New Genetic Operators for Searching for S-Boxes with Low Boomerang Uniformity

MAN KANG 1,2, AND MINGSHENG WANG 1,2

1The State Key Lab of Information Security, Institute of Information Engineering, Chinese Academy of Science, Beijing 100093, China
2School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Corresponding author: Man Kang (e-mail: kangman@iie.ac.cn).

This work was supported by the National Natural Science Foundation of China (No. 61772516, 61772517).

I. INTRODUCTION

The boomerang attack [1] is a variant of the differential attack. For ciphers that the probabilities of the differential characteristics decrease exponentially with respect to the growth of rounds, the boomerang attack can concatenate two short characteristics to form a longer characteristic with a better probability. In boomerang attack, two short parts \( E_0 \) and \( E_1 \) make up a larger characteristic \( E \). Assume that \( p \) is the probability of the differential characteristic \((\alpha, \beta)\) for \( E_0 \), and \( q \) is the probability of the differential characteristic \((\gamma, \delta)\) for \( E_1 \). Then the probability of the boomerang distinguisher is

\[ \Pr \left[ E^{-1} (E(x) \oplus \delta) \oplus E^{-1} (E(x \oplus \alpha) \oplus \delta) = \alpha \right] = p^2 q^2. \]

The boomerang attack is an effective cryptanalysis tool, which has been successfully applied to famous block ciphers such as AES, IDEA and SHACAL1 [2]–[5].

Boomerang connectivity table (BCT) [6] provides a unified representation for boomerang-style attacks, which has become a new tool of substitution boxes (S-boxes) for more accurately evaluating the probability of generating a right quartet in boomerang-style attacks. The boomerang uniformity [7] is the maximum value in BCT among all nonzero input differences and output differences that measures the resistance of an S-box to a boomerang attack. S-boxes are crucial nonlinear building blocks providing confusion in modern block ciphers. The emergence of cryptographic attacks has led to the development of criteria for resisting such attacks. Existing attacks require S-boxes to meet some cryptographic properties, including bijectivity, low differential uniformity [8], and high nonlinearity [9]. With the development of boomerang attacks, boomerang uniformity has become a new essential criterion for the S-box, which has attracted the interest of researchers.

Boura and Canteaut [7] completely characterized the BCT of all differentially 4-uniform permutations of 4 bits and then studied these objects for inverse functions and quadratic permutations. Their work provided the first examples of differentially 4-uniform S-boxes optimal against boomerang attacks for an even number of variables. The boomerang unifor-
mities of some specific permutations were studied in [10] and a class of 4-uniform BCT permutation polynomials over \(\mathbb{F}_{2^n}\) were obtained. Mesnager et al. [11] focused their research on the boomerang uniformity of quadratic permutations in even dimensions. A new family of optimal S-boxes was found by generalizing previous results on quadratic permutation with optimal BCT. Calderini and Villa [12] further studied the boomerang uniformity of some non-quadratically differentially 4-uniform functions. Wang et al. [13] studied the boomerang uniformity of all normalized permutation polynomials of degree up to six over the arbitrary finite field \(\mathbb{F}_2\) by using the resultant elimination method. Li et al. [14] presented infinite families of permutations of \(\mathbb{F}_{2^{2n}}\) for a positive odd integer \(n\), which have the best-known nonlinearity and boomerang uniformity 4.

In addition to mathematical methods, intelligent methods have also been used to create S-boxes in recent years. Reinforcement learning was used to train a method expressed in the Markov decision process to an agent to generate S-boxes that can effectively resist the side-channel attack [15]. Heuristic evolution strategy improved the initial S-Box created by a modular operation [16]. The S-box construction time was reduced by constrainedly maximizing the nonlinearity of the S-boxes created by a random-restart hill-climbing algorithm [17]. The S-boxes based on Chaos were designed in [18]. The combination of the Chaos method and intelligent algorithm was also used to generate S-boxes. An artificial bee colony algorithm was used to optimize the S-boxes generated by chaotic sequence [19]. A β-hill climbing search was applied to improve the S-boxes based on chaotic map [20]. As an intelligent algorithm simulating the evolution of nature, the genetic algorithm provides a practical solution to the combinatorial optimization problem that is difficult to deal with by traditional methods and provides a new idea and means for the complex problems in cryptography.

