Quantum circuit for implementing AES S-box with low costs

Huinan Chen[1], Binbin Cai[1,2,*], Fei Gao[3,†], Song Lin[1,‡]

[1]College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China

[2]Digital Fujian Internet-of-Things Laboratory of Environmental Monitoring, Fujian Normal University, Fuzhou 350117, China

[3]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

March 11, 2025

Abstract

Advanced Encryption Standard (AES) is one of the most widely used and extensively studied encryption algorithms globally, which is renowned for its efficiency and robust resistance to attacks. In this paper, three quantum circuits are designed to implement the S-box, which is the sole nonlinear component in AES. By incorporating a linear key schedule, we achieve a quantum circuit for implementing AES with the minimum number of qubits used. As a consequence, only 264/328/398 qubits are needed to implement the quantum circuits for AES-128/192/256. Furthermore, through quantum circuits of the S-box and key schedule, the overall size of the quantum circuit required for Grover's algorithm to attack AES is significantly decreased. This enhancement improves both the security and resource efficiency of AES in a quantum computing environment.

1 Introduction

In recent years, the deep integration of quantum mechanics and information science has driven the rapid advancement of quantum information [1, 2], positioning it as a frontier field in scientific research and technological innovation.

*cbb@fjnu.edu.cn

†gaof@bupt.edu.cn

‡lins95@fjnu.edu.cn

Unlike traditional computers that rely on binary bit states, quantum computers leverage the principles of superposition and entanglement, allowing qubits to process multiple states simultaneously and achieve unprecedented parallel computing capabilities [3, 4, 5, 6]. The foundational concepts of quantum superposition, entanglement, and measurement not only endow quantum computing with distinct computational advantages, but also equip it with the power to significantly accelerate the speed of solving specific problems, achieving exponential or super-linear speedups in specific scenarios [2, 7, 8, 9, 10, 11].

The acceleration capabilities of quantum computing in solving specific problems have had a profound impact on classical cryptography. As a result, quantum cryptography has emerged as a vital field for ensuring information security [1, 12, 13]. Grounded in the fundamental principles of quantum mechanics, quantum cryptography provides innovative solutions for secure communication. For instance, Quantum Key Distribution (QKD) [14, 15, 16, 17]leverages the inherent unclonability of quantum states to guarantee the absolute security of key transmission [18, 19], while enabling real-time detection of eavesdropping. This unique characteristic positions QKD as a pivotal technology for achieving unconditionally secure communication in future information networks. Simultaneously, quantum computing poses substantial challenges to traditional cryptographic systems. As advancements in quantum computing hardware accelerate, the minimum resources required to execute quantum cryptanalysis algorithms are becoming critical in determining the timeline for realizing quantum threats, such as Shor's algorithm [20, 21], Grover's algorithm [22, 23], and Simon's algorithm [24, 25]. This urgency has prompted the cryptographic community to accelerate efforts in developing technical solutions to counter quantum attacks, with the goal of mitigating or preventing the potential risks posed by quantum computing.

Grover's algorithm [22] is a fundamental quantum computing algorithm for unordered database searches, with its notable feature being a quadratic speedup. In classical computing, finding a target item in an unordered list of size requires an average of queries, whereas Grover's algorithm reduces this to by leveraging quantum superposition and interference. While this speedup is not exponential, it provides significant efficiency gains for large-scale search problems. Grover's algorithm has a more immediate impact on symmetric encryption schemes than on public-key cryptosystems [26, 27]. For instance, breaking a symmetric encryption algorithm with a key length of bits in the classical context requires attempts. However, Grover's algorithm reduces this complexity to , increasing

the vulnerability of symmetric encryption algorithms in a quantum computing environment. To counter this threat, the cryptographic community has proposed doubling the key length of symmetric encryption schemes to restore their security [28, 29]. Additionally, researchers have examined the resource requirements of Grover's algorithm, including the number of qubits, quantum gates, and circuit depth. These studies provide a quantitative framework for evaluating the actual risks quantum computing poses to symmetric encryption. As quantum computing technology progresses, the applications of Grover's algorithm extend beyond encryption cracking to combinatorial optimization problems. Investigating its implications for existing encryption standards and exploring new quantum-resistant algorithm designs has become a critical research direction in the field of quantum security.

AES [30] was developed to address the growing demand for more secure and efficient encryption methods. With the rapid increase in computing power during the late 20th century, the Data Encryption Standard (DES) [31] began to reveal significant limitations, including inadequate key length and limited resistance to attacks. In response, the National Institute of Standards and Technology (NIST) [32] launched an open competition in 1997 to identify a replacement. After rigorous evaluation, the Rijndael algorithm [33, 34], created by Belgian cryptographers Joan Daemen and Vincent Rijmen, was chosen in 2001 as the AES standard due to its high efficiency and robust resistance of attack. These features significantly enhance data diffusion and resistance to brute-force attacks. Its structural design, based on the Substitution-Permutation Network (SPN), improves encryption efficiency while strengthening resistance to differential and linear cryptanalysis. In comparison to the Feistel structure used in DES, AES achieves higher operational efficiency in both hardware and software implementations [34, 35, 36]. Serving as a cornerstone of modern information security, AES finds broad application in applications including wireless communication, network transmission, storage encryption, and the protection of government-sensitive data. Its flexibility and resilience serve as a reliable basis for ensuring information security in a wide range of sensitive scenarios.

Optimizing the quantum resources required by Grover's algorithm to attack the AES algorithm is of critical importance. On one hand, AES is one of the most extensively studied symmetric cryptographic algorithms globally. On the other hand, the quantum resources needed for Grover's algorithm to break AES served as a benchmark for evaluating the security strength of post-quantum cryptographic algorithms in NIST's 2016 post-quantum standardization call [37,

38]. Developing optimized quantum circuits for AES is essential for assessing the quantum resources required by Grover's algorithm in such attacks. Among the key factors influencing this process, implementing the AES S-box with minimal resources is crucial to enhancing the efficiency of quantum circuit designs.

The width of quantum circuits (i.e., the number of qubits) that can be efficiently implemented in existing quantum computers is limited, highlighting the importance to optimize the number of qubits required for implementing AES quantum circuits. In 2016, Grassl et al. [39] designed an AES S-box quantum circuit with a width of 40, using a zig-zag approach. They constructed an AES-128 quantum circuit with a final circuit width of 984. In 2018, Almazrooie et al. [40] optimized the key expansion strategy, further reducing the width of the AES-128 quantum circuit to 976. In 2020, Langenberg et al. [41] proposed an AES S-box quantum circuit with a width of 32, incorporating a zig-zag structure in the key expansion, which leading to a reduction in the AES-128 circuit width to 864. In the same year, Zou et al. [42] designed an AES S-box circuit with a width of 22 and combined it with an improved zig-zag method with a width of 23, leading to an AES-128 quantum circuit implementation with a width of 512. In 2022, Wang et al. [43] presented a straight-line iterative method for key expansion, reducing the AES-128 circuit width to 400. Later that year, Li et al. [44] further minimized the width of quantum circuits for AES-128, bringing it down to 270 by constructing AES S-box circuits with a width of 22 and applying a streamlining optimization strategy.

Although the circuits width and gate depth required to implement AES quantum circuits have been significantly reduced, further optimization is still possible. In this paper, we optimize the S-box quantum circuits for AES by minimizing the circuit width while substantially reducing quantum gates usage. As a result, we design quantum circuits for AES-128, AES-192 and AES-256 with minimal widths of 264, 328 and 392, respectively.

2 Preliminary

In this section, we provide an overview of the AES block cipher, composite field arithmetic, Grover's algorithm, and basic quantum gates.

2.1 The AES block cipher

The AES uses a 128-bit block size, with key lengths of 128, 192, and 256 bits, referred to as AES-128, AES-192, and AES-256, respectively. The number of

encryption rounds N_R for the algorithm is 10, 12, and 14 for each key length. At the beginning of encryption, the plaintext M is obtained through one AddRoundKey transformation, and then after N_R rounds of round function, the ciphertext C is obtained. Each round function operation consists of 4 subroutines, like SubBytes, ShiftRows, MixColumns and AddRoundKey. The final round function differs in that it contains only 3 operations, like SubBytes, ShiftRows and AddRoundKey.

2.1.1 The S-box

SubBytes is the sole nonlinear operation in AES, and the S-box used for this operation is implemented using a 8×8 lookup table, providing resistance to linear and differential attacks during encryption.

