 박사학위 논문으로 제출함

2025년 12월 일

한 성 대 학 교 대 학 원

정 보 컴 퓨 터 공 학 과

정보시스템공학전공

장 경 배

장경배의 정보컴퓨터공학 박사학위 논문을 인준함

2025년 12월 일

심사위원장 박명서 (인)

심 사 위 원 석병진 (인)

심 사 위 원 서화정 (인)

심 사 위 원 김수리 (인)

심 사 위 원 이동재 (인)

ABSTRACT

Improved Quantum Attacks on Cryptography The case of ECC, AES, and SHA-2/3

Jang, Kyung-Bae
Major in Information System Engineering
Dept. of Information and Computer Engineering
The Graduate School
Hansung University

Quantum computing is expected to mark one of the next major leaps forward in computational science. Although large-scale, fully functional quantum computers have not yet been realized, the need to evaluate the security of cryptographic systems against powerful quantum adversaries continues to grow.

This thesis explores advanced quantum attacks on modern cryptographic primitives, focusing on Elliptic Curve Cryptography (ECC), the Advanced Encryption Standard (AES), and the SHA-2/3 hash families.

It is well known that ECC becomes vulnerable once large-scale quantum computers capable of running Shor's algorithm are realized, and we present state-of-the-art quantum cryptanalysis in that direction.
Furthermore, this work provides refined estimates for NIST's post-quantum security categories: Levels 1, 3, and 5 correspond to Grover's key searches on AES-128/192/256, while Levels 2 and 4 relate to quantum collision search on SHA-2/3.

Overall, this thesis contributes to the design and optimization of quantum circuits for cryptanalysis, underscoring the importance of reassessing the post-quantum security of widely deployed cryptographic systems as scalable quantum computing becomes increasingly realistic.

Keywords: Quantum Computing, AES, SHA-2/3, ECC

Table of Contents

List of Tables

List of Figures

List of Algorithms

Ⅰ. Introduction

In the current state of cryptography, quantum computers are viewed as a major threat on the horizon. This is because quantum computers can naturally and efficiently handle certain types of problems that are difficult for classical computers.

1.1 Quantum Threats to Cryptography

Unfortunately, some of these problems are closely related to the mathematical foundations of several cryptographic systems. While these problems are still considered hard for classical computers (and therefore secure at the moment), their security may no longer hold once practical quantum computers appear.

Despite its popularity and widespread use, Elliptic Curve Cryptography (ECC) also suffers from the same type of quantum weakness as RSA. More specifically, Shor's algorithm [Sho94] for solving discrete logarithms in prime-order finite fields can be extended to elliptic curves as well, making ECC vulnerable.

Over the past few years, we have seen rapid improvements in quantum computing. As a result, it is widely accepted that ECC-based systems will not survive against attackers with a capable quantum computer. This has motivated the cryptographic community to search for quantum-secure alternatives to traditional public key schemes, which has led to what is now known as Post-Quantum Cryptography (PQC).

Symmetric key cryptography will likely not remain completely safe either. Depending on the design, a symmetric cipher may also show serious weaknesses against quantum attacks. One important reason is that many PQC schemes use symmetric key components internally (in addition to their usual standalone use). This can be seen in the current list of

candidates in the NIST PQC standardization process. For example, the Public Key Encryption (PKE) & Key Encapsulation Mechanism (KEM) finalist CRYSTALS-KYBER and the digital signature finalist CRYSTALS-DILITHIUM use SHA-3 in various stages. Even if the main construction of these schemes is based on a quantum resistant problem, an attacker might still be able to target the symmetric component alone, making it the actual security bottleneck of the overall design.

1.2 Quantum Security Analysis

NIST defines five post-quantum security levels, each based on symmetric key primitives (variants of AES for PKE & KEM, and variants of SHA-3 for digital signatures). As pointed out in [JNRV20, Section 1], this requires clear and concrete estimates of the quantum resources needed by an attacker.

Therefore, understanding the "*generic*" quantum security level of cryptographic schemes has become an important topic of research. A central tool for quantum attackers includes Shor's algorithm, which finds hidden periods underlying the mathematical foundations of ECC and RSA, and Grover's search [Gro96], which reduces the cost of exhaustive key search to roughly the square root of the classical complexity (with high probability).

1.3 This Thesis

With the ongoing and rapid progress toward building a practical quantum computer, a natural question arises: "*how difficult or easy it will actually be to break modern cryptographic systems?*". This thesis humbly strives to answer the question by presenting new results on the quantum cryptanalysis of ECC, AES, and SHA-2/3, which has only been possible by standing on the shoulder of giants (including but not limited

to [BBVHL20, PWLK22, TT23, XZL+20, LXZZ21, LSL+19, LWF+22, ZH22, LXX+23, LPZW23, YWS+24, LLLC23, KHJ18, ADMG+17, MSDM22, HS20, SJS23, LKL+24]).

In this thesis, we make a modest attempt to establish boundaries for the post-quantum security levels defined by NIST [NIS16, NIS22], ranging from Levels 1 to 5. Levels 1, 3, and 5 correspond to the quantum circuit size needed to recover a solution key for AES-128, AES-192, and AES-256 using Grover's algorithm, respectively. For levels 2 and 4, which relate to collision finding for the SHA-2 and SHA-3 hash families, the relevant quantum circuit sizes have not yet been defined (as far as we are aware, only classical circuit sizes are available).

We optimize quantum circuits for ECC (strictly speaking, point addition on elliptic curves) and representative symmetric key cryptographic algorithms, namely the variants of AES, SHA-2, and SHA-3, by exploring multiple optimization strategies. Our goal is to reduce the overall cost in various metrics, such as qubit count, gate count, and circuit depth (Toffoli depth, full depth), as well as trade-offs (e.g., Toffoli depth × qubit count, full depth × qubit count). Throughout this work, we carefully compare several design options and choose those that offer the most favorable balance.

Ⅱ. Background

2.1 Notations and Objective

Throughout this thesis, we use the following shorthand notations: #NOT (reversible NOT gate count), #CNOT (CNOT count), #Toffoli (Toffoli count), TD (Toffoli depth), #T (T count), Td (T-depth), #1qCliff (Clifford gate count), #Measure (measurement count), G (total number of gates), FD (full depth), and M (qubit count). The full depth is directly related to the execution time of a circuit. The importance of circuit depth is also emphasized in NIST's post-quantum security requirements. In estimating the cost of quantum attacks, NIST considered only gate count and depth (not the number of qubits) as evaluation metrics [NIS16, NIS22].

In this work, we optimize quantum circuits for the quantum cryptanalysis, with specific attention to qubit count, Toffoli depth, and full depth. We additionally examine trade-off metrics such as Toffoli depth × qubit count (the TD-M cost) and full depth × qubit count (the FD-M cost). In most cases, our circuits achieve the smallest Toffoli depth, full depth, TD-M cost, and FD-M cost, thus offering meaningful improvements over the existing state-of-the-art.

2.2 Quantum Gates

The Hadamard gate (H) produces a superposition state by mapping $|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Several classical logic gates also have quantum counterparts, as shown in Figure 2.1. The X gate (Pauli-X) flips the state of a qubit, i.e., $X(|0\rangle) = |1\rangle$ and $X(|1\rangle) = |0\rangle$ (analogous to the classical NOT gate). The CNOT gate performs a controlled-NOT operation: it flips the target qubit when the control

qubit is $|1\rangle$ (similar to a classical XOR). When used on qubits in superposition, CNOT can also create entanglement.

The Toffoli gate (CCNOT) generalizes this behavior by flipping the target qubit only when both control qubits are $|1\rangle$ (classical AND-controlled XOR). It plays an important role in quantum computation and is universal for classical reversible logic. The Clifford + T gate set consisting of H, S (phase), CNOT, and T gates forms a universal gate set capable of generating arbitrary unitary operations. Although the Clifford gates by themselves do not provide universality, adding the T gate makes universal quantum computation possible. In addition, Clifford gates are typically easier to implement, while T gates (and their T-depth) impose greater resource costs, making their reduction crucial for fault-tolerant quantum computing.

A Toffoli gate can be decomposed into Clifford and T gates, and the associated cost and depth vary depending on the chosen construction [Sel13, AMM+13, HLZ+17]. In our notation, a Clifford gate refers to CNOT and 1qCliff gates. The T-depth, which is important for quantum error correction, is determined by the number of T layers when a Toffoli gate is decomposed. After designing each quantum circuit, we decompose all Toffoli gates to estimate detailed quantum resources. In this work, when evaluating resource costs, we use the decomposition from [AMM+13], in which one Toffoli gate consists of (8 Clifford gates + 7 T gates), T-depth 4, and full depth 8 (see Figure 2.2). Many other decomposition methods are available.

We also incorporate the quantum AND gate from [JNRV20]. This AND gate decomposes into (11 Clifford gates + 4 T gates), T-depth 1, full depth 8, and requires 1 ancilla qubit. Its inverse operation (i.e., the uncompute, denoted $\text{AND}^\dagger$) depends on the measurement outcome of the target qubit and is counted as (5 Clifford gates + 1 measurement gate)

for resource estimation.

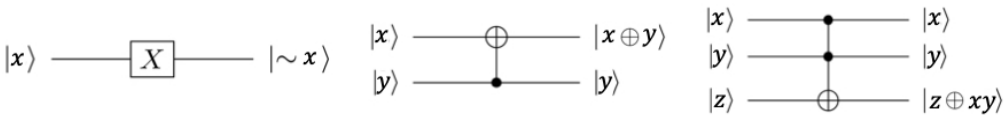


Figure 2.1: Common quantum gates (left: X gate, middle: CNOT gate, right: Toffoli (CCNOT) gate).

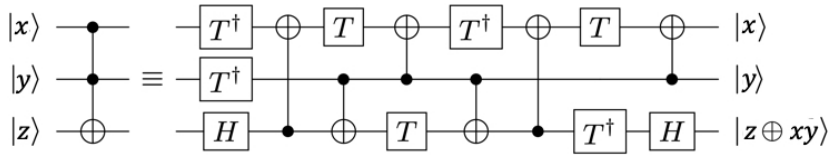


Figure 2.2: Decomposition of Toffoli gate in [AMM+13].

2.3 NIST Security Levels

NIST proposed the following guideline in [NIS16, NIS22] to handle the uncertainties involved in estimating the post-quantum security strengths of PQC candidates: "*Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on AES and collision search on SHA-2/3.*" Based on this principle, NIST defined the following security levels to evaluate post-quantum security:

① Level 1: The cipher should not be easier to break than AES-128.
② Level 2: The cipher should not be easier to break than SHA-256.
③ Level 3: The cipher should not be easier to break than AES-192.
④ Level 4: The cipher should not be easier to break than SHA-384.
⑤ Level 5: The cipher should not be easier to break than AES-256.

Each category is defined using a reference primitive that is relatively

easy to analyze. This primitive serves as a baseline for the various metrics that NIST considers important for evaluating practical security. In particular, NIST assigned a separate category to each security requirement, as summarized in Table 2.1.

Security	Cryptographic algorithm	Attack complexity (gates × full depth)
Lev. 1	AES-128	2^{157}/MAXDEPTH
Lev. 2	SHA-2/3-256	Undefined
Lev. 3	AES-192	2^{221}/MAXDEPTH
Lev. 4	SHA-2/3-384	Undefined
Lev. 5	AES-256	2^{285}/MAXDEPTH

Table 2.1 NIST post-quantum security levels.

For these categories, the estimated quantum resources were derived by examining the quantum gate count and full depth (including MAXDEPTH constraints). Levels 1, 3, and 5 correspond to key-search attacks on AES, while levels 2, 4, and the extended level correspond to collision searches on SHA-2/3. In an earlier NIST document [NIS16], the resource estimates for Grover's key search on AES were taken from the work of Grassl et al. [GLRS16] for levels 1, 3, and 5. More recently, NIST updated these levels based on Jaques et al. [JNRV20], who reduced the required quantum resources by improving the quantum circuits for AES.

However, the quantum resources associated with levels 2 and 4 (and the extended level) appear to be much less studied. We believe that this gap is caused by the lack of clear and consistent estimates for collision search on SHA-2/3—unlike what has been done for AES in [JBK+25, GLRS16, JNRV20, LPZW23]. Moreover, there is no optimized quantum circuit that fully aligns with NIST's methodology for estimating quantum attack costs. Historically, many works focused primarily on reducing the qubit count, which is not necessarily the correct objective for Grover's

algorithm (particularly when parallelization is considered).

2.4 NIST MAXDEPTH

Alongside this, NIST introduced a parameter called MAXDEPTH to limit the depth of quantum circuits. Although precise bounds for MAXDEPTH have not been clearly defined, the values 2^{40}, 2^{64}, and 2^{96} are generally viewed as reasonable indicators, based on the anticipated computational capability of a quantum machine over a year, a decade, or a millennium. Under this guideline, one would expect the depth of a Grover search circuit to stay below 2^{96} (the highest assumed MAXDEPTH bound). If the required depth exceeds this bound, following three approaches can be considered [KHJ18]:

① Outer parallelization: Restrict the circuit depth to $\leq 2^{96}$, which reduces the probability of successfully recovering the key.
② Inner parallelization: Divide the full search space into multiple smaller subspaces, each processed by a shallow-depth circuit, again reducing the success probability.
③ Ignore MAXDEPTH and compute cost as-is (see [KHJ18, Table 2]).

It is worth mentioning that earlier implementations such as [ZWS+20, LPS20, ASAM18, HS22] also did not seem to consider the MAXDEPTH constraint. Both outer and inner parallelization reduce the success probability because they lower the number of Grover iterations. Outer parallelization terminates the algorithm once the depth limit is reached, yielding suboptimal solution probabilities. Inner parallelization shortens each instance by reducing the search space, which similarly lowers the success rate. Additionally, parallelizing Grover search is generally inefficient: the analysis in [Zal99] shows that using S parallel instances

offers only a $\sqrt{S}$ reduction in depth. This is why we evaluate the FD^2-M and Td^2-M metrics for Grover's search in this thesis.

2.5 Methodology

We use the ProjectQ quantum programming framework to implement and simulate our quantum circuits. Since ProjectQ supports the classical simulation of Toffoli gates, we can check the correctness of large-scale quantum circuits by verifying their test vectors. A Toffoli gate can be simulated directly and is decomposed only when estimating resource costs. In contrast, ProjectQ does not provide classical simulation for AND gates. Therefore, we first validate the implemented circuit using Toffoli gates, and then replace the upper part of the circuit with AND gates when performing the final resource estimation.

Ⅲ. Quantum Attacks on ECC

Author's Contribution. This chapter is based on the results in [JSB+25a]. I was the main contributor to the design and optimization of the quantum circuits used in the quantum security evaluation of ECC.

3.1 Introduction

Elliptic Curve Cryptography (ECC) relies on the algebraic structure of elliptic curves over finite fields and has become an essential tool in modern public key cryptography. The idea of using elliptic curves for cryptographic purposes was introduced independently by Miller [Mil85] and Koblitz [Kob87] more than thirty years ago. Today, ECC is widely employed in many communication protocols. For example, key exchange [DH22] and digital signatures [ElG85, JMV01].

The security of ECC depends on the hardness of the discrete logarithm problem over elliptic curve groups, known as the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECC is attractive mainly because the best-known algorithms for solving ECDLP [GG16] run in exponential time with respect to the input size. As a result, ECC can offer strong security with significantly smaller key sizes compared to RSA (and other public key systems) [RSA78]. For example, recommendations by Barker [Bar20], on behalf of NIST, note that a 224-bit elliptic curve provides roughly the same level of classical security as a 2048-bit RSA modulus. It is worth noting that classical security in bits is typically lower than the key length itself, unlike what one sees in symmetric key ciphers.

Despite its popularity, ECC shares the same quantum vulnerability as RSA. Specifically, Shor's algorithm [Sho94], which solves discrete logarithms over finite fields of prime order, can be adapted to elliptic

curves as well.

3.2 Contributions

This chapter introduces improved quantum cryptanalysis for binary elliptic curves. Our main goal is to reduce the depth of quantum circuits, while the number of qubits is treated as a secondary consideration. Compared with prior work, our constructions achieve the smallest Toffoli depth and full circuit depth reported so far. For the depth − qubit count product, our in-place point addition provides improvements of approximately 73%−81%, and our out-of-place point addition achieves reductions of more than 92%. These improvements arise from several optimizations applied across three logical levels (Section 3.3).

At the component level, we refine the basic building blocks using depth-efficient quantum circuits for binary field operations. This includes an out-of-place squaring method with an additional optimization described in Section 3.3, along with the depth-optimized multiplication technique proposed by Jang et al. [JKL+23].

At the combination level, the division algorithm benefits from an inversion method based on *Fermat's Little Theorem* (FLT). The inversion process reuses qubits through reverse operations, which keeps the depth unchanged. Additional improvements occur in situations where multiple multiplications must be performed consecutively, such as during inversion, because the multiplication method of Jang et al. [JKL+23] efficiently reuses ancilla qubits.

At the architecture level, we present two point addition techniques, *FLT-in* and *FLT-out*. The FLT-in method adapts the in-place addition approach by adding a copy step for the control qubit in Shor's circuit and by compressing conditional operations using precomputed values. The FLT-out method performs point addition in an out-of-place manner,

where the output is computed independently and the input is preserved. Allowing the use of additional qubits substantially reduces both the depth and the gate count.

We then construct the quantum circuit required for Shor's algorithm (Section 3.5) using these point addition techniques and analyze its performance when estimating the post-quantum security of binary ECC (Section 3.6). Our evaluation shows that binary elliptic curves do not satisfy the NIST Level-1 quantum security requirement (Section 3.6). We also compare the quantum attack cost of ECC with that of RSA (Section 3.6). Section 3.7 concludes the chapter.

3.3 Background

3.3.1 Binary Fields

A binary field is a finite field of characteristic 2 and consists of 2^n elements for an integer $n \geq 1$. We denote such a field by $\mathbb{F}_{2^n}$. A common way to construct a binary field is through binary polynomials. Specifically, let $\mathbb{F}_2[x]$ denote the set of all polynomials in the formal variable x with coefficients in $\mathbb{F}_2$. If $m(x) \in \mathbb{F}_2[x]/m(x)$ is an irreducible polynomial of degree n (called the *modulus*), then the quotient $\mathbb{F}_2[x]/m(x)$ forms a finite field with 2^n elements.

3.3.2 Binary Elliptic Curves

Binary elliptic curves are defined over binary fields $\mathbb{F}_{2^n}$. A typical ordinary binary elliptic curve of degree n (that is, defined over a field of size 2^n) is written as:

$$B_{a,b} : y^2 + xy = x^3 + ax^2 + b, \text{ where } a, b \in \mathbb{F}_{2^n} \text{ and } b \neq 0.$$

This form is often referred to as the short or simplified Weierstrass

form. For any choice of a and b with $b \neq 0$, the curve is well-defined.

The elliptic curve consists of all points (x, y) that satisfy the curve equation over $\mathbb{F}_{2^n}$, along with a special point known as the point at infinity (denoted by ∞). This extra point forms part of the elliptic curve group.

The point at infinity acts as the identity element of the group. In particular, for any point $P \in B_{a,b}$, we have $P + \infty = \infty + P = P$. For any point $P = (u, v)$ on $B_{a,b}$, its inverse $-P$ is $(u, u+v)$ and it satisfies $P + (-P) = \infty$.

Consider two distinct points $P_1 = (u_1, v_1)$ and $P_2 = (u_2, v_2)$ on $B_{a,b}$ such that $P_1 \neq \pm P_2$. Their sum $P_1 + P_2$ is the point $Q = (u, v)$, where

$$u = \delta^2 + \delta + a - u_1 - u_2, \quad v = \delta(u_1 + u) - u - v_1, \quad \text{and } \delta = \frac{v_2 + v_1}{u_2 + u_1}.$$

Points on a binary elliptic curve can be expressed either in affine coordinates or in projective coordinates. In the affine representation, a point is written using two values (x, y) and this coordinate system is the one we use throughout this work.

3.3.3 Shor's attack on Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is widely valued for its strong security, which comes from the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP) on a classical computer. For a binary elliptic curve E over $\mathbb{F}_{2^n}$ and two points P and Q on E, the goal of ECDLP is to find an integer m such that $Q = [m]P$, where $[m]P$ is the scalar multiplication of P by m. Classical algorithms require substantial computational effort to solve this problem.

Shor's algorithm fundamentally changes this landscape by providing an

efficient quantum method for solving the discrete logarithm problem, assuming a sufficiently large quantum computer is available. The algorithm makes use of quantum parallelism (different from classical parallelism) and a *Quantum Fourier Transform* (QFT), allowing an exponential speedup. The main steps are summarized below.

Initialization: Prepare three quantum registers. The first two registers, k and ℓ, each with $n+1$ qubits, start in the $|0\rangle$ state. The third register is reserved for point addition. Apply a Hadamard gate to every qubit in the first two registers, producing a uniform superposition,

$$|\psi\rangle = H^{\otimes n+1}\,|k,\ell\rangle^{\otimes n+1} = \frac{1}{2^{n+1}}\sum_{k,\ell=0}^{2^{n+1}-1}|k,\ell\rangle.$$

Conditional Addition: Depending on the values stored in the first two registers, add the corresponding multiples of P and Q to the third register, giving

$$\frac{1}{2^{n+1}}\sum_{k,\ell=0}^{2^{n+1}-1}|k,\ell\rangle|[k]P+[\ell]Q\rangle.$$

In Shor's circuit, only the first and second registers are measured at the end. The third register, which holds $|[k]P+[\ell]Q$, is not needed for the final outcome and is discarded (see Figure 3.1).

Quantum Fourier Transform (QFT): The QFT is then applied to the first two registers, each consisting of $n+1$ qubits. This transformation uses Hadamard gates together with phase shift operations, allowing the algorithm to extract the period r the underlying function. Once the period is obtained, the discrete logarithm m can be computed through classical post-processing, as described in [Sho94].

Figure 3.1 presents the quantum circuit used in Shor's algorithm, showing the setup of the quantum registers, the conditional addition of

elliptic-curve points, the application of the QFT, and the measurement step. In addition, Shor's quantum circuit may be carried out with just one control qubit by utilizing the semi-classical Fourier transform technique [GN96].

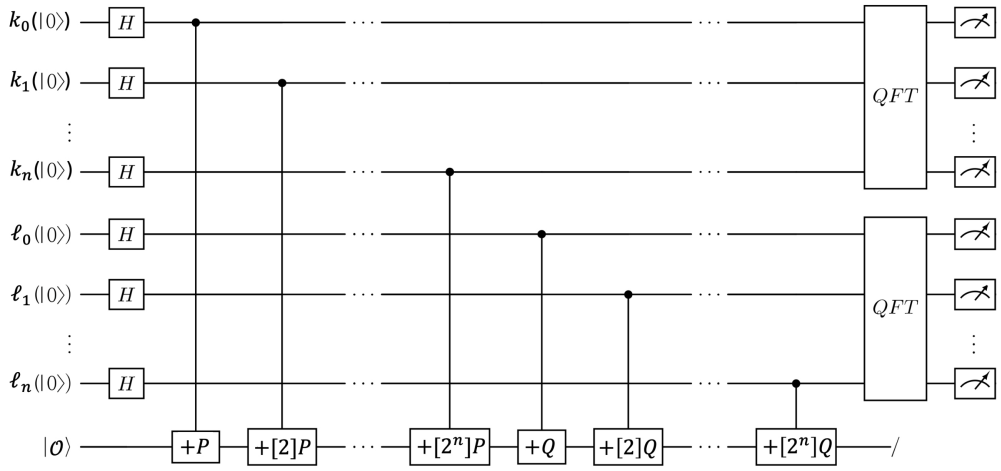


Figure 3.1: Shor's circuit for solving ECLDP.

3.4 Quantum Circuits for Binary Elliptic Curves

In this section, we describe our quantum circuit design for running Shor's algorithm on binary elliptic curves. We begin by presenting the quantum circuits used for arithmetic in binary fields. Using these building blocks, we then construct a depth-optimized quantum circuits for point addition, which is a key operation in Shor's algorithm for ECC.

3.4.1 Addition and Binary Shift

In the integer domain, both classical and quantum computing have long explored efficient adder designs, as shown in [CDKM08, DKRS04, Dra00, TTK09]. In contrast, addition in a binary field directly corresponds to an XOR operation and can therefore be implemented with CNOT gates in depth 1.

