IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 7, 2020, accepted December 13, 2020, date of publication December 16, 2020, date of current version December 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3045260

A Novel Optimization Method for Constructing Cryptographically Strong Dynamic S-Boxes

SALEH IBRAHIM [1,2] AND ALAA M. ABBAS [1,3]
[1]Department of Electrical Engineering, College of Engineering, Taif University, Al-Hawiya 21974, Saudi Arabia
[2]Computer Engineering Department, Faculty of Engineering, Cairo University, Giza 12613, Egypt
[3]Department of Electronics and Electrical Communications, Faculty of Electronic Engineering, Menoufia University, Menouf 32952, Egypt

Corresponding author: Saleh Ibrahim (saleh@eng.cu.edu.eg)

ABSTRACT The resistance of S-box-based cryptosystems to linear cryptanalysis is often determined by the nonlinearity (NL) and the linear approximation probability (LAP) of the underlying S-box. Constructing dynamic bijective S-boxes with high nonlinearity is a challenging problem. In this paper, we propose a novel S-box construction method based on the concept of constrained optimization. The proposed method uses a random-restart hill-climbing algorithm to construct randomized S-boxes and maximize the nonlinearity of each Boolean function under bijectivity constraints. The proposed algorithm dramatically reduced the S-box construction time. Compared to recent S-box construction methods, the proposed method strikes a better balance among the three design objectives of dynamic S-boxes, namely, cryptographic strength, dynamicity, and speed of construction. On the average, the proposed method constructs a new dynamic 8×8 S-box with NL=112 every 118 ms, whereas a NL=110 S-box can be generated in 5.3 ms, which makes it suitable for real time applications. The proposed method also constructs 8×8 S-boxes with NL=114, which is among the highest reported in literature. Moreover, we demonstrate the extensibility of the proposed constrained optimization formulation to improve other S-box design criteria. Namely, we propose an algorithm to optimize the LAP of an S-box while preserving its NL and bijectivity.

INDEX TERMS Cryptography, bijective substitution boxes, dynamic s-boxes, nonlinearity, constrained optimization.

I. INTRODUCTION

Image encryption cryptosystems should produce a certain confusion and diffusion level in the cipher image. One of the most important blocks in many encryption systems is the S-box. An S-box must satisfy certain design criteria pertaining to its resistance to a variety of cryptanalysis attacks [1]. Among these criteria, the bitwise functional nonlinearity (NL) and the linear approximation probability (LAP) tests determine the S-box resistance to linear cryptanalysis [2]. Since an S-box is the only nonlinear component in many ciphers, S-boxes with high nonlinearity are required. For ciphers that depend on secret key-dependent dynamic S-boxes, S-box construction faces many challenges. First, pseudorandom constructions of S-box tend to have an unsatisfactory nonlinearity distribution. Therefore, S-boxes must be carefully designed to guarantee high nonlinearity. Second, to prevent adversaries from obtaining enough information to infer the encryption key, the dynamic S-box must be changed frequently with each session, message or even with each message block. This requirement enforces stringent constraints on the S-box construction speed in real-time applications. Finally, there should be sufficiently many potential S-boxes to choose from to prevent brute-force guessing the secret S-box. Satisfying all three requirements simultaneously is still an open research issue.

When designing a dynamic S-box construction method, three objectives should be taken into account: 1) the quality of generated S-box (e.g., nonlinearity), 2) the dynamicity of the constructed S-boxes, and 3) the speed of S-box construction.

The quality of the S-box determines its cryptographic strength and consequently the resistance of the cipher using the S-box to resist various cryptanalysis attacks. Clearly, compromising on the quality of a dynamic S-box requires additional attention the design of the cipher to ensure its resistance to relevant cryptanalysis methods [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Ruofei Ma.

The dynamicity of constructed S-boxes indicates the number of different S-boxes a method can construct. Therefore, the dynamicity determines the search space facing an adversary attempting to discover the S-box function. For key-dependent S-boxes, this search space translates into a corresponding key space. The larger the potential number of S-boxes generated by a construction method the larger the key space added to the corresponding encryption method using the dynamic S-box [4]. Cipher designers using S-box construction methods that generate a significantly limited number of S-boxes must consider that the S-box function may be discovered by an adversary using brute-force.

The last design objective is reducing the S-box construction time. This is an important factor in ciphers that depend for their security on the dynamicity of the S-box. In this case, frequently changing the S-box function is crucial for thwarting cryptanalysis attacks as well as limiting the damage caused by a successful attack [4]. The faster an S-box construction method can construct a dynamic S-box, the more frequently the S-box can be changed. In other words, a fast dynamic-S-box construction method gives the cipher designer the flexibility to design more secure ciphers.

To the extent of our knowledge, existing methods for constructing S-boxes suffer in varying degrees from at least one of the following shortcomings: 1) relatively low nonlinearity, 2) limited number of potential S-boxes, or 3) high construction time.

S-box construction methods based on chaotic maps or other pseudorandom generators such as [5]–[20], are basically fast, but the average S-box nonlinearity generated by these methods is below $NL = 100$ and S-boxes with $NL = 108$ can hardly be constructed.

On the other hand, algebraic methods, such as [1], [2], [4], [21]–[27], use efficient algebraic construction to generate 8×8 S-boxes with high nonlinearity reaching $NL = 112$. However, they can only generate a limited number of different S-boxes. For instance, [27] generates only one S-box, [23] generates only 16 base S-boxes, [26] constructs 256 S-boxes and [4] constructs 462422016 S-boxes.

Optimization-based methods, such as [28]–[36], can also achieve relatively high nonlinearity, usually $NL \geq 110$ up to $NL = 114$, and have the advantage of generating a virtually unlimited number of S-boxes. Their main disadvantage, however, is the long execution time that disqualify them for real-time construction of highly dynamic strong S-boxes. For instance, the method presented in [29], divides the problem of constructing an $n \times n$ S-box into simpler subproblems of designing n balanced Boolean function. However, the problem of fitting n balanced Boolean functions into a bijective S-box is much harder. The authors of [29] used genetic algorithms to approach this problem and the best 8×8 S-box nonlinearity achieved was NL=110, which required evolving hundreds of generations, thus taking too long to qualify for real time construction of dynamic S-boxes.

In this paper, we present a novel method to strike a balance between these three objectives. The proposed S-box construction method is capable of fast generation of an unlimited number of randomized S-boxes with high nonlinearity. Figure 1 illustrates dynamic S-box construction objectives and how the three broad categories of dynamic S-box construction methods fare with these objectives. The proposed S-box construction method combines the advantages of the three categories to achieve the speed of chaotic and algebraic methods, and the dynamicity and quality of optimization methods.

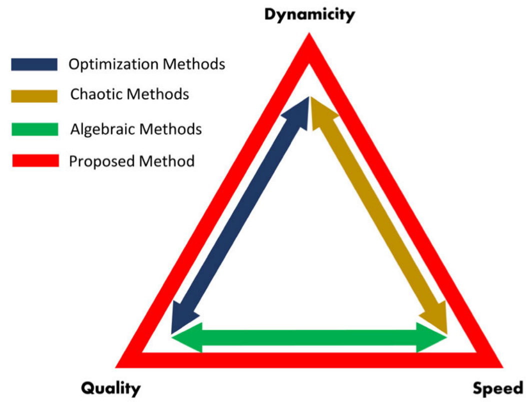

FIGURE 1. Design objectives of the proposed dynamic S-box construction method in comparison to existing methods.

The proposed method can generate a dynamic 8×8 S-boxes with high nonlinearity up to $NL = 110$ in a few milliseconds and with $NL = 112$ in just 118 milliseconds, on the average. This achievement makes the proposed method applicable to real time encryption schemes employing dynamic S-boxes. The dynamic S-boxes in many encryption schemes such as [7], [25], [37]–[44] can be replaced with higher security S-boxes generated by the proposed method.

This dramatic achievement is possible due to a combination of two novel algorithms. First, we propose a novel dynamic-programming algorithm for constructing a randomized Boolean function to fit within a partially constructed bijective S-box. Second, we propose a novel formulation, by deriving nonlinearity improvement constraints, such that when a Boolean function is modified under these constraints, its nonlinearity must improve. These constraints, as well as bijectivity constraints, enable our second algorithm to efficiently maximize the nonlinearity of the constructed coordinate Boolean function. By utilizing these two algorithms, we propose a third algorithm to incrementally construct a bijective S-box with high nonlinearity. As an icing on the cake, we demonstrate the reusability of our constrained optimization method to modify Biryukov's LAP refinement algorithm [45] such that nonlinearity is preserved during LAP improvement.

The rest of the paper is organized as follows. Section 2 presents the S-box security criteria and reviews existing S-box construction methods relevant to the proposed method. In Section 3, we present the basic constrained optimization formulation and the proposed algorithms. Section 4 evaluates the efficiency of the proposed algorithms, as well as the properties of constructed S-boxes and compares results with related methods. In Section 5, we present some concluding remarks and future prospect.

II. S-BOX DESIGN CRITERIA

A general $n \times m$ cryptographic substitution box, S, is a function from an n-bit input to an m-bit output, i.e., $S : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^m$, where $\mathbb{Z}_2 = \{0, 1\}$ and $\mathbb{Z}_2^n$ is an n-dimensional bit vector. S-boxes are a crucial component for many cryptographic applications. Therefore, numerous research papers have been devoted to secure and efficient S-box design [8]. In this section, we review the S-box design criteria and give a brief review of the most recent and relevant advances in S-box construction methods.

The S-Box design criteria determine the S-box suitability for a certain cryptographic application. These criteria include bijectivity, nonlinearity, linear approximation probability, differential approximation probability, input-output strict avalanche criterion, output-bit independence criterion.

A. BIJECTIVITY

A bijective S-box, $S : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^m$, is an invertible function, i.e., $\exists S^{-1} : \mathbb{Z}_2^m \rightarrow \mathbb{Z}_2^n$,such that $S^{-1}(S(x)) = x, \forall x \in \mathbb{Z}_2^n$. This implies that $m = n$, and each of the 2^n output patterns $y \in \mathbb{Z}_2^n$ must appear only once in the function output, i.e., $S(x_0) \neq S(x_1), \forall x_0, x_1 \in \mathbb{Z}_2^n, x_0 \neq x_1$. Most S-box-based encryption schemes require bijective S-boxes.

Bijectivity is closely related to the concept of balanced Boolean functions. The j th output bit of S, denoted f_j, is a Boolean function $f_j : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$. A balanced Boolean function, f, has an equal number of inputs that map to zero and that map to one. In other words, $\#\{i \in \mathbb{Z}_2^n | f(i) = 0\} = \#\{i \in \mathbb{Z}_2^n | f(i) = 1\} = 2^{n-1}$, where the operator means the set cardinality. It can easily be shown that a necessary condition for S to be bijective, is that its coordinate Boolean functions to be balanced. A sufficient condition for bijectivity of an S-box with coordinate functions $f_0, f_1, \ldots, f_{n-1}$, is

$$\#\{x \in \mathbb{Z}_2^n | f_j(x) = c_j, \forall j, 0 \leq j < m\} = 2^{n-m}, \quad \forall\, 0 < m \leq n, (c_0, c_1, \ldots, c_{m-1}) \in \mathbb{Z}_2^m \tag{1}$$

A proof of (1) can be found in [29].