Genetic algorithms have been increasingly used to generate S-boxes with good performances in recent years. The traditional genetic algorithm was used to generate S-boxes with good values of the confusion coefficient in terms of improving their side-channel resistance [21]. A method based on chaos and the genetic algorithm was proposed by [22] for designing an S-box. The full use of the traits of chaotic map and evolution process makes it possible to obtain a stronger S-box. A genetic algorithm working in a reversed way was proposed by [23], which can rapidly and repeatedly generate a large number of strong bijective S-boxes. Several genetic algorithms and problem sizes were explored by [24] to find functions having differential uniformity equal to 6. In addition, simulated annealing and genetic algorithm were used to optimize the design of symmetric-key primitives in [24].

S-boxes constructed by the Feistel structure have the advantage of low hardware implementation cost [26]; however, they have high boomerang uniformities. In this paper, a new genetic algorithm is introduced to improve the properties of the S-boxes created by the Feistel structure. The new genetic algorithm generates \(8 \times 8\) bijective S-boxes with low differential uniformity, high nonlinearity, and low boomerang uniformity. This is the first time a meta-heuristic algorithm has been used to search for S-boxes with low boomerang uniformity. A new crossover operator and a new mutation operator are proposed to improve the performance of the genetic algorithm. Benefiting from the full use of the advantages of gene exchange and gene mutation, the new genetic algorithm in this paper dramatically improves the properties of the S-boxes created by the Feistel structure. In addition, we compare the S-boxes generated by the new genetic algorithm and the S-boxes generated by the traditional genetic algorithm. The comparison results show that the S-boxes generated by our new genetic algorithm have better properties than those generated by the traditional genetic algorithm. The experimental results show the effectiveness and superiority of our new genetic algorithm.

This paper is organized as follows. Section 2 gives some preliminaries on necessary concepts. Section 3 describes our new genetic algorithm and the traditional genetic algorithm. Section 4 illustrates the experimental parameters and gives the results of this paper. Then the results are compared and analyzed. Finally, section 5 concludes this paper.

II. PRELIMINARIES

A bijective \(n \times n\) S-box is a permutation on \(\mathbb{F}_2^n\). Mathematically, S-box is a vectorial Boolean function \(F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n\), which can be defined as a vector \(F = (f_1, f_2, ..., f_n)\). The Boolean functions \(f_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2\) are called the coordinate function of \(F\). The component functions of an \(n \times n\)-function \(F\) are all the linear combinations of the coordinate functions with non all-zero coefficients.

**Definition 1 (Differential uniformity [8]):** Let \(F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n\) be a \(n \times n\) vectorial Boolean function. The derivative of S-box \(F\) with regard to vector \(a \in \mathbb{F}_2^n\) is \(b = F(x \oplus a) \oplus F(x)\). The difference distribution table (DDT) of \(F\) is

\[
\text{DDT}_F(a, b) = \# \{ x \in \mathbb{F}_2^n | F(x) \oplus F(x \oplus a) = b \}.
\]

The symbol \# here represents the number of solutions in the set. Differentially \(\delta_F\)-uniform is the maximum value of \(\text{DDT}_F(a, b)\) for every non-zero \(a \in \mathbb{F}_2^n\) and every \(b \in \mathbb{F}_2^n\), i.e.,

\[
\delta_F = \max_{a \in \mathbb{F}_2^n, a \neq 0} \text{DDT}_F(a, b).
\]

**Definition 2 (Nonlinearity and linearity [9]):** Let \(F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n\) be a \(n \times n\) vectorial Boolean function. The nonlinearity of an S-box \(F\) is defined as the minimum Hamming distance between all non-zero component functions of \(F\) and all \(n\)-variable affine Boolean functions, which can be represented by the Walsh spectrum,

\[
N_F = 2^{n-1} - \frac{1}{2} \max_{a, b \in \mathbb{F}_2^n, b \neq 0} |W_F(a, b)|.
\]
The linearity of a S-box is defined as
\[ \mathcal{L}_F = \max_{a, b \in \mathbb{F}_2^n \setminus \{0\}} |\mathcal{W}_F(a, b)| = \max_{a, b \in \mathbb{F}_2^n \setminus \{0\}} 2|\text{LAT}_F(a, b)|. \]

The linearly of a S-box is then defined as
\[ \mathcal{L}_F = \frac{\log (|\mathcal{F}|)}{\log (2^n)} \cdot \mathcal{L}_F. \]

