Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2024.052374

ARTICLE

A Novel Framework to Construct S-Box Quantum Circuits Using System Modeling: Application to 4-Bit S-Boxes

Yongjin Jeon, Seungjun Baek[#] and Jongsung Kim[*]

Department of Financial Information Security, Kookmin University, Seoul, 02707, Republic of Korea

*Corresponding Author: Jongsung Kim. Email: jskim@kookmin.ac.kr

[#]Co-first author: Seungjun Baek

Received: 31 March 2024 Accepted: 06 June 2024 Published: 20 August 2024

ABSTRACT

Quantum computers accelerate many algorithms based on the superposition principle of quantum mechanics. The Grover algorithm provides significant performance to malicious users attacking symmetric key systems. Since the performance of attacks using quantum computers depends on the efficiency of the quantum circuit of the encryption algorithms, research research on the implementation of quantum circuits is essential. This paper presents a new framework to construct quantum circuits of substitution boxes (S-boxes) using system modeling. We model the quantum circuits of S-boxes using two layers: Toffoli and linear layers. We generate vector spaces based on the values of qubits used in the linear layers and apply them to find quantum circuits. The framework finds the circuit by matching elements of vector spaces generated from the input and output of a given S-box, using the forward search or the meet-in-the-middle strategy. We developed a tool to apply this framework to 4-bit S-boxes. While the 4-bit S-box quantum circuit construction tool LIGHTER-R only finds circuits that can be implemented with four qubits, the proposed tool achieves the circuits with five qubits. The proposed tool can find quantum circuits of 4-bit odd permutations based on the controlled NOT, NOT, and Toffoli gates, whereas LIGHTER-R is unable to perform this task in the same environment. We expect this technique to become a critical step toward optimizing S-box quantum circuits.

KEYWORDS

System modeling; quantum circuit; S-box circuit; quantum computer

1 Introduction

Quantum computers accelerate many algorithms based on the superposition principle of quantum mechanics. Shor's algorithm [1] exponentially reduces the complexity of attacking public-key schemes on quantum computers. Since 2016, the National Institute of Standards and Technology (NIST) has been conducting the post-quantum cryptography standardization process [2]. For symmetric-key schemes, Grover's and Simon's algorithms [3,4] offer attackers significant performance to attack the schemes, but these algorithms do not entirely compromise the security such systems'. However, in a quantum computing environment, symmetric-key cryptography may have weak properties not yet studied for each algorithm. Most cryptanalysis, including attacks targeting these vulnerabilities and

Copyright © 2024 The Authors. Published by Tech Science Press.
This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

generic attacks, requires the implementation of cipher's quantum circuits, and its performance depends on the efficiency of the circuit. For future security, research on the implementation of the quantum circuits must be conducted, and it is necessary to know what performance bounds there are.

The substitution box (S-box) is a crucial component that provides confusion in symmetric-key schemes. When implementing a cipher as a quantum circuit, the linear layer can be implemented with only NOT and controlled-NOT (CNOT) gates. However, highly structured nonlinear layers, such as the S-box, must employ relatively expensive Toffoli gates and numerous qubits. In quantum circuits for symmetric-key schemes, the S-box incurs the highest cost.

The complexity of a quantum circuit is evaluated by the number of qubits and the Toffoli-depth defined by the number of non-parallelizable Toffoli gates. Optimizing these two parameters increases the implementation efficiency of quantum computers. This approach improves the attackers' ability to perform exhaustive search and dedicated attacks using Grover's algorithm in quantum computer. Hence, optimizing the quantum circuits of S-boxes is critical to assess the security of symmetric-key schemes against quantum computer-based attacks.

Extensive recent research has been conducted on finding efficient quantum circuits for the Advanced Encryption Standard (AES). Grassl et al. [5] initially proposed a quantum circuit for the AES and introduced a zig-zag structure to reduce the number of qubits required for its implementation. Subsequently, several studies have been conducted to reduce the number of qubits to implement the AES [6–9]. However, in NIST's post-quantum cryptography standardization process, the Toffoli-depth represents a critical parameter. In response, Jaques et al. [8] attempted to construct an AES quantum circuit with a shallow Toffoli-depth. Recently, Huang et al. [10] proposed an AES quantum circuit with the shallowest depth.

The terms and notations used in this paper are explained as follows:

- CNOT, NOT, Toffoli gates: Gates used in quantum circuit
- CNT-circuit: Quantum circuit using only CNOT, NOT, and Toffoli gates
- CT-circuit: Quantum circuit using only CNOT and Toffoli gates
- n: Size of S-box
- q: Number of qubits used in quantum circuit
- $\mathscr{C}$: quantum circuit $|\alpha\rangle|0\rangle \rightarrow |S(\alpha)\rangle|0\rangle$
- L_i and T_i: i-th linear layer and Toffoli layer used in the circuit, respectively
- x_i and y_i: The i-th qubits of the input and output of the quantum circuit
- $\mathscr{X}_i$: Vector space corresponding to the i-th linear layer of the quantum circuit

Contributions. This paper provides a new framework to construct quantum circuits $\mathscr{C} : |\alpha\rangle|0\rangle \rightarrow |S(\alpha)\rangle|0\rangle$ of S-boxes. We treat quantum circuits of S-boxes using only CNOT, NOT, and Toffoli gates, called CNT-circuits, using system modeling. Using the CNT-circuits, we provide a framework for finding quantum circuits of the S-boxes with low Toffoli-depths according to a limited number of qubits by matching elements of vector spaces generated from the inputs and outputs of the S-boxes. The framework employs a meet-in-the-middle strategy, in which the key is to analyze the vector spaces spanned by the values before and after the Toffoli layers. The framework provides a specialized search for the Toffoli-depth by ignoring the detailed implementations of the linear layers of S-boxes. To the best of our knowledge, no study has analyzed the Toffoli-depth and number of qubits using a vector space and basis analysis between Toffoli layers in the quantum circuits of S-boxes.

To verify the effectiveness of the framework, we propose a technique and tool for applying the framework to a 4-bit S-box. These components are currently used as essential elements in many Authenticated Encryption with Associated Data (AEAD) schemes and block ciphers [11–15]. In addition, `LIGHTER-R` [16] provides Toffoli-depth optimized quantum circuits of 4-bit S-boxes with a 4-qubit restriction. However, this approach fails if the target 4-bit S-boxes are odd permutations. This result occurs due to the theorem that odd permutations cannot be implemented with 4 qubits in CNT-circuits and requiress at least 5 qubits [17]. The algorithms offer a wider range of quantum circuits compared to LIGHTER-R in terms of the Toffoli-depth and number of qubits (up to 5). This improvement allows the algorithms to produce the quantum circuits of the 4-bit S-boxes of odd permutations. Given that half of 4-bit S-boxes are odd permutations, this result enables researchers to implement quantum circuits for all 4-bit S-boxes.

Paper Organization. Section 2 describes quantum computation and quantum circuits. Section 3 describes the properties of the CNT-circuit and the model in the quantum circuit. Section 4 describes the framework for finding quantum circuits of 4-bit S-boxes according to a limited number of qubits. Section 5 presents the results of the proposed algorithm in Section 4. Section 6 presents the conclusions and provides a discussion.