B. NONLINEARITY (NL)

The nonlinearity of an S-box measures the minimum distance between the set of coordinate Boolean functions of the S-box, $\{f_j\}$, and the set of all linear Boolean functions $\{L_\alpha(x) : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2 = \alpha \odot x, \alpha \in \mathbb{Z}_2^n\}$. The $\odot$ operator represents the modulo-2 dot product of the two bit vectors, i.e., $\alpha \odot x = \oplus_{0 \leq i < n}(\alpha_i x_i)$, where $\oplus$ is the modulo-2 addition operator.

To measure the Hamming distance between a Boolean function, f, and a linear Boolean function, L_α, we first calculate the correlation of the two functions using the Walsh-Hadamard transform (WHT),

$$\hat{f}(\alpha) = \frac{1}{2} \sum_{x \in \mathbb{Z}_2^n} (-1)^{f(x) \oplus (\alpha \odot x)} \tag{2}$$

The maximum absolute correlation between f and any linear Boolean function L_α is

$$\hat{F} = \max_{\alpha \varepsilon \mathbb{Z}_2^n} \left|\hat{f}(\alpha)\right| \tag{3}$$

The minimum distance between f and all linear Boolean functions, and thus the nonlinearity of f, is determined by the linear function with the maximum absolute correlation to f,

$$NL(f) = 2^{n-1} - \hat{F} \tag{4}$$

The maximum nonlinearity of a Boolean function is $2^{n-1} - 2^{(n/2)-1}$, but such high nonlinearity contradicts the bijectivity criterion [46]. The nonlinearity of an S-box function is the minimum nonlinearity of its Boolean functions.

$$NL(S) = \min_{0 \leq j < n} NL(f_j) \tag{5}$$

C. LINEAR APPROXIMATION PROBABILITY (LAP)

The LAP criterion measures the probability of obtaining a linear Boolean approximation of the S-box. LAP can be viewed as the maximum probability of approximating a linear Boolean function of the S-box output, $L_\beta(S(x)) = \beta \odot S(x)$ by a linear Boolean function $L_\alpha(x) = \alpha \odot x$. To resist the linear approximation attack, $L_\beta(S(x))$ should be as different from $L_\alpha(x)$ as possible for all $\alpha \in \mathbb{Z}_2^n$, and $\beta \in \mathbb{Z}_2^{n*}$, where $\mathbb{Z}_2^n$ is the set of nonzero elements of $\mathbb{Z}_2^{n*}$. Therefore, we aim to minimize the linear approximation probability given by the equation

$$LAP(S) = \frac{1}{2^n} \max_{a \in \mathbb{Z}_2^n, \beta \in \mathbb{Z}_2^{n*}} \Big| \{\# x \in \mathbb{Z}_2^n | \alpha \odot x = \beta \odot S(x)\} - 2^{n-1} \Big|. \tag{6}$$

The value of LAP should be close to zero to indicate that the output bits of the S-box are uncorrelated to the input bits. The authors of [45] proposed a combinatorial optimization algorithm to iteratively improve the LAP of an S-box by swapping two elements. Their method preserves bijectivity but doesn't preserve nonlinearity. This means that while attempting to improve LAP, their method may well decrease NL.

D. OTHER S-BOX DESIGN CRITERIA

Additional S-box tests, namely differential uniformity (DU), bit-independence criterion (BIC) and strict avalanche criterion (SAC), will be presented briefly.

The DU test measures the maximum number of S-box mappings that can be calculated by applying a constant difference to input bits and a constant difference to output bits.

$$DU(S) = \max_{a \in \mathbb{Z}_2^{n*}, \beta \in \mathbb{Z}_2^{n*}} \#\{x \in \mathbb{Z}_2^n | S(x \oplus \alpha) \oplus S(x) = \beta\} \tag{7}$$

The value of DU should be close to zero to indicate that the S-box is highly immune to differential attacks.

The BIC test measures the correlation between each pair of output bits when one input bit changes. The result of the test is an $n \times n$ symmetric matrix excluding the diagonal. The matrix element in row i and column j indicates the dependence between bits i and j of the S-box output.

$$BIC_{i,j}(S) = \frac{1}{n2^n} \sum_{1 \le k \le n} \#\left\{x \in \mathbb{Z}_2^n \mid V_k^i(x) = V_k^j(x)\right\}, \quad (8)$$

where $V_k^i(x) = 2^{i-1} \odot \left(S\left(x \oplus 2^{k-1}\right) \oplus S(x)\right)$.

The SAC test measures the probability of changing each output bit when an input bit is changed. The result of this test is an $n \times n$ matrix. The matrix element in row i and column j indicates the probability that output bit j changes when input bit i changes.

$$SAC_{i,j}(S) = \frac{1}{2^n} \#\left\{x \in \mathbb{Z}_2^n \mid V_i^j(x) \ne 0\right\} \quad (9)$$

The optimal value of each element of the SAC matrix is 0.5, which means that output bits are completely uncorrelated to input bits.

III. PROPOSED S-BOX CONSTRUCTION METHOD

The proposed S-box construction method consists of four algorithms. Algorithm 1 constructs a randomized n-input coordinate Boolean function satisfying the bijectivity constraints. Algorithm 2 attempts to improve the NL of a given coordinate Boolean function iteratively to reach the required value, NL_{min}, while maintaining bijectivity constraints. Algorithm 3 constructs an $n \times n$ S-box incrementally from coordinate functions by invoking Algorithm 1 and Algorithm 2 and appending the resulting coordinate Boolean function to the constructed S-box. Since Algorithm 2 may occasionally get stuck at a local maximum and fail to achieve the required NL, Algorithm 3 employs a random-restart hill-climbing technique to repeatedly invoke Algorithm 1 and Algorithm 2 until the required NL is achieved. Finally, Algorithm 4 iteratively swaps S-box elements to improve its LAP while maintaining its NL.

A. CONSTRUCTING A BIJECTIVE S-BOX INCREMENTALLY

A bijective S-box can be constructed incrementally using the constraints in (1). Namely, coordinate Boolean functions, $f_0, f_1, \ldots, f_{n-1}$, are appended one by one to the $n \times n$ S-box. After appending the first j functions, $f_0, f_1, \ldots,$ and f_{j-1}, to the S-box, the resulting partially constructed S-box is denoted $S_{j-1}(x) = \text{decimal}\left(f_0(x), f_1(x), \ldots, f_{j-1}(x)\right)$.

Algorithm 1 constructs a coordinate Boolean function, f_j, by adding one bit at a time while satisfying the bijectivity constraint in (1). To achieve this, the domain of the jth Boolean function, $f_j(x)$, is divided into 2^j segments, denoted $\sigma_{j,y} = \left\{x \in \mathbb{Z}_2^n, S_{j-1}(x) = y\right\}, 0 \le y < 2^j$, where y is a distinct output value of the partially constructed S-box, S_{j-1}. For instance, when generating the first coordinate Boolean function $f_0(x)$, there is only one segment, denoted $\sigma_{0,0} = \{x \in \mathbb{Z}_2^n\}$. For the second coordinate Boolean function $f_1(x)$, there are two segments, denoted $\sigma_{1,0} = \{x \in \mathbb{Z}_2^n, S_0(x) = 0\}$ and $\sigma_{1,1} = \{x \in \mathbb{Z}_2^n, S_0(x) = 1\}$. When generating the third Boolean function, $f_2(x)$, there are four groups, $\sigma_{2,y} = \left\{x \in \mathbb{Z}_2^n, S_1(x) = y\right\}$, where $0 \le y \le 3$. For each segment, we define two indicator variables $\zeta_{j,y}$ and $\omega_{j,y}$, which indicate the number of zeros and the number of ones of $f_j(x)$ in segment $x \in \sigma_{j,y}$.

$$\zeta_{j,y} = \#\{x \in \sigma_{j,y}, f_j(x) = 0\}, \text{ and}$$
$$\omega_{j,y} = \#\{x \in \sigma_{j,y}, f_j(x) = 1\}.$$

The bijectivity constraint, (1), can be rewritten as

$$\zeta_{j,y} = \omega_{j,y} = 2^{n-j-1}, \quad \forall\, 0 \le y < 2^j. \quad (10)$$

Whenever a bit, $f_j(x)$, is being added to a Boolean function f_j, the corresponding value of the partially constructed S-box function, $y = S_{j-1}(x)$, determines the segment, $\sigma_{j,y}$, to which the new bit will be added. We check the number of zeros, $\zeta_{j,y}$, and the number of ones, $\omega_{j,y}$, already in segment $\sigma_{j,y}$. If either $\zeta_{j,y}$ or $\omega_{j,y}$ has reached the maximum allowed within the segment, i.e., 2^{n-j-1}, the new bit is determined to be a one or a zero, respectively. Otherwise, the new bit is chosen at random. The process of constructing a coordinate Boolean function satisfying bijectivity constraints is listed in Algorithm 1. To simplify notation, the given partially constructed S-box with $j-1$ coordinates is denoted S. A simple worked example of Algorithm 1 can be found in the appendix.

Algorithm 1 Bijective Coordinate

Inputs: S-box input size, n, coordinate index, j, partially constructed S-box, S, pseudorandom bit generator, R
Output: n-input Boolean function, f_j

Initialize number of zeros and number of ones in each of the 2^j segments: $\zeta_{j,y} \leftarrow 0$, $\omega_{j,y} \leftarrow 0$, $\forall y \in \{0, 1, \ldots, 2^j - 1\}$
for $x = 0 : 2^n - 1$ **do**
  if $\zeta_{j,S(x)} = 2^{n-j-1}$ **then**
    $f_j(x) \leftarrow 1$
  else if $\omega_{j,S(x)} = 2^{n-j-1}$ then
    $f_j(x) \leftarrow 0$
  else
    $f_j(x) \xleftarrow{\mathcal{R}} 0, 1\}$
    if $f_j(x) = 0$ **then**
      $\zeta_{j,S(x)} \leftarrow \zeta_{j,S(x)} + 1$
    else
      $\omega_{j,S(x)} \leftarrow \omega_{j,S(x)} + 1$
    end
  end if
end for
output f_j■

B. IMPROVING NONLINEARITY

Since Algorithm 1 generates randomized Boolean functions with random nonlinearity, it is necessary to post-process its output to reach a desired high nonlinearity. For this purpose, we proposed Algorithm 2 which is a variation of standard hill-climbing search with nonlinearity as the objective. Given an initial coordinate Boolean function, f, a partially constructed $(j-1)$-coordinate S-box and a target nonlinearity NL_{min}, Algorithm 2 attempts to maximize the nonlinearity of f, defined by (4), by iteratively swapping bits of f, while maintaining bijectivity constraints defined by (10). A stopping condition, $NL(f) \geq NL_{min}$ is introduced to limit the search to a desired nonlinearity.

