IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2025, No. 1, pp. 592–622. DOI:10.46586/tosc.v2025.i1.592-622

How Small Can S-boxes Be?

Chenhao Jia[1], Tingting Cui[1,3](✉), Qing Ling[1], Yan He[1], Kai Hu[2,3,4], Yu Sun[2] and Meiqin Wang[2,3,4,5]

[1] School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China
[2] School of Cyber Science and Technology, Shandong University, Qingdao, China
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao, China
[4] State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, China
[5] Quancheng Laboratory, Jinan, China

222270059@hdu.edu.cn, cuitingting@hdu.edu.cn, lingqing@hdu.edu.cn, yan_he@hdu.edu.cn, kai.hu@sdu.edu.cn, yu.sun@mail.sdu.edu.cn, mqwang@sdu.edu.cn

Abstract. S-boxes are the most popular nonlinear building blocks used in symmetric-key primitives. Both cryptographic properties and implementation cost of an S-box are crucial for a good cipher design, especially for lightweight ones. This paper aims to determine the exact minimum area of optimal 4-bit S-boxes (whose differential uniform and linearity are both 4) under certain standard cell library. Firstly, we evaluate the upper and lower bounds upon the minimum area of S-boxes, by proposing a Prim-like greedy algorithm and utilizing properties of balanced Boolean functions to construct bijective S-boxes. Secondly, an SAT-aided automatic search tool is proposed that can simultaneously consider multiple cryptographic properties such as the uniform, linearity, algebraic degree, and the implementation costs such as area, and gate depth complexity. Thirdly, thanks to our tool, we manage to find the exact minimum area for different types of 4-bit S-boxes.

The measurement in this paper uses the gate equivalent (GE) as standard unit under UMC 180 nm library, all 2/3/4-input logic gates are taken into consideration. Our results show that the minimum area of optimal 4-bit S-box is 11 GE and the depth is 3. If we do not use the 4-input gates, this minimum area increases to 12 GE and the depth in this case is 4, which is the same if we only use 2-input gates. If we further require that the S-boxes should not have fixed points, the minimum area continue increasing a bit to 12.33 GE while keeping the depth. Interestingly, the same results are also obtained for non-optimal 4-bit bijective S-boxes as long as their differential uniform $\mathcal{U}(S) < 16$ and linearity $\mathcal{L}(S) < 8$ (i.e., there is no non-trivial linear structures) if only 2-input and 3-input gates are used. But the minimum area reduce to 9 GE if 4-input gates are involved. More strictly, if we require the algebraic degree of all coordinate functions of optimal S-boxes be 3, the minimum area is 14 GE with fixed point and 14.33 GE without fixed point, and the depth increases sharply to 8.

Besides determining the exact minimum area, our tool is also useful to search for a better implementation of existing S-boxes. As a result, we find out an implementation of Keccak's 5-bit S-box with 17 GE. As a contrast, the designer's original circuit has an area of 23.33 GE, while the optimized result by Lu et al. achieves an area of 17.66 GE. Also, we find out the first optimized implementation of SKINNY's 8-bit S-box with 26.67 GE.

Keywords: S-box · automatic search · good cryptography properties · minimum area · SAT

Received: 2024-09-01 Revised: 2024-11-22 Accepted: 2025-01-23 Published: 2025-03-07

1 Introduction

1.1 Background

The substitution box (S-box) is the most widely used building blocks in symmetric-key primitives, including block ciphers, hash functions, authenticated encryptions and message authenticated codes. Security of these primitives heavily depends on the cryptographic properties of their S-boxes, such as the differential uniformity, linearity, algebraic degree, *etc*, thus choosing an S-box with good cryptographic properties is crucial for cipher designers. On the other hand, the rapid development of the Internet of Things (IoT) prefers ciphers that can be easily deployed in resource-constrained devices, thus the implementation cost of an S-box, such as the area and latency, needs to be as small as possible. To sum up, both the cryptographic properties and implementation cost of S-boxes are important, especially for lightweight ciphers.

There are two active research hotspots about S-boxes: constructing new S-boxes and optimizing existing S-boxes. The former aims to construct an S-box with excellent cryptographic properties but very small implementation cost, while the latter would like to find the smallest implementation cost of an existing S-box. Both research hotspots are challenging, especially for large S-boxes.

Previous effective methods available to construct S-boxes can be mainly classified into 3 types:

- Type 1: choose an S-box from equivalent categories;
- Type 2: construct an S-box by mathematical methods or special structures;
- Type 3: solve out an S-box by automatic tools.

The Type 1 method is only effective for constructing 4-bit S-boxes. In 2007, Leander and Poschman [LP07] defined optimal 4-bit S-boxes according to their ability against differential and linear attacks, and classified all optimal 4-bit S-boxes as 16 categories up to the so-called CCZ equivalence. Many cipher directly uses 4-bit S-boxes from the 16 equivalent categories, such as Serpent [BAK98] and RECTANGLE [ZBL+15]. This classification was further developed by Zhang et al. in [ZBRL15] to 183 categories. In the new classification, Bad-input-Bad-output (BIBO) differential/linear patterns were taken into consideration. Thanks to these classifications, it becomes easy to choose optimal 4-bit S-boxes from different categories as the nonlinear building block of symmetric ciphers. However, the implementation cost is not considered at all. In terms of Type 2 method, combining affine transformations and the inverse function of a finite field to construct S-boxes is a typical mathematical method, just like the S-box used in AES. For sake of S-boxes with low implementation area, some special structures such as Feistel, MISTY, Bridge [LW14, CDL15, TLZ23] as well as cellular automata [MPLJ19, GSP+18] are also widely used. However, these design strategies seriously depend on structures, so it is difficult to reach the minimum area. The Type 3 method, i.e., solving out an S-box by automatic tool, is an interesting and comprehensive method. In [LMC+22], Lu et al. proposed an automatic search method based on SAT, which can simultaneously take multiple cryptographic properties into consideration. However, algebraic degree, the very important cryptographic property of S-box, is not covered in this method. Meanwhile, the implementation cost is also not considered yet. Beside that, there are also some explorations on constructing low-depth or low-latency S-boxes[Ras22, TLZ23].

Previous work on optimizing the implementations of existing S-boxes available are few. Jean et al. [JPST17] proposed an automatic tool LIGHTER to search for the circuits with small area for existing S-boxes. They use a graph-based meet-in-the-middle search algorithm under the assumption that every instructions is invertible. Despite of the efficiency and practical applicability for different S-boxes, it is infeasible to prove that their implementation costs are optimal. In FSE 2016, Stoffelen regarded the problem of finding an efficient implementation of a lightweight S-box as a SAT problem [Sto16]. With an

SAT solver, the implementation of an S-box can be solved out with the smallest number of gates. Based on Stoffelen's work, Lu et al. [LWH+21] proposed an improved search algorithm and try to find optimized implementations for existing S-boxes with the smallest area under certain standard cell library. However, all previous methods aforementioned are only effective on constructing 4-bit S-boxes and ineffective on larger S-boxes. For example, Lu et al.'s method failed to give an optimal implementation for the 5-bit KECCAK's S-box.

To summarize, there still remain several questions awaiting for answers on designing and optimizing implementation of S-boxes:

- Q1: What is the minimum area of all 4-bit S-boxes, if we would like to design new S-boxes or optimize existing S-boxes under certain standard cell library?
- Q2: How to optimize the implementation for existing n-bit ($n \geq 5$) S-boxes with smaller area?
- Q3: How to consider area and depth complexity of an S-box simultaneously when designing new S-boxes and optimizing existing S-boxes?
- Q4: How to cover the requirement on algebraic degree in the automatic search for S-boxes?

In this paper, we aim to propose an improved automatic search method based on SAT to answer these questions.

1.2 Contributions

Since the area cost of different logic gates depends on the technology library, we use the gate equivalent (GE) as the standard unit under the UMC 180nm standard cell library shown in Table 5 to measure and compare the area of S-boxes. The main contributions are briefly summarized as follows:

Propose Prim-like greedy algorithm and properties on balanced Boolean functions to tighten the upper bound and lower bound of the minimum area of S-boxes. Inspired by Prim algorithm in Graph Theory, we transform the process to find the minimum area of S-boxes into a minimum spanning tree problem under some conditions. As an application, to construct an optimal 4-bit S-box, the upper bound of minimum area is 12 GE (8 logic gates) by only using 2-input (or plus 3-input) gates, or 11 GE (6 logic gates) by using 2-input, 3-input and 4-input gates. To find the lower bound of the minimum area of S-boxes, we proposed some properties on balanced Boolean functions, which are helpful to speed up the search process. As a result, we found that it needs at least 9 GE to construct a bijective S-box with $\mathcal{U}(S) < 16$ and $\mathcal{L}(S) < 8$ by only using 2-input and 3-input gates. These findings are useful to shrink the possible range of the minimum area of S-box.

Improve the automatic search model for S-boxes in [LMC+22] via adding constraints on algebraic degree. Algebraic degree is an important cryptographic property of S-boxes. It can be found from the ANFs of S-box's coordinate functions. However, constructing ANF of a Boolean function is difficult by the truth table in SAT model. In this paper, we overcome this problem by the transformation from SoP expression of Boolean function to ANF expression. Therefore, it becomes able to express every coefficient in ANF and add constraints on algebraic degree within the SAT model.

Improve the automatic search method for optimizing implementation of existing S-boxes in [LWH+21] via some acceleration techniques and adding constraints on depth complexity. By utilizing the order-independence of inputs for 2-input and 3-input logic gates, as well as bit-permutation equivalence of S-boxes, the search space can be significantly reduced. Beside that, the requirement on depth complexity can be added into our improved model. As a result, we can search out better implementation

Table 1: Comparison of the minimum area cost of optimal 4-bit S-Boxes and a few higher-bit S-boxes.

S-box	Size	# Gate	Basis	Area (GE)			
				LIGHTER [JPST17]	Stoffelen et al. [Sto16]	Lu et al. [LWH+21]	Ours
LBLOCK S_0	4	10	$\{\mathcal{G}_1, \mathcal{G}_2\}$	16.33	23.00	16.33	16.33
PICCOLO	4	8	$\{\mathcal{G}_1, \mathcal{G}_2\}$	13.00	16.66	13.00	13.00
SKINNY-64	4	8	$\{\mathcal{G}_1, \mathcal{G}_2\}$	13.33	16.33	13.33	13.33
RECTANGLE	4	11	$\{\mathcal{G}_1, \mathcal{G}_2\}$	18.33	25.66	18.00	18.00
			$\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$	–	–	17.33	17.33
Ours	4	8	$\{\mathcal{G}_1, \mathcal{G}_2\}$	–	–	–	12.00
Ours	4	6	$\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$	–	–	–	11.00
KECCAK	5	13	$\{\mathcal{G}_1, \mathcal{G}_2\}$	–	–	17.66	17.00
SKINNY-128	8	16	$\{\mathcal{G}_1, \mathcal{G}_2\}$	–	–	–	26.67

of 5-bit S-box used in KECCAK with smaller area. Meanwhile, we give the optimizing implementation of 8-bit S-box used in SKINNY with smaller area for the first time. The results are summarized in Table 1.

Propose an automatic search model for S-boxes by considering multiple cryptographic properties and implementation area and depth complexity simultaneously based on SAT. In theory, combining the models in [LMC+22] and [LWH+21] and some slight changes makes it possible to find an S-box satisfying required cryptographic properties and area cost. However, by experiments, we found that the combined model could not be solved out for even 4-bit S-boxes when area cost is restricted as a lower value. In other words, the direct combined model cannot be used to find the minimum area of 4-bit S-boxes. However, based on the first three contributions, we made our model be able to solve the above problem. Specifically, by combining the improved models in the second and third contributions, it is able to determine the minimum area between upper bound and lower bound. Under the minimum area of S-boxes, we can further find the minimum depth. As a result, we apply it on 4-bit S-boxes. If only 2-input and 3-input logic gates are used to construct general bijective 4-bit S-boxes with $\mathcal{U}(S) < 16$ and $\mathcal{L}(S) < 8$, the minimum area is 12 GE with fixed point, as well as 12.33 GE without fixed point. So does optimal 4-bit S-boxes. Under each case above, the depth is at least 4. Furthermore, the minimum area is 14 GE with fixed point, as well as 14.33 GE without fixed point to construct optimal 4-bit S-boxes with all algebraic degree 3 of coordinate functions. Under each case above, the depth is at least 8. If all logic gates including 4-input gates are used to construct optimal 4-bit S-boxes, the minimum area is 11 GE without fixed point, now the gate depth is at least 3. If all logic gates are involved to construct general bijective 4-bit S-boxes, the minimum area is 9 GE, and the gate depth is also at least 3. All results are summarized in Table 2. All source codes of our tool and searched results are available at `https://github.com/Chenhao-Jia/Area-Optimized-Implementation-for-S-box`.

2 Preliminaries

2.1 Notations

This subsection introduces notations and conceptions used in this paper. Some simple notations are given in Table 3.

A Boolean function is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2$, while a vectorial Boolean function is a mapping from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$:

$$(x_0, x_1, \cdots, x_{n-1}) \mapsto (f_0(x_0, x_1, \cdots, x_{n-1}), f_1(x_0, x_1, \cdots, x_{n-1}), \cdots, f_{m-1}(x_1, x_2, \cdots, x_{n-1})),$$

where each $f_i(x_0, x_1, \cdots, x_{n-1})$ is a Boolean function. An S-box with n-bit input and

Table 2: Results on the minimum area cost of three types of S-boxes under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$, $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ and $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$.

Basis	$\mathcal{U}(S)$	$\mathcal{L}(S)$	Min deg(S)	Max deg(S)	Without fixed point	Optimal S-box	#Logic gate	Area UMC 180nm	Area TSMC 65nm	Minimum area	Depth
$\{\mathcal{G}_1, \mathcal{G}_2\}$	4	4	2	3	×	✓	8	12 GE	14 GE	✓	4
			2	3	✓	✓	8	12.33 GE	14.5 GE	✓	4
			3	3	×	✓	9	14 GE	16.5 GE	✓	8
			3	3	✓	✓	9	14.33 GE	17 GE	✓	8
	< 16	< 8	2	3	×	×	8	12 GE	14 GE	✓	4
			2	3	✓	×	8	12.33 GE	14.5 GE	✓	4
			2	2	×	×	8	12 GE	14 GE	✓	4
			2	2	✓	×	8	12.33 GE	14.5 GE	✓	4
$\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$	4	4	2	3	×	✓	8	12 GE	14 GE	✓	4
			2	3	✓	✓	8	12.33 GE	14.5 GE	✓	4
			3	3	×	✓	9	14 GE	16.5 GE	✓	8
			3	3	✓	✓	9	14.33 GE	17 GE	✓	8
	< 16	< 8	2	3	×	×	8	12 GE	14 GE	✓	4
			2	3	✓	×	8	12.33 GE	14.5 GE	✓	4
			2	2	×	×	8	12 GE	14 GE	✓	4
			2	2	✓	×	8	12.33 GE	14.5 GE	✓	4
$\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$	4	4	2	3	✓	✓	6	11 GE	13.5 GE	✓	3
	< 16	< 8	2	3	×	×	5	9 GE	11 GE	✓	3

Table 3: Notations used in this paper.