Shift and rotation can be realized using logical swaps, which rearrange qubit indices without using physical SWAP gates. Even when SWAP gates are used for convenience in circuit construction, they are commonly omitted from resource estimation. In our work, binary shift operations are implemented by simply reindexing qubits through logical swaps.

3.4.2 Squaring

Squaring in binary fields can be carried out using shift operations. As noted in Section 3.3, binary shifts can be implemented without quantum cost, and only the modular reduction requires CNOT gates. Prior studies [BBVHL20, PWLK22, TT23] mainly apply in-place squaring using PLU factorization, with only a few cases performed out-of-place.

In-place designs overwrite the input with the result, so no ancilla qubits are needed, but this usually increases circuit depth due to limited workspace. We instead adopt an out-of-place strategy in which the result is written to a fresh set of qubits. Since squaring is a linear operation, its output can be expressed as a binary non-singular matrix.

CNOT gates are applied according to the positions of 1s in the matrix. The input qubits act as control qubits, and the output qubits are used as targets. Constructing the matrix in this way removes redundant CNOT operations compared to schoolbook squaring. The total number of CNOT gates is the Hamming weight of the matrix.

3.4.2.1 Out-of-Place Implementation

A basic out-of-place implementation of an $n \times n$ binary matrix requires allocating n ancilla qubits initialized to $|0\rangle$. These ancilla qubits are first set to match the values of the original n input qubits, allowing the matrix to be preserved while writing the output to the new qubits.

Although out-of-place squaring typically achieves smaller depth than

the in-place method, it uses extra qubits. In our approach, however, we do not allocate new output qubits for every squaring step in division or point addition. Instead, output qubits are reused across multiple squaring operations, which is managed at the component-combination level (see Section 3.4.4).

Beyond this, we introduce an additional optimization aimed at compilers. By reordering the sequence of CNOT operations, we reduce quantum depth by more than 38% on average compared to the naive out-of-place design. A brief outline of this compiler-friendly method is given here, and full details are provided in [JSB+25a].

3.4.2.2 Compiler-Friendly Optimization

We analyze the structure of the out-of-place squaring matrices and design a deterministic algorithm that arranges CNOT operations in a compiler-friendly order. Quantum tools often fail to find an optimal schedule when the same qubits repeatedly participate in consecutive CNOT operations, even when parallelism exists. Our reordering avoids repeated access to the same input/output qubits in sequence, significantly lowering the overall depth.

3.4.3 Multiplication

Multiplication is a key operation in both inversion and point addition, so its efficiency directly affects overall circuit performance. The work in [BBVHL20], uses van Hoof's space-efficient multiplier [vH19], while [PWLK22] applies a modified form of this design. More recently, [TT23] adopts the multiplication method of Kim et al. [KKKH22], which minimizes Toffoli gate count while remaining space-efficient. A shared characteristic of these approaches is that they do not require ancilla qubits beyond the input and output registers. In other words, computing

$y = x \cdot y$ for polynomials of size n needs only $3n$ qubits. The quantum resource usage of the multipliers in m [vH19, PWLK22, TT23, JKL+23] is detailed in [JSB+25a].

In this thesis, we employ the depth-efficient Karatsuba multiplier proposed by [JKL+23]. This design lowers the Toffoli depth to one by introducing additional ancilla qubits. The Karatsuba method recursively breaks the multiplication into smaller subproblems. As discussed in [JKL+23], the operands for these reduced-size multiplications are copied so that the sub-multiplications can be carried out in parallel, improving both Toffoli depth and full depth (see [JKL+23, Figure 2] for details).

This multiplication technique is especially useful for inversion and point addition, which require several multiplications. Since the ancilla qubits introduced in one step can be reused in later steps, the approach fits well with our overall circuit structure.

3.4.4 Division based on Fermat's Little Theorem

In our quantum circuit design, we adopt an inversion algorithm based on Fermat's Little Theorem (FLT) to reduce circuit depth. Before presenting the circuit, we briefly recall how FLT is used to compute multiplicative inverses in the binary field by computing $a^{2^n-2} = a^{-1}$.

We construct a depth-efficient quantum circuit for Itoh-Tsujii inversion by combining the multiplication method from [JKL+23] with our out-of-place squaring approach.

For the case $n = 8$, based on the Itoh-Tsujii algorithm, the inverse of an element a can be written as:

$$a^{-1} = \left(\left((a^{2^{k_1}} - 1)^{2^{k_2}} \cdot a^{2^{k_2}-1} \right)^{2^{k_3}} \cdot a^{2^{k_3}-1} \right)^2 .$$

Using the second observation of the Itoh-Tsujii method, the

exponentiations $a^{2^{2k_1}}$, $a^{2^{2k_2}}$, and $a^{2^{2k_3}}$ can be represented as:

$$a^{2^{2k_3}-1} = a$$

$$A = a^{2^{2k_2}-1} = \left(a^{2^{2k_3}-1}\right)^{2^{2k_3}} \cdot a^{2^{2k_3}-1} = a^2 \cdot a$$

$$B = a^{2^{2k_1}-1} = \left(a^{2^{2k_2}-1}\right)^{2^{2k_2}} \cdot a^{2^{2k_2}-1} = A^{2^2} \cdot A$$

Figure 3.2 provides the quantum circuit for inversion when $n=8$. Here, $M(result)$ and $S(result)$ denote multiplication and squaring, and the label *result* refers to the output of each operation. The reverse operation $S^{\dagger}(result)$ resets the output qubits to $|0\rangle$.

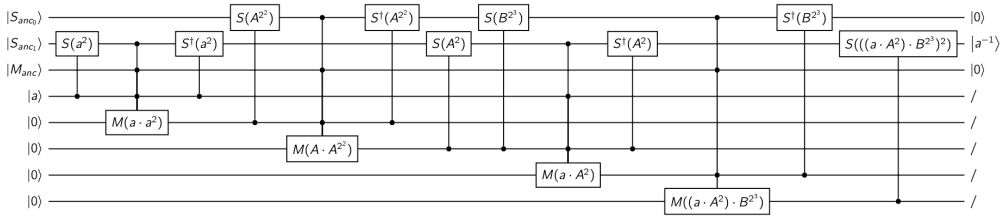


Figure 3.2: Proposed inversion quantum circuit for $n=8$

3.4.4.1 Multiplication in Inversion

As noted in Section 3.4.3, the multiplication method from [JKL+23] is effective in division and point addition. One reason is that the ancilla qubits used in the first multiplication can be reused in later multiplications, which avoids allocating new qubits each time.

For $n=8$, four multiplications are needed. In our implementation, only one ancilla set ($|M_{anc}\rangle$) is allocated for the first multiplication and then reused for the remaining multiplications. This achieves low-depth multiplication with minimal qubit overhead. Also, the same ancilla set is

also reused in the point addition circuit.

3.4.4.2 Squaring in Inversion

Our inversion circuit uses the out-of-place squaring method described in Section 3.3. Once a squaring result is used as input to a multiplication, it is no longer needed. We therefore reinitialize the output qubits and reuse them for subsequent squaring operations.

However, while reverse operations help reduce qubit usage, they also increase circuit depth. Consider Figure 3.2: if the output qubits reset by $S^{\dagger}(a^2)$ are needed again in $S(A^{2^2})$, the squaring of $S(A^{2^2})$ is delayed.

To avoid this delay, we prepare two output-qubit sets, $|S_{anc0}\rangle$ and $|S_{anc1}\rangle$, and use them alternately. In Figure 3.2, when $S^{\dagger}(a^2)$ resets $|S_{anc1}\rangle$, the squaring $S(A^{2^2})$ can proceed in parallel using $|S_{anc0}\rangle$. With this approach, only two output-qubit sets (total $2n$ qubits) are needed to construct low-depth squaring.

3.4.4.3 Performance Comparison of Division Algorithms

In Table 3.1, we compare the quantum resources required for performing division $h=h+f \cdot g^{-1}$ (one inversion, one multiplication, and one addition) with those reported in earlier works. Table 3.1 also includes the cost of reverse operations, which doubles the number of gates needed to reset ancilla qubits.

For [BBVHL20] and [PWLK22] in Table 3.1, we use the re-estimated costs from [TT23]. This is because [BBVHL20] only provided upper bounds for gate count and depth, and the updated estimates in [TT23] show improved performance. The results of [PWLK22] cannot be directly compared due to inconsistencies among their reported tables, but [TT23] gives corrected quantum resources for [PWLK22].

As shown in Table 3.1, previous works achieve lower Toffoli gate counts but require more CNOT gates than our method. This is because they rely on the Toffoli-optimized multiplier from [KKKH22]. Although our qubit count M is higher, we obtain the smallest circuit depth D and Toffoli depth. For the product of circuit depth and qubit count ($D \cdot M$), we observe improvements of about 31% for $n=8$ and between 72% and 78% for the remaining cases.

We cannot fully compare the full depth or the full depth and qubit product, since earlier works do not report their full depths. Even so, our method improves them as well, because our low Toffoli depth increases the full depth only by a small amount after decomposition, unlike in previous designs.

n	Method	#Toffoli	#CNOT	#Qubit (M)	Toffoli depth	Depth (D)	Full depth	D-M
8	[13]	243	2212	56	N/A	1314	N/A	73584
		3641	1516	67	N/A	4113	N/A	275571
	This work	270	2762	213	10	238	358	50694
16	[13]	1053	10814	144	N/A	5968	N/A	859392
		10403	5072	124	N/A	12145	N/A	1505980
	This work	1134	13510	777	14	458	654	182595
127	[13]	50255	502870	1778	N/A	203500	N/A	361823000
		277195	227902	903	N/A	378843	N/A	342095229
	This work	52440	681717	30971	24	2423	2645	75042733
163	[13]	18848	1601716	1956	N/A	342516	N/A	669961296
		438766	414586	1156	N/A	510628	N/A	590285968
	[114]	18848	1558180	3097	N/A	300924	N/A	931961628
	[129]	18848	1557528	2771	N/A	300920	N/A	833849320
		18848	1579944	1956	N/A	310830	N/A	607983480
	This work	87740	1176499	53133	20	2801	3046	148825533
233	[13]	30261	3374430	3029	N/A	459709	N/A	1392458561
		823095	834256	1646	N/A	992766	N/A	1634092836
	[114]	30261	3346938	4660	N/A	435001	N/A	2027104660
	[129]	30261	3345540	3961	N/A	434995	N/A	1723015195
		30261	3353750	3029	N/A	437747	N/A	1325935663
	This work	139106	2114587	82898	22	3476	3716	288153448
283	[13]	41032	5644678	3962	N/A	985710	N/A	3905383020
		1194498	1222600	1997	N/A	1449098	N/A	2893848706
	[114]	41032	5492126	6226	N/A	837106	N/A	5211821956
	[129]	41032	5489296	4811	N/A	837096	N/A	4027268856
		41032	5502090	3962	N/A	840612	N/A	3330504744
	This work	246552	3705491	144671	24	5412	5685	782959452
571	[13]	102951	26043772	9136	N/A	4401901	N/A	40215767536
		4434315	4857244	4014	N/A	5602181	N/A	22487154534
	[114]	102951	25189566	14275	N/A	3556815	N/A	50773534125
	[129]	95325	23458648	10849	N/A	3433263	N/A	37247470287
		95325	23514068	8565	N/A	3456469	N/A	29604656985
	This work	872788	14649243	500450	28	11723	12087	5866775350

Table 3.1: The required quantum resources for division.

3.4.5 Point Addition

In Shor's algorithm for ECC, the conditional point addition $|[k]P+[\ell]Q\rangle$ is carried out according to the control qubit in the first register (k and ℓ). If the control qubit $q=1$, the circuit computes $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2)$, whereas for $q=0$, the input point $P_1(x_1, y_1)$ is kept unchanged.

Earlier work [BBVHL20] introduced an in-place point-addition routine from [RNSL17] to binary fields, and follow-up papers [PWLK22, TT23] employed the same construction.

In our work, we redesign this in-place addition procedure of Algorithm 3.1 by modifying the scheme of [BBVHL20] and integrating our optimized circuits for addition, squaring, multiplication, and division. This results in a notable reduction in qubit usage while keeping the depth reasonably low. Figure 3.3(a) shows the original design, and Figure 3.3(b) highlights our updates. In Figure 3.3, the label D denotes the division subroutine (involving inversion and multiplication), and C refers to the copy operation for the control qubit q.

We also present an out-of-place variant (Algorithm 3.2; Figure 3.3(c)), which computes $P_3(x_3, y_3)$ on a separate output register while leaving $P_1(x_1, y_1)$ intact. This separation significantly decreases both the depth and the gate count.

3.4.5.1 In-Place Implementation

The in-place version of Algorithm 3.1 updates the coordinates of $P_1(x_1, y_1)$, and the final output depends on the control qubit q. Thus the point becomes either $P_3(x_3, y_3)$ or remains $P_1(x_1, y_1)$. Compared to [BBVHL20], our revised approach incorporates two main improvements:

1) Replicating the control qubit

Previous works [BBVHL20, PWLK22, TT23] used only a single control qubit for all controlled additions and controlled constant additions, so every CNOT and Toffoli gate had to be executed sequentially. This lowers the qubit cost but significantly increases depth. We instead copy the control qubit q into the ancilla set already used during multiplication (Algorithm 3.1; Steps 2, 8, 15), enabling depth-1 execution of controlled constant additions (CTRL_CONST_ADD) and controlled additions (CTRL_ADD). Because the ancilla set $|M_{anc}\rangle$ is already available, no extra qubits are needed. Additionally, we apply a tree-style copying strategy to further reduce depth and reset the copied qubits to $|0\rangle$ after use.

2) Optimizing the middle computation steps

In [BBVHL20], computing x_2+x_3 when $q=1$ (or x_1+x_2 when $q=0$) required one controlled constant addition for $q(a+x_2)$ and two controlled additions for $q \cdot \lambda^2$ and $q \cdot \lambda$ (Figure 3.3(a)). Our updated Algorithm 3.1 reorganizes these steps: Step 6 computes $\lambda^2+\lambda$ using a single squaring based on the matrices for λ^2 and λ (Section 3.3). Step 7 adds the constant $a+x_2$ to obtain $\lambda^2+\lambda+a+x_2$. Step 9 performs only one controlled addition using this precomputed value $\lambda^2+\lambda+a+x_2$. This reduces the number of controlled operations and improves the depth.

Algorithm 1 New In-place point addition quantum circuit.

Input: Constant a, fixed point $P_2(x_2, y_2)$, control qubit q, $P_1(x_1, y_1)$, M_{anc}, ancilla qubits for λ.
Output: $P_1 + P_2 = P_3(x_3, y_3)$ when $q = 1$, $P_1(x_1, y_1)$ when $q = 0$.

1:	$x \Leftarrow$ Constant Addition (x_2, x_1)	// $x = x_1 + x_2$
2:	$M_{anc} \Leftarrow$ Copy (q, M_{anc})	// Copy q to M_{anc}
3:	$y \Leftarrow$ Controlled Constant Addition (M_{anc}, y_2, y_1)	// $y = y_1 + q \cdot y_2$
4:	$\lambda \Leftarrow$ Division $(x_1, y_1, 0)$	// $\lambda = y/x$
5:	$y \Leftarrow$ Multiplication (x_1, λ, y_1)	// $y = y + x \cdot (y/x) = 0$
6:	$y \Leftarrow$ Squaring $(\lambda^2 + \lambda, y_1)$	// $y = \lambda^2 + \lambda$
7:	$y \Leftarrow$ Constant Addition $(a + x_2, y_1)$	// $y = \lambda^2 + \lambda + a + x_2$
8:	$M_{anc} \Leftarrow$ Copy (q, M_{anc})	// Copy q to M_{anc}
9:	$y \Leftarrow$ Controlled Addition (M_{anc}, y_1, x_1)	// $x = x_1 + x_2 + q(\lambda^2 + \lambda + a + x_2)$
10:	$y \Leftarrow$ Squaring $(\lambda^2 + \lambda, y_1)$	// $y = \lambda^2 + \lambda + a + x_2 + \lambda^2 + \lambda = a + x_2$
11:	$y \Leftarrow$ Constant Addition $(a + x_2, y_1)$	// $y = a + x_2 + a + x_2 = 0$
12:	$y \Leftarrow$ Multiplication (x_1, λ, y_1)	// $y = x \cdot \lambda$
13:	$\lambda \Leftarrow$ Division (x_1, y_1, λ)	// $\lambda = \lambda + (x \cdot \lambda)/x = 0$
14:	$x \Leftarrow$ Constant Addition (x_2, x_1)	// $x = x_1 + q(\lambda^2 + \lambda + a + x_2)$
15:	$M_{anc} \Leftarrow$ Copy (q, M_{anc})	// Copy q to M_{anc}
16:	$y \Leftarrow$ Controlled Constant Addition (y_2, y_1)	// $y = y + q \cdot y_2$
17:	$y_1 \Leftarrow$ Controlled Addition (x_1, y_1)	// $y = y + q \cdot x_3$
18:	**return** (x, y)	

Algorithm 3.1: New in-place point addition quantum circuit.

3.4.5.2 Out-of-Place Implementation

The out-of-place point addition in Algorithm 3.2 keeps the input point $P_1(x_1, y_1)$ intact while producing $P_3(x_3, y_3)$ regardless of the value of the control qubit q. Recall that the in-place method (Algorithm 3.1) must apply reverse operations to restore intermediate values, which is necessary when conditionally computing either $P_1(x_1, y_1)$ or $P_3(x_3, y_3)$. In contrast, the out-of-place method avoids these additional reversals by allocating separate output qubits and performing computations directly on them (see Figure 3.3(c)). Consequently, Algorithm 3.1(in-place) involves two divisions, two squarings, and two multiplications, whereas Algorithm 3.2 (out-of-place) requires only one division, one squaring, and one multiplication.

Moreover, the out-of-place approach reduces the number of controlled operations that depend on the control qubit q. In Algorithm 3.1, two controlled constant additions and two controlled additions are required (three controlled constant additions and three controlled additions in [BBVHL20, PWLK22, TT23]). In comparison, Algorithm 3.2 eliminates several of these operations.

In Algorithm 3.2, the outputs $P_1(x_1, y_1)$ and $P_3(x_3, y_3)$ are swapped

depending on the value of the control qubit q in the final steps (Steps 13 and 14). A controlled-swap is executed twice: once for $(x_1, y_1)$and once for (x_3, y_3). These two swap operations run in parallel, and similar to the in-place version, the same copy mechanism is applied.

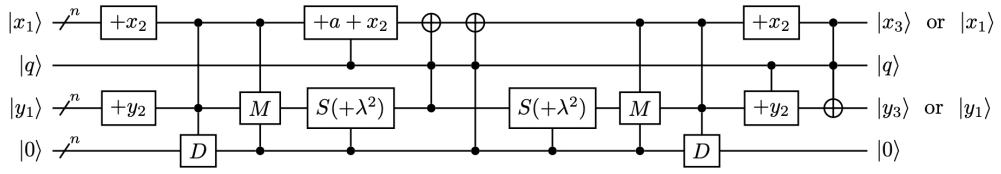
The corresponding quantum circuit for the out-of-place point addition is shown in Figure 3.3(c).

Algorithm 2 New out-of-place point addition quantum circuit.

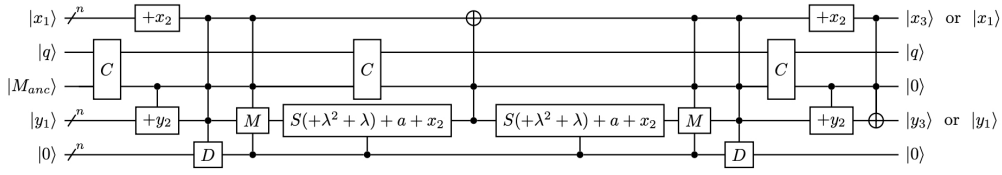
Input: Constantant a, fixed point $P_2(x_2, y_2)$, control qubit q, point $P_1(x_1, y_1)$, M_{anc}, qubits for (x, y), ancilla qubits for λ.
Output: $P_1 + P_2 = P_3(x_3, y_3)$ when $q = 1$, $P_1(x_1, y_1)$ when $q = 0$.

1:	$x \Leftarrow$ CNOT (x_1, x)	// $x = x_1$
2:	$y_1 \Leftarrow$ Constant Addition (y_2, y_1)	// $y_1 = y_1 + y_2$
3:	$x \Leftarrow$ Constant Addition (x_2, x)	// $x = x_1 + x_2$
4:	$\lambda \Leftarrow$ Division $(x, y_1, 0)$	// $\lambda = (y_1 + y_2)/(x_1 + x_2)$
5:	$y_1 \Leftarrow$ Constant Addition (y_2, y_1)	// $y_1 = y_1 + y_2 + y_2 = y_1$
6:	$x \Leftarrow$ Constant Addition $(a + x_2, x)$	// $x = x_1 + x_2 + a + x_2 = x_1 + a$
7:	$x \Leftarrow$ Squaring $(\lambda^2 + \lambda, x)$	// $x = x_1 + a + \lambda^2 + \lambda = x_2 + x_3$
8:	$y \Leftarrow$ Multiplication $(x, \lambda, 0)$	// $y = (x_2 + x_3)\lambda$
9:	$x \Leftarrow$ Constant Addition (x_2, x)	// $x = x_1 + a + \lambda^2 + \lambda + x_2 = x_3$
10:	$y \Leftarrow$ Constant Addition (y_2, y)	// $y = (x_2 + x_3)\lambda + y_2$
11:	$y \Leftarrow$ CNOT (x, y)	// $y = (x_2 + x_3)\lambda + y_2 + x_3 = y_3$
12:	$M_{anc} \Leftarrow$ Copy (q, M_{anc})	// Copy q to M_{anc}
13:	Controlled Swap (M_{anc}, x_1, x)	// $x = x_3$ (when $q = 1$) or x_1 (when $q = 0$)
14:	Controlled Swap (M_{anc}, y_1, y)	// $y = y_3$ (when $q = 1$) or y_1 (when $q = 0$)
15:	**return** (x, y)	

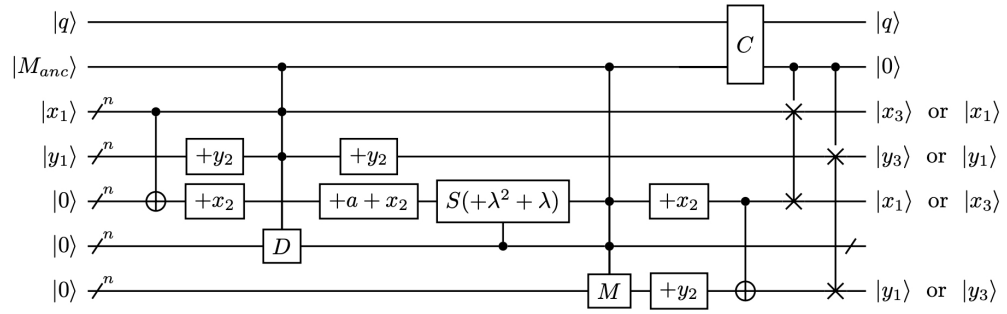
Algorithm 3.2: New out-of-place point addition quantum circuit.



(a) In-place point addition of [BBVHL20, Algorithm 3].



(b) In-place point addition in this work (ours, Algorithm 3.1).



(c) Out-of-place point addition in this work (ours, Algorithm 3.2).

Figure 3.3: Point addition quantum circuits.

3.4.6 Windowing

The windowing technique provides an efficient way to reduce the cost of conditional point additions by incorporating a superposition of a fixed point P_2. The idea is to employ quantum random-access memory (qRAM), placing qubits in a superposition over the indices $i = 0, 1, 2 \cdots, 2^{\ell}$. Using this, the addition of points is reframed as adding $[i]P_2$. A lookup table is prepared containing the precomputed points $T, T+[1]P_2, T+[2]P_2, \cdots, T+[2^{\ell}-1]P_2$, where T is a fixed point used to avoid encountering the point at infinity.

With this table, the value $[i]P_2$ can be retrieved via a superposed lookup, which greatly cuts down the number of explicit point additions. The effectiveness of this technique depends on the window size ℓ, which determines how many points are precomputed. A larger ℓ reduces the number of additions required but increases the cost of constructing and storing the qRAM lookup table.