To speed up the search, our goal is to define the neighborhood operator $\mathcal{N}(f)$, such that all neighbors, $f' \in \mathcal{N}(f)$, have a nonlinearity better than $NL(f)$. To achieve this goal, we use (2) and (3) to identify the linear functions, $L_\alpha = \alpha \odot x$, which are relevant to $NL(f)$. We adapt the method of [29] to keep track of the correlation of f with each linear function, L_α in an array $\hat{f}(\alpha)$, and find the set of nearest linear functions to f, $M_0 = \left\{\alpha \mid \left|\hat{f}(\alpha)\right| = \hat{F}\right\}$. The set of linear functions that may become the nearest to f in case $NL(f)$ is improved, are kept in $M_1 = \left\{\alpha \mid \left|\hat{f}(\alpha)\right| = \hat{F}-2\right\}$. To speed up calculation, the proposed algorithm incrementally updates these two sets after each iteration to reflect the refined f. To improve the nonlinearity, we choose two complementary elements, x_0 and x_1, such that $f(x_0) = 0, f(x_1) = 1$, and the swapping of the two elements brings f farther from the linear functions in M_0, and also keeps f at least as far as it already is from those in M_1. It turns out that the set of pairs (x_0, x_1) that improve nonlinearity can be precisely identified by the constraints given in Theorem 1. We use the following mathematical notation: "$\neg$" represents the logical negation (NOT) operator, "$\vee$" represents the logical disjunction (OR) operator, "$\wedge$" represents the logical conjunction (AND) operator, and "$\vdash$" represents the implication operator, i.e., $A \vdash B \equiv \neg A \vee B$.

Theorem 1: Given a Boolean function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2, n \geq 6$, and a pair of inputs x_0 and $x_1 \in \mathbb{Z}_2^n$ such that

$$f(x_0) = 0, \quad f(x_1) = 1, \tag{11}$$

the modified Boolean function $f' : \mathbb{Z}_2^n \to \mathbb{Z}_2$, defined as

$$f'(x) = \begin{cases} 1, & x = x_0 \\ 0, & x = x_1 \\ f(x), & \text{otherwise.} \end{cases} \tag{12}$$

has an improved nonlinearity, $NL\left(f'\right) = NL(f)+2$, provided that

$$\alpha \odot x_0 = \begin{cases} 0, & \hat{f}(\alpha) > 0 \\ 1, & \hat{f}(\alpha) < 0, \end{cases} \qquad \forall\, \alpha \in M_0 \tag{13}$$

$$\alpha \odot (x_0 \oplus x_1) = 1, \quad \forall\, \alpha \in M_0, \text{ and} \tag{14}$$

$$\begin{aligned} &(\alpha \odot (x_0 \oplus x_1) = 0) \vee \\ &\left(\alpha \odot x_0 = \begin{cases} 0, \hat{f}(\alpha) > 0 \\ 1, \hat{f}(\alpha) < 0 \end{cases}\right), \quad \forall\, \alpha \in M_1. \end{aligned} \tag{15}$$

Proof: Let $\Delta\hat{f}(\alpha) = \hat{f}'(\alpha) - \hat{f}(\alpha)$. From (2), we obtain

$$\Delta\hat{f}(\alpha) = \frac{1}{2}\sum_{x\in\mathbb{Z}_2^n} (-1)^{f'(x)\oplus(\alpha\odot x)} - \frac{1}{2}\sum_{x\in\mathbb{Z}_2^n} (-1)^{f(x)\oplus(\alpha\odot x)}.$$

From (12), it follows that $\forall x \notin \{x_0, x_1\}, f'(x) = f(x)$, and the corresponding terms of the summations are eliminated.

$$\begin{aligned} \therefore \Delta\hat{f}(\alpha) = \frac{1}{2}\Big(&\left((-1)^{f'(x_0)\oplus(\alpha\odot x_0)} + (-1)^{f'(x_1)\oplus(\alpha\odot x_1)}\right) \\ &- \left((-1)^{f(x_0)\oplus(\alpha\odot x_0)} + (-1)^{f(x_1)\oplus(\alpha\odot x_1)}\right)\Big) \end{aligned} \tag{16}$$

Substituting from (11) and (12)

$$\begin{aligned} \Delta\hat{f}(\alpha) &= \frac{1}{2}\left((-1)^{1\oplus(\alpha\odot x_0)} + (-1)^{0\oplus(\alpha\odot x_1)}\right) \\ &\quad -\frac{1}{2}\left((-1)^{0\oplus(\alpha\odot x_0)} + (-1)^{1\oplus(\alpha\odot x_1)}\right) \\ &= \frac{1}{2}\left(-(-1)^{(\alpha\odot x_0)} + (-1)^{(\alpha\odot x_1)}\right) \\ &\quad -\frac{1}{2}\left((-1)^{(\alpha\odot x_0)} - (-1)^{(\alpha\odot x_1)}\right) \\ &= -\left((-1)^{(\alpha\odot x_0)} - (-1)^{(\alpha\odot x_1)}\right) \end{aligned} \tag{17}$$

$$\therefore \Delta\hat{f}(\alpha) = -(-1)^{(\alpha\odot x_0)}\left(1 - (-1)^{\alpha\odot(x_0\oplus x_1)}\right) \tag{18}$$

There are three cases for (18), namely, when $\alpha \in M_0$, when $\alpha \in M_1$, and when $\alpha \notin M_0 \cup M_1$.

Case 1: when $\alpha \in M_0$. Substituting from (13) and (14) into (18)

$$\begin{aligned} \forall\, \alpha \in M_0\colon \Delta\hat{f}(\alpha) &= \begin{cases} -(-1)^0\left(1-(-1)^1\right), & \hat{f}(\alpha) > 0 \\ -(-1)^1\left(1-(-1)^1\right), & \hat{f}(\alpha) < 0 \end{cases} \\ &= \begin{cases} -2, & \hat{f}(\alpha) > 0 \\ 2, & \hat{f}(\alpha) < 0 \end{cases} \\ \therefore \forall\, \alpha \in M_0\colon \left|\hat{f}'(\alpha)\right| &= \begin{cases} \left|\hat{f}(\alpha) - 2\right|, & \hat{f}(\alpha) > 0 \\ \left|\hat{f}(\alpha) + 2\right|, & \hat{f}(\alpha) < 0 \end{cases} \end{aligned}$$

By definition $\left|\hat{f}(\alpha)\right| = \hat{F}, \forall\alpha \in M_0$, and according to [46], $\hat{F} \geq 2^{(n/2)-1}$, which yields $\hat{F} \geq 4$, for $n \geq 6$. Therefore, $\left|\hat{f}(\alpha)\right| \geq 4, \forall\alpha \in M_0$.

$$\therefore \forall\, \alpha \in M_0 : \left|\hat{f}'(\alpha)\right| = \begin{cases} \hat{f}(\alpha) - 2, & \hat{f}(\alpha) > 0 \\ \left|\hat{f}(\alpha)\right| - 2, & \hat{f}(\alpha) < 0. \end{cases}$$

$$\therefore \left|\hat{f}'(\alpha)\right| = \hat{F} - 2, \quad \forall\alpha \in M_0 \tag{19}$$

Case 2: when $\alpha \in M_1$. From (15) it follows that

$$\forall \alpha \in M_1 : (\alpha \odot (x_0 \oplus x_1) = 1) \vdash \left(\alpha \odot x_0 = \begin{cases} 0, & \hat{f}(\alpha) > 0 \\ 1, & \hat{f}(\alpha) < 0 \end{cases} \right)$$

Substituting in (18), we obtain

$$\forall \alpha \in M_1 \colon \Delta\hat{f}(\alpha) = \begin{cases} -(-1)^{(\alpha\odot x_0)}\left(1-(-1)^0\right), \alpha \odot (x_0 \oplus x_1) = 0 \\ -(-1)^0\left(1-(-1)^1\right), \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) > 0 \\ -(-1)^1\left(1-(-1)^1\right), \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) < 0. \end{cases}$$

$$\therefore \forall \alpha \in M_1 :$$

$$\Delta\hat{f}(\alpha) = \begin{cases} 0, & \alpha \odot (x_0 \oplus x_1) = 0 \\ -2, & \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) > 0 \\ 2, & \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) < 0. \end{cases}$$

$$\therefore \forall \alpha \in M_1 :$$

$$\left|\hat{f}'(\alpha)\right| = \begin{cases} \left|\hat{f}(\alpha)\right|, & \alpha \odot (x_0 \oplus x_1) = 0 \\ \left|\hat{f}(\alpha) - 2\right|, & \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) > 0 \\ \left|\hat{f}(\alpha) + 2\right|, & \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) < 0. \end{cases}$$

Since $\hat{F} \geq 4$ when $n \geq 6$, it follows from the definition of M_1 that $\left|\hat{f}(\alpha)\right| = \hat{F} - 2 \geq 2, \forall \alpha \in M_1$

$$\therefore \forall \alpha \in M_1\colon$$

$$\left|\hat{f}'(\alpha)\right| = \begin{cases} \left|\hat{f}(\alpha)\right|, \alpha \odot (x_0 \oplus x_1) = 0 \\ \hat{f}(\alpha) - 2, \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) > 0 \\ \left|\hat{f}(\alpha)\right| - 2, \alpha \odot (x_0 \oplus x_1) = 1, \hat{f}(\alpha) < 0. \end{cases}$$

$$\left|\hat{f}'(\alpha)\right| \leq \hat{F} - 2, \forall \alpha \in M_1 \tag{20}$$

Case 3: when $\alpha \notin M_0 \cup M_1$

$$\forall \alpha \notin M_0 \cup M_1 :$$

$$\Delta\hat{f}(\alpha) = -(-1)^{(\alpha\odot x_0)}\left(1-(-1)^{\alpha\odot(x_0\oplus x_1)}\right)$$

$$\therefore \forall \alpha \notin M_0 \cup M_1 :$$

$$\Delta\hat{f}(\alpha) = \begin{cases} 0, & \alpha \odot (x_0 \oplus x_1) = 0 \\ -2, & \alpha \odot (x_0 \oplus x_1) = 1, (\alpha \odot x_0) = 0 \\ 2, & \alpha \odot (x_0 \oplus x_1) = 1, (\alpha \odot x_0) = 0. \end{cases}$$

Since $\hat{f}'(\alpha) = \hat{f}(\alpha) + \Delta\hat{f}(\alpha)$,

$$\therefore \left|\hat{f}(\alpha)\right| - 2 \leq \left|\hat{f}'(\alpha)\right| \leq \left|\hat{f}(\alpha)\right| + 2, \forall \alpha \notin M_0 \cup M_1$$

From the definition of M_0 and M_1, it follows that

$$\left|\hat{f}(\alpha)\right| \leq \hat{F} - 4, \forall \alpha \notin M_0 \cup M_1$$

$$\therefore \left|\hat{f}'(\alpha)\right| \leq \hat{F} - 2, \forall \alpha \notin M_0 \cup M_1 \tag{21}$$

Combining the three cases, i.e., (19), (20), and (21), we obtain

$$\left|\hat{f}'(\alpha)\right| \leq \hat{F} - 2, \forall \alpha \in \mathbb{Z}_2^n$$

$$\therefore \hat{F}' = \hat{F} - 2 \tag{22}$$

$$\because NL\left(f'\right) = 2^{n-1} - \hat{F}' = 2^{n-1} - \left(\hat{F} - 2\right)$$

$$\therefore NL(f') = NL(f) + 2 \blacksquare$$

To preserve the bijectivity of the S-box when constructing the j th coordinate Boolean function f_j, we ensure that both x_0 and x_1 belong to the same segment $\sigma_{j,y}$ by asserting the constraint

$$y = S_{j-1}(x_0) = S_{j-1}(x_1). \tag{23}$$

Using the constraints defined in Equations (11), (13), (14), (15) and (23), we can identify pairs (x_0, x_1) that improve $NL(f)$ while preserving S-box bijectivity. As shown in Algorithm 2, nonlinearity improvement is iterated until the desired nonlinearity is achieved or a local maximum is encountered. Note that the WHT of the Boolean function, $\hat{f}$, is calculated only once at the beginning of the algorithm and subsequently updated using (17) after each iteration.

