Journal of King Saud University Computer and Information Sciences (2025) 37:55
https://doi.org/10.1007/s44443-025-00065-8

ORIGINAL PAPER

Efficient quantum circuit research and implementation for 16-bit S-boxes based on composite field

Lei Zhang[1] · Guoyuan Li[1] · Jingchen Dai[1] · Ruipeng Hong[1] · Chaoen Xiao[1] · Jianxin Wang[1] · Yifan Zhang[1]

Received: 18 March 2025 / Accepted: 18 May 2025
© The Author(s) 2025

Abstract
With the development of quantum computing technology, efficiently implementing the core component S-box of block ciphers in quantum environments has become an important research direction in the field of cryptographic engineering. Due to the limitations of the no-cloning theorem of quantum states, the implementation of S-boxes in quantum circuits faces the dual challenges of exponential growth in space complexity and excessive consumption of quantum resources, especially in 16-bit and larger S-boxes. To address these problems, this paper proposes a construction framework for 16-bit S-box quantum circuits based on composite fields. This method relies on normal bases representation to construct composite field structures, decomposing high-dimensional multiplicative inverse into low-dimensional subdomain operations through isomorphic mapping, thereby significantly reducing the quantum cost of nonlinear component. In the linear component, a dynamic optimal pivot selection strategy is proposed to effectively optimize the Gaussian elimination process while maintaining the row echelon form of the matrix, thereby reducing the number of CNOT gates. Taking the 16-bit S-box in the MK-3 algorithm as an example, experimental results show that the proposed method maintains correct circuit structure while reducing CNOT gate consumption in the linear component by more than 32.8%, and the utilization of quantum resources in the nonlinear component is more advantageous. This study provides theoretical support and an implementation path for the high-performance implementation of block ciphers in practical quantum computing environments.

Keywords Large S-box · Quantum circuit · Composite field · MK-3 · Optimize

1 Introduction

With the rapid advancement of quantum computing technology, quantum computers have demonstrated significant potential in solving complex computational problems and are gradually moving toward applications in general computing. Leveraging the powerful computational capabilities of quantum computing to efficiently execute cryptographic algorithms has become a focal point of cryptographic research. In today's information security field, block ciphers are widely applied in secure information transmission, data storage encryption, and communication protection for mobile and IoT devices (Abed et al. 2021; Rahman et al. 2022), but how to efficiently implement block ciphers on quantum circuits has gradually become a research hotspot (Jing et al. 2023). As the core nonlinear component of block ciphers, the cryptographic properties of the S-box directly determine the security strength and attack resistance of the algorithm. In the quantum computing environment, due to the constraints of the quantum no-cloning theorem, copying quantum states requires adding a large number of qubits, rendering traditional lookup table implementation methods ineffective. The quantum circuit implementation of S-boxes faces the dual challenges of exponential growth in space complexity and severe consumption of quantum resources (Lee et al. 2024). Therefore, in the research on the implementation of block cipher algorithms for quantum computing, how to efficiently and securely implement and optimize the S-box is particularly important, with the core scientific issues focusing on the design and optimization strategies of new quantum circuit architectures for S-boxes.

In the paradigm of quantum computing, mainstream approaches for S-box quantum circuit implementations rely on finite field theory to construct quantum circuit architectures (Canright 2005; Wood 2013; Stafford 2021; Liu et al. 2024; Saravanan and Kalpana 2018). The core mechanism transforms the nonlinear substitution mapping of S-boxes into algebraic expressions based on field operations

Extended author information available on the last page of the article

Published online: 02 June 2025

through the synergistic interaction of linear affine transformations and nonlinear multiplicative inverse in the finite field $GF(2^n)$. This enables the mathematical formalization of cryptographic permutations at the qubit level, providing theoretical support for efficient implementations in quantum environments. For optimization of the linear component's affine transformations, researchers commonly employ Gaussian elimination (Luo et al. 2022), PLU decomposition (Li et al. 2023, 2022; Chung et al. 2022), and permutation matrix methods (Li et al. 2024a), which systematically decompose matrix operations to reduce quantum gate counts and circuit depth. For the multiplicative inverse in the nonlinear component, research methods exhibit diverse characteristics: the Itoh-Tsujii algorithm based on finite field $GF(2^n)$ is applied for algebraic inverse calculations in quantum circuits (Luo et al. 2021; Grassl et al. 2016); tools like LIGHTER-R employ graph-based meet-in-the-middle algorithms to exhaustively enumerate inversion results (Li et al. 2023); the isomorphism between finite fields and composite fields is used to simplify the inversion process (Liu et al. 2024; Li et al. 2022; Saravanan and Kalpana 2018). These methods collectively promote significant optimizations in key metrics such as qubit count and T-depth for quantum S-box circuits, establishing both theoretical and technical foundations for the practical implementation of cryptographic quantum circuits.

Current quantum circuit research focuses on efficient implementation and resource optimization of 8-bit S-box structures (such as AES, SM4, etc.), primarily due to their dominant role in classical cryptographic designs (Liu et al. 2024; Li et al. 2022; Wang et al. 2022). However, with post-quantum cryptography's stringent requirements for quantum attack resistance, constructing large S-boxes of 16-bit and larger is emerging as a new research direction (Li et al. 2024b). In 2013, Wood (2013) constructed a 16-bit S-box based on finite field multiplicative inversion in composite field $GF((((2^2)^2)^2)^2)$ using polynomial bases and normal bases, reducing circuit area in FPGA implementations through operational module reuse. In 2015, this S-box was improved and applied to the authenticated encryption algorithm MK-3 based on a simplified duplex sponge construction (Kelly et al. 2015). In 2019, Xu et al. (2019) constructed a 16-bit S-box in the NBC algorithm using a 16-stage NFSR with 4 state-update functions through 20 iterations. In 2021, Stafford implemented the MK-3 algorithm's S-box on FPGAs (Stafford 2021) using polynomial bases in composite fields $GF((2^8)^2)$ and $GF(((2^4)^2)^2)$, capable of processing 8 or 4 inputs simultaneously to better adapt to and fully utilize the LUTs inside the FPGA. In 2022, Wu et al. (2023) designed a 4-round balanced two-branch Feistel-NFSR structure and obtained a 16-bit S-box through exhaustive search. In 2024, Wu et al. (2024) proposed a novel method for constructing 16-bit dynamic S-boxes by extending traditional 8-bit S-boxes and incorporating dynamic update mechanisms, combined with chaotic mappings and NFSR.

In current research, although optimization methods based on finite field theory and matrix decomposition have achieved significant progress in 8-bit S-box quantum circuit implementations, their extension to 16-bit S-boxes still faces challenges. 1. Existing approaches are constrained by the no-cloning theorem and inefficient elimination strategies, resulting in excessive quantum resource consumption in the linear component. 2. Existing methods struggle to balance the contradiction between the algebraic complexity of high-dimensional nonlinear operations and quantum resource consumption.

To address the aforementioned challenges, this paper makes the following contributions:

1. We propose a quantum circuit implementation framework based on normal bases construction for composite fields. Through an innovative co-design of the dynamic optimal pivot algorithm with computational optimization in composite field $GF(((2^4)^2)^2)$, we achieve an efficient construction of 16-bit S-box quantum circuits for the first time.
2. During matrix row operations in the linear component, the method detects consecutive identical element sequences and dynamically selects the row with the maximum consecutive identical elements as the pivot through a pivot selection strategy. By reducing redundant operations in row transformations, this approach significantly decreases the number of CNOT gates during elimination while maintaining the row echelon form structure that facilitates quantum implementation.
3. For the multiplicative inversion operation in the nonlinear component, the method employs an isomorphic mapping from finite field $GF(2^{16})$ to composite field $GF(((2^4)^2)^2)$. This hierarchical transformation decomposes high-dimensional inversion operations into lower-dimensional subfield operations, reducing quantum resource consumption while establishing an algebraic foundation for subsequent quantum circuit optimizations.

This paper takes the 16-bit S-box of the MK-3 algorithm as an example, and experimental results show that the proposed 16-bit S-box implementation consumes a total of 60 qubits, 116 Toffoli gates, and 622 CNOT gates. Compared with existing methods, the number of CNOT gates used in the linear component is reduced by more than 32.8%; the nonlinear component is more compact than equivalent existing modules, achieving superior overall performance.

The remainder of this paper is organized as follows. Section 2 reviews related work on quantum implementations of S-boxes, including their construction and implementation optimization. Section 3 presents a composite field-based 16-bit S-box construction designed for quantum circuits, describing the composite field $GF(((2^4)^2)^2)$ and the construction of the corresponding isomorphic matrix using normal bases. Section 4 provides detailed mathematical derivations and quantum circuit implementations for the key components of the 16-bit S-box. Section 5 presents and evaluates the implementation results of this paper. Section 6 concludes with a summary of contributions and discusses potential research directions.

2 Related work

2.1 S-box construction

In cryptography, the S-box, serving as the sole nonlinear component in block cipher algorithms, has maintained its design security and construction methods as a central research focus (Curlin et al. 2025; Abd-El-Atty 2023; Çavuşoğlu et al. 2017). Current S-box construction approaches primarily include finite field-based, structure-based, and special integer-based methodologies.

Finite field-based S-box constructions mainly use affine transformations and multiplicative inversion within finite fields to provide a complete algebraic expression describing the S-box mapping relationship. In 2005, Canright (2005) proposed the classical optimized implementation scheme for the AES algorithm's S-box using normal bases in the composite field $GF(((2^2)^2)^2)$. He mapped computations from the finite field $GF(2^8)$ to the composite field $GF(((2^2)^2)^2)$, simplifying operations within the composite field to obtain the results. In 2010, Boyar and Peralta (2010) redesigned the multiplicative inversion module over 16-element subfields within Canright's composite field structure, achieving a more compact implementation through novel SLP algorithms. In 2013, Wood (2013) constructed an S-box based on finite field multiplicative inversion in $GF((((2^2)^2)^2)^2)$ using polynomial bases and normal bases, following AES S-box design methodology. The composite field construction of this S-box is clear and allows extensive circuit reuse on FPGA implementations to reduce circuit area. In 2015, this S-box was improved and adopted in the MK-3 algorithm (Kelly et al. 2015). In 2021, Stafford (2021) implemented the MK-3 algorithm's S-box on FPGA using polynomial bases in the composite fields $GF((2^8)^2)$ and $GF(((2^4)^2)^2)$. This construction method can simultaneously process 8 or 4 inputs, thus better adapting to and fully utilizing FPGA internal LUTs. In 2023, Nandan and Rao (2023) constructed a more area-efficient AES S-box using dual-basis composite field expansion techniques in the composite fields $GF(((2^2)^2)^2)$ and $GF((2^4)^2)$.

Structure-based S-box constructions primarily employ specific cryptographic structures, such as Feistel structures, SPN structures, Lai-Massey structures, or their variants to create S-boxes. In 2014, Li and Wang (2014) constructed an 8-bit S-box with excellent cryptographic properties and low hardware implementation costs using a three-round Feistel structure. In 2019, Xu et al. (2019) constructed a 16-bit S-box using NFSR structure in the NBC algorithm to enhance both algorithm speed and security. In 2022, Wu et al. (2023) combined Feistel structure with NFSR structure to design a four-round balanced two-branch Feistel-NFSR structure for constructing 16-bit S-boxes. In 2024, Wu et al. (2024) proposed a new method for constructing 16-bit dynamic S-boxes by extending traditional 8-bit S-boxes and introducing a dynamic update mechanism, combined with chaotic mapping and NFSR.

In the field of S-box construction based on special integers, recent research has primarily focused on utilizing the unique properties of Gaussian integers, Eisenstein integers, and Quaternion integers to design more secure and efficient S-boxes. In 2023, Hazzazi et al. (2023) proposed an S-box design scheme based on Eisenstein integers that could generate a pair of S-boxes using a fixed set of parameters in resource-constrained environments, demonstrating the potential of this mathematical structure for enhancing cryptographic algorithm security. In 2024, Sajjad et al. (2024) designed a novel SPN architecture based on Gaussian integers for RGB image encryption and proposed a new method for constructing S-boxes using Gaussian integers, providing new ideas and technical approaches for image encryption. Also in 2024, Shah et al. (2024) published research on constructing S-boxes using Quaternion integers, implementing a design where each input corresponds to a pair of S-boxes through the algebraic properties of quaternions, significantly increasing computational complexity and enhancing cryptosystem security. In 2025, Sajjad et al. (2025) further extended the research on Gaussian integer-based S-boxes by designing an S-box scheme based on residue classes of Gaussian integers, expanding the application scope from single image encryption to multiple image encryption, marking a significant advancement in this direction. Currently, these methods mostly focus on the construction of 8-bit S-boxes; however, with appropriate parameter adjustments, they have shown potential to support the construction of 16-bit and larger large S-boxes, indicating an important research trend in future S-box design. Continued exploration in this field not only helps improve the security of existing cryptographic systems but also provides valuable theoretical foundations and technical approaches for the development of post-quantum cryptography.