SubBytes operation, which is denoted as $S(a)$, takes a byte $a \in F(2^8)$ as input, where $a = (a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0)$ can be represented by the polynomial $(a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0)$ with coefficients $\{0, 1\}$. We first compute the multiplicative inverse of a, denoted as a^{-1}, then apply an affine transformation involving matrix multiplication and modulo addition with a constant vector. The algebraic form of $S(a)$ is expressed as

$$S(a) = Aa^{-1} \oplus c, \tag{1}$$

where the matric A and vector c are defined as

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad c = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

2.1.2 The key expansion process

AES-128, AES-192, and AES-256 have different key expansion processes due to their varying initial key lengths. Considering 32 bits as a word, the initial key K_0 can be divided into N_K words for AES-128, AES-192 and AES-256 , where N_K equals 4, 6 and 8. For example, the initial key $K_0 = W_3W_2W_1W_0$ of AES-128, after N_R round key expansion process, produces $(N_R + 1)$ subkeys. The

subkey for the i-th round is denoted as K_i, where $0 \leq i \leq N_R$. The key expansion process for AES-128 and AES-192 is shown in Algorithm 1, while the key expansion process for AES-256 is shown in Algorithm 2. In these algorithms, $RotWord$ represents a circular left shift of a word by 8 bits, functioning similarly to $ShiftRows$, and $Rcon[j] = (RC[j], 00, 00, 00)$ is denoted that XOR the current state with the round constant, where $0 \leq j < 8$. The bytes in $Rcon[j]$ are represented in hexadecimal, and $RC[j]$ is the element with value x^{j-1} of $F(2^8)$.

Algorithm 1: The key expansion process for AES-128 and AES-192

For t from N_K to $4 * (N_R + 1)$

If $t == 0 \ mod \ 4$, then

$W_t = W_{t-4} \oplus SubBytes(RotWord(W_{t-1})) \oplus Rcon[t/4]$;

Otherwise, $W_t = W_{t-4} \oplus W_{t-1}$.

Algorithm 2: The key expansion process for AES-256

For t from N_K to $4 * (N_R + 1)$

If $t == 0 \ mod \ 8$, then

$W_t = W_{t-8} \oplus SubBytes(RotWord(W_{t-1})) \oplus Rcon[t/8]$;

If $t == 4 \ mod \ 8$, then

$$W_t = W_{t-8} \oplus SubBytes(RotWord(W_{t-1}));$$

Otherwise, $W_t = W_{t-8} \oplus W_{t-1}$.

The subkey for the i-th round of AES-128, AES-192, and AES-256 are all denoted as $K_i = W_{4i+3}W_{4i+2}W_{4i+1}W_{4i}$, where $0 \leq i \leq N_R$. Each subkey is 128 bits in length.

2.2 Composite field arithmetic

In combinatorial domain operations, elements in higher-order domains can be expressed as linear combinations of elements in lower-order domains, where operations in lower-order domains are simpler and less costly. Therefore, solving the multiplicative inverse of any element a in a finite domain can be transformed into an element in a combinatorial domain using a mapping matrix. After obtaining the multiplicative inverse of the corresponding element in the combinatorial domain, the multiplicative inverse of a can be derived using the mapping matrix,

resulting in a^{-1}.

Wolkerstorfer et al. [45] proposed an implementation of the AES's S-box using the composite fields $F((2^4)^2)$ and $F(2^4)$, i.e. $\begin{cases} F((2^4)^2): & x^2+x+\lambda \\ F(2^4): & x^4+x+1 \end{cases}$, where

$$\lambda = x^3 + x^2 + x \in F(2^4). \tag{2}$$

The mapping matrix $M: F(2^8) \to F((2^4)^2)$ and the inverse matrix $M^{-1}: F((2^4)^2) \to F(2^8)$ are defined as

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}, \; M^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

The algebraic expression of Eq.1 can be rewritten as

$$S(x) = AM^{-1}(Ma)^{-1} \oplus c, \tag{3}$$

where

$$AM^{-1} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

From Eq.3, it can be seen that $S(a)$ still requires affine transformations, but now the multiplicative inverse is computed for the element in $F((2^4)^2)$, rather than in $F(2^8)$.

Any element in $F((2^4)^2)$ can be expressed as $p = p_0 + p_1 x$, where $p_0,\ p_1 \in F(2^4)$. The multiplicative inverse of p can be found, and p^{-1} can be expressed as

$$p^{-1} = (p^{17})^{-1}(p_0 + p_1) + (p^{17})^{-1} p_1 x, \tag{4}$$

where

$$p^{17} = {p_1}^2 \times \lambda + (p_0 + p_1)p_0. \tag{5}$$

From Eq.4 and Eq.5, p^{-1} can be computed by $(p^{17})^{-1}(p_0 + p_1)$, $(p^{17})^{-1}p_1 x$, ${p_1}^2 \times \lambda$, and $(p_0 + p_1)p_0$.

2.3 Grover's algorithm

Grover's algorithm can be used to find a single target element y in an unstructured dataset of size $N(n = log_2 N)$, and there exists a function $f(x) : \{0,1\}^n \to \{0,1\}$ that satisfies

$$f(x) = \begin{cases} 1, & x = y, \\ 0, & otherwise. \end{cases}$$

Assuming that the function f can be easily and efficiently implemented, there exists O_f to determine whether the element x is the target element. To solve the above search problem, the classical exhaustive search algorithm requires $O(2^n)$ times O_f operations, whereas Grover's algorithm requires only $O(2^{n/2})$ time O_f operations.

Grover's algorithm consists of the following steps.

1. Prepare the quantum state $|0\rangle^{\otimes n}$, and perform $Hadamard$ operations on individual n qubits to get the state

$$|\psi\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle.$$

2. Construct $O_f : |x\rangle \overset{O_f}{\to} (-1)^{f(x)}|x\rangle$, such that when x is a solution to the problem, $f(x) = 1$, otherwise $f(x) = 0$.

3.After performing $R(R = \lfloor \frac{\pi}{4} 2^{n/2} \rfloor)$ times $(2|\psi\rangle\langle\psi| - I)O_f$ operation can obtain

$$[(2|\psi\rangle\langle\psi| - I)O_f]^R |\psi\rangle \approx |y\rangle.$$

4. Measure the state to obtain the target y.

2.4 Basic quantum gates

Figure 1 illustrates some base quantum gates used in this paper, where the last qubit of the $CNOT$ gate (see Figure 1(b)) and the $Toffoli$ gate (see Figure 1(c)) are the target qubits and the remaining are the control qubits

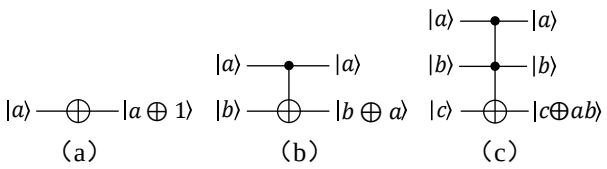

Figure 1: Some base quantum gates used in this paper. (a) The NOT gate; (b) The $CNOT$ gate; (c) The $Toffoli$ gate.

3 The quantum circuits of the S-box

The SubBytes operation is the sole nonlinear component of AES. Optimizing its quantum resource requirements is essential for achieving cost-efficient AES quantum circuit implementation. This section leverages the algebraic structure of the S-box (Section 2.1.1) and composite field arithmetic (Section 2.1.2) to design efficient quantum circuits, and compares the results with existing literature.

3.1 Quantum circuit implementations of M and AM^{-1}

A common approach for implementing quantum circuits for matrix multiplication is Permutation-Lower-Upper (PLU) decomposition. This method typically results in a higher number of quantum gates, particularly for $CNOT$ gates. Xiang et al. [46] propose a heuristic algorithm for optimizing matrix implementation based on matrix decomposition theory, which offers advantages in reducing the number of $CNOT$ gates required.

In this paper, we apply the algorithm from [46] to implement the quantum circuits for matrix multiplication of $U_M : |a\rangle \rightarrow |Ma\rangle$ and $U_{AM^{-1}} : |a\rangle \rightarrow |(AM^{-1})a\rangle$, as shown in Figure 2 and Figure 3, where $a = (a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a0)$. The U_M can be implemented with 8 qubits and 10 $CNOT$ gates, while $U_{AM^{-1}}$ can be implemented with 8 qubits and 15 $CNOT$ gates.

3.2 Quantum circuit implementations of the multiplicative inversion in $F(2^4)$

Li et al. [44] observed that solving the multiplicative inverse of an element in $F(2^4)$ can be represented as a 4×4 S-box. Let $\boldsymbol{b}$ be any element in $F(2^4)$, with its corresponding multiplicative inverse $\boldsymbol{b^{-1}}$ shown in Table 1.

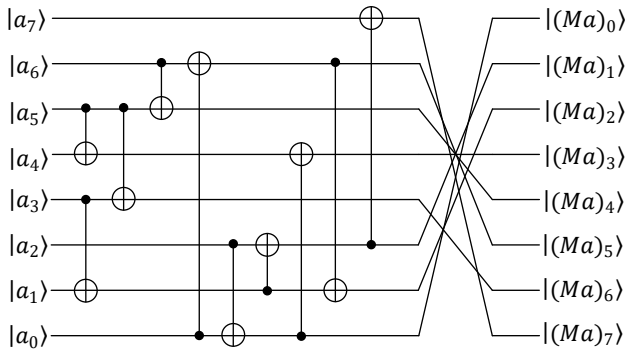

Figure 2: The quantum circuit for $U_M : |a\rangle \rightarrow |Ma\rangle$.