Prior works [BBVHL20, PWLK22, TT23] estimate the Toffoli-gate savings achieved through windowing for each field size nnn, assuming that the optimal window size ℓ is selected. Under this technique, additions involving the precomputed point $P_2(x_2, y_2)$ in Figure 3.3(a) of

[BBVHL20] are replaced with lookup-based additions, and controlled additions are replaced with regular ones. The same modification extends to Algorithm 3.1. Likewise, in Algorithm 3.2 (out-of-place), additions involving $P_2(x_2, y_2)$ become lookup additions, and the controlled-swap operations in the final step are removed.

Following these prior approaches [BBVHL20, PWLK22, TT23], we also evaluate the effect of windowing. Similar to earlier findings, our results show a decrease in Toffoli-gate complexity due to windowing. We additionally report the overall quantum gate reductions that result after decomposing the Toffoli gates.

3.5 Results

Table 3.2 summarizes the quantum resources needed for a single point addition over binary elliptic curves. As in Table 3.1, the values for [BBVHL20, PWLK22] in Table 3.2 are replaced with the updated estimates provided in [TT23].

Since the publication of [BBVHL20], two follow-up studies [PWLK22, TT23] have appeared. However, the improvements claimed in those works (particularly over the results of [BBVHL20]) are relatively modest. Indeed, Table 3.2 shows that the method in [BBVHL20] still provides the best trade-off between depth and qubit count for $n = 163, 283$ and $n = 571$ when compared with [PWLK22, TT23].

Our point-addition constructions outperform all previous designs [BBVHL20, PWLK22, TT23]. By introducing additional ancilla qubits, we achieve substantially lower Toffoli depths and overall circuit depths. Although our Toffoli count is higher than the re-estimated value from [TT23], this is due to their use of a Toffoli-optimized multiplication technique from [KKKH22], which reduces Toffoli gates at the cost of requiring more CNOTs. For the $D-M$ metric, our methods yield

improvements of roughly 73%−81% for the in-place version and more than 92% for the out-of-place version across all binary fields.

n	Method		#Toffoli	#CNOT	#Qubit (M)	Toffoli depth	Depth (D)	Full depth	D-M
8	[13]	(in-place)	7360	3522	68	N/A	8562	N/A	582216
	This work	(in-place)	664	6580	214	26	517	831	110638
		(out-of-place)	178	1761	241	7	159	236	38319
16	[13]	(in-place)	21016	11686	125	N/A	25205	N/A	3150625
	This work	(in-place)	2624	30560	778	34	959	1412	746102
		(out-of-place)	680	7953	859	9	283	402	243097
127	[13]	(in-place)	559141	497957	904	N/A	776234	N/A	701715536
	This work	(in-place)	113874	1464162	30972	54	4994	5507	154674168
		(out-of-place)	28659	370120	33157	14	1267	1394	42009919
163	[13]	(in-place)	40169	3357029	1957	N/A	706512	N/A	1382643984
			880005	982769	1157	N/A	1042736	N/A	1206445552
	[114]	(in-place)	40169	3269957	3098	N/A	623328	N/A	1931070144
	[129]	(in-place)	40169	3268653	2772	N/A	623320	N/A	1727843040
			40169	3313485	1957	N/A	643140	N/A	1258624980
	This work	(in-place)	193354	2540458	53134	46	5685	6227	302066790
		(out-of-place)	48583	650277	57521	12	1495	1631	85993895
233	[13]	(in-place)	64103	7059764	3030	N/A	953699	N/A	2889707970
			1649771	1979416	1647	N/A	2019813	N/A	3326632011
	[114]	(in-place)	64103	7004780	4661	N/A	904283	N/A	4214863063
	[129]	(in-place)	64103	7001984	3962	N/A	904271	N/A	3582721702
			64103	7018404	3030	N/A	909775	N/A	2756618250
	This work	(in-place)	303970	4516616	82899	50	7059	7589	585184041
		(out-of-place)	76342	1158949	89222	13	1800	1942	160599600
283	[13]	(in-place)	86481	11739723	3963	N/A	2017360	N/A	7994797680
			2393413	2895567	1998	N/A	2944136	N/A	5882383728
	[114]	(in-place)	86481	11434619	6227	N/A	1720152	N/A	10711386504
	[129]	(in-place)	86481	11428959	4812	N/A	1720132	N/A	8277275184
			86481	11454547	3963	N/A	1727164	N/A	6844750932
	This work	(in-place)	534762	7856986	144672	54	10957	11556	1585171104
		(out-of-place)	134115	2007399	154945	14	2902	3025	449650390
571	[13]	(in-place)	215241	53816483	9137	N/A	8931056	N/A	81603058672
			8877969	11443427	4015	N/A	11331616	N/A	45496438240
	[114]	(in-place)	215241	52108071	14276	N/A	7240884	N/A	103370859984
	[129]	(in-place)	199989	48646235	10850	N/A	6993780	N/A	75882513000
			199989	48757075	8566	N/A	7040192	N/A	60306284672
	This paper	(in-place)	1871402	30657812	500450	62	23596	24399	11808618200
		(out-of-place)	468707	7833731	531621	16	6313	6449	3356123373

Table 3.2: Required quantum resources for point addition.

3.5.1 Relation to Shor's Algorithm

Performing Shor's algorithm for ECDLP requires $2n+2$ point additions. Table 3.3 summarizes the quantum resources needed for this task. To compute the total gate complexity, we decompose Toffoli gates using the scheme from [AMM+13], where each Toffoli becomes 8 Clifford gates and 7 T gates, resulting in a T-depth of 4 and a full depth of 8.

The windowing shows that the number of point additions can be reduced by applying windowing. Each point addition requires six lookups. Based on this, we determine the optimal window size for each field. Table 3.4 provides the resulting Toffoli counts and total gate counts after applying windowing.

Point addition	Size n	Qubits	Gates (G)	Full depth (FD)	T-depth	Cost (G-FD)	NIST MAXDEPTH	Security
In-place	8	$1.67 \cdot 2^{7}$	$1.14 \cdot 2^{18}$	$1.83 \cdot 2^{13}$	$1.83 \cdot 2^{10}$	$1.04 \cdot 2^{32}$	Satisfied	Not satisfied
	16	$1.52 \cdot 2^{9}$	$1.13 \cdot 2^{21}$	$1.47 \cdot 2^{15}$	$1.13 \cdot 2^{12}$	$1.66 \cdot 2^{36}$		
	127	$1.89 \cdot 2^{14}$	$1.51 \cdot 2^{29}$	$1.34 \cdot 2^{20}$	$1.69 \cdot 2^{15}$	$1.02 \cdot 2^{50}$		
	163	$1.62 \cdot 2^{15}$	$1.66 \cdot 2^{30}$	$1.95 \cdot 2^{20}$	$1.84 \cdot 2^{15}$	$1.62 \cdot 2^{51}$		
	233	$1.26 \cdot 2^{16}$	$1.98 \cdot 2^{31}$	$1.69 \cdot 2^{21}$	$1.43 \cdot 2^{16}$	$1.67 \cdot 2^{53}$		
	283	$1.10 \cdot 2^{17}$	$1.05 \cdot 2^{33}$	$1.56 \cdot 2^{22}$	$1.87 \cdot 2^{16}$	$1.64 \cdot 2^{55}$		
	571	$1.91 \cdot 2^{18}$	$1.99 \cdot 2^{35}$	$1.70 \cdot 2^{24}$	$1.06 \cdot 2^{18}$	$1.70 \cdot 2^{60}$		
Out-of-place	8	$1.60 \cdot 2^{11}$	$1.22 \cdot 2^{16}$	$1.04 \cdot 2^{12}$	$1.97 \cdot 2^{8}$	$1.27 \cdot 2^{28}$		
	16	$1.42 \cdot 2^{14}$	$1.18 \cdot 2^{19}$	$1.67 \cdot 2^{13}$	$1.20 \cdot 2^{10}$	$1.97 \cdot 2^{32}$		
	127	$1.75 \cdot 2^{22}$	$1.53 \cdot 2^{27}$	$1.36 \cdot 2^{18}$	$1.75 \cdot 2^{13}$	$1.04 \cdot 2^{46}$		
	163	$1.90 \cdot 2^{23}$	$1.69 \cdot 2^{28}$	$1.02 \cdot 2^{19}$	$1.92 \cdot 2^{13}$	$1.72 \cdot 2^{47}$		
	233	$1.06 \cdot 2^{25}$	$1.00 \cdot 2^{30}$	$1.73 \cdot 2^{19}$	$1.49 \cdot 2^{14}$	$1.74 \cdot 2^{49}$		
	283	$1.14 \cdot 2^{26}$	$1.06 \cdot 2^{31}$	$1.64 \cdot 2^{20}$	$1.94 \cdot 2^{14}$	$1.74 \cdot 2^{51}$		
	571	$1.00 \cdot 2^{29}$	$1.98 \cdot 2^{33}$	$1.76 \cdot 2^{22}$	$1.12 \cdot 2^{16}$	$1.74 \cdot 2^{56}$		

Table 3.3: Required quantum resources for Shor's attack on ECC and the corresponding security analysis.

Point addition	n	Size (ℓ)	Steps	Look-ups	Toffoli gates	Gates (G)
In-place	8	5	4	24	$1.01 \cdot 2^{12}$	$1.35 \cdot 2^{16}$
	16	6	6	36	$1.24 \cdot 2^{14}$	$1.86 \cdot 2^{18}$
	127	10	26	156	$1.56 \cdot 2^{21}$	$1.30 \cdot 2^{26}$
	163	11	30	180	$1.56 \cdot 2^{22}$	$1.30 \cdot 2^{27}$
	233	12	40	240	$1.68 \cdot 2^{23}$	$1.46 \cdot 2^{28}$
	283	13	44	264	$1.66 \cdot 2^{24}$	$1.42 \cdot 2^{29}$
	571	14	82	492	$1.26 \cdot 2^{27}$	$1.20 \cdot 2^{32}$
Out-of-place	8	3	6	36	$1.54 \cdot 2^{10}$	$1.04 \cdot 2^{15}$
	16	5	8	48	$1.03 \cdot 2^{13}$	$1.45 \cdot 2^{17}$
	127	8	32	192	$1.94 \cdot 2^{19}$	$1.61 \cdot 2^{24}$
	163	10	34	204	$1.97 \cdot 2^{20}$	$1.58 \cdot 2^{25}$
	233	10	48	288	$1.01 \cdot 2^{22}$	$1.78 \cdot 2^{26}$
	283	11	52	312	$1.97 \cdot 2^{22}$	$1.70 \cdot 2^{27}$
	571	12	96	576	$1.48 \cdot 2^{25}$	$1.39 \cdot 2^{30}$

Table 3.4: Required quantum resources for Shor's attack on ECC with windowing technique.

3.5.2 Application to Shor's Algorithm

For in-place point addition, a key advantage is that the number of qubits does not increase throughout Shor's algorithm. Thus, the qubit count remains the same as in Table 3.2.

In contrast, the out-of-place design produces new output qubits for every point addition, increasing the total number of qubits used during Shor's algorithm. The accumulation of garbage qubits is a clear drawback when scaling quantum computations. However, the out-of-place method still offers far smaller depth and gate complexity compared to the in-place method.

Given these considerations, one must choose between in-place and out-of-place implementations depending on hardware constraints. Our results provide efficient options for either setting.

3.5.3. NIST Post-Quantum Security

NIST's post-quantum security guidelines include limits on quantum attack capabilities, particularly through the MAXDEPTH constraint 2^{21}, which caps the allowable quantum circuit depth (essentially the runtime).

The lower bound of MAXDEPTH is 2^{40} (with an upper limit around 2^{96}), and none of the full depths listed in Table 3.3 exceed this threshold. NIST also uses additional criteria to evaluate the robustness of cryptographic algorithms against large-scale quantum attacks.

For post-quantum security level 1, the benchmark attack cost corresponds to Grover's algorithm applied to AES-128. The metric used is the product of total gates and full depth ($G-FD$ in Table 3.3). The cost of Grover's search on AES-128 is 2^{156} based on the latest results in [JBK+25]. As expected, our costs in Table 3.3 are many orders of magnitude below this threshold, and therefore do not meet NIST's level-1 security requirements.

3.6 Conclusion

It is widely expected that public key systems based on binary elliptic curves will become vulnerable once sufficiently large quantum computers are available. While several studies have explored quantum attacks on binary ECC, progress since Banegas et al. [BBVHL20] has been relatively limited.

In this work, we substantially reduce the quantum resources needed to break binary ECC. Using FLT-based division together with depth-optimized point addition circuits in both in-place and out-of-place styles, we achieve the lowest depths reported so far. Our designs improve the trade-off metric (depth × qubits) by more than 73%−81% for the in-place version and over 92% for the out-of-place version across all binary fields, as summarized in Table 3.3. To the best of our knowledge, these represent the most efficient point-addition circuits available for quantum cryptanalysis of binary ECC.

As in prior works ([BBVHL20, PWLK22, TT23], our resource estimates are based on logical-level simulations without considering

hardware-level effects such as noise, decoherence, or error correction. These practical issues will inevitably increase resource requirements and remain an important direction for future research. Nonetheless, like earlier foundational studies, our work provides a clear and systematic estimate of the required quantum resources from a theoretical perspective.

Ⅳ. Quantum Attacks on AES

Author's Contribution. This chapter builds upon the work presented in [JBK+25]. I was chiefly responsible for optimizing the quantum circuit implementations used in the quantum security analysis of AES.

4.1 Introduction

Although fully scalable quantum computers are not yet available, the need to assess the resilience of symmetric key ciphers against powerful quantum attackers continues to grow. Motivated by this, our study investigates key-recovery attacks on the three AES variants (128, 192, and 256 bits) using Grover's search.

4.1.1 Contribution

We implement several quantum realizations of the AES family (AES-128, AES-192, and AES-256) and present the smallest Toffoli depth TD, full depth FD, and the best cost-depth trade-offs reported to date. These trade-offs include $FD-M$, and $FD-G$.

By slightly increasing the qubit count, we obtain a substantial reduction in full depth, which in turn lowers the overall circuit cost. This reduced depth is particularly beneficial when parallelization is needed due to depth constraints in Grover's algorithm.

Our implementations achieve the most favorable TD^2-M and FD^2-M trade-offs so far, which are crucial measures when evaluating parallel quantum search.

Our optimization strategy operates across three levels: component level (e.g., S-box, MixColumn), architecture level (assembling 16 S-boxes for SubBytes, four MixColumns for MixColumns, etc.), and resource sharing

between modules.

4.1.2 Preliminary: Grover's Key Search

For a block cipher with a k-bit secret key, a classical exhaustive search requires evaluating all 2^k possible keys. Grover's algorithm [Gro96] provides a quadratic speed-up, recovering the correct key with high probability after approximately $\left\lfloor \frac{\pi}{4}\sqrt{2^k} \right\rfloor$ iterations. Its procedure can be summarized as follows.

First, the key register is initialized into an equal superposition over all 2^k candidates by applying Hadamard gates.

The block cipher Enc is then embedded into Grover's oracle $U_{f(x)}$. Inside the oracle, a fixed plaintext p is encrypted under the superposed key register, and the resulting ciphertext is compared to the known ciphertext c through a predicate $f(x)$. The function outputs 1 when $Enc_k(p) = c$ and 0 otherwise. Whenever the comparison succeeds, the oracle marks the corresponding key by applying a phase flip (a Z gate).

The diffusion operator then amplifies the amplitude of the marked state. It consists of the sequence: Hadamard layer → X layer → multi-controlled Z on the k-qubit key register → X layer → Hadamard layer. As noted in [Per19], the X-gate layers can be reduced to a constant number of gates by applying a fixed pattern of X operations before the initial Hadamard transform, allowing the diffusion step to be expressed simply as (Hadarmard layer → multi-controlled Z → Hadamard layer).

Grover iterations repeat the oracle and diffusion operator until the marked state dominates the amplitude distribution. For a k-bit key, the optimal number of iterations is approximately $\left\lfloor \frac{\pi}{4}\sqrt{2^k} \right\rfloor$ [BBHT98],

giving a runtime of about $\sqrt{2^k}$ (i.e., quadratic reduction relative to classical brute force).

To guarantee retrieval of the unique, non-spurious key, one must use $r = \lceil k/n \rceil$ plaintext-ciphertext pairs in the exhaustive search, where n is the block size.

4.2 Quantum Circuits for AES

4.2.1. Regular, Shallow, and Shallow/Low Architectures

We categorize our AES quantum circuit implementations into three versions: regular, shallow, and shallow/low-depth. The regular version emphasizes parallel execution within each round while balancing depth and qubit usage.

All three versions apply reverse operations (i.e., uncomputation) to release ancilla qubits. While allocating fresh qubits for each step could reduce both gate count and depth, our design goal is to construct architectures that maintain a good depth-qubit balance using reverse operations rather than increasing the qubit count unnecessarily.

4.2.1.1 Regular Version

This version (originally introduced in [JNRV20] but refined in our work) uses the smallest number of qubits among our implementations and achieves a reduced Toffoli depth across all AES variants. The regular version focuses on maximizing parallelism inside each round. To reduce depth, additional ancilla sets are used (except for the in-place MixColumns), and these ancilla qubits are cleaned and reused through reverse operations, as shown in Figure 4.1(b). In this structure, the round execution and the reverse operation occur sequentially: the next round begins only after the reverse operation of the current round is finished. Parallel execution is achieved within each round (e.g., SubBytes, key

schedule, and MixColumns), but not across rounds.

4.2.1.2 Shallow Version

The shallow version achieves parallel execution across all rounds. This is done through a pipelined structure (Figure 4.3(b)), where the reverse operation of the previous round runs at the same time as the computation of the current round. Even and odd rounds alternate between compute and uncompute phases. This architecture requires more qubits, but significantly reduces both the Toffoli depth and full depth, since all parallelizable components of each round run together. As a result, the depth is determined mainly by the SubBytes + MixColumns cost in each round (excluding the final round where MixColumns is omitted).

4.2.1.3 Shallow/Low Version

In this pipeline, the full depth can be further reduced by choosing a different MixColumn implementation. The Toffoli depth remains the same, but selecting our quantum-friendly MixColumn lowers the full depth at the cost of additional qubits. This yields the shallow/low-depth version, which inherits all features of the shallow version except for the updated MixColumn. Notably, this extra qubit usage does not increase the total ancilla requirement, because the MixColumn operation utilizes ancilla qubits that become idle right after the SubBytes step.

4.2.2 Quantum Implementation of S-box (SubByte)

The S-box plays a major role in the AES round function and in the key schedule (specifically inside SubWord), and it usually consumes the largest portion of the quantum resources. We benchmark several existing S-box implementations and summarize the quantum resources required for each design in Table 4.1.

Earlier works such as [LPS20, ZWS+20] relied on the implementations from [BP10, BP12]. When the Boyar–Peralta S-boxes from these papers (originally designed for efficient hardware circuits) are translated to quantum circuits in a straightforward way, the version in [BP12] needs more ancilla qubits (120) than the quantumized version of [BP10] (107 ancilla qubits), but it gives a smaller depth. JNRV [JNRV20] directly used the S-box of [BP12] in their quantum design. The implementation from [BP10] was adapted and improved for quantum usage by Zuleger et al. in [ZLD+19]. In [LPS20, ZWS+20], Langenberg et al. took the first Boyar–Peralta design from [BP10] and constructed a quantum S-box with fewer qubits.

More recently, [JBKK24] discovered new AES S-box circuits using a heuristic search framework built upon the circuits from [BP10, BP12]. We port [JBKK24, Listing 17] to the quantum setting and evaluate its quantum cost in Table 4.1; this circuit is among the best in terms of depth in that work.

Huang and Sun introduced an improved quantum S-box for the design in [JNRV20] in their Asiacrypt 2022 paper [HS22]. Their work provides two versions: one reduces the Toffoli depth without extra qubits, while the other uses additional qubits to reduce the depth further. Another update appeared in the Asiacrypt 2023 paper [LPZW23], which applied a new method to the S-box from [HS22].

The S-box in [GLRS16] is based on field inversion using the Itoh–Tsujii algorithm.

Dansarie [Dan17, Dan21] also proposed a general method for classical S-box design that can handle any 8-bit S-box, unlike the AES-specific constructions in [BP10, BP12]. Using Dansarie's public source code, we generated five implementations (each roughly 400 lines of C code containing AND/OR/NOT operations). These were not used in our work

because of their high quantum cost (benchmarks are shown in Table 4.1).

LIGHTER-R [DBSC19] was used by the authors of [LGQW23] to produce a low-qubit S-box. Later, [LXX+23] introduced another design also targeting small qubit count. Because [LGQW23, LXX+23] do not provide detailed Clifford+T quantum benchmarks, we only list Toffoli depth and qubit count in Table 4.1.

Two additional S-box designs appeared in [LPZW23]. Combined with their 16-depth MixColumn, these allowed [LPZW23] to reduce overall circuit cost compared to an earlier version of this work, while using the same shallow architecture as ours.

To further improve performance, we propose new S-box implementations starting from the designs in [LPZW23]. Our versions achieve smaller full depth (82 → 67 and 69 → 58), reduced CNOT count (400 → 394 and 372 → 366), and smaller qubit count (90 → 84), as shown in Table 4.1. The technical reasoning behind these improvements is discussed in Appendix D.

We also identified two reversible AES S-boxes from [LYLL22, Appendices C and D], but those were provided only as raster images and were too hard to decode for reliable quantum conversion.

Finally, although the AES circuits in [ZWS+20] and [SF24b] used the inverse S-box (from [BP12] and [LPZW23]) to reduce qubit count in their architectures, we do not use the inverse S-box in the regular, shallow, or shallow/low-depth versions.

Method		#CNOT	#1qCliff	$\#T$	TD	M	Full depth
S-box [GLRS16]		1818	124	1792	88	40	951
S-box [BP10]		358	68	224	8	123	104
S-box [BP12]		392	72	238	6	136	85
S-box [LPS20]		628	98	367	40	32	514
S-box [ZWS+20]		437	72	245	55	22	339
S-box [Dan17, Dan21]	(391 lines)	1470	670	1218	66	399	640
	(406 lines)	1507	548	1245	74	414	709
	(413 lines)	1484	561	1169	62	421	591
	(409 lines)	1483	574	1190	74	416	693
	(400 lines)	2244	1006	2254	111	408	998
S-box [JBKK24]		472	72	238	4	209	69
S-box [HS22]		418	72	238	4	136	72
		824	160	546	3	198	69
S-box [LGQW23]		·	·	·	32	20	·
S-box [LXX+23]		·	·	·	24	21	·
		·	·	·	22	22	·
S-box [LPZW23]		400	72	238	4	76	82
		372	72	238	4	90	69
S-box (ours)	(low qubit count)	394	72	238	4	76	67
	(low full depth)	366	72	238	4	84	58
	(Toffoli depth 3)	781	160	546	3	152	56

Table 4.1: The required quantum resources for AES S-box.

4.2.3 Quantum Implementation of SubBytes

For the SubBytes implementation, we consider two different approaches depending on how ancilla qubits are handled.

The first option allocates new ancilla qubits for every SubBytes operation, which makes the design highly sensitive to the qubit count.

The second option reuses the ancilla qubits that were allocated at the beginning. In this case, no additional ancilla qubits are required for later SubBytes operations. The trade-off is that reverse operations must be performed to reset these qubits, increasing both the depth and the number of gates.

In this work, we adopt the second approach. Our reasoning is that keeping the qubit count low is more beneficial than avoiding the extra depth caused by the reverse operations. Since the ancilla qubits are reused, the initial allocation is the only stage where the qubit cost is

incurred. After choosing the S-box design, this implementation is used to construct the 16 S-boxes required for SubBytes.

