On the construction of quantum circuits for S-boxes with different criteria based on the SAT solver

Da Lin[1], Chunli Yang[2], Shengyuan Xu[3], Shizhu Tian[2], Bing Sun[1*]

[1]College of Science, National University of Defense Technology, Changsha, 410073, China.
[2]School of Cyber Science and Technology, Hubei University, Wuhan, 430062, China.
[3]Department of Fundamental Courses, Shandong University of Science and Technology, Taian, 271019, China.

*Corresponding author(s). E-mail(s): happy_come@163.com;
Contributing authors: linda_is@163.com; yyli@stu.hubu.edu.cn;
xushengyuan@sdust.edu.cn; Shizhutian@hubu.edu.cn;

Abstract

The substitution box (S-box) is often used as the only nonlinear component in symmetric-key ciphers, leading to a significant impact on the implementation performance of ciphers in both classical and quantum application scenarios by S-box circuits. Taking the Pauli-X gate, the CNOT gate, and the Toffoli gate (i.e., the NCT gate set) as the underlying logic gates, this work investigates the quantum circuit implementation of S-boxes based on the SAT solver. Firstly, we propose encoding methods of the logic gates and the NCT-based circuit, based on which we construct STP models for implementing S-boxes. By applying the proposed models to the S-boxes of several well-known cryptographic algorithms, we construct optimal implementations with different criteria for the 4-bit S-boxes and provide the implementation bounds of different criteria for the 5-bit S-boxes. Since S-boxes in the same affine equivalence class share most of the important properties, we then build STP models to further investigate optimizing S-box circuits based on affine equivalence. According to the applications, for almost all the tested 4-bit S-boxes, there always exists an equivalent S-box that can be implemented with half the number of logic gates. Besides, we encode some important cryptographic properties and construct STP models to design S-boxes with given criteria configurations on implementation and properties. As an application, we

find an S-box with the same cryptographic properties as the S-box of KECCAK that can be implemented with only 5 NCT gates, even though the application of our models indicates that implementing the KECCAK S-box requires more than 9 NCT gates. Notably, the inputs of the proposed models are tweakable, which makes the models possess some functions not currently available in the public tools for constructing optimized NCT-based circuits for S-boxes.

Keywords: S-box, implementation, the NCT gate set, the SAT solver

1 Introduction

As an important part of modern cryptography, the symmetric-key cryptographic primitives is a key technique to ensure the confidentiality and integrity of information, and thus its security is crucial. When designing a symmetric-key cryptographic primitive, it is necessary to ensure that the algorithm has the ability to resist known cryptographic attacks, such as differential cryptanalysis [1] and linear cryptanalysis [2]. With the rapid development of quantum technology, the symmetric-key cryptographic primitive also has to face the threat of quantum attacks, as the parallelism of quantum computing makes quantum computers possess advantages over classical ones in dealing with some specific problems [3–5].

Considering the threat posed in the forthcoming post-quantum era, the optimized quantum implementation of the symmetric-key cryptographic primitive has become an active research topic in recent years, such as [6–9]. The research interest of the cryptographic community in this field is mainly aroused by the fact that the quantum implementation of a cipher is an important component of the circuit for evaluating its quantum security level [10, 11] or conducting quantum attacks against it. In addition, considering the limited scale of the current quantum computers, the optimized quantum implementation circuits of cryptographic algorithms can be instantiated earlier in the future when quantum computers become robust enough.

For symmetric-key cryptographic primitives, *confusion* and *diffusion* [12] are two important cryptographic criteria, the establishment of which is primarily dependent on the nonlinear layer and the linear layer respectively. Generally, the Substitution box (S-box) is often used as the only nonlinear component of the symmetric-key cryptographic primitive, and thus its implementation and properties have great impacts on that of the whole cipher.

To construct optimized implementation schemes for S-boxes, there are many researches focusing on the development of tools. In 2017, Jean *et al.* [13] introduced LIGHTER, a tool that aims at implementing small S-boxes with small area in hardware application scenarios or fewer instructions in software application scenarios. To this end, several standard cell libraries are encoded in the tool, as well as the corresponding cost metric of each instruction. Later, the platform PEIGEN was presented at ToSC 2019 in [14]. In addition to constructing efficient hardware/software implementations of S-boxes, PEIGEN can also be applied to evaluate security and generate S-boxes. Notably, the instructions encoded in LIGHTER are consistent with the *in-place*

property of quantum logic gates. As a consequence, the reversible implementation returned by `LIGHTER` can also be applied to construct quantum circuits, especially the ones built with the NCT gate set [15–17]. Following the line of the research on constructing optimized implementations for S-boxes, `LIGHTER-R` [18] and DORCIS [19] are built based on `LIGHTER` for quantum application scenarios. The former constructs quantum circuits at the top level with the NCT gate set, while the latter considers the decomposition of the Toffoli gate and constructs Clifford+T-based circuits.

Most of the above-mentioned tools are based on heuristic algorithms. Although they can return good solutions, but it is relatively hard to prove those solutions are optimal. There are researches on searching for the optimal classical circuit implementations of S-boxes. In [20], Stoffelen defined a set of equations to represent the Algebraic Normal Form (ANF) of the S-box and converted them into Conjunctive Normal Form (CNF). By applying the SAT solver, the author constructed optimal implementations for several small S-boxes under various criteria. Later, Bilgin *et al.* [21] proposed methods to optimize the AND-depth, and Fan *et al.* [22] introduced 3-input gates to optimize the area. Recently, Feng *et al.* [23] proposed two methods to encode the implementations of S-boxes and introduced new SAT-based search methods.

In addition to the construction of optimized implementation schemes, the SAT solver has also been used to design new S-boxes [24]. In this case, some important cryptographic properties of S-boxes should be considered, such as differential uniformity, linearity, and their frequency. Bijective S-boxes with lowest differential uniformity and linearity are considered optimal, as they help to resist differential and linear cryptanalysis respectively. Besides, the frequency of differential uniformity in the differential distribution table (DDT) and that of linearity in the linear approximation table (LAT) are also considered to have important impacts on the attacks [24, 25]. Additionally, the number of active S-boxes is closely related to the probability of differential and linear attacks, a small number of active S-boxes will increase the probability of a successful attack. Therefore, the input and output differentials/masks of S-boxes with a Hamming weight of 1 (named bad input and bad output pattern, i.e., BIBO pattern, in [24]) would serve to perform differential/linear attacks.

This work investigates the optimization of S-box implementations constructed based on the NCT gate set by applying the SAT solver. It should be noted that the researches in [26] reveal that at least one additional ancilla qubit is required for implementing an odd permutation with NCT gates. In this work, we only focus on constructing optimized NCT-based circuits for even permutations for saving qubits.

Firstly, we encode the NCT gate set and the circuits built based on those gates, based on which we construct STP models to identify the optimal implementations of S-boxes for different criteria. As an application, we apply the proposed models to the 4-/5-bit S-boxes adopted in several well-known cryptographic algorithms. The results reveal that for those 4-bit S-boxes, the optimal implementations for all criteria can be constructed. For the 5-bit S-boxes, although the SAT solver cannot return their optimal implementation schemes in a reasonable time, the bounds of different criteria can be provided.

Moreover, we further investigate optimizing S-box implementations based on affine equivalence. The results show that the optimal implementation of a given 4-bit S-box

is not necessarily optimal among implementations of all the S-boxes in the same affine equivalence class. For most tested 4-bit S-boxes, there is always an S-box with/without fixed points and affine equivalent to the tested one but can be implemented with only about half of the gate count. For the 4-bit S-boxes with/without fixed points and affine equivalent to the optimal ones corresponding to even permutations, at least 4/5 gates are required to build their NCT-based circuits.

Additionally, by combing the encoding methods of the NCT-based circuits and the S-box properties, we construct models to search for S-boxes with given constraints on the implementation and various cryptographic properties. In our experiments, the SAT solver returns 3456 optimal 4-bit S-boxes with fixed points. Further observations show that only 4 NCT gates are enough to implement the S-boxes. Besides, each of those S-boxes contains the fixed point $S[0] = 0$. By placing restrictions on the Toffoli gate consumption, it can be verified that at least 4 Toffoli gates are required to construct NCT-based circuits for the optimal 4-bit S-boxes, which also implies that those S-boxes can be implemented at least in a Toffoli-depth of 4 without introducing additional ancilla qubits.

The rest of this paper is organized as follows: In Sect. 2, notations used throughout the paper, the SAT solver, some cryptographic properties of the S-box, and the NCT gate set are introduced. Section 3 provides the models of encoding the NCT gates and NCT-based circuits, as well as that for optimizing S-box circuits. The investigation of optimizing the NCT-based circuits of the S-box based on its properties is given in Sect. 4. Section 5 presents the application and Sect. 6 concludes the work.

2 Preliminaries

2.1 Notations

Notations used throughout this paper are presented in Table 1.

Table 1 Notations.

Notation	Description
$\mathbb{F}_2$	The finite field with two elements 0 and 1
$\mathbb{F}_2^n$	The n-dimensional vector space over $\mathbb{F}_2$
$a \oplus b$	The Xor of bits a and b over $\mathbb{F}_2$
$a \cdot b$	The And of bits a and b over $\mathbb{F}_2$
$wt(a)$	The Hamming weight of a
$a\|\|b$	The concatenation of a and b
$\bigwedge$	Logical operation And
$\bigvee$	Logical operation Or

2.2 The SAT Problem and Its Extension

The Boolean Satisfiability (SAT) problem studies whether there is a set of Boolean variables whose assignment satisfies the given Boolean formulas. If so, the problem

is said to be satisfiable; otherwise, it is said to be unsatisfiable. The SAT problem is a deterministic problem and has been proven to be Non-deterministic Polynomial Complete (NPC).

Usually, for a given Boolean formula, researchers first represent it as the standard input format of SAT solver, namely the form of CNF $\bigwedge_{i=0}^{m-1}\left(\bigvee_{j=0}^{n_i-1} x_{i,j}\right)=1$, where $x_{i,j}$ are variables, constants or the negation of Boolean variables. There are also many mature and efficient solvers for SAT problems, such as Minisat [27], Cryptominisat [28], and CaDiCal [29].

The Satisfiability Modulo Theories (SMT) problem is an extension of the SAT problem, in which certain Boolean variables are substituted with a suitable set of binary and (or) non-binary variables. STP [30] is a constraint solver for the theory of quantifier-free bitvectors that can solve the SMT problem, which is mainly used in previous researches. For STP, the CVC format is one of the frequently employed file-based input languages. For additional information, please refer to http://stp.github.io/.