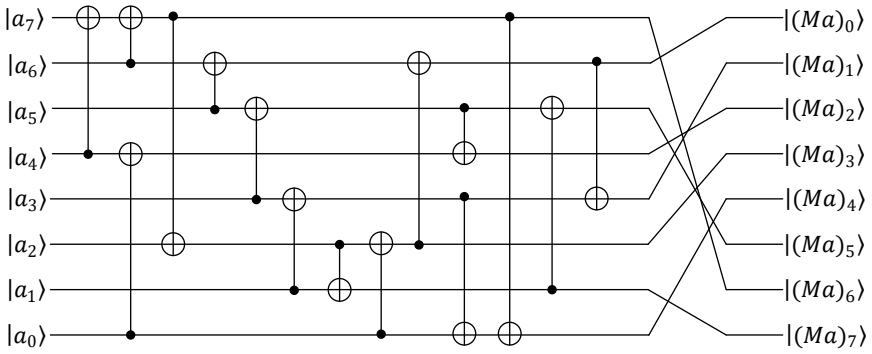

Figure 3: The quantum circuit for $U_{AM^{-1}} : |a\rangle \rightarrow |(AM^{-1})a\rangle$.

Input and Output	Elements															
$\boldsymbol{b}$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$\boldsymbol{b^{-1}}$	0	1	9	E	D	B	7	6	F	2	C	5	A	4	3	8

Table 1: The input elements and corresponding inverse output elements in $F(2^4)$

In this paper, we use the automated tool DORCIS [47] to realize the quantum circuit of the 4×4 lookup table, which solves the multiplicative inverse of elements in $F(2^4)$. The resulting quantum circuit is denoted as $F(2^4)_{inv} : |\boldsymbol{b}\rangle \rightarrow |\boldsymbol{b^{-1}}\rangle$, where $\boldsymbol{b} = (b_3\ b_2\ b_1\ b_0)$ is the input element and $\boldsymbol{b^{-1}} = (b_3^{-1}\ b_2^{-1}\ b_1^{-1}\ b_0^{-1})$ is the output element, as shown in Figure 4.

The $3-$*controlled NOT* gate appearing in Figure 4 can be implemented as $|a\rangle|b\rangle|c\rangle|d\rangle|0\rangle \rightarrow |a\rangle|b\rangle|c\rangle|d \oplus abc\rangle|0\rangle$ (shown in Figure 5(a)) with an auxiliary qubit, or as $|a\rangle|b\rangle|c\rangle|d\rangle|g\rangle \rightarrow |a\rangle|b\rangle|c\rangle|d \oplus abc\rangle|g\rangle$ (shown in Figure 5(b)) without any auxiliary qubits.

Based on Figures 4 and 5, quantum circuits for $F(2^4)_{inv0} : |\boldsymbol{b}\rangle|0\rangle \rightarrow |\boldsymbol{b^{-1}}\rangle|0\rangle$

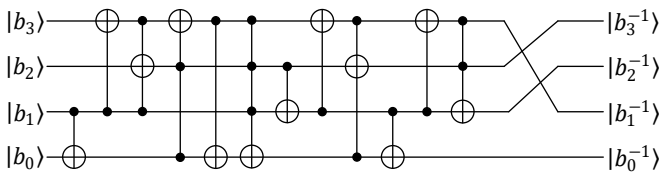

Figure 4: The quantum circuit for $F(2^4)_{inv}: \ |\boldsymbol{b}\rangle \rightarrow |\boldsymbol{b^{-1}}\rangle$.

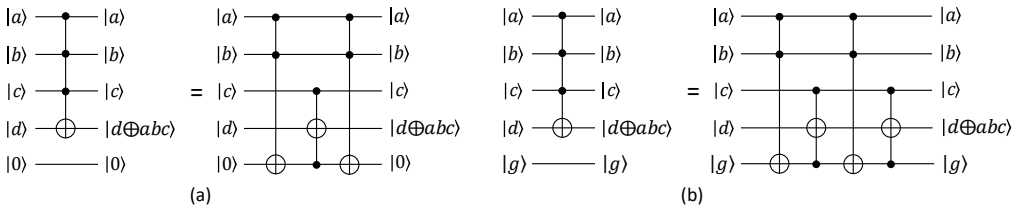

Figure 5: Two types of decomposition of $3-$*controlled NOT* gate. (a) Decomposition of a $3-$*controlled NOT* gate with an auxiliary qubit; (b) Decomposition of a $3-$*controlled NOT* gate without any auxiliary qubit.

and $F(2^4)_{inv1}: \ |\boldsymbol{b}\rangle|g\rangle \rightarrow |\boldsymbol{b^{-1}}\rangle|g\rangle$ can be constructed. The quantum resource estimates for implementing the multiplicative inverse in $F(2^4)$ are summarized in Table 2.

	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
$F(2^4)_{inv0}$	5	7	7	0	7
$F(2^4)_{inv1}$	5	8	7	0	8

Table 2: Quantum resource estimates for implementing the multiplicative inversion in $F(2^4)$. $\#qubits$ means the number of qubits. $\#Toffoli$, $\#CNOT$, and $\#NOT$ mean the number of $Toffoli$ gate, $CNOT$ gate and NOT gate

3.3 Quantum circuit implementation of $q^2 \times \lambda$

q can be written as $q = q_3y^3 + q_2y^2 + q_1y + q_0$, where $q_i (i \in 0, 1, 2, 3)$ is an element in $F(2^4)$ and $\lambda = x + x^2 + x^3 \in F(2^4)$ is defines in Eq.2. Through a calculation, $q^2 \times \lambda$ is expressed as

$$q^2 \times \lambda = (q_1 + q_0)y^3 + (q_3 + q_1 + q_0)y^2 + q_0y + (q_2 + q_1). \tag{6}$$

Based on Eq.2 and Eq.6, we can derive a quantum circuit for $U_{q^2\times\lambda}: \ |q\rangle \rightarrow |q^2 \times \lambda\rangle$ in Figure 6. The $U_{q^2\times\lambda}$ can be implemented with 4 qubits and 3 $CNOT$ gates.

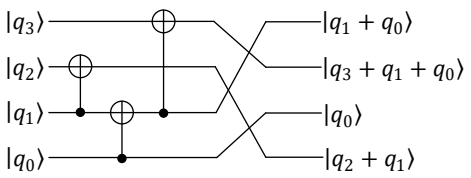

Figure 6: The quantum circuit for $U_{q^2\times\lambda}: \ |q\rangle \rightarrow |q^2 \times \lambda\rangle$.

3.4 Quantum circuit implementations of multiplication in $F(2^4)$

In $F(2^4)$, any element of a and b can written as $\begin{cases} a = a_3y^3 + a_2y^2 + a_1y + a_0 \\ b = b_3y^3 + b_2y^2 + b_1y + b_0 \end{cases}$. $a \cdot b$ is expressed as

$$a \cdot b = (ab)_3y^3 + (ab)_2y^2 + (ab)_1y + (ab)_0, \tag{7}$$

where $(ab)_i$ $(i \in 0, 1, 2, 3)$ is the i-th term of $a \cdot b$ and
$(ab)_3 = (a_3 + a_2 + a_1 + a_0)(b_3 + b_2 + b_1 + b_0) + (a_2 + a_0)(b_2 + b_0) + (a_1 + a_0)(b_1 + b_0) + a_2b_2 + a_0b_0$,
$(ab)_2 = (a_2 + a_0)(b_2 + b_0) + a_3b_3 + a_2b_2 + a_1b_1 + a_0b_0$,
$(ab)_1 = (a_3+a_2)(b_3+b_2)+(a_1+a_0)(b_1+b_0)+(a_1+a_0)(b_1+b_0)+a_2b_2+a_1b_1+a_0b_0$,
$(ab)_3 = (a_3 + a_2)(b_3 + b_2) + (a_3 + a_1)(b_3 + b_1) + a_3b_3 + a_1b_1 + a_0b_0$.

Based on Eq.7, we present a quantum circuit for $Mul_0: \ |a\rangle|b\rangle|0\rangle \rightarrow |a\rangle|b\rangle|a \cdot b\rangle$ of multiplication in $F(2^4)$ in Figure 7. Mul_0 can be implemented with 12 qubits, 9 $Toffoli$ gates and 29 $CNOT$ gates. The $Toffoli$ depth of Mul_0 is 4.