2 Quantum Computation and Quantum Circuits

A fundamental concept in classical computing involves a bit, characterized as either 0 or 1. Conversely, the qubit plays a role as a bit in quantum computing, holding 0 and 1 at the same time according to the superposition principle of quantum mechanics. The values $|0\rangle$ and $|1\rangle$ are orthonormal bases of the two-dimensional Hilbert space, also called the computational basis. The superposition state of a qubit can be represented as $\alpha|0\rangle + \beta|1\rangle$ ($\alpha, \beta \in \mathbb{C}$), and α and β are called the complex probability amplitude. The state of the qubit is destroyed by measurement, after which one can observe $|0\rangle$ or $|1\rangle$, with the respective probabilities of $|\alpha|^2$ and $|\beta|^2$ (thus, $|\alpha|^2 + |\beta|^2 = 1$ holds). To describe n qubits, we need a 2^n dimensional Hilbert space for which the orthonormal bases are $|00\cdots0\rangle$, $|00\cdots1\rangle$, $\ldots$, $|11\cdots1\rangle$, and total 2^n.

This work primarily concerns with quantum circuits consisting of CNOT, NOT, and Toffoli gates. A *CNOT* gate is the two-qubit gate defined by $CNOT : |a\rangle|b\rangle \mapsto |a\rangle|b \oplus a\rangle$, and a *NOT* gate is the single-qubit gate defined by $NOT : |a\rangle \mapsto |a \oplus 1\rangle$. A *Toffoli* gate is the three-qubit gate defined by $Toffoli : |a\rangle|b\rangle|c\rangle \mapsto |a\rangle|b\rangle|c \oplus ab\rangle$. A *Toffoli* gate can handle the XOR and AND of classical gates at once. Fig. 1 presents these quantum gates.

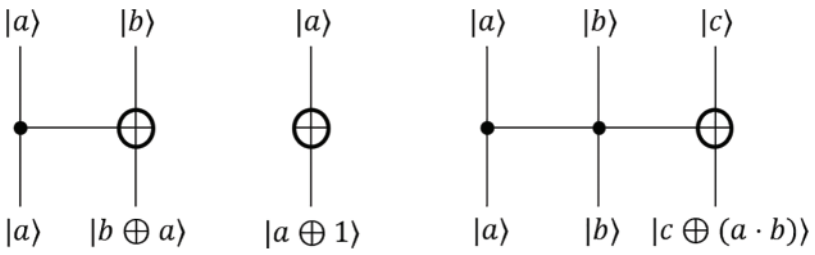

Figure 1: CNOT (left), NOT (middle), and Toffoli (right) gates

A quantum circuit using only CNOT, NOT, and Toffoli gates is defined as a CNT-circuit. In the CNT-circuit, the NOT gates can be moved to the circuit's last operation without changing the Toffoli-depth, using the properties in Fig. 2. The NOT gates gathered in the last operation are equivalent to

using an XORing on a constant value in the S-box. All S-boxes can be implemented with CNT-circuits; thus CT-circuits (without NOT gates) can implement all S-boxes satisfying $0 \mapsto 0$ [17]. Therefore, only CT-circuits are considered in this paper.

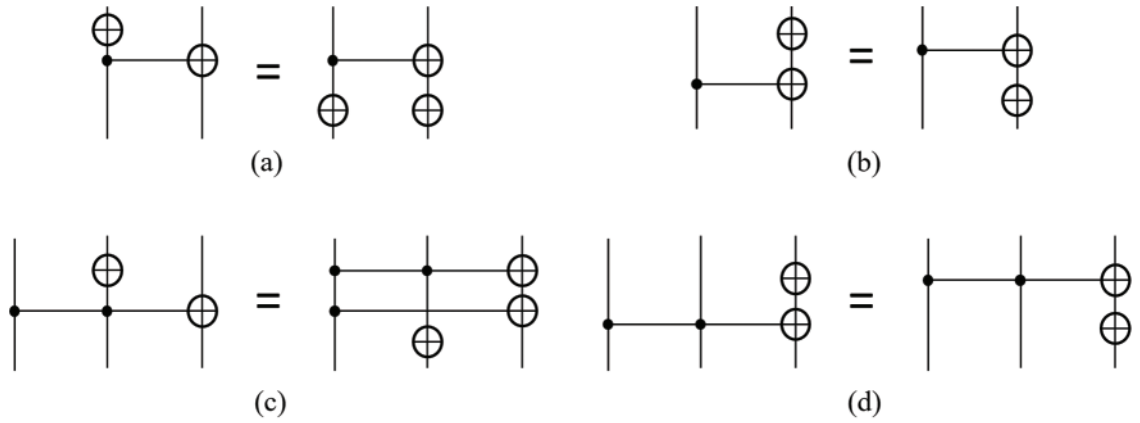

Figure 2: Properties of NOT gates

3 Modeling the Quantum Circuits of S-Boxes

We considered the n-bit S-box defined by the vectorial Boolean function $\mathbb{F}^n \to \mathbb{F}^n$. In the circuit of the S-box, n Boolean coordinate functions are represented by wires. Each wire connects to n input bits, and additional wires may be required depending on the circuit. All these wires become qubits in a quantum circuit.

We modeled CT-circuits for $\mathscr{C} : |\alpha\rangle|0\rangle \to |S(\alpha)\rangle|0\rangle$ of S-boxes satisfying $0 \mapsto 0$. Let $\mathscr{C}$ use q qubits with a Toffoli-depth of t. We defined the layers with only Toffoli gates as Toffoli layers and treated the layers between them as linear layers (including empty layers). In addition, $\mathscr{C}$ has $t+1$ linear layers, including the outermost two linear layers. We denote the i-th Toffoli layer as T_i and the i-th linear layer as L_i. We established the indices of the layers as represented in Eq. (1). The CNOT gates can be implemented without additional qubits [18], and their cost is exempted from the analysis model. Therefore, we omitted the detailed implementation of CNOT gates in the linear layer.

$$\mathscr{C} : L_t \circ T_t \circ L_{t-1} \circ T_{t-1} \circ \cdots \circ L_1 \circ T_1 \circ L_0. \tag{1}$$

To facilitate finding the circuit, we arranged the Toffoli gates in order within the Toffoli layers. We assumed that the control and target qubit positions of the Toffoli gates are fixed, and the exchange of wires that occurs while fixing them is absorbed by the linear layers. In detail, the control qubits of the i-th Toffoli gate use the $(3i-2)$-th and $(3i-1)$-th qubits, and the $3i$-th qubit serves as the target qubit. Afterward, Toffoli gates are arranged consecutively in the Toffoli layers. Fig. 3 depicts a Toffoli layer using k Toffoli gates.

Implementing linear layers is equivalent to knowing their input and output values. When Toffoli layers are implemented, the input value of the following linear layer can be determined through the output values of the previous linear layer. If the output values of t linear layers are determined, the entire circuit can be implemented.