2.2 S-box implementation optimization

In quantum circuit design, the primary optimization objectives are to minimize the number of qubits, quantum gates (particularly Toffoli and CNOT gates), and overall circuit depth (Liu et al. 2023; Haferkamp et al. 2022). Current research on quantum circuit implementations of S-boxes predominantly employs finite field-based constructions, focusing on optimizing affine transformations and inversion operations. Specifically, resource consumption is reduced by optimizing the linear component on one hand, and overall efficiency is improved by simplifying multiplication operations on the other (Boyar and Peralta 2010). By precisely adjusting these key parameters, not only can the performance of quantum circuits be significantly enhanced, but also a solid foundation can be provided for building efficient and secure post-quantum cryptographic systems.

For the implementation and optimization of linear components, the primary focus lies in optimizing linear operations, with particular emphasis on reducing the number of CNOT gates in linear computation circuits. Chung et al. (2022) and Li et al. (2023, 2022) utilized PLU decomposition, decomposing the original matrix into an upper triangular matrix, a lower triangular matrix, and a permutation matrix. After implementing quantum circuits for the upper and lower triangular matrices using CNOT gates, the permutation matrix can be realized by rewriting the indices of input and output bits, with its quantum resource overhead considered as zero. Luo et al. (2022) employed Gaussianelimination, where elementary row operations on matrices in modulo 2 can be equivalent to applying CNOT gates. By reducing the original matrix to an identity matrix through row operations and then reversing all the row operation steps, the quantum circuit can be obtained. Li et al. (2024a) adopted the permutation matrix method, proposing a greedy strategy-based matrix elimination approach that selects two rows that can eliminate the most 1s for each elimination step, ultimately obtaining a permutation matrix. The implementation of a matrix can also be viewed as solving a shortest linear program (SLP) problem along the path between each input and its corresponding output (Boyar et al. 2008). However, this method assumes that the circuit states both before and after the gates can be accessed by later points in the circuit, which is not directly compatible with the quantum computing paradigm (Chun et al. 2023).

The implementation and optimization of nonlinear components primarily focuses on reducing multiplicative complexity, with particular emphasis on minimizing AND gates (i.e., Toffoli gates) in nonlinear operations. In 2016, Grassl et al. (2016) employed the Itoh-Tsujii algorithm, constructing a quantum circuit for the AES S-box requiring 40 qubits through repeated applications of multiplication and squaring circuits. In 2018, Saravanan and Kalpana (2018) used tower field decomposition techniques to map elements from $GF(2^4)$ to the composite field $GF((2^2)^2)$ for inversion operations, then mapped them back to the finite field $GF(2^4)$, reducing the use of qubits and Toffoli gates. In 2020, Langenberg et al. (2019) adopted the combinatorial logic optimization techniques proposed by Boyar and Peralta (2010) to optimize inversion operations in the finite field $GF(2^4)$, reducing the use of Toffoli gates, and presented a quantum circuit for the AES algorithm S-box requiring 32 qubits. In 2023, Li et al. (2023) utilized the LIGHTER-R tool (Dasu et al. 2019) to treat the multiplication inversion operation in the finite field $GF(2^4)$ as a lookup table, optimizing nonlinear operations by searching for quantum circuits.

For 16-bit S-boxes, among current linear component optimization methods, PLU decomposition faces conflicts due to the constraints of the quantum no-cloning theorem, making implementation difficult without adding auxiliary qubits. Furthermore, PLU decomposition has been proven to generate a larger number of CNOT gates (Liu et al. 2024). Elimination strategies in Gaussian elimination and permutation matrix methods are inefficient for 16×16 matrices, resulting in higher consumption of CNOT gates. Current implementation and optimization of nonlinear components focus on 8-bit S-boxes, with no specific implementation schemes for components used in 16-bit S-boxes, making construction extension difficult. Meanwhile, existing finite field-based 16-bit S-box construction schemes demonstrate good implementation effects in FPGAs. However, in quantum circuits, these schemes face issues such as complex composite field constructions, high implementation costs for operation module quantum circuits, and numerous operation rounds, making it difficult to balance the contradiction between algebraic complexity of high-dimensional nonlinear operations and quantum resource consumption.

Therefore, this paper proposes a quantum circuit implementation framework based on normal bases for constructing composite fields. For the linear component, we employ a dynamic optimal pivot method to reduce redundant operations in row transformations, thereby decreasing CNOT gate counts during elimination. For the nonlinear component, through an isomorphic mapping from $GF(2^{16})$ to $GF(((2^4)^2)^2)$, we decompose high-dimensional inversion operations into low-dimensional subfield operations, significantly reducing quantum resource consumption.

3 16-bit S-box construction scheme for quantum circuits

To construct and implement a 16-bit S-box in quantum circuits, this paper adopts an S-box construction scheme based on normal bases for composite field implementation. This section presents the proposed scheme.

3.1 General construction scheme for large S-boxes

The traditional finite field construction of S-boxes processes the input through multiplicative inverse and affine transformation to generate the output, where the inversion may fail if the solution space is too large. The composite field method improves this by optimizing the inversion operation through isomorphic transformations between finite fields and composite fields. This paper proposes a general construction scheme for large S-boxes using the composite field method, which can be divided into two phases, with the process shown in Fig. 1.

The first phase involves constructing a composite field based on specific bases, ensuring it has an isomorphic relationship with the original finite field. By determining the values of the bases, isomorphic and inverse isomorphic matrices between the finite field and composite field are constructed, mapping the S-box input to the constructed composite field. The second phase computes the inverse of the S-box input in the composite field, where field extensions and subfields can be interconverted, transforming the inversion in the large field into a series of operations in subfields and smaller fields. Finally, the inverse result in the composite field is mapped back to the original finite field through the inverse isomorphic transformation, followed by an affine transformation to obtain the S-box output.The advantage of the composite field method lies in its clear algebraic structure and well-defined hierarchical relationships, which enable continuous decomposition of high-dimensional operations into low-dimensional subfield operations.

3.2 Composite field construction of 16-bit S-box

The implementation of the composite field method requires selecting a basis for constructing the composite field. This paper utilizes normal bases to construct the composite field $GF(((2^4)^2)^2)$ based on finite field extension theory. The theorem for finite field extension is presented below:

Theorem 1 *Lidl and Niederreiter (1997) Let $GF(p)$ be a finite field with p elements, and $Q(x)$ be an irreducible polynomial of degree n over $GF(p)$. Then the quotient ring $GF(p)/Q(x)$ can form a finite field with p^n elements.*

Theorem 1 provides the method of extending from a small field to a large field. In the construction of a composite field, the process is bottom-up, expanding from the small field to the large field. In contrast, the implementation of operations in the composite field involves transforming operations from the large field into a series of operations in the small field, which is a top-down implementation process.

Taking the 2^{16}-element composite field in this paper as an example, according to Theorem 1, the residue class $GF(2)/Q_1(x)$ can form a finite field. In $GF(2)$, the irreducible polynomial is uniquely determined, so we can directly specify $Q_1(x) = x^2 + x + 1$, with its two roots defined as normal bases W and W^2. Since the degree of $Q_1(x)$ is $n = 2$, through quadratic finite extension in $GF(2)$, we can construct the 4-element field $GF(2^2)$, where elements can be represented using normal bases as $x_1W^2 + x_0W$, where $x_0, x_1 \in GF(2)$. Similar to constructing finite field $GF(2^2)$, for composite field $GF((2^2)^2)$, we can determine the irreducible polynomial $Q_2(x) = x^2 + Ax + B$, with its roots denoted as normal bases Z and Z^4. Through quadratic extension over finite field $GF(2^2)$, we construct the 16-element field $GF((2^2)^2)$, whose elements can be represented as $x_1Z^4 + x_0Z$, where $x_0, x_1 \in GF(2^2)$. Similarly, composite fields $GF(((2^2)^2)^2)$ and $GF((((2^2)^2)^2)^2)$ can be derived using the same extension method. Elements in composite field $GF(((2^2)^2)^2)$ are represented as $x_1Y^{16} + x_0Y$, where $x_0, x_1 \in GF((2^2)^2)$, and elements in composite field $GF((((2^2)^2)^2)^2)$ are represented as $x_1T^{256} + x_0T$, where $x_0, x_1 \in GF(((2^2)^2)^2)$. This 2^{16}-element composite field structure consists of four layers of composite fields.

For a more compact composite field structure, this paper uses finite field $GF(2)$ to directly represent finite field $GF(2^4)$. Composite field $GF((2^2)^2)$ and finite field

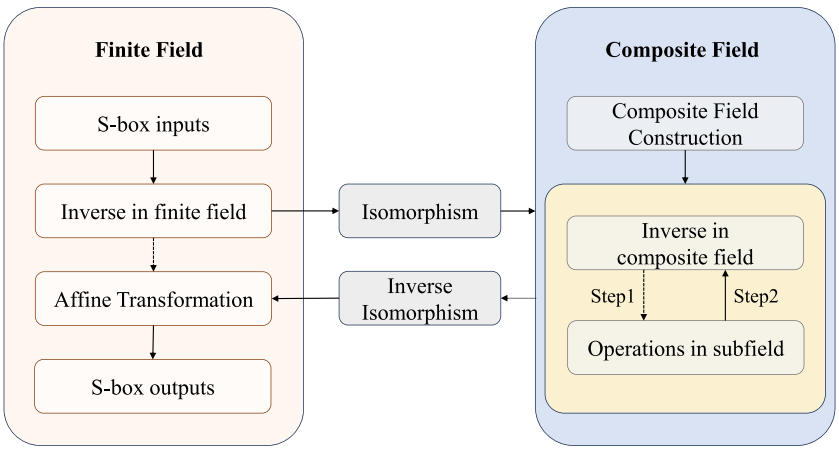


Fig. 1 Composite Field Method Flow

$GF(2^4)$ are isomorphic, and their elements can be derived through conversion. Elements $x_1Z^4+x_0Z$ in composite field $GF((2^2)^2)$, where $x_0, x_1 \in GF(2^2)$, can be converted by using the base field $GF(2)$ to transform x_0, x_1 over $GF(2^2)$ into $x_1W^2+x_0W$ and $x_3W^2+x_2W$, where $x_0, x_1, x_2, x_3 \in GF(2)$. This allows us to derive the result of finite field $GF(2^4)$ being directly represented by $GF(2)$, while keeping the remaining extension structure unchanged. We can convert the original four-layer composite field structure $GF(2^2) - GF((2^2)^2) - GF(((2^2)^2)^2) - GF((((2^2)^2)^2)^2)$ into a three-layer composite field structure $GF(2^4) - GF((2^4)^2) - GF(((2^4)^2)^2)$.The construction parameters for composite field $GF(((2^4)^2)^2)$ are shown in Table 1.

According to finite field extension theory, elements in $GF(((2^4)^2)^2)$ can be represented through normal bases as linear combinations of elements in the subfield $GF((2^4)^2)$, thereby transforming operations in $GF(((2^4)^2)^2)$ into a series of operations in the subfield $GF((2^4)^2)$. By decomposing high-dimensional operations into lower-dimensional operations layer by layer, the algebraic complexity of operations can be reduced. Higher-dimensional operations can be implemented layer by layer through the implementation of lower-dimensional operations. In the irreducible polynomials for composite field construction in this paper, the first-degree term coefficients are specified as $A, C, E = 1$, and the constant terms as $B = W^2$, $D = B^2Z$, $F = DY$.

3.3 Isomorphic matrix construction of 16-bit S-box

Theorem 2 *Lidl and Niederreiter (1997) If both F_1 and F_2 are n-th degree extensions of the finite field F, and their primitive polynomials over F are f_1 and f_2 respectively, then there are exactly n isomorphic matrices between the fields F_1 and F_2.*

Previous work has proven that if two fields have the same number of elements, then the fields are isomorphic, and an isomorphic transformation must exist. The number of isomorphic matrices in an isomorphic transformation is guaranteed by Theorem 2. This paper chooses to map finite field $GF(2^{16})$ to composite field $GF(((2^4)^2)^2)$, which has the same number of elements as the original finite field, satisfying the isomorphism condition.