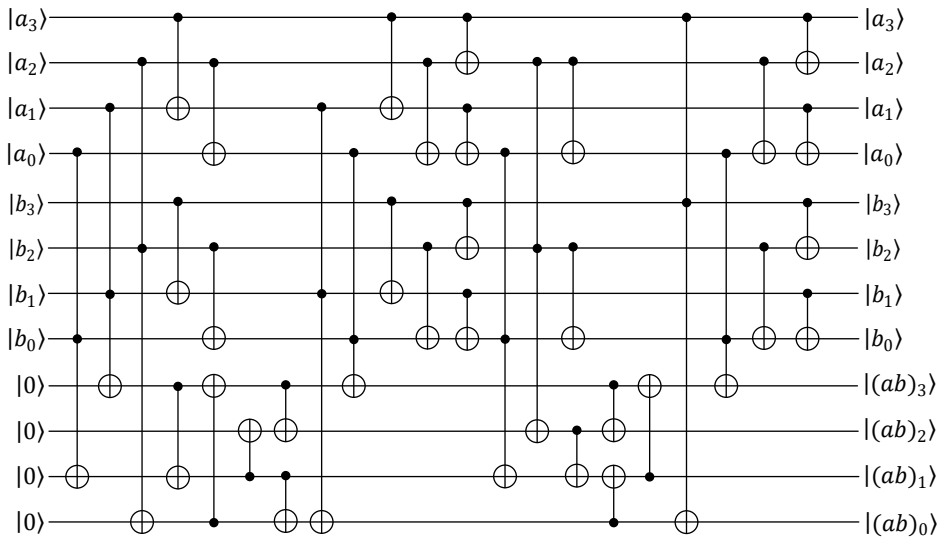

Figure 7: The quantum circuit for $Mul_0: \ |a\rangle|b\rangle|0\rangle \rightarrow |a\rangle|b\rangle|a \cdot b\rangle$.

The inverse operation in $F((2^4)^2)$ requires adding another number to the multiplication result in $F(2^4)$. For this purpose, in this paper, we refer to the method of [44], which is realized $Mul_1 : |a\rangle|b\rangle|h\rangle \rightarrow |a\rangle|b\rangle|h + a \cdot b\rangle$ without any auxiliary qubits in Figure 8, where Mul_0 denotes the quantum circuit shown in Figure 7. Mul_1 can be implemented with 12 qubits, 9 $Toffoli$ gates and 33 $CNOT$ gates. The $Toffoli$ depth of Mul_1 is 4.

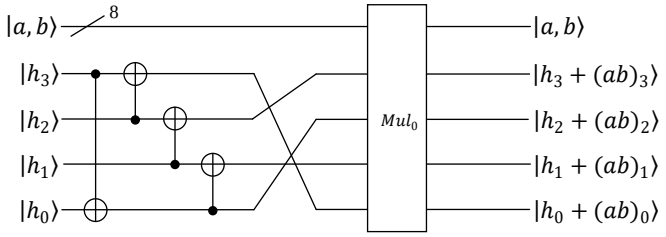

Figure 8: The quantum circuit for $Mul_1 : |a\rangle|b\rangle|h\rangle \rightarrow |a\rangle|b\rangle|h + a \cdot b\rangle$.

Referring to [44], we present the quantum circuit for $Mul_2 : |a\rangle|b\rangle|0\rangle \rightarrow |a \cdot b\rangle|b\rangle|0\rangle$, as shown in Figure 9, where $F(2^4)_{inv1}$ denotes the quantum circuit shown in sect.3.2, ${F(2^4)_{inv1}}^{\dagger}$ is the inverse process of $F(2^4)_{inv1}$, and $Mul_1^{\dagger}$ is the inverse process of Mul_1. Mul_2 can be implemented with 12 qubits, 33 $Toffoli$ gates and 72 $CNOT$ gates. The $Toffoli$ depth of Mul_2 is 23.

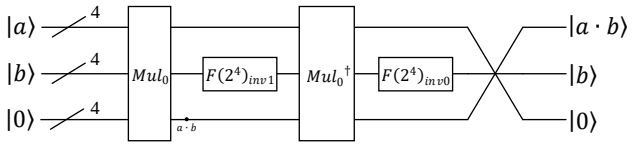

Figure 9: The quantum circuit for $Mul_2 : |a\rangle|b\rangle|0\rangle \rightarrow |a \cdot b\rangle|b\rangle|0\rangle$.

3.5 The quantum circuit of S-box

Based on Eq.3, the S-box can be computed by realizing the quantum circuits of M, $(Ma)^{-1}$, AM^{-1} and adding a vector c in order. The quantum circuits of M and AM^{-1} have been presented in sect.3.1. The addion of the vector c can be implemented with only 4 NOT gates. This section presents three distinct quantum circuits for implementing the S-box, depending on varying target qubit states.

3.5.1 The quantum circuit for c_1

When the target qubit is $|0\rangle$, the quantum circuit for $c_1 : \ |a\rangle|0\rangle \rightarrow |a\rangle|S(a)\rangle$ is provided. The first construction is the quantum circuit for $(Ma)^{-1}$ that implements the multiplicative inverse, and the quantum circuit for c_1 is introduced Subsequently.

Any element in $F((2^4)^2)$ can be denoted as $p = p_0 + p_1 x$, where $p_0, \ p_1 \in F(2^4)$. The multiplicative inverse of p in $F((2^4)^2)$, denoted as p^{-1}, can be determined through a four-part process involving $(p^{17})^{-1}(p_0 + p_1)$, $(p^{17})^{-1}p_1 x$, ${p_1}^2 \times \lambda$, and $(p_0 + p_1)p_0$. Therefore, by combining the quantum circuits of $U_{q^2 \times \lambda}$ (see Figure 6) , Mul_0 (as in Figure 7) and Mul_1 (see Figure 8) , and $F(2^4)_{inv0}$ (see sect.3.2), the quantum circuit $U_{inv0} : \ |p\rangle|0\rangle|0\rangle \rightarrow |p\rangle|p^{-1}\rangle|0\rangle$ for computing the multiplicative inversion in $F((2^4)^2)$ can be constructed as shown in Figure 10, where ${U_{(p^{-1})^{17}}}^\dagger$ is the inverse process of $U_{(p^{-1})^{17}}$. At this point, there are 8 auxiliary qubits remaining in the state $|0\rangle$, allowing the direct use of $F(2^4)_{inv0}$ to construct U_{inv0}, eliminating the need to use $F(2^4)_{inv1}$.

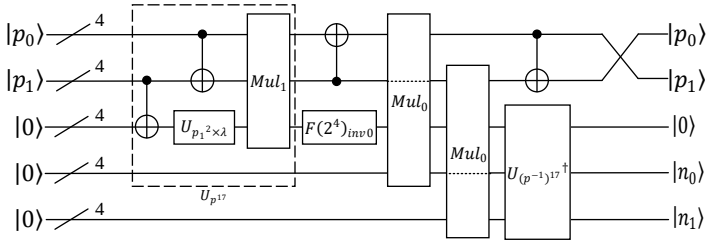

Figure 10: The quantum circuit for $U_{inv0} : \ |p\rangle|0\rangle|0\rangle \rightarrow |p\rangle|p^{-1}\rangle|0\rangle$.

By combining the quantum circuit of U_M (see Figure 2) , U_{inv0} (see Figure 10) , and $U_{AM^{-1}}$ (see Figure 3), the quantum circuit for $c_1 : \ |a\rangle|0\rangle \rightarrow |a\rangle|S(a)\rangle$ can be constructed as shown in Figure 11 with a wide of 20, where ${U_M}^\dagger$ is the inverse process of U_M. The quantum resources required for implementing c_1 is summarized in Table 3.

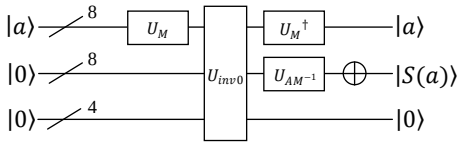

Figure 11: The quantum circuit for $c_1 : \ |a\rangle|0\rangle \rightarrow |a\rangle|S(a)\rangle$.

Schemes	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
Ours	20	43	196	4	23
[44]	22	48	236	4	36
	23	48	238	4	34
	24	46	240	4	32
[42]	22	52	326	4	41
	23	48	230	4	39
	24	46	332	4	37
[41]	32	55	314	4	40
[39]	40	512	357	4	144

Table 3: The quantum resources required for implementing c_1

3.5.2 The quantum circuit for c_2

When the target qubit is $|b\rangle$ instead of $|0\rangle$, the given quantum circuit is denoted as $c_2 : |a\rangle|b\rangle \rightarrow |a\rangle|b \oplus S(a)\rangle$. Since the target bit is not always $|0\rangle$, the quantum circuit U_{inv0} for computing the multiplicative inverse (as shown in Figure 10) is not suitable for constructing the quantum circuit .

Thus, by combining the quantum circuits of $U_{q^2\times\lambda}$ (see Figure 6), Mul_1 (see Figure 8), and $F(2^4)_{inv1}$ (see in sect.3.2), the quantum circuit U_{inv1} : $|p\rangle|h\rangle|0\rangle \rightarrow |p\rangle|h + p^{-1}\rangle|0\rangle$ for computing the multiplicative inverse in $F((2^4)^2)$ can be constructed as shown in Figure 12, where ${F(2^4)_{inv1}}^{\dagger}$ is the inverse process of $F(2^4)_{inv1}$. At this point, there are no remaining auxiliary qubits remaining in the $|0\rangle$ state, so the quantum circuit $F(2^4)_{inv1}$ is used directly instead of using $F(2^4)_{inv0}$.