Definition 3 (Boomerang uniformity [7]): Let \( F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \) be a \( n \times n \) invertible vectorial Boolean function. For input difference \( a \in \mathbb{F}_2^n \) and output difference \( b \in \mathbb{F}_2^n \), the entries of the boomerang connectivity table (BCT) are defined as
\[ \text{BCT}_F(a, b) = \# \{ x \in \mathbb{F}_2^n | F^{-1}(F(x) + b) + F^{-1}(F(x + a) + b) = a \}, \]
where \( F^{-1} \) denotes the compositional inverse of \( F \). The boomerang uniformity of \( F \) is defined as
\[ \beta_F = \max_{a, b \in \mathbb{F}_2^n, a, b \neq 0} \text{BCT}_F(a, b). \]

III. GENETIC ALGORITHMS

Genetic algorithm [27] is a computational model that simulates the evolution process of nature, which has been successfully applied to various optimization problems. In recent years, many researchers have also applied genetic algorithms to design block cipher primitives.

The genetic algorithm principle is based on Darwinian natural selection and Mendelian genetics. The selection method allows high-quality individuals to be more likely to survive and improves the quality of individuals in the population. Mendelian genetics provides a theoretical basis for the population to produce new individuals. The crossover operator recombines the genes of the two-parent individuals to generate two new individuals, which is the primary way to generate new individuals. The mutation operator generates new individuals by changing the genes at specific loci. As the primary way of generating new individuals in genetic algorithm, genetic operators have a significant impact on the performance of genetic algorithm. Traditional genetic operators are universal, but they cannot guarantee to generate better new individuals. We design new genetic operators for the genetic algorithm to produce better individuals in the process of evolution.

Algorithm 1 depicts the framework of our genetic algorithm. In Algorithm 1, the size of population \( P \) is \( N \). Individuals in the initial population are created by an unbalanced Feistel structure. \( r_p \) is a randomly generated probability. The parents in the population perform crossover according to probability \( p_c \). The probability of mutation is \( p_m \). In our work, the termination condition of the genetic algorithm is that the maximum number \( \text{MAX} \) of generations is reached. \( \mathcal{C}_F \) is the fitness function that calculates the fitness value \( f_p \) for the individual. Next, the components of the genetic algorithm will be introduced in detail.

**Algorithm 1** The Framework of Our Genetic Algorithm

1: for each \( p \in P \) do
2: \( p \leftarrow \text{Unbalanced Feistel structure}; \)
3: \( f_p \leftarrow \mathcal{C}_F(p); \)
4: end for
5: \( g \leftarrow 0; // \text{Number of iterations} \)
6: while \( g < \text{MAX} \) do
7: \( g++; \)
8: //Tournament selection operator;
9: for \( i \in [0, \frac{N}{2}] \) do
10: \( k \text{ individuals are randomly selected;} \)
11: \( \text{Two individuals with the lowest fitness values are copied into the new population;} \)
12: end for
13: //The process of crossover;
14: for \( i \in [0, \frac{N}{2}] \) do
15: if \( r_p < p_c \) then
16: \( (p, q) \leftarrow \text{randomly select two individuals from the population;} \)
17: \( (p, q) \leftarrow \text{Crossover operator (p, q);} \)
18: \( f_p \leftarrow \mathcal{C}_F(p); \)
19: \( f_q \leftarrow \mathcal{C}_F(q); \)
20: end if
21: end for
22: //The process of mutation;
23: for \( i \in [0, \frac{N}{2}] \) do
24: if \( r_p < p_m \) then
25: \( p \leftarrow \text{The i-th individual in P;} \)
26: \( p \leftarrow \text{Mutation operator (p);} \)
27: \( f_p \leftarrow \mathcal{C}_F(p); \)
28: end if
29: end for
30: end while

a: Permutation Encoding

The form of permutation encoding is intuitively more suitable for representing S-boxes. In this representation, the bijectivity property is automatically satisfied. An \( n \times n \) S-box is represented as an array of \( 2^n \) integer numbers with elements in range \([0, 2^n - 1]\). Each value occurs exactly once in an array and represents one entry for the S-box lookup table.

b: Initial Population

Individuals in the initial population are created by an unbalanced Feistel structure. We extend the method in [26] to generate \( 8 \times 8 \) S-boxes. Let \( f \) is a seven-variable nonlinear Boolean function, and the variable is \( x_i \in \mathbb{F}_2, 1 \leq i \leq 7 \).
One round conversion of unbalanced Feistel structure is
\[
t(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_0 \oplus f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)),
\]
where
\[
f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = x_1 \cdot x_2 \oplus x_3 \cdot x_4 \oplus x_5 \cdot x_6.
\]
$r_1, r_2, r_3, r_4, r_5, r_6, r_7$ are random integers in $[1, 7]$.