Algorithm 2 Refine Nonlinearity

Inputs: S-box input size, n, coordinate index, j, coordinate Boolean function, f, partially constructed S-box, S, desired nonlinearity, NL_{min}.
Output: Updated Boolean function, f.

Calculate $\hat{f}(\alpha) \leftarrow 1/2 \sum_{x\in\mathbb{Z}_2^n} (-1)^{f(x)\oplus(\alpha\odot x)}, \forall \alpha \in \mathbb{Z}_2^n$.
Find $\hat{F} \leftarrow \max_{\alpha\varepsilon\mathbb{Z}_2^n} \left|\hat{f}(\alpha)\right|$.
while $2^{n-1} - \hat{F} < NL_{min}$ **do**
  $M_0 \leftarrow \left\{\alpha \in Z_2^n, \text{suchthat} \left|\hat{f}(\alpha)\right| = \hat{F}\right\}$,
  $M_1 \leftarrow \left\{\alpha \in Z_2^n, \text{suchthat} \left|\hat{f}(\alpha)\right| = \hat{F} - 2\right\}$
  Choose $x_0, x_1 \in \mathbb{Z}_2^n$ such that constraints (11), (23), (13), (14), and (15) are satisfied.
  if no such choice exists **then**
    break
  else ▷ *update* $S, \hat{f}$ *and* $\hat{F}$ using (12), (17) and (22)
    $f(x_0) \leftarrow 1, f(x_1) \leftarrow 0, \hat{F} \leftarrow \left(\hat{F} - 2\right)$.
    $\hat{f}(\alpha) \leftarrow \hat{f}(\alpha) + (-1)^{\alpha\odot x_1} - (-1)^{\alpha\odot x_0}, \forall \alpha \in Z_2^n$.
  end if
end while
output $f \blacksquare$

C. CONSTRUCTING HIGHLY NONLINEAR BIJECTIVE S-BOXES

The process of the constructing a bijective $n \times n$ S-box with a given nonlinearity, NL_{min}, is listed in Algorithm 3. The algorithm starts with an empty S-box and incrementally appends coordinate Boolean functions produced by Algorithm 1 (Bijective Coordinate) and Algorithm 2 (Refine Nonlinearity), satisfying bijectivity and nonlinearity constraints. Since Algorithm 2 may get stuck at a local maximum, failing to achieve the required nonlinearity, Algorithm 3 uses a random-restart strategy to repeatedly invoke Algorithm 1 and Algorithm 2 until success. Once Algorithm 2 successfully produces a coordinate Boolean function with the required

NL, the resulting Boolean function is placed into the S-box and Algorithm 3 moves to the next coordinate, until all n coordinates are constructed.

Algorithm 3 Construct S-Box

Inputs: S-box input size, n, required nonlinearity, NL_{min}, pseudorandom bit generator $\mathcal{R}$
Output: $n \times n$ bijective S-box, $S(x)$

```
Initialize S(x) ← 0, ∀ x ∈ Z_{2^n}
for j = 0 : n − 1
    Initialize number of zeros and ones in each segment:
        ζ_{j,y} ← 0, ω_{j,y} ← 0, ∀y ∈ {0, 1, ..., 2^j − 1}
    found ← false
    for iter = 1 : iter_max
        f_j ← Bijective Coordinate (n, j, S, R)
        f_j ← Refine Nonlinearity (n, j, f_j, S, NL_min)
        if NL(f_j) ≥ NL_min then
                found ← true
                break
        end if
    end for
    if found then
        S(x) ← S(x) + 2^j f_j(x), ∀x ∈ Z_{2^n}
    else
        terminate unsuccessfully
    end if
end for
output S ■
```

To ensure that the proposed method is key-dependent, the arbitrary choices made during the S-box construction process are fully determined by a Mersenne twister (MT19937) pseudorandom bit generator (PRBG), $\mathcal{R}$. Thus, the seed of the PRBG can be considered the S-box construction key. If a need for the inverse S-box arises, the same key can be used to construct the S-box first, then its inverse can be calculated at a negligible cost.

D. IMPROVING LINEAR APPROXIMATION PROBABILITY

To further improve the properties of the resulting S-boxes we propose a LAP refinement algorithm by modifying the heuristic search method given in [45] to guarantee nonlinearity preservation. The Linear Approximation Table (LAT) is a matrix, in which the element in row a and column b represents the correlation between the Boolean function $L_b(S(x))$ and the linear Boolean function $L_a(x)$,

$$\ell l_S(a, b) = \#\{x \in \mathbb{Z}_2^n | a \odot x = b \odot S(x)\} - 2^{n-1}, \quad \forall\, a, b \in \mathbb{Z}_2^n$$

The maximum absolute LAT value is denoted Λ,

$$\Lambda = \max_{a \in \mathbb{Z}_2^n, b \in \mathbb{Z}_2^{n*}} |\ell l_S(a, b)|$$

We use the objective function proposed by [45]

$$R(S) = \sum_{l \geq 0} N_l \cdot 2^l$$

where N_l is the number of LAT elements with absolute value l, i.e., $N_l = \#\left\{a \in \mathbb{Z}_2^n, b \in \mathbb{Z}_2^{n*}, |\ell l_S(a, b)| = l\right\}$

To improve LAP, we select a pair of S-box elements to swap, $S(x_0)$ and $S(x_1)$. We choose a LAT element $\ell_S(a, b)$ that has an absolute maximum value, Λ. Then, we find the set of candidate elements of S that contribute to $\ell_S(a, b)$. After that, we pick a pair of different elements from L to swap such that $R(S)$ is minimized as proposed in [45]. However, the method in [45] disregards S-box nonlinearity, which usually decreases consequently. The proposed algorithm introduces a new constraint to guarantee nonlinearity preservation during element swapping. The constraint is defined in the following theorem.

Theorem 2: Given a bijective S-box $S : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$, with coordinate functions $f_0, f_1, \ldots, f_{n-1}$, and two inputs x_0, $x_1 \in \mathbb{Z}_2^n$, the modified S-box $S' : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ defined by

$$S'(x) = \left(f_0'(x), \ldots, f_{n-1}'(x)\right) = \begin{cases} S(x_1), & x = x_0 \\ S(x_0), & x = x_1 \\ S(x), & \text{otherwise} \end{cases} \tag{24}$$

has nonlinearity $NL\left(S'\right) \geq NL(S)$, provided that $\forall j, \alpha$, $0 \leq j < n, \alpha \in \mathbb{Z}_2^n$, at least one of following constraints is satisfied:

$$f_j(x_0) = f_j(x_1) \tag{25}$$

$$|\hat{f}_j(\alpha)| \leq \max_{0 \leq i < n} \hat{F}_i, \tag{26}$$

$$\left(f_j(x_0) \neq f_j(x_1)\right) \wedge \left(\hat{f}_j(\alpha) = \max_{0 \leq i < n} \hat{F}_i\right) \wedge \left(f_j(x_1) \oplus (\alpha \odot x_0) = 1\right) \tag{27}$$

$$\left(f_j(x_0) \neq f_j(x_1)\right) \wedge \left(\hat{f}_j(\alpha) = -\max_{0 \leq i < n} \hat{F}_i\right) \wedge \left(f_j(x_1) \oplus (\alpha \odot x_0) = 0\right) \tag{28}$$

$$\left(f_j(x_0) \neq f_j(x_0)\right) \wedge \left(\hat{f}_j(\alpha) = \max_{0 \leq i < n} \hat{F}_i\right) \wedge \left(f_j(x_0) \oplus (\alpha \odot x_0) = 1\right) \tag{29}$$

$$\left(f_j(x_0) \neq f_j(x_0)\right) \wedge \left(\hat{f}_j(\alpha) = -\max_{0 \leq i < n} \hat{F}_i\right) \wedge \left(f_j(x_0) \oplus (\alpha \odot x_0) = 0\right) \tag{30}$$

Proof: First, we show that for any $0 \leq j < n$, each of the conditions (25), through (30) is sufficient to preserve the nonlinearity of the Boolean function, f_j.

1) From (25), if $f_j(x_0) = f_j(x_1)$, then $f_j'(x_0) = f_j'(x_1)$. Consequently, $f_j' \equiv f_j$ and $NL\left(f_j'\right) = NL\left(f_j\right)$. Since $NL(S) = \min_{0 \leq j < n} NL\left(f_j\right)$, we conclude that $NL\left(f_j'\right) \geq NL(S)$.

2) From (26), if $\forall\alpha : \left|\hat{f}_j(\alpha)\right| < \max_{0\le i<n} \hat{F}_i$, then $\forall\alpha, \left|\hat{f}_j(\alpha)\right| + 2 \le \max_{0\le i<n} \hat{F}_i$. Since, $\left|\hat{f}_j'(\alpha)\right| \le \left|\hat{f}_j(\alpha)\right| + \left|\Delta\hat{f}_j(\alpha)\right|$ and from (18) $\left|\Delta\hat{f}_j(\alpha)\right| \le 2$, then $\left|\hat{f}_j'(\alpha)\right| \le \max_{0\le i<n} \hat{F}_i, \forall\alpha$. Therefore, $\hat{F}_j' \le \max_{0\le i<n} \hat{F}_i$. From the basic definition of NL, (3) and (4), we conclude that $NL\left(f_j'\right) \ge NL(S)$.
3) From (27), $f_j(x_1) \oplus (\alpha \odot x_0) = 1, \forall\alpha \in \mathbb{Z}_2^n$, and since $f_j(x_0) \ne f_j(x_1)$, then $f_j(x_0) \oplus (\alpha \odot x_0) = 0, \forall\alpha \in \mathbb{Z}_2^n$. From (24), $f_j'(x_0) = f_j(x_1)$ and $f_j'(x_1) = f_j(x_0)$. Therefore, $f_j'(x_0) \oplus (\alpha \odot x_0) = 1, \forall\alpha \in \mathbb{Z}_2^n$ and $f_j'(x_1) \oplus (\alpha \odot x_0) = 0, \forall\alpha \in \mathbb{Z}_2^n$. Substituting in (16), we obtain

$$\begin{aligned}\Delta\hat{f}_j(\alpha) = &\frac{1}{2}\left(-1 + (-1)^{f_j'(x_1)\oplus(\alpha\odot x_1)}\right)\\ &-\frac{1}{2}\left(1 + (-1)^{f_j(x_1)\oplus(\alpha\odot x_1)}\right) \le 0\\ \therefore\ &\hat{f}_j'(\alpha) \le \hat{f}_j(\alpha)\end{aligned}$$

Since $\hat{f}_j(\alpha) = \max_{0\le i<n} \hat{F}_i$, from (27),

$$\therefore \hat{f}_j'(\alpha) \le \max_{0\le i<n} \hat{F}_i$$

Therefore, $NL\left(\hat{f}\right) \ge NL(S)$.