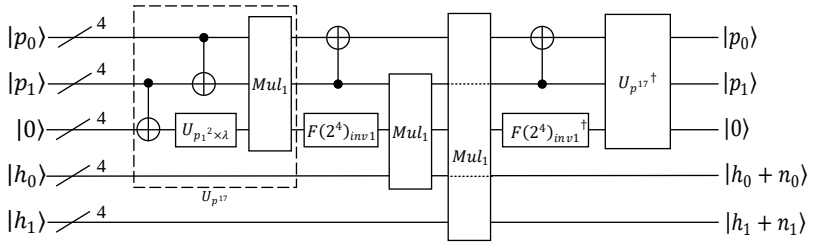

Figure 12: The quantum circuit for $U_{inv1} : |p\rangle|h\rangle|0\rangle \rightarrow |p\rangle|h + p^{-1}\rangle|0\rangle$.

By combining the quantum circuits of U_M (see Figure 2) , U_{inv1} (see Figure 12) , and $U_{AM^{-1}}$ (see Figure 3), the quantum circuit for $c_2 : |a\rangle|b\rangle \rightarrow |a\rangle|b \oplus S(a)\rangle$

can be constructed as shown in Figure 13 with a wide of 20, where ${U_M}^\dagger$ is the inverse process of U_M and ${U_{AM^{-1}}}^\dagger$ is the inverse process of $U_{AM^{-1}}$.

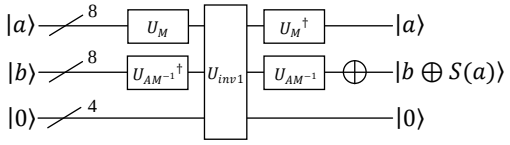

Figure 13: The quantum circuit for $c_2: \ |a\rangle|b\rangle \rightarrow |a\rangle|b \oplus S(a)\rangle$.

When an auxiliary qubit in the $|0\rangle$ state is added, the replacement in the quantum circuit shown in Figure 12 can be made, resulting in the construction of a quantum circuit for the S-box with a width of 21. The quantum resources required for implementing c_2 are shown in Table 4.

Schemes	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
Ours	20	52	226	4	32
	21	50	226	4	30
[44]	22	48	272	4	36
	23	48	274	4	34
	24	46	276	4	32
[43]	32	55	322	4	40
[42]	23	68	352	4	60
	24	64	356	4	58
	25	63	358	4	56

Table 4: The quantum resources required for implementing c_2

3.5.3 The quantum circuit for c_3

In order to reduce the number of qubits used in the quantum circuit of the S-box, this section proposes a quantum circuit implementation for $c_3: \ |a\rangle|0\rangle \rightarrow |S(a)\rangle|0\rangle$. By combining the quantum circuits of $U_{q^2\times\lambda}$ (see Figure 6), Mul_1 (see Figure 8) and Mul_2 (see Figure 9), and $F(2^4)_{inv0}$ (see in sect.3.2), the quantum circuit $U_{inv2}: \ |p\rangle|0\rangle \rightarrow |p^{-1}\rangle|0\rangle$ for computing the multiplicative inverse in $F((2^4)^2)$ can be constructed as shown in Figure 14. At this point, there are 4 auxiliary qubits remaining in the $|0\rangle$ state, allowing the direct use of $F(2^4)_{inv0}$ instead of $F(2^4)_{inv1}$ to construct U_{inv2} in $F((2^4)^2)$.

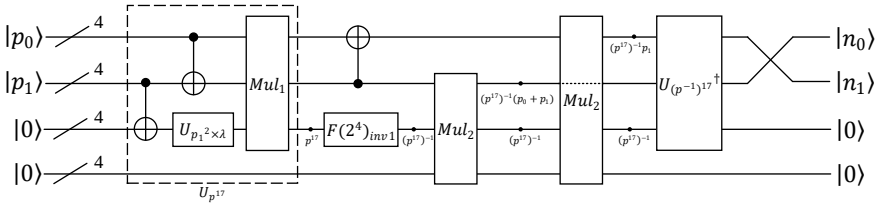

Figure 14: The quantum circuit for $U_{inv2}: \ |p\rangle|0\rangle \rightarrow |p^{-1}\rangle|0\rangle$.

By combining the quantum circuits of U_M (see Figure 2) , U_{inv2} (see Figure 14) , and $U_{AM^{-1}}$ (see Figure 3), the quantum circuit for $c_3: \ |a\rangle|0\rangle \rightarrow |S(a)\rangle|0\rangle$ can be constructed as shown in Figure 15 with a wide of 16. The quantum resources required to implement c_3 is summarized in Table 5.

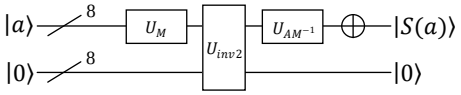

Figure 15: The quantum circuit for $c_3: \ |a\rangle|0\rangle \rightarrow |S(a)\rangle|0\rangle$.

Schemes	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
Ours	16	92	268	4	62
[44]	22	96	426	4	71
	23	96	426	4	67
	24	92	426	4	58

Table 5: The quantum resources required for implementing c_3

4 The quantum circuit for AES

The other three linear transformation structures of AES, ShiftRows, MixColumns and AddRoundKey, can all be implemented with $CNOT$ gates only. The quantum resources required for these three linear transformations are detailed below.

1. AddRoundKey: Each AddRoundKey operation involves XORing a 128-bit subkey with the current state. This can be implemented in parallel using 128 $CNOT$ gates.

2. ShiftRows: The ShiftRows operation only rearranges the output order of 16 bytes in the current state. It does not require any quantum operations for implementation.

3. MixColumns: The MixColumns operation processes 32 bits in the current state at a time and can be implemented using a 32×32 matrix. Xiang et al. [46] provides a quantum circuit for MixColumns operation with 92 $CNOT$ gates. As MixColumns can be applied to all 4 columns of the state simultaneously, and the total requirement for $CNOT$ gates is $92 \times 4 = 368$.

For the key expansion process, we implement the key expansion process of AES with Jaques et al.'s method [48], but we use our S-box circuit for c_2 (see Figure 13). The quantum circuit as shown in Figure 16 represents the full structure of the key expansion process, denoted as $\mathcal{KE} : |k\rangle_{i-1} \rightarrow |k\rangle_i$. The $SubBytes$ with arrow in Figure 16 represents the operation where the state at the tail of the arrow undergoes the SubBytes operation, followed by XORing with the state pointed to by the arrow, and the result is assigned to the state pointed to by the arrow. The $RotWord$ shown in Figure 16 is the RotWord operation, which can be realized by rearranging the order of qubits without any quantum operations, while the $Rcon[j]$ operation can be implemented using a single NOT gate. The key expansion process for AES-128, AES-192, and AES-256 can respectively utilize portions of the full key expansion structures shown in Figures 16(a), (b), and (c).

The initial key of AES undergoes N_R rounds of the key expansion process, resulting in (N_R+1) round keys. Based on Algorithms 1 and 2, each round of key expansion process only operates on 128 bits. Therefore, only the relevant subset of the full structure of the key expansion process is needed for each round, with other operations being neglected. Let $\mathcal{KE}_j^l$ represents the operation from $|k_j\rangle_{i-1}$ to $|k_l\rangle_{i-1}$ in $\mathcal{KE} : |k\rangle_{i-1} \rightarrow |k\rangle_i$, while ignoring other parts, where $0 \leq j \leq l$. The quantum resources required to implement the key expansion process for AES is shown in Table 6.

