Malaysian Journal of Mathematical Sciences 19(3): 933–959 (2025)
https://doi.org/10.47836/mjms.19.3.09

Malaysian Journal of Mathematical Sciences

Journal homepage: https://mjms.upm.edu.my

Improved Recursive Construction of S–box Satisfying Perfect Strict Avalanche Criterion

Pang, K.* [1], Abdul-Latip, S. F.* [2,3], Jamil, N. [4], and Abdul Rani, H.[5]

[1]*Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal 76100, Melaka, Malaysia*
[2]*Faculty of Artificial Intelligence and Cyber Security, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, Durian Tunggal 76100, Melaka, Malaysia*
[3]*Symmetric Division, Malaysia Cryptology Technology and Management Centre, c/o Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia*
[4]*College of IT, United Arab Emirates University, Al Ain, 15551, United Arab Emirates*
[5]*Cryptography Development Department, CyberSecurity Malaysia, Menara Cyber Axis, Cyberjaya, 63000, Selangor, Malaysia*

E-mail: pang.kok.an@utem.edu.my
shekhfaisal@utem.edu.my
**Corresponding author*

Received: 19 November 2024
Accepted: 24 March 2025

Abstract

A secure S–box must exhibit the Strict Avalanche Criterion (SAC), ensuring that a single–bit change in the input causes each output bit to change with a probability close to 50%, thereby complicating differential and linear cryptanalysis. Despite its importance, analyses of deployed ciphers suggest that current S–box designs can still be improved. In this paper, we introduce a novel recursive method for constructing S–boxes that achieve perfect SAC by leveraging smaller S–boxes with SAC when combined with bitwise rotations. This approach systematically generates larger S–boxes while preserving and enhancing the cryptographic strengths of their smaller counterparts. The resulting S–boxes not only meet perfect SAC but also demonstrate competitive security properties and can be implemented using simple logic circuits, making them especially suitable for resource-constrained environments. Our findings contribute significantly to S–box design and offer substantial implications for developing robust cryptographic systems.

Keywords: 5–bit S–box; perfect SAC; recursive S–box construction; security; cryptography.

1 Introduction

In the digital age, the importance of cryptography cannot be exaggerated. As we increasingly rely on electronic systems to store and transmit sensitive information, the need for robust security measures becomes paramount. Cryptography plays a crucial role in ensuring the confidentiality, integrity, and authenticity of digital data. The advent of the internet and the proliferation of digital devices have exponentially increased the volume of data generated and shared every day. This data, which often includes sensitive personal and financial information, is a prime target for malicious actors. Cryptography provides the necessary tools to protect this data, ensuring that it can be securely stored and transmitted. Moreover, in an era where privacy concerns are increasingly coming to the fore, cryptography offers a means to preserve individual privacy in online communications. Through the use of encryption, users can communicate securely, safe in the knowledge that their messages can only be read by the intended recipients. Cryptography also underpins the security of numerous modern technologies and systems. From online banking and e-commerce to secure email and blockchain technology, cryptography is a fundamental component that ensures the security and reliability of these systems.

In the realm of cryptography, few components are as critical and intriguing as cryptographic S–boxes, or substitution boxes. These non-linear transformation functions form the backbone of many symmetric key algorithms, playing a pivotal role in ensuring the security of our digital communications. S–boxes serve as the primary source of confusion in a cipher, transforming input bits into output bits in a manner that the relationship between secret keys and output bits is obscured. This non-linearity is crucial in thwarting linear and differential cryptanalysis attacks, thereby bolstering the security of the cryptographic system. The importance of S–boxes extends beyond their role in individual ciphers. They are integral to the broader landscape of digital security, underpinning everything from secure online transactions to confidential communications. In an era where data breaches and cyber threats are increasingly prevalent, the role of S–boxes in safeguarding sensitive information is more important than ever. However, the design and implementation of S–boxes are not without challenges. The quest for S–boxes that provide optimal resistance against known cryptanalytic attacks, while remaining efficient for hardware and software implementations, is an ongoing area of research.

The field of cryptography has witnessed significant advancements and diversification in recent years, with the study of S–boxes remaining a critical focus. However, most publications have concentrated on the construction of 4–bit and 8–bit S–boxes. For instance, Isa et al. [22] introduced an S–box constructed by applying a binomial operation to two power functions defined over the finite field $\mathbb{F}_{2^8}$. Rashidi [40] proposes two 4–bit S–boxes S_1 and S_2 and two 8–bit S–boxes SB_1 and SB_2 based on S_1 and S_2. The approach of using smaller S–boxes to build larger ones does not provide a general method for generating S–boxes of sizes other than 8–bit. In addition, 8–bit S–boxes [39, 38] involving composite multiplicative inverses in $GF\left(2^4\right)$ have also been proposed; these methods do not clarify how the methodology involving multiplicative inverses in $GF\left(2^4\right)$ can be adapted to generate S–boxes of sizes other than 8–bit. Furthermore, S–boxes with information redundancy [41] for detecting and preventing fault injections have been proposed and applied to 4–bit S–boxes, such as those in PRESENT [14], PRINCE [15] and SPONGENT [13]. The applicability of this masking technique to S–boxes beyond those used in PRESENT, PRINCE and SPONGENT or to S–boxes of other sizes remains uncertain.

The emphasis on 5–bit S–boxes arises from their increasing importance in modern cryptographic systems. In 2012, a significant milestone was reached with the selection of Keccak [11] as the new Secure Hash Algorithm 3 (SHA-3) standard [34]. More recently, the National Institute of Standards and Technology (NIST) selected Ascon [18], a cipher that employs a 5–bit S–box, as a

new standard for lightweight cryptography [35]. The advent of SHA-3 and the selection of the Ascon cipher underscore the importance of studying 5–bit S–boxes. Moreover, candidates from NIST competitions such as PRIMATEs [6], ICEPOLE [33] and SHAMASH [36] also adopt 5–bit S–boxes. In fact, S–boxes with an odd size and high differential uniformity, including 5–bit S–boxes, can be constructed using Almost Perfect Nonlinear (APN) functions, which are predominantly defined in odd dimensions [9]. These developments indicate that 5–bit S–boxes could play a pivotal role in the future of cryptography.

Modern cryptographic systems employ Shannon's principles of confusion and diffusion to ensure the security and confidentiality of data transmission. Confusion obscures the relationship between the secret key and the ciphertext, while diffusion ensures that a single bit change in the input results in an approximately 50% change in the ciphertext. The confusion layer is a critical component in providing nonlinearity in symmetric ciphers. This is achieved through the use of a nonlinear substitution function within the cipher.

A secure S–box should exhibit a strict avalanche criterion (SAC), a widely used measure for assessing the security of S–boxes. This criterion indicates that even a small change in a single input bit should propagate rapidly through the cipher, resulting in a high degree of randomness in the output. The SAC is evaluated by altering a single input bit and observing its effect on the output bits. Ideally, half of the output bits should change, yielding an optimal SAC value of 0.5. An S–box with an SAC value closer to 0.5 is deemed stronger. Weak SAC values render a cipher vulnerable to various cryptanalytic attacks. In contrast, stronger SAC values enhance the diffusion effect of a cipher. Analyses of Ascon, Keccak, PRIMATEs, and SHAMASH reveal SAC values of absolute 0 or 1, significantly deviating from the ideal SAC value of 0.5.

Increasing the number of rounds can enhance the security of a cipher. The issues caused by weak SAC values, such as lower diffusion, can be mitigated by increasing the number of rounds. For instance, in the case of TinyJAMBU, the author increased the number of initialization rounds from 384 to 640 to improve diffusion [43]. However, increasing the number of rounds results in higher latency and resource overhead. Achieving good SAC values is a method for addressing low diffusion, an area where ASCON, Keccak, and PRIMATEs show room for improvement.

Substitution layers, which serve as confusion layers for various cryptographic primitives, can be generated in multiple ways. One such method is the use of finite field arithmetic, as demonstrated in AES [10] and Camellia [7]. These substitution layers exhibit high nonlinearity. However, as shown in Table 1, the substitution layers generated through finite field arithmetic do not guarantee perfection in terms of Strict Avalanche Criterion (SAC).

Table 1: SAC values of different S–boxes available in the literature.

S–box	Minimum SAC value	Maximum SAC value	Average	References
AES	0.4531	0.5625	0.5049	[17]
Camellia $S1$	0.4531	0.5469	0.4983	[7]
Camellia $S2$	0.4531	0.5469	0.4983	[7]
Camellia $S3$	0.4531	0.5469	0.4983	[7]
Camellia $S4$	0.4531	0.5469	0.4983	[7]

In addition to finite field arithmetic, there are numerous studies that have suggested the generation of S–boxes using chaotic functions. For example, Rincu and Iana proposed an S–box design

combining Logistic map, Tent map and Piece-Wise Linear Chaotic Map [42]. Zhou et al. [51] proposed a novel S–box using chaotic sequences generated from a new two-dimensional discrete hyperchaotic map. Ali and Ali [5] suggested the generation of a new S–box using a piecewise linear chaotic map, while Liu et al. [27] proposed an S–box generation algorithm using a non-degeneracy discrete chaotic system. However, finite precision effects [49], dynamical degradation of chaotic systems [50], non-uniform distribution [1], and discontinuity in chaotic sequences [4] can compromise the chaotic properties of these S–boxes.

Several approaches have been proposed to enhance the SAC of S–boxes. Mohd Esa et al. [32] introduced an alternative primitive polynomial that yields an improved AES S–box with better SAC. However, the empirical nature of the primitive polynomial selection process leaves the relationship between primitive polynomials and resulting S–box SAC still unexplored. Li et al. [24] presented a 4–bit S–box that achieves perfect SAC. Unfortunately, similar to Mohd Esa et al.'s work, the underlying rationale for their design remains undisclosed, preventing its direct application in creating other S–boxes with perfect SAC.

A straightforward approach to generating a secure S–box involves generating all possible S–boxes and selecting the one with the most exceptional characteristics. However, this method is impractical as it would require $2^n!$ steps of computation. To circumvent the exhaustive search for a perfect S–box, Kim et al. proposed a recursive technique that constructs n–bit S–boxes with perfect SAC properties from antecedent $(n-1)$–bit S–boxes, which also exhibit perfect SAC. Despite this smart avoidance of the exhaustive $2^n!$ search, its implementation in logic gate circuits necessitates constructing the n–bit S–box from two identical $(n-1)$–bit S–boxes but fed with two distinct inputs, resulting in significant area overhead reserved for these $(n-1)$–bit S–boxes. We will demonstrate how the area can be reduced by rotating $(n-1)$–bit S–boxes that require only one input. Further details will be shown in the paper.

In this paper, we propose an S–box with perfect SAC values, constructed using an enhanced recursive method. Specifically, for a 5–bit S–box, our method enables the generation of 5–bit S–boxes with perfect SAC values in $2^{37.75}$ steps, a significant reduction from the $2^5! \approx 2^{117.7}$ steps required by exhaustive methods. Our approach facilitates the production of S–boxes with superior nonlinearity and iterative periods. Furthermore, these S–boxes can be implemented using simpler logic gate circuits compared to the work by Kim et al. [23]. This research contributes to the field by offering an efficient and effective method for S–box construction.

The following notations are used in this paper;

$P(X)$	:	Probability of an event X.
$+$	:	Addition over $GF(2)$.
Sn	:	An n–bit S–box.
Sn_a	:	The Boolean function of a-th bit of Sn
$Sn(\boldsymbol{x})$	:	The value of n–bit S–box, Sn, with input vector $\boldsymbol{x}$.
Sn^*	:	An incomplete Sn with one remaining unassigned output bit.
$HW(\boldsymbol{x})$	:	The Hamming weight of vector $\boldsymbol{x}$.
x_i	:	The i-th bit of $\boldsymbol{x}$ starting from the least significant bit.

2 Preliminaries

The Strict Avalanche Criterion (SAC) ensures that a small change in the input, even as minimal as flipping a single bit, should cause drastic changes in the output. This property is crucial for creating confusion in the ciphertext, thereby enhancing the security of the cryptographic system. In more precise terms, an S–box satisfies the SAC if, for every output bit, a change in each input bit affects each output bit with a probability of 0.5. This means that each output bit should be a complex and non-linear function of the input bits, making it computationally difficult for an attacker to predict the output based on the input, or vice versa. In the field of cryptography, it is generally accepted that an S–box exhibiting a SAC value approximating 0.5 is indicative of its robustness.