4) From (28), $f_j(x_1) \oplus (\alpha \odot x_0) = 0, \forall\alpha \in \mathbb{Z}_2^n$, and since $f_j(x_0) \ne f_j(x_1)$, then $f_j(x_0) \oplus (\alpha \odot x_0) = 1, \forall\alpha \in \mathbb{Z}_2^n$. From (24), $f_j'(x_0) = f_j(x_1)$ and $f_j(x_1) = f_j(x_0)$. Therefore, $f_j'(x_0) \oplus (\alpha \odot x_0) = 0, \forall\alpha \in Z_2^n$ and $f_j'(x_1) \oplus (\alpha \odot x_0) = 1, \forall\alpha \in \mathbb{Z}_2^n$. Substituting in (16), we obtain

$$\begin{aligned}\Delta\hat{f}_j(\alpha) = &\frac{1}{2}\left(1 + (-1)^{f_j'(x_1)\oplus(\alpha\odot x_1)}\right)\\ &-\frac{1}{2}\left(-1 + (-1)^{f_j(x_1)\oplus(\alpha\odot x_1)}\right) \ge 0\end{aligned}$$

Using (18) $\left|\Delta\hat{f}_j(\alpha)\right| \le 2$, we obtain

$$\therefore \hat{f}_j(\alpha) \le \hat{f}_j'(\alpha) \le \hat{f}_j(\alpha) + 2$$

Since $\hat{f}_j(\alpha) = -\max_{0\le i<n} \hat{F}_i$, from (28),

$$\begin{aligned}&\therefore -\max_{0\le i<n} \hat{F}_i \le \hat{f}_j'(\alpha) \le \max_{0\le i<n} -\hat{F}_i + 2\\ &\therefore \max_{0\le i<n} \hat{F}_i \ge -\hat{f}_j'(\alpha) \ge \max_{0\le i<n} \hat{F}_i - 2\\ &\therefore \max_{0\le i<n} \hat{F}_i \ge \left|\hat{f}_j'(\alpha)\right| \ge \max_{0\le i<n} \hat{F}_i - 2\end{aligned}$$

Therefore, $NL\left(\hat{f}_j\right) \ge NL(S)$

5. Following similar steps used to prove (27) and (28), it can be shown that each of (29) and (30) guarantees that $NL\left(\hat{f}_j\right) \ge NL(S)$.

Since satisfying any of the constraints defined by (25)-(30) guarantees that $NL\left(\hat{f}_j'\right) \ge NL(S), \forall 0 \le j < n \Rightarrow NL\left(S'\right) \ge NL(S)$, i.e., the NL of the S-Box is preserved ■

Using Theorem 2, the LAP optimization procedure is written in Algorithm 4. The optimization process is repeated recursively until no further improvement in the objective function can be found.

Algorithm 4 Refine LAP

Input: S-box, S
Output: Updated S-box, S

```
Calculate the LAT of S, ℓl_S(a, b), ∀a ∈ Z_2^n, b ∈ Z_2^{n*}
Find Λ = max_{a∈Z_2^n, b∈Z_2^{n*}} |ℓl_S(a, b)|
for each a, b such that |ℓl_S(a, b)| = Λ do
   Construct L ← {x ∈ Z_2^n | a ⊙ x = b ⊙ S(x)}
      for (x_0, x_1) ∈ L^2, x_0 < x_1 do
         if ∀ 0 ≤ j < n, α ∈ Z_2^n, any of constraints (25)
                     through (30) is satisfied, then
            S' ← S,
            Swap S'(x_0) ↔ S'(x_1)
            if R(S') < R(S) then
                S ← Refine LAP (S') ▷ recursion
            end if
      end if
   end for
end for
output S■
```

IV. PERFORMANCE ANALYSIS

In this section, we analyze the security properties of the S-boxes generated by the proposed method and compare them with recent S-boxes. The computational efficiency of the proposed method will also be examined.

A. S-BOX SECURITY ANALYSIS

The minimum nonlinearity required, NL_{min}, is an input to Algorithm 2 and Algorithm 3. Throughout our experiments, with $iter_{max} = 1000$, Algorithm 3 always constructed an 8×8 S-box with nonlinearity equal to NL_{min}, for all $NL_{min} \le 112$. Higher nonlinearity, $NL_{min} = 114$ requires a much larger number of iterations.

Figure 2 shows a sample dynamic 8×8 S-box, S_1, constructed using Algorithm 3 with NL_{min} set to 112, whereas Figure 3 shows the corresponding improved-LAP S-box, S_2. Similarly, Figure 4 presents a sample 8×8 S-box, S_3, constructed with NL_{min} set to 114, whereas Figure 5 shows the corresponding S-box, S_4, after LAP refinement with Algorithm 4. The highlighted elements in Figure 3 and Figure 5 indicate the elements changed from S_1 to S_2 and from S_3 to S_4, respectively.

Results of S-box security tests are listed in Table 1. Results show that Algorithm 3 successfully generated an S-box with $NL = 112$ and that Algorithm 4 successfully decreased its LAP from 0.1406 to 0.1094 while preserving nonlinearity at $NL = 112$. Similarly, Algorithm 3 successfully generated an S-box with $NL = 114$ and Algorithm 4 successfully

TABLE 1. S-box test results of sample S-boxes showing the LAP improvement due to Algorithm 4.

S-box	NL	LAP	DU	SAC			BIC-NL	BIC-SAC		
				min	max	Avg.		min	max	avg
S_1	112	0.1406	10	0.4219	0.5625	0.4963	102.43	0.4727	0.5273	0.4995
S_2	112	**0.1094**	10	0.4375	0.5781	0.5002	106.14	0.4668	0.5273	0.5010
S_3	114	0.1484	12	0.4219	0.5625	0.4995	104.14	0.4805	0.5371	0.5033
S_4	114	**0.1328**	12	0.4219	0.5625	0.4993	104.21	0.4766	0.5332	0.5040

157	253	199	207	148	111	28	140	84	32	12	151	220	112	89	35
114	212	242	96	185	224	18	44	0	161	246	222	16	195	146	122
1	11	126	3	68	153	216	141	67	210	206	152	184	13	248	103
70	240	42	48	47	200	106	138	61	20	232	49	85	251	43	204
82	10	218	197	50	160	33	237	72	198	109	179	30	71	192	23
120	136	134	255	76	215	45	87	81	245	56	156	219	91	104	64
5	8	254	202	125	119	170	6	115	24	88	244	174	213	249	171
165	217	194	14	59	40	52	155	128	51	176	168	105	149	69	127
90	57	175	144	121	252	123	173	133	38	150	158	53	25	80	189
238	7	187	234	107	21	29	205	208	180	154	181	22	116	172	178
78	221	37	182	101	77	92	74	169	236	60	36	46	108	73	34
188	94	39	62	177	231	159	124	196	93	55	139	235	228	203	166
132	117	4	100	164	26	241	247	142	250	229	79	41	190	226	27
102	186	15	75	110	183	58	17	113	86	214	145	135	201	162	230
239	54	2	9	118	130	66	167	209	191	137	163	98	97	223	233
99	243	143	63	227	31	19	131	129	95	147	211	193	83	65	225

FIGURE 2. S-box, S_1, constructed using Algorithm 3 with $NL_{min} = 112$.

191	253	71	207	212	107	25	192	4	128	12	23	216	112	73	47
123	220	240	164	176	224	18	109	56	161	246	222	16	195	146	114
64	11	126	7	68	153	252	141	67	210	206	152	184	5	120	103
70	242	42	48	45	200	106	138	61	158	232	49	85	251	43	76
122	10	218	197	50	96	33	237	72	198	44	163	30	199	132	151
248	136	134	255	204	215	37	87	81	245	0	28	219	91	104	1
13	8	244	202	125	119	170	6	115	24	88	180	174	213	249	171
165	217	194	14	59	40	52	155	185	51	160	168	105	149	69	127
78	57	175	144	121	254	82	173	133	38	150	156	53	21	80	189
238	3	187	234	239	17	29	205	208	148	154	181	22	116	32	178
90	221	99	182	117	77	92	74	169	236	60	36	46	108	89	34
188	94	39	62	177	231	157	124	196	93	55	139	235	228	203	166
140	101	84	100	172	26	209	247	142	250	229	79	41	190	226	27
102	186	15	75	110	183	58	20	113	86	214	145	135	201	162	230
111	54	2	9	118	130	66	167	241	159	137	179	98	97	223	233
35	243	143	63	227	31	19	131	129	95	147	211	193	83	65	225

FIGURE 3. S-box, S_2, after applying Algorithm 4 to S_1. Swapped elements are highlighted.

121	186	240	226	250	239	126	35	86	222	125	230	247	40	236	88
233	194	106	146	134	219	248	124	27	229	155	62	30	100	54	234
128	44	172	246	190	10	56	218	249	139	188	21	156	178	108	34
187	42	17	57	196	7	200	220	211	104	141	237	25	115	113	191
143	214	2	152	97	207	145	14	49	68	41	224	107	209	69	15
83	70	120	253	162	65	78	45	221	232	117	80	173	4	137	169
81	33	114	144	101	185	133	60	105	90	98	76	84	72	175	163
119	147	216	189	158	201	48	66	32	89	55	157	242	197	241	6
73	111	208	58	36	13	24	16	28	131	9	95	170	50	138	19
22	74	132	99	92	112	63	123	3	176	252	205	118	18	43	46
193	5	255	94	151	180	160	217	181	140	135	82	227	0	153	179
195	109	53	235	210	67	243	130	37	177	254	182	23	204	202	71
91	87	228	96	77	154	38	164	129	251	192	11	79	64	168	1
26	59	150	198	136	122	61	206	225	142	199	245	166	93	174	29
161	238	102	75	148	184	8	213	215	165	110	85	51	149	127	203
244	171	39	167	183	103	47	231	223	31	212	159	20	12	52	116

FIGURE 4. S-box, S_3, constructed using Algorithm 3 with $NL_{min} = 114$.

decreased its LAP from 0.1484 to 0.1328 while preserving nonlinearity at $NL = 114$.