	N_R	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
AES-128	10	132	2080	10000	190	1280
		133	2000	10000	190	1200
AES-192	12	196	2288	11064	209	1408
		197	2200	11064	209	1320
AES-256	14	260	2704	13000	229	1664
		261	2600	13000	229	1560

Table 6: The quantum resources required to implement the key expansion processes for AES

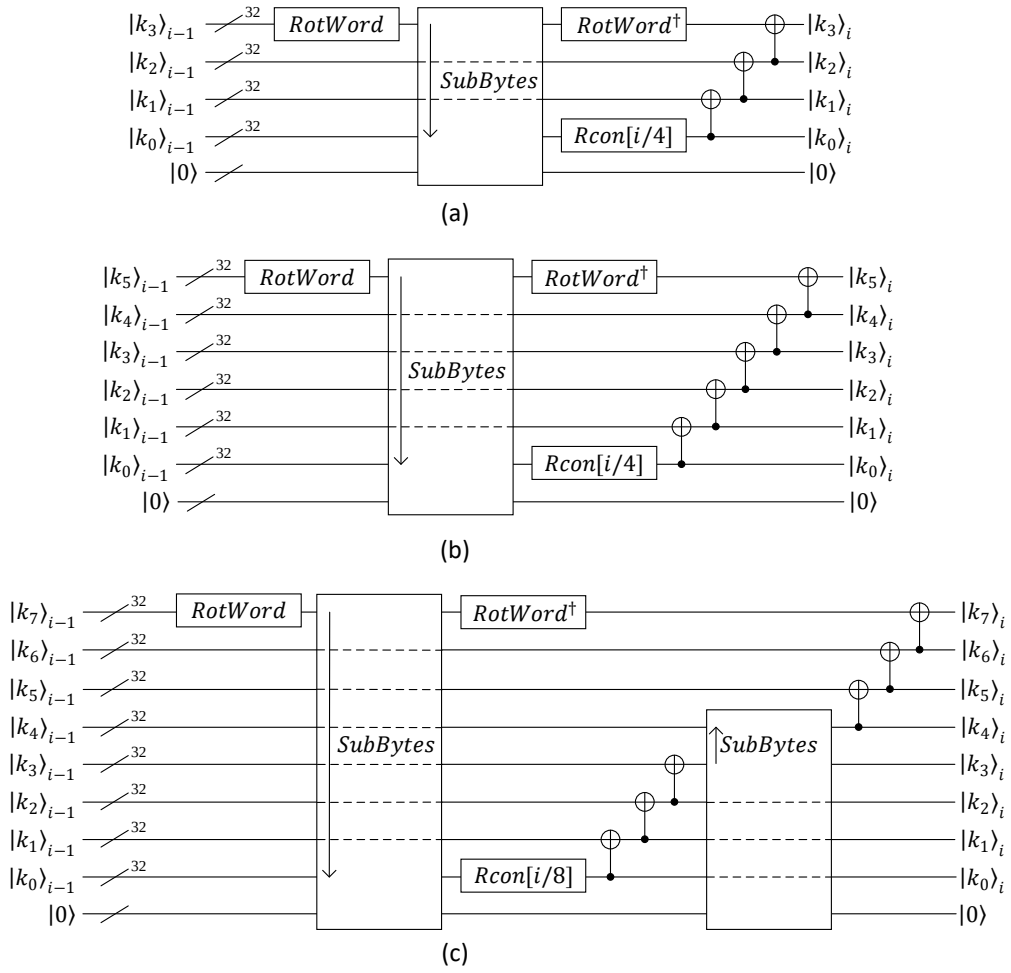

Figure 16: Three fully structures of the key expansion process. (a) A full structure of the key expansion process that can be used for AES-128. (b) A full structure of the key expansion process that can be used for AES-192. (c) A full structure of the key expansion process that can be used for AES-256.

The encryption process of the AES is presented in Figure 17, where $SubBytes$, $ShiftRows$ and $MixColumn$ are the operations of SubBytes, ShifrRows and MixColumns, and $\mathcal{KE}_j^l$ represents the operation from $|k_j\rangle_{i-1}$ to $|k_l\rangle_{i-1}$ in $\mathcal{KE}$: $|k\rangle_{i-1} \rightarrow |k\rangle_i$, while ignoring other parts. In Figures 17(a), (b) and (c), $|K_0\rangle$, $|K^2\rangle_0|K^1\rangle_0|K^0\rangle_0$, and $|K^1\rangle_0|K^0\rangle_0$ denote the 128-bit, 192-bit, and 256-bit initial keys for AES-128, AES-192, and AES-256, respectively. $|M\rangle$, $|M^1\rangle|M^0\rangle$ and $|M\rangle$ denote the plaintexts, while $|C\rangle$, $|C^1\rangle|C^0\rangle$ and $|C\rangle$ denote the ciphertexts.

Since the primary goal of this paper is to reduce the number of qubits required for implement the quantum circuit for AES, the SubBytes operation during

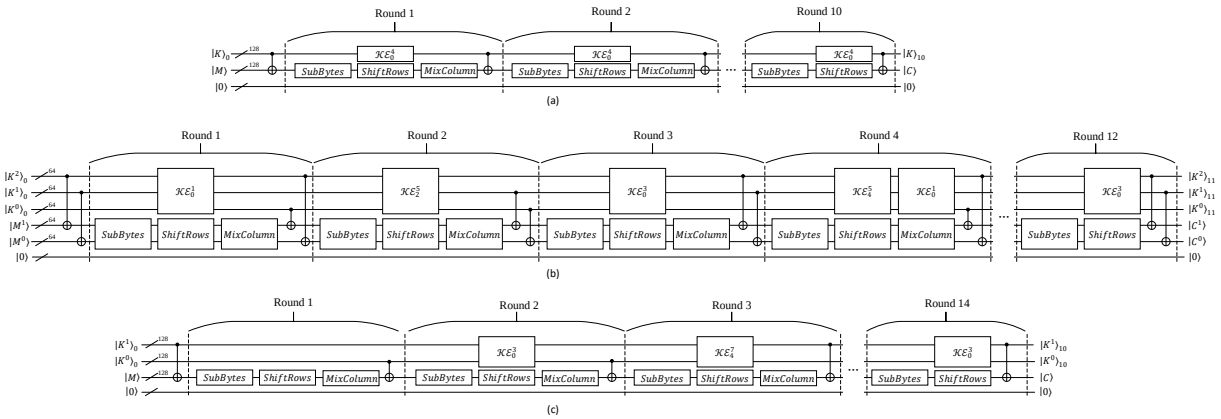

Figure 17: The encryption process of AES. (a) The encryption process of AES-128. (b) The encryption process of AES-192. (c) The encryption process of AES-256.

encryption is implemented using $c_3 : |a\rangle|0\rangle \rightarrow |S(a)\rangle|0\rangle$. The quantum resources required to implement the quantum circuit for AES-128, AES-192, and AESS-256 are summarized in Table 7, 8, and 9.

Schemes	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
Ours	264	16800	57840	830	11200
	265	16720	57840	830	11120
	268	16800	57840	830	9920
	269	16720	57840	830	9920
[44]	270	16508	81652	1072	11008
	328	15824	82928	1072	2184
	380	16480	81592	1072	1344
	400	15824	82928	1072	1108
[43]	400	19064	118980	4528	-
	656	18040	101174	1976	-
[42]	512	19788	128517	4528	2016
[41]	864	16940	107960	1507	1880
[40]	976	150528	192832	1370	-
[39]	984	151552	166548	1456	12672

Table 7: The quantum resources required to implement the quantum circuits for AES-128

Schemes	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
Ours	328	19952	68472	977	13312
	329	19864	68472	977	13224
	332	19952	68472	977	11904
	333	19864	68472	977	11904
[44]	334	19196	94180	1160	13144
	392	18400	95696	1160	2616
	444	19168	94168	1160	1596
	464	18400	95696	1160	1340
[42]	640	22380	152378	5128	2022
[41]	896	19580	125580	1692	1640
[39]	1112	172032	189432	1608	11088

Table 8: The quantum resources required to implement the quantum circuits for AES-192

Schemes	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
Ours	392	23312	79976	1125	15552
	393	23208	79976	1125	15448
	396	23312	79976	1125	13888
	397	23208	79976	1125	13888
[44]	398	23228	114476	1367	15756
	456	22264	116288	1367	3048
	508	23208	114376	1367	1880
	528	22264	116288	1367	1540
[42]	768	26774	177645	6103	2292
[41]	1232	23760	151011	1992	2160
[39]	1336	215040	233836	1943	14976

Table 9: The quantum resources required to implement the quantum circuits for AES-256

As shown in Table 7, 8,and 9, the proposed quantum circuit of the S-box significantly reduces the quantum resources needed to implement the quantum

circuit for AES. This optimization not only enhances the execution efficiency of the quantum circuit but also provides a feasible solution for implementing larger scale quantum encryption algorithms under limited quantum resources.

For AES-128, when the width is 264 or 265, the $Toffoli$ gate cannot be replaced by the $QAND$ gate (see in Figure 18) since there is no clean auxiliary qubit available in the circuit at this point. However, when the width is 268 or 269, the $Toffoli$ gate can be substitutes by the $QAND$ gate. (The 4 clean auxiliary qubits used to decompose the $Toffoli$ gate into a $QAND$ gate at this stage originate from the key expansion process. Since each round of key expansion involves only 4 SubBytes operations, whereas the encryption process requires 16 per round, it is feasible to use the 4 auxiliary qubits from the key expansion to implement the $QAND$ gate.)

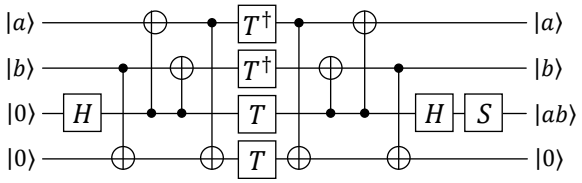

Figure 18: The $QAND$ gate.

For comparison with [48], the $Toffoli$ gates in AES-128 are decomposed using the scheme proposed in [48], which requires 7 T gates and the T depth is 3. The resulting data are presented in Table 10, where $DW(Toffoli)$ represents the product of the $Toffoli$ depth and width, and $DW(T)$ represents the product of T depth and width.