$\mathbb{Z}_n$	The finite set $\{0, \cdots, n-1\}$.
$\mathbb{F}_2$	The finite field with only two elements $\{0, 1\}$.
$\mathbb{F}_2^n$	The n-dimensional vector space over $\mathbb{F}_2$.
$x \oplus y$	Bitwise exclusive OR of x and y
$a \wedge b$	AND of Boolean variable a and b which equals to ab.
$a \vee b$	OR of Boolean variable a and b which equals to $ab + a + b$.
$\neg a$	NOT of Boolean variable a which equals to $a + 1$.
$\mathcal{G}_1$	The logic gate set with only 1 input, i.e. $\mathcal{G}_1 = \{NOT\}$.
$\mathcal{G}_2$	The nonlinear logic gate set with 2 inputs, i.e. $\mathcal{G}_2 = \{AND, NAND, OR, NOR\}$.
$\mathcal{G}_3$	The nonlinear logic gate set with 3 inputs, i.e. $\mathcal{G}_3 = \{AND3, NAND3, OR3, NOR3\}$.
$\mathcal{G}_4$	The nonlinear logic gate set with 4 inputs, i.e. $\mathcal{G}_4 = \{MAOI1, MOAI1\}$.
$\mathcal{G}l$	The linear logic gate set except NOT, i.e. $\mathcal{G}l = \{XOR, XNOR\}$.
$wt(u)$	The Hamming weight of u where $u \in \mathbb{F}_2^n, u = (u_0, \cdots, u_{n-1})$.
$f(x_0, \cdots, x_{n-1})$	A function mapping $\mathbb{F}_2^n$ to $\mathbb{F}_2$, also called an n-variable Boolean function.
$\langle x, y \rangle$	The inner product of a and b which equals to $\sum_{i=0}^{n-1} x_i y_i$.
$\#\{\}$	The number of qualified set elements.

[1] A Boolean variable is regard as a variable over $\mathbb{F}_2$ in this paper.

[2] Unless otherwise specified, $+$ and $\sum$ in this paper are operations over $\mathbb{F}_2$.

m-bit output can be represented by such a vectorial Boolean function.

Boolean circuits are defined according to the logic gates they contain. For example, a circuit might contain binary AND and OR gates and unary NOT gates. Each gate corresponds to a small Boolean function that takes a fixed number of bits as input and outputs a single bit. The most common gates used in a circuit are shown in Table 4. This paper considers implementations of S-boxes only with these gates.

2.2 Main Cryptographic Properties for an S-Box

We will first give the definitions of an S-box related to differential cryptanalysis, linear cryptanalysis and various forms of algebraic/cubic cryptanalysis.

Definition 1 (Differential Distribution Table(DDT)[BS90, BS93])**.** For a vectorial Boolean function $S : \mathbb{F}_2^m \to \mathbb{F}_2^n$, the DDT of S is a $2^m \times 2^n$ table whose rows correspond to the input difference α to S and whose columns correspond to the output difference β of S.

Table 4: Common logic gates and their corresponding expressions in standard cell library

Operation	Function	Operation	Function
NAND	$(a, b) \to \neg(a \wedge b)$	NAND3	$(a, b, c) \to \neg(a \wedge b \wedge c)$
NOR	$(a, b) \to \neg(a \vee b)$	NOR3	$(a, b, c) \to \neg(a \vee b \vee c)$
AND	$(a, b) \to (a \wedge b)$	AND3	$(a, b, c) \to (a \wedge b \wedge c)$
OR	$(a, b) \to (a \vee b)$	OR3	$(a, b, c) \to (a \vee b \vee c)$
NOT	$a \to \neg a$	MAOI1	$(a, b, c, d) \to \neg((a \wedge b) \vee (\neg(c \vee d)))$
XOR	$(a, b) \to (a \oplus b)$	MOAI1	$(a, b, c, d) \to \neg((a \vee b) \wedge (\neg(c \wedge d)))$
XNOR	$(a, b) \to \neg(a \oplus b)$		

The entry at index (α, β) is

$$\delta_S(\alpha, \beta) := \#\left\{x \in \mathbb{F}_2^m | S(x) \oplus S(x \oplus \alpha) = \beta\right\}. \tag{1}$$

Definition 2 (Differential Uniformity[Nyb93]). The differential uniformity of an S-box $S : \mathbb{F}_2^m \to \mathbb{F}_2^n$ is defined as:

$$\mathcal{U}(S) := \max_{\alpha \in \mathbb{F}_2^m \setminus \{0\}, \beta \in \mathbb{F}_2^n} \delta_S(\alpha, \beta). \tag{2}$$

Definition 3 (Linear Approximation Table (LAT)[Mat93]). For a vectorial Boolean function $S : \mathbb{F}_2^m \to \mathbb{F}_2^n$, the LAT of S is a $2^m \times 2^n$ table whose rows correspond to the input mask α to S and whose columns correspond to the output mask β of S. The entry at index (α, β) is

$$LAT_S(\alpha, \beta) := |\lambda_S(\alpha, \beta) - 2^{n-1}|. \tag{3}$$

where $\lambda_S(\alpha, \beta) = \#\left\{x \in \mathbb{F}_2^m | \alpha \cdot x \oplus \beta \cdot S(x) = 0\right\}$.

Definition 4 (Linearity[Mat93]). The linearity of an S-box $S : \mathbb{F}_2^m \to \mathbb{F}_2^n$ is defined as:

$$\mathcal{L}(S) := \max_{\alpha \in \mathbb{F}_2^m \setminus \{0\}, \beta \in \mathbb{F}_2^n} |LAT_S(\alpha, \beta)|.$$

In [LP07], Leander and Poschmann defined the *optimal 4-bit S-boxes* as those S-boxes that simultaneously achieve optimal differential uniformity and linearity:

Definition 5 (Optimal 4-bit S-boxes[LP07]). Let $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ be an S-box. If S fulfills the following conditions, then it is called an optimal 4-bit S-box:

1. *S is a bijection.*
2. $\mathcal{U}(S) = 4$.
3. $\mathcal{L}(S) = 4$.

Saarinen et al. put forward the definition of permutation-xor equivalence in [Saa11]. The algebraic degree, linearity and uniformity are example properties of (vectorial) Boolean functions that are invariant over any of these equivalences. Rasoolzadeh showed that the latency complexity is invariant also under the extended bit permutation equivalence [Ras22]. The definition of bit permutation equivalent is as follows:

Definition 6 (Bit permutation equivalent[ZBRL15]). Let P_1 and P_0 be two bit permutation matrices. The S-box S' defined by

$$S'(x) = P_0 S(P_1(x))$$

belongs to the permutation-xor equivalence set of S, $S' \in PE(S)$.

Thus, the S-boxes within the same bit permutation equivalent class share the same algebraic degree, uniformity, and linearity. Moreover, it is evident that the area and depth complexity of the S-boxes within this class are also identical.

When applying to optimal 4-bit S-boxes, the following theorem formalized as follows.

Theorem 1. [LP07] *Let S' permutation equivalent to S. If S is an optimal 4-bit S-box, then S' is an optimal 4-bit S-box as well.*

In addition to the differential uniformity and linearity, another metric to determine the security of an S-box is the algebraic degree.

Definition 7 (Algebraic Normal Form (ANF) of a Boolean function and its algebraic degree [CP02, DS09])**.** A Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ can be uniquely represented by an n-variate polynomial over $\mathbb{F}_2$, named the algebraic normal form of f:

$$f(x_0, x_1, \ldots, x_{n-1}) = \sum_{u \in \mathbb{F}_2^n} \alpha_u \prod_{i=0}^{n-1} x_i^{u_i}, \text{ where } \alpha_u \in \mathbb{F}_2.$$

where x_i, u_i are the i-th bit of x and u, respectively, and $x_i^{u_i} = x_i$ when $u_i = 1$ and 1 when $u_i = 0$. The algebraic degree $\deg(f)$ of function f is

$$\deg(f) = \max_{u \in \mathbb{F}_2^n} \{\mathrm{wt}(u) | \alpha_u \neq 0\}.$$

Lemma 1 ([CCH10, LN97])**.** *Let f be an n-variable balanced Boolean function. Then algebraic degree of f is at most $n-1$.*

Definition 8 (Algebraic Degree of an S-box[LP07])**.** The algebraic degree of an S-box $S : (x_0, \cdots, x_{n-1}) \mapsto (f_0, \cdots, f_{m-1})$ is

$$deg(S) = \max_i \{deg(f_i)\}.$$

2.3 Implementation of Boolean Functions

The area and latency are two primary metrics when measuring the hardware implementation cost of an S-box. The area of an S-box is typically measured in terms of gate equivalent (GE). In different libraries GEs of different gates are different. In this paper, we evaluate the area of S-boxes under different standard cell librarys as shown in Table 5. It should be noted that unless specified, we measure the area of S-boxes under UMC 180nm library. Table 5 also provides the area of each gate in the respective libraries.

Table 5: GEs of mentioned gates in different libraries.

Library	Gate	AND	NAND	XOR	NAND3	XOR3	NOT	MAOI1	MOAI1
		OR	NOR	XNOR	NOR3	XNOR3			
UMC 180nm	Area (GE)	1.33	1.00	3.00	1.33	4.67	0.67	2.67	2.00
	Ratio	4	3	9	4	14	2	8	6
TSMC 65nm	Area (GE)	1.50	1.00	3.00	1.50	5.50	0.50	2.50	2.50
	Ratio	6	4	12	6	22	2	10	10
Nangate 45nm	Area (GE)	1.33	1.00	2.00	1.33	4.00	0.67	2.67	2.00
	Ratio	4	3	9	4	12	2	–	–

[1] The ratio is measured in multiples of $\frac{1}{3}$ GE as the reference unit.

It can be seen that, AND and OR gates, as well as XOR gates, have a larger GE plus NOT than their inverse gates at UMC 180nm, so we do not take NOT gates into our consideration when pursuing GE minimization. We recall the definitions of *Gate Count Complexity* and *Gate Depth Complexity* to measure the cost of implementing Boolean functions as follows.

Definition 9 (Gate Count Complexity[Sto16])**.** The gate count complexity of a Boolean function is defined as the minimum number of logic gates required to implement this function.

Even though different types of gates have different implementation (area) costs, this definition is typically considered the first simplified estimation for the minimum area cost for hardware implementation of a function.

In [Sto16], Stoffelen transformed the problem of searching for hardware and software implementations of S-boxes into a Boolean satisfiability problem for solving. Compared to heuristic algorithms, the model constructed by Stoffelen can be used to search for the minimum number of standard logic gates required for hardware and software implementation of S-boxes, achieving the smallest possible number of standard logic gates used. However, since different standard logic gates have different hardware implementation areas, a circuit with the minimum number of standard logic gates does not necessarily have the minimum area.

Definition 10 (Gate Depth Complexity[Sto16])**.** The depth of a circuit is defined as the length of the longest paths from an input gate to an output gate.

In the case of the gate depth complexity, even though different types of gates have different implementation (delay) costs, this definition is usually considered the first estimation for the minimum delay cost for hardware implementation of a function.

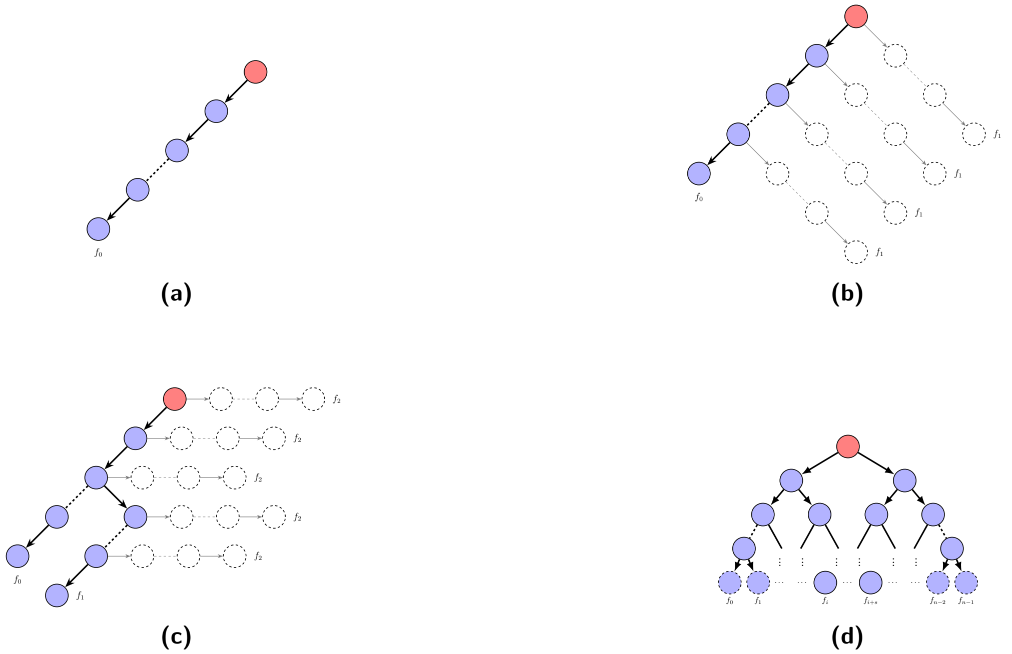
3 Tight Bounds on Minimum Area to Construct S-boxes

In this section, we first propose a Prim-like greedy algorithm to tighten the upper bound of the minimum area to construct S-boxes with certain cryptographic properties in subsection 3.1. Then, by finding some properties on balanced Boolean functions, the lower bound of minimum area to construct S-boxes is tightened in subsection 3.2.

3.1 Upper Bound of Minimum Area to Construct S-boxes under Certain Cryptographic Properties

In this part, we concentrate on finding the upper bound on the minimum area for constructing a bijective S-box with differential uniform $\mathcal{U}(S)$ and linearity $\mathcal{L}(S)$. Each S-box coordinate function $f_i(x)$ is composed of a certain number of logic gates, and meanwhile, different coordinate functions share some gates. To find the minimum area of an S-box is to find the optimal circuit implementation with minimum gate area. Thus, we transform the construction of an S-box into a minimum spanning tree problem under conditions with the help of graph theory. The circuit implementation process of an S-box can be regarded as a weighted diagraph, as shown in Figure 1.