### A. NEW GENETIC OPERATORS

#### a: New Crossover Operator

In order to improve the performance of the genetic algorithm, we design a new crossover operator for the genetic algorithm. The fitness function considers three properties of an S-box: differential uniformity, linearity, and boomerang uniformity. The smaller their values, the better. In each iteration, the new crossover operator takes advantage of gene exchange to reduce the values of three properties.

The new crossover operator is described in Algorithm 3. First, randomly select two individuals $p$ and $q$ from the population as the two parents. Let $p = p_0, \cdots, p_{2^n-1}$ and $q = q_0, \cdots, q_{2^n-1}$. The crossover processes performed on $p$ and $q$ are similar. We take individual $p$ as an example.

Find the input-output differential pair $(a, b)$ that satisfies $\text{DDT}_p(a, b) = \delta_p$ in the differential distribution table. For each pair $(a, b)$, find $p_i, i \in [0, 2^n-1]$ that increases $\text{DDT}_p(a, b)$ of $p$, and exchange $p_i$ and $p_j$ to obtain a new individual $p'$, where $p_j = q_i$. If $\delta_p' \leq \delta_p$, $\text{L}_p' \leq \text{L}_p$, $\beta_p' \leq \beta_p$, $\text{DDT}_p'(a, b) \leq \text{DDT}_p(a, b)$ and no new value is added to $\delta_p$ after exchange, replace $p$ with $p'$. If $\text{DDT}_p'(a, b) < \text{DDT}_p(a, b)$, find the next input-output differential pair $(a, b)$ satisfying $\text{DDT}_p'(a, b) = \delta_p'$ and repeat the process. If $\text{DDT}_p'(a, b) = \text{DDT}_p(a, b)$, find the elements adding $\text{DDT}_p(a, b)$ in $p'$ and perform the same operation to reduce $\text{DDT}_p(a, b)$. The process of reducing the boomerang uniformity is similar to that of reducing the differential uniformity. When reducing the linearity, it should be considered in two cases: $\text{L}_p = \max_{a, b \in \mathbb{F}_2^n \setminus \emptyset} \text{LAT}_p(a, b)$ and $\text{L}_p' = \max_{a, b \in \mathbb{F}_2^n \setminus \emptyset} -\text{W}_p(a, b)$. The purpose of gene exchange is to reduce the number of $b \cdot p_x = a \cdot x$; when $\text{L}_p = \max_{a, b \in \mathbb{F}_2^n \setminus \emptyset} -\text{W}_p(a, b)$, the purpose of gene exchange is to reduce the number of $b \cdot p_x \neq a \cdot x$.

#### b: New Mutation Operator

This paper also designs a new mutation operator. Randomly select a position $c_1 \in [0, 2^n-1]$. Exchange the gene at position $c_1$ in individual $p$ with genes at other positions in $p$ in turn to generate new individuals. For the new individual $p'$, if one or more of $\delta_p' < \delta_p$, $\text{L}_p' < \text{L}_p$ and $\beta_p' < \beta_p$ are satisfied, replace the original individual $p$ with the new individual $p'$; otherwise, retain the original individual $p$ and delete the new individual $p'$. The mutation process is described in Algorithm 4.

### B. TRADITIONAL GENETIC OPERATORS

#### a: Traditional Crossover Operator

The partially mapped crossover (PMX crossover) [29] is the traditional crossover operator we use. Randomly select two individuals $p$ and $q$ from the population as the two parents. Let $p = p_0, \cdots, p_{2^n-1}$ and $q = q_0, \cdots, q_{2^n-1}$. Randomly select two positions $(c_1, c_2)$, $c_1, c_2 \in [0, 2^n-1]$, and exchange the gene fragments of the two parents between $c_1$ and $c_2$. Check the elements in the uncrossed gene segment.
Algorithm 3 New Crossover Operator

Input: Parent individuals \( p \) and \( q \)
Output: Offspring individuals \( p \) and \( q \)