121	186	240	226	250	239	126	35	86	222	125	230	247	40	236	88
169	194	106	146	134	219	248	124	27	229	155	62	30	100	54	234
128	44	172	246	190	10	56	218	249	139	188	21	156	178	108	34
187	42	17	57	196	7	200	220	211	104	141	237	25	115	113	191
143	214	2	152	97	207	145	14	49	68	41	224	107	209	69	15
83	70	120	253	162	65	76	45	221	232	117	80	173	4	137	185
81	33	114	144	101	233	133	60	105	90	98	78	84	72	175	163
119	147	216	189	159	201	48	66	32	89	55	157	242	197	241	6
73	111	208	58	36	13	24	16	28	131	9	95	170	50	138	19
22	74	132	99	92	112	63	123	3	176	252	205	118	18	43	46
193	5	255	94	151	180	160	217	181	140	135	82	227	0	153	179
195	109	53	235	210	67	243	130	37	177	254	182	23	204	202	71
91	87	228	96	77	154	38	164	129	251	192	11	79	64	168	1
26	59	150	198	136	122	61	206	225	142	199	245	166	93	174	29
161	238	102	75	148	184	8	213	215	165	110	85	51	149	127	203
244	171	39	167	183	103	47	231	223	31	212	158	20	12	52	116

FIGURE 5. S-box, S_4, after applying Algorithm 4 to S_3. Swapped elements are highlighted.

For more detailed cryptographic analysis of S_2, Table 2 shows the input-output dependence matrix used for the SAC test and Table 3 shows the BIC-SAC matrix.

TABLE 2. Input-output dependence matrix for S_2.

0.5781	0.4688	0.5313	0.4688	0.5000	0.5313	0.5469	0.4531
0.5469	0.4688	0.4688	0.4688	0.5000	0.5156	0.5156	0.4688
0.4688	0.5313	0.4844	0.5469	0.5313	0.5625	0.4688	0.4844
0.5313	0.4688	0.5000	0.4531	0.5156	0.5313	0.5000	0.4688
0.4531	0.4844	0.5156	0.5156	0.5313	0.4688	0.5313	0.4688
0.5000	0.4688	0.5313	0.5000	0.4844	0.5625	0.5000	0.4375
0.5156	0.5000	0.5469	0.4531	0.5156	0.4688	0.4844	0.5469
0.4688	0.5156	0.5000	0.5313	0.5156	0.4844	0.4844	0.4531

TABLE 3. BIC-SAC matrix for S_2.

-	0.4805	0.4980	0.4844	0.5020	0.4883	0.5117	0.5176
0.4805	-	0.5059	0.5273	0.4668	0.5254	0.5117	0.5156
0.4980	0.5059	-	0.4766	0.4863	0.5137	0.4922	0.5156
0.4844	0.5273	0.4766	-	0.5000	0.4805	0.4980	0.5039
0.5020	0.4668	0.4863	0.5000	-	0.4902	0.5039	0.5176
0.4883	0.5254	0.5137	0.4805	0.4902	-	0.5098	0.4941
0.5117	0.5117	0.4922	0.4980	0.5039	0.5098	-	0.5098
0.5176	0.5156	0.5156	0.5039	0.5176	0.4941	0.5098	-

Table 4 present a comprehensive comparison of S-box security analysis. The sample S-boxes, S_2 and S_4, generated by the proposed method are compared with relevant dynamic S-boxes published in the last few years.

By comparing the nonlinearity and the linear approximation probability (LAP), which are the two objectives

TABLE 4. Comparison of sample S-box generated by the proposed method with recent S-boxes.

Category	S-box	NL			LAP	BIC		SAC				DU
		Min	Max	Avg		NL	SAC	Avg.	Max	Min	Offset	
Chaotic	Ref. [5], 2020	106	110	108	0.1250	104.29	0.4961	0.4990	0.5781	0.4063	0.0938	10
	Ref. [6], 2020	106	108	106.5	0.1328	104.07	0.5005	0.5010	0.5781	0.4219	0.0781	10
	Ref. [7], 2020	104	110	106.25	0.1328	103.93	0.5070	0.5029	0.5938	0.4219	0.0938	10
	Ref. [8], 2020	102	108	105.25	0.1328	102.6	0.4994	0.5037	0.6094	0.4062	0.1094	10
	Ref. [9], 2019	96	106	102.5	0.125	103.93	0.5057	0.5037	0.6094	0.3594	0.1406	10
	Ref. [10], 2019	104	108	106.75	0.1406	103.9	0.4997	0.5071	0.5938	0.4062	0.0938	14
	Ref. [11], 2019	106	110	107.75	0.125	105.07	0.5023	0.4976	0.5781	0.3906	0.1094	10
	Ref. [12], 2018	106	108	106.5	0.1328	104.2	0.5003	0.4978	0.5938	0.4375	0.0938	10
	Ref. [13], 2018	104	110	106	0.1328	104.21	0.5014	0.5197	0.625	0.4375	0.1250	10
	Ref. [14], 2018	106	110	108.5	0.1328	104.00	0.4971	0.5017	0.5938	0.4062	0.0938	10
	Ref. [15], 2018	102	108	104.5	0.1250	104.64	0.5013	0.4980	0.6406	0.4219	0.1406	12
	Ref. [16], 2017	106	108	106.75	0.1328	103.79	0.4951	0.5034	0.6250	0.4219	0.1250	10
	Ref. [17], 2017	102	108	105.25	0.1563	103.8	0.4971	0.5056	0.5781	0.4375	0.0781	10
	Ref. [18], 2017	102	110	105.5	0.1250	104.3	0.4988	0.5010	0.6094	0.4063	0.1094	12
	Ref. [19], 2016	96	106	102.5	0.1641	102.64	0.4026	0.5178	0.6719	0.3906	0.1719	54
	Ref. [20], 2014	108	112	109.25	0.0938	108.21	0.5056	0.5012	0.5938	0.4219	0.0938	8
Algebraic	Ref. [1], 2020	112	116	114	0.125	103.86	0.4978	0.4978	0.5625	0.4375	0.0625	12
	Ref. [2], 2020	112	112	112	0.0625	112	0.5010	0.4956	0.5625	0.4531	0.0625	4
	Ref. [4], 2020	112	112	112	0.0625	112	0.5030	0.5017	0.5625	0.4375	0.0625	4
	Ref. [21], 2019	112	112	112	0.0625	112	0.5046	0.5049	0.5625	0.4531	0.0625	4
	Ref. [22], 2019	106	108	107	0.1563	103.5	0.5040	0.497	0.578	0.4219	0.0781	10
	Ref. [24], 2017	106	108	107.25	0.1328	105.29	0.4980	0.5034	0.6094	0.4219	0.1094	12
	Ref. [25], 2017	112	112	112	0.1094	108	0.5027	0.5115	0.5469	0.4219	0.0781	8
Optimization	Ref. [28], 2020	114	116	114.5	0.1406	104.21	0.5047	0.5012	0.5938	0.4531	0.0938	10
	Ref. [29], 2020	110	112	110.25	0.125	106.79	0.5004	0.4953	0.5781	0.4219	0.0781	10
	Ref. [30], 2018	108	110	108.75	0.1328	94	0.5054	0.4946	0.5629	0.3906	0.1094	10
	Ref. [31], 2018	110	112	110.25	0.125	105.21	0.5052	0.5000	0.6094	0.4219	0.1094	10
	Ref. [32], 2018	108	112	109.25	0.125	104.29	0.4992	0.4985	0.6094	0.3594	0.1406	8
	Ref. [33], 2018	108	110	109.5	0.1328	104.07	0.5020	0.4985	0.5938	0.4063	0.0938	10
	Ref. [34], 2017	106	110	108	0.094	103.93	0.5057	0.5046	0.5938	0.4375	0.0938	8
	Ref. [35], 2017	104	110	106.5	0.1172	105.2	0.4984	0.5120	0.6406	0.4375	0.1406	10
	Ref. [36], 2015	106	110	107	0.125	105.5	0.5010	0.5015	0.5625	0.4063	0.0937	10
	Proposed (S_2)	**112**	**112**	**112**	**0.1094**	**106.14**	**0.5010**	**0.5002**	**0.5781**	**0.4375**	**0.0781**	**10**
	Proposed (S_4)	**114**	**114**	**114**	**0.1328**	**104.21**	**0.5040**	**0.4993**	**0.5625**	**0.4219**	**0.0781**	**12**

TABLE 5. S-box LAP refinement results for a sample of S-boxes generated by the proposed method.

Starting S-Box		Refined using Proposed		Refined using [45]	
LAP	NL	LAP	NL	LAP	NL
0.1250	106	0.0938	106	0.1016	104
0.1406	108	0.0938	108	0.0938	104
0.1328	110	0.1016	110	0.0938	104
0.1406	112	0.1094	112	0.0938	104
0.1484	114	0.1328	114	0.1016	104

optimized by the proposed method, the results of S_2 and S_4 surpass those of all chaotic S-boxes and optimization-based S-boxes. The results of S_2 and S_4 are close to the best algebraic S-boxes.

Table 4 also presents the remaining security analysis including BIC, SAC and DU tests.

To demonstrate the effectiveness of the proposed LAP refinement algorithm, Table 5 lists the LAP for a sample of initial S-boxes generated by the proposed Algorithm 3 with different NL_{min}. The initial S-boxes are then refined with Algorithm 4 and the new LAP and NL are reported in the table. The proposed algorithm successfully decreases LAP while retaining initial S-box nonlinearity. Biryukov's algorithm [45] was applied to the same initial S-boxes and the results are shown for comparison indicating an accidental loss in nonlinearity.

B. COMPUTATIONAL EFFICIENCY

As mentioned in the introduction there are several existing methods to construct 8×8 S-boxes with $NL_{min} = 112$. The main advantage of our proposed method is the capacity to efficiently generate an unlimited number of such S-boxes.

To demonstrate the computational efficiency of the proposed method, we implemented the proposed algorithms in Java and executed them on a PC with Intel Core i7-4790 @ 3.6GHz with 32GB of RAM.

The main algorithms to be evaluated are Algorithm 3, which generates an S-Box with a specific NL, and Algorithm 4, which improves the LAP.

The efficiency of Algorithm 3 can be demonstrated by observing the number of iterations required to find an S-box satisfying the given nonlinearity criterion and the corresponding construction time, as shown in Table 6. For each NL_{min}, $100 \leq NL_{min} \leq 112$, we constructed 1000 S-boxes and observed the average number of iterations and the average construction time.

TABLE 6. Average number of iterations and construction time of Algorithm 3 for each NL criterion.

NL_{min}	Iterations	Time (ms)
100	8.001	2.71
102	8.004	2.69
104	8.005	2.70
106	8.055	2.76
108	8.288	2.85
110	15.191	5.34
112	328.671	118.33

Results indicate that constructing a dynamic S-box with $NL \leq 108$ requires only one iteration per Boolean function, i.e., eight iterations in total. The construction time in this case is merely a few milliseconds. For $NL_{min} = 110$, the number of calls to Algorithm 2 increases slightly and the average construction time is 5.34 milliseconds. The number of iterations significantly increases for $NL_{min} = 112$ and the average construction time is 118 milliseconds. However, S-boxes with higher nonlinearities, $NL \geq 114$, take hours to construct.

These results show that the proposed constrained optimization formulation has a significant speed advantage over existing heuristic optimization methods, which simply treat bijectivity and nonlinearity as objectives. Results also show that the proposed incremental coordinate-by-coordinate construction method is more efficient than existing methods that attempt to construct the S-box function as a single unit.