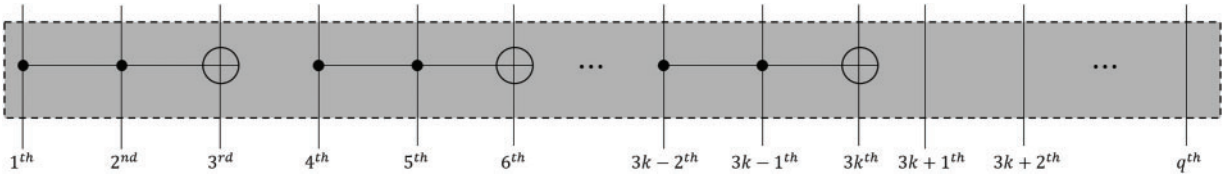

Figure 3: Toffoli layer in the proposed model

The qubit values at each point represent the input and output of the linear layer. We treated the qubit values as Boolean functions, and considered the vector space spanned by them. The vector spaces spanned at the input and output points of the linear layer are identical. We defined the vector space generated by L_i as $\mathscr{X}_i$, and each $\mathscr{X}_i$ is transformed into $\mathscr{X}_{i+1}$ by T_{i+1} corresponding to the $(i+1)$-th Toffoli layer. In addition, we defined the input and output of the S-box as $(x_0, \cdots, x_{n-1})$ and $(y_0, \cdots, y_{n-1})$, respectively, as follows:

$$span(x_0, \cdots, x_{n-1}) = \mathscr{X}_0 \xrightarrow{T_1} \mathscr{X}_1 \xrightarrow{T_2} \cdots \xrightarrow{T_t} \mathscr{X}_t = span(y_0, \cdots, y_{n-1}).$$

4 Exploring 4-Bit S-Box Quantum Circuits with the Meet-in-the-Middle Strategy: Up to 5 Qubits

4.1 Properties between Neighboring Vector Spaces

Let $\mathscr{X}_i$ be a vector space spanning L_i. Quantum circuits comprise reversible gates; thus, these circuits can be implemented in forward and backward directions.

forward: $\mathscr{X}_0 \to \mathscr{X}_1 \to \cdots \to \mathscr{X}_t$,

backward: $\mathscr{X}_t \to \mathscr{X}_{t-1} \to \cdots \to \mathscr{X}_0$.

Of the three qubits included in one Toffoli gate, only the target qubit changes in value. There are invariant qubit values; hence the intersection of two consecutive spaces, $\mathscr{X}_i$ and $\mathscr{X}_{i+1}$, is non-empty. This feature is also reflected in the intersection of $\mathscr{X}_i$ and $\mathscr{X}_{i+r}$ for a sufficiently small r. This logic is generalized in Theorem 1 and depicted in Fig. 4.

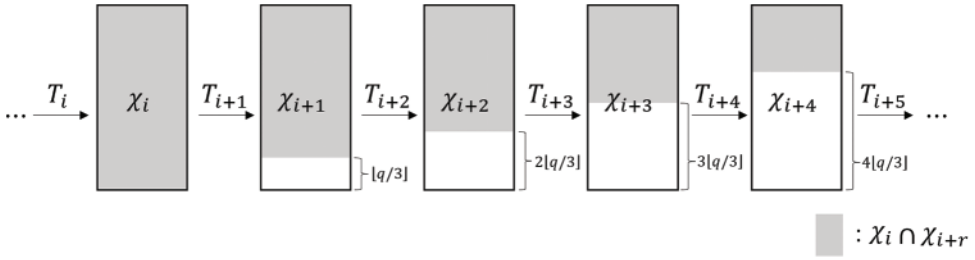

Figure 4: Depiction of Theorem 1

Theorem 1. Define the CT-circuit of S-box S as $\mathscr{C} : |\alpha\rangle|0\rangle \to |S(\alpha)\rangle|0\rangle$. If the circuit uses q qubits and has Toffoli-depth t, for any $i \le t - r$,

$$\dim(\mathscr{X}_i) - \dim(\mathscr{X}_i \cap \mathscr{X}_{i+r}) \le r\lfloor q/3 \rfloor,$$

$$\dim(\mathscr{X}_{i+r}) - \dim(\mathscr{X}_i \cap \mathscr{X}_{i+r}) \le r\lfloor q/3 \rfloor.$$

Proof. There are r Toffoli layers between the points of $\mathscr{X}_i$ and $\mathscr{X}_{i+r}$. A maximum of $\lfloor q/3 \rfloor$ Toffoli gates can be used in one Toffoli layer; hence, at most, $r\lfloor q/3 \rfloor$ values are not in $\mathscr{X}_i$. Therefore, we obtain $\dim(\mathscr{X}_i) - \dim(\mathscr{X}_i \cap \mathscr{X}_{i+r}) \leq r\lfloor q/3 \rfloor$. The lower equation is found using a similar process. ■

According to the above theorem, the lower bound of the Toffoli-depth is found for the quantum circuit $|\alpha\rangle|0\rangle \rightarrow |S(\alpha)\rangle|0\rangle$. The output of the S-box can be expressed by concatenating the outputs of n Boolean functions. The linear combination of these Boolean functions is called a *component function*. The *zero function*, a constant function that outputs only 0, is excluded from the definition of the component function. A Boolean function f that satisfies $f(x+y) = f(x) + f(y)$ for all values of $x, y \in \mathscr{F}_2^n$ is called a Boolean *linear function*.

Theorem 2. For $n \geq 3$, let $\mathscr{X}_0$ be the set of all n-variable Boolean linear functions, and let $\mathscr{X}_t$ be the set of all component functions of the S-box S (including the zero function). Then, the Toffoli-depth of the quantum circuit $\mathscr{C} : |\alpha\rangle|0\rangle \rightarrow |S(\alpha)\rangle|0\rangle$ of n-bit S using q-qubit is $(n - \dim(\mathscr{X}_0 \cap \mathscr{X}_t))/\lfloor q/3 \rfloor$ or greater.

Proof. This situation is a special case where $i = 0$ and $r = t$ in Theorem 1. The proof is as follows:

$$\dim(\mathscr{X}_0) - \dim(\mathscr{X}_0 \cap \mathscr{X}_t) \leq t\lfloor q/3 \rfloor,$$

$$n - \dim(\mathscr{X}_0 \cap \mathscr{X}_t) \leq t\lfloor q/3 \rfloor,$$

$$(n - \dim(\mathscr{X}_0 \cap \mathscr{X}_t))/\lfloor q/3 \rfloor \leq t.$$

■

We consider both forward and backward directions. The proposed algorithms confirm how many values of the newly constructed vector space belong to the opposite vector space. For example, we consider that $\mathscr{X}_i$ and $\mathscr{X}_{t-j}$ are obtained by implementing up to the i-th Toffoli layer in the forward direction and the j-th Toffoli layer in the backward direction. The algorithms select $\mathscr{X}_{i+1}$ which yields the greatest intersection with $\mathscr{X}_{t-j}$, to implement the $(i+1)$-th Toffoli layer in the forward direction. In this case, $\dim\left(\mathscr{X}_i \cap \mathscr{X}_{t-j}\right) < \dim\left(\mathscr{X}_{i+1} \cap \mathscr{X}_{t-j}\right)$ holds, and if $\mathscr{X}_{i+1} = \mathscr{X}_{t-j}$, the circuit is completely implemented.