1. \( p'' \leftarrow p; \)
2. \( p \leftarrow \) Exchange \((p, q)\);
3. \( q \leftarrow \) Exchange \((q, p'')\);
4. return \( p \) and \( q \);
5. The procedure of Exchange \((p, q)\);
6. \( p \leftarrow \) ReduceProcess\((p, q, \text{DDT})\);
7. \( p \leftarrow \) ReduceProcess\((p, q, \text{LAT})\);
8. \( p \leftarrow \) ReduceProcess\((p, q, \text{-LAT})\);
9. \( p \leftarrow \) ReduceProcess\((p, q, \text{BCT})\);
10. return \( p \);
11. The procedure of ReduceProcess \((p, q, T)\):
12. if \( T = \text{DDT} \) then
13. uniformity \((a, b) \leftarrow \text{DDT}_p(a, b) = \delta_p \);
14. condition \((p, q, i) \leftarrow p_i \oplus p_i \oplus a = b; \)
15. increase \((x, y) \leftarrow \text{DDT}_p(x, y) = \delta_p \) and \( \text{DDT}_q(x, y) = \delta_q \);
16. else if \( T = \text{LAT} \) then
17. uniformity \((a, b) \leftarrow \text{LAT}_p(a, b) = \frac{1}{2} \mathcal{L}_p; \)
18. condition \((p, q, i) \leftarrow b \cdot p_i = a \cdot i; \)
19. increase \((x, y) \leftarrow \text{LAT}_p(x, y) = \frac{1}{2} \mathcal{L}_p \) and \( \text{LAT}_q(x, y) < \frac{1}{2} \mathcal{L}_p; \)
20. else if \( T = \text{-LAT} \) then
21. \( T = \text{LAT}; \)
22. uniformity \((a, b) \leftarrow \text{LAT}_p(a, b) = -\frac{1}{2} \mathcal{L}_p; \)
23. condition \((p, q, i) \leftarrow b \cdot p_i \neq a \cdot i; \)
24. increase \((x, y) \leftarrow |\text{LAT}_p(x, y)| = \frac{1}{2} \mathcal{L}_p \) and \(|\text{LAT}_q(x, y)| \leq \frac{1}{2} \mathcal{L}_p; \)
25. else if \( T = \text{BCT} \) then
26. uniformity \((a, b) \leftarrow \text{BCT}_p(a, b) = \beta_p; \)
27. condition \((p, q, i) \leftarrow p^{-1}(p_i \oplus b) \oplus p^{-1}(p_i \oplus a \oplus b) = a; \)
28. increase \((x, y) \leftarrow \text{BCT}_p(x, y) = \beta_p \) and \( \text{BCT}_q(x, y) < \beta_q; \)
29. end if
30. for \( \text{uniformity} \,(a, b), a, b \in [0, 2^n - 1]\) do
31. \( p' \leftarrow \) Exchange \( p_i \) and \( p_j, \) where \( p_i = q_j; \)
32. for \( \text{condition} \,(a, b, i), i \in [0, 2^n - 1]\) do
33. if \( \delta_p \leq \beta_p, \mathcal{L}_{p'} \leq \mathcal{L}_p, \beta_{p'} \leq \beta_p, T_p(a, b) \leq T_p(a, b) \) and \( \mathcal{B}_{x, y} \in [0, 2^n - 1], \) increase \((x, y)\) then
34. \( p \leftarrow p'; \)
35. If \( T_p(a, b) \) is reduced, jump out of this loop;
36. end if
37. end for
38. end for
39. return \( p; \)

Algorithm 4 New Mutation Operator

Input: Parent individual \( p \)
Output: Offspring individual \( p \)

1: \( c_1 \leftarrow \) Randomly generate a position;
2: for \( i \in [0, 2^n - 1] \) do
3: if \( i \neq c_1 \) then
4: \( p' \leftarrow \) Exchange \( p_i \) and \( p_{c_1}; \)
5: \( f_{p'} \leftarrow \text{DDT}(p_{p'}); \)
6: if \( f_{p'} < f_p \) or \( f_{p'} = f_p, \delta_{p'} < \delta_p \) or \( f_{p'} = f_p, \delta_{p'} < \delta_p, \mathcal{L}_{p'} < \mathcal{L}_p; \) then
7: \( p \leftarrow p'; \)
8: end if
9: end if
10: end for
11: return \( p; \)

of the first parent \( p. \) If an element is the same as the element at position \( j, j \in [c_1, c_2], \) replace it with the element \( p_j \) in \( q. \) Repeat this process until \( p \) becomes a permutation with no repeating elements. Then perform the same operation on the second parent \( q. \) This process is described in Algorithm 5.