2.3 Cryptographic Properties of the S-box

When designing an S-box, the cryptographic properties of the S-box should be extensively analyzed, as they can propagate along with the round function and may eventually affect the security of the cipher. Differential cryptanalysis and linear cryptanalysis are the two most well-known attack methods. To judge the ability of cryptographic algorithms to resist those attacks, special attention has been paid to differential uniformity and linearity of the S-box.

The differential uniformity of an S-box is defined by its DDT.

Definition 1. *(Differential Distribution Table, Differential Uniformity [31]) Let* $S:\mathbb{F}_2^n\to\mathbb{F}_2^m$ *be a vectorial Boolean function. For any* $a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^m$*, define*

$$\delta_S(a,b)=\#\{x\in\mathbb{F}_2^n|S(x)\oplus S(x\oplus a)=b\}. \tag{1}$$

The differential distribution table of S *is a 2-dimensional table with size* $2^n\times 2^m$*, in which the element* $DDT[a,b]$ *in row* a *and column* b *is equal to* $\delta_S(a,b)$*, where* $a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^m$*. The differential uniformity of* S *is defined as:*

$$\mathcal{U}(S)\triangleq\max_{a\in\mathbb{F}_2^n\setminus\{0\},b\in\mathbb{F}_2^m}\delta_S(a,b). \tag{2}$$

Similarly, the linearity of an S-box is defined by its LAT.

Definition 2. *(Linear Approximation Table, Linearity of an S-box [32]) Let* $S:\mathbb{F}_2^n\to\mathbb{F}_2^m$ *be a vectorial Boolean function. For any* $a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^m$*, define*

$$\lambda_S(a,b)=\#\{x\in\mathbb{F}_2^n|a\cdot x\oplus b\cdot S(x)=0\}=\frac{1}{2}\sum_{x\in\mathbb{F}_2^n}(-1)^{a\cdot x\oplus b\cdot S(x)}. \tag{3}$$

The linear approximation table of S *is a* $2^n\times 2^m$ *table with the element* $LAT[a,b]$ *in row* a *and column* b *equals* $\lambda_S(a,b)$*, where* $a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^m$*. The linearity of* S *is defined*

as:

$$L(S) \triangleq \max_{a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^m\setminus\{0\}} |2\cdot\lambda_S(a,b)|. \tag{4}$$

Notably, for the S-boxes in the same affine equivalence class, the differential uniformity and the linearity can be preserved.

Definition 3. *(Affine Equivalence) Two S-boxes S_1 and S_2 are affine equivalent, if there exist two reversible linear transformations A, $B \in GL(n,\mathbb{F}_2)$ and two constants $a, b \in \mathbb{F}_2^n$ such that*

$$S_2 = B(S_1(A(x)\oplus a))\oplus b. \tag{5}$$

In addition, the frequency of the differential uniformity in DDT and that of the linearity in LAT, as well as BIBO patterns derived from DDT and LAT have also received much attention, as they may affect the probability of a successful attack.

Definition 4. *(Frequency of Differential Uniformity) For the DDT of a given vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the frequency of differential uniformity in DDT is defined as:*

$$\mathcal{U}_{Freq} \triangleq \#\{(a,b)|\delta_S(a,b) = \mathcal{U}(S), a \in \mathbb{F}_2^n\setminus\{0\}, b\in\mathbb{F}_2^m\}. \tag{6}$$

Definition 5. *(Frequency of Linearity) For the LAT of a given vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the frequency of linearity in LAT is defined as:*

$$\mathcal{L}_{Freq} \triangleq \#\{(a,b)||2\cdot\lambda_S(a,b)| = \mathcal{L}(S), a \in \mathbb{F}_2^n, b\in\mathbb{F}_2^m\setminus\{0\}\}. \tag{7}$$

Definition 6. *(Bad Input and Bad Output Pattern [24]) Given the DDT/LAT of a vectorial Boolean function $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, if there exists a pair $(a,b) \in \mathbb{F}_2^n \times \mathbb{F}_2^m$ satisfying that DDT[a, b]$\neq$ 0/LAT[a, b]$\neq$ 0 and $wt(a) = wt(b) = 1$. Then, (a,b) is called a bad input and bad output (BIBO) pattern in DDT/LAT. Denote by $\#\mathrm{BIBO}_{\mathrm{DDT}}$ and $\#\mathrm{BIBO}_{\mathrm{LAT}}$ the number of BIBO patterns in the DDT and LAT of S. Then, $\#\mathrm{BIBO}_{\mathrm{DDT}}$ and $\#\mathrm{BIBO}_{\mathrm{LAT}}$ can be calculated as:*

$$\begin{cases} \#\mathrm{BIBO}_{\mathrm{DDT}} = \#\{(a,b)|\mathrm{DDT}[a,b]\neq 0, wt(a)=wt(b)=1, a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^m\}, \\ \#\mathrm{BIBO}_{\mathrm{LAT}} = \#\{(a,b)|\mathrm{LAT}[a,b]\neq 0, wt(a)=wt(b)=1, a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^m\}. \end{cases} \tag{8}$$

2.4 The NCT-based Circuit and Its Criteria

The NCT-based circuit is constructed based on the NCT gate set.

The NCT Gate Set The NCT gate set consists of the Pauli-X gate, the CNOT gate and the Toffoli gate, the circuit symbols of which are depicted in Fig. 1, where $a, b, c \in \mathbb{F}_2$.

- The Pauli-X gate (or the NOT gate) is a single-qubit gate, and reverses the state of the qubit, i.e., transforms $|a\rangle$ to $|a\oplus 1\rangle$.
- The CNOT gate, also known as the Controlled-NOT gate or the C-NOT gate, reverses the state of the target qubit if the state of the control qubit is $|1\rangle$. The function of the CNOT gate can be represented as the mapping that transforms $|a\rangle|b\rangle$ to $|a\rangle|b\oplus a\rangle$.

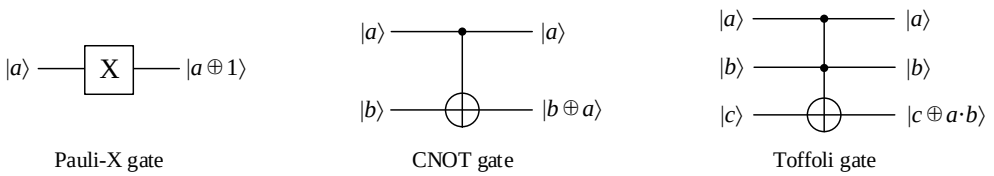

Fig. 1 The description of the NCT gates.

- The Toffoli gate, or the CC-NOT gate, reverses the state of the target qubit when both the control qubits are with state $|1\rangle$. The function of the Toffoli gate can be represented as the mapping that transforms $|a\rangle|b\rangle|c\rangle$ to $|a\rangle|b\rangle|c \oplus a \cdot b\rangle$.

Implementation Criteria Efficient implementations of S-boxes are of much importance, as they largely determine the implementation performance of the whole algorithm. Consequently, various criteria are considered when designing optimized implementation schemes for S-boxes. For example, to minimize the latency of the hardware implementation, depth is taken as the metric. For quantum applications, when building `LIGHTER-R` [18], gate count, two-qubit cost, and quantum cost are taken as the implementation criteria of NCT-based circuits. Actually, evaluating those criteria is equivalent to counting the number of different logic gates for implementing a specific function. Roughly, the criterion gate count considers the number of the NCT gates, the two-qubit cost is defined based on the number of 2-input gates, the quantum cost counts the number of Pauli-X, CNOT, and Controlled-V/-V† gates (see [33] for details). According to [18, 33], the costs of NCT gates under different criteria are listed in Table 2.

Table 2 Cost metrics of the NCT gates

criterion \ gate	Pauli-X	CNOT	Toffoli
gate count	1	1	1
two-qubit cost	0	1	5
quantum cost	1	1	5

In the NCT gate set, the implementation cost of the Toffoli gate is much more expensive than that of other gates. Therefore, the Toffoli gate consumption and the Toffoli-Depth of the NCT-based circuit have also attracted a lot of attention. For an NCT-based circuit, its Toffoli gate consumption counts the number of Toffoli gates required for implementing the circuit, while its Toffoli-Depth equals the layer of Toffoli gates in the circuit that cannot be applied in parallel.

3 Optimizing the S-box Implementation

In this section, the encoding equations related to the input and the output of the underlying logic gates are introduced, based on which the construction of the STP

models for searching for optimized S-box implementation schemes with various criteria is presented.

3.1 New Encoding Method

The methods of encoding quantum gates are quite different from those of classical ones, as quantum gates are reversible. Besides, when encoding quantum gates, the various number of gate inputs should also be considered. For instance, in [20], S-box implementations are restricted to the gates with input number less than 2 for optimizing various criteria, such as the NOT gate, the AND gate, and the OR gate. However, in the NCT gate set, the Toffoli gate is a 3-input gate. Additionally, for multi-input quantum gates, especially the CNOT gate and the Toffoli gate in the NCT gate set, the inputs are divided into the control qubit and the target qubit, which makes more cases should be considered when encoding the NCT gates. To further explain the logic gates that may be involved in each operation, the following example is given.

Example 1. *Given a 4-bit S-box, the input of which is denoted by* (x_0, x_1, x_2, x_3)*. In each operation, the corresponding NCT gate may change any of the S-box inputs. Suppose that the change occurs in* x_0*, i.e.,* x_0 *is applied as the target qubit, the possible quantum gate corresponding to the current operation may act as the following three different classes:*

- *the Pauli-X gate: As a single-qubit gate, the Pauli-X gate can only overwrite the state of* x_0 *as* $x_0 = x_0 \oplus 1$*, as the first gate shown in Fig. 2.*
- *the CNOT gate: In this case, the state of* x_0 *can be overwritten as* $x_0 = x_0 \oplus x_1$*,* $x_0 = x_0 \oplus x_2$ *or* $x_0 = x_0 \oplus x_3$*, as shown with the 2nd, 3rd and 4th gates in Fig. 2.*
- *the Toffoli gate: The state of* x_0 *can be overwritten by the Toffoli gate as* $x_0 = x_0 \oplus x_1 \cdot x_2$*,* $x_0 = x_0 \oplus x_1 \cdot x_3$ *or* $x_0 = x_0 \oplus x_2 \cdot x_3$*, as shown in Fig. 2 with the last 3 gates.*

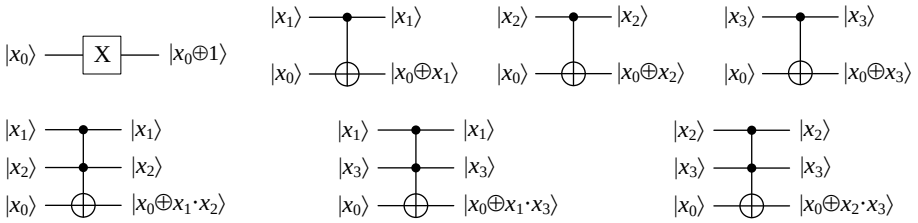

Fig. 2 Possible gates that $|x_0\rangle$ is used as the target qubit.