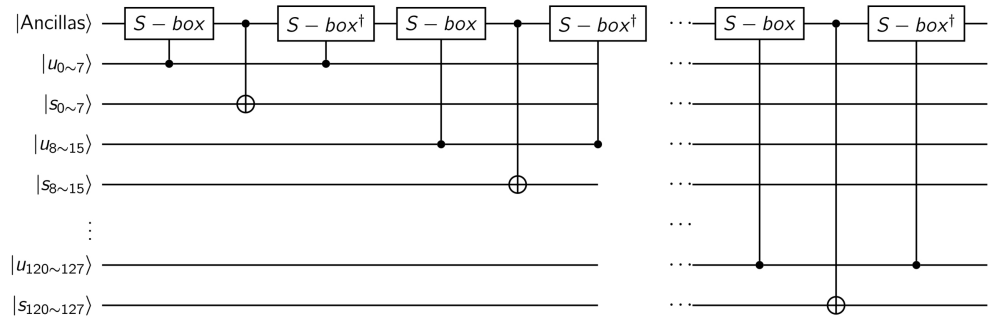
Figure 4.1 (a) illustrates the variant that uses the minimum number of qubits; however, it runs all S-boxes sequentially, which leads to a large depth. To reduce the depth, we instead allocate additional ancilla sets at the start.

Each AES round uses 16 S-boxes for SubBytes and 4 S-boxes in the key schedule, so 20 S-boxes are executed in parallel. Accordingly, we allocate:

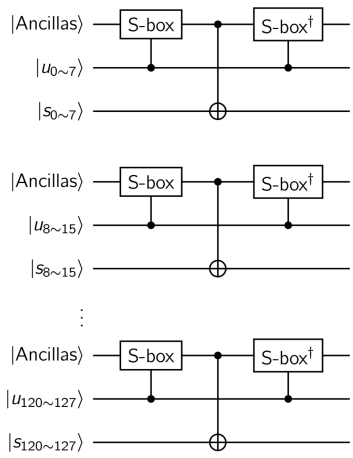
- 20 × 60 ancilla qubits for Toffoli-depth-4 low-qubit S-boxes,
- 20 × 68 ancilla qubits for Toffoli-depth-4 low-full-depth S-boxes,
- 20 × 136 ancilla qubits for Toffoli-depth-3 S-boxes.

Figure 4.1 (b) shows 16 S-boxes running in parallel with multiple ancilla sets. After these S-boxes complete, the ancilla qubits are no longer clean. They are restored to $|0\rangle$ via 16 S-box† operations, executed in parallel before the next round. This allows the same ancilla qubits to be reused each round. While reverse operations increase depth, allocating new ancilla qubits every time would be highly inefficient.

We evaluate these trade-offs carefully, and the shallow version later compensates for the depth overhead caused by the reverse operations.



(a) Sequential flow using one ancilla set.



(b) Parallel flow using multiple ancilla sets.

Figure 4.1: Quantum implementation of SubBytes.

4.2.4 Quantum Implementation of Key Schedule

In the AES key schedule, the SubWord operation is applied to a rearranged 32-qubit state. From the pool of ancilla qubits determined earlier (20 × 60, 20 × 68, or 20 × 136; see Section 4.2.3), 4 × (60, 68, or 136) ancilla qubits are assigned to execute the S-boxes for these 32 qubits in parallel, while the remaining 16 × (60, 68, or 136) ancilla qubits are used for SubBytes within the round.

Since SubWord only requires a permutation of the 32 qubits, this step uses logical swaps, meaning no quantum gates are consumed for rearranging the indices. Unlike SubBytes, where S-box outputs are written to fresh qubits, the key schedule does not allocate new qubits. The S-box outputs are combined directly into the state using CNOT operations. Because the SubWord operation proceeds in parallel with the SubBytes of the same round, no additional depth is introduced. This follows the same design strategy as in [JNRV20]. The 8-bit round constant is XORed into the key by applying X gates to the qubits $k_{0,1,\cdots,7}$ at positions where the round constan bit is 1. Afterward, the

internal CNOT operations of the key schedule are applied.

We also follow the common on-the-fly strategy, generating each round key immediately before use, unlike the classical approach where all round keys are precomputed. In our design, the key schedule is executed alongside SubBytes in the round function. Since the key schedule is executed in complete parallel with SubBytes, the overall depth behaves as if the key schedule were not present.

4.2.5 Quantum Implementation of AddRoundKey and ShiftRows.

The AddRoundKey step, which XORs the 128-qubit round key with the state, is straightforward and it is realized using 128 CNOT gates. For ShiftRows, one can in principle rely on swap gates, but in our implementation, the row rotation is carried out by logical swaps (by re-indexing qubits), so it incurs no quantum cost. Since there is no special optimization applied to either AddRoundKey or ShiftRows, we use these simple implementations in our quantum circuit design.

4.2.6 Quantum Implementation of MixColumn.

Prior works have explored both in-place implementations (e.g., assignments like $a \leftarrow a \oplus b$ or $b \leftarrow a \oplus b$, which fit into 32 qubits) and out-of-place implementations (e.g., $c \leftarrow a \oplus b$, requiring additional qubits).

Approaches in [Max19, LXZZ21, LWF+22] rely on temporary storage (ancilla/garbage qubits) or introduce additional depth from cleaning steps, and therefore do not remain fully in-place. For example, [XZL+20] reports a design with 92 in-place CNOT gates (depth 6), while [Max19] gives a 92-XOR out-of-place circuit (depth 6). These stood as the smallest-cost constructions until [LXZZ21] produced a 91-XOR out-of-place version (depth 7). More recently, [YWS+24] matched the

91-CNOT cost with an in-place implementation, notably improving the landscape for in-place MixColumn designs.

Another classical result comes from [LWF+22], which lowered the classical depth to 3 using 103 out-of-place XOR gates (similar to [BFI21]'s 103-XOR, depth-3 construction). Although [LSL+19] required slightly more XOR gates (105), that design tends to give shallower depth once mapped to quantum operations. [BFI21] also introduced two variants: (103 XOR, depth 3) and (95 XOR, depth 6). If translated directly, the former yields 206 CNOT gates, 135 qubits, and quantum depth 11 by offering a possible minor improvement over our shallow/low-depth version. The latter would give 190 CNOT gates and 127 qubits, but we were unable to confirm its correctness due to an encoding issue.

In this work, we translate the 91 CNOT classical MixColumn from [YWS+24] into a quantum circuit and adopt it for the regular version. For the shallow version, we select the in-place MixColumn of [SF24b], which has quantum depth 10.

Method		#CNOT	M	Depth
MixColumn (Naïve)	(Encoding [BP10])	184	64	25
	(Encoding [XZL+20])			52
MixColumn [GLRS16, ZWS+20]		277	32	39
MixColumn [KLSW17]		194	129	15
MixColumn [ASAM18]		275	32	200
MixColumn [Max19]		188	126	13
MixColumn [JNRV20]		277	32	111
MixColumn [TP19]		188	126	17
MixColumn [XZL+20]		92	32	30
MixColumn [ZH22]				29
MixColumn [LPZW23]		98	32	16
MixColumn [LXZZ21]		182	123	16
MixColumn [LWF+22]		206	135	13
MixColumn [LZW23]		204	134	13
MixColumn [BCC+24]		97	32	17
MixColumn [PD24]		159	76	18
MixColumn [LWS+22]	(ASIC1, EGT2)	95	32	40
	(ASIC1, EGT3)	95		40
	(ASIC2, EGT2)	96		40
	(ASIC2, EGT3)	96		32
MixColumn [BDK+21]	(95 b_1)	190	127	14
	(59 b_2, depth 4)	165	91	15
	(59 b_2, depth 5)	162	91	18
	(240.95 (ASIC4) b_3)	290	108	19
MixColumn [BFI21]	(103 XOR, depth 3)	206	135	11
	(95 XOR, depth 6)	190	127	15
MixColumn [YWS+24]	(adopted in regular version)	91	32	35
		98	32	13
MixColumn [SFX23]		198	131	14
MixColumn [LSL+19]		210	137	11
MixColumn [SF24b] (adopted in shallow version)		131	32	10
MixColumn (ours, adopted in shallow/low version)		169	96	8

Table 4.2: The required quantum resources for MixColumn.

4.2.7 Quantum Implementation of MixColumns.

For the 128-bit transformation (i.e., four parallel MixColumns), the above MixColumn circuits are used as-is. Since the regular and shallow variants use in-place MixColumn circuits, they incur no additional ancilla cost.

The situation differs for the shallow/low-depth version since our out-of-place MixColumn requires extra qubits. At first sight, this suggests a higher total qubit count. However, when combined with SubBytes, this increase disappears due to resource sharing.

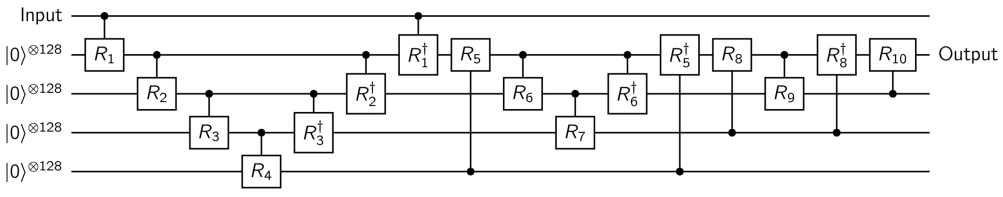
Recall that the S-box implementation used in SubBytes is also not in-place, and therefore uses 20×60, 20×68, or 20×136 ancilla qubits. These ancilla are returned to $|0\rangle$ after each SubBytes† step and

remain idle while MixColumns is executed. By using these idle qubits as temporary storage for MixColumns, we need only 64 qubits (32 input + 32 output) for our out-of-place MixColumn from [LSL+19], and no additional qubits beyond what SubBytes already allocated.

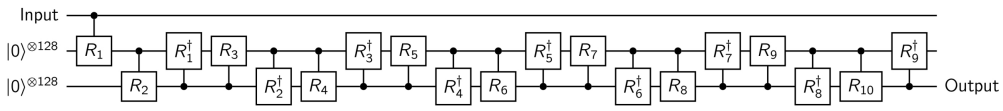
Thus, although the MixColumn circuit itself is not in-place, the overall qubit count does not increase once SubBytes-MixColumns interaction is considered. For any non-standalone MixColumn implementation in Table 4.2, the qubit requirement remains 64 (or 32 for fully in-place designs such as [ASAM18, XZL+20, ZH22]).

4.2.8 Architectures for AES Quantum Circuits.

Several architectural strategies exist for constructing quantum versions of AES, primarily differing in how they store the 128-qubit output produced after each SubBytes operation. Earlier works such as [GLRS16, ASAM18, LPS20] adopted a *zig-zag* structure (Figure 4.2(a)), which reuses four 128-qubit lines and reduces the total qubit count by repeatedly applying reverse operations after each round. A refined zig-zag design using only two lines was introduced in [ZWS+20] (Figure 4.2(b)), made possible by employing a quantum implementation of the inverse S-box. In this style of architecture, every round (and its uncomputation) alternates between two lines, with data being transferred via controlled operations.



(a) Introduced in [GLRS16] and adopted in [LPS20, ASAM18]



(b) Presented in [ZWS+20]

Figure 4.2: Zig-zag implementation of AES-128.

A pipeline architecture represents the opposite end of this design spectrum. Instead of saving qubits through uncomputation, the pipeline allocates a fresh 128-qubit register for each round. This eliminates the need for reversing rounds, significantly reducing depth and gate count. Although this approach uses more qubits, it is often more efficient in practice, especially when many ancilla qubits have already been allocated for parallel components. For our work, this trade-off favors the pipeline, since avoiding reverse operations for each round yields a substantial depth reduction with only a moderate increase in qubits. It is important to note that the reverse operations used for cleaning SubBytes-related ancilla are independent of the reverse operations used in the zig-zag architecture.

Figures 4.3(a) and 4.3(b) illustrate our pipeline approach for the regular version and the shallow/shallow-low-depth versions. Unlike Figure 4.2, where each block corresponds to a full AES round, Figures 10(a, b) separate SubBytes from the rest of the round operations. The symbol SB (with $\oplus$) denotes SubBytes followed by XOR operations.

4.2.8.1 Regular Architecture

The regular design emphasizes parallel execution of SubBytes, the key schedule, and MixColumns. Extra ancilla qubits enable parallelism and are reused after each round via reverse operations. In our design, SubBytes and the key schedule run concurrently, while MixColumns is paired with SubBytes†. The overall depth is dictated by the number of serial SubBytes and SubBytes† stages, since SubBytes† is deeper than

MixColumns. We employ the 91-CNOT MixColumn implementation from [YWS+24] (ported to quantum), which yields the smallest known CNOT count.

For AES-128, SubBytes appears 10 times and SubBytes† appears 9 times, giving 19 serial stages in total. AES-192 requires 23 such stages, and AES-256 requires 27. With an S-box of Toffoli depth 4 and low full depth, each SubBytes stage has depth 58. Hence, our circuits achieve depths of approximately 1090 for AES-128, 1294 for AES-192, and 1516 for AES-256.

The regular design was originally proposed in [JNRV20], but their implementation contained errors related to insufficient ancilla allocation, especially in SubBytes and the key schedule. We corrected these issues and produced a fully functional architecture. Although inspired by [JNRV20], the corrected and practically usable form used in this work is part of our contribution.

4.2.8.2 Shallow and Shallow/Low Architectures

The shallow architecture executes all parallelizable AES components simultaneously. Using an S-box with Toffoli depth 4 and low full depth, this structure needs two sets of ancilla qubits for SubBytes. The first SubBytes uses the first ancilla set; the next SubBytes uses the second, while SubBytes† cleans the first set. Because SubBytes and its uncomputation do not overlap in circuit operations, they can operate concurrently when they use separate ancilla sets. Effectively, every SubBytes† is shifted one time-slot to the right compared to the regular pipeline.

This highly parallel structure makes the per-round depth equal to SubBytes (58) plus MixColumns (10). Since depth is more critical than gate count in Grover's search, using the depth of 10 MixColumn from

[SF24b] is preferable to the depth of 13 version from [YWS+24], despite a minor increase in total CNOTs.

Under this design, AES-128 has depth around 686, AES-192 around 819, and AES-256 around 960. This version minimizes the Toffoli-based complexity metrics and depth-related metrics, making it ideal when parallelization is required.

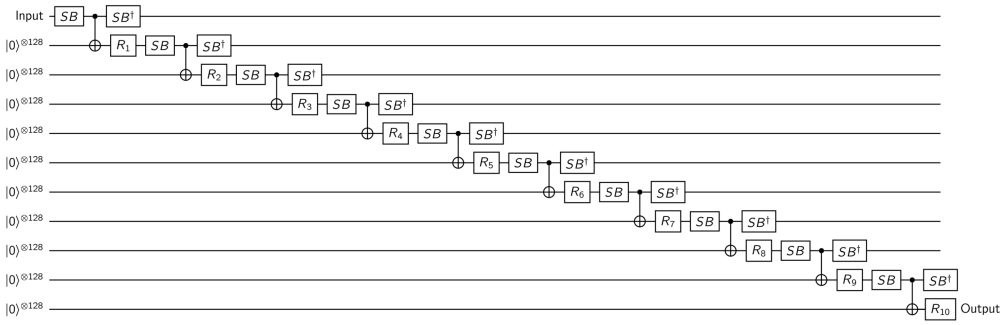
As in [LPZW23], we reduce the required ancilla in the second ancilla set by borrowing idle ancilla qubits from SubBytes†. Once SubBytes† finishes cleaning its ancilla, those qubits become available for the next round's SubBytes. This reduces the total ancilla requirement to slightly above one full set rather than two. We adopt this optimization, achieving even lower qubit usage than in [LPZW23].

The shallow/low-depth version differs from the shallow version only in the MixColumn component. It uses our out-of-place MixColumn implementation (based on [LSL+19]). Since MixColumns can reuse idle ancilla from SubBytes, the required ancilla size does not increase. This version achieves the lowest full depth and the smallest combined depth-gate product, which is often used for estimating quantum attack cost.

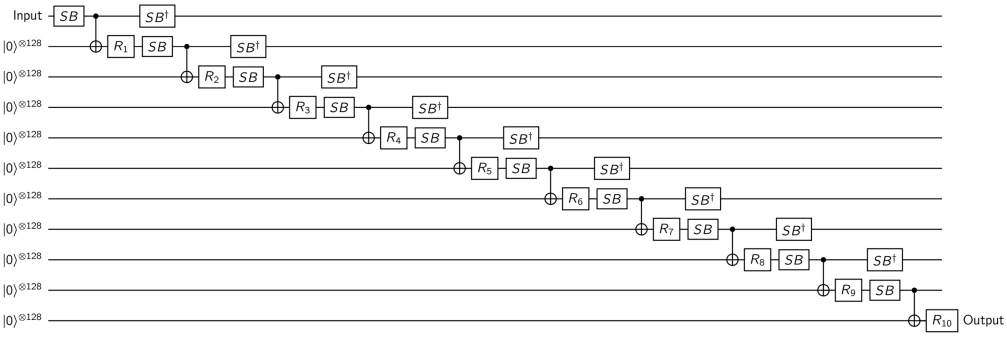
4.2.9 Optimization for Last Round

We further reduce qubit usage by replacing the output qubits of the final round with ancilla qubits originally used in SubBytes. Figure 4.3(c) illustrates the final three rounds of AES-128, showing how ancilla sets ($anc_{0,1,2,3,4}$) correspond to the Toffoli depth layers of the S-box. In the last round, SubBytes† initializes the ancilla sets from the previous round. The anc_3, anc_2, and anc_1 qubits become clean and can be directly reused as final round output qubits. Only anc_0 is not reused, since it would otherwise serve in a non-existent next round. This replacement eliminates

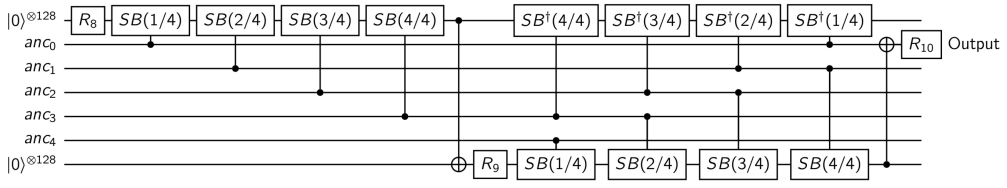
the need to allocate a fresh 128-qubit output register for the final round in both the shallow and shallow/low-depth versions.



(a) Regular architecture



(b) Shallow architecture (same for shallow/low architecture)



(c) Optimization of last round

Figure 4.3: Pipeline implementation of AES-128.

4.3 Performance

This section summarizes the performance of our AES quantum circuit implementations. Table 4.3 reports the quantum resources required by both our implementations and earlier AES quantum circuit designs. Since

Toffoli gates can be decomposed in various ways, the table follows the standard NCT-level (NOT, CNOT, Toffoli) convention for consistent comparison. Only Toffoli-based implementations are included, and not those using AND gates. In designs such as [GLRS16, ASAM18], the Itoh-Tsujii inversion is implemented directly, which results in very expensive SubBytes modules. Works like [LPS20, ZWS+20] improved efficiency by extending the S-box from [BP10], but circuit depth increased because all S-boxes were executed in sequence to save qubits.

Our implementations instead aim to minimize depth while taking qubit usage into account. The $TD-M$ metric from [ZWS+20] is one way to evaluate this trade-off. However, because Grover's search imposes depth limitations, parallelization becomes necessary, making depth even more critical. As highlighted by the TD^2-M metric in Table 4.3, the depth term dominates in realistic attack scenarios. All of our AES designs achieve strong trade-offs by lowering depth and gate count while using a reasonable number of qubits.

	AES	#CNOT	#NOT	#Toffoli	TD	M	TD-M cost	Full depth	TD^2-M cost
128	GLRS [GLRS16]	166548	1456	151552	12672	984	12469248	110799	158010310656
	ASAM [ASAM18]	192832	1370	150528	·	976	·	·	·
	LPS [LPS20]	107960	1570	16940	1880	864	1624320	28927	3053721600
	ZWSLW [ZWS+20]	128517	4528	19788	2016	512	1032192	·	2080899072
	HS [HS22]	126016	2528	17888	820	492	403440	·	330820800
		126016	2528	17888	1558	374	582692	·	907834136
	LXXZZ [LXX+23]	77984	2224	19608	476	474	225624	·	107397024
	LPZW [LPZW23]	65736	800	12920	40	3688	147520	840	5900800
		75024	800	12920	40	4844	193760	770	7750400
	SF [SF24b]	64750	800	12920	40	3268	130720	·	5228800
	Regular	73604	816	12920	76	2736	207936	1288	15803136
	Shallow ❁	75784	816	12824	40	3048	121920	776	4876800
	Shallow/low	79492	816	12824	40	4200	168000	748	6720000
	Regular	62784	816	12560	76	2896	220096	1090	16727296
	Shallow ✿	65092	816	12416	40	3268	130720	686	5228800
	Shallow/low	68800	816	12416	40	4420	176800	667	7072000
	Regular	120480	816	29640	57	4256	242592	1069	13827744
	Shallow ❀	120812	816	29496	30	6128	183840	665	5515200
	Shallow/low	124520	816	29496	30	7280	282816	647	10181376
192	GLRS [GLRS16]	189432	1608	172032	11088	1112	12329856	96956	136713443328
	LPS [LPS20]	125580	1692	19580	1640	896	1469440	25556	2409881600
	ZWSLW [ZWS+20]	152378	5128	22380	2022	640	1294080	·	2616629760
	LXXZZ [LXX+23]	90832	2568	22800	572	538	307736	·	176024992
	LPZW [LPZW23]	74456	896	14552	48	3944	189312	1010	9086976
		85808	896	14552	48	5356	257088	924	12340224
	Regular	83460	904	14552	92	3056	281152	1534	25865984
	Shallow ❁	86388	904	14592	48	3368	161664	932	7759872
	Shallow/low	90920	904	14592	48	4776	229248	900	11003904
	Regular	71272	904	14144	92	3216	295872	1294	295872
	Shallow ✿	73620	904	14000	48	3588	172224	819	8266752
	Shallow/low	78152	904	14000	48	4996	239808	797	11510784
	Regular	136264	904	33384	69	4576	315744	1270	21786336
	Shallow ❀	136812	904	33240	36	6448	232128	795	8356608
	Shallow/low	141344	904	33240	36	7856	282816	773	10181376
256	GLRS [GLRS16]	233836	1943	215040	14976	1336	20007936	130929	299638849536
	LPS [LPS20]	151011	1992	23760	2160	1232	2661120	33525	5748019200
	ZWSLW [ZWS+20]	177645	6103	26774	2292	768	1760256	·	4034506752
	LXXZZ [LXX+23]	110688	3069	27816	646	602	388892	·	251224232
	LPZW [LPZW23]	93288	1119	18360	56	4456	249536	1176	13974016
		106704	1119	18360	56	6124	342944	1074	19204864
	Regular	102932	1111	18088	108	3376	364608	1798	39377664
	Shallow ❁	104464	1111	17992	56	3688	206528	1086	11565568
	Shallow/low	109820	1111	17992	56	5352	299712	1047	16783872
	Regular	87780	1111	17576	108	3536	381888	1516	41243904
	Shallow ✿	89544	1111	17432	56	3908	218848	960	12255488
	Shallow/low	94900	1111	17432	56	5572	312032	934	17473792
	Regular	168580	1111	41496	81	4896	396576	1488	32122656
	Shallow ❀	168548	1111	41316	42	6768	284256	933	11938752
	Shallow/low	173904	1111	41316	42	8432	354144	907	14874048

❁: Toffoli depth 4 S-box (low qubit count).
✿: Toffoli depth 4 S-box (low full depth).
❀: Toffoli depth 3 S-box.

Table 4.3: The required quantum resources for AES quantum circuits (NCT level without decomposition).

In Table 4.3 (NCT level), Toffoli gates are not decomposed. For a finer-grained analysis, we further decompose both Toffoli and AND gates. Following [ZWS+20, Table 10], Table 4.4 provides these expanded resource counts. Table 4.4(a) shows decomposition of Toffoli gates into (8 Clifford + 7 T) with T-depth 4 and full depth 8, following one of the constructions in [AMM+13] (refer to Section 4.4). Table 4.4(b) lists the resources for AND gates from [JNRV20]: each AND requires 1 ancilla, is decomposed into (11 Clifford + 4 T) with T-depth 1and full depth 8, while the inverse $AND^{\dagger}$ (Figure 11(c)) uses (5 Clifford + 1 measurement) with full depth 4.

To convert a Toffoli-based AES circuit into an AND-based one, additional ancilla qubits are needed (specifically, as many as the maximum number of AND gates that operate in parallel). Although idle ancilla qubits from SubBytes or MixColumns could be reused, our implementations do not leave enough ancilla qubits unused, since they are already fully occupied. Consequently, for the AND-based version using the S-box with Toffoli depth 4, 360 ancilla qubits are required, while for the version using the depth-3 S-box, the corresponding ancilla allocation is also made.