In this diagraph, there are two sets: edge set E and vertex set V. Each edge $g_i \in E$ means a logic gate, while the weight w_i of g_i means the are cost of this gate. The vertexes (v_i, v_{i+1}) on both sides of edge g_i means the input/output states of corresponding logic gate. It is worth noting that, there may be multiple inputs for a gate. To describe clearly, the vertex v_i in graph only denotes the input with longest depth. But, in actual search with code, v_i is stored as an expression $g_i(x_0, x_i, \cdots, x_{n-1}, v_1, \cdots, v_{i-1})$. Finally, the output $(f_0, \cdots, f_{m-1})$ of S is located in some vertexs. To find the minimum area of S-boxes under certain differential uniform $\mathcal{U}(S)$ and linearity $\mathcal{L}(S)$ is to find the minimum of $\sum_{i=1}^{n} w_i$ in the diagraph to get special $(f_0, \cdots, f_{m-1})$ satisfies Lemma 2 and Lemma 3 as follows.



(a) (b) (c) (d)

Figure 1: Demo on Prim-like greedy Algorithm

Lemma 2 ([Ras22])**.** *For an n-bit bijective S-box $S = (f_0, \ldots, f_{n-1})$ with linearity linearity $\mathcal{L}(S)$, each of its component functions, namely $\langle \alpha, S \rangle$ with any $\alpha \in \mathbb{F}_2^n \setminus \{0\}$, is balanced and has a linearity of at most $\mathcal{L}(S)$.*

Using Lemma 2 makes it possible to filter out some of the possibilities, only by having some of the coordinate functions. Precisely, assume that f_0 and f_1 are already chosen, then without choosing other coordinate functions, we can check for balancedness and linearity of $f_0 \oplus f_1$. If $f_0 \oplus f_1$ is balanced and has a linearity at most ℓ, then we choose the third coordinate function, f_2. Again, we can check for balancedness and linearity of $f_0 \oplus f_2$, $f_1 \oplus f_2$, and $f_0 \oplus f_1 \oplus f_2$. Continuing in this way, after choosing the last coordinate function, f_{n-1}, we can check for balancedness and linearity of other $2^{n-1} - 1$ component functions. If these $2^{n-1} - 1$ conditions are met, then we have a bijective S-box with linearity at most ℓ.

Lemma 3 ([Ras22])**.** *For an n-bit S-box $S = (f_0, \ldots, f_{n-1})$ with differential uniform $\mathcal{U}(S)$, the differential uniformity of sub-S-box $S'_i = (f_0, \ldots, f_i)$ with $i < n$ is upper bounded by min$\{\mathcal{U}(S) \cdot 2^{n-i-1}, 2^n\}$.*

Lemma 3 can also be used to filter out coordinate functions that meet the differential uniformity requirement for the sub-S-box $S'_i = (f_0, f_1, \ldots, f_i)$.

Next, we illustrate the algorithm to find the minimum of $\sum_{i=1}^{n} w_i$ meeting certain differential uniform and linearity with the help of Lemma 2 and Lemma 3, which is shown in Algorithm 1. Inspired by Prim algorithm in graph theory, we construct the coordinate functions of an S-box one by one to achieve local optimization. At the very begin, we build a pre-computation table $T[cost]$ to store all combinations of logic gates, indexing by area cost as shown in Table 6. Please note that the combinations in this table are ordered by the number of used gates.

In detail, to construct the first coordinate function f_0, the steps are as follows.

- **Step 1:** Try the combination of logic gates from pre-computation $T[cost]$ successively according to area cost. Assume the current combination of logic gates is $\mathcal{I}_{f_0}$, and

Algorithm 1: Prim-like greedy Algorithm to achieve the locally minimum solution for area cost.

```
Input: $n$, $\mathcal{G}$, $V$
Output: $Cost$
$V^0, \ldots, V^{n-1} \leftarrow \emptyset$;
$V_y = \emptyset$;
global $Cost,\ Cost' = 0$;
global $iter = -1$;
Function FindingLocalMinimumArea($n, \mathcal{G}, V$):
    $iter{+}{+}$;
    for $i = 3, 4, 5...$ do
        if $count = 0$ then
            for $k = 1$ to $\lfloor i/3 \rfloor$ do
                if $count = 0$ then
                    for $g_0, \ldots, g_{k-1} \in \mathcal{G}$ and $\sum_{j=0}^{k-1} |g_j| = i$ do
                        $V^{iter} = V$;
                        for $v_0, \ldots, v_{n-1}\ (n = 2, 3, 4) \in V^{iter}$ do
                            $t_j = g_j(v_0, \ldots, v_{n-1}),\ V^{iter} \leftarrow V^{iter} \cup \{t_j\}$;
                        if $\{\alpha \in \mathbb{F}_2^{iter+1} \backslash \{0\} | \langle \alpha, V_y \cup t_{k-1} \rangle\}$ is balanced && $\mathcal{L}(V_y \cup t_{k-1}) \leq \mathcal{L}(S)$ then
                            if $\mathcal{U}(V_y \cup t_{k-1}) \leq min\{u \cdot 2^{n-iter-1}, 2^n\}$ then
                                $V \leftarrow V \cup V^{iter},\ V_y \leftarrow V_y \cup t_{k-1}$;
                                $count++$;
                                if $iter \leq n-1$ then
                                    FindingLocalMinimumArea $(n, \mathcal{G}, V)$;
                                break
    return
$Cost = Cost'/3$
```

Table 6: Precomputed table for gate combinations.

cost(GE)	combinations of logic gates	# combinations
$1/3 \times 1$	∅	0
$1/3 \times 2$	{NOT}	1
⋮	⋮	⋮

$1/3 \times 5$	{NOT, NAND}, {NOT, NOR}
	{NAND, NOT}, {NOR, NOT}
		4

⋮	⋮	⋮

the number of logic gates used in $\mathcal{I}_{f_0}$ is $|\mathcal{I}_{f_0}|$:

$$\mathcal{I}_{f_0} : v_0 \xrightarrow{g_0} v_1 \xrightarrow{g_1} \cdots \xrightarrow{g_{|\mathcal{I}_{f_0}|-2}} v_{|\mathcal{I}_{f_0}|-1} = f_0(x_0, \cdots, x_{n-1}), \tag{4}$$

where v_0 is the initial vertex, and $v_{|\mathcal{I}_{f_0}|-1}$ is the final vertex.

- **Step 2:** Try possible inputs of gate g_0 from initial input set $V = \{x_0, \cdots, x_{n-1}\}$, possible inputs of gate g_1 from $V \cup \{v_1\}$, ..., possible inputs of gate $g_{|\mathcal{I}_{f_0}|-1}$ from $V \cup \{v_1, v_2, \cdots, v_{|\mathcal{I}_{f_0}|-2}\}$.

- **Step 3:** Check if $v_{|\mathcal{I}_{f_0}|-1} = f_0(x_0, \cdots, x_{n-1})$ satisfies Lemma 2 and Lemma 3. If so, continue to construct the next coordinate function. Otherwise, return back to Step 2 and Step1.

  This process is shown in Figure 1(a). Now the area cost is $\sum_{i=0}^{|\mathcal{I}_{f_0}|-1} w_i$.

To construct the second coordinate function f_1 is slightly different with the process to construct f_0. The steps are as follows:

- **Step 1:** Try the combination of logic gates from the pre-computation table $T[cost]$ successively according to area cost. Assume the current combination of logic gates is $\mathcal{I}_{f_1}$, and the number of logic gates used in $\mathcal{I}_{f_1}$ is $|\mathcal{I}_{f_1}|$:

$$\mathcal{I}_{f_1} : v_0' \xrightarrow{g_{|\mathcal{I}_{f_0}|}} v_{|\mathcal{I}_{f_0}|} \xrightarrow{g_{|\mathcal{I}_{f_0}|+1}} v_{|\mathcal{I}_{f_0}|+1} \rightarrow \cdots \rightarrow v_{|\mathcal{I}_{f_0}|+|\mathcal{I}_{f_1}|-1} = f_1(x_0, \cdots, x_{n-1}). \tag{5}$$

- **Step 2:** Choose initial vertex v_0' from set $\mathcal{V} = \{v_0, v_1, \cdots, v_{|\mathcal{I}_{f_0}|-1}\}$ one by one.

- **Step 3:** Assume the current $v_0' = v_i$, $0 \leq i \leq |\mathcal{I}_{f_0}| - 1$. Try possible inputs of gate $g_{|\mathcal{I}_{f_0}|}$ from initial input set $V = \{x_0, \cdots, x_{n-1}, v_0, \cdots, v_i\}$, possible inputs of gate $g_{|\mathcal{I}_{f_0}|+1}$ from $V \cup \{v_{|\mathcal{I}_{f_0}|}\}$, ... , possible inputs of gate $g_{|\mathcal{I}_{f_0}|+|\mathcal{I}_{f_1}|-1}$ from $V \cup \{v_{|\mathcal{I}_{f_0}|}, v_{|\mathcal{I}_{f_0}|+1}, \cdots, v_{|\mathcal{I}_{f_0}|+|\mathcal{I}_{f_1}|-2}\}$.

- **Step 4:** Check if $v_{|\mathcal{I}_{f_0}|+|\mathcal{I}_{f_1}|-1} = f_1(x_0, \cdots, x_{n-1})$ satisfies Lemma 2 and Lemma 3. If so, continue to construct the next coordinate function. Otherwise, return back to Step 3, Step 2 and Step 1.

  This Process is shown in Figure 1(b). Now the area cost $\sum_{i=0}^{|\mathcal{I}_{f_0}|-1} w_i + \sum_{i=0}^{|\mathcal{I}_{f_1}|-1} w_{|\mathcal{I}_{f_0}|+i}$.

To construct the other coordinate functions $f_2, \cdots, f_{n-1}$ sequentially is similar to the process to construct f_1. But the difference lies in that the initial vertex set $\mathcal{V}$ and the input set V would be changed and larger. The process is shown in Figure 1(c)(d). In the end, the total area cost is $\sum_{i=0}^{|\mathcal{I}_{f_0}|+|\mathcal{I}_{f_1}|+\cdots+|\mathcal{I}_{f_{n-1}}|-1} w_i$.

Application to 4-bit S-boxes. We apply Algorithm 1 on 4-bit S-boxes with both differential uniformity and linearity as 4. By using different sets of logic gates, we get different upper bound of minimum area of 4-bit S-boxes under UMC 180nm library. When we only use 2-input logic gates from $\mathcal{G}_2$, the minimum area of S-boxes we found is 12 GE, composed of 8 logic gates. When we use 2-input and 3-input logic gates from $\{\mathcal{G}_2, \mathcal{G}_3\}$, the minimum area of S-boxes we found is 12 GE, composed of 8 logic gates. When we use 2-input, 3-input and 4-input logic gates from $\{\mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$ together, the minimum area of S-boxes we found is 11 GE, composed of 6 logic gates. The results are summarized in Table 7.

Table 7: Upper bound of minimum area of optimal 4-bit S-boxes under UMC 180nm library

$\mathcal{U}(S)$	$\mathcal{L}(S)$	basis	#logic gate	area (GE)
4	4	$\{\mathcal{G}_2\}$	8	12
4	4	$\{\mathcal{G}_2, \mathcal{G}_3\}$	8	12
4	4	$\{\mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$	6	11

3.2 Lower Bound of Minimum Area to Construct Bijective S-boxes

In this part, we try to seek the lower bound of the minimum area to construct a bijective S-box based on the balanced property of Boolean functions. By default, the algebraic degree of each coordinate function of the S-box should be no less than 2.

Firstly, we propose some properties to real how to construct balanced Boolean functions implemented by logic gates with 2 or 3 inputs.

Property 1. If f is an n-variables Boolean function with algebraic degree $l (l \geq 2)$ obtained by logic gate g, i.e. $f : V \xrightarrow{g} t = f(V = \{x_0, \cdots, x_{n-1}\})$, where g is a logic gate involved in this paper except $\mathcal{G}_1 \cup \mathcal{G}_4$ (*i.e.* $g \in (\mathcal{G}l \cup \mathcal{G}_2 \cup \mathcal{G}_3)$), then $f(x_0, \cdots, x_{n-1})$ is not balanced.

Proof. Suppose that $f(x_0, \cdots, x_{n-1})$ is a balanced Boolean function obtained by logic gate g in $(\mathcal{G}l \cup \mathcal{G}_2 \cup \mathcal{G}_3)$.

Case 1. *The gate g is in $\mathcal{G}l$.* Obviously, the Boolean function expression of gate g would be $g(x_i, x_j) = x_i + x_j$ or $g(x_i, x_j) = x_i + x_j + 1$. Thus, $deg(f) = deg(g) = 1$ which contradicts the algebraic degree of f being greater than or equal to 2.

Case 2. *The gate g is not in $\mathcal{G}l$.* According to the expressions of the logic gates given in Table 4, we have Boolean function expressions of gate g^2 in $\mathcal{G}_2$ and gate g^3 in $\mathcal{G}_3$ as shown in Equation 6 and 7 respectively.

$$g^2(x_i, x_j) = x_i x_j + \lambda_1(x_i + x_j) + \lambda_0, \tag{6}$$

$$g^3(x_i, x_j, x_k) = x_i x_j x_k + \lambda_1(x_i x_j + x_i x_k + x_j x_k + x_i + x_j + x_k) + \lambda_0, \tag{7}$$

where λ_0 and λ_1 are constants with values of 0 or 1 and $x_i, x_j, x_k \in \mathbb{F}_2$. Obviously, $g^2(x_i, x_j)$ is a binary Boolean function about variables x_i and x_j, and its leading term is $x_i x_j$, i.e. $deg(g^2(x_i, x_j)) = 2$. According to Lemma 1, $g^2(x_i, x_j)$ is not a balanced Boolean function. Similarly, $g^3(x_i, x_j, x_k)$ is also not a balanced Boolean function. Thus, whether $g = g^2$ or $g = g^3$, g is unbalanced.

Therefore, $f(x_0, \cdots, x_{n-1})$ with algebraic degree $l (l \geq 2)$ obtained by logic gate g is not a balanced Boolean function. □

Property 2. The function $f(x_0, \cdots, x_{n-1})$ is an n-variable balanced Boolean function with algebraic degree $l (l \geq 2)$ obtained by the composite use of logic gates g_0 and g_1, i.e. $f : V \xrightarrow{g_0} t_0 \xrightarrow{g_1} t_1 = f(V = \{x_0, \cdots, x_{n-1}\})$, where g_0 and g_1 are logic gates involved in this paper except $\mathcal{G}_1 \cup \mathcal{G}_4$ and the output t_0 of g_0 is an input of g_1, if and only if g_0 is a nonlinear gate and g_1 is a linear gate.

Proof. Denote t_0 and t_1 as the outputs of g_0 and g_1 respectively. There are four cases below.

Case 1. *The gate g_0 and g_1 are both in $\mathcal{G}l$.* Obviously, the Boolean function expression of gate g would be $g_1(t_0) = t_0 + 1$ or $g_1(t_0, x_i) = t_0 + x_i$ or $g_1(t_0, x_i) = t_0 + x_i + 1$. By the proof of Property 1, we have $deg(f) = deg(g_1) = deg(t_0) = 1$ which contradicts the algebraic degree of f being greater than or equal to 2.