Schemes	$\#qubits$	$Toffoli$ depth	$DW(Toffoli)$	T depth	$DW(T)$
Ours	264	11200	2956800	33600	8870400
	265	11120	2946800	33360	8840400
	268	9920	2658560	28480	7632640
	269	9920	2668480	28480	7661120
[48]	256	17140	4387840	29490	7549440

Table 10: Costs of the quantum circuits for AES-128

5 Grover search attack on AES

In this section, using the propose quantum circuit of AES (see in sect.4), a concrete resource estimation is conducted for mounting Grover search attack on AES. Firstly, Grover *Oracle* is designed for AES. And then, based on the design of Grover *Oracle*, resources required to perform Grover search attack on AES is estimated.

5.1 Resource Estimation of Grover *Oracle*

According to reference [50], $r = \lceil k/n \rceil$ instances of known plaintext-ciphertext pairs are needed to find the unique decryption key, where n is the block size and k is the key length of the block cipher. If k/n is an integer, $r = \lceil k/n \rceil$ should be replaced by $r = k/n + 1$, and the probability of successful key search is $\mathrm{e}^{-2^{k-rn}}$. Therefore, for AES-128, implementing the Grover search attack requires 2 known plaintext-ciphertext pairs, at which point the probability of a successful key search is $\mathrm{e}^{-2^{-128}} \approx 1$. Figure 19(a) shows the construction of the Grover search attack for 2 instances of known plaintext-ciphertext pairs considering AES-128, where $AES-128^{\dagger}$ is the inverse process of the quantum circuit $AES-128$. Similarly, for AES-192 and AES-256, the Grover search attack requires 2 and 3 instances of known plaintext-ciphertext pairs, as shown in Figures 19(b) and 19(c), respectively.

The Grover search attack consists of comparing rn-bit outputs of the AES instances with the given r ciphertexts. This operation can be implemented using $(2n \cdot r) - \mathit{controlled\ NOT}$ gates. We neglect some NOT gates which depend on the given ciphertexts and only consider the depth of the AES instances ignoring the multi-$-\mathit{controlled\ NOT}$ gates used in comparing the ciphertexts. The quantum resources required to implement the Grover *Oracle* are shown in Table 11.

5.2 Resource Estimation of Grover search attack on AES

Using the quantum estimates in Table 11, we provide the quantum resources required to implement the Grover search attack on AES are shown in Table 12. Compared to the quantum resource required for implementing Grover *Oracle* (i.e., O_f as discussed in sect.2.3), the resources required for other quantum operations for implementing $(2|\psi\rangle\langle\psi| - I)O_f$ (as discussed in sect.2.3) are relatively small. Therefore, when analyzing the overall resource requirements for Grover

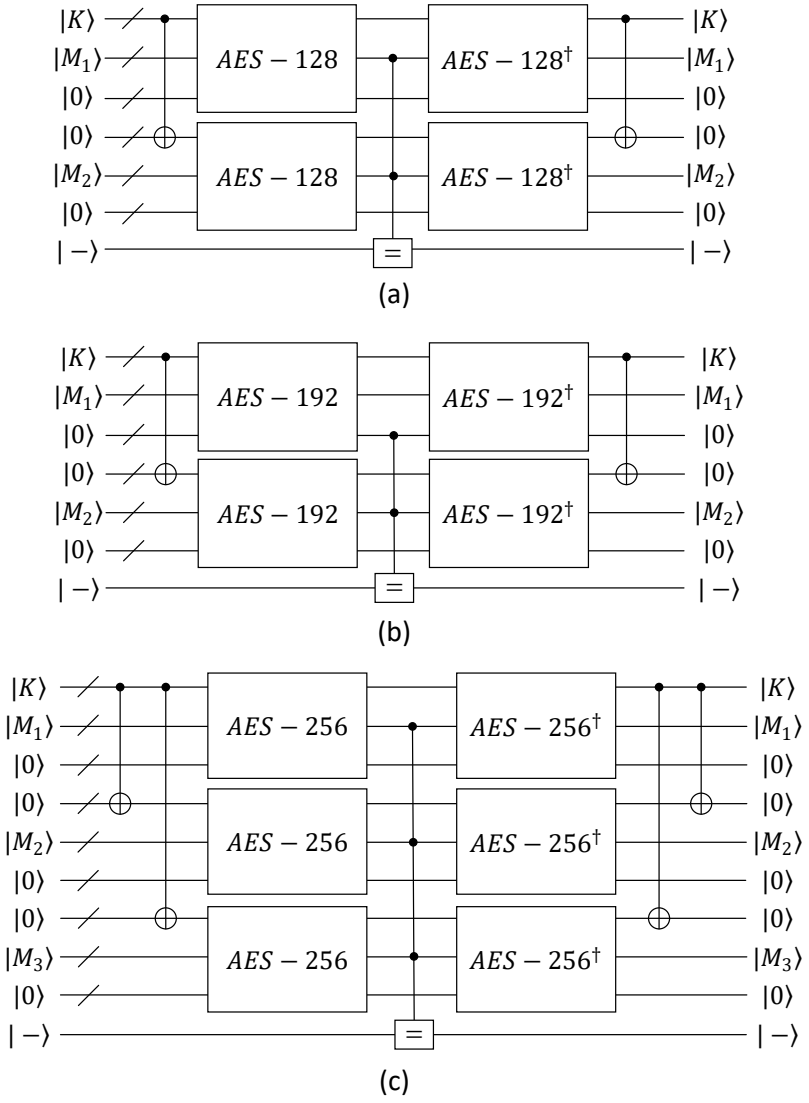

Figure 19: Grover *Oracle* for AES. (a) Grover *Oracle* for AES-128. (b) Grover *Oracle* for AES-192. (c) Grover *Oracle* for AES-256.

search attack, it is reasonable to ignore the consumption of these quantum resources.

	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
AES-128	529	67200	231616	3320	22400
	531	66880	231616	3320	22240
	537	67200	231616	3320	19840
	539	66880	231616	3320	19840
AES-192	657	79808	274272	3908	26624
	659	79456	274272	3908	26448
	665	79808	274272	3908	23808
	667	79456	274272	3908	23808
AES-256	1177	139872	480880	6750	31104
	1180	139248	480880	6750	30896
	1189	139872	480880	6750	27776
	1192	139248	480880	6750	27776

Table 11: The quantum resources required to implement the Grover $Oracle$

	$\#qubits$	$\#Toffoli$	$\#CNOT$	$\#NOT$	$Toffoli$ depth
AES-128	$1.0050 \times 2^{136.69}$	$1.0043 \times 2^{143.68}$	$1.0010 \times 2^{145.47}$	$1.0048 \times 2^{139.34}$	$1.0008 \times 2^{142.1}$
	$1.0018 \times 2^{136.70}$	$1.0065 \times 2^{143.67}$	$1.0010 \times 2^{145.47}$	$1.0048 \times 2^{139.34}$	1.0006×2^{142}
	$1.0061 \times 2^{136.71}$	$1.0043 \times 2^{143.68}$	$1.0010 \times 2^{145.47}$	$1.0048 \times 2^{139.34}$	$1.0043 \times 2^{141.92}$
	$1.0029 \times 2^{136.72}$	$1.0065 \times 2^{143.67}$	$1.0010 \times 2^{145.47}$	$1.0048 \times 2^{139.34}$	$1.0043 \times 2^{141.92}$
AES-192	1.0068×2^{137}	$1.0034 \times 2^{143.93}$	$1.0036 \times 2^{145.71}$	$1.0015 \times 2^{139.58}$	$1.0003 \times 2^{142.34}$
	$1.0029 \times 2^{137.01}$	$1.0055 \times 2^{143.92}$	$1.0036 \times 2^{145.71}$	$1.0015 \times 2^{139.58}$	$1.0006 \times 2^{142.33}$
	$1.0050 \times 2^{137.02}$	$1.0034 \times 2^{143.93}$	$1.0036 \times 2^{145.71}$	$1.0015 \times 2^{139.58}$	$1.0064 \times 2^{142.18}$
	$1.0011 \times 2^{137.03}$	$1.0055 \times 2^{143.92}$	$1.0036 \times 2^{145.71}$	$1.0015 \times 2^{139.58}$	$1.0064 \times 2^{142.18}$
AES-256	$1.0006 \times 2^{137.85}$	$1.0023 \times 2^{144.74}$	$1.0037 \times 2^{146.52}$	$1.0005 \times 2^{140.37}$	$1.0033 \times 2^{142.57}$
	$1.0032 \times 2^{137.85}$	$1.0051 \times 2^{144.73}$	$1.0037 \times 2^{146.52}$	$1.0005 \times 2^{140.37}$	$1.0036 \times 2^{142.56}$
	$1.0038 \times 2^{137.86}$	$1.0023 \times 2^{144.74}$	$1.0037 \times 2^{146.52}$	$1.0005 \times 2^{140.37}$	$1.0011 \times 2^{142.41}$
	$1.0063 \times 2^{137.86}$	$1.0051 \times 2^{144.73}$	$1.0037 \times 2^{146.52}$	$1.0005 \times 2^{140.37}$	$1.0011 \times 2^{142.41}$

Table 12: The quantum resources required to implement the Grover search attack on AES

6 Conclusion

This paper presents AES quantum circuits realized with fewer qubits. We first designed three different types of quantum circuits for the S-box. Additionally, we introduced a linear key expansion operation to ensure the complete functionality of AES quantum circuit of AES while minimizing the use of qubits.