From Example 1, when implementing a 4-bit S-box with the NCT gate set, there are 7 specific logic gates can be selected as the candidate of the current operation once the target qubit is fixed. It also means that there are a total of 28 different NCT gates can be chosen by each operation as the candidate. Suppose that k gates are consumed for implementing a 4-bit S-box, the complexity of conducting an exhaustive search for an optimal implementation scheme of the S-box is 28^k. For instance, for the case that

$k = 8$, the time complexity is approximately $2^{38.46}$. However, in fact, many S-boxes require more than 8 logic gates, making the calculation even more challenging.

Encoding the NCT Gates Three logic gates with different numbers of inputs are contained in the NCT gate set. Denote by q_0^i, q_1^i, q_2^i the inputs of the i-th gate and b_0^i, b_1^i, b_2^i the variables that determine which NCT gate the i-th gate corresponds to. We fix q_0^i as the target qubit of the i-th gate, that is, q_0^i will be overwritten by the current gate. We also use $+$ to represent $\oplus$ in NCT gates without confusion. To encode the NCT gate set, we define the output of the i-th gate, denoted by t^i, as follows:

$$t^i = q_0^i + b_0^i + b_1^i \cdot q_1^i + b_2^i \cdot q_1^i \cdot q_2^i. \tag{9}$$

To maintain the *in-place* property of the quantum gate, the t^i in Eq. (9) will be assigned to the input determined by q_0^i (the encoding of which will be introduced later). The expression of t^i with different values of b_0^i, b_1^i and b_2^i are listed in Table 3.

Table 3 Encoding the NCT gate set

(b_2^i, b_1^i, b_0^i)	t^i	corresponding quantum gate
$(0, 0, 1)$	$q_0^i + 1$	the Pauli-X gate
$(0, 1, 0)$	$q_0^i + q_1^i$	the CNOT gate
$(1, 0, 0)$	$q_0^i + q_1^i \cdot q_2^i$	the Toffoli gate

Denote by f the vectorial Boolean function with input $(x_0, x_1, \cdots, x_{n-1})$. Based on Eq. (9) and Table 3, the encoding of the NCT gates for implementing f can be described as follows:

- From Table 3, if $wt(b_0^i||b_1^i||b_2^i) = 1$ holds, one of the gates in the NCT gate set can be determined, which gives rise to the following restriction:

$$b_0^i + b_1^i + b_2^i = 1. \tag{10}$$

- Quantum gate is reversible and no ancilla qubits are introduced for the purpose of saving qubits in this work, which means that changes in state of the f input will be caused by each gate. Denote by $(x_0^{i-1}, x_1^{i-1}, \cdots, x_{n-1}^{i-1})$ the state of the f input before the application of the i-th gate. It follows that variables q_0^i, q_1^i and q_2^i are selected from $\{x_0^{i-1}, x_1^{i-1}, \cdots, x_{n-1}^{i-1}\}$, which can be restricted as follows:

$$\begin{cases} q_l^i = \sum_{j=0}^{n-1} a_{n\cdot l+j}^i \cdot x_j^{i-1}, \\ \sum_{j=0}^{n-1} a_{n\cdot l+j}^i = 1, \end{cases} \tag{11}$$

  where $l = 0, 1, 2$.

- The qubit can not be copied. It means that loading a qubit into multiple gates simultaneously is not allowed. As a result, q_0^i, q_1^i and q_2^i should be assigned to different inputs of f, which can be restricted by

$$a_j^i + a_{n+j}^i + a_{2n+j}^i \leqslant 1, \tag{12}$$

where $j \in [0, n-1]$ and n is the number of inputs of f.

- To eliminate symmetry, we restrict that only q_0^i will be changed in Eq. (9), while q_1^i and q_2^i are not. In other words, the update of q_0^i and its relationship with t^i should be encoded to maintain the *in-place* property of the quantum gate. According to Eq. (11), the variable q_0^i in Eq. (9) is restricted as $q_0^i = \sum_{j=0}^{n-1} a_j^i \cdot x_j^{i-1}$ with $\sum_{j=0}^{n-1} a_j^i = 1$, it follows that for any $j \in [0, n-1]$, if $a_j^i = 0$, the corresponding x_j^{i-1} will not be changed by the i-th gate. However, if $a_j^i = 1$, the corresponding input x_j^{i-1} should be updated to the output of the current gate. The above restriction can be described as follows:

$$x_j^i = a_j^i \cdot t^i + (1 + a^i) \cdot x_j^{i-1}, \tag{13}$$

where $j \in [0, n-1]$ and n is the number of inputs of f.

Encoding the Circuit By taking the output of the i-th gate, i.e., $(x_0^i, x_1^i, \cdots, x_{n-1}^i)$, as the input of the next gate, multiple gates can be encoded. Denote by $(x_0^k, x_1^k, \cdots, x_{n-1}^k)$ the output of the k-th gate. If each bit in $\{x_0^k, x_1^k, \cdots, x_{n-1}^k\}$ equals to a different output bit of the given vectorial Boolean function (denoted by $(y_0, y_1, \cdots, y_{n-1})$), then an NCT-based circuit of the vectorial Boolean function is constructed. Totally, there are $n!$ possible permutations map $(x_0^k, x_1^k, \cdots, x_{n-1}^k)$ to $(y_0, y_1, \cdots, y_{n-1})$. The relationship between $(x_0^k, x_1^k, \cdots, x_{n-1}^k)$ and $(y_0, y_1, \cdots, y_{n-1})$ can be expressed as

$$y_j = \sum_{i=0}^{n-1} m_{j\_i} \cdot x_i^k, \tag{14}$$

where $j \in [0, n-1]$. Notably, for $i, j \in [0, n-1]$, m_{j_i}s' in Eq. (14) form a permutation matrix.

The following 3-bit S-box is given as an example to illustrate the complete procedure of encoding the NCT-based circuit for a given vectorial Boolean function.

Example 2. *Denote the input and the output of the given 3-bit S-box as* (x_0, x_1, x_2) *and* (y_0, y_1, y_2) *respectively. The encoding of the first gate and the corresponding constraints are listed as follows:*

To encode the inputs of the current gate, the following constraints are given:

$$\begin{cases} q_0^1 = a_0^1 \cdot x_0 + a_1^1 \cdot x_1 + a_2^1 \cdot x_2, \\ q_1^1 = a_3^1 \cdot x_0 + a_4^1 \cdot x_1 + a_5^1 \cdot x_2, \\ q_2^1 = a_6^1 \cdot x_0 + a_7^1 \cdot x_1 + a_8^1 \cdot x_2, \\ a_0^1 + a_1^1 + a_2^1 = 1, \\ a_3^1 + a_4^1 + a_5^1 = 1, \\ a_6^1 + a_7^1 + a_8^1 = 1, \\ a_0^1 + a_3^1 + a_6^1 \leq 1, \\ a_1^1 + a_4^1 + a_7^1 \leq 1, \\ a_2^1 + a_5^1 + a_8^1 \leq 1. \end{cases} \tag{15}$$

The output of the current gate is encoded as

$$\begin{cases} t^1 = q_0^1 + b_0^1 + b_1^1 \cdot q_1^1 + b_2^1 \cdot q_1^1 \cdot q_2^1, \\ b_0^1 + b_1^1 + b_2^1 = 1. \end{cases} \tag{16}$$

After the current gate, the S-box inputs are updated as

$$\begin{cases} x_0^1 = a_0^1 \cdot t^1 + (1 + a_0^1) \cdot x_0, \\ x_1^1 = a_1^1 \cdot t^1 + (1 + a_1^1) \cdot x_1, \\ x_2^1 = a_2^1 \cdot t^1 + (1 + a_2^1) \cdot x_2. \end{cases} \tag{17}$$

The above steps encode the first gate, after which the S-box input is transformed from (x_0, x_1, x_2) *to* (x_0^1, x_1^1, x_2^1) *and will be regarded as the input of the next gate. After the application of* k *gates (the encoding of each is similar to that of the first gate shown with Eqs.* (15) - (17)*), the S-box input is transformed to* (x_0^k, x_1^k, x_2^k)*, corresponding to the S-box output, i.e.,* (y_0, y_1, y_2)*. Thus, we have*

$$\begin{cases} y_0 = m_{0\_0} \cdot x_0^k + m_{0\_1} \cdot x_1^k + m_{0\_2} \cdot x_2^k, \\ y_1 = m_{1\_0} \cdot x_0^k + m_{1\_1} \cdot x_1^k + m_{1\_2} \cdot x_2^k, \\ y_2 = m_{2\_0} \cdot x_0^k + m_{2\_1} \cdot x_1^k + m_{2\_2} \cdot x_2^k, \\ m_{0\_0} + m_{0\_1} + m_{0\_2} = 1, \\ m_{1\_0} + m_{1\_1} + m_{1\_2} = 1, \\ m_{2\_0} + m_{2\_1} + m_{2\_2} = 1, \\ m_{0\_0} + m_{1\_0} + m_{2\_0} = 1, \\ m_{0\_1} + m_{1\_1} + m_{2\_1} = 1, \\ m_{0\_2} + m_{1\_2} + m_{2\_2} = 1. \end{cases} \tag{18}$$

3.2 New STP Models

With the encoding of the underlying logic gates, as a direct application, it is possible to apply the SAT solver to search for an optimal implementation of an S-box with the criterion gate count. To this end, Algorithm 1 is introduced based on Sect. 3.1. Note that all values in the algorithm are determined by the S-box look-up table.

Algorithm 1 is designed to generate STP models for small S-boxes, then we can apply the SAT solver to solve the models. Note that we only present in Algorithm 1 the ANF representation for simplicity. In practical application, it should be converted into the CVC language, which is the input of the solver.

Converting ANF to CVC Firstly, use BITVECTOR(1) to represent a binary variable. Then, the operations XOR, ADD, and AND should be substituted with BVXOR, BVPLUS, and BVMULT, respectively. Besides, the concatenation is represented by the symbol @. To restrict variable x to be a constant, apply the constraint ASSERT($x = 0bin0$) or ASSERT($x = 0bin1$). In addition, to restrict that variable a is less than or equal to b, BVLE(a, b) is applied.