For a 16-bit element p in the finite field $GF(2^{16})$, its representation using polynomial bases is given by the (1):

$$p = p_{15}A^{15} + p_{14}A^{14} + \cdots + p_0A^0, \quad p_i \in \{0,1\},\ A^i \in \{0,1\} \tag{1}$$

To leverage the compactness of composite fields and the computational simplicity brought by element transformation, this paper chooses to compute by mapping elements represented in polynomial bases in finite field $GF(2^{16})$ to elements represented in normal bases in composite field $GF(((2^4)^2)^2)$. Through the example in Section 3.2, a 16-bit element $n = n_1T^{256}+n_0T$, where $n_0, n_1 \in GF((2^4)^2)$, represented in normal bases in composite field $GF(((2^4)^2)^2)$, can be transformed using elements and bases in subfield $GF((2^4)^2)$ as $n = (n_3Y^{16}+n_2Y)T^{256}+(n_1Y^{16}+n_0Y)T$, where $n_0, n_1, n_2, n_3 \in GF(2^4)$. Subsequently, by transforming element n through $GF(2^4)$ elements and basis operations, we ultimately derive (2) after simplification.

$$n = n_{15}W^2Z^4Y^{16}T^{256} + n_{14}WZ^4Y^{16}T^{256} + \cdots + n_0WZYT, n_i \in \{0,1\} \tag{2}$$

When an element p in finite field $GF(2^{16})$ is mapped to element n in composite field $GF(((2^4)^2)^2)$ through an isomorphic mapping, the mapping relationship can be established by combining (1) and (2):

$$\begin{bmatrix} n_{15} \\ n_{14} \\ \vdots \\ n_0 \end{bmatrix} = PN \begin{bmatrix} p_{15} \\ p_{14} \\ \vdots \\ p_0 \end{bmatrix} \begin{bmatrix} p_{15} \\ p_{14} \\ \vdots \\ p_0 \end{bmatrix} = NP \begin{bmatrix} n_{15} \\ n_{14} \\ \vdots \\ n_0 \end{bmatrix} \tag{3}$$

Equation 3 provides two mapping chains: the element p represented in polynomial bases is mapped to element n represented in normal bases through isomorphic matrix PN, and

Table 1 Construction Parameters of the Composite Field $GF(((2^4)^2)^2)$

Composite Field	Irreducible Polynomial	Element Representation	Normal Bases
$GF(2^2)$	$Q_1(x) = x^2 + x + 1$	$x_1W^2 + x_0W$, $x_0, x_1 \in GF(2)$	W, W^2
$GF(2^4)$	$Q_2(x) = x^2 + Ax + B$	$(x_3W^2 + x_2W)Z^4 + (x_1W^2 + x_0W)Z$, $x_0, x_1, x_2, x_3 \in GF(2)$	Z, Z^4
$GF((2^4)^2)$	$Q_3(x) = x^2 + Cx + D$	$x_1Y^{16} + x_0Y$, $x_0, x_1 \in GF(2^4)$	Y, Y^{16}
$GF(((2^4)^2)^2)$	$Q_4(x) = x^2 + Ex + F$	$x_1T^{256} + x_0T$, $x_0, x_1 \in GF((2^4)^2)$	T, T^{256}

element n is mapped back to element p through inverse isomorphic matrix NP. The 16 coefficients of element n are all derived from the multiplication of normal bases, which are a set of specific values determined during the construction of the composite field. After representing each coefficient in hexadecimal, a 16×16 inverse isomorphic matrix NP can be obtained. The specific values of the normal bases selected in this paper are as follows:

- $W^2 = 0x0733$; $W = 0x0732$
- $Z^4 = 0x8f9e$; $Z = 0x8f9f$
- $Y^{16} = 0xb0df$; $Y = 0xb0de$
- $T^{256} = 0xe5ef$; $T = 0xe5ee$

In finite fields, the isomorphism matrix PN can be obtained by inverting the inverse isomorphism matrix NP. Their specific values are provided in Appendix A. In this paper's matrix representation, elements with the largest row and column indices are placed in the leftmost column and topmost row of the matrix. However, quantum computing conventions typically place elements with the smallest row and column indices in the leftmost column and topmost row. Therefore, to align with quantum computing standards, the computed matrices NP and PN in this paper have undergone row and column order reversal.

4 Quantum circuit implementation of key components

This paper implements a composite-field-constructed 16-bit S-box employing the dynamic optimal pivot algorithm to eliminate redundant operations in 16×16 matrix row transformations for the linear component, while decomposing high-dimensional multiplicative inversion into low-dimensional subfield operations for the nonlinear component. This chapter provides detailed explanations for both components.

4.1 Linear component

The linear component comprises matrix operations of affine transformations and isomorphism mappings in the 16-bit S-box, where the computational results can be viewed as linear combinations of elements in column vectors. For 16-bit S-boxes, existing methods are limited by the quantum no-cloning theorem and inefficient elimination strategies, resulting in poor performance when implementing 16×16 matrices. This paper proposes a new elimination strategy using dynamic pivot selection based on Gaussian elimination, achieving an effective implementation for 16×16 matrix operations.

The dynamic optimal pivot algorithm takes a 16×16 matrix as input and converts it to a 16×16 identity matrix through a row transformation elimination strategy established by the algorithm. In this process, the pivot column i is defined as the index of the current target column for elimination. After completing one round of elimination operations on pivot column i, the i-th column of the matrix will contain the value 1 in only one row, while the i-th column elements in all other rows will be 0. Key terminology is defined as follows:

A consecutive-1 sequence of length n: Refers to an element sequence in a matrix row starting from pivot column i, where positions $i, i+1, i+2, ..., i+n-1$ consecutively contain the value 1, with n denoting the sequence length.

Consecutive identical sequence: Refers to the longest contiguous sequence where two rows share identical element values (either 0 or 1) in corresponding column positions starting from pivot column i during comparative analysis. It should be specifically noted that having identical elements only at pivot column i does not constitute a consecutive identical sequence.

To clarify the above definitions, suppose the first three rows of matrix A are as follows, with the current pivot column $i = 0$. Among these three rows, Row_0 has the longest consecutive-1 sequence with length $n = 6$. Row_1 and Row_2 both have consecutive-1 sequences of length $n = 2$. The consecutive identical sequence between Row_0 and Row_1 is [1, 1], between Row_0 and Row_2 is [1, 1], and between Row_1 and Row_2 is [1, 1, 0, 1, 1, 0, 0, 1]. Therefore, among these three rows, Row_1 and Row_2 have the longest consecutive identical sequence.

Row_0 = [1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0]
Row_1 = [1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1]
Row_2 = [1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0]

The algorithm flow of the dynamic optimal pivot algorithm consists of two main phases, with the detailed process illustrated in Fig. 2.

Non-pivot column elimination: The non-pivot column elimination procedure operates on rows 0 to $i-1$ of the matrix. Although these rows have undergone preliminary elimination processing in previous pivot columns, after several rounds of elimination, former non-pivot columns now become pivot columns, and there may still be uneliminated elements 1 in the current pivot column, necessitating supplementary elimination operations. The algorithm first traverses the specified row range to identify rows with value 1 in pivot column i, then incorporates them into the Non-pivot Column set NC. Subsequently, it progressively compares each row in set NC against all rows in the Candidate Row set CR to detect the longest consecutive identical sequence. Select two rows containing the longest consecutive identical sequence, use the rows in CR to perform elimination operations on the rows in NC, and remove the eliminated rows from NC.

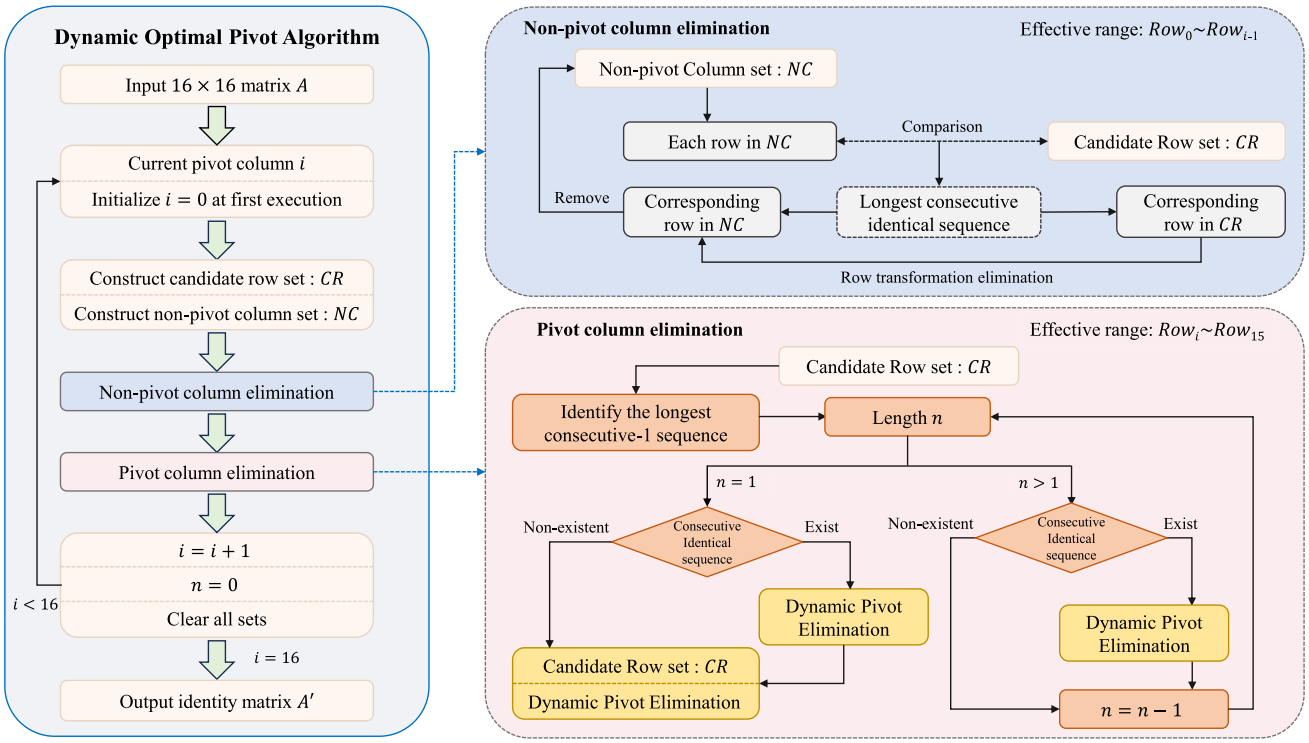


Fig. 2 Flowchart of the Dynamic Optimal Pivot Algorithm

Pivot column elimination: Pivot column elimination implements elimination operations for rows i to 15 in the matrix. First, the algorithm traverses the specified row range to identify rows with value 1 in pivot column i, incorporating them into the Candidate Row set CR. The set CR needs to complete the non-pivot column elimination part at this stage before continuing with subsequent operations. For all rows in set CR, the algorithm identifies the longest consecutive-1 sequence and records its length as n. When $n > 1$, the algorithm determines whether consecutive identical sequences exist among consecutive-1 sequences of length n; if they exist, dynamic pivot elimination is performed on these rows, the eliminated rows are removed from set CR, and $n = n-1$; if they do not exist, n is directly set to $n-1$. When n decreases to 1, the algorithm continues to determine whether rows with consecutive identical sequences still exist in set CR; if they do, dynamic pivot elimination is prioritized for them until no consecutive identical sequences remain in set CR. Subsequently, dynamic pivot elimination is implemented for the entire set CR. Finally, the pivot row PR is swapped with row i. After one round of elimination, $i = i + 1$, meaning the pivot column moves to the right, and all row sets used are emptied, with n reset to 0. The dynamic pivot elimination algorithm is shown in Algorithm 1.

In quantum circuit implementation, all elimination steps need to be reversed, with one CNOT gate representing one row transformation elimination. Therefore, the dynamic optimal pivot algorithm significantly reduces the use of CNOT gates by decreasing the number of elimination operations. The performance comparison results between the dynamic optimal pivot algorithm proposed in this paper and the Gaus-

Algorithm 1 Dynamic Pivot Elimination Algorithm.