Through these optimizations, we successfully designed quantum circuits for AES-128/192/256 that require only 264/328/398 qubits, and comprehensively estimated the consumption of other quantum resources based on these designs. Furthermore, to validate the practical performance of the optimized quantum circuits for AES, we implemented a Grover search attack on AES, demonstrating the significant advantages of the optimized design in reducing quantum resource usage.

References

[1] Nielsen M A, Chuang I L. Quantum computation and quantum information[M]. Cambridge university press, 2010.

[2] Preskill J. Quantum computing in the NISQ era and beyond[J]. Quantum, 2018, 2: 79.

[3] Yanofsky N S, Mannucci M A. Quantum computing for computer scientists[M]. Cambridge University Press, 2008.

[4] Deutsch D. Quantum theory, the Church-Turing principle and the universal quantum computer[J]. Proc. Roy. Soc. London Ser. A, 1989, 400: 96-117.

[5] Jozsa R. Quantum algorithms and the Fourier transform[J]. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 1998, 454(1969): 323-337.

[6] Ekert A, Jozsa R. Quantum computation and Shor's factoring algorithm[J]. Reviews of Modern Physics, 1996, 68(3): 733.

[7] Cleve R, Ekert A, Macchiavello C, et al. Quantum algorithms revisited[J]. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 1998, 454(1969): 339-354.

[8] Briegel H J, Raussendorf R. Persistent entanglement in arrays of interacting particles[J]. Physical Review Letters, 2001, 86(5): 910.

[9] Horodecki R, Horodecki P, Horodecki M, et al. Quantum entanglement[J]. Reviews of modern physics, 2009, 81(2): 865-942.

[10] Zeng Y, Dong Z, Wang H, et al. A general quantum minimum searching algorithm with high success rate and its implementation[J]. Science China Physics, Mechanics & Astronomy, 2023, 66(4): 240315.

[11] Zheng Q, Yu M, Zhu P, et al. Solving the subset sum problem by the quantum Ising model with variational quantum optimization based on conditional values at risk[J]. Science China Physics, Mechanics & Astronomy, 2024, 67(8): 280311.

[12] Gisin N, Ribordy G, Tittel W, et al. Quantum cryptography[J]. Reviews of modern physics, 2002, 74(1): 145.

[13] Shen A, Cao X Y, Wang Y, et al. Experimental quantum secret sharing based on phase encoding of coherent states[J]. Science China Physics, Mechanics & Astronomy, 2023, 66(6): 260311.

[14] Bennett C H, Brassard G. Quantum cryptography: Public key distribution and coin tossing[J]. Theoretical computer science, 2014, 560: 7-11.

[15] Bennett C H. Quantum cryptography using any two nonorthogonal states[J]. Physical review letters, 1992, 68(21): 3121.

[16] Li H W, Hao C P, Chen Z J, et al. Security of quantum key distribution with virtual mutually unbiased bases[J]. Science China Physics, Mechanics & Astronomy, 2024, 67(7): 270313.I.

[17] Lv M Y, Hu X M, Gong N F, et al. Demonstration of controlled high-dimensional quantum teleportation[J]. Science China Physics, Mechanics & Astronomy, 2024, 67(3): 230311.

[18] Lo H K, Chau H F. Unconditional security of quantum key distribution over arbitrarily long distances[J]. science, 1999, 283(5410): 2050-2056.

[19] Mayers D. Unconditional security in quantum cryptography[J]. Journal of the ACM (JACM), 2001, 48(3): 351-406.

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

[21] Ekert A, Jozsa R. Quantum computation and Shor's factoring algorithm[J]. Reviews of Modern Physics, 1996, 68(3): 733.

[22] Grover L K. A fast quantum mechanical algorithm for database search[C]//Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. 1996: 212-219.

[23] Biham E, Biham O, Biron D, et al. Grover's quantum search algorithm for an arbitrary initial amplitude distribution[J]. Physical Review A, 1999, 60(4): 2742.

[24] Simon D R. On the power of quantum computation[J]. SIAM journal on computing, 1997, 26(5): 1474-1483.

[25] Brassard G, Hoyer P, Tapp A. Quantum algorithm for the collision problem[J]. arXiv preprint quant-ph/9705002, 1997.

[26] Boyer M, Brassard G, Høyer P, et al. Tight bounds on quantum searching[J]. Fortschritte der Physik: Progress of Physics, 1998, 46(4-5): 493-505.

[27] Grassl M, Langenberg B, Roetteler M, et al. Applying Grover's algorithm to AES: quantum resource estimates[C]//International Workshop on Post-Quantum Cryptography. Cham: Springer International Publishing, 2016: 29-43.

[28] Dworkin M. Recommendation for Block Cipher Modes of Operation[J]. Methods and Techniques, 2001.

[29] Barker E. AR Transitioning the Use of Cryptographic Algorithms and Key Lengths[J]. National Institute of Standards and Technology: Washington, DC, USA, 2019.NIST

[30] Dworkin M J, Barker E, Nechvatal J R, et al. Advanced encryption standard (AES)[J]. 2001.

[31] Standard D E. Data encryption standard[J]. Federal Information Processing Standards Publication, 1999, 112: 3.

[32] NIST C F P. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process[J]. 2016.

[33] Daemen J. AES Proposal: Rijndael[J]. 1999.

[34] Joan D, Vincent R. The design of Rijndael: AES-the advanced encryption standard[J]. Information Security and Cryptography, 2002, 196.

[35] Schneier B. Applied cryptography: protocols, algorithms, and source code in C[M]. john wiley & sons, 2007.

[36] Ferguson N, Schneier B, Kohno T. Cryptography engineering: design principles and practical applications[M]. John Wiley & Sons, 2011.

[37] NIST C F P. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process[J]. 2016.

[38] Chen L, Chen L, Jordan S, et al. Report on post-quantum cryptography[M]. Gaithersburg, MD, USA: US Department of Commerce, National Institute of Standards and Technology, 2016.

[39] Grassl M, Langenberg B, Roetteler M, et al. Applying Grover's algorithm to AES: quantum resource estimates[C]//International Workshop on Post-Quantum Cryptography. Cham: Springer International Publishing, 2016: 29-43.

[40] Almazrooie M, Samsudin A, Abdullah R, et al. Quantum reversible circuit of AES-128[J]. Quantum information processing, 2018, 17: 1-30.

[41] Langenberg B, Pham H, Steinwandt R. Reducing the cost of implementing the advanced encryption standard as a quantum circuit[J]. IEEE Transactions on Quantum Engineering, 2020, 1: 1-12.

[42] Zou J, Wei Z, Sun S, et al. Quantum circuit implementations of AES with fewer qubits[C]//Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26. Springer International Publishing, 2020: 697-726.

[43] Wang Z G, Wei S J, Long G L. A quantum circuit design of AES requiring fewer quantum qubits and gate operations[J]. Frontiers of Physics, 2022, 17(4): 41501.

[44] Li Z Q, Cai B B, Sun H W, et al. Novel quantum circuit implementation of Advanced Encryption Standard with low costs[J]. Science China Physics, Mechanics & Astronomy, 2022, 65(9): 290311.

[45] Wolkerstorfer J, Oswald E, Lamberger M. An ASIC implementation of the AES SBoxes[C]//Topics in Cryptology—CT-RSA 2002: The Cryptographers' Track at the RSA Conference 2002 San Jose, CA, USA, February 18–22, 2002 Proceedings. Springer Berlin Heidelberg, 2002: 67-78.

[46] Xiang Z, Zeng X, Lin D, et al. Optimizing implementations of linear layers[J]. IACR Transactions on Symmetric Cryptology, 2020.

[47] Chun M, Baksi A, Chattopadhyay A. Dorcis: depth optimized quantum implementation of substitution boxes[J]. Cryptology ePrint Archive, 2023.

[48] Huang Z, Zhang F, Lin D. Constructing Quantum Implementations with the Minimal T-depth or Minimal Width and Their Applications[J]. Cryptology ePrint Archive, 2025.

[49] Amy M, Maslov D, Mosca M, et al. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2013, 32(6): 818-830.

[50] Jaques S, Naehrig M, Roetteler M, et al. Implementing Grover oracles for quantum key search on AES and LowMC[C]//Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30. Springer International Publishing, 2020: 280-310.