Models for Other Criteria To construct models for a specific criterion, the calculation of which should be explained first. In Algorithm 1, parameter K is taken as the input to restrict the number of allocated gates, and thus the minimal K such that the

Algorithm 1 Construct NCT-based Circuits for the S-box

Require: Look-up table of the S-box S, the number of gates K.
Ensure: **STPModel**(S, K).

for $s = 0$ to $2^n - 1$ **do**
  for $i = 0$ to $n - 1$ **do**
    $x^0_{s_i} = x_{s_i}$; ▷ Initialize the inputs
  end for
end for
for $s = 0$ to $2^n - 1$ **do**
  for $k = 1$ to K **do**
    for $l = 0$ to 2 **do**
      $q^k_{s_l} = \sum_{i=0}^{n-1} a^k_{n\cdot l+i} \cdot x^k_{s_i}$;
    end for
    $t^k_s = q^k_{s_0} + b^k_0 + b^k_1 \cdot q^k_{s_1} + b^k_2 \cdot q^k_{s_1} \cdot q^k_{s_2}$; ▷ The k-th logic gate
    for $i = 0$ to $n - 1$ **do**
      $x^k_{s_i} = a^k_i \cdot t^k_s + (1 + a^k_i) \cdot x^{k-1}_{s_i}$; ▷ Update the inputs of the S-box
    end for
  end for
  for $j = 0$ to $n - 1$ **do**
    $y_{s_j} = \sum_{i=0}^{n-1} m_{j\_i} \cdot x^k_{s_i}$; ▷ Encode the S-box outputs
  end for
end for
for $k = 1$ to K **do**
  $\sum_{i=0}^{n-1} a^k_{n\cdot l+i} = 1$;
  $\sum_{l=0}^{2} a^k_{n\cdot l} \leqslant 1$;
  $b^k_0 + b^k_1 + b^k_2 = 1$;
end for

SAT solver returns a solution successfully is the fewest gates required for implementing the S-box. The criteria shown in Sect. 2.4 can be calculated as follows:

- Two-qubit Cost: In an NCT-based circuit, only the CNOT gate consumption and the Toffoli gate consumption influence the value of its two-qubit cost. From Table 3 and Eq. (10), among b^i_0, b^i_1 and b^i_2, if the current gate is a CNOT gate, only the value of b^i_1 equals 1. For the case that the current gate is a Toffoli gate, only the value of b^i_2 is equal to 1. Therefore, for an NCT-based circuit with K gates, its two-qubit cost, denoted by T, can be calculated as:

$$T = \sum_{i=1}^{K} (b^i_1 + 5 \times b^i_2). \tag{19}$$

- Quantum Cost: Similar to the calculation of the two-qubit cost of an NCT-based circuit with K gates, denote by Q the quantum cost of the circuit. Then, Q can be

calculated as:

$$Q = \sum_{i=1}^{K} (b_0^i + b_1^i + 5 \times b_2^i). \tag{20}$$

- Toffoli Gate Consumption and Toffoli-Depth: Applying 2 Toffoli gates in parallel requires an NCT-based circuit to have at least 6 qubits, as the qubit cannot be loaded into multiple gates simultaneously. Therefore, for an n-bit circuit with $n \in [3, 5]$, the Toffoli-depth and the Toffoli gate consumption of the circuit are equal. From Eq. (9), the Toffoli gate is marked by b_2^i. Denote by D the Toffoli gate consumption of an NCT-based circuit with K gates. Then, D can be calculated as:

$$D = \sum_{i=1}^{K} b_2^i. \tag{21}$$

Based on Algorithm 1, the models for designing circuits with criteria two-qubit cost, quantum cost, and the Toffoli gate consumption (as well as the Toffoli-Depth of a small-scale function) can be constructed by adding Eq. (19), Eq. (20), and Eq. (21) in the end of Algorithm 1, respectively, where T, Q, and D are new parameters taken as inputs of the algorithm.

4 Further Insights on Constructing Implementations

In this section, constructing optimized implementations for S-boxes based on the SAT solver is further investigated by taking their cryptographic properties into consideration.

4.1 Optimization based on Affine Equivalence

Applying an invertible affine transformation before and after an S-box does not change the resistance of the S-box against most attacks [43]. In classical application scenarios, if an optimized implementation can be designed for any of the S-boxes affine equivalent to a specific one adopted in a cryptographic cipher, one can replace the specific S-box with the affine equivalent one to improve the implementation performance of the cipher without changing its security. This inspires us to further explore the construction of S-boxes with better implementations for quantum application scenarios based on affine equivalence.

Model of Affine Equivalence To find all the S-boxes affine equivalent to the bijective S-box $S_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$, the following issues should be considered:

- The definition of affine equivalence should be satisfied. If the given S-box and the candidate one (denoted by S_2) are affine equivalent, Eq. (5) holds.
- When constructing affine equivalent S-boxes, the transformations A and B searched for Eq. (5) should be linear and be encoded. Specifically, take A as an example, if A is linear, it follows that

$$A(x_1) \oplus A(x_2) = A(x_1 \oplus x_2), \tag{22}$$

where $x_1, x_2 \in \mathbb{F}_2^n$ are inputs of A.

- Both the affine equivalent S-boxes S_1 and S_2 should be bijective, which means that, for any $i, j \in \mathbb{F}_2^n$ and $i \neq j$, we have

$$S_2(i) \neq S_2(j). \tag{23}$$

Model for Implementation With the encoding methods of affine equivalence and the NCT-based circuit, by applying the SAT solver, it is possible to search for an optimal implementation of the S-boxes in the same affine equivalence class with the criterion of gate count. Assuming that the solver outputs a solution when the gate count is set to K, it means that there is an S-box in the affine equivalence class can be implemented with K gates. However, if the solver outputs unsatisfiable, it indicates that none of the S-boxes in the affine equivalence class can be implemented with K gates. At this point, the number of gates should be increased to $K + 1$. The details are illustrated in Algorithm 2.

Algorithm 2 Implement the S-boxes in the Same Affine Equivalence Class

Require: Look-up table of the S-box S_1, the number of gates K
Ensure: **STPModel**(S_1, K)

for $i = 0$ to $2^n - 2$ **do**
  for $j = i$ to $2^n - 1$ **do**
    $A[i \oplus j] = A[i] \oplus A[j]$; ▷ A and B are linear
    $B[i \oplus j] = B[i] \oplus B[j]$;
  end for
end for
for $i = 0$ *to* $2^n - 2$ **do**
  for $j = i$ *to* $2^n - 1$ **do**
    $A[i] \neq A[j]$;
    $B[i] \neq B[j]$;
    $S_2[i] \neq S_2[j]$; ▷ All mappings are bijective
  end for
end for
for $i = 0$ *to* $2^n - 1$ **do**
  $S_2[i] = B[S_1[A[i] \oplus a]] \oplus c$; ▷ S_2 is affine equivalent to S_1
end for
STPModel(S_2, K); ▷ Apply Algorithm 1

It should be noted that, in Algorithm 2, S_1 is known, and S_2 is undetermined. After solving the STP model, a corresponding set of solutions will be returned. For the case that both S_1 and S_2 are given, with minor modifications, the algorithm can also be used to determine whether the two S-boxes are in the same equivalence class.

Moreover, in order to identify the optimal implementation scheme of the S-boxes in the same affine equivalence class, the traditional methods require that one should list all the S-boxes in the same affine equivalence class and then construct an STP model for each S-box. By solving all the models, the optimal implementation can

be obtained. Due to the numerous affine equivalence of an S-box, the process can be time-consuming. However, by applying Algorithm 2, the affine equivalent S-boxes that satisfy the given conditions can be output directly, as well as their optimized implementations.

4.2 Opimization based on Good Cryptographic Properties

To resist differential and linear attacks, the fundamental cryptographic properties of differential uniformity and linearity are investigated. Additionally, the ability of a cipher to resist differential and linear attacks is influenced by the frequency of occurrence of differential uniformity and linearity, as well as the impact of the BIBO. Therefore, we consider these properties comprehensively. Moreover, previous researches show that the S-box implementation of PRØST [34] outperforms that of other ciphers significantly. The notable difference between the PRØST S-box and other ones lies in the fixed point. Therefore, the fixed point is also taken into consideration in this work.

The implementation of S-boxes based on the above properties can be broadly divided into two parts: one related to the DDT and the other related to the LAT. The calculation of DDT and that of LAT are different, while the calculation of frequency and the calculation of BIBO are the same. In the CVC language, it is necessary to define the number of bits for variables. Moreover, addition is involved both in calculations of BIBO, the frequency of DDT, and the frequency of LAT. If the maximum value that the addition of variables can achieve is ignored, overflow issues may arise. Similarly, subtraction in LAT may result in borrowing. Below, we provide a specific example to explain the computation results in STP.

Example 3. *Assuming that variables a, b, and c are defined as 4-bit variables with values $0x1$, $0xF$, and $0x2$, respectively. If there exist two variables $m = a + b$ and $n = a - c$, then m is equal to $0x0$, and n is equal to $0xF$ in the CVC language.*

Model for Properties Related to DDT For an $n \times n$ S-box, Lu *et al.* [24] defined DDT as a set of mappings from $\mathbb{F}_2^{2n}$ to $\mathbb{F}_2^n$. However, in this work, the approach in [24] would lead to overflow issues (if a term in the DDT of a 4-bit S-box is 16, then it would be 0 in actual computation), which would result in the actual differential uniformity exceeding the value we specified. To avoid this problem, we define the DDT as a set of mappings from $\mathbb{F}_2^{2n}$ to $\mathbb{F}_2^{n+1}$ and consider the following issues:

- The Calculation of Difference: According to Definition 1, each entry $\mathrm{DDT}[a, b]$, where $a, b \in \mathbb{F}_2^n$, is determined by the number of $x \in \mathbb{F}_2^n$ that satisfies $S(x) \oplus S(x \oplus a) = b$. To determine whether the current x has a difference, we introduce a flag variable, denoted as $\mathrm{IsTrue}[a, b, x]$. If the current x has a difference, $\mathrm{IsTrue}[a, b, x] = 1$, otherwise, $\mathrm{IsTrue}[a, b, x] = 0$. The sum of these flag variables is equal to $\mathrm{DDT}[a, b]$, i.e., $\mathrm{DDT}[a, b] = \mathrm{IsTrue}[a, b, 0] + \cdots + \mathrm{IsTrue}[a, b, 2^n - 1]$.
- The Achieving of Differential Uniformity: Constraints are added to ensure that all $\mathrm{DDT}[a, b]$ must be less than or equal to the given differential uniformity $\mathcal{U}$. Additionally, we introduce the constraint $\mathrm{DDT}[1, 1] = \mathcal{U}$ OR $\mathrm{DDT}[1, 2] =$

$\mathcal{U}$ OR $\cdots$ OR $\mathrm{DDT}[2^n-1, 2^n-1] = \mathcal{U}$ to ensure that the specified differential uniformity is met [1]. This constraint is not presented in the work of Lu *et al.* [24] as they did not independently consider the differential uniformity, but rather combined it with the frequency.