4.2 Exploring 4-Bit S-Box Quantum Circuits in the Forward Direction

We describe the process of implementing the 4-bit S-box $S(x_0, x_1, x_2, x_3) = (y_0, y_1, y_2, y_3)$ using 4 and 5 qubits. Due to the limited number of qubits, only one Toffoli gate is in each Toffoli layer. Let $\mathfrak{P}_0 = span(x_0, x_1, x_2, x_3)$ and $\mathfrak{P}_t = span(y_0, y_1, y_2, y_3)$. If $\mathfrak{P}_0 = \mathfrak{P}_t$ holds, S is a linear function; thus, the Toffoli-depth is zero. The linear function can be implemented without ancilla qubits (i.e., with 4 qubits). The proposed algorithm takes vector space pairs using the definition below.

Definition 1. The pair of vector spaces in the forward and backward directions are denoted as $(\mathscr{X}, \mathscr{X}')$. $\mathscr{Y}$ represents the vector space that lies ahead of or is equal to $\mathscr{X}$ in the forward direction, whereas $\mathscr{Y}'$ represents the vector space that lies ahead or is equal to $\mathscr{X}'$ in the backward direction. $(\mathscr{Y}, \mathscr{Y}')$ is *closer* than $(\mathscr{X}, \mathscr{X}')$ if one of the following conditions is satisfied:

- $\mathscr{Y} = \mathscr{Y}'$ holds, or
- $\dim(\mathscr{Y} \cap \mathscr{Y}') > \dim(\mathscr{X} \cap \mathscr{X}')$.

We defined the vector space corresponding to the i-th linear layer in the forward direction as $\mathscr{X}_i$ and the vector space corresponding to the j-th linear layer in the backward direction as $\mathscr{X}_{t-j}$. The algorithms take $(\mathscr{X}_i, \mathscr{X}_{t-j})$ as input, and $(\mathscr{X}_{i+1}, \mathscr{Y})$ as output, which is a closer pair than $(\mathscr{X}_i, \mathscr{X}_{t-j})$. In addition, $\mathscr{Y}$ can be either $\mathscr{X}_{t-j}$ or $\mathscr{X}_{t-j-1}$.

Algorithm 1: Forward finding for quantum circuits

input: $\mathscr{X}_i$, $\mathscr{X}_{t-j}$, n, q where $n \leq q \leq 5$
output: a set $\mathscr{D}$ of pairs $(B_{i+1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j})$ where B_{i+1} is the basis of $\mathscr{X}_{i+1}$.
$B_i \leftarrow$ a basis of $\mathscr{X}_i$
$\mathscr{D} \leftarrow \{\}$
$F^i_{\rightarrow} \leftarrow \{ab \oplus c | a, b(\neq a), c \in \mathscr{X}_i$, if $\dim(\mathscr{X}_i) = q$, then $a, b,$ and c are linearly independent$\}$
for $p \in F^i_{\rightarrow}$ **do**
  for $a, b, c \in \mathscr{X}_i$ **do**
    for *each case* B_{i+1} *made from a,b,c* **do**
      $\mathscr{X}_{i+1} \leftarrow span(B_{i+1})$
      if $(\mathscr{X}_{i+1}, \mathscr{X}_{t-j})$ *is closer than* $(\mathscr{X}_i, \mathscr{X}_{t-j})$ **then**
        $\mathscr{D} \leftarrow \mathscr{D} \cup \{(B_{i+1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j})\}$
return $\mathscr{D}$

Algorithm 1 finds the vector space pairs only in the forward direction. If we generate a set $F^i_{\rightarrow}$ that collects all possible target qubit values after the $(i+1)$-th Toffoli layer T_{i+1}, then

$$F^i_{\rightarrow} = \{ab \oplus c | a, b(\neq a), c \in \mathscr{X}_i, \text{if } q = \dim(\mathscr{X}_i), \text{ then } a, b, \text{and } c \text{ are linearly independent}\}.$$

We consider quantum circuits with q-qubit. When $\dim(\mathscr{X}_i) = q$, the values of all qubits at the input point of T_{i+1} form a basis for T_{i+1}, such that the elements of this set are independent of each other.

There can be several combinations of $a, b, c \in \mathscr{X}_i$ that satisfy $ab \oplus c \in F^i_{\rightarrow}$. For each a, b, and c, we can construct a basis B_i of $\mathscr{X}_i$, where B_i must be constructed so that a and b belong to it. If c is linearly independent of $\{a, b\}$, c is also adjusted to belong to B_i. Let d and e be linearly independent of $\{a, b, c\}$ and $\{a, b, c, d\}$, respectively. The cases in which B_i is possible are as follows:

1. When $\dim(\mathscr{X}_i) = 5$, $B_i = \{a, b, c, d, e\}$;
2. When $\dim(\mathscr{X}_i) = 4$ holds and c is linearly dependent of $\{a, b\}$, $B_i = \{a, b, d, e\}$;
3. When $\dim(\mathscr{X}_i) = 4$ holds and a, b, c are linearly independent, $B_i = \{a, b, c, d\}$.

Let B_{i+1} be the basis of $\mathscr{X}_{i+1}$ to be generated. In Case 1, c changes to $ab \oplus c$, resulting in $B_{i+1} = \{a, b, ab \oplus c, d, e\}$. In Case 2, we add a new basis $ab \oplus c$, $B_{i+1} = \{a, b, ab \oplus c, d, e\}$ holds. In Case 3, c can change to $ab \oplus c$ or $ab \oplus c$ can be newly added, and B_{i+1} becomes $\{a, b, ab \oplus c, d\}$ or $\{a, b, ab \oplus c, c, d\}$. If $(\mathscr{X}_{i+1}, \mathscr{X}_{t-j})$ is closer than $(\mathscr{X}_i, \mathscr{X}_{t-j})$ for all cases, we adopt these spaces and store $(B_{i+1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j})$. After this process is performed for all $ab \oplus c$, the stored set of $(B_{i+1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j})$ becomes the output.