AES		#CNOT	#1qCliff	#T	T-depth	M	Full depth	Td-M cost	FD-M cost
128	Regular	148244	14416	87560	304	2736	1288	831744	3523968
	Shallow ❁	150064	19264	87104	160	3048	776	487680	2365248
	Shallow/low	153772	19264	87104	160	4200	748	672000	3141600
	Regular	138144	15496	87920	304	2896	1090	880384	3156640
	Shallow ✿	139588	19168	86912	160	3268	686	522880	2241848
	Shallow/low	143296	19168	86912	160	4420	667	707200	2948140
	Regular	298320	40496	207480	228	4256	1069	970368	4549664
	Shallow ❀	297788	39168	206472	120	6128	665	735360	4075120
	Shallow/low	301496	39168	206472	120	7280	647	873600	4710160
192	Regular	167508	16136	98600	368	3056	1534	1124608	4687904
	Shallow ❁	170804	21568	99008	192	3368	932	646656	3138976
	Shallow/low	175336	21568	99008	192	4776	900	916992	4298400
	Regular	156136	173496	99008	368	3216	1294	1183488	4161504
	Shallow ✿	157620	21272	98000	192	3588	819	688896	2938572
	Shallow/low	162152	21272	98000	192	4996	797	959232	3981812
	Regular	336568	43192	233688	276	4576	1270	1262976	5811520
	Shallow ❀	336252	43864	232680	144	6448	795	928512	5126160
	Shallow/low	340784	43864	232680	144	7856	773	1131264	6072688
256	Regular	207364	19879	122520	432	3376	1798	1458432	6070048
	Shallow ❁	208320	26703	121944	224	3688	1086	826112	4005168
	Shallow/low	213676	240379	121944	224	5352	1047	1198848	5603544
	Regular	193236	21415	123032	432	3536	1516	1527552	5360576
	Shallow ✿	193936	26607	122024	224	3908	960	875392	3751680
	Shallow/low	199292	26607	122024	224	5572	934	1248128	5204248
	Regular	417556	53383	290472	324	4896	1488	1586304	7285248
	Shallow ❀	416444	53983	289212	168	6768	933	1137024	6314544
	Shallow/low	421800	53983	289212	168	8432	907	1416576	7647824

❁: Toffoli depth 4 S-box (low qubit count).
✿: Toffoli depth 4 S-box (low full depth).
❀: Toffoli depth 3 S-box.

(a) Toffoli-based (Clifford + T)

AES		#CNOT	#1qCliff	#T	#Measure	T-depth	M	Full depth	Td-M cost	FD-M cost
128	Regular	134124	43896	27200	6120	40	2968	1021	118720	3030328
	Shallow ❁	136208	43608	27200	6024	40	3408	716	136320	2440128
	Shallow/low	139916	43608	27200	6024	40	4688	684	187520	3206592
	Regular	120944	40296	27200	5760	40	3160	826	126400	2610160
	Shallow ✿	123156	39936	27200	5580	40	3700	666	148000	2464200
	Shallow/low	126864	39936	27200	5580	40	4852	647	194080	3139244
	Regular	259320	94056	62400	14040	30	4864	895	145920	4353280
	Shallow ❀	258900	93456	62400	13860	30	6864	635	205920	4358640
	Shallow/low	262608	93456	62400	13860	30	8016	617	240480	4945872
192	Regular	151324	49456	30464	6936	48	3288	1209	157824	3975192
	Shallow ❁	154292	49672	30464	6976	48	3728	855	178944	3187440
	Shallow/low	158824	49672	30464	6976	48	5264	817	252672	4300688
	Regular	136488	45376	30464	6528	48	3480	975	167040	3393000
	Shallow ✿	139100	45088	30464	6384	48	4020	795	192960	3195900
	Shallow/low	143632	45088	30464	6384	48	5428	773	260544	4195844
	Regular	291952	105952	69888	15912	36	5184	895	186624	5484672
	Shallow ❀	292228	105472	69888	15768	36	7184	759	258624	5452656
	Shallow/low	296760	105472	69888	15768	36	8592	737	309312	6332304
256	Regular	186708	61519	37536	8704	56	3608	1415	202048	5105320
	Shallow ❁	170320	61231	37536	8608	56	4048	1002	226688	4056096
	Shallow/low	193404	61231	37536	8608	56	5712	957	319872	5466384
	Regular	168284	56399	37536	8192	56	3800	1139	212800	4328200
	Shallow ✿	170320	56111	37536	8048	56	4340	932	243040	4044880
	Shallow/low	175676	56111	37536	8048	56	6004	906	336224	5439624
	Regular	360772	131743	86112	19968	42	5504	1238	231168	6813952
	Shallow ❀	360400	131143	86112	19788	42	7504	891	315168	6686064
	Shallow/low	365756	131143	86112	19788	42	9168	865	385056	7930320

❁: Toffoli depth 4 S-box (low qubit count).
✿: Toffoli depth 4 S-box (low full depth).
❀: Toffoli depth 3 S-box.

(b) AND-based

Table 4.4: The required quantum resources for AES quantum circuits with decomposition.

4.4 Quantum Key Search on AES

In this section, we estimate the cost of performing Grover's search for exhaustive key recovery based on our quantum circuit implementations for the three AES variants. The dominant cost comes from the oracle, since the diffusion operator is negligible and straightforward to implement. For this reason, most prior works (including [GLRS16] and [LPS20]) evaluate only the oracle and exclude the diffusion step from their estimates.

Within the oracle, the quantum circuit of the target cipher encrypts a known plaintext while the key is in superposition. The resulting ciphertext is then compared to the known ciphertext, followed by uncomputation to complete each Grover iteration. This comparison uses an n-controlled NOT gate acting on the n-qubit ciphertext. Following the approach in [WR14] and also used in [GLRS16, LPS20], we estimate that an n-controlled NOT gate contributes $(32n-84)$ additional T gates. For a 128-controlled NOT gate, this is 4012 T gates. However, compared to the total number of gates in our AES-128 oracle (e.g., 532960 total gates and 180880 T gates for the regular version with Toffoli-depth of 4 S-box), this increase is negligible. The extra T-depth is relatively large, but earlier works also ignored this overhead. Similarly, [JNRV20] did not include the multi-controlled NOT gate in their estimates. Therefore, we follow the same convention and only add $(32n-84)$ T gates without modifying the T-depth.

To ensure unique key recovery (i.e., preventing false positives), Grassl et al. [GLRS16] analyzed attacks using r known plaintext-ciphertext pairs (for $r=3, 4, 5$). Langenberg et al. [LPS20] later showed that choosing $r=\lceil k/n \rceil$ is sufficient, where k is the key size and n is the block size. We use the same setting.

When $r=1$, the oracle executes the AES circuit twice in series. Thus, the oracle cost is simply twice the circuit cost (excluding qubits). When $r \geq 2$, r block cipher circuits are executed twice in parallel. Only one key schedule is needed, because there is a single input key across all plaintexts.

Overall, the quantum complexity of exhaustive key search is approximately: Oracle cost $\times \left\lfloor \frac{\pi}{4}\sqrt{2^k} \right\rfloor$, where k is the key length. We evaluate this cost using the decomposed Clifford+T gate count multiplied by the full depth.

Tables 4.5(a) and 4.5(b) provide the cost of Grover key search for AES-128, AES-192, and AES-256 using two S-box choices (Toffoli depth 4 and 3), for both Toffoli-based and AND-gate-based implementations. These tables allow us to identify the most efficient strategy for Grover's search under a given depth constraint.

For AES-128, parallelization is unnecessary because its depth is below MAXDEPTH (2^{96}). Thus, without considering parallel execution, the shallow/low-depth design using the Toffoli depth of 4 S-box (low full depth) yields the lowest attack cost in terms of $FD-G$. However, when the more realistic $FD-M$ metric is used, the shallow version with the Toffoli depth of 4 S-box (low full depth) becomes the most efficient. If T-depth is more important (for example, under error-correction constraints) then the best options are the shallow version with the Toffoli depth of 4, low-qubit S-box (Toffoli-based), and the regular version with the same S-box when AND-gates are used (corresponding values are in Tables 4.4(a) and 4.4(b)).

For AES-192 and AES-256, parallelization becomes unavoidable because their depth exceeds MAXDEPTH. Parallel Grover search is highly inefficient, so minimizing FD^2-M and Td^2-M becomes essential. Under

these constraints, the shallow version with the Toffoli depth of 4 S-box (low full depth) is the most efficient in terms of FD^2-M. When optimization focuses on T-depth (Td^2-M), the shallow version with a Toffoli depth of 4, low qubit S-box (Toffoli version) and the regular version with the same S-box (AND version) give the best results.

AES		r	# Qubit (M)	Total gates (G)	Full depth (FD)	FD-G cost ($FD \times G$)	FD-M cost ($FD \times M$)	Cost under MAXDEPTH	
								FD^2-M	Td^2-M
128	Regular	1	2737	$1.511 \cdot 2^{82}$	$1.976 \cdot 2^{74}$	$1.493 \cdot 2^{157}$	$1.320 \cdot 2^{86}$	$1.303 \cdot 2^{161}$	$1.159 \cdot 2^{157}$
	Shallow ✿		3049	$1.549 \cdot 2^{82}$	$1.190 \cdot 2^{74}$	$1.843 \cdot 2^{156}$	$1.772 \cdot 2^{85}$	$1.054 \cdot 2^{160}$	$\mathbf{1.431 \cdot 2^{155}}$
	Shallow/low		4201	$1.571 \cdot 2^{82}$	$1.147 \cdot 2^{74}$	$1.802 \cdot 2^{156}$	$1.176 \cdot 2^{86}$	$1.350 \cdot 2^{160}$	$1.971 \cdot 2^{155}$
	Regular		2897	$1.459 \cdot 2^{82}$	$1.672 \cdot 2^{74}$	$1.220 \cdot 2^{157}$	$1.182 \cdot 2^{86}$	$1.976 \cdot 2^{160}$	$1.227 \cdot 2^{157}$
	Shallow ✿		3629	$1.484 \cdot 2^{82}$	$1.053 \cdot 2^{74}$	$1.562 \cdot 2^{156}$	$\mathbf{1.680 \cdot 2^{85}}$	$\mathbf{1.768 \cdot 2^{159}}$	$1.534 \cdot 2^{155}$
	Shallow/low		4421	$1.506 \cdot 2^{82}$	$1.023 \cdot 2^{74}$	$\mathbf{1.542 \cdot 2^{156}}$	$1.104 \cdot 2^{86}$	$1.130 \cdot 2^{160}$	$1.037 \cdot 2^{156}$
	Regular		4257	$1.643 \cdot 2^{83}$	$1.640 \cdot 2^{74}$	$1.347 \cdot 2^{158}$	$1.704 \cdot 2^{86}$	$1.397 \cdot 2^{161}$	$1.016 \cdot 2^{157}$
	Shallow ❀		6129	$1.634 \cdot 2^{83}$	$1.021 \cdot 2^{74}$	$1.668 \cdot 2^{157}$	$1.527 \cdot 2^{86}$	$1.558 \cdot 2^{160}$	$1.613 \cdot 2^{155}$
	Shallow/low		7281	$1.645 \cdot 2^{83}$	$1.986 \cdot 2^{73}$	$1.634 \cdot 2^{157}$	$1.765 \cdot 2^{86}$	$1.753 \cdot 2^{160}$	$1.917 \cdot 2^{155}$
192	Regular	2	5681	$1.578 \cdot 2^{115}$	$1.177 \cdot 2^{107}$	$1.857 \cdot 2^{222}$	$1.632 \cdot 2^{119}$	$1.920 \cdot 2^{226}$	$1.767 \cdot 2^{222}$
	Shallow ✿		6217	$1.619 \cdot 2^{115}$	$1.429 \cdot 2^{106}$	$1.156 \cdot 2^{222}$	$1.084 \cdot 2^{119}$	$1.549 \cdot 2^{225}$	$\mathbf{1.049 \cdot 2^{221}}$
	Shallow/low		9033	$1.646 \cdot 2^{115}$	$1.380 \cdot 2^{106}$	$1.135 \cdot 2^{222}$	$1.521 \cdot 2^{119}$	$1.049 \cdot 2^{226}$	$1.524 \cdot 2^{221}$
	Regular		5969	$1.526 \cdot 2^{115}$	$1.985 \cdot 2^{106}$	$1.514 \cdot 2^{222}$	$1.446 \cdot 2^{119}$	$1.435 \cdot 2^{226}$	$1.857 \cdot 2^{222}$
	Shallow ✿		6613	$1.546 \cdot 2^{115}$	$1.256 \cdot 2^{106}$	$1.941 \cdot 2^{221}$	$\mathbf{1.013 \cdot 2^{119}}$	$\mathbf{1.273 \cdot 2^{225}}$	$1.115 \cdot 2^{221}$
	Shallow/low		9429	$1.573 \cdot 2^{115}$	$1.222 \cdot 2^{106}$	$\mathbf{1.922 \cdot 2^{221}}$	$1.406 \cdot 2^{119}$	$1.718 \cdot 2^{225}$	$1.591 \cdot 2^{221}$
	Regular		8417	$1.36 \cdot 2^{117}$	$1.948 \cdot 2^{106}$	$1.324 \cdot 2^{224}$	$1.000 \cdot 2^{120}$	$1.949 \cdot 2^{226}$	$1.469 \cdot 2^{222}$
	Shallow ❀		11761	$1.709 \cdot 2^{116}$	$1.219 \cdot 2^{106}$	$1.041 \cdot 2^{223}$	$1.750 \cdot 2^{119}$	$1.066 \cdot 2^{226}$	$1.118 \cdot 2^{221}$
	Shallow/low		14577	$1.722 \cdot 2^{116}$	$1.186 \cdot 2^{106}$	$1.021 \cdot 2^{223}$	$1.055 \cdot 2^{120}$	$1.251 \cdot 2^{226}$	$1.386 \cdot 2^{221}$
256	Regular	2	6257	$1.905 \cdot 2^{147}$	$1.379 \cdot 2^{139}$	$1.313 \cdot 2^{287}$	$1.053 \cdot 2^{152}$	$1.452 \cdot 2^{291}$	$1.339 \cdot 2^{287}$
	Shallow ✿		6881	$1.944 \cdot 2^{147}$	$1.666 \cdot 2^{138}$	$1.619 \cdot 2^{286}$	$1.399 \cdot 2^{151}$	$1.165 \cdot 2^{290}$	$\mathbf{1.579 \cdot 2^{285}}$
	Shallow/low		10209	$1.976 \cdot 2^{147}$	$1.606 \cdot 2^{138}$	$1.586 \cdot 2^{286}$	$1.000 \cdot 2^{152}$	$1.607 \cdot 2^{290}$	$1.171 \cdot 2^{286}$
	Regular		6545	$1.838 \cdot 2^{147}$	$1.163 \cdot 2^{139}$	$1.068 \cdot 2^{287}$	$1.858 \cdot 2^{151}$	$1.08 \cdot 2^{291}$	$1.401 \cdot 2^{287}$
	Shallow ✿		7189	$1.866 \cdot 2^{147}$	$1.472 \cdot 2^{138}$	$1.373 \cdot 2^{286}$	$\mathbf{1.291 \cdot 2^{151}}$	$\mathbf{1.901 \cdot 2^{289}}$	$1.649 \cdot 2^{285}$
	Shallow/low		10517	$1.546 \cdot 2^{148}$	$1.432 \cdot 2^{138}$	$\mathbf{1.358 \cdot 2^{286}}$	$1.838 \cdot 2^{151}$	$1.316 \cdot 2^{290}$	$1.206 \cdot 2^{286}$
	Regular		8993	$1.035 \cdot 2^{149}$	$1.141 \cdot 2^{139}$	$1.180 \cdot 2^{288}$	$1.252 \cdot 2^{152}$	$1.429 \cdot 2^{291}$	$1.080 \cdot 2^{287}$
	Shallow ❀		12337	$1.033 \cdot 2^{149}$	$1.431 \cdot 2^{138}$	$1.478 \cdot 2^{287}$	$1.077 \cdot 2^{152}$	$1.541 \cdot 2^{290}$	$1.589 \cdot 2^{285}$
	Shallow/low		15665	$1.041 \cdot 2^{149}$	$1.391 \cdot 2^{138}$	$1.448 \cdot 2^{287}$	$1.329 \cdot 2^{152}$	$1.849 \cdot 2^{290}$	$1.009 \cdot 2^{286}$

✿: Toffoli depth 4 S-box (low qubit count).
✿: Toffoli depth 4 S-box (low full depth).
❀: Toffoli depth 3 S-box.

(a) Toffoli-based (Clifford + T)

	AES	r	#qubit (M)	Total gates (G)	Full depth (FD)	FD-G cost ($FD \times G$)	FD-M cost ($FD \times M$)	Cost under MAXDEPTH	
								FD^2-M	Td^2-M
128	Regular	1	2969	$1.278 \cdot 2^{82}$	$1.566 \cdot 2^{74}$	$1.001 \cdot 2^{157}$	$1.135 \cdot 2^{86}$	$1.778 \cdot 2^{160}$	$1.360 \cdot 2^{151}$
	Shallow ✿		3409	$1.289 \cdot 2^{82}$	$1.099 \cdot 2^{74}$	$1.416 \cdot 2^{156}$	$\mathbf{1.828 \cdot 2^{85}}$	$1.004 \cdot 2^{160}$	$1.562 \cdot 2^{151}$
	Shallow/low		4689	$1.311 \cdot 2^{82}$	$1.050 \cdot 2^{74}$	$1.376 \cdot 2^{156}$	$1.201 \cdot 2^{86}$	$1.261 \cdot 2^{160}$	$1.074 \cdot 2^{152}$
	Regular		3161	$1.176 \cdot 2^{82}$	$1.268 \cdot 2^{74}$	$1.490 \cdot 2^{156}$	$1.956 \cdot 2^{85}$	$1.239 \cdot 2^{160}$	$1.448 \cdot 2^{151}$
	Shallow ✿		3701	$1.186 \cdot 2^{82}$	$1.021 \cdot 2^{74}$	$1.211 \cdot 2^{156}$	$1.845 \cdot 2^{85}$	$\mathbf{1.885 \cdot 2^{159}}$	$1.695 \cdot 2^{151}$
	Shallow/low		4853	$1.208 \cdot 2^{82}$	$1.986 \cdot 2^{73}$	$\mathbf{1.200 \cdot 2^{156}}$	$1.176 \cdot 2^{86}$	$1.168 \cdot 2^{160}$	$1.111 \cdot 2^{152}$
	Regular		4865	$1.294 \cdot 2^{83}$	$1.373 \cdot 2^{74}$	$1.776 \cdot 2^{157}$	$1.630 \cdot 2^{86}$	$1.119 \cdot 2^{161}$	$\mathbf{1.281 \cdot 2^{151}}$
	Shallow ❀		6865	$1.290 \cdot 2^{83}$	$1.949 \cdot 2^{73}$	$1.257 \cdot 2^{157}$	$1.633 \cdot 2^{86}$	$1.591 \cdot 2^{160}$	$1.807 \cdot 2^{151}$
	Shallow/low		8017	$1.301 \cdot 2^{83}$	$1.893 \cdot 2^{73}$	$1.231 \cdot 2^{157}$	$1.852 \cdot 2^{86}$	$1.752 \cdot 2^{160}$	$1.055 \cdot 2^{152}$
192	Regular	2	6073	$1.332 \cdot 2^{115}$	$1.854 \cdot 2^{106}$	$1.234 \cdot 2^{222}$	$1.374 \cdot 2^{119}$	$1.274 \cdot 2^{226}$	$1.018 \cdot 2^{217}$
	Shallow ✿		6865	$1.347 \cdot 2^{115}$	$1.311 \cdot 2^{106}$	$1.765 \cdot 2^{221}$	$\mathbf{1.098 \cdot 2^{119}}$	$1.440 \cdot 2^{225}$	$1.150 \cdot 2^{217}$
	Shallow/low		9937	$1.374 \cdot 2^{115}$	$1.253 \cdot 2^{106}$	$1.721 \cdot 2^{221}$	$1.519 \cdot 2^{119}$	$1.904 \cdot 2^{225}$	$1.665 \cdot 2^{217}$
	Regular		6425	$1.224 \cdot 2^{115}$	$1.496 \cdot 2^{106}$	$1.831 \cdot 2^{221}$	$1.173 \cdot 2^{119}$	$1.755 \cdot 2^{225}$	$1.077 \cdot 2^{217}$
	Shallow ✿		7397	$1.234 \cdot 2^{115}$	$1.219 \cdot 2^{106}$	$1.504 \cdot 2^{221}$	$1.100 \cdot 2^{119}$	$\mathbf{1.341 \cdot 2^{225}}$	$1.240 \cdot 2^{217}$
	Shallow/low		10213	$1.261 \cdot 2^{115}$	$1.186 \cdot 2^{106}$	$\mathbf{1.495 \cdot 2^{221}}$	$1.478 \cdot 2^{119}$	$1.753 \cdot 2^{225}$	$1.712 \cdot 2^{217}$
	Regular		9489	$1.349 \cdot 2^{116}$	$1.623 \cdot 2^{106}$	$1.094 \cdot 2^{223}$	$1.879 \cdot 2^{119}$	$1.525 \cdot 2^{226}$	$\mathbf{1.773 \cdot 2^{216}}$
	Shallow ❀		13089	$1.348 \cdot 2^{116}$	$1.165 \cdot 2^{106}$	$1.570 \cdot 2^{222}$	$1.861 \cdot 2^{119}$	$1.084 \cdot 2^{226}$	$1.223 \cdot 2^{217}$
	Shallow/low		15905	$1.361 \cdot 2^{116}$	$1.130 \cdot 2^{106}$	$1.537 \cdot 2^{222}$	$1.096 \cdot 2^{120}$	$1.239 \cdot 2^{226}$	$1.486 \cdot 2^{217}$
256	Regular	2	6649	$1.605 \cdot 2^{147}$	$1.085 \cdot 2^{139}$	$1.741 \cdot 2^{286}$	$1.761 \cdot 2^{151}$	$1.910 \cdot 2^{290}$	$1.499 \cdot 2^{281}$
	Shallow ✿		7441	$1.504 \cdot 2^{147}$	$1.537 \cdot 2^{138}$	$1.155 \cdot 2^{286}$	$1.396 \cdot 2^{151}$	$1.072 \cdot 2^{290}$	$1.678 \cdot 2^{281}$
	Shallow/low		10769	$1.643 \cdot 2^{147}$	$1.468 \cdot 2^{138}$	$1.205 \cdot 2^{286}$	$1.929 \cdot 2^{151}$	$1.416 \cdot 2^{290}$	$1.214 \cdot 2^{282}$
	Regular		7001	$1.474 \cdot 2^{147}$	$1.747 \cdot 2^{138}$	$1.287 \cdot 2^{286}$	$1.493 \cdot 2^{151}$	$1.304 \cdot 2^{290}$	$1.579 \cdot 2^{281}$
	Shallow ✿		7973	$1.483 \cdot 2^{147}$	$1.429 \cdot 2^{138}$	$1.059 \cdot 2^{286}$	$\mathbf{1.390 \cdot 2^{151}}$	$\mathbf{1.987 \cdot 2^{289}}$	$1.798 \cdot 2^{281}$
	Shallow/low		11301	$1.515 \cdot 2^{147}$	$1.389 \cdot 2^{138}$	$\mathbf{1.052 \cdot 2^{286}}$	$1.916 \cdot 2^{151}$	$1.330 \cdot 2^{290}$	$1.274 \cdot 2^{282}$
	Regular		10065	$1.628 \cdot 2^{148}$	$1.899 \cdot 2^{138}$	$1.545 \cdot 2^{287}$	$1.166 \cdot 2^{152}$	$1.107 \cdot 2^{291}$	$\mathbf{1.267 \cdot 2^{281}}$
	Shallow ❀		13665	$1.625 \cdot 2^{148}$	$1.367 \cdot 2^{138}$	$1.110 \cdot 2^{287}$	$1.140 \cdot 2^{152}$	$1.558 \cdot 2^{290}$	$1.720 \cdot 2^{281}$
	Shallow/low		16993	$1.641 \cdot 2^{148}$	$1.327 \cdot 2^{138}$	$1.088 \cdot 2^{287}$	$1.376 \cdot 2^{152}$	$1.826 \cdot 2^{290}$	$1.069 \cdot 2^{282}$

✿: Toffoli depth 4 S-box (low qubit count).
✿: Toffoli depth 4 S-box (low full depth).
❀: Toffoli depth 3 S-box.