- The Calculation of Frequency: Similar to the calculation of DDT. For a given differential uniformity $\mathcal{U}$, the calculation of the frequency of $\mathcal{U}$ can be achieved by introducing a flag variable for each element $\mathrm{DDT}[a, b]$, denoted by $\mathrm{IsTrue}_{\mathrm{DDT}}[a, b]$, to mark if $\mathrm{DDT}[a, b] = \mathcal{U}$ holds. If $\mathrm{DDT}[a, b] = \mathcal{U}$, $\mathrm{IsTrue}_{\mathrm{DDT}}[a, b] = 1$, otherwise, $\mathrm{IsTrue}_{\mathrm{DDT}}[a, b] = 0$. The frequency of $\mathcal{U}$, denoted by FreqDDT, can be calculated as $\mathrm{FreqDDT} = \mathrm{IsTrue}_{\mathrm{DDT}}[1, 0] + \cdots + \mathrm{IsTrue}_{\mathrm{DDT}}[2^n - 1, 2^n - 1]$.
- The Calculation of BIBO: The calculation of BIBO is similar to that of the frequency. However, not all entries in the DDT are required; it only needs to consider the entries with Hamming weight of 1, i.e., $\mathrm{DDT}[a, b]$ with $a, b \in \{1, 2, 4, \cdots, 2^{n-1}\}$.

Details are presented in Algorithm 3, where the input parameters are $\mathcal{U}, \mathcal{U}_{Freq}$, and $\mathrm{BIBO}_{\mathrm{DDT}}$, representing differential uniformity, the frequency of differential uniformity, and the number of BIBOs in DDT, respectively. We set $\mathcal{U}$ to be equal to a specific value, while $\mathcal{U}_{Freq}$ and $\mathrm{BIBO}_{\mathrm{DDT}}$ are set to be less than or equal to the specific values. It is also worth noting that by setting different restrictions on $\mathcal{U}_{Freq}$ and $\mathrm{BIBO}_{\mathrm{DDT}}$, S-boxes with various frequencies and BIBO counts for different levels of differential uniformity can be found, which also means that the parameters $\mathcal{U}_{Freq}$ and $\mathrm{BIBO}_{\mathrm{DDT}}$ are optional for Algorithm 3.

Model for Properties Related to LAT The procedure of constructing a model for properties related to LAT is similar to Algorithm 3. Therefore, the details are presented in Appendix A with Algorithm 5.

Model for Implementation With the encoding methods of the S-box property (Algorithms 3 and 5) and the NCT-based circuit (Algorithm 1), combination models can be constructed directly for designing S-boxes and their implementations under specific conditions. To this end, additional constraints need to be added. Firstly, it is essential to ensure that the S-boxes are bijective. Secondly, there is an optional constraint for fixed points[2]. Since the properties are directly calculated from the S-box, and the NCT-based circuit implementation scheme is represented based on the input and output bits of the S-box, it is necessary to combine them together. For all $i \in \mathbb{F}_2^n$, the input i of the S-box corresponds to bits $i = x^0_{i_{n-1}} @ \cdots @ x^0_{i_1} @ x^0_{i_0}$, and the output $S[i]$ corresponds to bit $S[i] = y_{i_{n-1}} @ \cdots @ y_{i_1} @ y_{i_0}$. It implies that the S-box can be implemented by the current NCT-based circuit. The details are presented in Algorithm 4, where constraints on frequencies and BIBO patterns are optional.

Using the SAT solver, solutions that satisfy all the constraint conditions can be output. To obtain all S-boxes that meet the conditions, when STP outputs an S-box, it can be removed from the solution space, and this process can be repeated until the solver outputs no solution.

[1] Without this constraint, when specifying a differential uniformity of 6, the output S-box's differential uniformity might be 4.

[2] It only indicates the absence of fixed point restrictions in the program, implying that the output S-box may or may not have fixed points

Algorithm 3 Properties Related to DDT

Require: Differential uniformity $\mathcal{U}$, frequency of the differential uniformity $\mathcal{U}_{Freq}$, the number of BIBO in DDT $\mathrm{BIBO}_{\mathrm{DDT}}$

Ensure: **STPModel**($\mathcal{U}$, $\mathcal{U}_{Freq}$, $\mathrm{BIBO}_{\mathrm{DDT}}$)

for $a = 0$ to $2^n - 1$ **do**
  for $b = 0$ to $2^n - 1$ **do**
    for $x = 0$ to $2^n - 1$ **do**
      if $S[x \oplus a] = S[x] \oplus b$ **then**
        $\mathrm{IsTrue}[a, b, x] = 1$;
      else
        $\mathrm{IsTrue}[a, b, x] = 0$;
      end if
    end for
    $\mathrm{DDT}[a, b] = \mathrm{IsTrue}[a, b, 0] + \cdots + \mathrm{IsTrue}[a, b, 2^n - 1]$; ▷ Definition of DDT
    $\mathrm{DDT}[a, b] \leq \mathcal{U}$; ▷ Definition of differential uniformity
    if $\mathrm{DDT}[a, b] = \mathcal{U}$ **then**
      $\mathrm{IsTrue}_{\mathrm{DDT}}[a, b] = 1$;
    else
      $\mathrm{IsTrue}_{\mathrm{DDT}}[a, b] = 0$;
    end if
  end for
end for
$\mathrm{DDT}[1, 1] = \mathcal{U}\ OR \cdots OR\ \mathrm{DDT}[2^n - 1, 2^n - 1] = \mathcal{U}$; ▷ Ensure $\mathcal{U}$ is satisfied
$\mathrm{FreqDDT} = \mathrm{IsTrue}_{\mathrm{DDT}}[1, 0] + \mathrm{IsTrue}_{\mathrm{DDT}}[1, 1] + \cdots + \mathrm{IsTrue}_{\mathrm{DDT}}[2^{n-1}, 2^{n-1}]$;
$\mathrm{FreqDDT} \leq \mathcal{U}_{Freq}$;
$Z = [1, 2, 4, \cdots, 2^{n-1}]$; ▷ The set of elements in $[0, 2^{n-1}]$ with Hamming weight 1
for $a \in Z$ **do**
  for $b \in Z$ **do**
    if $\mathrm{DDT}[a, b] \neq 0$ **then**
      $\mathrm{BIBO}[a, b] = 1$;
    else
      $\mathrm{BIBO}[a, b] = 0$;
    end if
  end for
end for
$\mathrm{BIBODDT} = \mathrm{BIBO}[1, 1] + \mathrm{BIBO}[1, 2] + \cdots + \mathrm{BIBO}[2^{n-1}, 2^{n-1}]$; ▷ Def. of BIBO
$\mathrm{BIBODDT} \leq \mathrm{BIBO}_{\mathrm{DDT}}$;

5 Applications

This section presents the three applications of the proposed models: the first focuses on constructing optimal implementation schemes for S-boxes based on various criteria; the second is to investigate further optimization based on affine equivalence; the last combines the implementation and some typical cryptographic properties of S-boxes.

Algorithm 4 Combine the S-box Circuit and Cryptographic Properties

Require: The number of gates K, differential uniformity $\mathcal{U}$, linearity $\mathcal{L}$, frequency of $\mathcal{U}/\mathcal{L}$ $\mathcal{U}_{Freq}/\mathcal{L}_{Freq}$, the number of BIBO in DDT/LAT $\text{BIBO}_{\text{DDT}}/\text{BIBO}_{\text{LAT}}$

Ensure: **STPModel**(K,$\mathcal{U}$, $\mathcal{L}$,$\mathcal{U}_{Freq}$, $\mathcal{L}_{Freq}$, BIBO_{DDT}, BIBO_{LAT})

for $i = 0$ to $2^n - 1$ **do**
  $S[i] \neq i$; ▷ The S-box S has no fixed points (optional)
end for
for $i = 0$ to $2^n - 2$ **do**
  for $j = i$ to $2^n - 1$ **do**
    $S[j] \neq S[i]$; ▷ The S-box S is a bijection
  end for
end for
STPModel($\mathcal{U}$, $\mathcal{U}_{Freq}$, BIBO_{DDT}); ▷ Apply Algorithm 3
STPModel($\mathcal{L}$, $\mathcal{L}_{Freq}$, BIBO_{LAT}); ▷ Apply Algorithm 5
for $i = 0$ *to* $2^n - 1$ **do**
  $i = x^0_{i_{n-1}} @ \cdots @ x^0_{i_1} @ x^0_{i_0}$;
  $S[i] = y_{i_{n-1}} @ \cdots @ y_{i_1} @ y_{i_0}$;
end for
STPModel(S, K); ▷ Apply Algorithm 1

5.1 Optimizing the S-box Implementation with Various Criteria

To construct optimized NCT-based circuits for small S-boxes, Dasu *et al.* [18] assigned the NCT gates with different cost metrics of gate count, two-qubit cost and quantum cost, which also be adopted as criteria in our models in this section.

Application to 4-bit S-boxes Based on Algorithm 1, Eqs. (19) and (20), we construct models to search for optimal implementation schemes for the 4-bit S-boxes adopted in some well-known cryptographic ciphers [34–40], the look-up tables of which are listed in Appendix B. To explain the transformation from the output of the SAT solver to an NCT-based circuit, the S-box of `GIFT` [40] is taken as the input of Algorithm 1 and the following example is given.