Avalanche property of Sn is measured by calculating the probability of change of $Sn_\imath$ for $0 \leq \imath < n$, given an input change $\Delta\boldsymbol{x}$ where $HW\left(\Delta\boldsymbol{x}\right) = 1$. The ideal value for SAC is 0.5, meaning that $P\left(Sn_\imath\left(\boldsymbol{x}\right) + Sn_\imath\left(\boldsymbol{x} + \Delta\boldsymbol{x}\right) = 1\right) = 0.5$. The SAC of $Sn_\imath$ can be calculated using (1),

$$\delta\left(Sn_\imath, \boldsymbol{x}, \Delta\boldsymbol{x}\right) = \frac{\#\{\boldsymbol{x} \in \mathbb{F}_2^n | Sn_\imath\left(\boldsymbol{x}\right) + Sn_\imath\left(\boldsymbol{x} + \Delta\boldsymbol{x}\right) = 1\}}{2^n}, \tag{1}$$

where $\delta\left(Sn_\imath, \boldsymbol{x}, \Delta\boldsymbol{x}\right)$ denotes $P\left(Sn_\imath\left(\boldsymbol{x}\right) + Sn_\imath\left(\boldsymbol{x} + \Delta\boldsymbol{x}\right) = 1\right)$. In the remaining sections of this paper, we denote $\lambda\left(f, g, \boldsymbol{x}\right)$ as $P\left(f\left(\boldsymbol{x}\right) + g\left(\boldsymbol{x}\right) = 1\right)$ for any functions f and g. An S–box that exhibits perfect SAC must satisfy both Propositions 2.1 and 2.2.

Proposition 2.1. *For $HW\left(\Delta\boldsymbol{x}\right) = 1$, it is required that the probability $\delta\left(Sn_\imath, \boldsymbol{x}, \Delta\boldsymbol{x}\right) > 0$. In this case, every variable $\{x_0, x_1, \ldots, x_{\imath-1}\}$ must exist in all terms of $Sn_\imath$ with nonzero coefficients.*

Proof. If $x_\jmath \in \{x_0, x_1, \ldots, x_{\imath-1}\}$ is not a term or a subterm of any monomials within $Sn_\imath$, then $Sn_\imath\left(\boldsymbol{x}\right) + Sn_\imath\left(\boldsymbol{x} + \Delta\boldsymbol{x}\right) = 0$, thus $\delta\left(Sn_\imath, \boldsymbol{x}, \Delta\boldsymbol{x}\right) = 0$. □

In order for perfect SAC to be satisfied, it is necessary that $\delta\left(Sn_\imath, \boldsymbol{x}, \Delta\boldsymbol{x}\right) = 0.5$. This implies that all input variables must be present as a term or in any subterms of all $Sn_\imath$ with nonzero coefficients.

Proposition 2.2. *For any $x_\jmath \in \{x_0, x_1, \ldots, x_{\jmath-1}\}$ that only exists as a nonzero linear term in $Sn_\imath(\boldsymbol{x})$, flipping the value of $x_\jmath$ will flip the right-hand-side of $Sn_\imath(\boldsymbol{x})$ with a probability of 1.*

Proof. The behavior of XOR operations within a Boolean equation implies that flipping any term will result in a corresponding change in the value of $Sn_\imath\left(\boldsymbol{x}\right)$. □

It is important to note that an algebraic term exhibiting an absolute linear relationship with the Boolean function will invariably induce changes with a probability of 1 when the input is altered. As such, it is imperative that the Boolean functions of S–boxes are designed to avoid this property.

In addition to Propositions 2.1 and 2.2, an n–bit bijective S–box must have a maximum degree of $n - 1$ [20]. As a result, the degree of all terms in $Sn_\imath\left(\boldsymbol{x}\right)$ must be bounded by $n - 1$.

3 Construction of S–box

We construct S–boxes by adopting an approach similar to the recursive method proposed by Kim et al. [23]. Kim's method relies on using smaller S–boxes to build larger ones. For an n–bit S–box S_n that is constructed based on an m–bit S–box with $m = n - 1$, the following holds,

$$\begin{aligned} Sn\left(\boldsymbol{x}\right) &= Sm\left(\boldsymbol{x}\right), \\ Sn\left(\boldsymbol{x} + 2^m\right) &= Sm\left(\boldsymbol{x} + \Delta\boldsymbol{x}\right). \end{aligned} \tag{2}$$

As shown in (2), Kim's method necessitates the use of both $Sm\left(\boldsymbol{x}\right)$ and $Sm\left(\boldsymbol{x} + \Delta\boldsymbol{x}\right)$ for the generation of Sn. This implementation, however, may complicate the circuit, as it must incorporate both $Sm\left(\boldsymbol{x}\right)$ and $Sm\left(\boldsymbol{x} + \Delta\boldsymbol{x}\right)$.

In this paper, we improve Kim's method by generating Sn solely from $Sm\left(\boldsymbol{x}\right)$, making the implementation simpler and easier. We discuss the rationale in detail in Section 6.

Similar to Kim's method, our construction of a 5–bit S–box is initiated by first constructing a 3–bit S–box $S3$ which satisfies a perfect strict avalanche criterion. We formulated simple degree-2 near-bent polynomials [47] for a 3–bit S–box with input variables x_a, x_b, and x_c for three output bits, as illustrated in (3),

$$\begin{aligned} S3_0 &= x_a x_b + x_a x_c + x_b, \\ S3_1 &= x_a x_b + x_b x_c + x_c + 1, \\ S3_2 &= x_a x_c + x_b x_c + x_a. \end{aligned} \tag{3}$$

Based on the values of $S3$, we create a 4–bit S–box, $S4^*$. For any random integer $\mathfrak{v}$, the assignment of $S4^*\left(\boldsymbol{x}\right)$ is shown in (4),

$$S4^*\left(\boldsymbol{x}\right) = \begin{cases} S3\left(\boldsymbol{x}\right), & \text{for } \boldsymbol{x} < 2^3, \\ S3\left(\boldsymbol{x} + 2^3\right) \lll \mathfrak{v}, & \text{for } \boldsymbol{x} \geq 2^3. \end{cases} \tag{4}$$

The construction of $S4^*$ based on (4) is not completed yet, as there is still an unassigned output bit $S4^*_3 \in \{\alpha_0, \alpha_1, \ldots, \alpha_{15}\}$ as shown in Table 2. To complete the 2^4 output values of $S4^*$, we need to determine the new output bit of $S4^*$ to form perfect SAC for $S4$.

In order to determine the values of $S4_3$, which, when combined with the values generated from (4), we search for all 16 candidate values, i.e., $(\alpha_0, \alpha_1, \ldots, \alpha_{15})$, as shown in Table 2. The number of computations required is $\binom{2^n}{2^{n-1}}$, since, in a bijective S–box Sn, both $\#\{Sn_\imath\left(\boldsymbol{x}\right) | Sn_\imath\left(\boldsymbol{x}\right) = 0\}$ and $\#\{Sn_\imath\left(\boldsymbol{x}\right) | Sn_\imath\left(\boldsymbol{x}\right) = 1\}$ are equal to 2^{n-1} for all $0 \leq \imath < n$. In the case of computing the values of $S4_3$, $\binom{16}{8} \approx 2^{13.65}$ is required. Therefore, generating all 4–bit S–boxes from all 3! possible $S3$ with 3 possible rotations $\mathfrak{v}$, the total required computation of it is $\binom{16}{8} \cdot 3! \cdot 3 = 2^{17.82}$. Algorithm 1 shows the generation of 4–bit S–boxes.

Table 2: Values of $S4$ derived from $S3$.

$\boldsymbol{x}$	$S4\,(\boldsymbol{x})$	$S4_3$	$S4_2$	$S4_1$	$S4_0$
0	$S3\,(0)$	α_0	$S3_2\,(0)$	$S3_1\,(0)$	$S3_0\,(0)$
1	$S3\,(1)$	α_1	$S3_2\,(1)$	$S3_1\,(1)$	$S3_0\,(1)$
2	$S3\,(2)$	α_2	$S3_2\,(2)$	$S3_1\,(2)$	$S3_0\,(2)$
3	$S3\,(3)$	α_3	$S3_2\,(3)$	$S3_1\,(3)$	$S3_0\,(3)$
4	$S3\,(4)$	α_4	$S3_2\,(4)$	$S3_1\,(4)$	$S3_0\,(4)$
5	$S3\,(5)$	α_5	$S3_2\,(5)$	$S3_1\,(5)$	$S3_0\,(5)$
6	$S3\,(6)$	α_6	$S3_2\,(6)$	$S3_1\,(6)$	$S3_0\,(6)$
7	$S3\,(7)$	α_7	$S3_2\,(7)$	$S3_1\,(7)$	$S3_0\,(7)$
8	$S3\,(0) <<< \wp$	α_8	$S3_{2-\wp}\,(0)$	$S3_{1-\wp}\,(0)$	$S3_{-\wp}\,(0)$
9	$S3\,(1) <<< \wp$	α_9	$S3_{2-\wp}\,(1)$	$S3_{1-\wp}\,(1)$	$S3_{-\wp}\,(1)$
10	$S3\,(2) <<< \wp$	α_{10}	$S3_{2-\wp}\,(2)$	$S3_{1-\wp}\,(2)$	$S3_{-\wp}\,(2)$
11	$S3\,(3) <<< \wp$	α_{11}	$S3_{2-\wp}\,(3)$	$S3_{1-\wp}\,(3)$	$S3_{-\wp}\,(3)$
12	$S3\,(4) <<< \wp$	α_{12}	$S3_{2-\wp}\,(4)$	$S3_{1-\wp}\,(4)$	$S3_{-\wp}\,(4)$
13	$S3\,(5) <<< \wp$	α_{13}	$S3_{2-\wp}\,(5)$	$S3_{1-\wp}\,(5)$	$S3_{-\wp}\,(5)$
14	$S3\,(6) <<< \wp$	α_{14}	$S3_{2-\wp}\,(6)$	$S3_{1-\wp}\,(6)$	$S3_{-\wp}\,(6)$
15	$S3\,(7) <<< \wp$	α_{15}	$S3_{2-\wp}\,(7)$	$S3_{1-\wp}\,(7)$	$S3_{-\wp}\,(7)$

Algorithm 1 The algorithm for generating 4–bit S–boxes with perfect SAC

procedure GENERATES4(An empty group of S–boxes, $\mathbb{S}4$)
  for all possible 3–bit S–box $S3$ based on (3) **do**
    for all $0 \leq \wp < 3$ **do**
      Initialize an empty $S4$.
      for all $\boldsymbol{x}$ from 0 to 7 **do**
        $S4\,(\boldsymbol{x}) \leftarrow S3\,(\boldsymbol{x})$
        $S4\,(\boldsymbol{x}+8) \leftarrow S3\,(\boldsymbol{x}) <<< \wp$
      end for
      for all l from 0 to 2^{16} such that $HW(l) = 8$ **do**
        for all j from 0 to 15 **do**
          Assign $S4_3\,(j)$ as $\frac{l}{2^j}\ mod\ 2$
        end for
        if S4 fulfills bijectivity and perfect SAC **then**
          $\mathbb{S}4 \cup \{S4\}$
        end if
      end for
    end for
  end for
  return $\mathbb{S}4$
end procedure

Generating a 5–bit S–box, $S5$, fulfiling perfect SAC is also done in a similar fashion. For $0 \leq j < 32$, $S5^*\,(j)$ can be assigned with $S4\,(j)$, while $S5^*\,(16+j)$ is assigned with $S4\,(j)$ rotated by another

random integer, w, as shown in (5),

$$S5^{*}\left(\boldsymbol{x}\right)=\begin{cases}S4\left(\boldsymbol{x}\right), & \text{for } \boldsymbol{x}<2^{4},\\ S4\left(\boldsymbol{x}+2^{4}\right)\ <<<\ w, & \text{for } \boldsymbol{x}\geq 2^{4}.\end{cases} \tag{5}$$

The new output bit, $S5_4^{*}\in\{\beta_0,\beta_1,\ldots,\beta_{31}\}$, can be found in $\binom{32}{16}\approx 2^{29.16}$ steps. With h number of $S4$s generated from (4), where each $S4$ can produce four $S5^{*}$s with all four possible rotations w, the total number of steps required is $\binom{16}{8}\cdot 3!\cdot 3+\binom{32}{16}\cdot 4h$. Algorithm 2 shows the generation of $S5$s. Table 3 shows the new bits of $S5$s.