Input: Row set S containing the longest consecutive identical sequence
Output: Optimal Pivot Row PR
Starting column i of the longest consecutive sequence
Termination column j of the longest consecutive sequence
Initialize the candidate optimal pivot row set CPR as an empty set
Initialize the pivot row set PR as an empty set
Maximum next 1 column index $M\_1 = -1$
for each row $r \in S$ **do**
  Next 1 column index N_1 = Find next 1 column starting from column j
  if $N\_1 > M\_1$ **then**
    $N\_1 = M\_1$, $CPR = \{r\}$
  else if $N\_1 = M\_1$ **then**
    add r to CPR
  end if
  if size of $CPR == 1$ **then**
    PR = sole row r in set CPR
  else
    Create new set and recursively continue optimal pivot row selection
  end if
  if $r! = PR$ **then**
    Row_Elimination(PR, r)
  end if
end for

sian elimination method and permutation matrix method are shown in Table 5. The implementation of linear component is provided in Appendix B. Experiments demonstrate that this paper's algorithm significantly improves the resource utilization efficiency of 16×16 matrices in quantum circuit implementation.

4.2 Nonlinear component

The nonlinear component of the S-box refers to multiplicative inverse in finite field $GF(2^{16})$, which is mapped to inversion in composite field $GF(((2^4)^2)^2)$ through isomorphic transformation. Due to the conversion relationships in composite fields, operations in $GF(((2^4)^2)^2)$ can be simplified into operations in its subfields.

4.2.1 Operations on $GF(((2^4)^2)^2)$

As known from Section 3.2, elements a and b in the composite field $GF(((2^4)^2)^2)$ can be represented by the two roots of the irreducible polynomial, namely the normal bases T^{256} and T. The irreducible polynomial in the composite field $GF(((2^4)^2)^2)$ is $Q_4(x) = x^2 + Ex + F$, with trace $T^{256} + T = E$ and norm $T(T^{256}) = F$. Thus, the trace can be simplified to:

$$1 = E^{-1}(T + T^{256}) \tag{4}$$

If a and b are inverses of each other, the definition of inverse yields (5).

$$\begin{aligned} ab &= (a_1T^{256} + a_0T)(b_1T^{256} + b_0T) \\ &= (a_1b_1)(T^{256})^2 + (a_1b_0 + a_0b_1)(T^{256})T \\ &\quad + (a_0b_0)T^2 \\ &= (a_1b_1)(ET^{256} + F) + (a_1b_0 + a_0b_1)F \\ &\quad + (a_0b_0)(ET + F) \\ &= (a_1b_1E)T^{256} + (a_0b_0E)T \\ &\quad + (a_1b_1 + a_1b_0 + a_0b_1 + a_0b_0)F \\ &= (a_1b_1E)T^{256} + (a_0b_0E)T \\ &\quad + [(a_1 + a_0)(b_1 + b_0)F]E^{-1}(T^{256} + T) \\ &= [(a_1b_1E) + (a_1 + a_0)(b_1 + b_0)FE^{-1}]T^{256} \\ &\quad + [(a_0b_0E) + (a_1 + a_0)(b_1 + b_0)FE^{-1}]T \\ &= 1 \end{aligned} \tag{5}$$

Combining (4) and (5) yields:

$$\begin{cases} E^{-1} = (a_1b_1E) + (a_1 + a_0)(b_1 + b_0)FE^{-1} \\ E^{-1} = (a_0b_0E) + (a_1 + a_0)(b_1 + b_0)FE^{-1} \\ \qquad\qquad \Rightarrow a_1b_1 + a_0b_0 = 0 \end{cases} \tag{6}$$

Multiplying both sides of (6) by a_0E and simplifying yields:

$$\begin{aligned} a_0 &= (a_1a_0b_1)E^2 + (a_1a_0b_0 + a_0{}^2b_1)F \\ &= (a_1a_0b_1)E^2 + (a_1{}^2b_1 + a_0{}^2b_1)F \\ &= [(a_1a_0)E^2 + (a_1{}^2 + a_0{}^2)F]b_1 \end{aligned} \tag{7}$$

Similarly, $a_1 = [(a_1a_0)E^2 + (a_1{}^2 + a_0{}^2)F]b_0$. This paper sets the trace $E = 1$, and by organizing the results, the inverse element b of the 16-bit element a in the composite field $GF(((2^4)^2)^2)$ can be obtained, as shown in (8).

$$\begin{aligned} b_1 &= [(a_1a_0) + (a_1{}^2 + a_0{}^2)F]^{-1}a_0 \\ b_0 &= [(a_1a_0) + (a_1{}^2 + a_0{}^2)F]^{-1}a_1 \end{aligned} \tag{8}$$

From (8), we can derive the quantum circuit diagram for the inversion in the composite field $GF(((2^4)^2)^2)$, which can be transformed into three operational sub-modules on the subfield $GF((2^4)^2)$: multiplication, square-scale F, and inversion, where F is the scaling factor of the scaling operation. Figure 3 is scaled for 8 qubits, where the CNOT gates perform CNOT operations on each qubit.

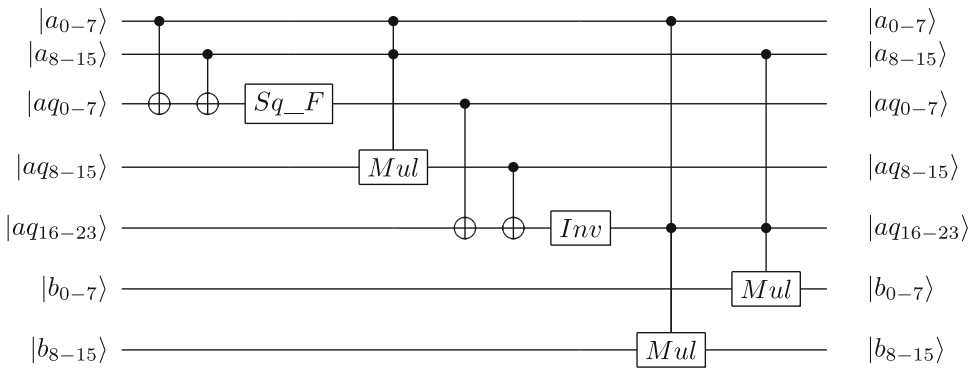


Fig. 3 Quantum Circuit Diagram for Multiplicative Inversion in $GF(((2^4)^2)^2)$

4.2.2 Operations on $GF((2^4)^2)$

Since there are transformation relationships for operations in the composite field, operations in the larger and smaller fields can be transformed into each other. Therefore, operations in $GF((2^4)^2)$ can also be transformed into operations in the subfield $GF(2^4)$.

Multiplication operation

The implementation steps for multiplication are similar to those for inversion. If elements c and d exist in the composite field $GF((2^4)^2)$, their irreducible polynomial is $Q_3(x) = x^2 + x + D$, with the scaling factor D.

$$\begin{aligned} cd &= (c_1Y^{16} + c_0Y)(d_1Y^{16} + d_0Y) \\ &= [(c_1d_1) + (c_1 + c_0)(d_1 + d_0)D]Y^{16} \\ &\quad + [(c_0d_0) + (c_1 + c_0)(d_1 + d_0)D]Y \end{aligned} \tag{9}$$

The results indicate that the multiplication operation in the composite field $GF((2^4)^2)$ under the normal bases representation can be transformed into multiplication and scale D operations on the subfield. The quantum circuit diagram is shown in Fig. 4, scaled to 4 qubits.

Scaling operation

For the scaling operation, the scaling factor is the norm of the irreducible polynomial of each layer of the composite field, which can be determined as a constant value by selecting the coefficients. The scaling factor F in the composite field $GF((2^4)^2)$ is the norm of the irreducible polynomial $Q_4(x)$, and it lies in the same composite field as the element c in $GF((2^4)^2)$. Both can be represented using a normal bases. Thus, the square-scale F can be expressed as:

$$\begin{aligned} c^2F &= (c_1Y^{16} + c_0Y)^2(f_1Y^{16} + f_0Y) \\ &= [f_1{c_1}^2 + f_0D({c_1}^2 + {c_0}^2)]Y^{16} \\ &\quad + [f_0{c_0}^2 + f_1D({c_1}^2 + {c_0}^2)]Y \end{aligned} \tag{10}$$

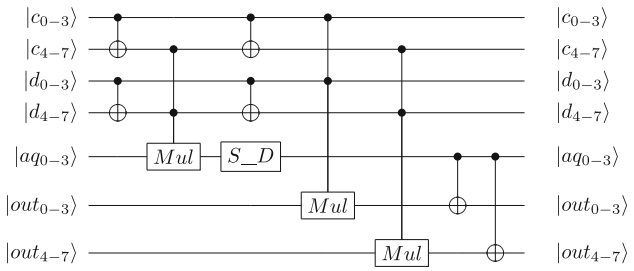


Fig. 4 Quantum Circuit Diagram for Multiplication in $GF((2^4)^2)$

Different values of the norm F will affect the quantum resource consumption during implementation, which will be discussed in the compatibility experiments section. Here, only the value is given. From the perspective of serializing the composite field's upper and lower layers and the compact circuit, this paper chooses $f_1 = 0$, $f_0 = D$, and thus $F = DY$. Therefore, $c^2F = [D^2(c_1 + c_0)^2]Y^{16} + [Dc_0{}^2]Y$, this indicates that the square-scale F in the composite field $GF((2^4)^2)$ can be composed of the square-scale D in the subfield and the square-scale D^2, with its quantum circuit shown in Fig. 5.

Inversion operation

The steps for multiplicative inversion in the composite field $GF((2^4)^2)$ are similar to those in $GF(((2^4)^2)^2)$, and its quantum circuit diagram is shown in Fig. 6.

4.2.3 Operations on $GF(2^4)$

After the operations in the 2^{16}-element field are transformed to the 2^4-element field, some operations can be performed directly without further transforming to the smaller field $GF(2^2)$, as long as computational capabilities allow. Therefore, this paper optimizes the operations by directly representing the elements of the composite field $GF(2^4)$ using elements from the base field $GF(2)$.

Multiplication operation

In $GF(2^4)$, the following transformation relationship of the normal bases can be determined from the properties of the irreducible polynomial and the normal bases:

– $Z^2 + Z = W^2$, $Z^3 = W(Z^2 + 1)$, $Z^4 = Z^2 + W$, $Z^5 = B$, $Z^7 = Z^2W^2$, $Z^8 = Z^2 + 1$, $Z^{10} = Z^4 + Z^2$

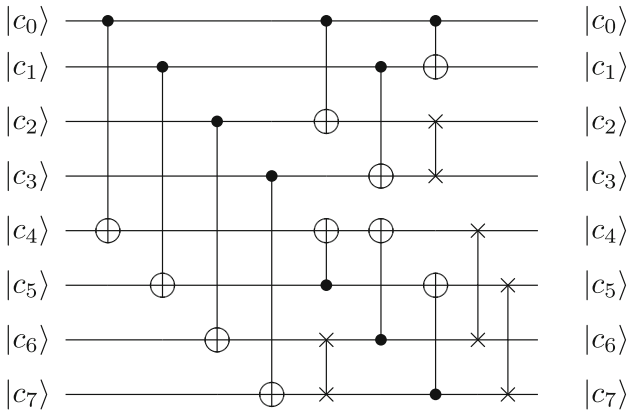


Fig. 5 Quantum Circuit for Square-Scale F in $GF((2^4)^2)$

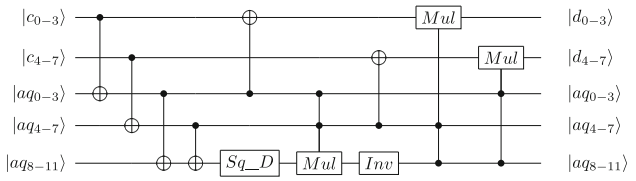


Fig. 6 Quantum Circuit Diagram for Multiplicative Inversion in $GF((2^4)^2)$

– $B = W^2, B^2 = W, B^3 = 1$

Any two elements e, f can be represented as:

$$\begin{cases} e = e_3W^2Z^4 + e_2WZ^4 + e_1W^2Z + e_0WZ \\ f = f_3W^2Z^4 + f_2WZ^4 + f_1W^2Z + f_0WZ \end{cases} \tag{11}$$

The multiplication result, combined with the normal bases transformation relation, can be derived as:

$$\begin{aligned} ef = &(ef)_3W^2Z^4 + (ef)_2WZ^4 \\ &+ (ef)_1W^2Z + (ef)_0WZ \end{aligned} \tag{12}$$

Among them, $(ef)_3$, $(ef)_2$, $(ef)_1$, $(ef)_0$ are respectively:

$$\begin{aligned} (ef)_3 &= (e_0+e_1+e_2+e_3)(f_0+f_1+f_2+f_3)+ \\ &\quad (e_0+e_2)(f_0+f_2)+(e_2+e_3)(f_2+f_3)+e_3f_3 \\ (ef)_2 &= (e_0+e_2)(f_0+f_2)+(e_1+e_3)(f_1+f_3) \\ &\quad +(e_2+e_3)(f_2+f_3)+e_2f_2 \\ (ef)_1 &= (e_0+e_1+e_2+e_3)(f_0+f_1+f_2+f_3)+ \\ &\quad (e_0+e_1)(f_0+f_1)+(e_0+e_2)(f_0+f_2)+e_1f_1 \\ (ef)_0 &= (e_0+e_1)(f_0+f_1)+(e_0+e_2)(f_0+f_2) \\ &\quad +(e_1+e_3)(f_1+f_3)+e_0f_0 \end{aligned} \tag{13}$$

The quantum circuit for multiplication in $GF(2^4)$ is shown in Fig. 7 according to the equation. It requires a total of 12 qubits, 9 Toffoli gates, and 27 CNOT gates.