Next, we investigate the computational efficiency of the proposed LAP refinement algorithm (Algorithm 4). The algorithm was applied to initial S-boxes with varying nonlinearities. For each nonlinearity constraint, the experiment was repeated 100 times and the average time taken to reach a set of LAP objectives was reported in Table 7. It can be observed that the execution time of Algorithm 4 decreases for S-boxes with higher nonlinearity. This is because when the nonlinearity constraints are more restrictive, there are fewer candidate elements for the swapping process, which reduces the search space and consequently the execution time. When the nonlinearity is too high (NL≥114), the nonlinearity constraints are often too restrictive to allow for LAP improvement.

TABLE 7. S-box LAP refinement time for a sample of S-boxes generated by the proposed method.

NL_{min}	LAP Refinement time (s)			
	LAP=0.1172	LAP=0.1094	LAP=0.1016	LAP=0.0938
106	1.21	2.70	6.66	44.8
108	1.02	2.42	6.25	29.6
110	0.97	2.50	5.83	—
112	0.95	2.39	5.55	—

In traditional generate-test methods, most of the computational time is spent testing random S-boxes against security criteria. For instance, testing the NL criterion for an 8×8 S-box takes 2.5 ms using our implementation. With a generate-test method such as [20], an average of 7000 random S-boxes must be generated and tested to find an S-box with NL = 106, which requires a total of 17 seconds. On the other hand, the proposed generate-optimize method constructs an S-box in 2.76 ms with guaranteed NL = 106, thus avoiding the need for a posterior NL test. Namely, the gradual improvement of the constructed S-box allows incremental update of test results without the need for performing the tests all over after each optimization step. For higher NL criteria, traditional generate-test methods are computationally too expensive, whereas our method can generate dynamic S-boxes up to NL = 112 in real time, as shown in Table 6.

A similar argument can be made about the LAP test, which takes 75 ms using our implementation. Generate-test methods must generate hundreds of thousands of random S-boxes to find an S-box with LAP = 0.1094, which requires hours of testing. On the other hand, the proposed LAP refinement method, finds such an S-box in about 2.5 seconds.

DAP, SAC and BIC tests take 20 ms, 0.1 ms, 0.5 ms, respectively using our implementation, which can be expensive when repeated in generate-test methods. Since some cryptosystems require S-boxes to be tested for all criteria, the proposed generate-optimize method can be extended to include DAP, SAC and BIC as optimization objectives to avoid the time wasted in repeated testing.

V. CONCLUSION

In this work, a novel formulation of S-box design as a constrained optimization problem was proposed. The S-box is constructed incrementally, one coordinate function at a time, with nonlinearity as an objective, and both bijectivity and nonlinearity improvement as constraints. This formulation offers a considerable performance advantage over existing heuristic optimization methods. Compared with dynamic S-box construction methods in general, the proposed method produces better quality S-boxes than chaotic methods, offers a larger key space than algebraic method, and faster construction than other optimization methods. The proposed method successfully obtains a dynamic 8×8 S-box with nonlinearity 112 in an average of 118 ms, whereas nonlinearity 110 can be achieved in 5.3 ms, which enables real-time applications. Another advantage of the proposed method is the ability to construct dynamic S-boxes with higher nonlinearities, such as NL = 114. The constrained optimization formulation approach can be extended to other S-box design criteria. We demonstrated this extensibility by developing a linear-approximation-probability optimization algorithm with bijectivity and nonlinearity treated as constraints. We suggest that future work can extend our method to include differential uniformity objective. Another future prospect is to investigate the combination of nonlinearity and linear approximation probability as simultaneous objectives under bijectivity constraints.

APPENDIX
WORKED EXAMPLE OF ALGORITHM 1

To better understand how Algorithm 1 works, we provide this simple use case, in which we trace Algorithm 1 as it is invoked repeatedly to construct a bijective 3×3 S-box.

First, we invoke Algorithm 1 with $n = 3, j = 0$, and $S = (0, 0, \ldots, 0)$. We assume that $\mathcal{R}$ generates the sequence $(1, 0, 1, 1, 1)$. As shown in Figure 6, Algorithm 1 starts by initializing $\zeta_{0,0}$ and $\omega_{0,0}$ to zero. The maximum allowed value for each of them is $2^{n-j-1} = 4$. The loop counter, x, starts from 0 up to 7. When $x = 0$, both indicators, $\zeta_{0,0}$ and $\omega_{0,0}$ are smaller than the maximum, 4. Therefore, a random bit is drawn from $\mathcal{R}$, which happens to be 1. Consequently, $f_0(0)$ is set to 1, and the number of ones, $\omega_{0,0}$, is incremented to 1. In the next iteration, both indicators are still smaller than the maximum. Therefore, $\mathcal{R}$ is queried and returns 0. Hence, $f_0(x)$ is set to 0, and the number of zeros is incremented and so on. When x reaches 5, $\omega_{0,0}$ is already saturated at 4, which forces f_0 (5) to zero. The same happens with f_0 (6) and $f_0(7)$. Note that there is no need to query the random generator for any of these three elements.

$\mathcal{R}$		1	0	1	1	1			
x	$S(x)$	$f_0(x)$							
0	0	1							
1	0		0						
2	0			1					
3	0				1				
4	0					1			
5	0						0		
6	0							0	
7	0								0
$\zeta_{0,0}$	0		1						
$\omega_{0,0}$	0	1		2	3	4			

FIGURE 6. Calculating f_0 of 3×3 S–box using Algorithm 1.

The resulting Boolean function, f_0, is placed in the partially constructed S-box, $S_0 = f_0$, i.e., $S_0 = (1, 0, 1, 1, 1, 0, 0, 0)$.

To construct the next coordinate Boolean function, f_1, we invoke Algorithm 1 again with $n = 3$, $j = 1$, and $S = (1, 0, 1, 1, 1, 0, 0, 0)$. We assume that $\mathcal{R}$ generates the sequence $(0, 0, 1, 0)$. As shown in Figure 7, Algorithm 1 identifies $2^j = 2$ segments of f_1 and initializes the number of ones and number of zeros in each of the segment of f_1 to zeros, i.e., $\zeta_{1,0} = \omega_{1,0} = \zeta_{1,1} = \omega_{1,1} = 0$. The maximum allowed ones or zeros in each segment if $2^{n-j-1} = 2$.

At $x = 0$, the corresponding segment index is S (0), which is 1. Both indicators of segment 1, namely, $\zeta_{1,1}$ and $\omega_{1,1}$, are smaller than the maximum, 2. Therefore, $\mathcal{R}$ generates a random bit, which happens to be 0. Consequently, f_1 (0) is set to 0 and the corresponding indicator $\zeta_{1,1}$ is incremented.

$\mathcal{R}$		0	0	1	0		0		
x	S	$f_1(x)$							
0	1	0							
1	0		0						
2	1			1					
3	1				0				
4	1					1			
5	0						0		
6	0							1	
7	0								1
$\zeta_{1,0}$	0		1				2		
$\omega_{1,0}$	0								
$\zeta_{1,1}$	0	1			2				
$\omega_{1,1}$	0			1					

FIGURE 7. Calculating f_1 of 3×3 S–box using Algorithm 1.

When $x = 1$, the corresponding segment index is S (1), which is 0. All indicators of segment 0, $\zeta_{1,0}$ and $\omega_{1,0}$, are smaller than the maximum, 2. Therefore, $\mathcal{R}$ generates a random bit, which happens to be 0. Consequently, f_1 (1) is set to 0 and the corresponding indicator $\zeta_{1,0}$ is incremented. This goes on until $x = 4$, at which case, the corresponding number of zeros indicator, $\zeta_{1,1}$, is saturated at 2. Therefore, f_1 (4) is set to 1 without the need for querying $\mathcal{R}$.

At $x = 5$, the corresponding segment is neither saturated with zeros nor ones. So, f_1 (5) is chosen randomly using $\mathcal{R}$. When f_1 (5) is set to 0, the number of zeros in the corresponding segment, i.e., $\zeta_{1,0}$ reaches 2. As a result, both f_1 (6) and f_1 (7) are set to one, since both fall in the same segment, indicated by S (6) $= S(7) = 0$.

The resulting Boolean function, f_1, is appended to the partially constructed S-box, S, to obtain $S_1 = (f_0, f_1) = f_0 + 2f_1 = (1, 0, 3, 1, 3, 0, 2, 2)$.

To construct the last coordinate Boolean function, f_2, we invoke Algorithm 1 again with $n = 3$, $j = 2$, and $S = (1, 0, 3, 1, 3, 0, 2, 2)$. We assume that $\mathcal{R}$ generates the sequence $(0, 1, 0, 1)$. As shown in Figure 8, Algorithm 1 identifies $2^j = 4$ segments of f_2 and initializes the number of ones and number of zeros in each of the segment to zero. The maximum allowed ones or zeros in each segment is one.

At $x = 0$, the corresponding segment index is S (0) $= 1$. Both indicators of segment 1, $\zeta_{2,1}$ and $\omega_{2,1}$, are zero. So, f_2 (0) is chosen at random from $\mathcal{R}$, which generates 0. After setting f_2 (0) to 0, the number of zeroes in segment 1, i.e., $\zeta_{2,1}$ is incremented. f_2 (1) and f_2 (2) are treated similarly. However, at $x = 3$, the corresponding segment, which has index S (3) $= 1$, is already saturated with zeros, i.e., $\zeta_{2,1} = 1$. Therefore, f_2 (3) is determined to be one, without querying $\mathcal{R}$. f_2 (4), f_2 (5) and f_2 (7) are similarly determined, whereas f_2 (6) is chosen randomly by $\mathcal{R}$.

The resulting Boolean function, f_2, is appended to the partially constructed S-box, S, to obtain $S_2 = (f_0, f_1, f_2) = f_0 + 2f_1 + 4f_2 = (1, 4, 3, 5, 7, 0, 6, 2)$. The obtained 3×3 S-box is clearly bijective.

$\mathcal{R}$		0	1	0				1	
x	S	$f_2(x)$							
0	1	0							
1	0		1						
2	3			0					
3	1				1				
4	3					1			
5	0						0		
6	2							1	
7	2								0
$\zeta_{2,0}$	0								
$\omega_{2,0}$	0		1						
$\zeta_{2,1}$	0	1							
$\omega_{2,1}$	0								
$\zeta_{2,2}$	0								
$\omega_{2,2}$	0							1	
$\zeta_{2,3}$	0			1					
$\omega_{2,3}$	0								

FIGURE 8. Calculating f_2 of 3 × 3 S–box using Algorithm 1.

Following a similar procedure, a bijective $n \times n$ S-box, for any arbitrary n, can be incrementally constructed by repeatedly invoking Algorithm 1 n times. It's worth noting that Algorithm 1 only guarantees bijectivity. To optimize nonlinearity, Algorithm 2 should be invoked after each call to Algorithm 1.

REFERENCES

[1] M. Ahmad, E. Al-Solami, A. M. Alghamdi, and M. A. Yousaf, "Bijective S-boxes method using improved chaotic map-based heuristic search and algebraic group structures," *IEEE Access*, vol. 8, pp. 110397–110411, 2020.

[2] M. S. M. Malik, M. A. Ali, M. A. Khan, M. Ehatisham-Ul-Haq, S. N. M. Shah, M. Rehman, and W. Ahmad, "Generation of highly nonlinear and dynamic AES substitution-boxes (S-Boxes) using chaos-based rotational matrices," *IEEE Access*, vol. 8, pp. 35682–35695, 2020.