(b) AND-based

Table 4.5: The required quantum resources for Grover's key search on AES

Finally, Table 4.6 summarizes the NIST security levels implied by our quantum resource estimates, alongside previous works. Our designs consistently reduce the effective quantum cost for Grover's search across all AES variants, resulting in higher NIST security levels compared to earlier estimates. These values reflect the product of decomposed gate count and full depth for $2^{k/2}$ Grover iterations. MAXDEPTH is not applied in Table 4.6; if it were enforced, the values would scale down accordingly by dividing by MAXDEPTH.

Level (AES)	N$^+$'16 [NIS16b] (G$^+$ [GLRS16])	L$^+$ [LPS20]	N$^+$'22 [NIS22b]	This work		
				Regular	Shallow	Shallow/low
1 (128)	2^{170} ($2^{168.6683}$)	$2^{162.6093}$	2^{157}	❁: $2^{157.0014}$ ✿: $2^{156.5753}$ ❀: $2^{157.8286}$	❁: $2^{156.5018}$ ✿: $2^{156.2762}$ ❀: $2^{157.3300}$	❁: $2^{156.4605}$ ✿: $\mathbf{2^{156.2630}}$ ❀: $2^{157.2998}$
3 (192)	2^{233} ($2^{233.4645}$)	$2^{227.6491}$	2^{221}	❁: $2^{222.3033}$ ✿: $2^{221.8726}$ ❀: $2^{223.1296}$	❁: $2^{221.8197}$ ✿: $2^{221.5880}$ ❀: $2^{222.6508}$	❁: $2^{221.7832}$ ✿: $\mathbf{2^{221.5801}}$ ❀: $2^{222.6201}$
5 (256)	2^{298} ($2^{298.3467}$)	$2^{292.3100}$	2^{285}	❁: $2^{286.7999}$ ✿: $2^{286.3640}$ ❀: $2^{287.6276}$	❁: $2^{286.2079}$ ✿: $2^{286.0827}$ ❀: $2^{287.1506}$	❁: $2^{286.2690}$ ✿: $\mathbf{2^{286.0731}}$ ❀: $2^{287.1217}$

❁: Toffoli depth 4 S-box (low qubit count).
✿: Toffoli depth 4 S-box (low full depth).
❀: Toffoli depth 3 S-box.

Table 4.6: Thresholds for NIST post-quantum security levels

4.5 Conclusion

In this work, we consolidate a broad range of recent advances on AES building blocks, particularly optimized designs for S-boxes, MixColumns, and architectural strategies, into a unified framework. By doing so, we significantly reduce the quantum circuit cost for the AES family. In addition to revisiting component level optimizations, we analyze the architectural design space and incorporate efficient constructions of S-boxes and MixColumns.

Our circuits achieve the smallest Toffoli depth and full depth across all AES variants, improving upon the results of [ZWS+20] and [HS22] by more than 97% and 95%, respectively. Moreover, we reduce the Toffoli depth-qubit count product by over 87% and 69%, and achieve more than 99% and 98% improvement in the squared product compared with prior works. Our results improve upon recent advances from Asiacrypt'22 [HS22], Asiacrypt'23 [LPZW23], and Asiacrypt'24 [SF24b], and, to the best of our knowledge, represent the state of the art in quantum analysis of AES.

Much of the previous literature on AES quantum implementations has emphasized minimizing qubit usage [GLRS16, LPS20, ZWS+20, ASAM18, WWL22]. Our approach takes a different direction by allowing a moderate increase in qubits to substantially reduce depth, thereby improving cost metrics that multiply depth or gate count by qubit count. Lower depth also enables a higher number of feasible Grover iterations, which is pivotal for reducing the overall quantum search complexity in exhaustive key search.

Overall, this work integrates nearly all relevant research on AES quantum circuits (including recent contributions such as [HS22, LPZW23, LGQW23, LXX+23, ZH22]) allowing us to select optimal components and propose three architectural variants. Notably, the shallow architecture we introduce has already been adopted in [LPZW23, SF24b]. We provide four new AES S-box designs and one new MixColumn construction, and our implementations currently give the most efficient known quantum attacks on all three AES variants. Beyond circuit construction, we also present detailed discussions on theoretical aspects such as NIST security levels [NIS16, NIS22], which we expect to be valuable references for future research.

V. Quantum Attacks on SHA-2/3

Author's Contribution. This chapter draws on the findings reported in [JLO+25]. My primary role was the optimization of the quantum circuits employed in the quantum security evaluation of SHA-3. The optimization of the SHA-2 quantum circuits was carried out chiefly by Sejin Lim (the second author of [JLO+25]), and I provided partial support to her work.

5.1 Introduction

NIST specifies five post-quantum security levels [NIS16, NIST22a], with Levels 1, 3, and 5 corresponding to the quantum circuit complexities of recovering AES-128, AES-192, and AES-256 keys via Grover's algorithm. In contrast, Levels 2 and 4, associated with collision search for the SHA-2 and SHA-3 hash families, currently lack official quantum circuit cost estimates. To the best of our knowledge, only classical circuit sizes have been reported. Motivated by this gap, we present an effort to optimize quantum implementations of the major cryptographic hash functions, namely the SHA-2 and SHA-3 variants, incorporating a range of design-level improvements. Overall, our aim is to deliver more resource-efficient quantum circuits for these hash functions, achieving substantial improvements over prior works [LLLC23, KHJ18, ADMG+17, MSDM22, HS20, SJS23, LKL+24] and enabling more concrete candidates for NIST's quantum post-quantum security level definitions.

5.1.1 Contribution

5.1.1.1 Carry-save adder for SHA-2.

We develop improved quantum circuits for SHA-2 by employing the

Quantum Carry-Save Adder (QCSA) [Gos98] for multi-operand addition, which is one of the dominant cost components in SHA-2. Using the QCSA enables us to realize an addition path of roughly depth 1 within each round—substantially outperforming the previous best result reported by Lee et al. [LLLC23], which had a depth of 3. Our design further incorporates several efficient techniques, including output-qubit reuse and fixed-input optimizations, all formulated within an out-of-place computation framework.

5.1.1.2 New architecture for SHA-3.

We introduce a new architectural concept, the interval architecture, which allows extensive reuse and reduction of ancilla qubits without increasing the circuit depth. This is enabled by inserting a reverse operation every four rounds within our out-of-place SHA-3 design. With these techniques, our SHA-3 circuits achieve the smallest known depth while maintaining a practical qubit count.

5.1.1.3 Quantum attack complexity for NIST levels 2 and 4.

Under realistic assumptions for Grover's search, our depth-optimized SHA-2 and SHA-3 implementations provide a significantly better trade-off compared to qubit-optimized approaches. Using the cost estimates of our quantum collision search circuits, we formulate explicit quantum-attack complexities for NIST security levels 2 and 4, which had not been previously defined.

5.2 Background

5.2.1 Quantum Collision Search (based on Grover's algorithm)

In Asiacrypt 2017, Chailloux, Naya-Plasencia, and Schottenloher introduced a quantum collision-search algorithm, commonly referred to

as the CNS algorithm [CNPS17], which achieves a query complexity of $O(2^{2n/5})$ while requiring only $O(2^{n/5})$ classical memory. Although its asymptotic cost is higher than that of the BHT algorithm [BHT97], CNS avoids the need for quantum memory, making it far more practical. For this reason, we adopt the CNS algorithm when estimating the quantum attack cost on SHA-2 and SHA-3.

The CNS method is built upon Quantum Amplitude Amplification (QAA) [BHMT02], a generalization of Grover's algorithm, and consists of two stages: generating a list and then applying amplitude amplification to locate a collision. Below, we summarize the complexity argument (a detailed treatment appears in [CNPS17]).

Let S_H^d be the set of all input/output pairs $(x, H(x))$ where the hash output begins with d zero bits. In the first stage, a list L of size 2^{t-d} is produced from S_H^d using Grover's algorithm at a cost of $2^{d/2}$. Hence, constructing the entire list incurs complexity $2^{t-d/2}$. The optimal values of t and d are $t=\frac{3n}{5}$ and $d=\frac{2n}{5}$.

In the second stage, QAA is invoked $2^{(n-t-1)/2}$ times. Accessing list L requires $2^{d/2}+2^{t-d}$ operations, giving a total cost of $2^{(n-t-1)/2}(2^{d/2}+2^{t-d})+2^{t-2/d}$.

With the optimized parameters above, the overall collision-finding complexity becomes $O(2^{2n/5})$ with classical memory $O(2^{n/5})$.

The authors of CNS also proposed a parallelization technique that further lowers the time exponent. Using 2^s quantum instances, the exponent becomes $\frac{2n-3s}{5}$ (valid for $s \le n/4$). In our work, to define meaningful boundaries for the NIST post-quantum security categories, we follow their recommendation and set $s=n/6$, assuming sufficiently large classical

memory. Under this assumption, the collision-search complexities for SHA-2-n and SHA-3-n (with $n = 256, 384, 512$) reduce to approximately 2^{76} (level 2) and 2^{115} (level 4), respectively.

These values match the required complexities for levels 1, 3, and 5, which are based on Grover's exhaustive key search for AES-128, AES-192, and AES-256: 2^{64}, 2^{96} and 2^{128}, respectively.

We stress that running $2^{n/6}$ parallel quantum instances requires a substantial number of qubits. Nonetheless, the use of $2^{n/6}$ parallelization in CNS is intended to provide consistent boundaries for levels 2 and 4. These bounds may be adjusted by reducing the degree of parallelization rather than increasing the overall search complexity.

5.2.2 Related work on quantum circuits for SHA-2 and SHA-3

The first quantum circuit constructions for SHA-2 were introduced by Amy et al. [ADMG+17]. Their design, as well as the follow-up work in [KHJ18], resulted in large circuit depths because the core components, Ch, Maj, and the modular additions, were executed strictly one after another. Later, Lee et al. [LLLC23] provided improved SHA-2 circuits that significantly lowered the Toffoli depth while keeping the qubit usage practical.

For SHA-3, [ADMG+17] proposed an in-place round architecture aimed at qubit minimization, which required every subroutine to be fully reversible. Song et al. [SJS23] later designed a SHA-3 circuit that reduces T-depth by removing reversibility constraints and employing additional ancilla. Other works, such as Häner et al. [HS20] and Meuli et al. [MSDM22], introduced heuristic methods for optimizing Toffoli-related costs (Toffoli count, T-count, and T-depth). These papers did not provide explicit SHA-2 or SHA-3 circuits; instead, they estimated resource requirements using their optimization frameworks.

More recently, Lee et al. [LKL+24] developed SHA-3 circuits with reduced Toffoli depth while preserving the in-place architecture. Although not the main focus of their work, they also constructed an out-of-place SHA-3 design that uses fewer qubits than those in [SJS23, HS20, MSDM22].

Overall, our work delivers the lowest Toffoli depth and full depth implementations for both SHA-2 and SHA-3 to date. We also obtain the best or second-best performance for the combined metrics of depth × qubit count. Furthermore, when considering quantum collision search, our circuits enable the most efficient parallelization strategy for Grover's algorithm.

5.3 Quantum Circuits for SHA-2

In this section, we explain depth-optimized quantum circuit implementations for SHA-2, concentrating on the SHA-2-256 variant. Since SHA-2-384 and SHA-2-512 follow essentially the same internal design, the proposed techniques apply to these versions as well.

5.3.1 Quantum Implementation of Σ_0, Σ_1, σ_0 and σ_1

SHA-2 uses four linear layer operations, where the symbols "$\ggg$" and "$\lll$" denote right rotation and right shift, respectively:

$$\begin{aligned}
\Sigma_0(a) &= (a \ggg 2) \oplus (a \ggg 13) \oplus (a \ggg 22).\\
\Sigma_1(e) &= (e \ggg 6) \oplus (e \ggg 11) \oplus (e \ggg 25).\\
\sigma_0(W_{t-15}) &= (W_{t-15} \ggg 7) \oplus (W_{t-15} \ggg 18) \oplus (W_{t-15} \gg 3).\\
\sigma_1(W_{t-2}) &= (W_{t-2} \ggg 17) \oplus (W_{t-2} \ggg 19) \oplus (W_{t-2} \gg 10).
\end{aligned}$$

Since each of these operations is composed solely of XOR functions, every linear layer can be expressed as a binary matrix. By applying PLU factorization to this matrix, we obtain three component matrices, permutation, lower triangular, and upper triangular. Using these factors,

the linear layer can be implemented in an in-place manner, following the approach described in [RBC23].

In [LLLC23], the authors provided in-place realizations of Σ_0, Σ_1, σ_0 and σ_1 by applying PLU factorization. However, their method increases the quantum depth because many CNOT gates must operate within a limited qubit space (only inside the input register). As a result, up to 193 CNOT gates are executed on 32 qubits, producing a sequential execution pattern with a maximum depth of 55.

5.3.1.1 Out-of-place Approach.

In-place implementations reuse the existing variables and overwrite them. In contrast, out-of-place implementations introduce fresh output variables so that the original inputs remain unchanged. We design out-of-place versions of Σ_0, Σ_1, σ_0 and σ_1. We allocate 32 qubits as the output register and compute the results by XOR-ing the inputs into the output using CNOT gates. For example, Σ_0 is computed with CNOT $(a \gg 2 \rightarrow output)$, CNOT$(a \gg 13 \rightarrow output)$, and CNOT$(a \gg 22 \rightarrow output)$.

Consequently, each of our Σ_0, Σ_1, σ_0 and σ_1 circuits achieves depth 3 and uses at most 96 CNOT gates. Because σ_0 and σ_1 include shift operations ($\gg 3$ and $\gg 10$), the required CNOT count drops by 3 and 10, respectively, when excluding zero-padding regions. Note that swaps and shifts can be implemented purely by reindexing, without quantum gates. For example, Swap (x, y) can be done logically by mapping $x_{new} = y$ and $y_{new} = x$.

5.3.1.2 Reuse of Output Qubits.

The results of Σ_0, Σ_1, σ_0 and σ_1 serve as operands for the adders. After the additions, these results are no longer needed. We therefore restore the output register to the clean state $|0\rangle$ by reversing the earlier

operations, enabling the same qubits to be reused. By paying the initial cost once, all subsequent linear layers can benefit from low depth and minimal qubit usage. The required quantum resources are summarized in Table 5.1.

Linear operation	Method	#CNOT	#Qubit (reuse)	Depth
Σ_0	[LLLC23]	166	32	55
	This work (Out-of-place)	96	64 (32)	3
Σ_1	[LLLC23]	166	32	44
	This work (Out-of-place)	96	64 (32)	3
σ_0	[LLLC23]	193	32	50
	This work (Out-of-place)	93	64 (32)	3
σ_1	[LLLC23]	142	32	40
	This work (Out-of-place)	86	64 (32)	3

Table 5.1: The required quantum resources for linear operations in SHA-2.

5.3.2 Quantum Implementation of *Ch* and *Maj*

In SHA-2, the *Ch* and *Maj* operations are as follows:

$$Ch(e, f, g) = (e \cdot f) \oplus (\sim e \cdot g).$$
$$Maj(a, b, c) = (a \cdot b) \oplus (a \cdot c) \oplus (b \cdot c).$$

We adopt the optimized circuits from [CDKM08, LLLC23], each of which uses only a single Toffoli gate. Once the additions for generating W_t are finished (see Section 5.3.3), the *Ch* and *Maj* outputs are no longer needed. Since the input qubits (a, b, c, d, e, f, g) are required again in the next round, we subsequently uncompute *Ch* and *Maj* to restore their inputs. Table 5.2 lists the resource counts of our *Ch* and *Maj* implementations.

5.3.3 Quantum Implemenation of Multi-operand Addition

In SHA-2, additions dominate the resource cost. They are required both in the round function and in the message-scheduling step:

$$h' = \Sigma_0(a) + Maj(a,b,c) + \Sigma_1(e) + Ch(e,f,g) + h + K_i + W_i.$$
$$d' = d + \Sigma_1(e) + Ch(e,f,g) + h + K_i + W_i.$$

$$W_t = M_t \quad (0 \le t \le 15)$$
$$= \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} \quad (16 \le t \le 63).$$

In [LLLC23], the authors formed a 3-adder critical path using Draper's adder [DKRS04] and Takahashi's adder [TTK09], which operate only on two inputs $(x+y)$.

5.3.3.1 Carry-Save Adder

We instead employ a multi-operand adder called the Quantum Carry-Save Adder (QCSA) [Gos98] to parallelize additions. The Wallace-tree-based QCSA proposed in [KLW+25] greatly reduces depth for multi-operand additions. Although QCSA requires many ancilla qubits for parallelization, we reclaim most of these qubits by uncomputing intermediate results. With effective use of QCSA, we achieve a critical path of roughly 1 (improving the previous best value of 3 from [LLLC23]) and obtain a low overall depth.

Figure 5.1 illustrates our QCSA-based implementation. Three QCSA modules are stacked vertically and operate in parallel, each having a critical path of about 1. For both the round function and message scheduling, we use QCSAs configured with 7, 4, and 4 operands. The detailed procedure is summarized below:

1) Partial Sums

QCSA is composed of Quantum Full Adders (QFA, black dashed boxes, three operands) and Quantum Half Adders (QHA, blue dashed boxes, two operands). Carry bits appear on the high-order wires. Partial sums are computed until only two operands remain (the yellow dashed boxes). These steps (①-1 to ①-4) are executed in four sequential layers;

the three QCSA modules run in parallel.

①-1: The additions for computing h' are executed. Five operands (Σ_1, Ch, h, K_i, W_i) overlap with those used in computing d′ , so we reorder them for efficiency. The result of h + Σ_1 + Ch is stored in the Ch register (red). The $W_i + K_i + \Sigma_0$ partial sum is stored in Σ_0 (red), as these operands are needed later for d'.

①-2: Updated Ch and Σ_0 feed into the QFA for $Ch + \Sigma_0 + Maj$. $W_i + K_i + d$ is computed simultaneously, with the result stored in d.

①-3: The QHA is applied to compute $d + Ch$ (only two operands). In parallel, $W_i + \sigma_0 + W_{i+9}$ and the rest of $Ch + \Sigma_0 + Maj$. are computed.

①-4: Partial sums continue until two operands remain.

② Total Sum: The two remaining operands are added using Draper's adder (out-of-place).

③ Reverse: We uncompute all intermediate additions from step ①, restoring the updated operands to their original states and resetting ancilla qubits.

Table 5.2 compares our addition costs with [KHJ18, LLLC23]. Our depth is determined by the leftmost QCSA block in Figure 5.1, giving the lowest Toffoli and full depths.

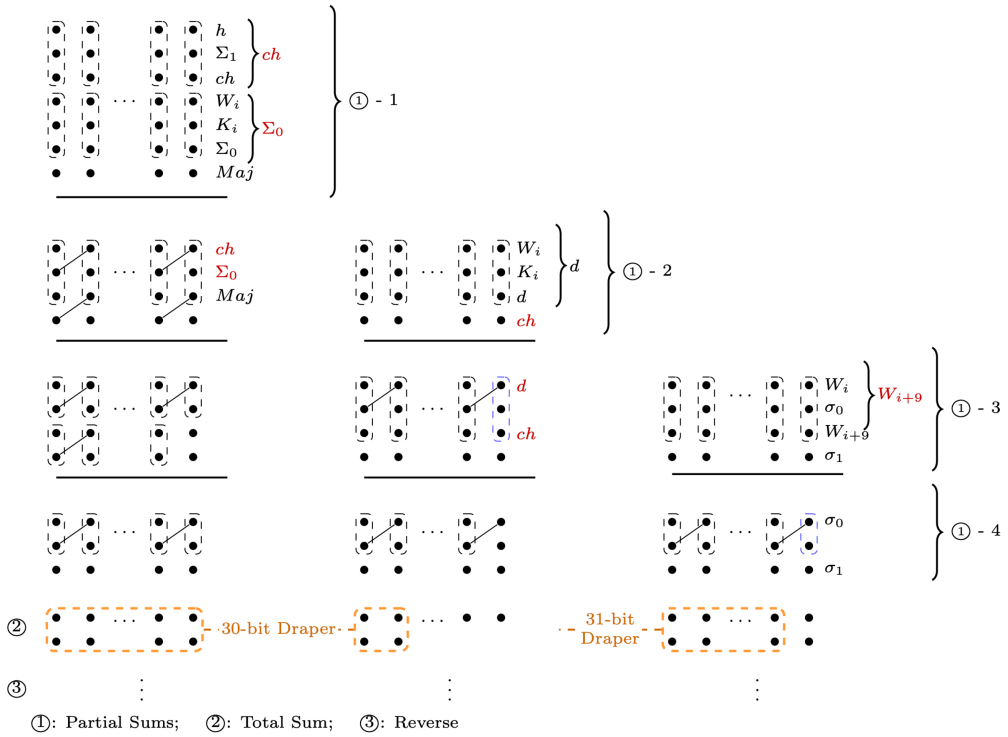


Figure 5.1: Overview of the additions in SHA-2 using QCSA.

Method	#CNOT	#1qCliff	#T	Toffoli depth	#Qubit (reuse)	Full depth
[KHJ18]	18367	3362	19124	224	501 (85)	1777
[LLLC23]	-	-	-	66	546 (162)	-
This work	9220	1487	6546	19	819 (371)	180

Table 5.2: The required quantum resources for multiple additions in SHA-2 (one round).

5.3.3.2 Optimizing for fixed input length.

In SHA-2-256, the input length is fixed at 256 bits. Words W_8 and W_{15} are padded with constants (via X gates), and $W_9 - W_{14}$ are set to zero. Thus, after W_8 and W_{15} are consumed, we reset and reuse these 256 qubits for later W values. We also skip additions involving $W_9 - W_{14}$ since they are zero. Applying this from rounds $10 - 15$ reduces the Toffoli

depth by 12 (2 per round).

5.3.3.3 Considerations for AND gate realizations

For an AND gate, the target qubit must be $|0\rangle$ (clean), unlike the Toffoli gate where its value is irrelevant. To realize AND-based circuits for Ch, Maj, and partial sums (step ① in the QCSA), additional ancillae are needed. However, for Ch and Maj, we borrow idle qubits (clean $|0\rangle$) from Draper's adders within the QCSA (step ③ of Figure 5.1), so extra ancilla qubits are allocated only for the AND gates inside Draper's adders. Borrowed qubits are reinitialized after the AND operations.

5.4 Quantum Circuits for SHA-3

SHA-3 consists of 24 rounds, where each round applies the sequence of operations θ, ρ, π, χ, and ι. The quantum version processes a 1600-qubit state, denoted $S[x][y][z]$, which forms a three-dimensional array with dimensions of size 5, 5, and 64.

Amy et al. [ADMG+17] introduced an in-place round architecture, in which the reversed forms of θ and χ (i.e., $\theta^{-1}$and χ^{-1}) were used to reset and reuse qubits. In contrast, Meuli et al. [MSDM22] and Häner et al. [HS20] proposed an out-of-place design primarily aimed at reducing the Toffoli depth by optimizing the χ operation, which is the only component of SHA-3 that requires Toffoli gates. Song et al. [SJS23] also adopted an out-of-place design to further reduce both Toffoli and full depths. The out-of-place approaches in [HS20, MSDM22] achieve better performance, in terms of qubit count and Toffoli depth, than the design of [SJS23].

More recently, Lee et al. [LKL+24] developed Toffoli-depth-optimized SHA-3 circuits while still preserving an in-place design, offering substantial improvements over [ADMG+17]. They additionally provided

an alternative out-of-place SHA-3 implementation that lowers qubit usage relative to [MSDM22, HS20, SJS23].

Among these out-of-place methods (i.e., depth-focused implementations), the best result before our work was due to Lee et al. [LKL+24], achieving a Toffoli depth of 24 while requiring 43200 qubits. With our new architecture, we provide the lowest Toffoli depth to date and reduce the number of required qubits to 24 and 22400, respectively.