Scaling operation

In the composite field $GF(2^4)$, there are three scaling operations: scale D, square-scale D, and square-scale D^2, where $D = B^2Z$.

In the composite field $GF(2^4)$, there are three scaling operations: scale D, square-scale D, and square-scale D^2,

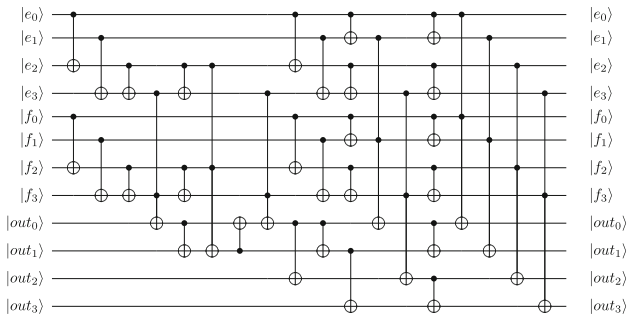


Fig. 7 Quantum Circuit for Multiplication in $GF(2^4)$

where $D = B^2Z$.

$$\begin{aligned} eD &= [(e_3W^2 + e_2W)Z^4 \\ &\quad + (e_1W^2 + e_0W)Z]B^2Z \\ &= [(e_3 + e_1)W^2 + (e_2 + e_0)W]Z^4 \\ &\quad + [(e_3 + e_0)W^2 + (e_2 + e_1 + e_0)W]Z \end{aligned} \tag{14}$$

The result of scale D is given by (14), and its quantum circuit implementation is shown in Fig. 8a, requiring a total of 4 qubits, 5 CNOT gates, and 1 Swap gate.

$$\begin{aligned} e^2D &= [(e_3W^2 + e_2W)Z^4 \\ &\quad + (e_1W^2 + e_0W)Z]^2B^2Z \\ &= [(e_2 + e_0)W^2 + (e_3 + e_1)W]Z^4 \\ &\quad + [(e_1 + e_0)W^2 + (e_0)W]Z \end{aligned} \tag{15}$$

$$\begin{aligned} e^2D^2 &= [(e_3W^2 + e_2W)Z^4 \\ &\quad + (e_1W^2 + e_0W)Z]^2(B^2Z)^2 \\ &= [(e_2 + e_1)W^2 + (e_3 + e_1 + e_0)W]Z^4 \\ &\quad + [e_2W^2 + e_3W]Z \end{aligned} \tag{16}$$

The results of square-scale D and square-scale D^2 can be obtained by calculation as (15) and (16), and their quantum circuits are shown in Fig. 8b and c, respectively. Square-scale D requires a total of 4 qubits, 3 CNOT gates, and 1 Swap gate. Square-scale D^2 requires a total of 4 qubits, 3 CNOT gates, and 3 Swap gates.

Inversion operation

This paper treats the inverse operation in $GF(2^4)$ as an input-output mapping in a lookup table, which is then viewed as a 4×4 S-box. The quantum circuit for this lookup table is implemented using the DORCIS tool (Chun et al. 2023).

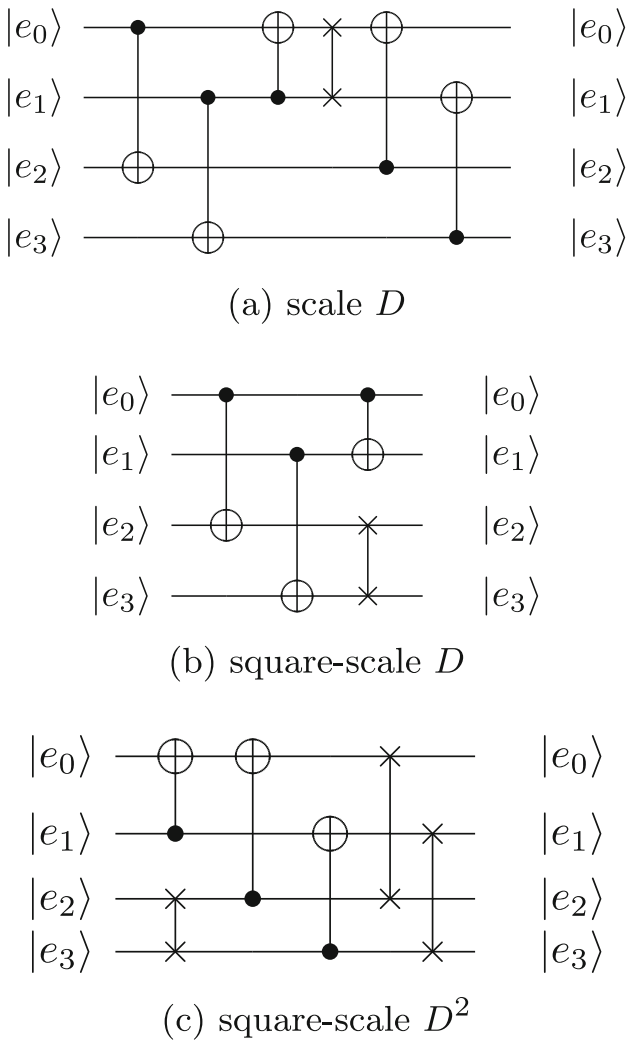


Fig. 8 Quantum Circuit for Scaling Operations in $GF(2^4)$

The DORCIS tool can search for quantum circuits of 4×4 or 3×3 S-boxes, and its circuit depth or T-depth is lower compared to those generated by the LIGHTER-R tool. The inverse operation in $GF(2^4)$ is represented as a 4×4 S-box, with the input-output mapping shown in Table 2.

Based on the lookup table provided in Table 2, the quantum circuit generated by the DORCIS tool is shown in Fig. 9, which includes a CCCNOT gate. According to Li et al. (2023), a CCCNOT gate requires 5 qubits and can be implemented with 4 Toffoli gates, as shown in Fig. 10. Therefore, implementing the quantum circuit shown in Fig. 9 requires 5 qubits, 8 Toffoli gates, and 6 CNOT gates.

Table 2 Lookup Table for Inversion in $GF(2^4)$

Elements															
input	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
output	0	c	8	4	3	a	7	6	2	d	5	e	1	9	b	f

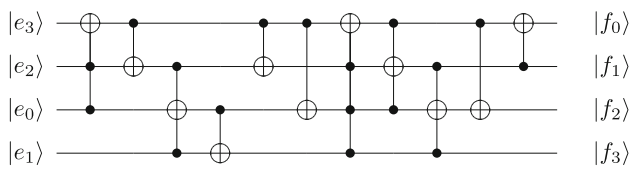


Fig. 9 Quantum Circuit for Multiplicative Inversion in $GF(2^4)$

5 Results evaluation

This paper proposes a 16-bit large S-box construction over composite field $GF(((2^4)^2)^2)$ using normal bases. We implemented and optimized the 16-bit S-box of MK-3 algorithm in quantum circuits. To evaluate the effectiveness, we compared our scheme with existing implementations. We also generated new 16-bit S-boxes to test the generality of our scheme.

5.1 Implementation platform

This paper implements quantum circuits using the open-source quantum computing framework ProjectQ (version 0.8.0) (Steiger et al. 2018) developed by ETH Zurich, which is based on Python (version 3.11.9). We use this framework for simulation and resource estimation of our quantum circuit implementation. We use the ClassicalSimulator in the ProjectQ framework for simulation and the ResourceCounter for resource counting. The ClassicalSimulator only executes classical gate operations. These operations are deterministic and do not generate quantum superposition or entanglement. The operations have perfect fidelity with 100% reliability (Jones et al. 2019). The ResourceCounter simulator does not

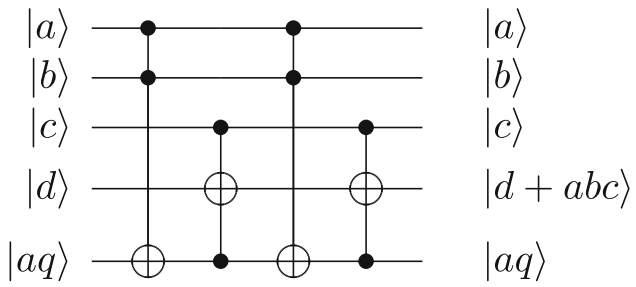


Fig. 10 Quantum Circuit of CCCNOT Gate

actually perform quantum computation simulation; instead, it simply counts and tallies the number of various quantum gate operations.

The MK-3 algorithm is an authenticated encryption algorithm that employs a simplified duplex sponge structure, containing 32 16-bit S-boxes. The irreducible polynomial used for its S-box is: $P(x) = x^{16} + x^5 + x^3 + x + 1$, with the construction formula shown in (17). The S-box consists of an affine transformation combined with inversion in a finite field. The affine transformation matrix is denoted as matrix A, and the column vector is denoted as c in this paper.

$$S(x) = \begin{bmatrix} 0&0&1&0&0&0&0&1&0&0&1&1&1&1&1&0 \\ 1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&0 \\ 1&1&0&0&1&0&1&1&0&1&0&1&0&0&1&1 \\ 1&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0 \\ 1&1&0&0&0&1&1&0&0&1&1&1&1&0&1&1 \\ 0&1&0&0&0&0&1&1&0&1&1&1&1&1&0&1 \\ 0&0&1&0&1&0&1&0&1&1&0&0&1&1&0&0 \\ 1&0&1&1&1&0&1&1&0&0&0&1&0&1&1&1 \\ 0&1&0&0&0&0&0&0&1&0&0&1&1&1&0&1 \\ 1&0&1&1&0&0&0&1&0&0&1&0&1&0&0&0 \\ 1&0&1&0&0&1&1&1&0&0&1&1&0&1&0&0 \\ 1&0&1&1&1&0&1&1&1&1&0&1&1&0&0&1 \\ 1&0&1&0&0&1&0&1&1&0&0&1&0&0&0&1 \\ 0&1&0&0&0&1&1&1&1&0&0&0&0&0&0&1 \\ 1&0&0&0&1&1&0&1&0&1&1&1&1&0&0&0 \\ 1&1&0&1&0&1&1&0&1&0&0&1&1&0&0&0 \end{bmatrix} \begin{bmatrix} x_{15} \\ x_{14} \\ x_{13} \\ x_{12} \\ x_{11} \\ x_{10} \\ x_9 \\ x_8 \\ x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix}^{-1} \oplus \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{17}$$

5.2 Implementation in quantum circuits

According to the discussion in Section 3, the implementation of the 16-bit S-box can be divided into three steps:

1. Divide the entire S-box circuit into 11 submodules using the constructed composite field.
2. Perform optimization design for each submodule based on algebraic methods or algorithms.
3. Cascade the optimized submodules according to their logical relationships.

The schematic diagram of the 16-bit large S-box quantum implementation circuit in this paper is shown in Fig. 11. In the figure, using 8 qubits as the scale, $x_0 \sim x_7$ and $x_8 \sim x_{15}$ are 8-bit quantum registers, representing the lower 8 bits and higher 8 bits of input x. The output after processing through the S-box is represented as y. To reduce the number of CNOT gates, this paper combines the isomorphic inverse matrix with the affine transformation matrix, denoted as NP_A. The operations in the figure represent computations over $GF((2^4)^2)$, where PN is the isomorphic matrix, Sq_F is the square scaling F, Mul is multiplication, Inv is multiplicative inversion, and NP_A is the isomorphic inverse_affine matrix. c is the column vector of the affine transformation, which can be implemented by adding NOT gates at the corresponding positions.

This paper uses the ResourceCounter simulator to measure the quantum circuit resource consumption for each part and the overall implementation of the 16-bit large S-box in the MK-3 algorithm, with results shown in Table 3.

The standard input-output pairs of the 16-bit S-box in the MK-3 algorithm contains 2^{16} elements, which cannot be enumerated exhaustively in this paper. Therefore, this paper selected four sets of input-output results as test cases, which are simulated using the ClassicalSimulator and displayed as shown in Fig. 12. By comparing with the standard input-output pairs of the MK-3 algorithm, we demonstrate that our quantum implementation of the S-box correctly performs its intended function.