Example 4. *Denote by* (x_0, x_1, x_2, x_3) *and* (y_0, y_1, y_2, y_3) *the input and the output of the* `GIFT` *S-box, when the number of logic gates is set to 8, the solver outputs the following solution:*

$$
\begin{array}{llll}
0\ \mathtt{x}^0_0 = \mathtt{x}_0, & 4\ \mathtt{x}^1_1 = \mathtt{x}^0_1 + \mathtt{x}^0_0 \cdot \mathtt{x}^0_2, & 8\ \mathtt{x}^5_1 = \mathtt{x}^4_1 + 1, & 12\ \mathtt{y}_0 = \mathtt{x}^8_3, \\
1\ \mathtt{x}^0_1 = \mathtt{x}_1, & 5\ \mathtt{x}^2_0 = \mathtt{x}^1_0 + \mathtt{x}^1_1 \cdot \mathtt{x}^1_3, & 9\ \mathtt{x}^6_3 = \mathtt{x}^5_3 + \mathtt{x}^5_0 \cdot \mathtt{x}^5_1, & 13\ \mathtt{y}_1 = \mathtt{x}^8_1, \\
2\ \mathtt{x}^0_2 = \mathtt{x}_2, & 6\ \mathtt{x}^3_3 = \mathtt{x}^2_3 + \mathtt{x}^2_2, & 10\ \mathtt{x}^7_1 = \mathtt{x}^6_1 + \mathtt{x}^6_3, & 14\ \mathtt{y}_2 = \mathtt{x}^8_0, \\
3\ \mathtt{x}^0_3 = \mathtt{x}_3, & 7\ \mathtt{x}^4_2 = \mathtt{x}^3_2 + \mathtt{x}^3_1, & 11\ \mathtt{x}^8_2 = \mathtt{x}^7_2 + \mathtt{x}^7_0 \cdot \mathtt{x}^7_1, & 15\ \mathtt{y}_3 = \mathtt{x}^8_2.
\end{array}
$$

The quantum implementation corresponds to the above operations is shown in Fig. 3.

The experimental results reveal that optimal solutions can be obtained for all tested 4-bit S-boxes. The details are shown in Table 4, where `GC`, `2qC` and `QC` represent the criterion gate count, two-qubit cost, and quantum cost, respectively. Additionally,

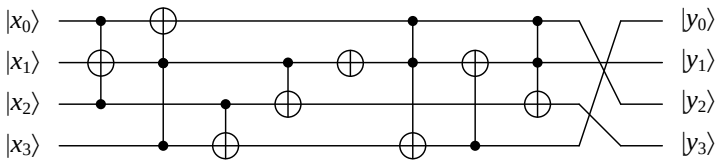

Fig. 3 NCT-based circuit of the GIFT S-box.

Table 4 also provides cryptographic properties of each S-box. Notably, apart from the GIFT S-box, all S-boxes share the same properties listed in the table except BIBO.

Table 4 Implementations of the 4-bit S-boxes

S-box	$\mathcal{U}$	$\mathcal{L}$	$\mathcal{U}_{Freq}$	$\mathcal{L}_{Freq}$	BIBO_{DDT}	BIBO_{LAT}	GC	2qC	QC
PRØST	4	8	24	36	5	8	4	20	20
PICCOLO	4	8	24	36	4	7	9	20	25
SKINNY	4	8	24	36	4	7	10	20	26
PRESENT	4	8	24	36	0	8	11	25	27
RECTANGLE	4	8	24	36	2	2	10	23	26
LAC	4	8	24	36	4	6	8	22	24
GIFT	6	8	2	36	1	3	8	22	24

Moreover, we apply the tool LIGHTER-R to the S-boxes listed in Table 4, and the results reveal that, for each criterion, implementations of those S-boxes returned by LIGHTER-R are also optimal.

In general, the search for the optimal solution based on the SAT solver is considered to be time-consuming. However, in our experiments, the search procedures of all the models constructed based on various criteria consume only a few seconds for most S-boxes (less than 0.1s for the PRØST S-box).

Application to 5-bit S-boxes Constructing an optimal implementation of a 5-bit S-box is often considered tricky due to the high complexity. For this reason, the tool LIGHTER-R can only be used to construct NCT-based circuits for small-scale S-boxes, such as the 3-/4-bit ones. Although the solver may fail to output any valid solutions within a reasonable time for a 5-bit S-box, with an appropriate value of the criterion assigned to the model, the solver responds quickly. As a consequence, Algorithm 1, as well as that combined with Eqs. (19) - (21), can be used to provide a lower bound of a certain criterion for implementing a 5-bit S-box. In this case, we take the criterion gate count as an example and take the 5-bit S-boxes of ASCON [41] and KECCAK [42] as the inputs of the models, the results are shown in Table 5, where GC represents the gate count.

Table 5 Implementations of the 5-bit S-boxes

S-box	$\mathcal{U}$	$\mathcal{L}$	$\mathcal{U}_{Freq}$	$\mathcal{L}_{Freq}$	BIBO_{DDT}	BIBO_{LAT}	GC
ASCON	8	16	20	40	0	0	>12
KECCAK	8	16	20	40	5	5	>9

5.2 Constructing S-box Implementations based on Affine Equivalence

Both the look-up table of an S-box and a representation of an affine equivalence class can be taken as the input of Algorithm 2, in which Algorithm 1 is embedded. Recall that the models for criteria gate count, two-qubit cost, and quantum cost can all be constructed based on Algorithm 1. In this section, the criterion gate count will be taken as an example to exhibit the performance of the proposed models. We initialize the number of allocated gates K to 1 and construct models. If the SAT solver does not output a solution for a long time, we increase the value of K by 1 and repeat the procedure until the solver outputs a solution.

Taking S-boxes as Inputs As did in Sect. 5.1, we take the S-boxes adopted in [34–40] as the inputs of the models. The results are listed in Table 6, where GC represents the criterion gate count. Notably, no restrictions of the fixed point are imposed on the models for Table 6. For the case that the fixed point is not allowed, the results are listed in Table 7.

Table 6 Results based on affine equivalence

S-box	GC	S-box in the affine equivalence class	GC
PRØST	4	the PRØST S-box	4⋆
PICCOLO	9	0, 8, 4, 14, 1, 15, 5, 9, 2, 10, 7, 11, 13, 3, 12, 6	4⋆
SKINNY	10	0, 8, 1, 7, 2, 13, 3, 10, 4, 12, 15, 9, 14, 11, 5, 6	4⋆
PRESENT	11	0, 1, 8, 7, 4, 13, 12, 11, 2, 3, 9, 14, 10, 5, 15, 6	5†
RECTANGLE	10	0, 8, 2, 14, 1, 15, 9, 3, 12, 7, 10, 5, 13, 4, 6, 11	5†
LAC	8	0, 1, 8, 9, 4, 13, 14, 11, 2, 15, 7, 10, 6, 3, 5, 12	4⋆
GIFT	8	0, 8, 1, 15, 2, 14, 3, 9, 4, 13, 10, 5, 7, 12, 11, 6	5†

⋆ Optimal.
† No solution is returned in 1 day if the gate count is reduced by 1.

Table 7 Results based on affine equivalence with no fixed point is allowed

S-box	GC	S-box in the affine equivalence class	GC
PRØST	4	2, 0, 15, 8, 3, 1, 10, 13, 14, 4, 5, 12, 9, 7, 6, 11	5†
PICCOLO	9	8, 12, 7, 2, 0, 4, 10, 13, 11, 3, 6, 15, 1, 14, 9, 5	5⋆
SKINNY	10	2, 7, 6, 13, 15, 10, 12, 3, 0, 1, 4, 14, 8, 9, 11, 5	5†
PRESENT	11	8, 12, 7, 2, 0, 4, 10, 13, 11, 3, 6, 15, 1, 14, 9, 5	6†
RECTANGLE	10	4, 0, 5, 8, 6, 3, 10, 14, 13, 1, 11, 7, 15, 2, 12, 9	6†
LAC	8	8, 10, 0, 13, 9, 11, 7, 2, 12, 6, 4, 3, 15, 5, 1, 14	5†
GIFT	8	2, 3, 4, 9, 0, 1, 7, 14, 10, 15, 12, 5, 11, 6, 8, 13	6†

⋆ Optimal.
† No solution is returned in 1 day if the gate count is reduced by 1.

From Table 6, without the restrictions on the fixed point, there is always an S-box in the class that can be implemented with only 4 or 5 gates. Compared to the optimal

circuits designed for the given S-boxes (listed in the second column of Table 6), except that for the PRØST S-box, the gate consumption of each new one is reduced by half or more. For the case that no fixed point is allowed, it can be observed from Table 7 that one more gate is required by each S-box.

Taking Affine Equivalence Classes as Inputs For 4-bit S-boxes, there are 302 affine equivalence classes. As pointed out in [43], 16 classes are optimal (listed in Table C1 in Appendix C). Among those classes, $G_0, G_1, G_2, G_4, G_5, G_7, G_8$ and G_{13} are even, that is, they can be implemented without introducing additional ancilla qubits. The S-boxes in PRØST, PICCOLO, SKINNY, and LAC are in G_8. From Table 6, the implementation of the PRØST S-box is optimal in the class. As a result, there is no need to deal with the class G_8. Similarly, both the S-boxes of PRESENT and RECTANGLE are in G_1, which can also be ignored, as one can deduce from Table 6 that the implementation returned by our model consumes at least 5 gates for the S-boxes in G_1. The optimal implementations with criterion gate count of the remaining classes returned by our models are shown in Table 8. Besides, we also construct models for the case that the fixed point is not allowed, the results are listed in Table 9.

Table 8 Results based on affine equivalence classes

S-box	representation	GC
G_0	0, 4, 2, 6, 1, 15, 7, 12, 8, 14, 13, 11, 9, 5, 10, 3	4*
G_1	0, 8, 2, 14, 1, 15, 9, 3, 12, 7, 10, 5, 13, 4, 6, 11	5†
G_2	0, 1, 8, 14, 4, 7, 12, 9, 2, 3, 11, 5, 15, 13, 6, 10	4*
G_4	0, 9, 2, 10, 1, 8, 7, 6, 4, 15, 14, 5, 13, 11, 12, 3	7†
G_5	0, 4, 2, 15, 1, 11, 9, 14, 8, 6, 10, 5, 13, 12, 7, 3	7†
G_7	0, 13, 1, 9, 2, 10, 7, 12, 4, 8, 15, 14, 11, 6, 5, 3	7†
G_8	the PRØST S-box	4*
G_{13}	0, 4, 2, 10, 8, 12, 13, 7, 1, 15, 3, 5, 11, 14, 6, 9	7†

$\star$ Optimal.
† No solution is returned in 1 day if the gate count is reduced by 1.

The results listed in Tables 8 - 9 show that NCT-based circuits of the S-boxes in $G_0, G_1, G_2, G_4, G_5, G_7, G_8, G_{13}$ consume at least 4 gates if no restrictions are imposed on the fixed point. However, with the restriction that the fixed point is not allowed, one more gate is required by each S-box.

5.3 Combining Implementations and Cryptographic Properties of S-boxes

Similar to Sect. 5.2, the criterion gate count will be taken as an example in this section to exhibit the performance of the proposed models.