We propose an improved out-of-place construction that simultaneously minimizes Toffoli depth, reduces overall depth, and substantially lowers qubit consumption. The following sections provide detailed explanations of how each component is optimized.

5.4.1 Quantum Implementation of θ

In the θ step, the value of $S(x, y, z)$ is updated by XORing it with the XOR of the $S(x-1, z)$ column and the $S(x+1, z-1)$ column as follows.

$$S[x][y][z] = S[x][y][z]) \oplus \left(\bigoplus_{i=0}^{4} S[x-1][i][z] \oplus S[x+1][i][z-1] \right).$$

For the θ operation, Amy et al. [ADMG+17] used 17,600 CNOT gates and additionally applied its reverse operation θ^{-1} to reset qubits to the zero state. While this reduces qubit usage through reuse, computing θ^{-1} requires 1,360,000 CNOT gates (far more than θ itself). Even if one assumes the reverse merely resets the inputs, the cost still equals another 17600 CNOTs. Their combined implementation uses a total of 1377600 CNOT gates with depth 300.

Our design avoids the reverse operation entirely, as our philosophy prioritizes depth minimization rather than minimizing qubit usage. We

introduce an optimized θ construction that uses only 4800 CNOT gates and achieves a depth of 15.

Our method applies an *all-in-one* strategy. 320 ancilla qubits are allocated to store XOR values computed across the columns $S[x-1][z]$ and $S[x+1][z-1]$ for all indices, requiring 1600 CNOT gates and depth 5. These pre-computed values are repeatedly reused to update $S[x][y][z]$, avoiding recomputation. Updating each state requires 3200 CNOTs with an additional depth of 10. Consequently, θ is implemented with 4800 CNOT gates and depth 15.

By comparison, Song et al. [SJS23] also omitted θ^{-1} but allocated 1600 ancilla qubits, resulting in 24,000 CNOT gates and depth 79. In [MSDM22, HS20], only optimizations for χ were presented, without specifying detailed circuits for θ. Similarly, [LKL+24] focused exclusively on χ and did not give a construction for θ.

Table 5.3 summarizes the required quantum resources for our θ implementation in comparison to previous work. The resource figures for [ADMG+17] include both θ and θ^{-1}, and 1600 of the 3200 ancilla qubits in their approach are reused. In our method, the 320 ancilla qubits used in the XOR preparation stage are reused as well.

Method	#CNOT	#Qubit (reuse)	Depth
[ADMG+17]	1377600	3200 (1600)	300
[SJS23]	24000	3200	79
This work	4800	1920 (320)	15

Table 5.3: The required quantum resources for θ in SHA-3.

5.4.2 Quantum Implementation of χ

The χ step is the part of the SHA-3 quantum circuit that consumes the most quantum resources, since it is the only operation in the algorithm that requires Toffoli gates. It is the sole non-linear component

of SHA-3, and its effect on the state S is expressed as:

$$S[x][y][z] = S[x][y][z] \oplus (\sim S[x+1][y][z] \cdot S[x+2][y][z]).$$

Following our two main design principles, allocating ancilla qubits and reusing them, we construct a depth-optimized quantum circuit for χ. Figure 5.2 provides the quantum circuit implementing this χ operation.

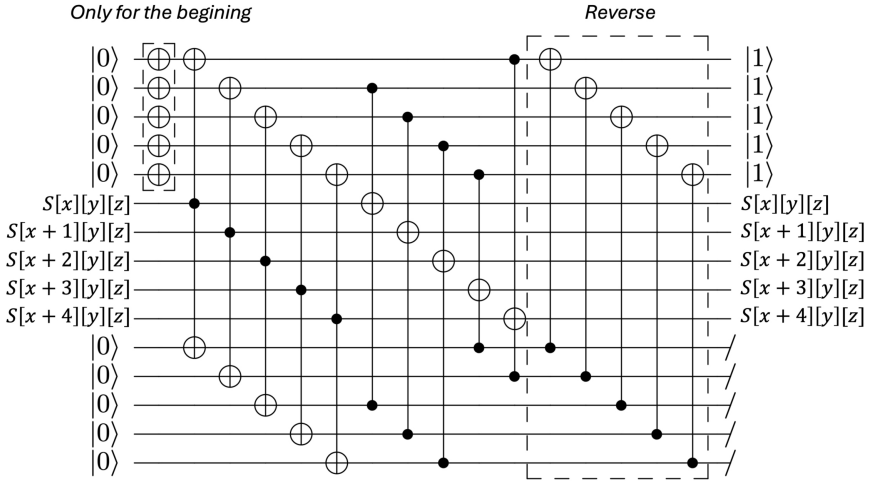


Figure 5.2: Quantum circuit diagram of χ.

Our χ circuit is constructed so that all Toffoli gates run in parallel, yielding a Toffoli depth of one. Following our first design principle, we allocate 3200 ancilla qubits to create two copies of the state S. These copies are generated using 3200 CNOT gates, and one of them is inverted using an additional set of 1600 X gates, corresponding to the negation term in χ. This enables us to independently prepare all operands needed for χ, leading to a χ implementation with depth one and 1600 parallel Toffoli operations.

We provide a χ circuit that achieves minimal depth by allowing extra ancilla qubits. Nevertheless, as emphasized earlier, our design philosophy

seeks to keep depth small while simultaneously reducing the overall qubit count through the reuse of ancilla qubits, consistent with our second design principle.

5.4.2.1 Reusing ancilla qubits.

One of the duplicated states of S can be readily reset or reused by applying the reverse of the copying operation (i.e., another 1600 CNOT gates performed after the forward step; see Figure 5.2). The ancilla qubits corresponding to the other copy (bottom lines in Figure 5.2) will also be reset/reused within our new architecture, which is explained in Section 5.4.3.

Table 5.4 summarizes the quantum resources for implementing χ (including χ^{-1}) and compares them with prior works [SJS23, ADMG+17, LKL+24]. Although those references do not explicitly describe their circuit constructions, it is likely that several ideas similar to those used in our χ implementation were employed.

Method	#CNOT	#1qCliff	#T	Toffoli depth	#Qubit (reuse)	Full depth
[ADMG+17]	33280	14400	24640	11	3200 (1600)	121
[SJS23]	10240	6400	11200	5	2240	47
[LKL+24]	15680	4480	11840	7	1600	62
Ours	14400	6400	11200	1	4800 (3200)	10

Table 5.4: The required quantum resources for χ in SHA-3.

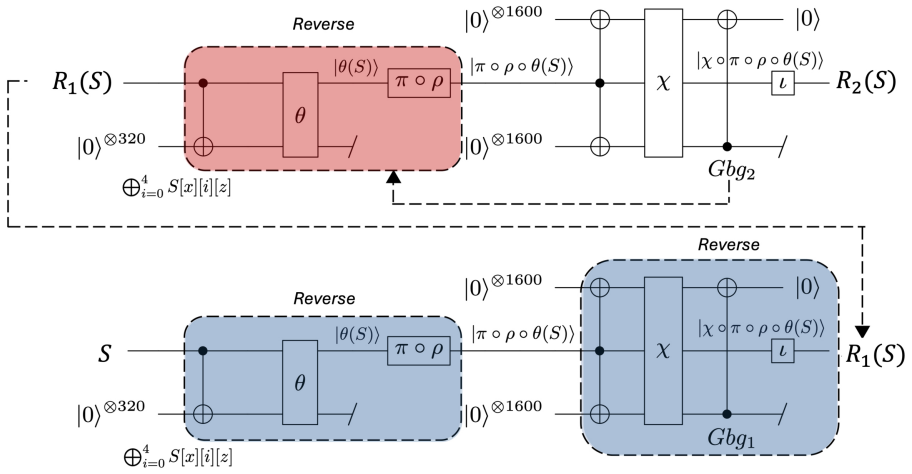
5.4.3 Interval Architecture

We propose an *interval architecture* that enables extensive reuse of ancilla qubits during the SHA-3 round operations. This architecture periodically applies reverse operations at fixed intervals while the quantum circuit progresses, as illustrated in Figure 5.3. Although the idea is broadly applicable, in our SHA-3 design the most effective interval is four rounds. By applying this interval strategy, we are able to reuse a

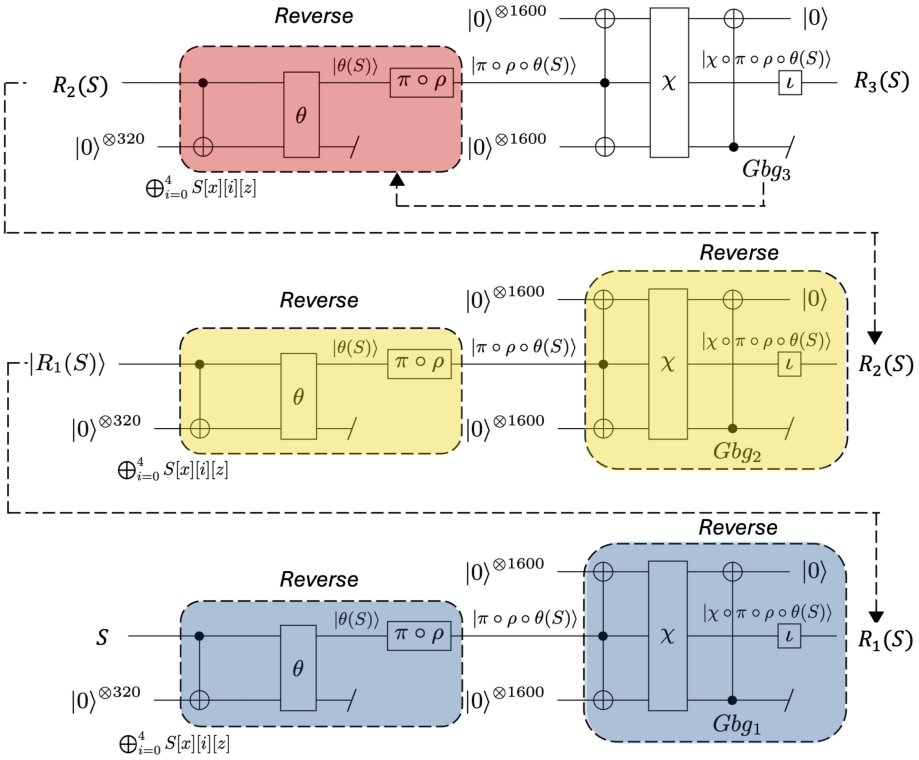
large portion of ancilla qubits and thereby reduce the total qubit count of the SHA-3 implementation by 26800, all without increasing the circuit depth.

Recall the out-of-place round implementation introduced in Section 5.4.2. In that approach, two copies are prepared for the parallel execution of χ, although only one of them is initialized while the other is discarded as garbage. We denote the garbage qubits produced in round ras $Gbg(r)$.

However, as illustrated in Figure 5.3(a), if the out-of-place architecture is viewed across two consecutive rounds rather than just one, the reuse of ancilla qubits becomes possible. Specifically, the qubits Gbg_1 from Round 1 can be initialized during Round 2 by using Gbg_2. Executing the reverse sequence $\pi \circ \rho \circ \theta$ (the red region in Figure 5.3(a)) on Gbg_2 restores its state, enabling the ancilla qubits for θ in Round 2 to be initialized through Gbg_2. Because the state of Gbg_2 transitions to $R_1(S)$ after the reverse step, we can then run the reverse operations of Round 1 (the blue regions in Figure 5.3(a)) using Gbg_2 as well. Consequently, all ancilla qubits used in Round 1 (3520 = 1600 + 1600 + 320) are cleaned and reusable in the following rounds.



(a) 2-round interval



(b) 3-round interval

Figure 5.3: Interval architecture for SHA-3

From this two-round interval process, we extract two criteria for

determining the optimal interval length in SHA-3:

First reverse operation: only the 320 ancilla qubits used in θ can be reset.

Final reverse operation: all 3520 ancilla qubits (1600 + 1600 + 320) can be reset.

If the interval extends to three rounds, as shown in Figure 5.3(b), the first and last reverse operations function similarly, except that Gbg_3 replaces Gbg_2. For the middle round (i.e., Round 2; yellow regions in Figure 5.3(b)), all ancilla qubits are also reset. Importantly, the initialized copy of the χ input in Round 2 (top lane) must be reused until the last reverse operation. Thus, strictly speaking, only Gbg_2 and the θ ancilla qubits for Round 2 are actually reinitialized. From this, we determine:

Middle reverse operation: 1920 ancilla qubits (1600 + 320) can be reset.

In the SHA-3 implementation, the optimal interval for the reverse-operation schedule is found to be four rounds. A detailed justification is provided in the following subsection.

5.4.3.1 Shallow Technique

In previous works such as [GLRS16, JNRV20, ZWS+20, ADMG+17], quantum circuits delay forward progress until reverse rounds complete, enabling ancilla reuse but increasing circuit depth.

In contrast, our method performs forward and reverse rounds simultaneously without waiting for the reverse stage to finish. To support this, we allocate a sufficient number of ancilla qubits only during the initial stage. This idea, originally employed in [JBK+25] for AES (referred to as the shallow architecture), is incorporated into our interval design.

The key idea is to start a new instance when the reverse phase begins, which corresponds to Rounds n for interval n. After the first

stage, the ancilla qubits recovered from reverse operations can be reused immediately, without increasing circuit depth.

Let the interval be n rounds. Reverse operations begin at the end of Round n, and the new instance of Round $n+1$ starts at the same time. As the round-n reverse has just started, all ancilla qubits must be freshly allocated for Round $n+1$. Specifically, 320 qubits for θ and 3200 qubits for χ are needed, totaling 3520 qubits at this stage. From Round $n+2$, however, ancilla qubits recovered from reverse operations become available and are reused.

Through this analysis, the optimal interval is determined to be $n=4$ requiring 20800 ancilla qubits.

5.4.3.2 Consideration of AND Gates

For χ, 1600 AND gates require 1600 additional ancilla qubits. These ancilla qubits are initialized once after the AND operations and reused afterward. The target qubits of the AND gates, however, are not clean due to the gate structure. To resolve this without affecting performance, we apply a minor circuit modification described in Figure 5.2. We postpone the copying of the bottom lines and redirect AND targets from the middle lines to the bottom lines. Then we perform the postponed copying step. Lastly, we use the bottom lines to hold the result and execute reverse operations on the middle lines.

5.4.4 Quantum Implementation of ρ, π, and ι

In SHA-3, the ρ and π steps perform bit rotations and permutations. Following the same strategy used for SHA-2, these operations are implemented through logical swaps and therefore require no additional quantum resources.

The ι step, which XORs the round constant into the state S, belongs to the class of classical-quantum operations. Since the round constants

are known in advance, the required modifications can be applied using only X gates at the positions where the constant has a value of 1. This yields a straightforward implementation of ι using solely X gates, consistent with standard practices in quantum realizations of SHA-3 [ADMG+17, JBK+25, SJS23].

5.5 Results

In this section, we compare our quantum circuit designs for SHA-2 and SHA-3 against prior works [ADMG+17, KHJ18, LLLC23, MSDM22, HS20, SJS23, LKL+24]. Tables 5.5 and 5.6 summarize the quantum resources required for our SHA-2 and SHA-3 circuits, using the Toffoli-based and AND-based implementations, respectively.

5.5.1 SHA-2 Quantum Circuits

Most prior studies focused solely on implementing the quantum circuit for SHA-2-256. In contrast, we extend our implementations to cover SHA-2-384 and SHA-2-512 as well. Our SHA-2 circuits achieve the lowest Toffoli depth and full depth among existing works. Although the full depth was not provided in [LLLC23], the T-depth optimization technique introduced there increases the full depth as a trade-off. While our TD-M is slightly weaker than the trade-off metric in [LLLC23], our design achieves improvements in key indicators for Grover's parallelization (TD^2-M, Td^2-M, and FD^2-M) through depth-oriented optimization.

Cipher	Method	#CNOT	#1qCliff	$\#T$	Toffoli depth (TD)	#Qubit (M)	Full depth (FD)	TD-M	FD-M	TD^2-M	FD^2-M
SHA-2-256	[ADMG+17]	534272	515952	401584	57184	2402	528768	$1.02 \cdot 2^{27}$	$1.18 \cdot 2^{30}$	$1.79 \cdot 2^{42}$	$1.19 \cdot 2^{49}$
	[ADMG+17] (Opt.)	4209072	173264	228992	57184	2402	830720	$1.02 \cdot 2^{27}$	$1.86 \cdot 2^{30}$	$1.79 \cdot 2^{42}$	$1.47 \cdot 2^{50}$
	[KHJ18]	-	-	-	10112	938	-	$1.13 \cdot 2^{23}$	-	$1.40 \cdot 2^{36}$	-
	[LLLC23]	-	-	-	4418	962	-	$1.01 \cdot 2^{22}$	-	$1.09 \cdot 2^{34}$	-
	[HS20]	-	-	90292	1607	23684	-	$1.13 \cdot 2^{25}$	-	$1.78 \cdot 2^{35}$	-
	[MSDM22]	-	-	90292	1607	23957	-	$1.15 \cdot 2^{25}$	-	$1.80 \cdot 2^{35}$	-
	This work	693832	84086	495089	**1332**	5715	**12791**	$1.81 \cdot 2^{22}$	$\mathbf{1.09 \cdot 2^{26}}$	$\mathbf{1.18 \cdot 2^{33}}$	$\mathbf{1.70 \cdot 2^{39}}$
SHA-2-384	This work	1847124	225008	1335511	**1824**	13773	**17257**	$1.50 \cdot 2^{24}$	$\mathbf{1.77 \cdot 2^{27}}$	$\mathbf{1.33 \cdot 2^{35}}$	$\mathbf{1.87 \cdot 2^{41}}$
SHA-2-512	[HS20]	-	-	231788	3,303	60448	-	$1.49 \cdot 2^{27}$	-	$1.20 \cdot 2^{39}$	-
	[MSDM22]	-	-	231788	3304	59995	-	$1.48 \cdot 2^{27}$	-	$1.19 \cdot 2^{39}$	-
	Ours	1864872	226533	1346011	**1828**	13901	**17303**	$1.51 \cdot 2^{24}$	$\mathbf{1.79 \cdot 2^{27}}$	$\mathbf{1.35 \cdot 2^{35}}$	$\mathbf{1.89 \cdot 2^{41}}$

Table 5.5: The required quantum resources for SHA-2 quantum circuits.

Method	#CNOT	#1qCliff	$\#T$	Toffoli depth (TD)	#Qubit (M)	Full depth (FD)	TD-M	FD-M	TD^2-M	FD^2-M
[ADMG+17]	33269760	169045	591360	264	3200	10128	$1.61 \cdot 2^{19}$	$1.93 \cdot 2^{24}$	$1.66 \cdot 2^{27}$	$1.19 \cdot 2^{38}$
	34260480	215125	499200	264	3200	11040	$1.61 \cdot 2^{19}$	$1.05 \cdot 2^{25}$	$1.66 \cdot 2^{27}$	$1.42 \cdot 2^{38}$
[HS20]	·	·	153600	24	46400	·	$1.06 \cdot 2^{20}$	·	$1.59 \cdot 2^{24}$	·
[MSDM22]	·	·	153600	24	44798	·	$1.03 \cdot 2^{20}$	·	$1.54 \cdot 2^{24}$	·
[SJS23]	821760	153688	268800	120	55360	2860	$1.58 \cdot 2^{22}$	$1.18 \cdot 2^{27}$	$1.48 \cdot 2^{29}$	$1.65 \cdot 2^{38}$
[LKL+24]	·	·	153600	24	43200	·	$1.98 \cdot 2^{19}$	·	$1.48 \cdot 2^{24}$	·
	·	·	284160	168	1600	·	$1.03 \cdot 2^{18}$	·	$1.35 \cdot 2^{25}$	·
This work	752000	124937	425600	**24**	22400	**578**	$1.03 \cdot 2^{19}$	$\mathbf{1.54 \cdot 2^{23}}$	$\mathbf{1.54 \cdot 2^{23}}$	$\mathbf{1.74 \cdot 2^{32}}$

Table 5.6: The required quantum resources for SHA-3 quantum circuits.

5.5.2 SHA-3 Quantum Circuits

Our SHA-3 quantum circuits deliver the strongest overall performance in terms of depth and depth-qubit trade-offs, except for the Toffoli-depth-qubit and T-depth-qubit products (TD-M and Td-M), where [LKL+24] reports the lowest values. However, for the primary metrics relevant to Grover's parallelization, TD^2-M, Td^2-M, and FD^2-M, our circuits maintain the best performance.

In [LKL+24], the main goal was to minimize qubit count (and they additionally presented a Z0 variant for out-of-place implementation). Our work instead focuses on minimizing circuit depth. For comparison, we use their Z1 version, which has the smallest qubit count and TD-M among their in-place designs (Z1, ..., Z5). Although their qubit usage is lower, our circuits achieve smaller Toffoli and full depths. While [LKL+24] does not report full depth, we expect our full-depth-qubit product (FD-M) to be lower, since their Toffoli depth reduction technique increases both CNOT count and full depth.

5.5.3 Quantum Collision Search for SHA-2 and SHA-3

Using the SHA-2 and SHA-3 quantum circuits developed in this work, we estimate the quantum resources required for collision search using the CNS algorithm. For an n-bit output of SHA-2 or SHA-3, the

entire 2^n input space must be explored. According to the CNS algorithm [CNPS17], Grover's circuit finds a collision with a query complexity of $2^{(2n/5-3s/5)}$, where $s=n/6$ based on our earlier analysis in Section 5.2.1. The cost of the diffusion operator is typically omitted in resource estimation [JBK+25, LPZW23, GLRS16, JNRV20], and we follow the same practice.

Thus, the overall complexity of quantum collision search for SHA-2 and SHA-3 is approximately (cost of Grover's oracle) × $2^{(2n/5-3s/5)}$, where n is the hash output size and $s=n/6$.

Tables 5.7 and 5.8 present the required quantum resources for collision search on SHA-2 and SHA-3. These include the primary metrics for evaluation, such as the full depth and gate count product (G-FD) representing quantum attack cost, and other trade-off metrics (FD-M, Td-M, FD^2-M, Td^2-M). Both Toffoli-based and AND-based decompositions are used to estimate the total quantum resource requirements.

Cipher	#Gate (G)	Full depth (FD)	T-depth (Td)	#Qubit (M)	G-FD	FD-M	Td-M	Fd^2-M	Td^2-M
SHA-2-256	$1.49 \cdot 2^{97}$	$1.06 \cdot 2^{91}$	$1.77 \cdot 2^{89}$	$1.1 \cdot 2^{55}$	$\mathbf{1.77 \cdot 2^{188}}$	$1.18 \cdot 2^{146}$	$1.97 \cdot 2^{144}$	$1.26 \cdot 2^{237}$	$1.75 \cdot 2^{234}$
SHA-2-384	$1.47 \cdot 2^{137}$	$1.9 \cdot 2^{129}$	$1.6 \cdot 2^{128}$	$1.68 \cdot 2^{77}$	$\mathbf{1.39 \cdot 2^{267}}$	$1.59 \cdot 2^{207}$	$1.35 \cdot 2^{206}$	$1.51 \cdot 2^{337}$	$1.08 \cdot 2^{335}$
SHA-2-512	$1.95 \cdot 2^{175}$	$1.25 \cdot 2^{168}$	$1.06 \cdot 2^{167}$	$1.06 \cdot 2^{99}$	$\mathbf{1.22 \cdot 2^{344}}$	$1.34 \cdot 2^{267}$	$1.13 \cdot 2^{266}$	$1.68 \cdot 2^{435}$	$1.20 \cdot 2^{433}$
SHA-3-256	$1.70 \cdot 2^{97}$	$1.54 \cdot 2^{86}$	$1.02 \cdot 2^{84}$	$1.08 \cdot 2^{57}$	$\mathbf{1.30 \cdot 2^{184}}$	$1.67 \cdot 2^{143}$	$1.1 \cdot 2^{141}$	$1.29 \cdot 2^{230}$	$1.12 \cdot 2^{225}$
SHA-3-384	$1.12 \cdot 2^{136}$	$1.01 \cdot 2^{125}$	$1.34 \cdot 2^{122}$	$1.36 \cdot 2^{78}$	$\mathbf{1.14 \cdot 2^{261}}$	$1.39 \cdot 2^{203}$	$1.84 \cdot 2^{200}$	$1.41 \cdot 2^{328}$	$1.23 \cdot 2^{323}$
SHA-3-512	$1.48 \cdot 2^{174}$	$1.34 \cdot 2^{163}$	$1.77 \cdot 2^{160}$	$1.72 \cdot 2^{99}$	$\mathbf{1.98 \cdot 2^{337}}$	$1.15 \cdot 2^{263}$	$1.52 \cdot 2^{260}$	$1.55 \cdot 2^{426}$	$1.35 \cdot 2^{421}$

Table 5.7: The required quantum collision search on SHA-2 and SHA-3 using Toffoli-based implementation.