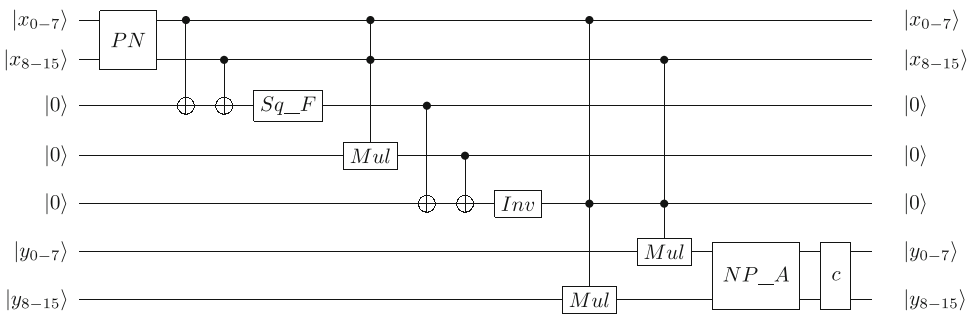


Fig. 11 Quantum Circuit Diagram for MK-3 S-box

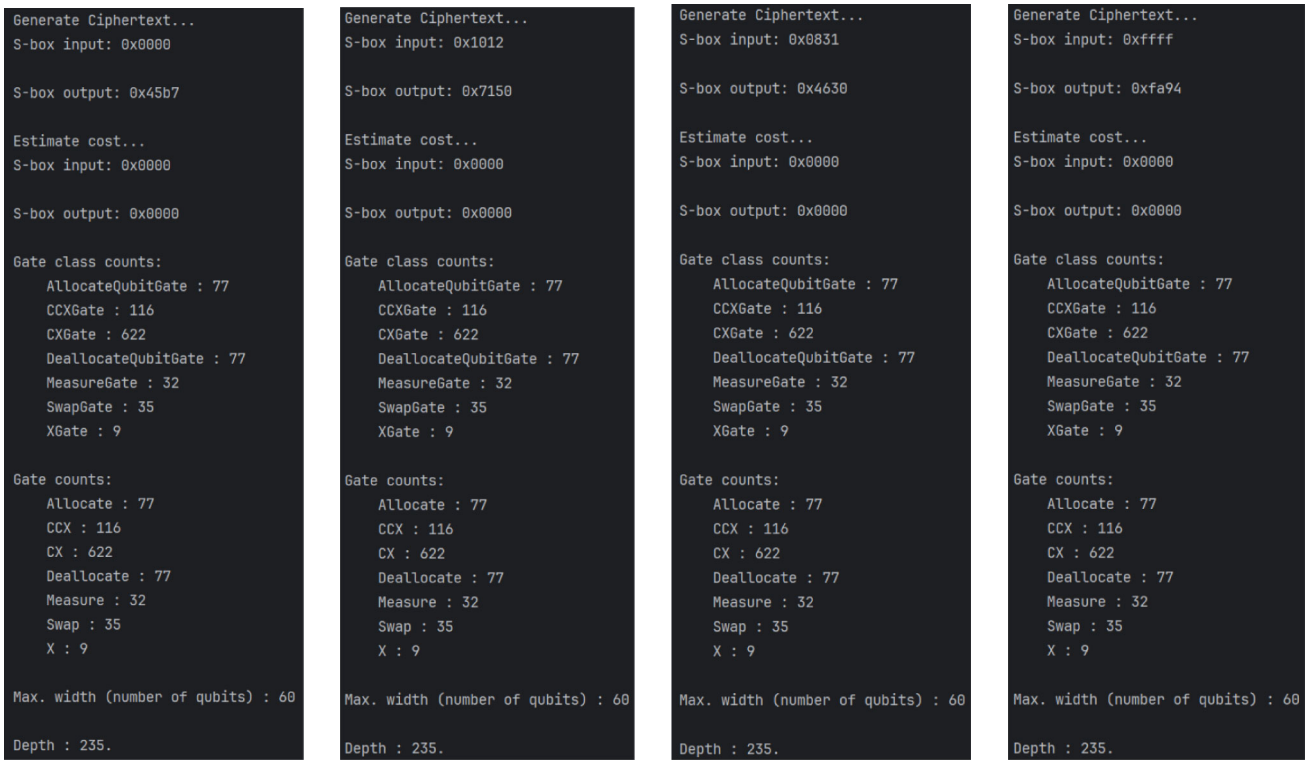
Table 3 Quantum Resource of the S-box and Its Component Operations

Operation	Qubits	Toffoli	CNOT	NOT	Swap	Depth
PN	16	0	73	0	12	45
Sq_F	8	0	10	0	4	6
Mul	28	27	110	0	1	30
Inv	21	35	119	0	4	61
NP_A	16	0	83	0	12	52
S-box	60	116	622	9	35	235

The input state of quantum circuits is modified by applying NOT gates to qubits, causing the ResourceCounter simulator to report varying NOT gate counts for different S-box input values. Specifically, when the S-box input is $0x0000$, the quantum circuit consumes 9 NOT gates. Building on this, for each 1 that appears in the 16-bit input number, the circuit adds one NOT gate. Therefore, when the input is $0x$ffff, the usage of NOT gates reaches the maximum value of 25, while the remaining quantum circuit structure and the number of other quantum gates remain unchanged. Since NOT gates can be implemented in parallel on different qubits, the variation in the number of NOT gates under different input conditions will not significantly increase the critical path depth of the circuit, having limited impact on overall circuit delay. Considering the total number of other types of quantum gates in this circuit (such as CNOT, Toffoli, etc.), the variation in the number of NOT gates (9 to 25) has minimal impact on overall resource consumption, accounting for approximately $0.0115\% \sim 0.0313\%$ of the total gate count, and this difference will not significantly change the overall complexity of the circuit.

5.3 Compatibility experiment

During the implementation process, operations such as scaling involve parameter selection issues. Different parameter values can affect the overall resource consumption of the implementation. This paper conducts compatibility experiments for this purpose. Taking the operation process of



(a) Input: $0x0000$ (b) Input: $0x1012$ (c) Input: $0x0831$ (d) Input: $0x$ffff

Fig. 12 Correctness Verification of MK-3 S-box

square-scale F in the composite field $GF((2^4)^2)$ as given in (10) as an example, the coefficient F becomes a constant value after determining f_1 and f_0. This paper considers all possible values of the coefficients f_1 and f_0. The impact of these coefficients on the resource consumption of the implementation of the square-scale F is shown in Table 4.

The first two columns in the table represent the scaling coefficients F in the composite field $GF((2^4)^2)$, expressed using the scaling coefficient D in the composite field $GF(2^4)$. The middle two columns show the coefficients of the square-scale F calculation results. Due to the symmetry of the normal basis, the calculation results in the table also exhibit a symmetric distribution. The next column indicates the operations required on the subfield as shown by the calculation results, which also represents the number of qubits required, with each operation requiring at least 4 qubits. The last column shows the number of CNOT gates consumed by the implementation. To make the S-box structure more compact, the implementation needs to include fewer operations on the subfield. Therefore, this paper selects the scheme with $f_1 = 0$, $f_0 = D$, which requires the optimal number of operations and CNOT gates among all the constructions.

5.4 Evaluation

This paper presents the first implementation of a 16-bit large S-box on quantum circuits. We compares the implementation of 16×16 matrices in the linear component using Gaussian elimination (Luo et al. 2022), the permutation matrix method (Saravanan and Kalpana 2018), and the dynamic optimal pivot algorithm proposed in this paper, which significantly reduces elimination steps compared to other methods.

Table 5 Number of CNOT Gates Comparison of Quantum Circuit Implementations for Linear Component

	PN	NP_A
Gaussian elimination (Luo et al. 2022)	135	122
Permutation matrix method (Li et al. 2024a)	125	127
This paper	73	83

As shown in Table 5, the optimal dynamic pivot algorithm proposed in this paper reduces the number of CNOT gates by 62 and 39 (approximately 46.7% and 32.8%) for matrices PN and NP_A respectively compared to Gaussian elimination; and reduces CNOT gates by 52 and 44 (approximately 42.4% and 35.4%) compared to the permutation matrix method. The comparison results indicate that: Gaussian elimination uses a fixed and repetitive elimination order, failing to flexibly adjust elimination steps according to actual conditions, which limits computational efficiency. Although the permutation matrix method aims to eliminate the maximum number of "1" elements each time, these positions may reintroduce "1" elements in subsequent elimination

Table 4 Impact of Different Values of Square-Scale Coefficient F

$F = f_1Y^{16} + f_0Y$		$(c_1Y^{16} + c_0Y)^2F = [f_1{c_1}^2 + f_0D({c_1}^2 + {c_0}^2)]Y^{16} + [f_0{c_0}^2 + f_1D({c_1}^2 + {c_0}^2)]Y$		Components	CNOT
D	0	$D{c_1}^2$	$D^2(c_1 + c_0)^2$	$Sq\_D + Sq\_D^2(+)$	3+5
0	D	$D^2(c_1 + c_0)^2$	$D{c_0}^2$	$Sq\_D + Sq\_D^2(+)$	3+5
D	1	$D{c_0}^2$	${c_0}^2 + D^2(c_1 + c_0)^2$	$Sq + Sq\_D + Sq\_D^2(+)$	3+5
1	D	${c_1}^2 + D^2(c_1 + c_0)^2$	$D{c_1}^2$	$Sq + Sq\_D + Sq\_D^2(+)$	3+5
D	D	$D{c_1}^2+$ $D^2(c_1 + c_0)^2$	$D{c_0}^2+$ $D^2(c_1 + c_0)^2$	$2 \times Sq\_D+$ $2 \times Sq\_D^2(+)$	$(3 + 5) \times 2$
D^2	0	$D^2{c_1}^2$	$D^3(c_1 + c_0)^2$	$Sq\_D^2 + Sq\_D^3(+)$	3+6
0	D^2	$D^3(c_1 + c_0)^2$	$D^2{c_0}^2$	$Sq\_D^2 + Sq\_D^3(+)$	3+6
D^2	1	$D^2{c_1}^2 + D(c_1 + c_0)^2$	${c_0}^2 + D^3(c_1 + c_0)^2$	$Sq + Sq\_D^2+$ $Sq\_D(+) + Sq\_D^3(+)$	3+5+6
1	D^2	${c_1}^2 + D^3(c_1 + c_0)^2$	$D^2{c_0}^2 + D(c_1 + c_0)^2$	$Sq + Sq\_D^2+$ $Sq\_D(+) + Sq\_D^3(+)$	3+5+6
D^2	D^2	$D^2{c_1}^2+$ $D(c_1 + c_0)^2$	$D^2{c_0}^2+$ $D(c_1 + c_0)^2$	$2 \times Sq\_D^2$ $2 \times Sq\_D(+)$	$(3 + 5) \times 2$

Table 6 Resource Comparison of Quantum Circuit Implementations for Nonlinear Component

Operation	Implementation	Qubits	Toffoli	CNOT
$GF(2^4)\_Inv$	(Liu et al. 2024)	5	8	7
	(Li et al. 2023)	5	9	5
	(Li et al. 2022)	6	6	6
	This paper	5	8	6
$GF(2^4)\_Mul$	(Liu et al. 2024)	12	9	25
	(Li et al. 2023)	12	9	23
	(Li et al. 2022)	12	9	23
	This paper	12	9	27
$GF(2^8)\_Inv$	(Liu et al. 2024)	20	52	158
	(Li et al. 2023)	20	54	152
	(Li et al. 2022)	22	48	180
	This paper	21	35	119

processes, resulting in redundant elimination steps. The strategy of the dynamic optimal pivot algorithm in this paper is dynamic elimination, where the scope of the pivot row is limited to specific row sets that have consecutive identical sequences with itself. This strategy both ensures that the row transformation elimination process can effectively eliminate consecutive elements and avoids generating too many new columns to be eliminated in the area near the pivot column. Through this systematic column-by-column elimination and dynamic pivot selection method, the algorithm can effectively reduce the number of elimination operations in matrix calculations.

For the nonlinear component, this paper compares key components in the multiplicative inversion in $GF(2^{16})$ with quantum implementations of equivalent functional subcomponents in finite fields $GF(2^4)$ and $GF(2^8)$, with comparison results shown in Table 6.