Case 2. *The gate g_0 is in $\mathcal{G}l$ and g_1 is in $\mathcal{G}_2 \cup \mathcal{G}_3$.* It is clear that $t_0 = g_0(x_i, x_j) = x_i + x_j$ or $t_0 = g_0(x_i, x_j) = x_i + x_j + 1$ and $Pr(t_0 = 0) = \frac{1}{2}$. Depending on the number of inputs to g_1, there are two subcases.

Subcase 1. *The gate g_1 is in $\mathcal{G}_2$.* According to Equation 6, we have

$$t_1 = g^2(t_0, x_i) = t_0x_i + \lambda_1(t_0 + x_i) + \lambda_0 = \begin{cases} (\lambda_1 + 1)x_i + \lambda_1 + \lambda_0 & t_0 = 1 \\ \lambda_1 x_i + \lambda_0 & t_0 = 0. \end{cases}$$

Since the value of t_0 is determined by two input variables, whether there is a dependency between t_0 and x_i, we have $0 < Pr(x_i = u|t_0 = v) < 1$ where $u, v \in \{0, 1\}$. Thus

$$\begin{aligned} Pr(t_1 = 0) =& Pr(t_0 = 1) \cdot Pr((\lambda_1 + 1)x_i + \lambda_1 + \lambda_0 = 0) \\ &+ Pr(t_0 = 0) \cdot Pr(\lambda_1 x_i + \lambda_0 = 0) \\ =& \begin{cases} \frac{1}{2}Pr(x_i = 0|t_0 = 1) + \frac{1}{2} & \lambda_1 = 0, \lambda_0 = 0 \\ \frac{1}{2}Pr(x_i = 1|t_0 = 1) & \lambda_1 = 0, \lambda_0 = 1 \\ \frac{1}{2}Pr(x_i = 0|t_0 = 0) & \lambda_1 = 1, \lambda_0 = 0 \\ \frac{1}{2} + \frac{1}{2}Pr(x_i = 1|t_0 = 0) & \lambda_1 = 1, \lambda_0 = 1 \end{cases} \\ \neq& \frac{1}{2}. \end{aligned} \tag{8}$$

Subcase 2. *The gate g_1 is in $\mathcal{G}_3$.* According to Equation 6, we have

$$\begin{aligned} t_1 = g^3(t_0, x_i, x_j) &= t_0x_ix_j + \lambda_1(t_0x_i + t_0x_j + x_ix_j + t_0 + x_i + x_j) + \lambda_0 \\ &= \begin{cases} t_0x_i + \lambda_1(t_0 + x_i) + \lambda_0 & x_i = x_j \\ \lambda_1 + \lambda_0 & x_i \neq x_j. \end{cases} \end{aligned} \tag{9}$$

Thus

$$\begin{aligned} Pr(t_1 = 0) =& Pr(x_i = x_j) \cdot Pr(t_0x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0) \\ &+ Pr(x_i \neq x_j) \cdot Pr(\lambda_1 + \lambda_0 = 0) \\ =& \begin{cases} \frac{1}{2}Pr(t_0x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0|x_i = x_j) + \frac{1}{2} & \lambda_1 + \lambda_0 = 0 \\ \frac{1}{2}Pr(t_0x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0|x_i = x_j) & \lambda_1 + \lambda_0 \neq 0 \end{cases} \end{aligned} \tag{10}$$

Similar to **Subcase 1**, the value of t_0 is determined by two input variables, thus we can obtain that $0 < Pr(t_0x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0|x_i = x_j) < 1$ whether there is a dependency between t_0 and x_i. Therefore,

$$Pr(t_1 = 0) \neq \frac{1}{2}. \tag{11}$$

By Equation 8 and 11, we have $Pr(t_1 = 0) \neq \frac{1}{2}$ when the gate g_0 is in $\mathcal{G}l$ and g_1 is in $\mathcal{G}_2 \cup \mathcal{G}_3$. Since $f(x_0, \cdots, x_n) = t_1$, f is not a balanced Boolean function when the gate g_0 is in $\mathcal{G}l$ and g_1 is in $\mathcal{G}_2 \cup \mathcal{G}_3$.

Case 3. *The gate g_0 and g_1 are both in $\mathcal{G}_2 \cup \mathcal{G}_3$.* Obviously, by the proof of Property 1, $Pr(t_0 = 0) \neq \frac{1}{2}$. Depending on the number of inputs to g_1, there are two subcases.

Subcase 1. *The gate g_1 is in $\mathcal{G}_2$.* According to Equation 6, we have

$$t_1 = g^2(t_0, x_i) = t_0x_i + \lambda_1(t_0 + x_i) + \lambda_0 = \begin{cases} (\lambda_1 + 1)t_0 + \lambda_1 + \lambda_0 & x_i = 1 \\ \lambda_1 t_0 + \lambda_0 & x_i = 0. \end{cases}$$

Note that, whether there is a dependency between t_0 and x_i, we have $0 < Pr(t_0 = u|x_i =$

$v) < 1$ where $u, v \in 0, 1$. Thus,

$$\begin{aligned} Pr(t_1 = 0) =& Pr(x_i = 1) \cdot Pr((\lambda_1 + 1)t_0 + \lambda_1 + \lambda_0 = 0) \\ &+ Pr(x_i = 0) \cdot Pr(\lambda_1 t_0 + \lambda_0 = 0) \\ =& \begin{cases} \frac{1}{2}Pr(t_0 = 0|x_i = 1) + \frac{1}{2} & \lambda_1 = 0, \lambda_0 = 0 \\ \frac{1}{2}Pr(t_0 = 1|x_i = 1) & \lambda_1 = 0, \lambda_0 = 1 \\ \frac{1}{2}Pr(t_0 = 0|x_i = 0) & \lambda_1 = 1, \lambda_0 = 0 \\ \frac{1}{2} + \frac{1}{2}Pr(t_0 = 1|x_i = 0) & \lambda_1 = 1, \lambda_0 = 1 \end{cases} \\ \neq& \frac{1}{2} \end{aligned} \tag{12}$$

Subcase 2. *The gate g_1 is in $\mathcal{G}_3$.* According to Equation 7, we have

$$\begin{aligned} t_1 = g^3(t_0, x_i, x_j) &= t_0 x_i x_j + \lambda_1(t_0 x_i + t_0 x_j + x_i x_j + t_0 + x_i + x_j) + \lambda_0 \\ &= \begin{cases} t_0 x_i + \lambda_1(t_0 + x_i) + \lambda_0 & x_i = x_j \\ \lambda_1 + \lambda_0 & x_i \neq x_j. \end{cases} \end{aligned} \tag{13}$$

Thus,

$$\begin{aligned} Pr(t_1 = 0) =& Pr(x_i = x_j) \cdot Pr(t_0 x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0) \\ &+ Pr(x_i \neq x_j) \cdot Pr(\lambda_1 + \lambda_0 = 0) \\ =& \begin{cases} \frac{1}{2}Pr(t_0 x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0|x_i = x_j) + \frac{1}{2} & \lambda_1 + \lambda_0 = 0 \\ \frac{1}{2}Pr(t_0 x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0|x_i = x_j) & \lambda_1 + \lambda_0 \neq 0 \end{cases} \end{aligned} \tag{14}$$

About the value of $Pr(t_0 x_i + \lambda_1(t_0 + x_i) + \lambda_0 = 0|x_i = x_j)$, we can obtain that it is greater than 0 and less than 1 by the former case. Therefore,

$$Pr(t_1 = 0) \neq \frac{1}{2}. \tag{15}$$

By Equation 8 and 15, we have $Pr(t_1 = 0) \neq \frac{1}{2}$ when the gate g_0 and g_1 are both in $\mathcal{G}_2 \cup \mathcal{G}_3$. Since $f(x_0, \cdots, x_n) = t_1$, f is not a balanced Boolean function when the gate g_0 and g_1 are both in $\mathcal{G}_2 \cup \mathcal{G}_3$.

Case 4. *The gate g_0 is in $\mathcal{G}_2 \cup \mathcal{G}_3$ and g_1 is in $\mathcal{G}l$.* There does exist some g_0 and g_1 such that f is balanced, e.g.,

$$\begin{aligned} t_0 = g_0(x_i, x_j) = x_i x_j, \qquad t_1 = g_1(t_0, x_k) = t_0 + x_k, \\ f(x_0, \cdots, x_{n-1}) = t_1 = g_1(t_0, x_k) = t_0 + x_k = x_i x_j + x_k, \end{aligned} \tag{16}$$

where $0 \leq i, j, k < n, i \neq j, i \neq k$ and $j \neq k$. Obviously,

$$Pr(f = 0) = Pr(t_1 = 0) = Pr(x_k = 0) \cdot Pr(x_i x_j = 0) + Pr(x_k = 0) \cdot (1 - Pr(x_i x_j = 0)) = \frac{1}{2}.$$

Thus, f is indeed a balanced Boolean function.

In sum, f is balanced Boolean function if and only if g_0 is a nonlinear gate and g_1 is a linear gate. □

From the above properties, according to the cost of logic gates in Table 5, it takes at least 3 GE to achieve a balanced Boolean function with an algebraic degree of at least 2.

We define "continuous composite use of logic gates" as the case that the output of the previous gate is the input of the next gate like a chain. Then we propose the following property.

Property 3. Let $f(x_0, \cdots, x_{n-1})$ is an n-variable Boolean function with algebraic degree $l(l \geq 2)$ obtained by $\widetilde{g}$ and g_1, i.e. $f : V \xrightarrow{\widetilde{g}} t_m \xrightarrow{g_1} t_{m+1} = f(V = \{x_0, \cdots, x_{n-1}\})$, where $\widetilde{g}$ is some continuous composite use of logic gates except $\mathcal{G}_1 \cup \mathcal{G}_4$, g_1 is a nonlinear logic gate with 2 inputs, and the output t_m of $\widetilde{g}$ is an input of g_1. If t_m is the first balanced output among the outputs of the logic gates used in $\widetilde{g}$, then f is not a balanced Boolean function.

Proof. Since t_m reaches an balanced state, we have $Pr(t_m = 0) = \frac{1}{2}$. Since the gate g_1 is in $\mathcal{G}_2$, according to Equation 6, we have

$$t_{m+1} = g^2(t_m, t_s) = t_m t_s + \lambda_1(t_m + t_s) + \lambda_0 = \begin{cases} (\lambda_1 + 1)t_s + \lambda_1 + \lambda_0 & t_m = 1 \\ \lambda_1 t_s + \lambda_0 & t_m = 0. \end{cases}$$

where t_s is one of the outputs of the logic gates used in $\widetilde{g}$ or directly belongs to V. And

$$\begin{aligned} Pr(t_{m+1} = 0) =& Pr(t_m = 1) \cdot Pr((\lambda_1 + 1)t_s + \lambda_1 + \lambda_0 = 0) \\ &+ Pr(t_m = 0) \cdot Pr(\lambda_1 t_s + \lambda_0 = 0) \\ =& \begin{cases} \frac{1}{2}Pr(t_s = 0|t_m = 1) + \frac{1}{2} & \lambda_1 = 0, \lambda_0 = 0 \\ \frac{1}{2}Pr(t_s = 1|t_m = 1) & \lambda_1 = 0, \lambda_0 = 1 \\ \frac{1}{2}Pr(t_s = 0|t_m = 0) & \lambda_1 = 1, \lambda_0 = 0 \\ \frac{1}{2} + \frac{1}{2}Pr(t_s = 1|t_m = 0) & \lambda_1 = 1, \lambda_0 = 1. \end{cases} \end{aligned} \tag{17}$$

When t_s is one of the outputs of the logic gates used in $\widetilde{g}$, $Pr(t_s = 0) \neq \frac{1}{2}$. Since $\widetilde{g}$ is some continuous composite use of logic gates, t_m can be regarded as the output of t_s as an input through the continuous composite use of logic gates. On the other hand,

$$\begin{aligned} Pr(t_m = u) &= Pr(t_s = 0, t_m = u) + Pr(t_s = 1, t_m = u) \\ &= Pr(t_m = u|t_s = 0)Pr(t_s = 0) + Pr(t_m = u|t_s = 1)Pr(t_s = 1) = \frac{1}{2}, u = 0, 1. \end{aligned} \tag{18}$$

If $Pr(t_s = 0, t_m = u) = 0$, then $Pr(t_m = u) = Pr(t_m = u|t_s = 1)Pr(t_s = 1) \leq Pr(t_s = 1) < \frac{1}{2}$, which is in contradiction with $Pr(t_m = 0) = \frac{1}{2}$. Thus, $Pr(t_s = 0, t_m = u) \neq 0$. Similarly, $Pr(t_s = 1, t_m = u) \neq 0$. Therefore, $0 < Pr(t_s = 0, t_m = u) < 1$ and $0 < Pr(t_s = 1, t_m = u) < 1$. Furthermore, $0 < Pr(t_s = u|t_m = v) < 1$ where $u, v \in 0, 1$. According to Equation 17, we have $Pr(t_{m+1}) \neq \frac{1}{2}$. Then we have f is not a balanced function.

When $t_s \in V$, then the case become **Case 2.** in Property 2 and we have f is not a balanced function.

In sum, f is not balanced. □

Application on 4-bit S-boxes. From [LP07], it is known that the optimal 4-bit S-boxes can be derived from 16 representative S-boxes of CCZ equivalence through affine equivalence. By applying affine transformations to the coordinate functions of these 16 optimal S-boxes, it is observed that each coordinate function of an optimal 4-bit S-box must contain quadratic terms. Therefore, it is evident that a structure where g_1, i.e. $f : V \xrightarrow{\widetilde{g}} t_m \xrightarrow{g_1} t_{m+1} = f(V = \{x_0, \cdots, x_{n-1}\})$ (g_1 is a linear component) cannot construct a second balanced Boolean function where t_m is the first balanced Boolean function with algebraic degree $l(l \geq 2)$. This is because either t_{m+1} lacks quadratic terms or $t_m + t_{m+1}$ lacks quadratic terms. Property 1 and 3 shows that the second balanced Boolean function cannot be obtained by adding only one nonlinear gate with 2 inputs. What's more, we found that only adding one nonlinear gate with 3 inputs or only adding two nonlinear gates (with 2 or 3 inputs) cannot reach the second balanced Boolean function when the cost of $\widetilde{g}$ does not exceed 3 GE. And if only using nonlinear gates, two balanced Boolean functions cannot be obtained under the combined gates with a cost not exceeding 5 GE.