Algorithm 2 The algorithm for generating 5–bit S–boxes with perfect SAC

Initialize an empty list $\mathbb{S}4$.
$\mathbb{S}4 \leftarrow$ GenerateS4($\mathbb{S}4$).
$\mathbb{S}4 \leftarrow$ MinimizeS4($\mathbb{S}4$, t).
Create an empty list $\mathbb{S}5$.
for all $S4\in\mathbb{S}4$ **do**
  for all $0\leq w<4$ **do**
    Initialize an empty $S5$.
    for all $\boldsymbol{x}$ from 0 to 16 **do**
      $S5\left(\boldsymbol{x}\right)\leftarrow S4\left(\boldsymbol{x}\right)$
      $S5\left(\boldsymbol{x}+16\right)\leftarrow S4\left(\boldsymbol{x}\right)<<<w$
    end for
    for all w from 0 to 2^{32} such that $HW\left(w\right)=16$ **do**
      for all u from 0 to 31 **do**
        Assign $S5_4\left(u\right)$ as $\frac{w}{2^u}\ mod\ 2$
      end for
      if S5 fulfills bijectivity and perfect SAC **then**
        $\mathbb{S}5\cup\{S5\}$
      end if
    end for
  end for
end for
return $\mathbb{S}5$.

Table 3: Values of $S5$ derived from $S4$.

$\boldsymbol{x}$	$S5\,(\boldsymbol{x})$	$S5_4$	$S5_3$	$S5_2$	$S5_1$	$S5_0$
0	$S4\,(0)$	β_0	$S4_3\,(0)$	$S4_2\,(0)$	$S4_1\,(0)$	$S4_0\,(0)$
1	$S4\,(1)$	β_1	$S4_3\,(1)$	$S4_2\,(1)$	$S4_1\,(1)$	$S4_0\,(1)$
2	$S4\,(2)$	β_2	$S4_3\,(2)$	$S4_2\,(2)$	$S4_1\,(2)$	$S4_0\,(2)$
3	$S4\,(3)$	β_3	$S4_3\,(3)$	$S4_2\,(3)$	$S4_1\,(3)$	$S4_0\,(3)$
4	$S4\,(4)$	β_4	$S4_3\,(4)$	$S4_2\,(4)$	$S4_1\,(4)$	$S4_0\,(4)$
5	$S4\,(5)$	β_5	$S4_3\,(5)$	$S4_2\,(5)$	$S4_1\,(5)$	$S4_0\,(5)$
6	$S4\,(6)$	β_6	$S4_3\,(6)$	$S4_2\,(6)$	$S4_1\,(6)$	$S4_0\,(6)$
7	$S4\,(7)$	β_7	$S4_3\,(7)$	$S4_2\,(7)$	$S4_1\,(7)$	$S4_0\,(7)$
8	$S4\,(8)$	β_8	$S4_3\,(8)$	$S4_2\,(8)$	$S4_1\,(8)$	$S4_0\,(8)$
9	$S4\,(9)$	β_9	$S4_3\,(9)$	$S4_2\,(9)$	$S4_1\,(9)$	$S4_0\,(9)$
10	$S4\,(10)$	β_{10}	$S4_3\,(10)$	$S4_2\,(10)$	$S4_1\,(10)$	$S4_0\,(10)$
11	$S4\,(11)$	β_{11}	$S4_3\,(11)$	$S4_2\,(11)$	$S4_1\,(11)$	$S4_0\,(11)$
12	$S4\,(12)$	β_{12}	$S4_3\,(12)$	$S4_2\,(12)$	$S4_1\,(12)$	$S4_0\,(12)$
13	$S4\,(13)$	β_{13}	$S4_3\,(13)$	$S4_2\,(13)$	$S4_1\,(13)$	$S4_0\,(13)$
14	$S4\,(14)$	β_{14}	$S4_3\,(14)$	$S4_2\,(14)$	$S4_1\,(14)$	$S4_0\,(14)$
15	$S4\,(15)$	β_{15}	$S4_3\,(15)$	$S4_2\,(15)$	$S4_1\,(15)$	$S4_0\,(15)$
16	$S4\,(0) <<< w$	β_{16}	$S4_{3-w}\,(0)$	$S4_{2-w}\,(0)$	$S4_{1-w}\,(0)$	$S4_{-w}\,(0)$
17	$S4\,(1) <<< w$	β_{17}	$S4_{3-w}\,(1)$	$S4_{2-w}\,(1)$	$S4_{1-w}\,(1)$	$S4_{-w}\,(1)$
18	$S4\,(2) <<< w$	β_{18}	$S4_{3-w}\,(2)$	$S4_{2-w}\,(2)$	$S4_{1-w}\,(2)$	$S4_{-w}\,(2)$
19	$S4\,(3) <<< w$	β_{19}	$S4_{3-w}\,(3)$	$S4_{2-w}\,(3)$	$S4_{1-w}\,(3)$	$S4_{-w}\,(3)$
20	$S4\,(4) <<< w$	β_{20}	$S4_{3-w}\,(4)$	$S4_{2-w}\,(4)$	$S4_{1-w}\,(4)$	$S4_{-w}\,(4)$
21	$S4\,(5) <<< w$	β_{21}	$S4_{3-w}\,(5)$	$S4_{2-w}\,(5)$	$S4_{1-w}\,(5)$	$S4_{-w}\,(5)$
22	$S4\,(6) <<< w$	β_{22}	$S4_{3-w}\,(6)$	$S4_{2-w}\,(6)$	$S4_{1-w}\,(6)$	$S4_{-w}\,(6)$
23	$S4\,(7) <<< w$	β_{23}	$S4_{3-w}\,(7)$	$S4_{2-w}\,(7)$	$S4_{1-w}\,(7)$	$S4_{-w}\,(7)$
24	$S4\,(8) <<< w$	β_{24}	$S4_{3-w}\,(8)$	$S4_{2-w}\,(8)$	$S4_{1-w}\,(8)$	$S4_{-w}\,(8)$
25	$S4\,(9) <<< w$	β_{25}	$S4_{3-w}\,(9)$	$S4_{2-w}\,(9)$	$S4_{1-w}\,(9)$	$S4_{-w}\,(9)$
26	$S4\,(10) <<< w$	β_{26}	$S4_{3-w}\,(10)$	$S4_{2-w}\,(10)$	$S4_{1-w}\,(10)$	$S4_{-w}\,(10)$
27	$S4\,(11) <<< w$	β_{27}	$S4_{3-w}\,(11)$	$S4_{2-w}\,(11)$	$S4_{1-w}\,(11)$	$S4_{-w}\,(11)$
28	$S4\,(12) <<< w$	β_{28}	$S4_{3-w}\,(12)$	$S4_{2-w}\,(12)$	$S4_{1-w}\,(12)$	$S4_{-w}\,(12)$
29	$S4\,(13) <<< w$	β_{29}	$S4_{3-w}\,(13)$	$S4_{2-w}\,(13)$	$S4_{1-w}\,(13)$	$S4_{-w}\,(13)$
30	$S4\,(14) <<< w$	β_{30}	$S4_{3-w}\,(14)$	$S4_{2-w}\,(14)$	$S4_{1-w}\,(14)$	$S4_{-w}\,(14)$
31	$S4\,(15) <<< w$	β_{31}	$S4_{3-w}\,(15)$	$S4_{2-w}\,(15)$	$S4_{1-w}\,(15)$	$S4_{-w}\,(15)$

From the $S5$s generated, we propose a candidate, $\mathcal{P}$, that possesses large nonlinearity, large iterative period, and has neither fixed points nor inverse fixed points. The proposed S–box is presented in Table 4.

Table 4: Input and output pairs of $\mathscr{P}$.

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathscr{P}(\boldsymbol{x})$	10	6	8	25	19	20	23	13	17	11	16	12	5	2	31	14

$\boldsymbol{x}$	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\mathscr{P}(\boldsymbol{x})$	21	28	1	3	22	24	30	27	18	7	0	9	26	4	15	29

4 Performance Analysis

The evaluation of a nonlinear component encompasses several aspects, including bijectivity, nonlinearity, strict avalanche criterion, bit-independence criterion, differential approximation, and linear approximation. These aspects have been discussed in various studies, such as Carlet et al.'s work on identifying cryptographically strong S–boxes based on the sum of S–box values in matrix form and the difference in Hamming weights between inputs and outputs [16], and Durasevic et al.'s search for S–boxes ranging from 4–bit to 8–bit with high boomerang uniformity [19].

We conduct a comparison of the security results with other published 5–bit S–boxes, such as the S–boxes of aforementioned Ascon [18] and Keccak [11], as well as S–boxes utilized in PRIMATEs [6], ICEPOLE [33], and SHAMASH [36]. Additionally, we compare our generated S–box with a recent novel 5–bit S–box proposed by Thakor et al. [44].

The method introduced by Kim et al. [23] can be employed to generate S–boxes of various sizes, including 5–bit S–boxes. To the best of our knowledge, we strive to generate the most optimal S–box, $\mathscr{K}$, that possesses the highest possible nonlinearity using this method. Table 5 shows $\mathscr{K}$ that we have generated.

Table 5: Input and output pairs of $\mathscr{K}$.

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathscr{K}(\boldsymbol{x})$	24	26	23	27	9	6	29	28	18	0	3	31	30	1	20	5

$\boldsymbol{x}$	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\mathscr{K}(\boldsymbol{x})$	10	8	11	7	22	25	12	13	16	2	15	19	17	14	21	4

4.1 Bijectivity

An S–box exhibits the property of bijectivity when each input is mapped to a unique output. This ensures that no two different inputs are mapped to the same output and that no output remains unmapped. The bijectivity of an S–box can be determined using (6) [21]. For an n–bit S–box and for all $\boldsymbol{y}$,

$$\#\{\boldsymbol{x}|Sn(\boldsymbol{x}) = \boldsymbol{y}\} = 1. \tag{6}$$

In our method, initially, $S3$ was generated for $S4\,(i)$, while $S\,(8+i)$ was assigned the rotated values of $S3$. A search process consisting of 2^{16} steps was then employed to identify $S4$, with the condition that only bijective S–boxes were retained. This process preserves the bijectivity of $S4$. The same method was used to generate $S5$ from $S4$. Similarly, a 2^{32}-step approach was utilized to generate bijective 5–bit S–boxes. As a result, the selected S–box satisfied the criteria for bijectivity.

4.2 Nonlinearity

The nonlinearity of a function is a measure of its ability to resist linear attacks [26]. The nonlinearity, NL, of function f is given by (7) [28], where $W_f(\boldsymbol{a})$ represents the Walsh Hadamard Transform of $f(\boldsymbol{x})$ and is defined as $W_f(\boldsymbol{a}) = \sum_{\boldsymbol{x}\in\mathbb{F}_2^n}(-1)^{f(\boldsymbol{x})+\boldsymbol{a}\cdot\boldsymbol{x}}$ for all $\boldsymbol{a}\in\mathbb{F}_2^n$,

$$NL(f) = 2^{n-1}\left(1 - 2^{-n}\max|W_f(\boldsymbol{a})|\right). \tag{7}$$

The proposed S–box $\mathscr{P}$ achieves an average nonlinearity of 10.8, which aligns with cryptographic robustness criteria for resisting linear cryptanalysis. This value is higher than those of Ascon and Keccak. Table 6 details the nonlinearity distribution across the five output bits of $\mathscr{P}$, while Table 7 provides a comparative analysis of its average nonlinearity against benchmark S–boxes.

Table 6: Nonlinearity of $\mathscr{P}$.

$NL\,(\mathscr{P}_0)$	12
$NL\,(\mathscr{P}_1)$	10
$NL\,(\mathscr{P}_2)$	10
$NL\,(\mathscr{P}_3)$	10
$NL\,(\mathscr{P}_4)$	12

5 Bit-Independence Criterion

The bit-independence criterion [31] stipulates that each input bit should influence every output bit such that the alterations in the output bits are independent of each other. The independence between the changes in each output bit can be analyzed by measuring the nonlinearity [3] (refer to Section 4.2) and SAC [8] (refer to Section 2) of $Sn_\imath + Sn_\jmath$ for all $0 \le \imath, \jmath < n$ where $\imath \ne \jmath$.

The average BIC-nonlinearity of $\mathscr{P}$ is 9.8, indicating satisfactory cryptographic performance. The maximum and minimum BIC-SAC values are 0.5125 and 0.475, respectively, which are very close to the ideal value of 0.5. Tables 8 and 9 present the BIC and BIC-SAC of $\mathscr{P}$, respectively.