For multiplicative inversion in the composite field $GF(2^4)$, this paper shows better performance in terms of Toffoli or CNOT gates compared to references (Li et al. 2023) and (Liu et al. 2024), and while it uses 2 more Toffoli gates than reference (Li et al. 2022), it requires 1 fewer qubit. For multiplication in the composite field $GF(2^4)$, this paper uses slightly more CNOT gates than the literature listed below. However, for 8-bit S-boxes in $GF(2^8)$, our multiplicative inversion reduces Toffoli gates by $13 \sim 19(27.1\% \sim 35.2\%)$ and CNOT gates by $33 \sim 61$ $(21.7\% \sim 33.9\%)$ versus prior constructions, with merely 1 qubit difference. Compared to the previously mentioned methods, our proposed approach for the inversion operation in 8-bit S-boxes not only improves computational efficiency but also optimizes the compactness of the quantum circuit. Overall, the construction scheme provided by this paper demonstrates excellent performance and design.

To test the universality and effectiveness of the large S-box under different parameter settings or structural characteristics, this paper selected irreducible polynomials different from the one specified in the MK-3 algorithm's S-box $(P(x) = x^{16} + x^5 + x^3 + x + 1)$ along with normal bases to generate new 16-bit large S-boxes. To control variables, these S-boxes use the same affine transformation (17) as the S-box in the MK-3 algorithm, but demonstrate different mapping relationships in their results.

The S-box construction in this paper targets S-boxes based on finite field construction. The inversion component of this type of S-box does not involve specific numerical operations, but merely performs finite field or composite field inversion operations on given inputs. Therefore, under the design scheme of the S-box construction in this paper, changing S-box parameters to use other different S-boxes will not affect the resource consumption and simulation results of the inversion component. Regarding the verification of the universality of the nonlinear component in this paper's S-box construction, comparisons with modules having the same functionality in existing 8-bit S-boxes are shown in Table 6, which demonstrates the universality of the nonlinear component construction in this paper's S-box.

The differences caused by different parameter settings are in the matrix operation steps of the linear component, specifically the differences in quantum circuit resource consumption for matrix operations in simulation experiments. This paper uses two different sets of parameters, resulting in two different isomorphic matrices PN and isomorphic inverse_affine matrices NP_A. By comparing the CNOT gate consumption differences in the implementation of the

Table 7 CNOT Gate Counts for the Linear Component Under Different Parameters

	MK-3 S-box	16-bit S-box-2	16-bit S-box-3
Gaussian elimination (Luo et al. 2022)	$135 + 122$	$128 + 129$	$120 + 135$
Permutation matrix method (Li et al. 2024a)	$125 + 127$	$117 + 134$	$124 + 123$
This paper	$73 + 88$	$73 + 80$	$67 + 77$

above two matrices using the optimal dynamic pivot algorithm proposed in this paper against the Gaussian elimination and the permutation matrix method, the results are shown in Table 7.

The results show that the optimal dynamic pivot algorithm proposed in this paper still has significant advantages in elimination steps and resource consumption when compared with the Gaussian elimination and the permutation matrix method for 16-bit large S-boxes constructed with different parameters, demonstrating the universality and effectiveness of the method proposed in this paper. The specific normal bases parameters and irreducible polynomials used in the construction of the S-box can be found in Appendix A.

6 Summary

This paper addresses the problem of efficient quantum implementation of 16-bit S-boxes, presenting the first quantum realization study of 16-bit S-boxes. Through a 16-bit S-box construction based on normal bases in the composite field $GF(((2^4)^2)^2)$, we successfully implemented an efficient quantum circuit for the MK-3 algorithm's S-box. This method significantly reduces the quantum resources required to implement a 16-bit large S-box, including the number of qubits and quantum gates, through more compact implementation of operations. In the linear component, the optimal dynamic pivot algorithm proposed in this paper achieves more efficient matrix operations through systematic column-by-column elimination and dynamic pivot selection methods. In the nonlinear component, we decomposed high-dimensional inversion operations into low-dimensional subdomain operations, optimizing the quantum circuit structure of nonlinear components. Experimental results show that the optimal dynamic pivot algorithm proposed in this paper can significantly reduce the use of CNOT gates in the linear component of quantum circuits; and outperforms corresponding components of existing S-boxes in terms of quantum resource consumption in the nonlinear component.

This paper's 16-bit S-box construction based on composite fields provides theoretical support for quantum applications of block ciphers. From an implementation perspective, current mainstream quantum computing technology has achieved substantial breakthroughs, with IBM's Condor processor reaching 1,121 qubits and the University of Science and Technology of China's "Zuchongzhi 3" having 105 readable qubits (Gao et al. 2025), far exceeding the 60-qubit scale required by this paper. Furthermore, the S-box construction in this paper only uses basic quantum operations, avoiding complex quantum superposition and entanglement transformations, therefore demonstrating excellent compatibility with existing quantum hardware and ensuring lower error rates and accumulated errors. These design characteristics demonstrate significant implementation feasibility on current NISQ devices.

In quantum key distribution and quantum secure communication application scenarios, the quantum S-box presented in this paper can serve as a core component of various security protocols, supporting key functions such as key generation, key expansion and processing, and limited encryption processing of sensitive data. The scheme proposed in this paper provides important theoretical and technical support for building efficient quantum block cipher algorithms in the future, demonstrating the forward-looking nature and practicality of this paper in the field of quantum cryptography.

In future research, we will explore other S-box construction methods in depth, including but not limited to S-box constructions based on special integers such as Gaussian integers and Quaternion integers, Eisenstein integers, structure-based S-box constructions, and more. These methods represent cutting-edge development trends in the field of S-box design, with significant research potential and theoretical value. Systematic research on these S-box construction methods can not only significantly enhance the attack resistance and security performance of modern cryptographic systems but also provide a solid theoretical foundation and implementation path for post-quantum cryptography, with profound significance for the long-term development of cryptography.

Appendix A: Parameters

1. Matrix PN and NP_A for the MK-3 Algorithm S-box Under This Paper's Construction

$$PN = \begin{bmatrix}
1&0&1&1&1&1&0&0&0&1&0&1&1&0&1&1\\
1&0&1&1&0&1&0&0&0&0&1&1&1&0&1&0\\
1&1&1&1&0&1&0&1&1&0&0&1&0&0&0&1\\
1&1&1&1&0&1&1&1&0&0&0&1&0&0&1&0\\
1&1&0&1&0&0&0&1&1&0&1&1&1&0&0&0\\
1&0&1&0&0&1&0&1&1&0&0&1&1&1&0&0\\
1&0&1&1&1&0&0&1&1&1&0&1&0&0&1&0\\
1&1&1&0&1&0&1&0&0&1&1&1&1&0&0&1\\
1&0&0&0&1&1&0&0&1&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&1&0&1&1&1&0\\
1&0&0&0&0&0&1&1&0&1&0&0&1&1&0&1\\
1&0&1&1&1&0&1&0&0&0&1&1&1&0&0&0\\
1&0&1&0&0&0&0&1&1&1&1&0&0&1&0&1\\
1&1&0&1&1&0&0&0&0&0&0&1&1&0&0&0\\
1&1&1&0&0&1&1&1&0&1&0&1&1&1&0&0\\
1&1&0&0&0&1&1&0&0&0&0&0&0&0&1&0
\end{bmatrix}$$

$$NP\_A = \begin{bmatrix} 1&0&1&0&1&1&1&0&0&0&1&0&0&1&0&1 \\ 0&1&0&1&1&1&1&1&1&0&1&0&1&0&0&0 \\ 0&1&1&1&1&1&1&1&0&0&1&0&1&1&0&1 \\ 1&0&1&0&1&0&0&0&1&0&1&0&0&1&0&1 \\ 0&1&1&1&0&0&1&0&1&1&0&0&0&1&0&0 \\ 1&1&0&0&0&0&0&0&0&0&1&1&0&1&0&0 \\ 0&1&1&0&0&1&0&0&1&0&0&1&0&0&0&1 \\ 0&1&0&0&0&0&0&1&0&1&1&1&0&1&0&1 \\ 0&0&1&0&0&1&0&1&1&0&1&1&0&0&1&0 \\ 0&0&1&1&1&0&1&1&1&0&0&0&0&1&1&0 \\ 1&1&0&0&0&1&0&1&0&0&0&0&0&1&1&1 \\ 0&1&0&1&0&0&1&0&0&0&0&1&1&1&1&0 \\ 0&0&1&1&0&0&0&1&1&1&1&0&0&0&1&1 \\ 0&1&0&1&0&1&1&1&0&0&1&1&1&0&1&0 \\ 1&0&0&0&0&1&1&0&1&1&0&1&0&0&0&0 \\ 1&0&1&0&0&0&0&0&1&0&1&1&1&0&1&1 \end{bmatrix}$$

2. Composite Field Construction Parameters in this Paper

MK-3 S-box

Irreducible Polynomials: $P(x) = x^{16} + x^5 + x^3 + x + 1$

Normal Bases:

- $W^2 = 0x0733$; $W = 0x0732$
- $Z^4 = 0x8f9e$; $Z = 0x8f9f$
- $Y^{16} = 0xb0df$; $Y = 0xb0de$
- $T^{256} = 0xe5ef$; $T = 0xe5ee$

3. Different 16-bit S-box Construction Parameters

16-bit S-box-2

Irreducible Polynomials: $P(x) = x^{16} + x^{12} + x^3 + x + 1$

Normal Bases:

- $W^2 = 0x015f$; $W = 0x015e$
- $Z^4 = 0x0145$; $Z = 0x0144$
- $Y^{16} = 0x17c5$; $Y = 0x17c4$
- $T^{256} = 0xa057$; $T = 0xa056$

16-bit S-box-3

Irreducible Polynomials: $P(x) = x^{16} + x^5 + x^3 + x^2 + 1$

Normal Bases:

- $W^2 = 0xaccb$; $W = 0xacca$
- $Z^4 = 0x90c5$; $Z = 0x90c4$
- $Y^{16} = 0xc583$; $Y = 0xc582$
- $T^{256} = 0x5097$; $T = 0x5096$

Appendix B: Dynamic optimal pivot algorithm

The specific procedures for implementing the isomorphic matrix PN and the inverse isomorphic matrix NP_A using the dynamic optimal pivot algorithm presented in this paper are shown in Tables 8 and 9.

Table 8 Implementation of PN

CNOT (x[15], x[9])	CNOT (x[15], x[8]))
CNOT (x[15], x[4])	CNOT (x[15], x[3])
CNOT (x[15], x[2])	CNOT (x[14], x[10])
CNOT (x[14], x[4])	CNOT (x[13], x[9])
CNOT (x[13], x[3])	CNOT (x[14], x[13])
CNOT (x[13], x[11])	CNOT (x[13], x[7])
CNOT (x[12], x[13])	CNOT (x[12], x[10])
CNOT (x[12], x[4])	CNOT (x[13], x[6])
Swap (x[12], x[11])	CNOT (x[12], x[13])
CNOT (x[13], x[14])	CNOT (x[12], x[9])
CNOT (x[14], x[5])	CNOT (x[12], x[0])
Swap (x[13], x[10])	CNOT (x[13], x[14])
CNOT (x[14], x[11])	CNOT (x[14], x[8])
CNOT (x[11], x[7])	CNOT (x[13], x[2])
CNOT (x[11], x[1])	Swap (x[14], x[9])
CNOT (x[14], x[11])	CNOT (x[11], x[12])
CNOT (x[14], x[5])	CNOT (x[14], x[3])
CNOT (x[11], x[6])	CNOT (x[11], x[4])
Swap (x[11], x[8])	CNOT (x[11], x[12])
CNOT (x[12], x[13])	CNOT (x[13], x[10])
Swap (x[12], x[7])	CNOT (x[12], x[10])
CNOT (x[10], x[4])	CNOT (x[10], x[3])
Swap (x[15], x[6])	CNOT (x[15], x[13])
CNOT (x[15], x[12])	CNOT (x[13], x[8])
CNOT (x[12], x[11])	CNOT (x[13], x[2])
CNOT (x[12], x[1])	CNOT (x[12], x[0])
Swap (x[9], x[5])	CNOT (x[9], x[7])
CNOT (x[9], x[10])	CNOT (x[9], x[15])
CNOT (x[9], x[5])	CNOT (x[10], x[14])
Swap (x[13], x[4])	CNOT (x[13], x[7])
CNOT (x[13], x[6])	CNOT (x[7], x[8])
Swap (x[4], x[3])	CNOT (x[4], x[12])
CNOT (x[4], x[9])	CNOT (x[4], x[6])
Swap (x[12], x[2])	CNOT (x[12], x[14])
Swap (x[9], x[1])	Swap (x[10], x[0])
CNOT (x[10], x[8])	CNOT (x[10], x[12])
CNOT (x[10], x[9])	CNOT (x[12], x[5])
CNOT (x[12], x[1])	CNOT (x[12], x[6])
CNOT (x[6], x[11])	CNOT (x[6], x[0])
CNOT (x[9], x[15])	CNOT (x[9], x[4])
CNOT (x[9], x[14])	CNOT (x[4], x[13])
CNOT (x[14], x[7])	CNOT (x[14], x[2])
CNOT (x[2], x[3])	