Thus, the optimal way to construct a second coordinate function is that if and only if g_0 is a nonlinear gate and g_1 is a linear gate in $f : V \xrightarrow{\widetilde{g}} t_m \xrightarrow{g_1} t_{m+1} \xrightarrow{g_2} t_{m+2} = f(V = \{x_0, \cdots, x_3\})$ where $Pr(t_m = 0) = \frac{1}{2}$ and the cost of $\widetilde{g}$ is 3 GE. From the perspective of area optimization, constructing two balanced Boolean functions requires at least 2 nonlinear gates and 2 linear gates. On this basis, constructing the other two coordinate functions requires at least 2 additional nonlinear gates. Using this combination of gates as a constraint in the model described in Section 4, we found that no such 4-bit S-box exists. Therefore, constructing a 4-bit S-box requires at least one more gate or replacing one of the nonlinear gates with a linear gate. The minimum area for 4 nonlinear gates and 2 linear gates was found to be 8 GE. Adding at least one more gate or replace one nonlinear gate results in a total minimum area of 9 GE. Hence, we roughly tightened the lower bound for the minimum area construct a 4-bit S-box to be 9 GE if only 2-input and 3-input logic gates are used.

4 Improved Automatic Search Model Considering Multiple Cryptographic Properties and Implementation Cost Simultaneously

Using the Prim-Like greedy algorithm, we explore the upper bound of the minimum area for an S-box under specific cryptographic properties, as well as study the lower bound of the area for bijective S-boxes in Section 3. In order to further determine the minimum area for an S-box with given cryptographic properties, we first propose an improved automatic search method to construct S-boxes under certain cryptographic properties in Section 4.1. Then we propose an improved optimization method for S-boxes in Section 4.2. Finally in Section 4.3, we combine these two models above which can take multiple cryptographic properties and area, gate depth into consideration simultaneously.

4.1 Improved Automatic Search Method to Construct S-boxes under Certain Cryptographic Properties

4.1.1 Original model in [LMC+22]

In [LMC+22], Lu et al. constructed the STP-based automatic S-box search by first determining the constraints of the S-boxes cryptographic properties in the CVC language format, including differential uniform $\mathcal{U}(S)$, linearty $\mathcal{L}(S)$, frequency of $\mathcal{U}(S)$ and $\mathcal{L}(S)$, bijectivity, fix point and so on. According to Definition 1 $\sim$ 4, $\mathcal{U}(S)$ and $\mathcal{L}(S)$ in this model are constrained as follows:

$$\begin{aligned} \delta_S(\alpha,\beta) \ &\leq \ \mathcal{U}(S), \\ |\lambda_S(\alpha,\beta) \ - \ 2^{n-1}| \ &\leq \ \mathcal{L}(S). \end{aligned}$$

To obtain $\delta_S(\alpha,\beta)$ and $\lambda_S(\alpha,\beta)$, we need to traverse the constraints across the entire DDT and LAT under (α,β):

$$\text{IsTrue}_{DDT}(\alpha,\beta,x) = \begin{cases} 1, & if\ S(x\oplus\alpha) = S(\alpha)\oplus\beta, \\ 0, & others. \end{cases}$$

$$\text{IsTrue}_{LAT}(\alpha,\beta,x) = \begin{cases} 1, & if\ \alpha\cdot x\oplus\beta\cdot S(x) = 0, \\ 0, & others. \end{cases}$$

Then, $\delta_S(\alpha, \beta)$ and $\lambda_S(\alpha, \beta)$ are constructed as follows:

$$\delta_S(\alpha, \beta) = \sum_{x=0}^{2^n-1} \mathrm{IsTrue}_{DDT}(\alpha, \beta, x),$$

$$\lambda_S(\alpha, \beta) = \sum_{x=0}^{2^n-1} \mathrm{IsTrue}_{LAT}(\alpha, \beta, x).$$

For the constraints of other cryptographic properties of an S-box, please refer to [LMC+22].

4.1.2 Improved model by considering algebraic degree

Lu et al.'s method can consider many cryptographic properties, but unfortunately, it fails to cover the algebraic degree of an S-box which is also a significant cryptographic property. An S-box's algebraic degree can be get from the ANF expression of the S-box's coordinate functions. However, it is difficult to construct ANF in the SAT model. In this part, we give a method to add the constraints about the algebraic degree into the SAT model via SoP expression of Boolean function.

The definition of SoP expression for Boolean function is defined as follows:

Definition 11 (SoP expression for Boolean function [Sas12])**.** Assume $f(x): \mathbb{F}_2^n \to \mathbb{F}_2$ is a Boolean function, then its SoP expression is:

$$f(x) = \bigvee (f(u) \cdot g_u(x)), \tag{19}$$

where $g_u(x)$ is the Minterm Boolean function defined as follows:

$$g_u(x) = \prod_{j=0}^{n-1} (x_j + u_j + 1) = \begin{cases} 1, x = u \\ 0, x \neq u, \end{cases} \tag{20}$$

where $x = (x_0, x_1, \ldots, x_{n-1})$ and $u = (u_0, u_1, \ldots, u_{n-1})$.

From Definition 11, it is easy to find that there are 2^n different Minterm Boolean function $g_u(x)$ according to the value of u. Each Minterm $g_u(x) = 1$ only when $x = u$. This means that only one of 2^n items $f(u) \cdot g_u(x)$ in Equation (19) will be 1 for given x, while other items are all zero. As a result, the SoP expression of $f(x)$ can be directly transformed as

$$f(x) = \sum f(u) \cdot g_u(x). \tag{21}$$

Actually, the SoP expression cannot be directly used to constrain the algebraic degree, we need to convert the SoP expression in Equation (21) into the ANF expression. It is achieved by combining Definition 11 and Equations (20) (21) as follows:

$$\begin{aligned} f(x) &= \bigvee_{u \in \mathbb{F}_2^n} f(u) \cdot g_u(x) \\ &= \sum_{u \in \mathbb{F}_2^n} f(u) \cdot \prod_{j=0}^{n-1} (x_j + u_j + 1). \\ &= \sum_{u \in \mathbb{F}_2^n} (f(u) \cdot \sum_{v \in \mathbb{F}_2^n} ((1+u_0)^{\overline{v_0}} \cdot (1+u_1)^{\overline{v_1}} \cdots (1+u_{n-1}^{\overline{v_{n-1}}})) \cdot x_0^{v_0} x_1^{v_1} \ldots x_{n-1}^{v_{n-1}}) \\ &= \sum_{u \in \mathbb{F}_2^n} (f(u) \cdot \sum_{v \in \mathbb{F}_2^n} \alpha_v^u \cdot x_0^{v_0} x_1^{v_1} \ldots x_{n-1}^{v_{n-1}}) \\ &= (\sum_{u \in \mathbb{F}_2^n} f(u) \cdot \alpha_{2^n-1}^u) \cdot x_0 \cdots x_{n-1} + \cdots + (\sum_{u \in \mathbb{F}_2^n} f(u) \cdot \alpha_1^u) \cdot x_3 + (\sum_{u \in \mathbb{F}_2^n} f(u) \cdot \alpha_0^u), \end{aligned}$$

where $\alpha_v^u = (1+u_0)^{\overline{v_0}} \cdot (1+u_1)^{\overline{v_1}} \cdots (1+u_{n-1}^{\overline{v_{n-1}}})$.

Until now, all coefficients in ANF of $f(x)$, which are $\sum f(u) \cdot \alpha_{2^n-1}^u$, $\sum f(u) \cdot \alpha_{2^n-2}^u$, $\cdots$, $\sum f(u) \cdot \alpha_0^u$, can be expressed into SAT model since all α_v^u ($v \in \mathbb{F}_2^n$) can be pre-computed. If we require the algebraic degree of $f(x)$ at least d, then at least one coefficient $\{wt(v) \geq d \mid \sum f(u) \cdot \alpha_v^u\}$ is not zero. In other word,

$$\begin{cases} \sum_{wt(v)>d}(f(u) \cdot \alpha_v^u) = 0, \\ \sum_{wt(v)=d}(f(u) \cdot \alpha_v^u) \geq 1. \end{cases} \tag{22}$$

Actually an S-box: $\mathbb{F}_2^n \to \mathbb{F}_2^n$ can be regarded as a vectorial Boolean function $(f_0(x), \cdots, f_{n-1}(x))$. To set the algebraic degree of an S-box as d is to set the algebraic degree of at least one coorinate function as d.

Algorithm 2: Improved automatic search model for n-bit S-boxes under certain cryptographic properties.

```
Input: $\mathcal{U}(S)$, $\mathcal{L}(S)$, deg(S) = d, pre-computed $\alpha[\mathbf{u}][\mathbf{v}]$
Output: CVC-based search model.
for $in \leftarrow 0$ to $2^n - 1$ do
    for $out \leftarrow 0$ to $2^n - 1$ do
        for $v \leftarrow 0$ to $2^n - 1$ do
            $ASSERT(IF\ y_0^{v\oplus in}@\cdots@y_{n-1}^{v\oplus in} = y_0^v@\cdots y_{n-1}^v \oplus out\ THEN$
             $\text{IsTrue}_{DDT}(in, out, v) = 1\ ELSE\ \text{IsTrue}_{DDT}(in, out, v) = 0);$
            $ASSERT(IF\ in \cdot v = out \cdot y_0^v@\cdots@y_{n-1}^v\ THEN$
             $\text{IsTrue}_{LAT}(in, out, v) = 1\ ELSE\ \text{IsTrue}_{LAT}(in, out, v) = 0);$
        $ASSERT(\delta_S[in][out] = \sum_{v=0}^{2^n-1} \text{IsTrue}_{DDT}(in, out, v));$
        $ASSERT(\lambda_S[in][out] = \sum_{v=0}^{2^n-1} \text{IsTrue}_{LAT}(in, out, v));$
        $ASSERT(\delta_S[in][out] \leq \mathcal{U}(S));$
        $ASSERT(2^{n-1} - \mathcal{L}(S) \leq \lambda_S[in][out] \leq 2^{n-1} + \mathcal{L}(S));$
for $X \leftarrow 0$ to $2^n - 1$ do
    $x_0@x_1@\cdots@x_{n-1} = X;$
    $ASSERT(X \neq y_0^X@y_1^X@\cdots@y_{n-1}^X);$ // without fixed point
    for $Z \leftarrow X + 1$ to $2^n - 1$ do
        $ASSERT(y_0^X@\cdots@y_{n-1}^X \neq y_0^Z@\cdots@y_{n-1}^Z);$ //bijective S-box

// Constrain the algebraic degree of an S-box to be $d$.
$ASSERT(\sum_{i=0}^{n-1}\sum_{u=0}^{2^n-1}\sum_{wt(v)=d}(y_i^u \cdot \alpha[u][v]) \geq 1);$
for $i \leftarrow 0$ to $n - 1$ do
    $ASSERT(\sum_{u=0}^{2^n-1}\sum_{wt(v)>d}(y_i^u \cdot \alpha[u][v]) = 0);$
```

For example, assume the algebraic degree of one coordinate function f of a 4-bit S-box to be 3. In the ANF of $f(x)$, we only need to focus on the coefficients of the cubic monomials, i.e., the coefficients of $x_0x_1x_2$, $x_0x_1x_3$, $x_0x_2x_3$, $x_1x_2x_3$, which are $\sum_{u\in\mathbb{F}_2^n} f(u)\alpha_{14}^u$, $\sum_{u\in\mathbb{F}_2^n} f(u)\alpha_{13}^u$, $\sum_{u\in\mathbb{F}_2^n} f(u)\alpha_{11}^u$ and $\sum_{u\in\mathbb{F}_2^n} f(u)\alpha_7^u$ respectively. Since for an n-bit Boolean function, each ANF expression of $g_u(x)$ can be computed. By expanding all $g_u(x)$, we obtain all of α_{14}^u, α_{13}^u, α_{11}^u, α_7^u, which are shown in Table 8.

Finally, we constrain the algebraic degree of one coordinate function of an S-box to be 3, that is:

$$\sum_{u\in\mathbb{F}_2^n} f(u)\alpha_{14}^u + \sum_{u\in\mathbb{F}_2^n} f(u)\alpha_{13}^u + \sum_{u\in\mathbb{F}_2^n} f(u)\alpha_{11}^u + \sum_{u\in\mathbb{F}_2^n} f(u)\alpha_7^u \geq 1. \tag{23}$$

Table 8: Coefficient of each cubic monomial in the ANF expansion of $g_u(x)$ for a 4-bit Boolean function.

u of $g_u(x)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
α_{14}^u	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
α_{13}^u	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
α_{11}^u	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
α_{7}^u	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Combining the new findings on algebraic degree and original model in [LMC⁺22], we propose an improved automatic search model based on SAT, which is shown in Algorithm 2.

4.2 Improved Optimization Implementation Method for Given S-boxes

4.2.1 Original model in [LWH⁺21]

The model proposed in [LWH⁺21], built upon the Gate Count Complexity model from [Sto16], optimizes the area cost of an S-box by assigning a weight to each gate corresponding to its area. This approach enables to calculate the minimum area for a given S-box and support 4-input gates. Assume there are k gates in total, the expressions for four possible inputs of the i-th ($i = 0, 1, \ldots, k-1$) gate can be written as:

$$\begin{cases} q_{4i} &= a_0^i \cdot x_0 + \cdots + a_{n-1}^i \cdot x_{n-1} + a_n^i \cdot t_0 + \cdots + a_{n+i-1}^i \cdot t_{i-1}, \\ q_{4i+1} &= b_0^i \cdot x_0 + \cdots + b_{n-1}^i \cdot x_{n-1} + b_n^i \cdot t_0 + \cdots + b_{n+i-1}^i \cdot t_{i-1}, \\ q_{4i+2} &= c_0^i \cdot x_0 + \cdots + c_{n-1}^i \cdot x_{n-1} + c_n^i \cdot t_0 + \cdots + c_{n+i-1}^i \cdot t_{i-1}, \\ q_{4i+3} &= d_0^i \cdot x_0 + \cdots + d_{n-1}^i \cdot x_{n-1} + d_n^i \cdot t_0 + \cdots + d_{n+i-1}^i \cdot t_{i-1}. \end{cases} \tag{24}$$

where x_i ($i = 0, 1, \ldots, n-1$) denotes the i-th input bit of S-box, and t_i ($i = 0, 1, \ldots, k-1$) denotes the output bit of the i-th gate.