Table 7: Comparison of S–boxes.

		Ascon	Keccak	PRIMATEs	ICEPOLE	Shamash	Thakor's S–box	$\mathscr{K}$	**Our work ($\mathscr{P}$)**
Nonlinearity	Lowest	8	8	12	8	12	8	8	**10**
	Highest	8	8	12	8	12	10	12	**12**
	Average	8	8	12	8	12	8.4	8.8	**10.8**
SAC	Lowest	0	0	0.5	0.125	0.5	0.25	0.5	**0.5**
	Highest	1	1	1	0.875	1	0.75	0.5	**0.5**
	Average	0.62	0.4	0.54	0.425	0.6	0.54	0.5	**0.5**
BIC-Nonlinearity	Lowest	8	8	12	8	12	8	8	**8**
	Highest	12	12	12	12	12	10	12	**10**
	Average	11.2	10	12	10	12	9	9.2	**9.8**
BIC-SAC	Lowest	0.3	0.5	0.5	0.5	0.5	0.45	0.4	**0.475**
	Highest	0.6	0.6	0.6	0.6	0.5	0.575	0.6	**0.55**
	Average	0.52	0.55	0.51	0.55	0.5	0.5075	0.49	**0.5125**
Differential uniformity, D		8	8	2	8	2	8	32	**6**
Linear probability, L		0.25	0.25	0.125	0.25	0.125	0.25	0.5	**0.25**
Number of fixed points		0	2	0	0	1	0	0	**0**
Number of inverse fixed points		0	0	2	2	0	1	0	**0**
Shortest iterative period		6	1	2	2	1	4	16	**32**
References		[18]	[11]	[6]	[33]	[36]	[44]	[23]	**This paper**

Table 8: BIC-nonlinearity of proposed S–box.

$S_{k+\ell}$	$\ell = 0$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
$k = 0$	-	10	10	10	10
$k = 1$	10	-	10	10	10
$k = 2$	10	10	-	10	10
$k = 3$	10	10	10	-	8
$k = 4$	10	10	10	8	-

Table 9: BIC-SAC of proposed S–box.

$S_{k+\ell}$	$\ell = 0$	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
$k = 0$	-	0.4750	0.5250	0.4750	0.5500
$k = 1$	0.4750	-	0.5250	0.5250	0.5250
$k = 2$	0.5250	0.5250	-	0.5000	0.5250
$k = 3$	0.4750	0.5250	0.5000	-	0.5000
$k = 4$	0.5500	0.5250	0.5250	0.5000	-

5.1 Differential uniformity

Differential cryptanalysis [12] represents a significant method of attack in the annals of cryptography. The resilience of Sn to differential cryptanalysis can be quantified using (8),

$$D = \max_{\Delta \boldsymbol{x} \in \mathbb{F}_2^n, \Delta \boldsymbol{y} \in \mathbb{F}_2^n} \left(\# \left(\Delta \boldsymbol{y} = Sn\left(\boldsymbol{x}\right) + Sn\left(\boldsymbol{x} + \Delta \boldsymbol{x}\right) \right) \right), \tag{8}$$

where $\Delta \boldsymbol{x}, \Delta \boldsymbol{y} \neq 0$. A lower value of differential uniformity, D, signifies a stronger resistance of the S–box to differential cryptanalysis [45]. Furthermore, D also denotes the maximum differential value of the Difference Distribution Table.

The D value of $\mathscr{P}$ is 6, which is lower than those of Ascon, Keccak, ICEPOLE, and Thakor's S–boxes. This result is summarised in Table 7. Table 12 presents the difference distribution table of $\mathscr{P}$.

5.2 Linear probability

Linear cryptanalysis [30] is another prominent cryptanalytic attack in the field of cryptography. The robustness of Sn against this form of cryptanalysis can be evaluated using (9),

$$L = \max_{\Gamma_{\boldsymbol{x}} \in \mathbb{F}_2^n, \Gamma_{\boldsymbol{y}} \in \mathbb{F}_2^n} \left| \frac{\# \left(\boldsymbol{y} \cdot \Gamma_{\boldsymbol{y}} = Sn\left(\boldsymbol{x} \cdot \Gamma_{\boldsymbol{x}}\right) \right)}{2^n} - \frac{1}{2} \right|, \tag{9}$$

where linear masks $\Gamma_{\boldsymbol{x}}, \Gamma_{\boldsymbol{y}} \neq 0$. An S–box exhibits stronger resistance to linear cryptanalysis when it has a lower linear probability, L. Additionally, L represents the maximum linear value of the Linear Approximation Table.

The L value of $\mathscr{P}$ is 0.25, matching those of Ascon, Keccak, and most of the S–boxes being compared here. The L values for all S–boxes under comparison are summarized in Table 7. The linear approximation table of $\mathscr{P}$ is presented in Table 13.

5.3 Fixed points, reverse fixed points and short period ring

In order to construct a secure S–box, it is imperative to identify and eliminate the presence of fixed points, reverse fixed points, and short period rings [2]. A fixed point, as defined in [46], occurs when an input value in an S–box maps directly to itself, as illustrated in (10). Conversely, a reverse fixed point is a scenario where the input of an n–bit S–box maps to its own bitwise complement, as depicted in (11),

$$Sn\left(\boldsymbol{x}\right) = \boldsymbol{x}, \tag{10}$$

$$Sn\left(\boldsymbol{x}\right) = 2^n - 1 - \boldsymbol{x}. \tag{11}$$

The presence of fixed points within an S–box can potentially expose secret data to an attacker through intercepted ciphertext [29]. Consequently, it is crucial to ensure that the finalized S–box is devoid of any fixed points [48]. Furthermore, a strong S–box should circumvent short iterative periods [25], a condition where $f^m\left(\boldsymbol{x}\right) = \boldsymbol{x}$ holds true for a small value of m.

As shown in Table 7, there are no fixed points or inverse fixed points in $\mathscr{P}$. The iterative period of $\mathscr{P}$ is 32, which is the highest achievable since there are only $2^5 = 32$ entries in a 5–bit S–box.

Table 7 presents the number of fixed points and inverse fixed points, as well as the iterative periods of all S–boxes analyzed in this paper.

5.4 SAC comparison with other S–boxes

The S–boxes of Ascon and Keccak display SAC values of an absolute 0 or 1. In contrast, the S–boxes of PRIMATE and SHAMASH demonstrate better SAC conditions, albeit with an absolute SAC value of 1. Thakor's S–box does not exhibit SAC values of absolute 0 or 1, instead presenting a minimum SAC value of 0.25 and a maximum of 0.75. The S–box of ICEPOLE displays a broader range of SAC values, with a minimum of 0.125 and a maximum of 0.875. $\mathscr{K}$ and $\mathscr{P}$, do not exhibit SAC values of absolute 0 or 1 either, but they demonstrate a more consistent set of values at 0.5. Table 10 presents the SAC values for five output bits of $\mathscr{P}$. As illustrated in Table 7, $\mathscr{P}$ adheres strictly to a perfect SAC value of 0.5, distinguishing it from other S–boxes.

Table 10: SAC of proposed S–box.

$\Delta \boldsymbol{x}$	2^0	2^1	2^2	2^3	2^4
$\delta\left(\mathscr{P}_0, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$	0.5	0.5	0.5	0.5	0.5
$\delta\left(\mathscr{P}_1, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$	0.5	0.5	0.5	0.5	0.5
$\delta\left(\mathscr{P}_2, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$	0.5	0.5	0.5	0.5	0.5
$\delta\left(\mathscr{P}_3, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$	0.5	0.5	0.5	0.5	0.5
$\delta\left(\mathscr{P}_4, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$	0.5	0.5	0.5	0.5	0.5

6 Rationale Behind Recursive Construction of S–boxes

Generation of $S5$ candidates is initiated from $S3$s as elucidated in Section 3. Our findings indicate that $x_a x_b + x_a x_c + x_b$ satisfies a perfect strict avalanche criterion as shown in Table 11. A constant 1 is added to the polynomial of f_1 to prevent fixed points. This gives 2^3 possible output of $S3$ which is the result of using a 3–bit S–box.

The generation of 3–bit S–boxes utilizes a fundamental Boolean function, $x_a x_b + x_a x_c + x_b$, which incorporates all input variables x_a, x_b and x_c, as demonstrated in Proposition 2.1. Additionally, the function avoids the characteristic of any x_ι existing solely in linear terms, as emphasized in Proposition 2.2.

The presence of the constant 1 at $S3_1$ in (3) might initially appear superfluous, given that the security implications for the intermediate variables $S3$ and $S4$ are not substantial. Nevertheless, the absence of this constant results in a lack of solutions based on our experimental data. The inclusion of the constant 1 proves beneficial in stimulating a larger pool of potential candidates, as it facilitates an increased number of options during the generation of the remaining bits for $S4$ and $S5$. To illustrate, when $S4\,(0) = 0$, the remaining bit for $S5\,(0)$ can only be 1 in order to prevent fixed points such that $S5\,(0) = 16$.

Table 11: Characteristics of polynomials generated for $S3$.

Polynomials	Fulfiling Proposition 2.1	Fulfiling Proposition 2.2	Maximum degree not exceeding $\boldsymbol{n-1}$	Perfect SAC (Proposition 6.4)	Bijective
$S3_0 = x_a$ $S3_1 = x_b$ $S3_2 = x_c$			✔		✔
$S3_0 = x_a + x_b$ $S3_1 = x_a + x_c$ $S3_2 = x_b + x_c$			✔		
$S3_0 = x_a x_b$ $S3_1 = x_a x_c$ $S3_2 = x_b x_c$		✔	✔		
$S3_0 = x_a x_b + x_c$ $S3_1 = x_a x_c + x_b$ $S3_2 = x_b x_c + x_a$	✔		✔		
$S3_0 = x_a x_b + x_a x_c$ $S3_1 = x_a x_b + x_b x_c$ $S3_2 = x_a x_c + x_b x_c$	✔	✔	✔	✔	
$S3_0 = x_a x_b + x_a x_c + x_a$ $S3_1 = x_a x_b + x_b x_c + x_b$ $S3_2 = x_a x_c + x_b x_c + x_c$	✔	✔	✔	✔	
$S3_0 = x_a x_b + x_a x_c + x_b$ $S3_1 = x_a x_b + x_b x_c + x_c$ $S3_2 = x_a x_c + x_b x_c + x_a$	✔	✔	✔	✔	✔

Both $Sn^*(\boldsymbol{x})$ and $Sn^*\left(\boldsymbol{x} + 2^{n-1}\right)$ are assigned from $Sm(\boldsymbol{v})$ where $m = n-1$, as shown in (4) and (5). Equation (12) shows the generalization of both equations with a random number u,

$$Sn^*(\boldsymbol{x}) = \begin{cases} Sm(\boldsymbol{x}), & \text{for } \boldsymbol{x} < 2^m, \\ Sm(\boldsymbol{x}) <<< u, & \text{for } \boldsymbol{x} \geq 2^m. \end{cases} \tag{12}$$

From (12), it can be observed that $Sn_\imath^*$ is derived from two output bits of Sm, which are $Sm_\imath$ for $Sn_r^*(0)$ to $Sn_r^*(2^m - 1)$ and $Sm_\jmath$ for $Sn_r^*(2^m)$ to $Sn_r^*(2^n - 1)$. Sm with perfect SAC can be used for generating Sn that also possesses perfect SAC as well. This characteristic is supported by Propositions 6.1, 6.2, 6.3, and 6.4.

Proposition 6.1. $\lambda(Sm_\imath, Sm_\jmath, \boldsymbol{x}) = 0.5$.

Proof. Due to bijectivity of Sm, $\lambda(Sm_\imath, Sm_\jmath, \boldsymbol{x})$ is equivalent to $P(x_\imath + x_\jmath = 1)$.
Since $P(x_\imath + x_\jmath = 1) = 0.5$,

$$\lambda(Sm_\imath, Sm_\jmath, \boldsymbol{v}) = 0.5.$$

□

Proposition 6.2. *If Sm has perfect SAC, one of its characteristics is that for $t = 0$ and $0 \leq p < 2^{m-1-t}$,* $\delta(Sm_\imath, 2p, 1) = 0.5$.