Application to 4-bit S-boxes The optimal differential uniformity and linearity of 4-bit S-boxes is 4 and 8 respectively [43]. Therefore, we set $\mathcal{U} = 4$ and $\mathcal{L} = 8$ as the inputs of Algorithm 4 and apply the model to search for S-boxes with fixed points. In this case, the solver returns 3456 solutions until $K = 4$. To construct more S-boxes

Table 9 Results based on affine equivalence classes with no fixed point is allowed

S-box	representation	GC
G_0	8, 11, 12, 5, 15, 6, 10, 9, 0, 1, 4, 14, 2, 3, 7, 13	$5^\star$
G_1	8, 12, 7, 2, 0, 4, 10, 13, 11, 3, 6, 15, 1, 14, 9, 5	$6^\dagger$
G_2	4, 0, 5, 11, 6, 3, 7, 10, 12, 8, 15, 1, 9, 14, 2, 13	$5^\star$
G_4	1, 9, 15, 10, 5, 12, 11, 13, 0, 8, 2, 14, 7, 6, 3, 4	$7^\dagger$
G_5	2, 10, 0, 5, 3, 13, 1, 14, 6, 12, 11, 8, 4, 9, 15, 7	$8^\dagger$
G_7	4, 6, 15, 14, 5, 9, 10, 13, 0, 3, 8, 12, 11, 1, 2, 7	$8^\dagger$
G_8	2, 0, 15, 8, 3, 1, 10, 13, 14, 4, 5, 12, 9, 7, 6, 11	$5^\dagger$
G_{13}	1, 0, 9, 11, 7, 8, 3, 12, 13, 4, 5, 14, 6, 15, 10, 2	$8^\dagger$

$\star$ Optimal.
† No solution is returned in 1 day if the gate count is reduced by 1.

with various cryptographic properties, we then construct three kinds of models: the first is constructed by fixing the differential uniformity, linearity and frequency as that of the GIFT S-box; the second is constructed as did in [14] by restricting that $\mathcal{U} = 4$, $\mathcal{L} = 8, \mathcal{U}_{Freq} \leq 15$ and $\mathcal{L}_{Freq} \leq 30$; the last takes all the properties into consideration. Models for the case that the fixed point is not allowed can be constructed similarly. The application results of the models are listed in Table 10[3], where '-' indicates no constraints on the current properties.

Table 10 Application to 4-bit S-boxes

$\mathcal{U}$	$\mathcal{L}$	$\mathcal{U}_{Freq}$	$\mathcal{L}_{Freq}$	BIBO_{DDT}	BIBO_{LAT}	GC	S-box
With Fixed Points							
4	8	-	-	-	-	$4^\star$	{3456}
6	8	2	36	-	-	$5^\star$	√
4	8	15	30	-	-	$6^\dagger$	√
4	8	24	36	4	6	$4^\star$	√
Without Fixed Points							
4	8	-	-	-	-	$5^\star$	√
4	8	15	30	-	-	$7^\star$	√
4	8	24	36	4	4	$7^\dagger$	√

$\star$ Optimal.
† No solution is returned in 1 day if the gate count is reduced by 1.

With the restriction that the fixed point is allowed, all the solutions can be returned in a reasonable time, which gives rise to the following observation.

Observation 1. *All the 3456 optimal S-boxes with fixed points can be implemented with only 4 NCT gates. Besides, those S-boxes have at least one fixed point* $S[0] = 0$.

Note that when constructing the above models, we only take the total number of allocated gates (i.e., K) as the input of the algorithms, no additional constraints

[3] Appendix D also presents some S-boxes obtained by assigning various values to different properties.

are placed on gates. In fact, we can also limit the Toffoli gate count allocated for the models (by Eq. (21)), from which the Toffoli-Depth of the circuit can also be derived. Combined with our experiments with restriction on the Toffoli gate consumption and the results listed in Table 10, the following observation is presented.

Observation 2. *The NCT-based circuits of the optimal 4-bit S-boxes with/without fixed points consume at least 4 Toffoli gates, which also indicates that, unless introducing ancilla qubits, one cannot implement a 4-bit optimal S-box within a Toffoli-Depth of 3.*

Application to 5-bit S-boxes Although the solver cannot find optimal solutions for existing 5-bit S-boxes in a reasonable time. However, using the proposed models, it is possible to construct optimized implementations of S-boxes that are equivalent or superior to existing ones in cryptographic properties. As a consequence, similar to the construction of the models for 4-bit S-boxes, we construct different models for 5-bit S-boxes by fixing various properties. Note that when considering all the properties, we set the cryptographic properties input to the models the same as that of the KECCAK S-box. The application results of those models are listed in Table 11[4].

Table 11 Application to 5-bit S-boxes

$\mathcal{U}$	$\mathcal{L}$	$\mathcal{U}_{Freq}$	$\mathcal{L}_{Freq}$	BIBO_{DDT}	BIBO_{LAT}	GC	**S-box**
With Fixed Points							
8	16	-	-	-	-	5	√
2	8	-	-	-	-	[5,9]	×
8	16	10	30	-	-	6	√
8	16	20	40	5	5	5	√
Without Fixed Points							
8	16	-	-	-	-	6	√
8	16	10	30	-	-	7	√
8	16	20	40	5	5	6	√

Compared with the result listed in Table 5 that the gate consumption of the KECCAK S-box implementation is more than 9, the application of our proposed models reveals that we can construct S-boxes with the same properties as the KECCAK S-box but consume only 5 gates.

6 Conclusion

In this work, the optimization of the NCT-based circuits for S-boxes based on the SAT solver is investigated. To this end, we proposed encoding methods for the NCT-based circuits and some important cryptographic properties of S-boxes. Based on the proposed methods, we constructed various STP models to identify the optimal implementation schemes of S-boxes with different criteria, to further investigate optimizing the S-box implementations based on affine equivalence, and to search for S-boxes that match the given restrictions on the implementation and properties. Since the inputs of

[4]Appendix D also presents some S-boxes obtained by assigning various values to different properties.

the models proposed in this work are tweakable, apart from applications of the models to search for S-boxes that match given criterion configurations, one can also apply the models with restrictions placed on the number of each logic gate and the Toffoli-depth. It should be noted that the above functions are not currently available in the public tools for constructing optimized NCT-based circuits for S-boxes.

For the purpose of saving qubits, only the implementations of even permutations are considered in this work. If more ancilla qubits are allowed, NCT-based circuits for odd permutations can be designed. Besides, it is widely believed that applying ancilla qubits may help to optimize the performance of S-box circuits in criteria such as Depth. We leave the solution of the above problems based on the SAT solver for the case that more ancilla qubits are available as possible direction for future research.

Appendix A The Model for LAT-related Properties

The calculation of LAT is similar to that of DDT, with the difference that negative values may appear in LAT. However, the CVC language does not support the representation of negative numbers. Therefore, we manually take the absolute value of each entry in LAT, which is called ABLAT. From Definition 2, $\mathrm{LAT}[a,b] = \mathrm{LATT}[a,b] - 2^{n-1}$, where $a, b \in \mathbb{F}_2^n$ and $\mathrm{LATT}[a,b] = \#\{x \in \mathbb{F}_2^m | a \cdot x \oplus b \cdot S(x) = 0\}$. If $\mathrm{LATT}[a,b] \geq 2^{n-1}$, then $0 \leq \mathrm{LAT}[a,b] \leq 2^{n-1}$, and $\mathrm{ABLAT}[a,b] = \mathrm{LAT}[a,b]$. However, if $\mathrm{LATT}[a,b] \leq 2^{n-1}$, carry out borrowing will occur, as illustrated in Example 3. In this case, $\mathrm{LAT} = \mathrm{LATT}[a,b] + 2^n - 2^{n-1} = \mathrm{LATT}[a,b] + 2^{n-1}$, $2^{n-1} \leq \mathrm{LAT}[a,b] \leq 2^n$, the absolute value is given by $\mathrm{ABLAT}[a,b] = 2^n - \mathrm{LAT}[a,b]$. The calculation of frequency and the calculation of BIBO in LAT are similar to those in DDT, and we have omitted this process in Algorithm 5.

Appendix B Look-up Tables of the Related S-boxes

`PRESENT`: [12, 5, 6, 11, 9, 0, 10, 13, 3, 14, 15, 8, 4, 7, 1, 2]
`GIFT`: [1, 10, 4, 12, 6, 15, 3, 9, 2, 13, 11, 7, 5, 0, 8, 14]
`PICCOLO`: [14, 4, 11, 2, 3, 8, 0, 9, 1, 10, 7, 15, 6, 12, 5, 13]
`LAC`: [14, 9, 15, 0, 13, 4, 10, 11, 1, 2, 8, 3, 7, 6, 12, 5]
`PRØST`: [0, 4, 8, 15, 1, 5, 14, 9, 2, 7, 10, 12, 11, 13, 6, 3]
`RECTANGLE`: [6, 5, 12, 10, 1, 14, 7, 9, 11, 0, 3, 13, 8, 15, 4, 2]
`SKINNY`: [12, 6, 9, 0, 1, 10, 2, 11, 3, 8, 5, 13, 4, 14, 7, 15] 22, 12, 11, 19]
`ASCON`: [4, 11, 31, 20, 26, 21, 9, 2, 27, 5, 8, 18, 29, 3, 6, 28, 30, 19, 7, 14, 0, 13, 17, 24, 16, 12, 1, 25, 22, 10, 15, 23]
`KECCAK`: [0, 5, 10, 11, 20, 17, 22, 23, 9, 12, 3, 2, 13, 8, 15, 14, 18, 21, 24, 27, 6, 1, 4, 7, 26, 29, 16, 19, 30, 25, 28, 31]

Algorithm 5 Properties Related to LAT

Require: Linearity $\mathcal{L}$
Ensure: **STPModel**($\mathcal{L}$)

for $a = 0$ to $2^n - 1$ **do**
  for $b = 0$ to $2^n - 1$ **do**
    for $x = 0$ to $2^n - 1$ **do**
      if $a \cdot x = b \cdot S[x]$ **then**
        IsTrueLAT$[a, b, x] = 1$;
      else
        IsTrueLAT$[a, b, x] = 0$;
      end if
    end for
    LAT$[a, b] =$ LATT$[a, b] - 2^{n-1}$; ▷ Definition of LAT
    LATT$[a, b] =$ IsTrueLAT$[a, b, 0] + \cdots +$ IsTrueLAT$[a, b, 2^{n-1}]$;
    if LAT$[a, b] \geq 2^{n-1}$ **then**
      ABLAT$[a, b] = 2^n -$ LAT$[a, b]$;
    else
      ABLAT$[a, b] =$ LAT$[a, b]$; ▷ Definition of ABLAT
    end if
    ABLAT$[a, b] \leq \mathcal{L}$; ▷ Definition of linearity
  end for
end for