Each input bit of the i-th gate comes either from one input bit of S-box or one output bit from any preceding gate. Thus, in each expression for $q_{4i}, q_{4i+1}, q_{4i+2}, q_{4i+3}$, only one coefficient is 1, other coefficients are all 0, i.e. that is

$$\begin{cases} \sum_{j=0}^{n+i-1} a_j^i &= 1, \\ \sum_{j=0}^{n+i-1} b_j^i &= 1, \\ \sum_{j=0}^{n+i-1} c_j^i &= 1, \\ \sum_{j=0}^{n+i-1} d_j^i &= 1. \end{cases} \tag{25}$$

Within the search model, we can collect constraints between the inputs and output of every logic gate under $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$ using the following formula:

$$\begin{aligned} t_i =& \beta_0^i \cdot q_{4i} \cdot q_{4i+1} \cdot q_{4i+2} \cdot q_{4i+3} + \beta_0^i \cdot q_{4i} \cdot q_{4i+1} \cdot q_{4i+2} + \beta_0^i \cdot q_{4i} \cdot q_{4i+1} \cdot q_{4i+3} + \\ & \beta_0^i \cdot q_{4i+2} \cdot q_{4i+3} + \beta_0^i \cdot q_{4i+2} + \beta_0^i \cdot q_{4i+3} + \beta_1^i \cdot q_{4i} \cdot q_{4i+1} \cdot q_{4i+2} + \\ & \beta_2^i \cdot q_{4i} \cdot q_{4i+1} + \beta_2^i \cdot q_{4i} \cdot q_{4i+2} + \beta_2^i \cdot q_{4i+1} \cdot q_{4i+2} + \beta_2^i \cdot q_{4i} + \beta_2^i \cdot q_{4i+1} + \\ & \beta_2^i \cdot q_{4i+3} + \beta_3^i \cdot q_{4i} + \beta_3^i \cdot q_{4i+1} + \beta_3^i \cdot q_{4i+2} + \beta_4^i \cdot q_{4i} \cdot q_{4i+1} + \\ & \beta_5^i \cdot q_{4i} + \beta_5^i \cdot q_{4i+1} + \beta_6^i \cdot q_{4i} + \beta_7^i, \end{aligned} \tag{26}$$

where the coefficients $(\beta_0^i, \beta_0^i, \ldots, \beta_7^i)$ determine the type of the i-th gate and its area. The corresponding relationship between $(\beta_0^i, \beta_0^i, \ldots, \beta_7^i)$ and logic gates is shown in Table 9.

In the end, each of n output bits y_i ($i = 0, 1, \ldots, n-1$) of S-box also comes either from one input bit of S-box or one output bit from any preceding gate. Thus, the expression for

Table 9: Corresponding relationship between the values of $(\beta_0^i, \beta_1^i, \ldots, \beta_7^i)$ and logic gates in [LWH$^+$21].

Logic Gates	β_0^i	β_1^i	β_2^i	β_3^i	β_4^i	β_5^i	β_6^i	β_7^i	Cost/GE
NOT	0	0	0	0	0	0	1	1	0.67
XOR	0	0	0	0	0	1	0	0	3.00
XNOR	0	0	0	0	0	1	0	1	3.00
NOT	0	0	0	0	0	1	1	1	0.67
AND	0	0	0	0	1	0	0	0	1.33
NAND	0	0	0	0	1	0	0	1	1.00
OR	0	0	0	0	1	1	0	0	1.33
NOR	0	0	0	0	1	1	0	1	1.00
AND3	0	1	0	0	0	0	0	0	1.33
NAND3	0	1	0	0	0	0	0	1	1.67
OR3	0	1	1	0	0	0	0	0	1.33
NOR3	0	1	1	0	0	0	0	1	1.67
XOR3	0	0	0	1	0	0	0	0	4.67
XNOR3	0	0	0	1	0	0	0	1	4.67
MAOI1	1	0	0	0	0	0	0	0	2.67
MOAI1	1	0	0	0	0	0	0	1	2.00

each y_i can be written as:

$$\begin{cases} y_0 & = a_0^{k,0} \cdot x_0 + \cdots + a_{n-1}^{k,0} \cdot x_{n-1} + a_n^{k,0} \cdot t_0 + \cdots + a_{n+k-1}^{k,0} \cdot t_{k-1}, \\ y_1 & = a_0^{k,1} \cdot x_0 + \cdots + a_{n-1}^{k,1} \cdot x_{n-1} + a_n^{k,1} \cdot t_0 + \cdots + a_{n+k-1}^{k,1} \cdot t_{k-1}, \\ \vdots & \quad \vdots \\ y_{n-1} & = a_0^{k,n-1} \cdot x_0 + \cdots + a_{n-1}^{k,n-1} \cdot x_{n-1} + a_n^{k,n-1} \cdot t_0 + \cdots + a_{n+k-1}^{k,n-1} \cdot t_{k-1}. \end{cases} \tag{27}$$

Similar to add constraints on coefficients as that in Equation (25), here we can describe the constraints on output bits as well.

Additionally, the constraints on area should be added into the model. since all logic gates are multiples of 0.33 GE, the proportion of area for different logical gates is used to measure the area in our model for the sake of simplicity and convenience. For example, the area of AND in the model is set to be 4, since its actual area is 4 times of 0.33GE. By summing up the area of all k logic gates and ensuring it is no more than the target area $Cost_{target}$, we can find the optimized implementation for a given S-box under such target area. Furthermore, by gradually reducing the target area, we can find the minimum-area implementation for a given S-box. The corresponding constraints in the model are as follows.

$$ASSERT(\ Cost = BVPLUS(g[\beta_0^0@\beta_1^0@...@\beta_7^0], \ldots, g[\beta_0^{k-1}@\beta_1^{k-1}@...@\beta_7^{k-1}]));$$
$$ASSERT(\ BVLE(\ Cost, Cost_{target})\ );$$

where $g([\beta_0^i@\beta_1^i@...@\beta_7^i])$ is a pre-computed array that stores the area of each logic gate according to the value of $(\beta_0^i, \beta_1^i, \ldots, \beta_7^i)$.

4.2.2 Improved model on reducing the search space

Most standard cell libraries include 2-input gates in $\mathcal{G}_2$ and 3-input gates in $\mathcal{G}_3$. The logical representation of these gates are shown in Table 3. Via observing the logic gates, we find that the output of each 2-input/3-input gates cannot be affected by the order of inputs which is summarized in Observation 1.

Observation 1. *For any n-input gate $g \in \mathcal{G}_n$ $(n = 2, 3)$ and $g(x_0, \ldots, x_{n-1}) : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we have*

$$g(x_0, \ldots, x_{n-1}) = g(P(x_0, \ldots, x_{n-1})),$$

where $P(x_0, \ldots, x_{n-1})$ denotes any simple permutation on $\mathbb{F}_2^n$.

With Observation 1, it is able to add constraints on the order of inputs for each 2-input/3-input logic gate in the model as follows:

$$\begin{cases} a_0^i || a_1^i || \cdots || a_{n+i-1}^i & > \ b_0^i || b_1^i || \cdots || b_{n+i-1}^i \\ b_0^i || b_1^i || \cdots || b_{n+i-1}^i & > \ c_0^i || c_1^i || \cdots || c_{n+i-1}^i \end{cases}.$$

By such constraints, the search space can be reduced at least 50% on inputs of every gate. Since MAOI1 and MOAI1 gates in $\mathcal{G}_4$ do not comply with Observation 1, these two gates must be restricted in the SAT model when applying this acceleration technique.

Besides that, since two S-boxes within the same bit permutation equivalent class share the same cryptographic properties and implementation area, it is no need to distinguish the order of output bits when searching a new S-box. Similar to shrink the search space on each logic gate, we can add constraints on the order of $(y_0, y_1, \ldots, y_{n-1})$ to further reduce the search space just as follows :

$$\begin{cases} a_0^{k,0} || a_1^{k,0} || \cdots || a_{n+k-1}^{k,0} & > \ a_0^{k,1} || a_1^{k,1} || \cdots || a_{n+k-1}^{k,1}, \\ a_0^{k,1} || a_1^{k,1} || \cdots || a_{n+k-1}^{k,1} & > \ a_0^{k,2} || a_1^{k,2} || \cdots || a_{n+k-1}^{k,2}, \\ & \vdots \\ a_0^{k,n-2} || a_1^{k,n-2} || \cdots || a_{n+k-1}^{k,n-2} & > \ a_0^{k,n-1} || a_1^{k,n-1} || \cdots || a_{n+k-1}^{k,n-1}. \end{cases}$$

By adding these $n-1$ constraints, we need only search through one of the $n!$ possible output orders. As a result, the search space is reduced to $\frac{1}{n!}$ of the original search space.

4.2.3 Improved model by constraining on gate depth complexity

The automatic model in [LWH+21] can only be used to optimize the area of a given S-box, but it cannot be used to optimize the gate depth of a given S-box. In this part, we propose a technique to take the gate depth into our model.

To describe the gate depth in the model, we set new variables D_i in the model. Firstly, we set $D_i, 0 \le i < n$ to represent the depth of the i-th input bit position of S-box. Of course, we have

$$D_0 = D_1 = \cdots = D_{n-1} = 0.$$

Secondly, we set $D_{n+i}, 0 \le i$ to represent the depth of the i-th gate used in the model. According to Equation (24) and (25), each input bit of the i-th gate comes from one input bit of S-box or one output bit of preceding gates. That means the depth of the i-th gate increases by 1 compared to the largest depth of related preceding gates or input bit positions.

For 2-input gates, we have

$$\begin{cases} 0 < \beta_0^i @ \beta_1^i @ ... @ \beta_7^i \le 0b00001101, \\ D_{n+i} = \max\{\sum_{j=0}^{n+i-1} a_j^i \cdot D_j, \ \sum_{j=0}^{n+i-1} b_j^i \cdot D_j\} + 1. \end{cases} \tag{28}$$

For 3-input gates, we have

$$\begin{cases} 0b00001101 < \beta_0^i @ \beta_1^i @ ... @ \beta_7^i \le 0b01100001, \\ D_{n+i} = \max\{\sum_{j=0}^{n+i-1} a_j^i \cdot D_j, \ \sum_{j=0}^{n+i-1} b_j^i \cdot D_j, \ \sum_{j=0}^{n+i-1} c_j^i \cdot D_j\} + 1. \end{cases} \tag{29}$$

For 4-input gates, we have

$$\begin{cases} 0b10000000 \leq \beta_0^i @ \beta_1^i @ ... @ \beta_7^i, \\ D_{n+i} = \max\{\sum_{j=0}^{n+i-1} a_j^i \cdot D_j,\ \sum_{j=0}^{n+i-1} b_j^i \cdot D_j,\ \sum_{j=0}^{n+i-1} c_j^i \cdot D_j,\ \sum_{j=0}^{n+i-1} d_j^i \cdot D_j\} + 1. \end{cases} \tag{30}$$

These three cases above could be involved into the model with if statement.

Combining the techniques on shrink of the search space in Section 4.2.2 and description of gate depth in Section 4.2.3, as well as the original model in [LMC+22], we propose an improved automatic optimization model based on SAT, which is shown in Algorithm 3.

This algorithm can optimize the area and gate depth of a given S-box, when the number of required gates k and implementation area $Cost_{target}$ are fixed. At the very beginning, we need to pre-store all ordered combination of gates satisfying k gates and $Cost_{target}$ area into the array $\mathcal{P}[]$ and pre-store the area of each logic gate according to the value of $\beta_0^i @ \beta_1^i @ ... @ \beta_7^i$ in $g[]$. By running codes corresponding to Algorithm 3, we can get a CVC file. With the help of solver such as STP, an optimized implementation result could be solved out, or no solution. As we indeed do not know how many gates and what is the minimum area for a given S-box, we have to first try all possible cases of area from more to less, then try all possible cases of #gates under each fixed area from more to less.

4.3 Improved Automatic Method to Search New S-boxes

In this part, thanks to the improved automatic search method in Section 4.1 and improved optimization implementation method in Section 4.2, we propose an automatic method to search new S-boxes which can consider the cryptographic properties, implementation area and gate depth simultaneously.

In detail, by combining Algorithm 2 and Algorithm 3, it is able to search optimization implementation when the number of gates and total area are given. As we indeed do not know how many gates and what is the required area to construct minimum-area S-boxes, we have to first try all possible cases of area from more to less, then try all possible cases of #gates under each fixed area from more to less. The workload of traversal is enormous. To solve this problem, we can use the work in Section 3 to obtain the tight upper bound and lower bound of minimum area to construct S-boxes at the very beginning. At last, by utilizing the constraints on gate depth proposed in Section 4.2.3, it is able to optimize the gate depth under the minimum area.

5 Results

5.1 Results on the Minimum Area Cost of 4-bit S-boxes

In this section, we apply the tight bounds on minimum area of S-boxes in Section 3 and the automatic search method in Section 4 to seek the minimum area of 4-bit S-boxes under three different basis: $\{\mathcal{G}_1, \mathcal{G}_2\}$, $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$, and $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$. In detail, three types of S-boxes are taken into consideration, including the optimal 4-bit S-boxes, the optimal 4-bit S-boxes with algebraic degree of 3 for all coordinate functions, as well as the general 4-bit S-boxes with algebraic degree of at least 2 for all coordinate functions. Generally, the algebraic degree of every coordinate function of an S-box should not be 1, or it will lead to nonzero linear approximation pattern with probability 1. Thus, we do not care the S-boxes with algebraic degree of 1 for some coordinate functions.

Minimum area cost of the optimal 4-bit S-boxes. Firstly, we seek the minimum area of optimal 4-bit S-boxes under UMC 180nm library. As a result, we find out that the minimum area of optimal 4-bit S-boxes under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$ and $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$ is 12 GE with fixed point as well as 12.33 GE without fixed point while the minimum gate