Table 9 Implementation of NP_A

CNOT (x[15], x[14])	CNOT (x[15], x[13])
CNOT (x[15], x[12])	CNOT (x[15], x[6])
CNOT (x[15], x[5])	CNOT (x[15], x[5])
CNOT (x[15], x[4])	CNOT (x[14], x[15])
CNOT (x[14], x[9])	CNOT (x[14], x[2])
CNOT (x[14], x[1])	CNOT (x[14], x[0])
Swap (x[14], x[13])	CNOT (x[14], x[15])
CNOT (x[14], x[11])	CNOT (x[14], x[10])
CNOT (x[14], x[8])	CNOT (x[14], x[7])
CNOT (x[14], x[5])	CNOT (x[14], x[2])
CNOT (x[14], x[0])	CNOT (x[15], x[3])
Swap (x[13], x[12])	CNOT (x[13], x[11])
CNOT (x[13], x[10])	CNOT (x[13], x[9])
CNOT (x[13], x[8])	CNOT (x[13], x[5])
CNOT (x[13], x[3])	CNOT (x[13], x[1])
Swap (x[14], x[11])	CNOT (x[14], x[12])
CNOT (x[14], x[10])	CNOT (x[12], x[4])
CNOT (x[14], x[2])	CNOT (x[14], x[1])
CNOT (x[10], x[1])	Swap (x[10], x[9])
CNOT (x[10], x[14])	CNOT (x[10], x[9])
CNOT (x[9], x[15])	CNOT (x[9], x[8])
CNOT (x[14], x[7])	CNOT (x[9], x[5])
CNOT (x[10], x[3])	CNOT (x[9], x[2])
CNOT (x[14], x[0])	CNOT (x[8], x[7])
CNOT (x[8], x[5])	CNOT (x[8], x[3])
CNOT (x[8], x[2])	CNOT (x[8], x[1])
Swap (x[13], x[7])	CNOT (x[13], x[15])
CNOT (x[13], x[12])	CNOT (x[15], x[7])
CNOT (x[13], x[1])	Swap (x[14], x[6])
CNOT (x[14], x[15])	CNOT (x[14], x[11])
CNOT (x[14], x[9])	CNOT (x[15], x[3])
Swap (x[8], x[5])	CNOT (x[8], x[14])
CNOT (x[8], x[13])	CNOT (x[13], x[10])
CNOT (x[14], x[12])	CNOT (x[12], x[5])
CNOT (x[13], x[2])	CNOT (x[14], x[0])
Swap (x[11], x[4])	CNOT (x[11], x[9])
CNOT (x[11], x[8])	CNOT (x[8], x[7])
Swap (x[11], x[3])	CNOT (x[11], x[12])
CNOT (x[12], x[4])	Swap (x[8], x[2])
CNOT (x[8], x[12])	CNOT (x[8], x[15])
CNOT (x[8], x[6])	CNOT (x[12], x[9])
Swap (x[7], x[1])	CNOT (x[7], x[5])
CNOT (x[7], x[11])	CNOT (x[7], x[6])
CNOT (x[11], x[13])	CNOT (x[11], x[1])
CNOT (x[6], x[4])	CNOT (x[4], x[2])
Swap (x[14], x[0])	CNOT (x[14], x[15])
CNOT (x[14], x[5])	CNOT (x[5], x[10])
CNOT (x[15], x[3])	CNOT (x[3], x[0])

Author Contributions Lei Zhang: Supervision, Resources.
Guoyuan Li: Writing - Original Draft, Conceptualization, Methodology, Software, Investigation, Formal Analysis.
Jingchen Dai: Data Curation, Investigation.
Ruipeng Hong: Software, Formal Analysis.
Chaoen Xiao: Supervision, Writing - Review & Editing.
Jianxin Wang: Resources, Writing - Review & Editing.
Yifan Zhang: Conceptualization, Visualization.

Funding This work is supported by The National Natural Science Foundation of China (No: 62072014) and The Fundamental Research Funds for the Central Universities (No: 3282024051, 3282024009).

Availability of Data and Material All data generated or analyzed during this study are included in this published article.

Declarations

Competing Interests The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

References

Abd-El-Atty B (2023) Efficient s-box construction based on quantum-inspired quantum walks with pso algorithm and its application to image cryptosystem. Complex Intell Syst 9:4817–4835

Abed S, Jaffal R, Mohd BJ et al (2021) Performance evaluation of the SM4 cipher based on field-programmable gate array implementation. IET Circ Devices Syst 15(2):121–135

Boyar J, Matthews P, Peralta R (2008) On the shortest linear straight-line program for computing linear forms. In: Mathematical foundations of computer science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings, Lecture Notes in Computer Science, vol 5162. Springer, pp 168–179

Boyar J, Peralta R (2010) A new combinational logic minimization technique with applications to cryptology. In: Experimental algorithms, 9th international symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings, Lecture notes in computer science, vol 6049. Springer, pp 178–189

Canright D (2005) A very compact s-box for AES. In: Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings, Lecture Notes in Computer Science, vol 3659. Springer, pp 441–455

Çavuşoğlu Ü, Kaçar S, Pehlivan I et al (2017) Secure image encryption algorithm design using a novel chaos based s-box. Chaos, Solitons Fractals 95:92–101

Chun M, Baksi A, Chattopadhyay A (2023) DORCIS: depth optimized quantum implementation of substitution boxes. IACR Cryptol ePrint Arch p 286

Chung D, Lee S, Choi D et al (2022) Alternative tower field construction for quantum implementation of the AES s-box. IEEE Trans Comput 71(10):2553–2564

Curlin PS, Heiges J, Chan C et al (2025) A survey of hardware-based aes sboxes: area, performance, and security. ACM Comput Surv 57(9)

Dasu VA, Baksi A, Sarkar S et al (2019) LIGHTER-R: optimized reversible circuit implementation for sboxes. In: 32nd IEEE International System-on-Chip Conference, SOCC 2019, Singapore, September 3-6, 2019. IEEE, pp 260–265

Gao D, Fan D, Zha C et al (2025) Establishing a new benchmark in quantum computational advantage with 105-qubit zuchongzhi 3.0 processor. Phys Rev Lett 134:090601

Grassl M, Langenberg B, Roetteler M et al (2016) Applying grover's algorithm to aes: quantum resource estimates. In: Post-quantum cryptography. Springer International Publishing, pp 29–43

Haferkamp J, Faist P, Kothakonda NBT et al (2022) Linear growth of quantum circuit complexity. Nat Phys 18:528–532

Hazzazi M, Sajjad M, Bassfar Z et al (2023) Nonlinear components of a block cipher over Eisenstein integers. Comput Mater Cont 77:3659

Jing X, Li Y, Zhao G et al (2023) Quantum circuit implementation and resource analysis of Lblock and Lici. Quantum Inf Process 22(9):347

Jones T, Brown A, Bush I et al (2019) QuEST and high performance simulation of quantum computers. Sci Rep 9:10736

Kelly M, Kaminsky A, Kurdziel MT et al (2015) Customizable sponge-based authenticated encryption using 16-bit s-boxes. In: 34th IEEE Military Communications Conference, MILCOM 2015, Tampa, FL, USA, October 26-28, 2015. IEEE, pp 43–48

Langenberg B, Pham H, Steinwandt R (2019) Reducing the cost of implementing AES as a quantum circuit. IACR Cryptol ePrint Arch p 854

Lee Y, Lee J, Chung B et al (2024) Physical qubits estimation for quantum S-box. In: 15th International conference on Information and Communication Technology Convergence, ICTC 2024, Jeju Island, Republic of Korea, October 16-18, 2024. IEEE, pp 1369–1370

Li Z, Cai B, Sun H et al (2022) Novel quantum circuit implementation of Advanced Encryption Standard with low costs. Sci China Phys Mech Astron 65:1–17

Li Z, Gao F, Qin S et al (2023) Quantum circuit for implementing Camellia S-box with low costs. Sci Sin Phys Mech Astron 53(4):1–9

Li Q, Luo Q, Lyu Y et al (2024) Quantum circuit optimization for SM4 cryptographic algorithm S-box. J Cryptologic Res 11:1–11

Li Y, Zhang W, Lin Y et al (2024) A circuit area optimization of MK-3 S-box. Cybersecur 7(1):17

Lidl R, Niederreiter H (1997) Finite fields. Cambridge University Press

Liu J, Tan X, Li M et al (2024) Efficient quantum circuit implementation of the SM4 S-box. Sci Sin Phys Mech Astron 54(4):1–9

Liu Q, Preneel B, Zhao Z et al (2023) Improved quantum circuits for AES: reducing the depth and the number of qubits. In: Guo J, Steinfeld R (eds) Advances in cryptology - ASIACRYPT 2023 - 29th international conference on the theory and application of cryptology and information security, Guangzhou, China, December 4-8, 2023, Proceedings, Part III, Lecture Notes in Computer Science, vol 14440. Springer, pp 67–98

Li Y, Wang M (2014) Constructing s-boxes for lightweight cryptography with feistel structure. In: Batina L, Robshaw M (eds) Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings, Lecture Notes in Computer Science, vol 8731. Springer, pp 127–146

Luo Q, Li X, Yang G (2021) Quantum circuit implementation of S-box for SM4 cryptographic algorithm. J Univ Electron Sci Technol China 50(6):820–826

Luo Q, Li X, Yand G et al (2022) Quantum circuit implementation of s-box for sm4 cryptographic algorithm based on composite field arithmetic. J Univ Electron Sci Technol China 51(6):812–818

Nandan V, Rao RGS (2023) Low-power aes s-box design using dual-basis tower field extension method for cyber security applications. Complex Intell Syst 9:2959–2967

Rahman Z, Yi X, Billah M et al (2022) Enhancing AES using chaos and logistic map-based key generation technique for securing iot-based smart home. IACR Cryptol ePrint Arch p 410

Sajjad M, Shah T, ul Haq T et al (2024) SPN based RGB image encryption over Gaussian integers. Heliyon 10:1–17

Sajjad M, Shah T, Hamza R et al (2025) Multiple color images security by SPN over the residue classes of Gaussian integer $Z[i]_h$. Sci Rep 15:6425

Saravanan P, Kalpana P (2018) Novel reversible design of Advanced Encryption Standard cryptographic algorithm for wireless sensor networks. Wirel Pers Commun 100(4):1427–1458

Shah T, Khan DA, Ali A (2024) Design of nonlinear component of block cipher using quaternion integers. Multim Tools Appl 83(9):25657–25674

Stafford DF (2021) Evaluating performance and efficiency of a 16-bit substitution box on an FPGA. Master's thesis, Rochester Institute of Technology

Steiger DS, Häner T, Troyer M (2018) ProjectQ: an open source software framework for quantum computing. Quantum 2:49

Wang Z, Wei S, Long G (2022) A quantum circuit design of AES requiring fewer quantum qubits and gate operations. Front Phys 17

Wood CA (2013) Large substitution boxes with efficient combinational implementations. Master's thesis, Rochester Institute of Technology

Wu X, Dou D, Wei Y et al (2023) A 16-bit S-box design method based on Feistel-NFSR structure. J Cryptologic Res 10(01):146–154

Wu X, Wu T, Huang Z et al (2024) Construction of 16-bit dynamic S-box based on chaotic map and NFSR. Inf Sec Commun Privacy 11:10–19

Xu H, Duan M, Tan L et al (2019) On the NBC algorithm. J Cryptologic Res 6(6):760–767

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Lei Zhang[1] · Guoyuan Li[1] · Jingchen Dai[1] · Ruipeng Hong[1] · Chaoen Xiao[1] · Jianxin Wang[1] · Yifan Zhang[1]

✉ Guoyuan Li
20232922@mail.besti.edu.cn; lgy957266799@163.com

Lei Zhang
zhanglei@besti.edu.cn

Jingchen Dai
15139027395@163.com

Ruipeng Hong
ChinaDvBishop@outlook.com

Chaoen Xiao
xce@besti.edu.cn

Jianxin Wang
wangjianxin@besti.edu.cn

Yifan Zhang
zhang1fan2000@163.com

[1] Beijing Electronic Science and Technology Institute, 100070 Beijing, China