Proof. According to (1), when $\Delta \boldsymbol{x} = 1$, $\delta\left(Sm_{\imath}, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$ is effectively $\delta\left(Sm_{\imath}, 2p, 1\right)$. Since $\delta\left(Sm_{\imath}, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$ is stipulated to be 0.5, $\delta\left(Sm_{\imath}, 2p, 1\right)$ must also be 0.5. □

Proposition 6.3. *If Sm has perfect SAC, one of its characteristics is that for $1 \leq t \leq m-1$, $0 \leq q < 2^t$ and $0 \leq p < \dfrac{2^{m-1}}{2^t}$, $\delta\left(Sm_{\imath}, 4^t p + q, 2^t\right) = 0.5$.*

Proof. $\delta\left(Sm_{\imath}, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$ is equivalent to $\delta\left(Sm_{\imath}, 4^t p + q, 2^t\right)$ when $2^2 \leq \Delta \boldsymbol{x} \leq 2^{m-1}$. Since $\delta\left(Sm_{\imath}, \boldsymbol{x}, \Delta \boldsymbol{x}\right) = 0.5$ to fulfill the principle of perfect SAC, $\delta\left(Sm_{\imath}, 4^t p + q, 2^t\right)$ must also be 0.5. □

Proposition 6.4. *When Sm has perfect SAC, the resultant Sn^* will also have perfect SAC.*

Proof. As shown in (12), $Sn_{\imath}^*$ is derived from $Sm_{\imath}$ and $Sm_{\jmath}$. When $\Delta \boldsymbol{x} = 2^{n-1}$, the function $\delta\left(Sn_{\imath}^*, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$ is equivalent to $\lambda\left(Sm_{\imath}, Sm_{\jmath}, \boldsymbol{x}\right)$. Given that Sm exhibits perfect SAC, it follows that $\lambda\left(Sm_{\imath}, Sm_{\jmath}, \boldsymbol{x}\right) = 0.5$, which consequently leads to $\delta\left(Sn_{\imath}^*, \boldsymbol{x}, \Delta \boldsymbol{x}\right) = 0.5$ as stated in Proposition 6.1.

For $1 \leq \Delta \boldsymbol{x} \leq 2^{n-2}$, the function $\delta\left(Sn_{\imath}^*, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$ is equivalent to $\delta\left(Sm_{\imath}, 2p, 1\right)$ for $\Delta \boldsymbol{x} = 1$, while $\delta\left(Sn_{\imath}^*, \boldsymbol{x}, \Delta \boldsymbol{x}\right)$ is equivalent to $\delta\left(Sm_{\imath}, 4^t p + q, 2^t\right)$ for $2^2 \leq \Delta \boldsymbol{x} \leq 2^{m-1}$. According to Propositions 6.2 and 6.3, if Sm exhibits perfect SAC, both $\delta\left(Sm_{\imath}, 2p, 1\right)$ and $\delta\left(Sm_{\imath}, 4^t p + q, 2^t\right)$ must equal 0.5, resulting in $\delta\left(Sn_{\imath}^*, \boldsymbol{x}, \Delta \boldsymbol{x}\right) = 0.5$. Given that $\delta\left(Sn_{\imath}^*, \boldsymbol{x}, \Delta \boldsymbol{x}\right) = 0.5$ for all possible values of $\Delta \boldsymbol{x}$, it can be concluded that Sn^* also exhibits perfect SAC. □

The S–box construction proposed in this paper guarantees the existence of a connection among all output bits of the S–boxes, which is a requirement for achieving a perfect SAC as stated in Proposition 6.5.

Proposition 6.5. *For an n–bit S–box, Sn and its antecedent $(n-1)$–bit S–box Sm, if $Sn^*\left(2^{n-1} + \boldsymbol{x}\right) = Sn^*\left(\boldsymbol{x}\right) = Sm\left(\boldsymbol{x}\right)$ for all $0 \leq v < 2^{n-1}$, then $Sn_0, Sn_1, \ldots, Sn_{n-2}$ are independent of Sn_{n-1}.*

Proof. Since $2^{n-1} + \boldsymbol{x} \equiv \boldsymbol{x}\left(mod\, 2^{n-1}\right)$ and $Sn^*\left(2^{n-1} + \boldsymbol{x}\right) = Sn^*\left(\boldsymbol{x}\right)$, Sn_{n-1} is independent of the other $n-1$ output bits of Sn. □

When generating a 4–bit S–box, the value from $S4^*\left(\boldsymbol{x} + 8\right)$ is rotated from $S4^*\left(\boldsymbol{x}\right)$. This approach is utilized to ensure that the precomputed three bits of the resulting 4–bit S–box are interconnected with the remaining bit in accordance with Proposition 2.1. Similarly, for the generation of 5–bit S–boxes, the value from $S5^*\left(\boldsymbol{x} + 16\right)$ is rotated from $S5^*\left(\boldsymbol{x}\right)$ to prevent independence between the remaining bit and the other 4 output bits.

The proposed S–box can be constructed using logic gates as follows. Please note that we describe the operations based on the order of operations used in the GCC compiler, where the bitwise AND operation ($\wedge$) is performed first, followed by XOR ($\bigoplus$) and OR ($\vee$) operations,

$$
\begin{aligned}
S3_0 &= x_0 \wedge \left(x_1 \oplus x_2\right) \oplus x_2, \\
S3_1 &= \overline{x_2 \wedge \left(x_0 \oplus x_1\right) \oplus x_1}, \\
S3_2 &= x_1 \wedge \left(x_0 \oplus x_2\right) \oplus x_0, \\
S4_3 &= \overline{x_2 \vee x_0} \oplus \overline{x_3} \wedge x_1 \wedge x_0 \oplus x_3 \left(\overline{x_2} \vee x_1\right),
\end{aligned}
$$

$$
\begin{aligned}
S4_2 &= x_3 \wedge S3_0 \vee \overline{x_3} \wedge S3_2,\\
S4_1 &= x_3 \wedge S3_2 \vee \overline{x_3} \wedge S3_1,\\
S4_0 &= x_3 \wedge S3_1 \vee \overline{x_3} \wedge S3_0,\\
S5_4 &= x_1 \wedge x_0 \oplus x_2\left(\overline{x_4} \vee x_1\right) \oplus x_3 \wedge \overline{x_1} \wedge x_0 \oplus x_4\left(\overline{x_1} \vee x_0\right) \oplus x_3 \wedge x_2 \wedge x_1 \wedge \overline{x_0} \oplus \overline{x_4} \wedge x_3\left(\overline{x_2 \wedge x_0}\right),\\
S5_3 &= x_4 \wedge S4_2 \vee \overline{x_4} \wedge S4_3,\\
S5_2 &= x_4 \wedge S4_1 \vee \overline{x_4} \wedge S4_2,\\
S5_1 &= x_4 \wedge S4_0 \vee \overline{x_4} \wedge S4_1,\\
S5_0 &= x_4 \wedge S4_3 \vee \overline{x_4} \wedge S4_0.
\end{aligned}
$$

The only output bits that vary among all $S5$s are $S4_3$ and $S5_4$, due to random generations. Although $S3_0$, $S3_1$, and $S3_2$ also vary among all $S5$s, the structure of their logical expression remains identical, i.e., $x_a \wedge (x_b \oplus x_c) \oplus x_c$. The remaining output bits can be derived based on the values of their antecedent S–boxes, rendering our S–box lightweight and equipped with a simple logic gate circuit.

In contrast to Kim et al. [23]'s proposal, our approach generates $S5$ solely from $S3\left(\boldsymbol{x}\right)$ and $S4\left(\boldsymbol{x}\right)$, thereby simplifying the overall process. Kim's method requires not only the computation of $S3\left(\boldsymbol{x}\right)$ and $S4\left(\boldsymbol{x}\right)$ but also those computed using the input $\boldsymbol{x}+\Delta\boldsymbol{x}$, which necessitates additional memory cells. In our work, we allocate 3 memory cells for storing $S3_0\left(\boldsymbol{x}\right)$, $S3_1\left(\boldsymbol{x}\right)$, and $S3_2\left(\boldsymbol{x}\right)$, and 4 memory cells for storing $S4_0\left(\boldsymbol{x}\right)$, $S4_1\left(\boldsymbol{x}\right)$, $S4_2\left(\boldsymbol{x}\right)$, and $S4_3\left(\boldsymbol{x}\right)$. In Kim's method, these same cells for the original input are required in addition to an extra 3 cells for storing $S3_0\left(\boldsymbol{x}+\Delta\boldsymbol{x}\right)$, $S3_1\left(\boldsymbol{x}+\Delta\boldsymbol{x}\right)$, and $S3_2\left(\boldsymbol{x}+\Delta\boldsymbol{x}\right)$, as well as another 4 cells for storing $S4_0\left(\boldsymbol{x}+\Delta\boldsymbol{x}\right)$, $S4_1\left(\boldsymbol{x}+\Delta\boldsymbol{x}\right)$, $S4_2\left(\boldsymbol{x}+\Delta\boldsymbol{x}\right)$, and $S4_3\left(\boldsymbol{x}+\Delta\boldsymbol{x}\right)$. This distinction is further illustrated in the logic circuit of $\mathcal{K}$ shown below,

$$
\begin{aligned}
S3_0\left(\boldsymbol{x}\right) &= x_1 \oplus x_2\left(\overline{x_0 \oplus x_1}\right),\\
S3_1\left(\boldsymbol{x}\right) &= x_0 \wedge \overline{x_1} \oplus x_1 \wedge \overline{x_2},\\
S3_2\left(\boldsymbol{x}\right) &= x_1 \wedge \overline{x_0} \oplus x_0 \wedge x_2,\\
S3_0\left(\boldsymbol{x}+1\right) &= x_1 \oplus x_2\left(x_0 \oplus x_1\right),\\
S3_1\left(\boldsymbol{x}+1\right) &= x_0 \wedge \overline{x_1} \oplus \overline{x_1 \wedge x_2},\\
S3_2\left(\boldsymbol{x}+1\right) &= x_1 \wedge x_0 \oplus \overline{x_0} \wedge x_2,\\
S4_3\left(\boldsymbol{x}\right) &= \overline{x_1 \vee x_3} \oplus x_0\left(x_1 \oplus x_2\right) \oplus x_2\left(x_1 \oplus x_3\right),\\
S4_2\left(\boldsymbol{x}\right) &= x_3 \wedge S3_2\left(\boldsymbol{x}+1\right) \vee \overline{x_3} \wedge S3_2\left(\boldsymbol{x}\right),\\
S4_1\left(\boldsymbol{x}\right) &= x_3 \wedge S3_1\left(\boldsymbol{x}+1\right) \vee \overline{x_3} \wedge S3_1\left(\boldsymbol{x}\right),\\
S4_0\left(\boldsymbol{x}\right) &= x_3 \wedge S3_0\left(\boldsymbol{x}+1\right) \vee \overline{x_3} \wedge S3_0\left(\boldsymbol{x}\right),\\
S4_3\left(\boldsymbol{x}+1\right) &= \overline{x_1 \vee x_3} \oplus \overline{x_0}\left(x_1 \oplus x_2\right) \oplus x_2\left(x_1 \oplus x_3\right),\\
S4_2\left(\boldsymbol{x}+1\right) &= x_3 \wedge S3_2\left(\boldsymbol{x}\right) \vee \overline{x_3} \wedge S3_2\left(\boldsymbol{x}+1\right),\\
S4_1\left(\boldsymbol{x}+1\right) &= x_3 \wedge S3_1\left(\boldsymbol{x}\right) \vee \overline{x_3} \wedge S3_1\left(\boldsymbol{x}+1\right),\\
S4_0\left(\boldsymbol{x}+1\right) &= x_3 \wedge S3_0\left(\boldsymbol{x}\right) \vee \overline{x_3} \wedge S3_0\left(\boldsymbol{x}+1\right),\\
S5_4 &= \overline{x_1} \wedge x_2 \oplus \overline{x_3} \wedge x_4 \oplus \overline{x_3\left(x_0 \oplus x_1 \oplus x_2\right)},\\
S5_3 &= x_4 \wedge S4_3\left(\boldsymbol{x}+1\right) \vee \overline{x_4} \wedge S4_3\left(\boldsymbol{x}\right),\\
S5_2 &= x_4 \wedge S4_2\left(\boldsymbol{x}+1\right) \vee \overline{x_4} \wedge S4_2\left(\boldsymbol{x}\right),\\
S5_1 &= x_4 \wedge S4_1\left(\boldsymbol{x}+1\right) \vee \overline{x_4} \wedge S4_1\left(\boldsymbol{x}\right),\\
S5_0 &= x_4 \wedge S4_0\left(\boldsymbol{x}+1\right) \vee \overline{x_4} \wedge S4_0\left(\boldsymbol{x}\right).
\end{aligned}
$$

In terms of gate counts, our implementation employs fewer gates than Kim's method. Specifically, our design requires 13 XOR gates, 31 AND gates, 11 OR gates, and 17 NOT gates, whereas Kim's approach uses 30 XOR gates, 37 AND gates, 12 OR gates, and 23 NOT gates. Overall, the reduced gate counts in our work demonstrate a more efficient implementation compared to Kim's method.