Table C1 Affine equivalence classes

Equivalence class	representative
G_0	0, 1, 2, 13, 4, 7, 15, 6, 8, 11, 12, 9, 3, 14, 10, 5
G_1	0, 1, 2, 13, 4, 7, 15, 6, 8, 11, 14, 3, 5, 9, 10, 12
G_2	0, 1, 2, 13, 4, 7, 15, 6, 8, 11, 14, 3, 10, 12, 5, 9
G_3	0, 1, 2, 13, 4, 7, 15, 6, 8, 12, 5, 3, 10, 14, 11, 9
G_4	0, 1, 2, 13, 4, 7, 15, 6, 8, 12, 9, 11, 10, 14, 5, 3
G_5	0, 1, 2, 13, 4, 7, 15, 6, 8, 12, 11, 9, 10, 14, 3, 5
G_6	0, 1, 2, 13, 4, 7, 15, 6, 8, 12, 11, 9, 10, 14, 5, 3
G_7	0, 1, 2, 13, 4, 7, 15, 6, 8, 12, 14, 11, 10, 9, 3, 5
G_8	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 9, 5, 10, 11, 3, 12
G_9	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 11, 3, 5, 9, 10, 12
G_{10}	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 11, 5, 10, 9, 3, 12
G_{11}	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 11, 10, 5, 9, 12, 3
G_{12}	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 11, 10, 9, 3, 12, 5
G_{13}	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 12, 9, 5, 11, 10, 3
G_{14}	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 12, 11, 3, 9, 5, 10
G_{15}	0, 1, 2, 13, 4, 7, 15, 6, 8, 14, 12, 11, 9, 3, 10, 5

Appendix C Affine Equivalence Classes with $\mathcal{U} = 4$ and $\mathcal{L} = 8$

Appendix D Results of the S-boxes with Various Restrictions on Properties

Table D2 S-boxes obtained by placing various restraints on properties

$\mathcal{U}$	$\mathcal{L}$	$\mathcal{U}_{Freq}$	$\mathcal{L}_{Freq}$	BIBO_{DDT}	BIBO_{LAT}	GC	**S-box**
4	8	24	36	4	6	4	S_1
4	8	15	30	9	12	6	S_2
6	8	2	36	8	8	5	S_3
4	8	24	36	4	7	5	S_4
4	8	24	36	4	4	7	S_5
4	8	15	30	9	12	7	S_6
8	16	20	40	5	5	5	S_7
8	16	7	28	8	10	6	S_8
8	16	20	40	5	5	6	S_9
8	16	8	23	8	14	7	S_{10}

S_1: [0, 8, 4, 14, 2, 10, 7, 13, 1, 15, 5, 9, 12, 3, 11, 6]
S_2: [0, 4, 1, 5, 8, 11, 3, 9, 2, 14, 15, 6, 10, 7, 12, 13]
S_3: [0, 8, 4, 7, 1, 11, 13, 12, 2, 6, 5, 10, 3, 15, 14, 9]
S_4: [1, 0, 9, 8, 3, 10, 12, 15, 5, 14, 6, 13, 7, 4, 11, 2]
S_5: [4, 7, 0, 9, 6, 12, 8, 10, 13, 2, 11, 14, 15, 1, 3, 5]
S_6: [4, 5, 0, 1, 12, 3, 10, 8, 6, 15, 7, 13, 14, 11, 9, 2]
S_7: [0, 8, 16, 24, 4, 12, 21, 29, 2, 31, 23, 10, 6, 26, 18, 14, 1, 22, 17, 11, 13, 27, 28, 7, 3, 9, 30, 25, 15, 5, 19, 20]
S_8: [0, 1, 16, 31, 4, 13, 20, 22, 8, 9, 10, 21, 12, 5, 14, 28, 2, 3, 18, 29, 15, 6, 19, 17, 26, 23, 24, 11, 27, 30, 25, 7]
S_9: [1, 5, 0, 6, 11, 15, 2, 4, 9, 13, 8, 26, 23, 19, 30, 12, 17, 22, 16, 21, 27, 20, 18, 31, 29, 14, 28, 25, 3, 24, 10, 7]
S_{10}: [2, 6, 10, 12, 7, 3, 30, 29, 0, 4, 8, 14, 1, 5, 25, 26, 27, 28, 19, 31, 13, 11, 21, 15, 16, 23, 24, 20, 17, 22, 9, 18]

References

[1] Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, New York (1993)

[2] Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993,

Proceedings. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer, Heidelberg (1993)

[3] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, pp. 212–219. ACM, New York (1996)

[4] Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997)

[5] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)

[6] Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (ed.) Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9606, pp. 29–43. Springer, Cham (2016)

[7] Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and lowmc. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 280–310. Springer, Cham (2020)

[8] Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of AES with fewer qubits. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12492, pp. 697–726. Springer, Cham (2020)

[9] Huang, Z., Sun, S.: Synthesizing quantum circuits of AES with lower t-depth and less qubits. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13793, pp. 614–644. Springer, Cham (2022)

[10] NIST: Submission rrequirements and evaluation criteria for the post-quantum cryptography standardization process (2016). https://csrc.nist.rip/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf

[11] NIST: Call for additional digital signature schemes for the post-quantum cryptography standardization process (2023). https://csrc.nist.gov/csrc/media/

Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf

[12] Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**(4), 656–715 (1949)

[13] Jean, J., Peyrin, T., Sim, S.M., Tourteaux, J.: Optimizing implementations of lightweight building blocks. IACR Trans. Symmetric Cryptol. **2017**(4), 130–168 (2017)

[14] Bao, Z., Guo, J., Ling, S., Sasaki, Y.: PEIGEN - a platform for evaluation, implementation, and generation of s-boxes. IACR Trans. Symmetric Cryptol. **2019**(1), 330–394 (2019)

[15] Li, Z., Gao, F., Qin, S., Wen, Q.: New record in the number of qubits for a quantum implementation of AES. IACR Cryptol. ePrint Arch., 18 (2023)

[16] Lin, D., Xiang, Z., Xu, R., Zhang, S., Zeng, X.: Optimized quantum implementation of AES. Quantum Inf. Process. **22**(9), 352 (2023)

[17] Lin, D., Xiang, Z., Xu, R., Zeng, X., Zhang, S.: Quantum circuit implementations of SM4 block cipher based on different gate sets. Quantum Inf. Process. **22**(7), 282 (2023)

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

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

[20] Stoffelen, K.: Optimizing s-box implementations for several criteria using SAT solvers. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 140–160. Springer, Heidelberg (2016)

[21] Bilgin, B., Meyer, L.D., Duval, S., Levi, I., Standaert, F.: Low AND depth and efficient inverses: a guide on s-boxes for low-latency masking. IACR Trans. Symmetric Cryptol. **2020**(1), 144–184 (2020)

[22] Fan, Y., Wang, W., Li, Z., Lu, Z., Yiu, S., Wang, M.: Forced independent optimized implementation of 4-bit s-box. In: Baek, J., Ruj, S. (eds.) Information Security and Privacy - 26th Australasian Conference, ACISP 2021, Virtual Event, December 1-3, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13083, pp. 151–170. Springer, Heidelberg (2021)

[23] Feng, J., Wei, Y., Zhang, F., Pasalic, E., Zhou, Y.: Novel optimized implementations of lightweight cryptographic s-boxes via sat solvers. IEEE Trans. Circuits Syst. I Regul. Pap., 1–14 (2023)

[24] Lu, Z., Mesnager, S., Cui, T., Fan, Y., Wang, M.: An stp-based model toward designing s-boxes with good cryptographic properties. Des. Codes Cryptogr. **90**(5), 1179–1202 (2022)

[25] Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES. In: Johansson, T. (ed.) Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers. Lecture Notes in Computer Science, vol. 2887, pp. 247–260. Springer, Heidelberg (2003)

[26] Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **22**(6), 710–722 (2003)

[27] Eén, N., Sörensson, N.: An extensible sat-solver. In: Giunchiglia, E., Tacchella, A. (eds.) Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers. Lecture Notes in Computer Science, vol. 2919, pp. 502–518. Springer, Heidelberg (2003)

[28] Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5584, pp. 244–257. Springer, Heidelberg (2009)

[29] Biere, A.: Cadical at the sat race 2019. In: Marijn, J.H.H., Matti, J., Martin, S. (eds.) Proceedings of SAT RACE 2019: Solver and Benchmark Descriptions, vol. B-2019-1 of Department of Computer Science Series of Publications, pp. 8–9. Department of Computer Science, University of Helsinki (2019)

[30] Ganesh, V., Dill, D.L.: A decision procedure for bit-vectors and arrays. In: Damm, W., Hermanns, H. (eds.) Computer Aided Verification, 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4590, pp. 519–531. Springer, Heidelberg (2007)

[31] Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. Lecture Notes in Computer Science, vol. 765, pp. 55–64. Springer, Heidelberg (1993)

[32] Dobbertin, H.: Construction of bent functions and balanced boolean functions

with high nonlinearity. In: Preneel, B. (ed.) Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings. Lecture Notes in Computer Science, vol. 1008, pp. 61–74. Springer, ??? (1994)

[33] Saeedi, M., Markov, I.L.: Synthesis and optimization of reversible circuits - A survey. CoRR **abs/1110.2574** (2011). http://arxiv.org/abs/1110.2574

[34] Kavun, E.B., Lauridsen, M.M., Leander, G., Rechberger, C., Schwabe, P., Yalçın, T.: Prøst v1.1. CAESAR submission (2014). http://competitions.cr.yp.to/round1/proestv11.pdf

[35] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)

[36] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer, Heidelberg (2016)

[37] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)

[38] Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci. China Inf. Sci. **58**(12), 1–15 (2015)

[39] Zhang, L., Wu, W., Wang, Y., Wu, S., Zhang, J.: LAC: A lightweight authenticated encryption cipher. CAESAR submission (2014). http://competitions.cr.yp.to/round1/lacv1.pdf

[40] Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 321–345. Springer, Cham (2017)

[41] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.1. CAESAR

submission (2015). http://competitions.cr.yp.to/round2/asconv11.pdf

[42] Bertoni, G., Daemen, J., Peeters, M., Assche, G.: The keccak reference (January 2011). http://keccak.noekeon.org/

[43] Leander, G., Poschmann, A.: On the classification of 4 bit s-boxes. In: Carlet, C., Sunar, B. (eds.) Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4547, pp. 159–176. Springer, Heidelberg (2007)