Algorithm 3: Improved optimization model for a given S-box under given number of gates and area.

```
Input: an $n$-bit S-box, number of gates $k$, target area cost $Cost_{target}$,
       pre-computed array $g[]$, pre-computed array $\mathcal{P}[]$.
Output: A CVC-based model.
for $X \leftarrow 0$ to $2^n - 1$ do
    $x_0^X @ x_1^X @ \cdots @ x_{n-1}^X = X$;
    for $i \leftarrow 0$ to $k-1$ do
        $ASSERT(q_{4i}^X = \sum_{j=0}^{n-1} a_j^i \cdot x_j^X + \sum_{j=0}^{i-1} a_{n+j}^i \cdot t_j^X)$;
        $ASSERT(q_{4i+1}^X = \sum_{j=0}^{n-1} b_j^i \cdot x_j^X + \sum_{j=0}^{i-1} b_{n+j}^i \cdot t_j^X)$;
        $ASSERT(q_{4i+2}^X = \sum_{j=0}^{n-1} c_j^i \cdot x_j^X + \sum_{j=0}^{i-1} c_{n+j}^i \cdot t_j^X)$;
        $ASSERT(q_{4i+3}^X = \sum_{j=0}^{n-1} d_j^i \cdot x_j^X + \sum_{j=0}^{i-1} d_{n+j}^i \cdot t_j^X)$;
        $ASSERT(t_j^X = \beta_0^i \cdot q_{4i}^X \cdot q_{4i+1}^X \cdot q_{4i+2}^X \cdot q_{4i+3}^X + \cdots + \beta_7)$; // by Equation (26)
    for $i \leftarrow 0$ to $n-1$ do
        $ASSERT(y_i^X = \sum_{j=0}^{n-1} a_j^{k,i} \cdot x_j^X + \sum_{j=n}^{n+k-1} a_j^{k,i} \cdot t_j^X)$;
for $i \leftarrow 0$ to $k-1$ do
    $ASSERT(\sum_{t=0}^{n+i-1} a_t^i = 1)$;
    $ASSERT(\sum_{t=0}^{n+i-1} b_t^i = 1)$;
    $ASSERT(\sum_{t=0}^{n+i-1} c_t^i = 1)$;
    $ASSERT(\sum_{t=0}^{n+i-1} d_t^i = 1)$;
for $i \leftarrow 0$ to $n-1$ do
    $ASSERT(\sum_{t=0}^{n+k-1} a_t^{k,i} = 1)$;
for $i \leftarrow 0$ to $k-1$ do
    $ASSERT(a_0^i || a_1^i || \cdots || a_{n+i-1}^i > b_0^i || b_1^i || \cdots || b_{n+i-1}^i)$;
    $ASSERT(b_0^i || b_1^i || \cdots || b_{n+i-1}^i > c_0^i || c_1^i || \cdots || c_{n+i-1}^i)$;
for $i \leftarrow 0$ to $n-2$ do
    $ASSERT(a_0^{k,i} || a_1^{k,i} || \cdots || a_{n+k-1}^{k,i} > a_0^{k,i+1} || a_1^{k,i+1} || \cdots || a_{n+k-1}^{k,i+1})$;
for $i \leftarrow 0$ to $n-1$ do
    $ASSERT(D_i = 0)$;
for $i \leftarrow 0$ to $k-1$ do
    $ASSERT(D_{n+i} = 1 +$
    $\max\{\sum_{j=0}^{n+i-1} a_j^i \cdot D_j, \sum_{j=0}^{n+i-1} b_j^i \cdot D_j, \sum_{j=0}^{n+i-1} c_j^i \cdot D_j, \sum_{j=0}^{n+i-1} d_j^i \cdot D_j\})$;
for $i \leftarrow 0$ to $k-1$ do
    $\mathcal{P}[i] = \beta_0^i @ \beta_1^i @ \ldots @ \beta_7^i$    // $[\beta_0^i @ \beta_1^i @ \ldots @ \beta_7^i]$ records the types of
     gates from the 0-th gate to the $(k-1)$-th gate.
$ASSERT(Cost = \sum_{i=0}^{k-1} g[\beta_0^i @ \beta_1^i @ \ldots @ \beta_7^i])$;
$ASSERT(Cost = Cost_{target})$;
```

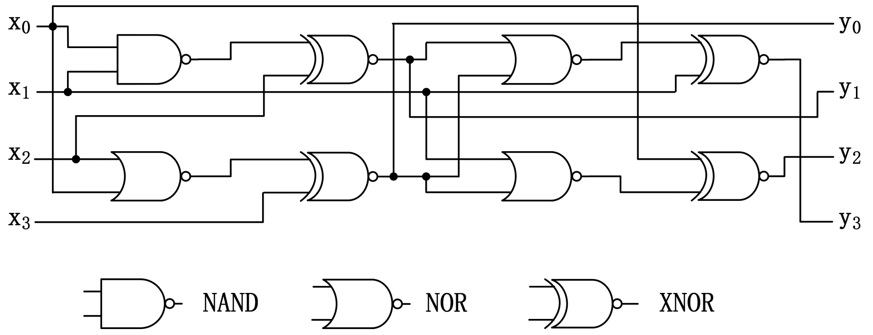


Figure 2: Implementation circuit of an optimal 4-bit S-box with minimum area of 12 GE and gate depth of 4 under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$

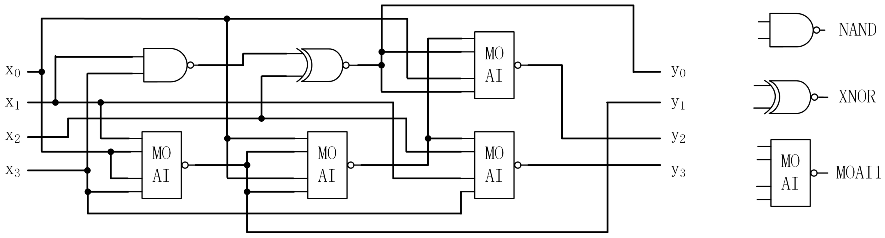


Figure 3: Implementation of the optimal 4-bit S-box whose minimum area is 11 GE and gate depth is 3 under the basis $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$.

depth under 12 GE is 4. By analyzing, all these optimal S-boxes with minimum area need at least 8 gates, including 4 NAND/NOR gates and 4 XNOR gates. Totally, we find out over 20000 optimal S-boxes with 12 GE, which can be further divided into some classes according to bit permutation equivalence. An instance of the optimal S-boxes with 12 GE is shown as follows and its implementation circuit is shown in Figure 2.

$$S(x) = \{0x0, 0xb, 0xf, 0x5, 0x3, 0xa, 0xe, 0x6, 0x9, 0x2, 0xd, 0x7, 0xc, 0x4, 0x8, 0x1\}$$

Secondly, we seek the minimum area of optimal 4-bit S-boxes under the basis $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$. By using the Prim-like greedy algorithm in Section 3.1, we find a tight upper bound on the minimum area of optimal S-boxes as 11 GE. After that, we use the automatic search model in Section 4.3 to search optimal S-boxes under the constraint of area no more than 11 GE. In the end, we find that the minimum area of optimal 4-bit S-boxes is exactly 11 GE including 6 gates while the minimum depth under the S-box of 11 GE is 3. Surprisingly, the founded S-box has no fixed point. An instance of the optimal S-boxes with 11 GE is shown as follows and its implementation circuit is shown in Figure 3.

$$S(x) = \{0x7, 0x2, 0xd, 0xb, 0x6, 0xa, 0xc, 0x3, 0x1, 0x0, 0x9, 0x8, 0x4, 0xe, 0xf, 0x5\}$$

Minimum area cost of the optimal 4-bit S-boxes with algebraic degree of 3 for all coordinate functions. According to Definition 8, the algebraic degree of optimal 4-bit S-boxes is exactly 3. It means the algebraic degree of at least one coordinate function for a given S-box should be 3. In other words, the algebraic degree of some coordinate functions can be 2. In this part, we care the optimal 4-bit S-boxes with algebraic degree of 3 for all coordinate functions. Intuitively, this type of S-boxes has better cryptographic properties, but needs more area cost. We search the minimum area for this type of S-boxes under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$ and $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$, respectively. As a result, we find out that the minimum area of such type of S-boxes is 14 GE with fixed point as well as 14.33 GE without

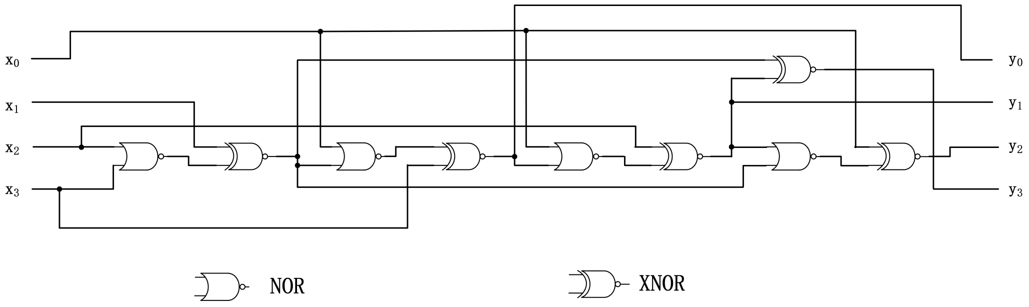


Figure 4: Implementation circuit of an optimal 4-bit S-box with algebraic degree of 3 for all coordinate functions under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$. (Its area is 14 GE and gate depth is 8.)

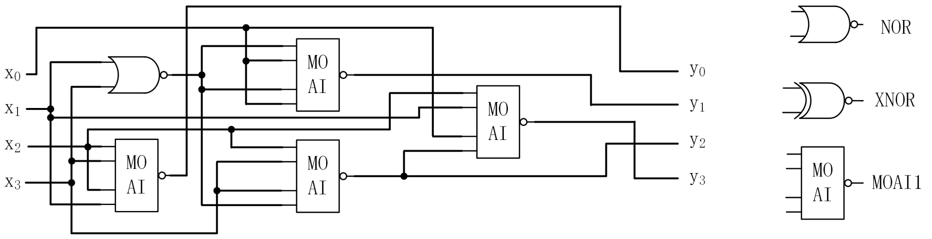


Figure 5: Implementation circuit of a general 4-bit S-box with minimum area of 9 GE and gate depth of 3 under the basis $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$.

fixed point while the minimum gate depth under 14 GE or 14.33 GE is 8. An example of the optimal 4-bit S-box with 14 GE is shown as follows, while its implementation circuit is shown in Figure 4.

$$S(x) = \{0x1, 0x2, 0xa, 0x7, 0xc, 0x5, 0x8, 0x0, 0xf, 0xe, 0x6, 0x9, 0xd, 0x4, 0xb, 0x3\}$$

Unfortunately, it is unable to determine the minimum area of the optimal S-boxes with algebraic degree of 3 for all coordinate functions under the basis $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$, because the search space is too large to solve out in time.

Minimum area cost of the general S-boxes with algebraic degree of at least 2 for all coordinate functions. In this part, we expand the search scope to general S-boxes with algebraic degree of at least 2 for all coordinate functions, i.e., $\mathcal{U}(S) < 16$ and $\mathcal{L}(S) < 8$. It means no obvious weaknesses on differential uniformity and linearity. Intuitively, this type of S-boxes has worse cryptographic properties, intuitively one might assume that they require less area cost than the optimal 4-bit S-boxes. As a result, we find out that the minimum area of this type of S-boxes is also 12 GE with fixed point and 12.33 GE without fixed point under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$ and $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$, which are the same as the minimum area of optimal 4-bit S-boxes. Besides, when 4-input gates are involved, we find out the minimum area is 9 GE, which is shown in Figure 5.

5.2 Improved Optimized Implementation on KECCAK and SKINNY-128's S-boxes

In this part, we propose the improved optimized implementation on 5-bit S-box used in KECCAK cipher and 8-bit S-box used in SKINNY cipher by using the improved automatic optimization method in Section 4.2. The two S-boxes are provided in Appendix A.

Until now, the best previous optimized implementation of KECCAK's S-box was given by Lu et al. in [LWH+21]. It needs 17.66 GE. In this paper, we find out the minimum

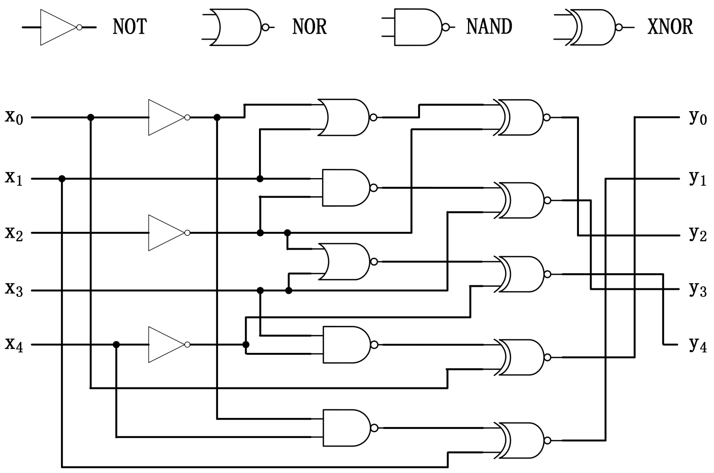


Figure 6: Optimized implementation circuit of KECCAK's S-box with minimum area of 17 GE.

area cost of KECCAK's S-box is exactly 17 GE in terms of the theoretical GE of the UMC 180nm library. Figure 6 shows the optimal implementation circuit of KECCAK's S-box with minimum area of 17 GE.

By using our improved optimization method, we propose the optimized implementation of SKINNY-128's S-box for the first time. It needs 26.67 GE, which is smaller than 29.33 GE calculated according to the circuit in its design document. Figure 7 shows the optimized implementation circuit of SKINNY's S-box with area of 26.67 GE.

Remark. There are mainly four methods to optimize the implementation area of a given S-box or search new S-box with smaller area, including Lighter in [JPST17], Stoffelen's method in [Sto16], Lu et al.'s method in [LWH$^+$21] and our method in this paper. The first three methods can only be used to optimize the area of a given S-box, they cannot be used to find new S-boxes with small area. In detail, Lighter and Stoffelen's method can only optimize the S-boxes with size of no more than 4 bits, while Lu et al.'s method can roughly optimize 5-bit S-box such as Keccak's S-box. Our method in this paper cannot only be used to search new S-boxes with small area, but also can be used to optimize large S-boxes, even 8-bit ones. Because of the acceleration techniques in Section 4.2, the search space is significantly reduced so that our method can achieve a more accurate optimization for large S-boxes. Thus, we search the optimization implementation of KECCAK's S-box better than Lu et al.'s result, while propose the first optimized implementation result for SKINNY-128's S-box. Since the search complexity increases as the number of logic gates grows, our method has been limited to optimize the lightweight large S-boxes. Besides KECCAK's and SKINNY-128's S-boxes, we also optimize other lightweight S-boxes such as the one used in Ascon cipher, but no better results are found.

5.3 Discussion

Extension to different cell libraries. Though the main results of this paper are obtained by using the GE as the standard unit under the UMC 180nm standard cell library to measure and compare the areas of S-boxes, our tool can be easily generalized to other libraries. With different technological libraries, we need to re-calculate the unit of GEs and model the corresponding GE values of different gates. The remaining model keep unchanged. Indeed, the cost GEs of one S-box might differ under different cell libraries. We

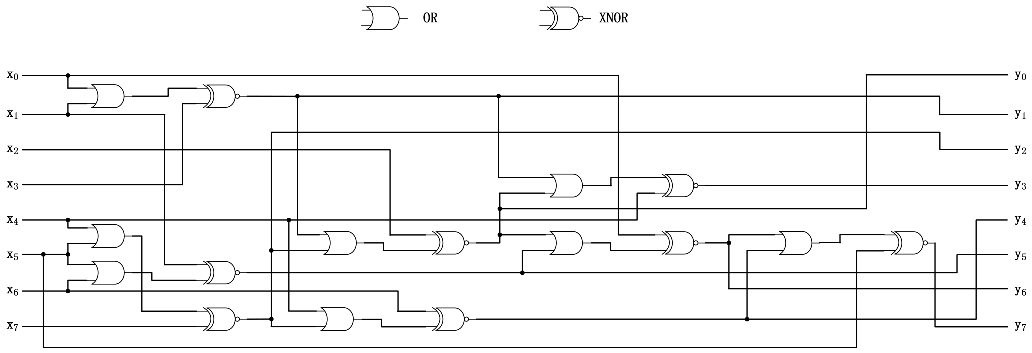


Figure 7: Optimized implementation circuit of SKINNY-128's S-box with the area of 26.67 GE.

conduct the experiments under the TSMC 65nm Library. The minimum area of optimal 4-bit S-boxes is 14 GE under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$ or $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$, while 13.5 GE under the basis $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4\}$. The minimum area of optimal 4-bit S-boxes with algebraic degree of 3 for all coordinate functions is 16.5 GE under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$ or $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$. The minimum area of optimal 4-bit S-boxes with algebraic degree of at least 2 for all coordinate functions is 14 GE under the basis $\{\mathcal{G}_1, \mathcal{G}_2\}$ or $\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$. Although the minimum area of an S-box varies under different libraries, the minimum-area S-boxes under the UMC 180nm library are the minimum-area ones under the TSMC 65nm library as well. Note that there are more minimum-area S-boxes under the TSMC 65nm library, because the area of both MAOI1 and MOAI1 gates are identical in this library. In a short, under any certain cell library, our model can solve out the minimum area of 4-bit optimal S-boxes and obtain their logic circuit implementation.