Algorithm 3 The algorithm for picking 4–bit S–boxes that surpasses $\mathcal{S}$

procedure MINIMIZES4(A set $\mathbb{S}4$, $\mathcal{S}$)
  Initialize an empty set $\mathbb{T}$.
  for all $S4 \in \mathbb{S}4$ **do**
    Create an empty list $S5$.
    for all $0 \leq u < 4$ **do**
      for all $0 \leq \boldsymbol{x} < 2^4$ **do**
        $S5^*(\boldsymbol{x}) \leftarrow S4(\boldsymbol{x})$
        $S5^*(\boldsymbol{x} + 2^4) \leftarrow S4(\boldsymbol{x}) <<< u$
      end for
    end for
    $sum \leftarrow 0$
    $\mathcal{A} \leftarrow 1$
    for all $0 \leq p < 4$ **do**
      $sum \leftarrow sum + NL(S5_p^*)$.
      if $\delta(f_p, \boldsymbol{x}, \triangle\boldsymbol{x}) \neq 0.5$ where $HW(\Delta\boldsymbol{x}) = 1$ **then**
        $\mathcal{A} \leftarrow 0$
      end if
    end for
    if $sum \geq \mathcal{S}$ and $\mathcal{A} = 1$ **then**
      $\mathbb{T} \cup \{S5^*\}$
    end if
  end for
  return $\mathbb{T}$.
end procedure

7 Optimization

The process for generating $S5$ candidates can be expedited further by discarding unfavourable $S5^*$ candidates. Since $S5^*$ S–boxes are still incomplete, one method to accomplish this involves pre-screening $S5^*$ S–boxes based on their nonlinearity and SAC values. These values can be computed for each output bit separately, unlike other security criteria discussed in Section 4, which require the evaluation of the entire S–box.

Let $\mathbb{S}5$ be a group consisting of all generated $S5^*$, we define $\mathcal{S}$ as in (13),

$$\mathcal{S} = \max_{S5^* \in \mathbb{S}5} \left(\sum_{u=0}^{3} NL(S5_u^*) \right). \tag{13}$$

In our experiment, we have determined $\mathcal{S} = 42$ by (7). To optimize the search process for 5–bit S–boxes, we employ the MINIMIZES4() function in Algorithm 3 to filter out $S5^*$ that do not meet $\mathcal{S}$, by setting $\mathcal{S}$ as the threshold. As a result, we have identified 96 S–boxes that meet $\mathcal{S}$, as detailed

in Appendix 9.2. The algorithm for picking favourable $S5$s is outlined in Algorithm 3. As a result, this computation only requires $\binom{16}{8} \cdot 3! \cdot 3 + \binom{32}{16} \cdot 4 \cdot 96 \approx 2^{37.75}$ steps.

8 Conclusions

This paper presented an improved recursive method for constructing 5–bit S–boxes that satisfy perfect SAC. The proposed method efficiently generates S–boxes with strong cryptographic properties, including high nonlinearity, optimal iterative periods, and perfect SAC values. Additionally, the design facilitates practical implementation using simple logic gate circuits, making it suitable for lightweight cryptographic applications. The findings demonstrate the practicality and effectiveness of the proposed method in achieving secure and efficient S–box designs.

8.1 Future work

The method proposed in the paper results in the construction of S–boxes characterized by a high algebraic degree. While this property can be advantageous in certain cryptographic applications due to its contribution to non-linearity and resistance to specific cryptanalytic attacks, it may also introduce significant challenges in practical implementations. In particular, the elevated algebraic degree tends to increase the complexity of achieving a secure Threshold Implementation, as it requires additional resources to prevent leakage and ensure robustness against side-channel attacks [37]. Addressing this challenge and exploring efficient approaches to implement such S–boxes securely is left as an open problem for future research.

Acknowledgement This research was supported by CyberSecurity Malaysia and Fundamental Research Grant Scheme (FRGS) of Universiti Teknikal Malaysia Melaka (FRGS/1/2022/FTMK/F00526) funded by the Ministry of Higher Education, Malaysia.

Conflicts of Interest The authors declare no conflict of interest.

References

[1] G. Ablay (2022). Lyapunov exponent enhancement in chaotic maps with uniform distribution modulo one transformation. *Chaos Theory and Applications*, *4*(1), 45–58. https://doi.org/10.51537/chaos.1069002.

[2] J. A. Aboytes-González, C. Soubervielle-Montalvo, I. Campos-Cantón, O. E. Perez-Cham & M. T. Ramírez-Torres (2023). Method to improve the cryptographic properties of S–boxes. *IEEE Access*, *11*, 99546–99557. https://doi.org/10.1109/ACCESS.2023.3313180.

[3] M. Ahmad, R. Alkanhel, W. El-Shafai, A. D. Algarni, F. E. Abd El-Samie & N. F. Soliman (2022). Multi-objective evolution of strong S–boxes using non-dominated sorting genetic algorithm-II and chaos for secure telemedicine. *IEEE Access*, *10*, 112757–112775. https://doi.org/10.1109/ACCESS.2022.3209202.

[4] A. R. Alharbi, S. S. Jamal, M. F. Khan, M. A. Gondal & A. A. Abbasi (2023). Construction and optimization of dynamic S–boxes based on gaussian distribution. *IEEE Access*, *11*, 35818–35829. https://doi.org/10.1109/ACCESS.2023.3262313.

[5] T. S. Ali & R. Ali (2022). A novel color image encryption scheme based on a new dynamic compound chaotic map and S–box. *Multimedia Tools and Applications*, *81*(15), 20585–20609. https://doi.org/10.1007/s11042-022-12268-6.

[6] E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, F. Mendel, B. Mennink, N. Mouha, Q. Wang & K. Yasuda. PRIMATEs v1.02: Submission to the CAESAR Competition 2014. https://competitions.cr.yp.to/round2/primatesv102.pdf.

[7] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima & T. Tokita (2001). Camellia: A 128-bit block cipher suitable for multiple platforms – Design and analysis. In *Selected Areas in Cryptography*, 7th Annual International Workshop, SAC 2000 Waterloo, Ontario, Canada, August 14–15, 2000 Proceedings 7 pp. 39–56. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44983-3_4.

[8] F. Artuğer (2023). A new S–box generator algorithm based on 3D chaotic maps and whale optimization algorithm. *Wireless Personal Communications*, *131*(2), 835–853. https://doi.org/10.1007/s11277-023-10456-7.

[9] D. Bartoli & M. Timpanella (2022). On a conjecture on APN permutations. *Cryptography and Communications*, *14*(4), 925–931. https://doi.org/10.1007/s12095-022-00558-7.

[10] Y. Belenky, V. Bugaenko, L. Azriel, H. Chernyshchyk, I. Dushar, O. Karavaev, O. Maksimenko, Y. Ruda, V. Teper & Y. Kreimer (2022). Redundancy AES masking basis for attack mitigation (RAMBAM). *IACR Transactions on Cryptographic Hardware and Embedded Systems*, *2022*(2), 69–91. https://doi.org/10.46586/tches.v2022.i2.69-91.

[11] G. Bertoni, J. Daemen, M. Peeters & G. Van Assche (2013). Keccak. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science* pp. 313–314. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38348-9_19.

[12] E. Biham & A. Shamir (1991). Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, *4*, 3–72. https://doi.org/10.1007/BF00630563.

[13] A. Bogdanov, G. Knežević Leander, D. Toz, K. Varici & V. Ingrid (2011). SPONGENT: A lightweight hash function. In *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13*, volume 6917 of *Lecture Notes in Computer Science* pp. 312–325. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-23951-9_21.

[14] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin & C. Vikkelsoe (2007). PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007*, volume 4727 of *Lecture Notes in Computer Science* pp. 450–466. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74735-2_31.

[15] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen & T. Yalçın (2012). PRINCE – A low-latency block cipher for pervasive computing applications. In *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*, volume 7658 of *Lecture Notes in Computer Science* pp. 208–225. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-34961-4_14.

[16] C. Carlet, M. Djurasevic, D. Jakobovic & S. Picek (2020). A search for additional structure: The case of cryptographic S–boxes. In *Parallel Problem Solving from Nature – PPSN XVI*, volume 12270 of *Lecture Notes in Computer Science* pp. 343–356. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-58115-2_24.

[17] J. Daemen & V. Rijmen (2020). *The Design of Rijndael: The Advanced Encryption Standard (AES)*. Information Security and Cryptography. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-60769-5.

[18] C. Dobraunig, M. Eichlseder, F. Mendel & M. Schläffer (2021). Ascon v1. 2: Lightweight authenticated encryption and hashing. *Journal of Cryptology*, *34*, Article ID: 33. https://doi.org/10.1007/s00145-021-09398-9.

[19] M. Durasevic, D. Jakobovic, L. Mariot, S. Mesnager & S. Picek (2023). On the evolution of boomerang uniformity in cryptographic S–boxes. In *Applications of Evolutionary Computation*, volume 13989 of *Lecture Notes in Computer Science* pp. 237–252. Springer Nature Switzerland, Cham. https://doi.org/10.1007/978-3-031-30229-9_16.

[20] S. G. Garba, A. A. Obiniyi, M. A. Ibrahim & B. I. Ahmad (2022). Towards finding an optimal S–box for lightweight block cipher. In *2022 5th Information Technology for Education and Development (ITED)*, pp. 1–8. IEEE, Abuja, Nigeria. https://doi.org/10.1109/ITED56637.2022.10051435.

[21] S. Ibrahim & A. M. Abbas (2020). A novel optimization method for constructing cryptographically strong dynamic S–boxes. *IEEE Access*, *8*, 225004–225017. http://dx.doi.org/10.1109/ACCESS.2020.3045260.

[22] H. Isa, N. Jamil & M. R. Z'aba (2015). Improved S–box construction from binomial power functions. *Malaysian Journal of Mathematical Sciences*, *9*(S), 21–35.

[23] K. Kim, T. Matsumoto & H. Imai (1991). A recursive construction method of S–boxes satisfying strict avalanche criterion. In *Advances in Cryptology-CRYPTO'90: Proceedings 10*, volume 537 of *Lecture Notes in Computer Science* pp. 565–574. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-38424-3_39.

[24] L. Li, J. Liu, Y. Guo & B. Liu (2022). A new S–box construction method meeting strict avalanche criterion. *Journal of Information Security and Applications*, *66*, Article ID: 103135. https://doi.org/10.1016/j.jisa.2022.103135.

[25] H. Liu, J. Liu & C. Ma (2023). Constructing dynamic strong S–box using 3D chaotic map and application to image encryption. *Multimedia Tools and Applications*, *82*(16), 23899–23914. https://doi.org/10.1007/s11042-022-12069-x.

[26] J. Liu, S. Mesnager & L. Chen (2017). On the nonlinearity of S–boxes and linear codes. *Cryptography and Communications*, *9*(3), 345–361. https://doi.org/10.1007/s12095-015-0176-z.

[27] X. Liu, X. Tong, Z. Wang & M. Zhang (2022). Uniform non-degeneracy discrete chaotic system and its application in image encryption. *Nonlinear Dynamics*, *108*(1), 653–682. https://doi.org/10.1007/s11071-021-07198-1.

[28] A. Mahboob, M. Asif, M. Nadeem, A. Saleem, S. M. Eldin & I. Siddique (2022). A cryptographic scheme for construction of substitution boxes using quantic fractional transformation. *IEEE Access*, *10*, 132908–132916. https://doi.org/10.1109/ACCESS.2022.3230141.

[29] A. Manzoor, A. H. Zahid & M. T. Hassan (2022). A new dynamic substitution box for data security using an innovative chaotic map. *IEEE Access*, *10*, 74164–74174. https://doi.org/10.1109/ACCESS.2022.3184012.