4.3 Exploring 4-Bit S-Box Quantum Circuits Using Meet-in-the-Middle Strategy

We describe Algorithm 2 based on the meet-in-the-middle strategy in the same environment as the previous section. When constructing a circuit, if Algorithm 1 outputs an empty set, we proceed with Algorithm 2, which generates $F^i_{\rightarrow}$ in the same way as Algorithm 1. Subsequently, we create a set $F^j_{\leftarrow}$ that collects all the target qubit values that the Toffoli gates in T_{t-j-1} can have.

$$F^j_{\leftarrow} = \{\alpha\beta \oplus \gamma | \alpha, \beta(\neq \alpha), \gamma \in \mathscr{X}_{t-j}, \text{if } \dim\left(\mathscr{X}_{t-j}\right) = q, \text{ then } \alpha, \beta, \text{and } \gamma \text{ are linearly independent}\}.$$

For each $ab \oplus c \in F^i_{\rightarrow}$ and $a, b, c \in \mathscr{X}_i$, we considered $\mathscr{X}_{i+1}$ and B_{i+1} for all cases constructed in the mentioned manner. For each $\alpha\beta \oplus \gamma \in F^j_{\leftarrow}$ and $\alpha, \beta, \gamma \in \mathscr{X}_{t-j}$, the same process can be repeated to obtain $\mathscr{X}_{t-j-1}$ and B_{t-j-1}. If $(\mathscr{X}_{i+1}, \mathscr{X}_{t-j-1})$ is closer than $(\mathscr{X}_i, \mathscr{X}_{t-j})$ for all cases, we adopt these spaces

and store $(B_{i+1}, B_{t-j-1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j-1})$. After this process is conducted for all $ab \oplus c$ and $\alpha\beta \oplus \gamma$, the stored set of $(B_{i+1}, B_{t-j-1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j-1})$ represents the output. Fig. 5 depicts the change in the intersection due to the meet-in-the-middle strategy.

Algorithm 2: Meet-in-the-middle finding for quantum circuits

input: $\mathscr{X}_i$, $\mathscr{X}_{t-j}$, n, q where $n \leq q \leq 5$
output: a set $\mathscr{D}$ of pairs $(B_{i+1}, B_{t-j-1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j-1})$ where B_{i+1} and B_{t-j-1} are the bases of $\mathscr{X}_{i+1}$ and $\mathscr{X}_{t-j-1}$, respectively.

$B_i \leftarrow$ a basis of $\mathscr{X}_i$
$B_{t-j} \leftarrow$ a basis of $\mathscr{X}_{t-j}$
$\mathscr{D} \leftarrow \{\}$
$\mathscr{X}_{set} \leftarrow \{\}$
$F^i_{\rightarrow} \leftarrow \{ab \oplus c | a, b(\neq a), c \in \mathscr{X}_i$, if $\dim(\mathscr{X}_i) = q$, then $a, b,$ and c are linearly independent$\}$
$F^j_{\leftarrow} \leftarrow \{\alpha\beta \oplus \gamma | \alpha, \beta(\neq \alpha), \gamma \in \mathscr{X}_{t-j}$, if $\dim\left(\mathscr{X}_{t-j}\right) = q$, then $\alpha, \beta,$ and γ are linearly independent$\}$
for $p \in F^i_{\rightarrow}$ **do**
  for $a, b, c \in \mathscr{X}_i$ **do**
    for *each case* B_{i+1} *made from a,b,c* **do**
      if $span(B_{i+1}) \notin \mathscr{X}_{set}$ **then**
        $\mathscr{X}_{set} \leftarrow \mathscr{X}_{set} \cup \{span(B_{i+1})\}$
for $\mathscr{X}_{i+1} \in \mathscr{X}_{set}$ **do**
  for $\rho \in F^j_{\leftarrow}$ **do**
    for $\alpha, \beta, \gamma \in \mathscr{X}_{t-j}$ **do**
      for *each case* B_{t-j-1} *made from* α, β, γ **do**
        $\mathscr{X}_{t-j-1} \leftarrow span(B_{t-j-1})$
        if $(\mathscr{X}_{i+1}, \mathscr{X}_{t-j-1})$ is *closer than* $(\mathscr{X}_i, \mathscr{X}_{t-j})$ **then**
          $\mathscr{D} \leftarrow$ set $\cup \{(B_{i+1}, B_{t-j-1}, \mathscr{X}_{i+1}, \mathscr{X}_{t-j-1})\}$
return $\mathscr{D}$

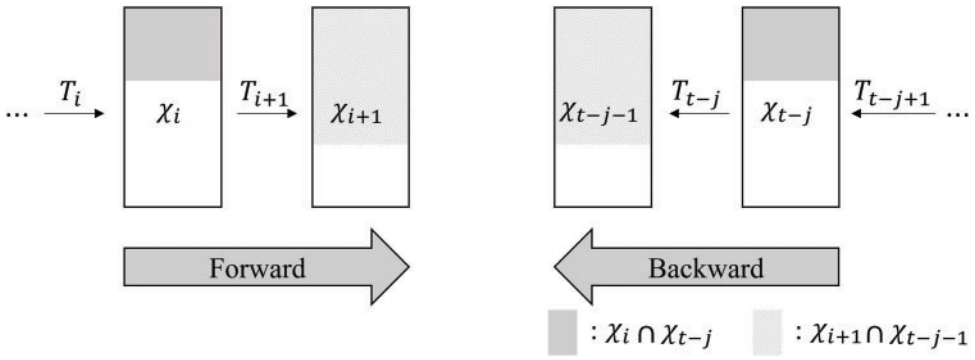

Figure 5: Description of the meet-in-the-middle strategy

The process of constructing $F^i_{\rightarrow}$ and $F^j_{\leftarrow}$ is determined by the dimensions of $\mathscr{X}_i$ and $\mathscr{X}_{t-j}$, respectively. Thus, if the spatial dimension is d_{im}, the computational cost of $F^i_{\rightarrow}$ or $F^j_{\leftarrow}$ is $2^{3d_{im}}$ because a, b, c, α, β, and γ have $2^{d_{im}}$ cases. When applying Algorithm 2, the complexity of finding both $ab \oplus c \in F^i_{\rightarrow}$ and $\alpha\beta \oplus \gamma \in F^j_{\leftarrow}$ is less than $2^{6d_{im}}$ (e.g., if $d_{im} = 5$, the complexity bound is 2^{30}). The memory complexity depends on how many close spaces are stored; thus, it depends on the S-box.

Searching two linear layers is possible only using the forward search, not the meet-in-the-middle strategy. However, the forward search requires a complexity of $2^{6d_{im}}$, so it takes longer than the meet-in-the-middle strategy. This speed difference can be seen experimentally, which is why we use a meet-in-the-middle strategy.

5 Results for Some 4-bit S-Boxes

We applied the proposed algorithms to various 4-bit S-boxes. First, we considered all 4-bit optimal S-boxes classified by Leander and Poschmann [19,20] to demonstrate the validity of the proposed algorithms (see Table 1). Moreover, LIGHTER-R could not find the circuits of odd permutations (i.e., G_3, G_6, G_9, G_{10}, G_{11}, G_{12}, G_{14}, and G_{15}), whereas the proposed algorithm could.