Cipher	#Gate (G)	Full depth (FD)	T-depth (Td)	#Qubit (M)	G-FD	FD-M	Td-M	Fd^2-M	Td^2-M
SHA-2-256	$1.49 \cdot 2^{97}$	$1.58 \cdot 2^{90}$	$1.18 \cdot 2^{87}$	$1.13 \cdot 2^{55}$	$\mathbf{1.18 \cdot 2^{188}}$	$1.81 \cdot 2^{145}$	$1.35 \cdot 2^{142}$	$1.43 \cdot 2^{236}$	$1.60 \cdot 2^{229}$
SHA-2-384	$1.32 \cdot 2^{137}$	$1.45 \cdot 2^{129}$	$1.12 \cdot 2^{126}$	$1.72 \cdot 2^{77}$	$\mathbf{1.91 \cdot 2^{266}}$	$1.25 \cdot 2^{207}$	$1.93 \cdot 2^{203}$	$1.81 \cdot 2^{336}$	$1.08 \cdot 2^{330}$
SHA-2-512	$1.76 \cdot 2^{175}$	$1.91 \cdot 2^{167}$	$1.48 \cdot 2^{164}$	$1.09 \cdot 2^{99}$	$\mathbf{1.68 \cdot 2^{343}}$	$1.05 \cdot 2^{267}$	$1.62 \cdot 2^{263}$	$1.00 \cdot 2^{435}$	$1.20 \cdot 2^{428}$
SHA-3-256	$1.31 \cdot 2^{97}$	$1.39 \cdot 2^{86}$	$1.79 \cdot 2^{81}$	$1.16 \cdot 2^{57}$	$\mathbf{1.83 \cdot 2^{183}}$	$1.62 \cdot 2^{143}$	$1.04 \cdot 2^{139}$	$1.13 \cdot 2^{230}$	$1.87 \cdot 2^{220}$
SHA-3-384	$1.73 \cdot 2^{135}$	$1.84 \cdot 2^{124}$	$1.18 \cdot 2^{120}$	$1.46 \cdot 2^{78}$	$\mathbf{1.59 \cdot 2^{260}}$	$1.35 \cdot 2^{203}$	$1.73 \cdot 2^{198}$	$1.24 \cdot 2^{328}$	$1.02 \cdot 2^{319}$
SHA-3-512	$1.14 \cdot 2^{174}$	$1.21 \cdot 2^{163}$	$1.56 \cdot 2^{158}$	$1.84 \cdot 2^{99}$	$\mathbf{1.39 \cdot 2^{337}}$	$1.12 \cdot 2^{263}$	$1.44 \cdot 2^{258}$	$1.36 \cdot 2^{426}$	$1.12 \cdot 2^{417}$

Table 5.8: The required quantum collision search on SHA-3 using AND-based implementation.

5.5.4 Update on NIST Post-Quantum Security Levels.

Table 5.9 summarizes our updated post-quantum security levels, determined by selecting the lowest quantum attack complexities for the SHA-2 and SHA-3 hash functions from Tables 5.8 and 5.9. Below, we provide a concise discussion of the rationale behind these updates.

When NIST first introduced the notion of post-quantum security levels in 2016 [NIS16], the recommended assignments, AES-128, AES-192, and AES-256 at levels 1, 3, and 5, were based on the estimates by Grassl et al. [GLRS16], which at the time predicted comparatively high quantum attack costs. Since then, several studies have substantially lowered the quantum complexity of attacking AES [JBK+25, JNRV20, LPZW23], and NIST has revised its estimates accordingly, primarily following the results of Jaques et al. [JNRV20].

For hash functions, NIST assigned SHA-2/3-256 to level 2 and SHA-2/3-384 to level 4, arguing that real-world cryptanalytic success is often maximized when highly parallel computations are possible. Under this perspective, collision search on hash functions is considered easier than key search on block ciphers (which parallelize poorly). However, the quantum complexities corresponding to levels 2 and 4, both of which relate to collision search on SHA-2 and SHA-3, have yet to be formally defined.

In this work, we establish quantum attack complexities for levels 2 and 4 (Table 5.9) fully aligned with NIST's principles and guidelines [NIS16]. Specifically, we assume that a highly parallelized quantum collision search (the CNS algorithm [BHMT02]) should naturally fall between the complexities of levels 1 and 3, and between levels 3 and 5, respectively.

Security	Hash algorithm	Complexity
Level 2	SHA-2-256	2^{188}/MAXDEPTH
	SHA-3-256	2^{183}/MAXDEPTH
Level 4	SHA-2-384	2^{266}/MAXDEPTH
	SHA-3-384	2^{260}/MAXDEPTH

Table 5.9: Newly defined bounds for the NIST post-quantum security levels (proposed in this work).

5.6 Conclusion

This work focuses on depth-optimized quantum circuit implementations for the SHA-2 and SHA-3 cryptographic hash families. Our approach systematically improves both algorithm-level components (such as the linear layer and arithmetic in the round function) and architectural aspects (including interval and shallow designs). By integrating the latest and most effective optimization techniques across all categories, our implementations achieve the lowest reported quantum resource costs for collision search on SHA-2 and SHA-3 variants. Further, we quantify NIST's post-quantum security levels 2 and 4 using our updated quantum attack complexities.

Ⅵ. Conclusion

This thesis investigated how quantum attacks fundamentally reshape the security landscape of modern cryptography. Through detailed analysis, we developed optimized quantum circuits for ECC, AES, and the SHA-2/3 families, allowing more accurate and realistic estimates of their post-quantum security.

In Chapter 3, we presented improved quantum attacks for ECC based on Shor's algorithm. For symmetric key cryptography, in Chapters 4 and 5, we designed several optimized architectures for AES and the SHA-2/3 hash functions, focusing on depth-efficient S-boxes, MixColumn operations, linear layers, and circuit architectures. These refinements enabled tighter bounds for Grover-based key search and quantum collision search, leading to updated and more precise interpretations of NIST's post-quantum security levels.

Across all parts of this work, our emphasis was on constructing practical and scalable quantum circuits that reflect realistic constraints, such as circuit depth, qubit count, and algorithmic structure (rather than relying solely on theoretical asymptotics). Through this approach, we derived lower quantum security margins than previously estimated, especially when depth-optimized circuits are taken into account.

Ultimately, this thesis highlights the importance of continually revisiting post-quantum security assumptions as quantum computing advances. We hope that the analyses, circuit optimization techniques, and updated benchmarks presented here contribute to a more accurate understanding of how to design robust and secure cryptosystems for the upcoming era of quantum computing.

References

[ADMG+17] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John Schanck. Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In Selected Areas in Cryptography - SAC 2016, pages 317-337. Springer International Publishing, 2017.

[AMM+13] Matthew Amy, Dmitri Maslov, Michele Mosca, Martin Roetteler, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 32(6):818-830, Jun 2013.

[ASAM18] Mishal Almazrooie, Azman Samsudin, Rosni Abdullah, and Kussay N. Mutter. Quantum reversible circuit of AES-128. Quantum Information Processing, 17(5):1-30, may 2018.

[Bar20] Elaine Barker. NIST special publication 800-57 part 1, revision 5, recommendation for key management: Part 1- general. NIST, Tech. Rep, page 171, 2020.

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. Fortschritte der Physik: Progress of Physics, 46(4-5):493-505, 1998.

[BBVHL20] Gustavo Banegas, Daniel J Bernstein, Iggy Van Hoof, and Tanja Lange. Concrete quantum cryptanalysis of binary elliptic curves. Cryptology ePrint Archive, 2020.

[BCC+24] Anubhab Baksi, Sumanta Chakraborty, Anupam Chattopadhyay, Matthew Chun, SK Hafizul Islam, Kyungbae Jang, Hyunji Kim, Yujin Oh, Soham Roy, Hwajeong Seo, and Siyi Wang. Quantum implementation of linear and non-linear layers. IEEE International System-on-Chip Conference (SOCC), 2024.

[BDK+21] Anubhab Baksi, Vishnu Asutosh Dasu, Banashri Karmakar, Anupam Chattopadhyay, and Takanori Isobe. Three input exclusive-or gate support for boyar-peralta's algorithm. In Avishek Adhikari, Ralf Küsters, and Bart Preneel, editors, Progress in Cryptology - INDOCRYPT 2021, Jaipur, India, December 12-15, 2021, Proceedings, volume 13143 of Lecture Notes in Computer Science, pages 141-158. Springer, 2021.

[BFI21] Subhadeep Banik, Yuki Funabiki, and Takanori Isobe. Further results on efficient implementations of block cipher linear layers. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 104(1):213-225, 2021.

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. Contemporary Mathematics, 305:53-74, 2002.

[BHT97] Gilles Brassard, Peter Hoyer, and Alain Tapp. Quantum algorithm for the collision problem. arXiv preprint quant-ph/9705002, 1997.

[BP10] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In Paola Festa, editor, Experimental Algorithms, pages 178-189, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the aes s-box. In IFIP International Information Security Conference, pages 287-298. Springer, 2012.

[CDKM08] Steven Cuccaro, Thomas Draper, Samuel Kutin, and David Moulton. A new quantum ripple-carry addition circuit. arXiv, 2008.

[CNPS17] André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In ASIACRYPT 2017, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23, pages 211-

240. Springer, 2017.

[Dan17] Marcus Dansarie. Cryptanalysis of the SoDark family of cipher algorithms. PhD thesis, Naval Postgraduate School, Dudley Knox Library, 2017.

[Dan21] Marcus Dansarie. sboxgates: A program for finding low gate count implementations of S-boxes. Journal of Open Source Software, 6(62):2946, 2021.

[DBSC19] Vishnu Asutosh Dasu, Anubhab Baksi, Sumanta Sarkar, and Anupam Chattopadhyay. LIGHTER-R: optimized reversible circuit implementation for sboxes. In 32nd IEEE International System-on-Chip Conference, SOCC 2019, Singapore, September 3-6, 2019, pages 260-265, 2019.

[DH22] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography, page 365-390. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022.

[DKRS04] Thomas G Draper, Samuel A Kutin, Eric M Rains, and Krysta M Svore. A logarithmic-depth quantum carry-lookahead adder. arXiv preprint quant-ph/0406142, 2004.

[Dra00] Thomas G Draper. Addition on a quantum computer. arXiv preprint quant-ph/0008033, 2000.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In Advances in Cryptology, pages 10-18. Springer, 1985.

[GG16] Steven D Galbraith and Pierrick Gaudry. Recent progress on the elliptic curve discrete logarithm problem. Designs, Codes and Cryptography, 78:51-72, 2016.

[GLRS16] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's algorithm to AES: Quantum resource estimates. In Tsuyoshi Takagi, editor, Post-Quantum

Cryptography, pages 29–43, Cham, 2016. Springer International Publishing.

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

[Gos98] Phil Gossett. Quantum carry-save arithmetic. arXiv preprint quant-ph/9808061, 1998.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 212–219, 1996.

[HLZ+17] Yong He, Ming-Xing Luo, E Zhang, Hong-Ke Wang, and Xiao-Feng Wang. Decompositions of n-qubit toffoli gates with linear circuit complexity. International Journal of Theoretical Physics, 56(7):2350–2361, 2017.

[HS20] Thomas Häner and Mathias Soeken. Lowering the T-depth of quantum circuits by reducing the multiplicative depth of logic networks. arXiv preprint arXiv:2006.03845, 2020.

[HS22] Zhenyu Huang and Siwei Sun. Synthesizing quantum circuits of AES with lower t-depth and less qubits. In Shweta Agrawal and Dongdai Lin, editors, Advances in Cryptology – ASIACRYPT 2022 – 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part III, volume 13793 of Lecture Notes in Computer Science, pages 614–644. Springer, 2022.

[JBK+25] Kyungbae Jang, Anubhab Baksi, Hyunji Kim, Gyeongju Song, Hwajeong Seo, and Anupam Chattopadhyay. Quantum analysis of aes. IACR Communications in Cryptology, 2(1), 2025.

[JBKK24] Yongjin Jeon, Seungjun Baek, Giyoon Kim, and Jongsung Kim. A framework for generating s-box circuits with boyar-peralta

algorithm-based heuristics, and its applications to aes, snow3g, and saturnin. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2025(1):586–631, Dec. 2024.

[JKL+23] Kyungbae Jang, Wonwoong Kim, Sejin Lim, Yeajun Kang, Yujin Yang, and Hwajeong Seo. Optimized implementation of quantum binary field multiplication with Toffoli depth one. In Information Security Applications: 23rd International Conference, WISA 2022, Jeju Island, South Korea, August 24–26, 2022, Revised Selected Papers, pages 251–264. Springer, 2023.

[JLO+25] Kyungbae Jang, Sejin Lim, Yujin Oh, Hyunjun Kim, Anubhab Baksi, Sumanta Chakraborty, and Hwajeong Seo. Quantum implementation and analysis of sha-2 and sha-3. IEEE Transactions on Emerging Topics in Computing, pages 1–15, 2025.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). International journal of information security, 1:36–63, 2001.

[JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing grover oracles for quantum key search on AES and LowMC. In Anne Canteaut and Yuval Ishai, editors, Advances in Cryptology – EUROCRYPT 2020, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II, volume 12106 of Lecture Notes in Computer Science, pages 280–310. Springer, 2020.

[JSB+25a] Kyungbae Jang, Vikas Srivastava, Anubhab Baksi, Santanu Sarkar, and Hwajeong Seo. New quantum cryptanalysis of binary elliptic curves. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2025(2):781–804, 2025.

[KHJ18] Panjin Kim, Daewan Han, and Kyung Chul Jeong. Time space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2. Quantum Information Processing, 17:1–39,

2018.

[KKKH22] Sunyeop Kim, Insung Kim, Seonggyeom Kim, and Seokhie Hong. Toffoli gate count optimized space-efficient quantum circuit for binary field multiplication. Cryptology ePrint Archive, 2022.

[KLSW17] Thorsten Kranz, Gregor Leander, Ko Stoffelen, and Friedrich Wiemer. Shorter linear straight-line programs for mds matrices. IACR Transactions on Symmetric Cryptology, 2017(4):188–211, Dec. 2017.

[KLW+25] Hyunjun Kim, Sejin Lim, Siyi Wang, Kyungbae Jang, Anubhab Baksi, Anupam Chattopadhyay, and Hwajeong Seo. Tree-based quantum carry save adder. 2025.

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

[LGQW23] Zhenqiang Li, Fei Gao, Sujuan Qin, and Qiaoyan Wen. New record in the number of qubits for a quantum implementation of aes, 2023.

[LKL+24] Jongheon Lee, Yousung Kang, You-Seok Lee, Boheung Chung, and Dooho Choi. Toffoli-depth reduction method preserving in-place quantum circuits and its application to sha3-256. Quantum Information Processing, 23(4):153, 2024.

[LLLC23] Jongheon Lee, Sokjoon Lee, You-Seok Lee, and Dooho Choi. T-depth reduction method for efficient SHA-256 quantum circuit construction. IET Information Security, 17(1):46–65, 2023.

[LPS20] Brandon Langenberg, Hai Pham, and Rainer Steinwandt. Reducing the cost of implementing the advanced encryption standard as a quantum circuit. IEEE Transactions on Quantum Engineering, 1:1–12, 01 2020.

[LPZW23] Qun Liu, Bart Preneel, Zheng Zhao, and Meiqin Wang. Improved quantum circuits for aes: Reducing the depth and the number of qubits. In International Conference on the Theory and Application of

Cryptology and Information Security, pages 67–98. Springer, 2023.
[LSL+19] Shun Li, Siwei Sun, Chaoyun Li, Zihao Wei, and Lei Hu. Constructing low-latency involutory mds matrices with lightweight circuits. IACR Transactions on Symmetric Cryptology, 2019(1):84–117, Mar. 2019.
[LWF+22] Qun Liu, Weijia Wang, Yanhong Fan, Lixuan Wu, Ling Sun, and Meiqin Wang. Towards low-latency implementation of linear layers. IACR Transactions on Symmetric Cryptology, 2022(1):158–182, Mar. 2022.
[LWS+22] Qun Liu, Weijia Wang, Ling Sun, Yanhong Fan, Lixuan Wu, and Meiqin Wang. More inputs makes difference: Implementations of linear layers using gates with more than two inputs. IACR Transactions on Symmetric Cryptology, 2022(2):351–378, Jun. 2022.
[LXX+23] Da Lin, Zejun Xiang, Runqing Xu, Shasha Zhang, and Xiangyong Zeng. Optimized quantum implementation of aes. Quantum Information Processing, 22(9):352, 2023.
[LXZZ21] Da Lin, Zejun Xiang, Xiangyong Zeng, and Shasha Zhang. A framework to optimize implementations of matrices. In Kenneth G. Paterson, editor, Topics in Cryptology – CT-RSA 2021 – Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17–20, 2021, Proceedings, volume 12704 of Lecture Notes in Computer Science, pages 609–632. Springer, 2021.
[LYLL22] Qing-bin Luo, Guo-wu Yang, Xiao-yu Li, and Qiang Li. Quantum reversible circuits for GF(2^8) multiplicative inverse. EPJ Quantum Technology, 2022.
[LZW23] Qun Liu, Zheng Zhao, and Meiqin Wang. Improved heuristics for low-latency implementations of linear layers. In Cryptographers' Track at the RSA Conference, pages 524–550. Springer, 2023.
[Max19] Alexander Maximov. AES MixColumn with 92 XOR gates.

Cryptology ePrint Archive, Report 2019/833, 2019.
[Mil85] Victor S Miller. Use of elliptic curves in cryptography. In Conference on the theory and application of cryptographic techniques, pages 417–426. Springer, 1985.
[MSDM22] Giulia Meuli, Mathias Soeken, and Giovanni De Micheli. XOR-and-inverter graphs for quantum compilation. npj Quantum Information, 8(1):7, 2022.
[NIS16] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, 2016.
[NIS22] NIST. Call for additional digital signature schemes for the post-quantum cryptography standardization process, 2022.
[PD24] Meltem Kurt Pehlivanoğlu and Mehmet Ali Demir. Optimizing implementations of linear layers using two and higher input xor gates. PeerJ Computer Science, 10:e1820, 2024.
[Per19] Simone Perriello. Design and development of a quantum circuit to solve the Information Set Decoding problem. PhD thesis, Politecnico di Milano, Scuola di Ingegneria Industriale e dell'Informazione, 2019.
[PWLK22] Dedy Septono Catur Putranto, Rini Wisnu Wardhani, Harashta Tatimma Larasati, and Howon Kim. Another concrete quantum cryptanalysis of binary elliptic curves. Cryptology ePrint Archive, 2022.
[RBC23] Soham Roy, Anubhab Baksi, and Anupam Chattopadhyay. Quantum implementation of ascon linear layer. Cryptology ePrint Archive, Paper 2023/617, 2023.
[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In ASIACRYPT 2017, Hong Kong, China, December 3–7, 2017, Proceedings, Part II 23, pages 241–270. Springer, 2017.
[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Commun.

ACM, 21(2):120–126, February 1978.

[Sel13] Peter Selinger. Quantum circuits of t-depth one. Physical Review A, 87(4):042302, 2013.

[SF24b] Haotian Shi and Xiutao Feng. Quantum circuits of aes with a low-depth linear layer and a new structure. In International Conference on the Theory and Application of Cryptology and Information Security, pages 358–395. Springer, 2024.

[SFX23] Haotian Shi, Xiutao Feng, and Shengyuan Xu. A framework with improved heuristics to optimize low-latency implementations of linear layers. IACR Transactions on Symmetric Cryptology, 2023(4):489–510, Dec. 2023.

[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.

[SJS23] Gyeongju Song, Kyungbae Jang, and Hwajeong Seo. Improved low-depth SHA3 quantum circuit for fault-tolerant quantum computers. Applied Sciences, 13(6):3558, 2023.

[TP19] Quan Quan Tan and Thomas Peyrin. Improved heuristics for short linear programs. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1):203–230, Nov. 2019.

[TT23] Ren Taguchi and Atsushi Takayasu. Concrete quantum cryptanalysis of binary elliptic curves via addition chain. In Cryptographers' Track at the RSA Conference, pages 57–83. Springer, 2023.

[TTK09] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. Quantum addition circuits and unbounded fan-out. arXiv preprint arXiv:0910.2530, 2009.

[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.

[WR14] Nathan Wiebe and Martin Roetteler. Quantum arithmetic and numerical analysis using repeat-until-success circuits, 2014.

[WWL22] Ze-Guo Wang, Shi-Jie Wei, and Gui-Lu Long. A quantum circuit design of aes requiring fewer quantum qubits and gate operations. Frontiers of Physics, 17(4):1－7, 2022.

[XZL+20] Zejun Xiang, Xiangyong Zeng, Da Lin, Zhenzhen Bao, and Shasha Zhang. Optimizing implementations of linear layers. IACR Trans. Symmetric Cryptol., 2020(2):120－145, 2020.

[YWS+24] Yufei Yuan, Wenling Wu, Tairong Shi, Lei Zhang, and Yu Zhang. A framework to improve the implementations of linear layers. IACR Transactions on Symmetric Cryptology, 2024(2):322－347, Jun. 2024.

[ZH22] Chengkai Zhu and Zhenyu Huang. Optimizing the depth of quantum implementations of linear layers. In Inscrypt 2022, Beijing, China, December 11－13, 2022, volume 13837 of Lecture Notes in Computer Science, pages 129－147. Springer, 2022.

[ZLD+19] Jian Zou, Yongyang Liu, Chen Dong, Wenling Wu, and Le Dong. Observations on the quantum circuit of the SBox of AES. Cryptology ePrint Archive, Report 2019/1245, 2019.

[ZWS+20] Jian Zou, Zihao Wei, Siwei Sun, Ximeng Liu, and Wenling Wu. Quantum circuit implementations of AES with fewer qubits. In Shiho Moriai and Huaxiong Wang, editors, Advances in Cryptology – ASIACRYPT 2020, pages 697－726, Cham, 2020. Springer International Publishing.

국 문 초 록

암호 알고리즘에 대한 고도화된 양자 공격 연구
ECC, AES, SHA-2/3에 대한 분석

한 성 대 학 교 대 학 원
정 보 컴 퓨 터 공 학 과
정 보 시 스 템 공 학 전 공
장 경 배

양자 컴퓨터는 향후 과학 기술 분야에서의 가장 큰 도약 중 하나가 될 것으로 사료된다. 아직 대규모의 완전한 양자컴퓨터는 개발되지 않았지만, 강력한 양자 공격자를 가정한 암호시스템의 보안성 평가는 안전한 양자내성암호 시스템 구축을 위해 필수적이다.

본 학위논문은 현대 암호시스템을 대상으로 한 고도화된 양자 공격 기법을 탐구하며, 특히 타원곡선암호, AES 대칭키 암호, SHA-2/3 해시 함수에 초점을 맞춘다.

Shor 알고리즘을 동작시킬 수 있는 대규모의 양자컴퓨터가 개발된다면 ECC의 안전성은 무너지게된다. 본 논문에서는 이러한 관점에서 ECC에 대해 최적화된 양자 암호 분석 기법을 제시한다. 또한 본 연구는 미국 NIST에서 정의한 양자 후 보안 등급에 대한 추정치를 제공한다. 보안 레벨 1, 3, 5는 AES-128/192/256에 대한 Grover 기반 키 검색 복잡도에, 레벨 2와 4는 SHA-2/3에 대한 양자 충돌 쌍 탐색 복잡도에 대응되며 이에 대한 새로운

추정치를 제시한다.

본 학위논문은 양자 암호 분석을 위한 양자 회로의 설계와 최적화에 기여하며, 현실적인 규모의 양자컴퓨팅이 점점 가까워지는 상황에서 현대 암호시스템의 양자 후 보안성 평가에 대한 중요성을 강조한다.

주요어: 양자 컴퓨터, AES, SHA-2/3, ECC