[30] M. Matsui (1993). Linear cryptanalysis method for DES cipher. In *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science* pp. 386–397. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48285-7_33.

[31] A. Mihalkovich & M. Levinskas (2021). Avalanche effect and bit independence criterion of perfectly secure Shannon cipher based on matrix power. *Mathematical Models in Engineering*, *7*(3), 50–53. https://doi.org/10.21595/mme.2021.22234.

[32] N. F. Mohd Esa, S. F. Abdul-Latip & N. A. Abu (2022). A new design of substitution box with ideal strict avalanche criterion. *Malaysian Journal of Mathematical Sciences*, *16*(4), 697–715. https://doi.org/10.47836/mjms.16.4.04.

[33] P. Morawiecki, K. Gaj, E. Homsirikamol, K. Matusiewicz, J. Pieprzyk, M. Rogawski, M. Srebrny & M. Wójcik (2014). ICEPOLE: High-speed, hardware-oriented authenticated encryption. In *Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16*, volume 8731 of *Lecture Notes in Computer Science* pp. 392–413. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44709-3_22.

[34] NIST. Selects winner of secure hash algorithm (SHA-3) competition. Technical report National Institute of Standards and Technology 2012. https://www.nist.gov/news-events/news/2012/10/nist-selects-winner-secure-hash-algorithm-sha-3-competition.

[35] NIST. Lightweight cryptography standardization process: NIST selects ascon. Technical report National Institute of Standards and Technology 2023. https://www.nist.gov/news-events/news/2023/02/lightweight-cryptography-standardization-process-nist-selects-ascon.

[36] D. Penazzi & M. Montes. Shamash (and shamashash) 2019. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/ShamashAndShamashash-spec.pdf.

[37] E. Piccione, S. Andreoli, L. Budaghyan, C. Carlet, S. Dhooghe, S. Nikova, G. Petrides & V. Rijmen (2023). An optimal universal construction for the threshold implementation of bijective S–boxes. *IEEE Transactions on Information Theory*, *69*(10), 6700–6710. https://doi.org/10.1109/TIT.2023.3287534.

[38] B. Rashidi (2021). Compact and efficient structure of 8-bit S–box for lightweight cryptography. *Integration*, *76*, 172–182. https://doi.org/10.1016/j.vlsi.2020.10.009.

[39] B. Rashidi (2021). Lightweight 8-bit S–box and combined S–box/S–box^{-1} for cryptographic applications. *International Journal of Circuit Theory and Applications*, *49*(8), 2348–2362. https://doi.org/10.1002/cta.3041.

[40] B. Rashidi (2023). Lightweight cryptographic S–boxes based on efficient hardware structures for block ciphers. *The ISC International Journal of Information Security*, *15*(1), 137–151. https://doi.org/10.22042/isecure.2022.275268.646.

[41] B. Rashidi (2024). Fault-tolerant and error-correcting 4-bit S–boxes for cryptography applications with multiple errors detection. *The Journal of Supercomputing*, *80*, 1464–1490. https://doi.org/10.1007/s11227-023-05530-7.

[42] C.-I. Rîncu & V.-G. Iana (2014). S–box design based on chaotic maps combination. In *2014 10th International Conference on Communications* (*COMM*), pp. 1–4. IEEE, Bucharest, Romania. https://doi.org/10.1109/ICComm.2014.6866741.

[43] W. L. Teng, I. Salam, W. C. Yau, J. Pieprzyk & R. C. W. Phan (2022). Cube attacks on round-reduced TinyJAMBU. *Scientific Reports*, *12*(1), Article ID: 5317. https://doi.org/10.1038/s41598-022-09004-3.

[44] V. A. Thakor, M. A. Razzaque, A. D. Darji & A. R. Patel (2023). A novel 5-bit S–box design for lightweight cryptography algorithms. *Journal of Information Security and Applications*, *73*, Article ID: 103444. https://doi.org/10.1016/j.jisa.2023.103444.

[45] S. Tian, C. Boura & L. Perrin (2020). Boomerang uniformity of popular S–box constructions. *Designs, Codes and Cryptography*, *88*(9), 1959–1989. https://doi.org/10.1007/s10623-020-00785-0.

[46] M. Wang, H. Liu & M. Zhao (2023). Construction of a non-degeneracy 3D chaotic map and application to image encryption with keyed S–box. *Multimedia Tools and Applications*, *82*(22), 34541–34563. https://doi.org/10.1007/s11042-023-14988-9.

[47] J. Wolfmann (2017). Sequences of bent functions and near-bent functions. *Cryptography and Communications*, *9*(6), 729–736. https://doi.org/10.1007/s12095-017-0212-2.

[48] Y. Q. Zhang, J. L. Hao & X. Y. Wang (2020). An efficient image encryption scheme based on S–boxes and fractional-order differential logistic map. *IEEE Access*, *8*, 54175–54188. https://doi.org/10.1109/ACCESS.2020.2979827.

[49] W. Zhao, Z. Chang, C. Ma & Z. Shen (2023). A pseudorandom number generator based on the chaotic map and quantum random walks. *Entropy*, *25*(1), Article ID: 166. https://doi.org/10.3390/e25010166.

[50] J. Zheng & H. Hu (2022). A highly secure stream cipher based on analog-digital hybrid chaotic system. *Information Sciences*, *587*(C), 226–246. https://doi.org/10.1016/j.ins.2021.12.030.

[51] S. Zhou, Y. Qiu, X. Wang & Y. Zhang (2023). Novel image cryptosystem based on new 2D hyperchaotic map and dynamical chaotic S–box. *Nonlinear Dynamics*, *111*(10), 9571–9589. https://doi.org/10.1007/s11071-023-08312-1.

9 Appendix

9.1 Differential distribution and linear approximation

Tables 12 and 13 show the difference distribution table and linear approximation table of the proposed S–box.

Table 12: Differential distribution table of the proposed S–box.

	$S5(\boldsymbol{x})$																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	4	2	4	0	0	0	2	2	0	0	0	0	0	2	0	0	0	2	4	0	0	0	2	2	0	0	0	0	0	6
2	0	0	2	0	2	0	0	2	0	4	0	0	2	0	2	2	0	2	2	2	2	2	0	0	2	0	2	0	0	0	0	2
3	0	0	0	0	0	2	0	2	0	0	2	6	0	2	2	0	0	0	0	0	0	2	0	2	0	0	2	2	0	6	2	0
4	0	2	4	2	2	0	0	2	0	0	2	0	0	0	2	0	0	0	0	0	2	2	0	0	2	4	0	0	2	0	0	4
5	0	0	0	2	0	0	2	2	0	0	0	6	0	4	0	0	0	0	0	2	0	0	2	2	0	0	0	2	0	4	4	0
6	0	0	0	2	0	0	0	0	0	0	2	0	0	4	4	0	0	0	0	2	0	4	4	0	0	0	2	0	0	4	4	0
7	0	4	2	0	0	2	0	0	0	2	2	0	4	0	0	0	0	0	2	0	0	2	0	4	0	2	2	0	0	0	4	0
8	0	4	0	0	0	2	0	2	2	0	0	2	4	0	0	0	0	0	0	2	0	0	6	0	2	0	2	2	0	2	0	0
9	0	0	2	0	2	0	0	4	2	0	0	0	0	0	2	4	0	2	2	0	2	0	0	2	0	0	0	0	0	0	4	4
10	0	0	4	2	0	4	0	0	0	0	0	0	2	0	0	0	0	0	0	0	4	2	0	0	0	4	6	0	0	0	0	4
11	0	2	0	2	0	0	4	2	0	0	0	4	2	0	0	0	0	2	0	0	0	2	2	0	4	0	2	0	2	2	0	0
12	0	0	0	2	0	0	0	0	0	2	2	0	2	0	0	4	0	0	0	0	4	2	0	0	4	6	0	0	4	0	0	0
13	0	2	0	2	0	0	2	2	2	0	2	0	2	2	0	0	0	2	0	0	0	2	4	0	2	0	0	4	2	0	0	0
14	0	2	0	2	0	2	4	0	2	0	0	2	2	0	0	0	0	2	0	0	0	0	2	2	2	0	2	2	2	2	0	0
15	0	4	2	2	2	0	0	2	2	2	0	0	0	0	0	0	0	2	6	2	2	0	0	0	0	2	0	0	0	0	2	0
16	0	0	0	2	0	0	4	0	0	2	4	0	0	0	0	4	4	0	0	2	0	4	0	0	0	2	0	0	4	0	0	0
17	0	0	2	0	0	2	4	0	2	0	0	0	0	0	2	4	0	2	0	0	0	0	2	4	0	0	0	2	2	0	2	2
18	0	0	2	2	2	0	4	0	4	0	0	0	2	0	0	0	0	0	2	2	2	0	0	0	0	0	0	4	6	0	0	0
19	0	2	0	0	2	0	0	0	2	0	0	2	2	0	0	2	2	0	2	2	2	0	0	4	2	0	0	2	0	2	2	0
20	0	2	0	0	0	2	0	0	2	4	0	0	2	4	0	0	0	4	2	0	0	4	2	0	0	0	0	2	0	0	0	2
21	0	2	4	0	2	0	0	0	0	0	0	0	0	2	2	0	2	4	2	2	0	2	2	2	0	0	0	0	0	2	2	0
22	0	0	0	0	2	2	2	0	2	2	2	2	0	0	2	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	0	2
23	0	0	4	0	0	2	0	0	2	0	2	4	0	0	0	2	4	4	0	0	0	2	0	0	2	0	2	0	0	0	0	2
24	0	2	0	2	0	0	2	0	4	0	2	0	0	2	2	0	2	4	0	0	0	0	0	0	2	6	0	0	2	0	0	0
25	0	0	0	0	2	0	0	4	0	4	4	0	0	4	6	0	0	0	0	0	0	0	0	2	2	0	0	2	0	0	2	0
26	0	2	0	2	2	2	2	0	0	4	0	0	0	0	0	2	8	0	2	2	0	0	0	2	0	0	0	0	2	0	0	0
27	0	0	0	2	4	0	0	0	2	0	0	2	2	4	0	0	0	0	2	2	0	0	2	0	0	2	2	2	2	0	0	2
28	0	2	0	2	0	2	2	0	0	2	2	0	2	2	0	0	2	0	2	4	0	0	0	0	0	0	4	0	0	0	2	2
29	0	0	0	0	0	4	0	4	0	0	2	0	0	0	4	2	2	2	4	2	0	0	2	0	0	0	2	2	0	0	0	0
30	0	0	0	0	2	4	0	2	0	0	0	2	2	0	0	4	4	0	2	0	2	0	2	2	0	0	0	0	0	2	2	0
31	0	0	0	0	2	0	0	2	0	2	2	0	0	2	2	0	2	0	0	2	6	0	0	2	2	0	0	2	0	4	0	0

Table 13: Linear approximation table of the proposed S–box.