Table 1: Toffoli depths of optimal S-boxes using 5 qubits

Class	G_0	G_1	G_2	G_3	G_4	G_5	G_6	G_7
Toffoli-depth	4	4	4	7	5	5	6	5
Class	G_8	G_9	G_{10}	G_{11}	G_{12}	G_{13}	G_{14}	G_{15}
Toffoli-depth	4	6	6	6	6	5	6	6

Second, we consider the 4-bit S-boxes of GIFT [15], SKINNY [14] and Saturnin [21] (see Table 2). The proposed algorithms and LIGHTER-R output identical Toffoli-depths in the circuit implementation when using 4 qubits. We executed the proposed algorithm using 5 qubits but outputted the same Toffoli-depths. To compare the results of this study with those of existing circuits, we checked the AND-depth, which relates closely to the Toffoli-depth. Quantum circuits with the Toffoli-depth at the same value as the AND-depth always exist, so comparison is possible [10]. These values represented the same AND-depths of the classical implementation, as claimed by the designers of GIFT and SKINNY. The GIFT quantum circuit with 4 qubits was written in Algorithm 3. In the algorithm, x_0 and y_0 are least significant bits, and x_3 and y_3 are most significant bits.

Algorithm 3: Circuits of GIFT S-box

input: (x_0, x_1, x_2, x_3)
output: (y_0, y_2, y_3, y_4)
$(x_0, x_1, x_2, x_3) \leftarrow (x_0 \oplus x_1, x_0, x_2 \oplus x_3, x_1 \oplus x_2)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_0 \oplus x_3, x_1, x_2, x_2 \oplus x_3)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_3, x_0 \oplus x_2 \oplus x_3, x_2, x_1 \oplus x_2)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_2, x_1 \oplus x_2 \oplus x_3, x_0, x_1 \oplus x_2)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_0 \oplus x_1, x_1 \oplus x_3, x_1 \oplus x_2 \oplus x_3, x_1)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_0 \oplus 1, x_1, x_2, x_3)$
return (x_0, x_1, x_2, x_3)

In the case of Saturnin, we found a more efficient circuit with an AND-depth of 5, rather than a circuit with an AND-depth of 6 that the designers found. Fig. 6 is the circuit of Saturnin's S-box σ_0

that we found. We omitted the expression of CNOT gates in L_i. The values of the wires leading to the same output on L_i are XORed.

Table 2: Toffoli depths of special S-boxes using 4 qubits

Cipher	Saturnin	SKINNY	GIFT
Toffoli-depth	5	4	4

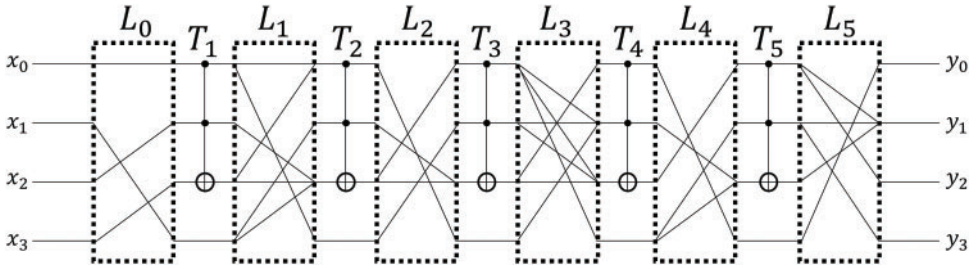

Figure 6: Circuit of Saturnin S-box σ_0

Discussion of the Results Based on the Algorithms. For each i, the proposed algorithms take a pair $(\mathscr{X}_i, \mathscr{X}_{t-j})$ as input and select the closer pair $(\mathscr{X}_{i+1}, \mathscr{Y})$, where $\mathscr{Y}$ can be either $\mathscr{X}_{t-j}$ or $\mathscr{X}_{t-j-1}$. The value of j is determined by the number of times Algorithm 2 is repeated. In the process, a pair that is not closer to any i and j is never selected. This fact incurs a weakness in that the algorithms sometimes fail to find circuits with the minimum Toffoli-depth. However, we can determine the whole circuit's lower bound of Toffoli-depth using Theorem 2. If the algorithms find a circuit with this lower bound, that implies the minimum Toffoli-depth. Furthermore, the algorithms offer the advantage of being able to find all circuits with such a lower bound. This result occurs because, a forward finding can discover a circuit if one with that lower bound exists (see Algorithm 1). If the output circuit does not have the lower bound, then the Toffoli-depth of the S-box is greater than the lower bound. We can confirm that the results of G_3, G_6, G_9, G_{10}, G_{11}, G_{12}, G_{14}, G_{15}, Saturnin, SKINNY, and GIFT are the minimum Toffoli-depth.

Discussion on 6 Qubits. In this case, the method of this paper can be applied as is, and only the number of available Toffoli gates in a Toffoli layer increases. The proposed algorithm uses only one Toffoli gate in a Toffoli layer, so the algorithm selects three qubits. When using 6 qubits, two Toffoli gates must be found; therefore all 6 qubits are selected. Accordingly, the computational cost for the forward direction becomes $2^{6d_{im}}$, and the computational cost for the meet-in-the-middle strategy becomes under $2^{12d_{im}}$.

Discussion on Other Methods. There are tools for creating quantum circuits, including LIGHTER-R and DORCIS [22]. The LIGHTER-R tool is based on the breadth-frist search algorithm and evaluates paths based on the cost entered by the user. In addition, DORCIS is based on LIGHTER-R, but the difference is that it evaluates the path based on the depth of the circuit. In the proposed framework, the vector space pairs corresponding to the forward and backward linear layers become the nodes, and the patterns for selecting the vector space pairs become the paths. In addition, the proposed method evaluates the path based on the intersection of a vector space pair. In the breadth-frist search based algorithm, all CNOT gates become nodes, but this proposed method ignores the CNOT gates, so the search is reduced.

The authors of DORCIS used GIFT's S-box to compare its performance with LIGHTER-R. Both algorithms achieved Toffoli-depth 4 using 4 qubits, and we obtained the same result using the same number of qubits. This implies that our tool can handle all the tasks that existing tools are capable of.

6 Conclusion

This paper presents a new framework to construct quantum circuits of S-boxes according to a limited number of qubits. To construct such circuits, we analyzed the dimension and basis before and after the Toffoli layer to find qubits for which the equations match based on the forward search or the meet-in-the-middle strategy. We employed the proposed tool to find the circuits of 4-bit S-boxes and verified its effectiveness in practice. Through the proposed framework, we discovered all quantum circuits of odd permutations among all 4-bit optimal S-boxes classified by Leander and Poschmann. We also implemented quantum circuits of S-boxes for several well-known block ciphers, in which a more efficient quantum circuit of Saturnin's S-box was found. The proposed technique can be applied to find circuits for S-boxes larger than 4 bits, which is left for future work. This technique contributes to the research field regarding finding optimized quantum circuits of S-boxes.

Acknowledgement: None.

Funding Statement: This research was supported by the MSIT (Ministry of Science and ICT), Republic of Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-RS-2022-00164800) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Author Contributions: All authors contributed to the entire process from theoretical design to paper writing. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The results of the proposed framework for optimal 4-bit S-boxes have been added to the appendix.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Shor, P. W. Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings of 35th Annual Symposium on Foundations of Computer Science; 20–22 Nov 1994; IEEE Computer Society.
2. NIST. Post-quantum cryptography standardization; 2016. Available from: https://csrc.nist.gov/projects/post-quantum-cryptography. [Accessed 2024].
3. Grover LK. A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing; 1996; New York, NY, USA: Association for Computing Machinery. p. 212–9.
4. Simon DR. On the power of quantum computation. SIAM J Comput. 1997;26(5):1474–83. doi:10.1137/S0097539796298637.