Extension to other "special gates". Our method is generic that can represent all the gates listed by Lighter. Equation (26) provides a general formula for the relationship between the inputs and outputs of all the logic gates within this model under the UMC 180nm Library. In this formula, $(\beta_0, \beta_1, \ldots, \beta_7)$ controls the type of logic gate, which is provided in Table 9. and the cascaded values of β_i contains some redundancy. Thus, the formula for the input-output relationship of logic gates provided in this paper can represent more gates, such as the ANDN and ORN gates from the TSMC 65nm library, which can be represented as $\beta_0 @ \beta_1 @ \cdots @ \beta_7 = 0b00001010$ and $\beta_0 @ \beta_1 @ \cdots @ \beta_7 = 0b00001011$ respectively. What's more, for more "special gates" which may existed in other cell libraries, users can introduce new variables $\beta_i (i > 7)$to involve them in the model.

Practical synthesis results. We synthesized the current S-boxes used in PICCOLO, SKINNY, RECTANGLE, LBLOCK and KECCAK under the Nangate 45nm library and the TSMC 65nm library respectively. Besides, we synthesized the optimal 4-bit S-box obtained in this paper, as shown in Figure 3. We compared three synthesis methods in total. The first method is based on the lookup table (LUT), where the mapping table of the S-box will be automatically optimized into a circuit implementation with a relatively small area by synthesizers. The second method is using the optimized tool in Lighter [JPST17], which is currently one of the most effective tools for area-optimized synthesis of small S-boxes. We use the optimized S-box implementations achieved by [JPST17] to synthesise again both in TSMC 65nm library and Nangate 45nm library. The third method is based on the improved automatic search model in this paper. We used the model in Section 4 to obtain an optimized implementation of the S-boxes and, where possible, achieve the optimal implementation. The resulting logic circuit was synthesized and compared with the previous two methods. The synthesis results are summarized in Table 10.

Table 10: The synthesis results of different S-boxes under the Nangate 45nm library and TSMC 65nm library. The columns of “LUT” contains the synthesis areas of LUTs for different libraries. The columns of “Light” means the circuits that are synthesized are from the Light tool. The “Synthesis” and “Theory” in the “This paper” columns mean the circuits are from our SAT tool, where the area values of “Theory” is calculated by us according to the numbers of gates in the corresponding libraries.

S-box	Nangate 45nm library				TSMC 65nm library			
	LUT	LIGHTER	This paper		LUT	LIGHTER	This paper	
			Synthesis	Theory			Synthesis	Theory
PICCOLO	$13.832\mu m^2$	$9.576\mu m^2$	$9.576\mu m^2$	–	$35.04\mu m^2$	$26.88\mu m^2$	$24.96\mu m^2$	–
	17.33GE	12.00GE	12.00GE	12.00GE	18.25GE	14.00GE	13.00GE	14.00GE
SKINNY-64	$14.364\mu m^2$	$9.576\mu m^2$	$9.576\mu m^2$	–	$36.48\mu m^2$	$24.96\mu m^2$	$24.96\mu m^2$	–
	18.00GE	12.00GE	12.00GE	12.00GE	19.00GE	13.00GE	13.00GE	14.00GE
RECTANGLE	$18.088\mu m^2$	$15.428\mu m^2$	$14.896\mu m^2$	–	$48.00\mu m^2$	$36.48\mu m^2$	$36.00\mu m^2$	–
	22.67GE	19.33GE	18.67GE	18.00GE	25.00GE	19.00GE	18.75GE	21.50GE
LBLOCK S_0	$15.162\mu m^2$	$12.768\mu m^2$	$12.768\mu m^2$	–	$36.96\mu m^2$	$34.08\mu m^2$	$31.20\mu m^2$	–
	19.00GE	16.00GE	16.00GE	16.33GE	19.25GE	17.50GE	16.25GE	19.50GE
KECCAK	$13.832\mu m^2$	–	$13.566\mu m^2$	–	$35.04\mu m^2$	–	$35.52\mu m^2$	–
	17.33GE	–	17.00GE	17.00GE	18.25GE	–	18.50GE	19.00GE
SKINNY-128	$74.746\mu m^2$	–	$18.088\mu m^2$	–	$180.48\mu m^2$	–	$50.88\mu m^2$	–
	93.67GE	–	22.67GE	24.00GE	94.00GE	–	26.50GE	28.00GE
S-box in Fig.3	–	–	–	–	$29.76\mu m^2$	–	$22.08\mu m^2$	–
	–	–	–	–	15.50GE	–	11.50GE	13.50GE

From Table 10, we can see that the synthesis results obtained by Lighter and our method are both better than those obtained using the LUT-based method. A more detailed comparison shows that for the current 4-bit S-boxes, the synthesis area obtained by our method is no larger than the results achieved by Lighter. Moreover, our method yields better results than Lighter for some cases, such as the result of RECTANGLE’s S-box in both libraries, and results for the S-boxes of PICCOLO and LBlock S_0 under the TSMC 65nm library. Additionally, we synthesized the optimized circuits of the KECCAK’s S-box and SKINNY-128’s S-box. Notably, the synthesis implementation of the SKINNY-128’s S-box obtained using our method shows significant improvement compared to the circuit optimized by LUT-based method. Finally, we synthesized the optimal 4-bit S-box shown in Figure 3. In the TSMC 65nm library, which provides MOAI1 and MAOI1 gates, we achieved a circuit implementation area of 11.5 GE. This is smaller than the area of the existing lightweight optimal 4-bit S-boxes.

6 Conclusion

In this paper, we propose an improved automatic search method to obtain the minimum area of optimal 4-bit S-boxes under certain technological library. Besides that, we study the minimum area of other 4-bit S-boxes with differential cryptographic properties, as well as propose the optimizing implementation of the existing S-boxes such as KECCAK′s 5-bit S-box and SKINNY′s 8-bit S-box with smaller area. To measure and compare the area cost, we use the gate equivalent (GE) as standard unit under UMC 180 nm library which has 2/3/4-input logic gates. Of course, our automatic search method is also effective under other technique libraries and can be used to find better optimizing implementation of other existing S-boxes. However, this method can only determine the minimum area of 4-bit S-boxes and cannot optimize the implementation of complex 8-bit S-boxes yet. In the future work, it may be possible to significantly shrink the search space via some techniques based on graph theory and cryptographic theory.

Acknowledgments

This research is supported by the National Key R&D Program of China (Grant No. 2024YFA1013000, 2023YFA1009500), the National Natural Science Foundation of China (Grant No. 62032014, U2336207, 62402283). Tingting Cui is specially supported by the Open Project Program from Key Laboratory of Cryptologic Technology and Information Security (Ministry of Education), Shandong University. Kai Hu is also supported by the Natural Science Foundation of Jiangsu Province (BK20240420) and Program of Qilu Young Scholars of Shandong University.

References

[BAK98] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In Serge Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. 537:2–21, 1990.

[BS93] Eli Biham and Adi Shamir. Differential cryptanalysis of the data encryption standard. Springer, 1993.

[CCH10] Claude Carlet, Yves Crama, and Peter L. Hammer. Boolean functions for cryptography and error-correcting codes. pages 257–397, 2010.

[CDL15] Anne Canteaut, Sébastien Duval, and Gaëtan Leurent. Construction of lightweight s-boxes using feistel and MISTY structures. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 373–393. Springer, 2015.

[CP02] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer, 2009.

[GSP+18] Ashrujit Ghoshal, Rajat Sadhukhan, Sikhar Patranabis, Nilanjan Datta, Stjepan Picek, and Debdeep Mukhopadhyay. Lightweight and side-channel secure 4x4 s-boxes from cellular automata rules. *IACR Cryptol. ePrint Arch.*, page 832, 2018.

[JPST17] Jérémy Jean, Thomas Peyrin, Siang Meng Sim, and Jade Tourteaux. Optimizing implementations of lightweight building blocks. *IACR Trans. Symmetric Cryptol.*, 2017(4):130–168, 2017.

[LMC+22] Zhenyu Lu, Sihem Mesnager, Tingting Cui, Yanhong Fan, and Meiqin Wang. An stp-based model toward designing s-boxes with good cryptographic properties. *Des. Codes Cryptogr.*, 90(5):1179–1202, 2022.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite fields*. Number 20. Cambridge university press, 1997.

[LP07] Gregor Leander and Axel Poschmann. On the classification of 4 bit s-boxes. In Claude Carlet and Berk Sunar, editors, *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings*, volume 4547 of *Lecture Notes in Computer Science*, pages 159–176. Springer, 2007.

[LW14] Yongqiang Li and Mingsheng Wang. Constructing s-boxes for lightweight cryptography with feistel structure. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 2014.

[LWH+21] Zhenyu Lu, Weijia Wang, Kai Hu, Yanhong Fan, Lixuan Wu, and Meiqin Wang. Pushing the limits: Searching for implementations with the smallest area for lightweight s-boxes. In Avishek Adhikari, Ralf Küsters, and Bart Preneel, editors, *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*, volume 13143 of *Lecture Notes in Computer Science*, pages 159–178. Springer, 2021.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[MPLJ19] Luca Mariot, Stjepan Picek, Alberto Leporati, and Domagoj Jakobovic. Cellular automata based s-boxes. *Cryptogr. Commun.*, 11(1):41–62, 2019.

[Nyb93] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 55–64. Springer, 1993.

[Ras22] Shahram Rasoolzadeh. Low-latency boolean functions and bijective s-boxes. *IACR Trans. Symmetric Cryptol.*, 2022(3):403–447, 2022.

[Saa11] Markku-Juhani O. Saarinen. Cryptographic analysis of all 4 × 4-bit s-boxes. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 118–133. Springer, 2011.

[Sas12] Tsutomu Sasao. *Switching theory for logic synthesis*. Springer Science & Business Media, 2012.

[Sto16] Ko Stoffelen. Optimizing s-box implementations for several criteria using SAT solvers. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 140–160. Springer, 2016.

[TLZ23] Shizhu Tian, Yitong Liu, and Xiangyong Zeng. A further study on bridge structures and constructing bijective s-boxes for low-latency masking. *Des. Codes Cryptogr.*, 91(11):3709–3739, 2023.

[ZBL+15] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang, and Ingrid Verbauwhede. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Sci. China Inf. Sci.*, 58(12):1–15, 2015.

[ZBRL15] Wentao Zhang, Zhenzhen Bao, Vincent Rijmen, and Meicheng Liu. A new classification of 4-bit optimal s-boxes and its application to present, RECTANGLE and SPONGENT. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 494–515. Springer, 2015.

A Appendix

The S-boxes used in KECCAK and SKINNY-128 are shown as follows:

$$\begin{aligned}S_{\text{KECCAK}}(x) =\{&\texttt{0x00, 0x09, 0x12, 0x0b, 0x05, 0x0c, 0x16, 0x0f,}\\&\texttt{0x0a, 0x03, 0x18, 0x01, 0x0d, 0x04, 0x1e, 0x07,}\\&\texttt{0x14, 0x15, 0x06, 0x17, 0x11, 0x10, 0x02, 0x13,}\\&\texttt{0x1a, 0x1b, 0x08, 0x19, 0x1d, 0x1c, 0x0e, 0x1f }\}\end{aligned}$$

$S_{\text{SKINNY}}(x) = \{$

```
0x65,0x4c,0x6a,0x42,0x4b,0x63,0x43,0x6b,0x55,0x75,0x5a,0x7a,0x53,0x73,0x5b,0x7b,
0x35,0x8c,0x3a,0x81,0x89,0x33,0x80,0x3b,0x95,0x25,0x98,0x2a,0x90,0x23,0x99,0x2b,
0xe5,0xcc,0xe8,0xc1,0xc9,0xe0,0xc0,0xe9,0xd5,0xf5,0xd8,0xf8,0xd0,0xf0,0xd9,0xf9,
0xa5,0x1c,0xa8,0x12,0x1b,0xa0,0x13,0xa9,0x05,0xb5,0x0a,0xb8,0x03,0xb0,0x0b,0xb9,
0x32,0x88,0x3c,0x85,0x8d,0x34,0x84,0x3d,0x91,0x22,0x9c,0x2c,0x94,0x24,0x9d,0x2d,
0x62,0x4a,0x6c,0x45,0x4d,0x64,0x44,0x6d,0x52,0x72,0x5c,0x7c,0x54,0x74,0x5d,0x7d,
0xa1,0x1a,0xac,0x15,0x1d,0xa4,0x14,0xad,0x02,0xb1,0x0c,0xbc,0x04,0xb4,0x0d,0xbd,
0xe1,0xc8,0xec,0xc5,0xcd,0xe4,0xc4,0xed,0xd1,0xf1,0xdc,0xfc,0xd4,0xf4,0xdd,0xfd,
0x36,0x8e,0x38,0x82,0x8b,0x30,0x83,0x39,0x96,0x26,0x9a,0x28,0x93,0x20,0x9b,0x29,
0x66,0x4e,0x68,0x41,0x49,0x60,0x40,0x69,0x56,0x76,0x58,0x78,0x50,0x70,0x59,0x79,
0xa6,0x1e,0xaa,0x11,0x19,0xa3,0x10,0xab,0x06,0xb6,0x08,0xba,0x00,0xb3,0x09,0xbb,
0xe6,0xce,0xea,0xc2,0xcb,0xe3,0xc3,0xeb,0xd6,0xf6,0xda,0xfa,0xd3,0xf3,0xdb,0xfb,
0x31,0x8a,0x3e,0x86,0x8f,0x37,0x87,0x3f,0x92,0x21,0x9e,0x2e,0x97,0x27,0x9f,0x2f,
0x61,0x48,0x6e,0x46,0x4f,0x67,0x47,0x6f,0x51,0x71,0x5e,0x7e,0x57,0x77,0x5f,0x7f,
0xa2,0x18,0xae,0x16,0x1f,0xa7,0x17,0xaf,0x01,0xb2,0x0e,0xbe,0x07,0xb7,0x0f,0xbf,
0xe2,0xca,0xee,0xc6,0xcf,0xe7,0xc7,0xef,0xd2,0xf2,0xde,0xfe,0xd7,0xf7,0xdf,0xff}.
```