	$S5\,(\boldsymbol{x})$																															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	4	6	-2	4	0	-2	-2	2	-2	0	0	-2	2	0	8	2	-2	0	0	2	-2	-4	-4	4	0	2	2	4	0	-2	-2
2	0	0	0	0	-2	2	6	2	-2	-2	2	2	4	0	0	4	-4	4	0	0	-2	-6	2	-2	2	2	2	-6	4	0	4	0
3	0	0	6	2	6	2	0	0	0	0	2	-2	-2	2	4	-4	-2	-2	0	4	-4	0	2	2	-2	-2	-4	0	4	0	6	-2
4	0	4	2	-6	-2	-2	-4	0	0	-4	-2	-2	-2	-2	0	4	-4	4	-2	2	-2	2	4	4	-4	-4	2	-2	-2	2	0	0
5	0	-4	-4	0	-6	2	-2	-2	-6	-2	2	-2	-4	4	4	4	-2	-2	2	2	0	4	-4	0	0	0	0	0	2	-2	2	-2
6	0	-4	-2	-2	0	-4	-6	2	-2	2	-4	4	-2	2	0	0	4	4	-2	2	-4	-4	2	-2	2	2	-4	0	2	2	0	-4
7	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	-8	0	-2	2	2	6	2	-2	-2	2	-2	2	-6	-2	2	-2	-2	2
8	0	0	0	0	0	-4	0	4	0	-4	-4	0	-4	-4	0	0	-2	-6	2	-2	6	-2	2	2	-2	6	-2	-2	2	-2	2	-2
9	0	-4	-2	-2	0	0	2	-2	-2	-2	0	-4	2	-2	0	0	-4	-4	-2	2	0	-4	6	-2	2	-2	0	8	2	2	-4	0
10	0	0	-4	-4	2	2	-2	-2	6	-6	2	-2	-4	0	0	-4	-2	2	2	-2	-4	0	0	-4	4	4	0	0	-2	-2	2	2
11	0	0	2	-2	-2	6	-4	0	-4	0	6	-2	-2	-6	-4	-4	4	0	-2	-2	2	-2	0	0	0	0	2	-2	2	2	0	-4
12	0	4	-2	6	-2	2	0	0	4	4	2	-2	-2	-2	0	4	-2	2	0	0	0	4	6	-2	2	2	-4	0	0	0	-2	-6
13	0	4	0	-4	-2	2	-2	2	2	2	2	2	4	4	4	-4	-4	0	0	4	2	-2	-2	2	-2	6	2	2	0	0	-4	-4
14	0	-4	-2	-2	4	4	-2	2	2	2	4	0	2	-2	0	8	2	-2	0	0	-2	-2	0	4	-4	4	-2	2	-4	0	2	2
15	0	0	0	0	0	4	0	4	0	-4	4	8	-4	4	0	0	0	0	0	0	4	0	4	0	0	-4	-4	0	0	0	-4	4
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-2	-2	2	-6	-6	2	-2	-2	-6	2	-2	-2	6	6	-6	2
17	0	4	-2	-2	0	4	2	2	-2	2	-4	-4	-2	2	0	0	0	-4	2	2	0	-4	-2	-2	2	-2	-4	-4	-6	6	0	0
18	0	4	-4	8	2	2	-2	-2	-2	-6	-2	2	0	0	0	0	2	-2	-2	2	-4	-4	0	0	-4	0	4	0	-2	-2	-2	-2
19	0	4	2	2	-2	-2	-4	0	-4	0	2	2	2	-6	4	0	-4	0	-2	-2	-2	-2	-4	0	4	0	-6	2	-2	-2	0	4
20	0	0	-2	2	-2	-6	0	0	4	-4	6	2	2	-2	0	0	2	-2	4	4	0	0	-2	2	2	-2	0	0	0	8	2	-2
21	0	0	0	0	-2	2	-2	2	2	2	-2	-2	0	-4	4	0	4	0	8	4	-2	-2	2	2	2	-2	2	-2	4	-4	-4	4
22	0	4	-2	-2	4	0	-2	-2	-6	-2	0	0	6	2	0	0	2	2	8	-4	2	2	4	0	0	0	-2	2	0	0	2	-2
23	0	0	0	0	0	0	0	-8	0	0	0	0	0	0	8	0	4	0	-4	0	4	0	4	0	0	4	0	-4	0	4	0	4
24	0	0	0	0	0	4	0	-4	0	4	-4	8	-4	-4	0	0	-4	0	4	0	0	0	0	0	0	0	4	4	0	4	4	0
25	0	4	-2	-2	4	-4	6	2	-6	2	4	0	-6	-2	0	0	2	2	0	4	-2	2	0	0	0	4	2	2	0	0	-2	2
26	0	-4	0	4	6	2	-2	2	-2	-2	-2	-2	0	0	0	0	-4	4	0	0	2	2	-2	6	6	2	2	-2	0	4	-4	0
27	0	4	-2	-2	-2	2	0	0	0	0	-2	2	2	2	-4	0	2	-6	-4	0	-4	4	2	6	6	2	0	0	4	0	2	2
28	0	0	-6	-2	6	-2	-4	0	0	4	2	2	2	-2	0	0	-4	-4	-2	2	2	2	0	-4	0	-4	2	-6	2	-2	0	0
29	0	0	-4	-4	2	2	6	-2	2	-2	-2	2	0	-4	4	0	2	2	-2	-2	0	0	-4	4	0	-4	-4	0	2	-2	-2	-6
30	0	-4	6	-2	0	0	2	-2	-2	-2	0	4	2	-2	0	0	0	-4	2	2	-4	4	2	-2	2	2	0	-4	-6	-2	-4	-4
31	0	0	0	0	0	-4	0	-4	0	4	4	0	-4	4	0	0	-2	-2	2	-6	-2	-6	2	6	2	-2	2	-2	-2	-2	-2	-2

9.2 4–bit S–boxes that yield 5–bit S–boxes with high nonlinearities

Tables 14 and 15 present the 4–bit S–boxes derived from our experiment, which in turn able to generate 5–bit S–boxes exhibiting high degrees of nonlinearity.

Table 14: 4–bit S–boxes that yield 5–bit S–boxes with high nonlinearities (Part 1).

	Input															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	10	8	11	7	6	9	4	5	12	0	14	15	13	2	1	3
	2	0	11	15	6	9	4	13	12	8	14	7	5	10	1	3
	10	0	11	15	6	1	4	13	12	8	14	7	5	2	9	3
	2	0	11	15	14	1	12	5	4	8	6	7	13	10	9	3
	10	8	3	7	6	9	4	13	12	0	14	15	5	2	1	11
	2	8	3	7	14	9	12	5	4	0	6	15	13	10	1	11
	10	8	3	7	14	1	12	5	4	0	6	15	13	2	9	11
	2	0	3	15	14	1	12	13	4	8	6	7	5	10	9	11
	10	8	3	7	14	1	12	5	9	0	13	15	11	4	2	6
	2	0	11	15	14	1	12	5	9	8	13	7	3	4	10	6
	2	0	3	15	14	1	12	13	9	8	5	7	11	4	10	6
	2	8	3	7	14	9	12	5	1	0	13	15	11	4	10	6
	10	0	11	15	6	1	4	13	9	8	5	7	3	12	2	14
	10	8	11	7	6	9	4	5	1	0	13	15	3	12	2	14
	10	8	3	7	6	9	4	13	1	0	5	15	11	12	2	14
	2	0	11	15	6	9	4	13	1	8	5	7	3	12	10	14
	2	0	6	9	11	15	4	13	12	8	5	10	14	7	1	3
	10	8	6	9	11	7	4	5	12	0	13	2	14	15	1	3
	2	0	14	1	11	15	12	5	4	8	13	10	6	7	9	3
	10	0	6	1	11	15	4	13	12	8	5	2	14	7	9	3
	2	8	14	9	3	7	12	5	4	0	13	10	6	15	1	11
	10	8	6	9	3	7	4	13	12	0	5	2	14	15	1	11
	2	0	14	1	3	15	12	13	4	8	5	10	6	7	9	11
	10	8	14	1	3	7	12	5	4	0	13	2	6	15	9	11
	10	8	14	1	3	7	12	5	9	0	11	4	13	15	2	6
	2	0	14	1	3	15	12	13	9	8	11	4	5	7	10	6
	2	0	14	1	11	15	12	5	9	8	3	4	13	7	10	6
	2	8	14	9	3	7	12	5	1	0	11	4	13	15	10	6
	10	0	6	1	11	15	4	13	9	8	3	12	5	7	2	14
	10	8	6	9	3	7	4	13	1	0	11	12	5	15	2	14
	10	8	6	9	11	7	4	5	1	0	3	12	13	15	2	14
	2	0	6	9	11	15	4	13	1	8	3	12	5	7	10	14
	10	11	8	7	6	4	9	5	12	14	0	15	13	1	2	3
	10	11	0	15	6	4	1	13	12	14	8	7	5	9	2	3
$S4$	2	11	0	15	6	4	9	13	12	14	8	7	5	1	10	3
	2	11	0	15	14	12	1	5	4	6	8	7	13	9	10	3
	10	3	8	7	6	4	9	13	12	14	0	15	5	1	2	11
	10	3	8	7	14	12	1	5	4	6	0	15	13	9	2	11
	2	3	8	7	14	12	9	5	4	6	0	15	13	1	10	11
	2	3	0	15	14	12	1	13	4	6	8	7	5	9	10	11
	10	3	8	7	14	12	1	5	9	13	0	15	11	2	4	6
	2	11	0	15	14	12	1	5	9	13	8	7	3	10	4	6
	2	3	0	15	14	12	1	13	9	5	8	7	11	10	4	6
	2	3	8	7	14	12	9	5	1	13	0	15	11	10	4	6

Table 15: 4–bit S–boxes that yield 5–bit S–boxes with high nonlinearities (Part 2).

	Input															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	10	11	0	15	6	4	1	13	9	5	8	7	3	2	12	14
	10	11	8	7	6	4	9	5	1	13	0	15	3	2	12	14
	10	3	8	7	6	4	9	13	1	5	0	15	11	2	12	14
	2	11	0	15	6	4	9	13	1	5	8	7	3	10	12	14
	2	6	0	9	11	4	15	13	12	5	8	10	14	1	7	3
	2	14	0	1	11	12	15	5	4	13	8	10	6	9	7	3
	10	6	0	1	11	4	15	13	12	5	8	2	14	9	7	3
	10	6	8	9	11	4	7	5	12	13	0	2	14	1	15	3
	2	14	0	1	3	12	15	13	4	5	8	10	6	9	7	11
	2	14	8	9	3	12	7	5	4	13	0	10	6	1	15	11
	10	6	8	9	3	4	7	13	12	5	0	2	14	1	15	11
	10	14	8	1	3	12	7	5	4	13	0	2	6	9	15	11
	2	14	0	1	3	12	15	13	9	11	8	4	5	10	7	6
	2	14	0	1	11	12	15	5	9	3	8	4	13	10	7	6
	10	14	8	1	3	12	7	5	9	11	0	4	13	2	15	6
	2	14	8	9	3	12	7	5	1	11	0	4	13	10	15	6
	10	6	0	1	11	4	15	13	9	3	8	12	5	2	7	14
	2	6	0	9	11	4	15	13	1	3	8	12	5	10	7	14
	10	6	8	9	3	4	7	13	1	11	0	12	5	2	15	14
	10	6	8	9	11	4	7	5	1	3	0	12	13	2	15	14
	10	11	6	4	0	15	1	13	12	14	5	9	8	7	2	3
	10	11	6	4	8	7	9	5	12	14	13	1	0	15	2	3
	2	11	6	4	0	15	9	13	12	14	5	1	8	7	10	3
	2	11	14	12	0	15	1	5	4	6	13	9	8	7	10	3
	10	3	6	4	8	7	9	13	12	14	5	1	0	15	2	11
	10	3	14	12	8	7	1	5	4	6	13	9	0	15	2	11
	2	3	14	12	0	15	1	13	4	6	5	9	8	7	10	11
	2	3	14	12	8	7	9	5	4	6	13	1	0	15	10	11
	2	11	14	12	0	15	1	5	9	13	3	10	8	7	4	6
	2	3	14	12	0	15	1	13	9	5	11	10	8	7	4	6
	10	3	14	12	8	7	1	5	9	13	11	2	0	15	4	6
	2	3	14	12	8	7	9	5	1	13	11	10	0	15	4	6
	10	11	6	4	0	15	1	13	9	5	3	2	8	7	12	14
	2	11	6	4	0	15	9	13	1	5	3	10	8	7	12	14
	10	11	6	4	8	7	9	5	1	13	3	2	0	15	12	14
	10	3	6	4	8	7	9	13	1	5	11	2	0	15	12	14
	10	6	11	4	0	1	15	13	12	5	14	9	8	2	7	3
	2	6	11	4	0	9	15	13	12	5	14	1	8	10	7	3
	2	14	11	12	0	1	15	5	4	13	6	9	8	10	7	3
	10	6	11	4	8	9	7	5	12	13	14	1	0	2	15	3
	2	14	3	12	0	1	15	13	4	5	6	9	8	10	7	11
$S4$	10	6	3	4	8	9	7	13	12	5	14	1	0	2	15	11
	10	14	3	12	8	1	7	5	4	13	6	9	0	2	15	11
	2	14	3	12	8	9	7	5	4	13	6	1	0	10	15	11
	2	14	3	12	0	1	15	13	9	11	5	10	8	4	7	6
	2	14	11	12	0	1	15	5	9	3	13	10	8	4	7	6
	10	14	3	12	8	1	7	5	9	11	13	2	0	4	15	6
	2	14	3	12	8	9	7	5	1	11	13	10	0	4	15	6
	10	6	11	4	0	1	15	13	9	3	5	2	8	12	7	14
	2	6	11	4	0	9	15	13	1	3	5	10	8	12	7	14
	10	6	3	4	8	9	7	13	1	11	5	2	0	12	15	14
	10	6	11	4	8	9	7	5	1	3	13	2	0	12	15	14