5. Grassl M, Langenberg B, Roetteler M, Steinwandt R. Applying grover's algorithm to AES: quantum resource estimates. In: Takagi T, editor. Post-Quantum Cryptography—7th International Workshop. Cham: Springer; 2016.
6. Langenberg B, Pham H, Steinwandt R. Reducing the cost of implementing the advanced encryption standard as a quantum circuit. IEEE Trans Quantum Eng. 2020;1:1–12. doi:10.1109/TQE.2020.2965697.
7. Zou J, Wei Z, Sun S, Liu X, Wu W. Quantum circuit implementations of AES with fewer qubits. In: Moriai S, Wang H, editors. Advances in Cryptology—ASIACRYPT 2020—26th International Conference on the Theory and Application of Cryptology and Information Security; 2020; Daejeon, Republic of Korea: Springer.
8. Jaques S, Naehrig M, Roetteler M, Virdia F. Implementing grover oracles for quantum key search on AES and lowmc. In: Canteaut A, Ishai Y, editors. Advances in Cryptology-EUROCRYPT 2020—39th Annual International Conference on the Theory and Applications of Cryptographic Techniques; 2020; Zagreb, Croatia: Springer.
9. Lin D, Xiang Z, Xu R, Zhang S, Zeng X. Optimized quantum implementation of AES. Quant Inf Process. 2023;22(9):352. doi:10.1007/s11128-023-04043-9.
10. Huang Z, Sun S. Synthesizing quantum circuits of AES with lower t-depth and less qubits. In: Agrawal S, Lin D, editors. Advances in Cryptology—ASIACRYPT 2022—28th International Conference on the Theory and Application of Cryptology and Information Security; 2022; Taipei, Taiwan: Springer. doi: 10.1007/978-3-031-22969-5.
11. Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, et al. PRESENT: an ultra-lightweight block cipher. In: Paillier P, Verbauwhede I, editors. Cryptographic Hardware and Embedded Systems—CHES 2007, 9th International Workshop; 2007; Vienna, Austria: Springer.
12. Albrecht MR, Driessen B, Kavun EB, Leander G, Paar C, Yalçin T. Block ciphers—focus on the linear layer (feat. PRIDE). In: Garay JA, Gennaro R, editors. Advances in Cryptology—CRYPTO 2014—34th Annual Cryptology Conference; 2014; Santa Barbara, CA, USA: Springer.
13. Zhang WT, Bao ZZ, Lin DD, Rijmen V, Yang BH, Verbauwhede I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci China Inf Sci. 2015;58(12):1–15. doi:10.1007/s11432-015-5459-7.
14. Beierle C, Jean J, Kölbl S, Leander G, Moradi A, Peyrin T, et al. The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw M, Katz J, editors. Advances in Cryptology—CRYPTO 2016—36th Annual International Cryptology Conference; 2016; Santa Barbara, CA, USA: Springer.
15. Banik S, Pandey SK, Peyrin T, Sasaki Y, Sim SM, Todo Y. GIFT: a small present - towards reaching the limit of lightweight encryption. In: Fischer W, Homma N, editors. Cryptographic Hardware and Embedded Systems—CHES 2017—19th International Conference; 2017; Taipei, Taiwan: Springer.
16. Dasu VA, Baksi A, Sarkar S, Chattopadhyay A. LIGHTER-R: optimized reversible circuit implementation for sboxes. In: 32nd IEEE International System-on-Chip Conference; 2019; Singapore: IEEE.
17. Shende VV, Prasad AK, Markov IL, Hayes JP. Synthesis of reversible logic circuits. IEEE Trans Comput Aided Des Integr Circuits Syst. 2003;22(6):710–22.
18. Jiang J, Sun X, Teng S, Wu B, Wu K, Zhang J. Optimal space-depth trade-off of CNOT circuits in quantum logic synthesis. In: Chawla S, editor. Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020; 2020; Salt Lake City, UT, USA: SIAM.
19. Leander G, Poschmann A. On the classification of 4 bit s-boxes. In: Carlet C, Sunar B, editors. Arithmetic of Finite Fields, First International Workshop, WAIFI 2007; 2007; Madrid, Spain: Springer.
20. Zhang W, Bao Z, Rijmen V, Liu M. A new classification of 4-bit optimal s-boxes and its application to present, rectangle and spongent. In: Fast Software Encryption: 22nd International Workshop, FSE 2015; 2015; Istanbul, Turkey: Springer.
21. Canteaut A, Duval S, Leurent G, Naya-Plasencia M, Perrin L, et al. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. IACR Trans Symmetric Cryptol. 2020;2020(S1):160–207.

22. Chun M, Baksi A, Chattopadhyay A. DORCIS: depth optimized quantum implementation of substitution boxes. Available from: https://eprint.iacr.org/2023/286. [Accessed 2023].

Appendix A: Results for the 4-Bit Optimal S-Boxes

Algorithms 4–11 are the results of the proposed framework for the optimal S-boxes $G_0, G_1, \cdots, G_{15}$. In inputs/outputs, x_0 and y_0 are least significant bits, and the rightmost 0 represents ancilla qubits. The line below refers to the Toffoli layer.

$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$.

All lines except the Toffoli layers refer to linear layers.

Algorithm 4: Circuits of G_0 (left), G_1 (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_1 \oplus x_2, x_1 \oplus x_3, x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_0 \oplus x_1 \oplus x_2, x_2, x_0 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_1 \oplus x_2, x_0, x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_2, x_0, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_3, x_0 \oplus x_3, x_1 \oplus x_2, x_0 \oplus x_2 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_2 \oplus x_3, x_1 \oplus x_3, x_0 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_0 \oplus x_3, x_0, x_1 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_2, x_3, x_2, x_0, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_1, x_3, x_0 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_0 \oplus x_2 \oplus x_3, x_0 \oplus x_1, x_0 \oplus x_1 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

Algorithm 5: Circuits of G_2 (left), G_3 (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_3, x_1 \oplus x_3, x_0 \oplus x_1 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1, x_0 \oplus x_3, x_0, x_0 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_0 \oplus x_1 \oplus x_2, x_2, x_0, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_0, x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_2, x_2 \oplus x_3, x_0 \oplus x_3, x_1 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_1 \oplus x_3, x_1 \oplus x_3, x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_1 \oplus x_3, x_3, x_0, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_1 \oplus x_2, 0, x_2, x_0 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3 \oplus x_4, x_0 \oplus x_1 \oplus x_4, x_3, x_4, x_0 \oplus x_2 \oplus x_3)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4, x_2, x_0 \oplus x_2, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_4, x_1 \oplus x_3, x_2, x_2 \oplus x_3, x_0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_4, x_0 \oplus x_1 \oplus x_3, x_2, x_2 \oplus x_3, x_0 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_3, x_0 \oplus x_3 \oplus x_4, x_0, x_0 \oplus x_1, 0)$
$(x_0, x_1, x_2, x_3, 0)$