[3] M. Fahad Khan, A. Ahmed, K. Saleem, and T. Shah, "A novel design of cryptographic SP-network based on gold sequences and chaotic logistic tent system," *IEEE Access*, vol. 7, pp. 84980–84991, 2019.

[4] A. Razaq, H. Alolaiyan, M. Ahmad, M. A. Yousaf, U. Shuaib, W. Aslam, and M. Alawida, "A novel method for generation of strong substitution-boxes based on coset graphs and symmetric groups," *IEEE Access*, vol. 8, pp. 75473–75490, 2020.

[5] S. Ibrahim, H. Alhumyani, M. Masud, S. S. Alshamrani, O. Cheikhrouhou, G. Muhammad, M. S. Hossain, and A. M. Abbas, "Framework for efficient medical image encryption using dynamic S-Boxes and chaotic maps," *IEEE Access*, vol. 8, pp. 160433–160449, 2020.

[6] D. Lambić, "A new discrete-space chaotic map based on the multiplication of integer numbers and its application in S-box design," *Nonlinear Dyn.*, vol. 100, no. 1, pp. 699–711, 2020.

[7] Q. Lu, C. Zhu, and X. Deng, "An efficient image encryption scheme based on the LSS chaotic map and single S-Box," *IEEE Access*, vol. 8, pp. 25664–25678, 2020.

[8] F. Özkaynak, "On the effect of chaotic system in performance characteristics of chaos based s-box designs," *Phys. A, Stat. Mech. Appl.*, vol. 550, Jan. 2020, Art. no. 124072.

[9] A. A. Abd El-Latif, B. Abd-El-Atty, and S. E. Venegas-Andraca, "A novel image steganography technique based on quantum substitution boxes," *Opt. Laser Technol.*, vol. 116, pp. 92–102, Aug. 2019.

[10] A. H. Zahid and M. J. Arshad, "An innovative design of substitution-boxes using cubic polynomial mapping," *Symmetry*, vol. 11, no. 3, 2019, Art. no. 437.

[11] L. Yi, X. Tong, Z. Wang, M. Zhang, H. Zhu, and J. Liu, "A novel block encryption algorithm based on chaotic S-Box for wireless sensor network," *IEEE Access*, vol. 7, pp. 53079–53090, 2019.

[12] D. Lambić, "S-box design method based on improved one-dimensional discrete chaotic map," *J. Inf. Telecommun.*, vol. 2, no. 2, pp. 181–191, 2018.

[13] X. Wang, A. Akgul, U. Cavusoglu, V.-T. Pham, D. V. Hoang, and X. Nguyen, "A chaotic system with infinite equilibria and its S-box constructing application," *Appl. Sci.*, vol. 8, no. 11, p. 2132, 2018.

[14] E. Al Solami, M. Ahmad, C. Volos, M. Doja, and M. Beg, "A new hyperchaotic system-based design for efficient bijective substitution-boxes," *Entropy*, vol. 20, no. 7, p. 525, 2018.

[15] L. Liu, Y. Zhang, and X. Wang, "A novel method for constructing the S-box based on spatiotemporal chaotic dynamics," *Appl. Sci.*, vol. 8, no. 12, p. 2650, 2018.

[16] D. Lambić, "A novel method of S-box design based on discrete chaotic map," *Nonlinear Dyn.*, vol. 87, no. 4, pp. 2407–2413, Mar. 2017.

[17] A. Belazi, M. Khan, A. A. A. El-Latif, and S. Belghith, "Efficient cryptosystem approaches: S-boxes and permutation-substitution-based encryption," *Nonlinear Dyn.*, vol. 87, no. 1, pp. 337–361, Jan. 2016.

[18] A. Belazi and A. A. A. El-Latif, "A simple yet efficient S-box method based on chaotic sine map," *Optik*, vol. 130, pp. 1438–1444, Feb. 2017.

[19] M. Khan and Z. Asghar, "A novel construction of substitution box for image encryption applications with Gingerbreadman chaotic map and S_8 permutation," *Neural Comput. Appl.*, vol. 29, no. 4, pp. 993–999, 2016.

[20] D. Lambić, "A novel method of S-box design based on chaotic map and composition method," *Chaos, Solitons Fractals*, vol. 58, pp. 16–21, Jan. 2014.

[21] S. S. Jamal, A. Anees, M. Ahmad, M. F. Khan, and I. Hussain, "Construction of cryptographic S-Boxes based on mobius transformation and chaotic tent-sine system," *IEEE Access*, vol. 7, pp. 173273–173285, 2019.

[22] A. Zahid, M. Arshad, and M. Ahmad, "A novel construction of efficient substitution-boxes using cubic fractional transformation," *Entropy*, vol. 21, no. 3, p. 245, 2019.

[23] T. Shah and D. Shah, "Construction of highly nonlinear S-boxes for degree 8 primitive irreducible polynomials over $\mathbb{Z}_2$," *Multimedia Tools Appl.*, vol. 78, no. 2, pp. 1219–1234, 2019.

[24] K. E. Attaullah, S. S. Jamal, and T. Shah, "A novel algebraic technique for the construction of strong substitution box," *Wireless Pers. Commun.*, vol. 99, no. 1, pp. 213–226, 2018.

[25] A. Belazi, A. A. A. El-Latif, A.-V. Diaconu, R. Rhouma, and S. Belghith, "Chaos-based partial image encryption scheme based on linear fractional and lifting wavelet transforms," *Opt. Lasers Eng.*, vol. 88, pp. 37–50, Jan. 2017.

[26] M. Khan and N. A. Azam, "Right translated AES gray S-boxes," *Secur. Commun. Netw.*, vol. 8, no. 9, pp. 1627–1635, 2015.

[27] J. Daemen and V. Rijmen, *The Design of Rijndael*. New York, NY, USA: Springer-Verlag, 2002, p. 255.

[28] M. M. Dimitrov, "On the design of chaos-based S-Boxes," *IEEE Access*, vol. 8, pp. 117173–117181, 2020.

[29] Y. Wang, Z. Zhang, L. Y. Zhang, J. Feng, J. Gao, and P. Lei, "A genetic algorithm for constructing bijective substitution boxes with high nonlinearity," *Inf. Sci.*, vol. 523, pp. 152–166, Jan. 2020.

[30] T. Zhang, C. L. Philip Chen, L. Chen, X. Xu, and B. Hu, "Design of highly nonlinear substitution boxes based on I-Ching operators," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3349–3358, Dec. 2018.

[31] A. A. Alzaidi, M. Ahmad, M. N. Doja, E. A. Solami, and M. M. S. Beg, "A new 1D chaotic map and β-hill climbing for generating substitution-boxes," *IEEE Access*, vol. 6, pp. 55405–55418, 2018.

[32] M. Ahmad, M. N. Doja, and M. M. S. Beg, "ABC optimization based construction of strong substitution-boxes," *Wireless Pers. Commun.*, vol. 101, no. 3, pp. 1715–1729, 2018.

[33] A. A. Alzaidi, M. Ahmad, H. S. Ahmed, and E. A. Solami, "Sine-cosine optimization-based bijective substitution-boxes construction using enhanced dynamics of chaotic map," *Complexity*, vol. 2018, pp. 1–16, Dec. 2018.

[34] M. Rodinko, R. Oliynykov, and Y. Gorbenko, "Optimization of the high nonlinear S-boxes generation method," *Tatra Mountains Math. Publications*, vol. 70, no. 1, pp. 93–105, 2017.

[35] T. Farah, R. Rhouma, and S. Belghith, "A novel method for designing S-box based on chaotic map and teaching-learning-based optimization," *Nonlinear Dyn.*, vol. 88, no. 2, pp. 1059–1074, 2017.
[36] M. Ahmad, D. Bhatia, and Y. Hassan., "A novel ant colony optimization based scheme for substitution box design," *Procedia Comput. Sci.*, vol. 57, pp. 572–580, Jan. 2015.
[37] X. Wang, Ü. Çavusoglu, S. Kacar, A. Akgul, V. T. Pham, S. Jafari, F. E. Alsaadi, and X. Q. Nguyen, "S-box based image encryption application using a chaotic system without equilibrium," *Appl. Sci.*, vol. 9, no. 4, p. 781, 2019.
[38] A. U. Rehman, J. S. Khan, J. Ahmad, and S. O. Hwang, "A new image encryption scheme based on dynamic s-boxes and chaotic maps," *3D Res.*, vol. 7, no. 1, p. 7, 2016.
[39] P. Devaraj and C. Kavitha, "An image encryption scheme using dynamic S-boxes," *Nonlinear Dyn.*, vol. 86, no. 2, pp. 927–940, 2016.
[40] A. Ullah, S. S. Jamal, and T. Shah, "A novel scheme for image encryption using substitution box and chaotic system," *Nonlinear Dyn.*, vol. 91, no. 1, pp. 359–370, 2018.
[41] E. Hasanzadeh and M. Yaghoobi, "A novel color image encryption algorithm based on substitution box and hyper-chaotic system with fractal keys," *Multimedia Tools Appl.*, vol. 79, nos. 11–12, pp. 7279–7297, 2019.
[42] A. Belazi, A. A. A. El-Latif, and S. Belghith, "A novel image encryption scheme based on substitution-permutation network and chaos," *Signal Process.*, vol. 128, pp. 155–170, Nov. 2016.
[43] X. Zhang, Y. Mao, and Z. Zhao, "An efficient chaotic image encryption based on alternate circular S-boxes," *Nonlinear Dyn.*, vol. 78, no. 1, pp. 359–369, Oct. 2014.
[44] X.-P. Zhang, R. Guo, H.-W. Chen, Z.-M. Zhao, and J.-Y. Wang, "Efficient image encryption scheme with synchronous substitution and diffusion based on double S-boxes," *Chin. Phys. B*, vol. 27, no. 8, 2018, Art. no. 080701.
[45] A. Biryukov and L. Perrin, "On reverse-engineering S-boxes with hidden design criteria or structure," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 2015, pp. 116–140.
[46] W. Meier and O. Staffelbach, "Nonlinearity criteria for cryptographic functions," presented at the Adv. Cryptol. (EUROCRYPT), 1989.

SALEH IBRAHIM received the B.Sc. and M.Sc. degrees in computer engineering from Cairo University, Egypt, in 2000 and 2004, respectively, and the Ph.D. degree in computer science and engineering from the University of Connecticut, USA, in 2010.

He is currently an Assistant Professor with the Electrical Engineering Department, Taif University, Saudi Arabia. He has been an Assistant Professor with the Computer Engineering Department, Cairo University, since 2011. He has published several research articles in high impact journals and international conferences. His current research interests include information security and computer networks.

ALAA M. ABBAS received the Ph.D. degree from Menofia University, Egypt, in 2008.

He is currently an Associate Professor with the Electronics and Electrical Communications Engineering Department, Faculty of Electronic Engineering, Menofia University. He is also an Assistant Professor with the Electrical Engineering Department, College of Engineering, Taif University, Saudi Arabia. His areas of interests include image processing, watermarking, image encryption, and cryptography.