Algorithm 6: Circuits of G_4 (left), G_5 (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_1 \oplus x_3, x_1 \oplus x_3, x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_3, x_3, x_1, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_2, x_3, x_2, x_0 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2 \oplus x_3, x_1 \oplus x_3, x_2, x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_0 \oplus x_3, x_2 \oplus x_3, x_1 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_2 \oplus x_3, x_0 \oplus x_3, x_0 \oplus x_1, x_0, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1, x_0 \oplus x_1 \oplus x_3, x_2 \oplus x_3, x_1 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_3, x_0 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_1 \oplus x_3, x_0, x_0 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_2 \oplus x_3, x_1, x_0 \oplus x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_0 \oplus x_2, x_0, x_0 \oplus x_1, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1, x_0 \oplus x_2 \oplus x_3, x_1 \oplus x_3, x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

Algorithm 7: Circuits of G_6 (left), G_7 (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_2, x_0 \oplus x_1 \oplus x_2, x_0 \oplus x_3, x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_1 \oplus x_3, x_0 \oplus x_3, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_2, 0, x_3, x_1 \oplus x_3)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_1 \oplus x_2, x_2 \oplus x_4, x_2, x_0 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3 \oplus x_4, x_1 \oplus x_3, x_3, x_0, x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_3, x_4, x_3, x_0, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_3, x_1 \oplus x_3, x_0 \oplus x_2, x_2 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_2, x_0 \oplus x_3, x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_1 \oplus x_2 \oplus x_3, x_1 \oplus x_2, x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_2, x_0, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_1 \oplus x_2 \oplus x_3, x_2, x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_2, x_1 \oplus x_2, x_0 \oplus x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_0 \oplus x_3, x_2, x_0 \oplus x_1, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

Algorithm 8: Circuits of G_8 (left), G_9 (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_2, x_0 \oplus x_3, x_1 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_1, x_0, x_0 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_1, x_2, x_0, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_1 \oplus x_2 \oplus x_3, x_2, x_0 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_0, x_1, x_1 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_0, x_2 \oplus x_3, x_1 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_1 \oplus x_2, x_2, x_0 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_3, 0, x_0, x_1)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_2, x_2 \oplus x_4, x_2, x_0 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3 \oplus x_4, x_1 \oplus x_4, x_3, x_0, x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_3 \oplus x_4, x_3, x_1, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_3, x_1, x_0 \oplus x_1 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

Algorithm 9: Circuits of G_{10} (left), G_{11} (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_0 \oplus x_1 \oplus x_2, x_2 \oplus x_3, x_1 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1 \oplus x_3, x_2 \oplus x_3, x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_1 \oplus x_3, 0, x_3, x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1, x_0 \oplus x_1, 0, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_2, x_0 \oplus x_2 \oplus x_4, x_0 \oplus x_2, x_0 \oplus x_3 \oplus x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_3, x_0 \oplus x_3 \oplus x_4, x_2 \oplus x_3, x_0, x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_1 \oplus x_3, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_2 \oplus x_3, x_1 \oplus x_2, x_1 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_0, x_2 \oplus x_3, x_1 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_0 \oplus x_1 \oplus x_2 \oplus x_3, 0, x_2 \oplus x_3, x_0 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1, x_0 \oplus x_1, 0, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2 \oplus x_3, x_0 \oplus x_3, x_0 \oplus x_2 \oplus x_3 \oplus x_4, x_0 \oplus x_2, x_1 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3 \oplus x_4, x_0 \oplus x_1 \oplus x_3, x_3, x_0, x_2 \oplus x_3)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_4, x_1 \oplus x_3 \oplus x_4, x_0, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_1 \oplus x_3, x_0, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_3, x_1, x_2 \oplus x_3, x_0 \oplus x_2 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

Algorithm 10: Circuits of G_{12} (left), G_{13} (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_2 \oplus x_3, x_2 \oplus x_3, x_1 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2, x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_2 \oplus x_3, x_0 \oplus x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_0 \oplus x_2 \oplus x_3, x_0, x_0 \oplus x_1 \oplus x_2 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_2, x_3, x_0 \oplus x_2, x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2, x_1 \oplus x_3, x_2 \oplus x_3, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_2, x_3, x_0 \oplus x_2, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_0 \oplus x_2 \oplus x_3, x_2 \oplus x_3, x_1 \oplus x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2 \oplus x_4, x_1 \oplus x_2 \oplus x_3, x_2, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_2 \oplus x_4, x_3, x_2 \oplus x_3 \oplus x_4, x_0 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_1 \oplus x_2 \oplus x_3, x_2, x_3, x_2 \oplus x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3 \oplus x_4, x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4, x_3, x_2, x_0 \oplus x_3)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2 \oplus x_3 \oplus x_4, x_1 \oplus x_2 \oplus x_4, x_2 \oplus x_4, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1 \oplus x_3, x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_3, x_0 \oplus x_2, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

Algorithm 11: Circuits of G_{14} (left), G_{15} (right)

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_1, x_1 \oplus x_3, x_0 \oplus x_1 \oplus x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1 \oplus x_2, x_3, x_2, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_1 \oplus x_2, 0, x_0, x_1)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_4, x_0 \oplus x_1 \oplus x_2 \oplus x_4, x_2 \oplus x_3, x_2, x_0 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3 \oplus x_4, x_0 \oplus x_1 \oplus x_4, x_3, x_2, x_0 \oplus x_3)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3, x_1 \oplus x_3 \oplus x_4, x_4, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_2 \oplus x_3, x_1, x_2 \oplus x_3, x_0 \oplus x_1 \oplus x_3, 0)$
return $(x_0, x_1, x_2, x_3, 0)$

input: $(x_0, x_1, x_2, x_3, 0)$
output: $(y_0, y_2, y_3, y_4, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_2 \oplus x_3, x_0 \oplus x_1, 0, x_1 \oplus x_3, x_0 \oplus x_1 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_1, x_0 \oplus x_1, 0, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3, x_0 \oplus x_1 \oplus x_2 \oplus x_3, x_0 \oplus x_2 \oplus x_4, x_0 \oplus x_2, x_1 \oplus x_2)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_4, x_2 \oplus x_4, x_4, x_3, x_0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_4, x_1 \oplus x_2 \oplus x_3, x_2, x_3, x_0 \oplus x_3 \oplus x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_3 \oplus x_4, x_1, x_3, x_2, x_0 \oplus x_3)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_3 \oplus x_4, x_1 \oplus x_3, x_0 \oplus x_4, x_3, 0)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0, x_1, x_2 \oplus x_0x_1, x_3, x_4)$
$(x_0, x_1, x_2, x_3, x_4) \leftarrow (x_0 \oplus x_1 \oplus x_3, x_0 \oplus x_2 \oplus x_3, x_1 \oplus x_3, x_2, 0)$
return $(x_0, x_1, x_2, x_3, 0)$