Algorithm 5 Traditional Crossover Operator

Input: Parent individuals \( p \) and \( q \)
Output: Offspring individuals \( p \) and \( q \)

1: \((c_1, c_2) \leftarrow \) Randomly generate two positions;
2: exchange\((p; q; c_1, c_2) = \)
3: end if
4: \( q \) and \( \mathcal{B}_{c_1, c_2}; \)
5: \( \mathcal{B}_{q_1, q_2}; \)
6: end if
7: \( q \) and \( \mathcal{B}_{q_1, q_2}; \)
8: end if
9: end if
10: end for
11: return \( p; \)

b: Traditional Mutation Operator

The inversion mutation [30] is the traditional mutation operator we use. First, two positions \((c_1, c_2), c_1, c_2 \in [0, 2^n –


TABLE 1. The best S-box created by the unbalanced Feistel structure.

| p1 | p2 | p3 | p4 | p5 | p6 | p7 | p8 | p9 | p10 | p11 | p12 | p13 | p14 | p15 | p16 | p17 | p18 | p19 | p20 | p21 | p22 | p23 | p24 | p25 | p26 | p27 | p28 | p29 | p30 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 154 | 150 | 48 | 38 | 27 | 130 | 171 | 6 | 56 | 159 | 249 | 16 | 96 | 49 | 53 | 23 |
| 164 | 100 | 122 | 155 | 151 | 251 | 176 | 143 | 136 | 239 | 111 | 28 | 47 | 144 | 62 | 98 |
| 234 | 190 | 138 | 33 | 133 | 157 | 120 | 25 | 17 | 218 | 72 | 223 | 244 | 128 | 118 | 18 |
| 252 | 149 | 66 | 188 | 35 | 233 | 182 | 146 | 34 | 229 | 240 | 191 | 167 | 107 | 137 | 201 |
| 205 | 46 | 231 | 183 | 253 | 225 | 37 | 185 | 119 | 180 | 233 | 195 | 132 | 245 | 57 | 217 |
| 61 | 140 | 213 | 256 | 244 | 87 | 209 | 200 | 238 | 242 | 32 | 129 | 60 | 177 | 199 |
| 198 | 142 | 127 | 3 | 153 | 118 | 230 | 147 | 36 | 93 | 65 | 15 | 227 | 67 | 241 | 235 |
| 169 | 103 | 71 | 85 | 192 | 106 | 236 | 102 | 160 | 106 | 204 | 113 | 42 | 81 | 82 |
| 83 | 174 | 175 | 80 | 70 | 163 | 110 | 134 | 75 | 78 | 69 | 9 | 39 | 210 | 105 | 116 |
| 14 | 186 | 226 | 189 | 8 | 90 | 84 | 162 | 250 | 196 | 79 | 88 | 41 | 12 | 89 | 74 |
| 43 | 172 | 254 | 19 | 95 | 104 | 181 | 152 | 246 | 24 | 194 | 74 | 30 | 55 | 2 | 178 |
| 220 | 48 | 207 | 63 | 193 | 165 | 168 | 161 | 117 | 94 | 166 | 237 | 109 | 108 | 86 | 135 |
| 219 | 248 | 255 | 115 | 10 | 4 | 52 | 197 | 121 | 247 | 131 | 212 | 203 | 44 | 124 | 54 |
| 170 | 215 | 126 | 179 | 21 | 11 | 50 | 58 | 68 | 91 | 20 | 5 | 77 | 40 | 64 | 187 |
| 97 | 221 | 6 | 59 | 52 | 222 | 31 | 125 | 216 | 158 | 7 | 51 | 141 | 128 | 206 | 99 |
| 139 | 243 | 232 | 228 | 184 | 208 | 211 | 92 | 13 | 76 | 173 | 112 | 29 | 202 | 145 | 204 |

TABLE 2. The best S-box generated by the new genetic algorithm.

| Table | Population | Uniformity | Nonlinearity | Boomerang |
|-------|------------|------------|--------------|-----------|
| 154   | 150        | 83         | 60           | 150       |
| 164   | 100        | 52         | 60           | 150       |
| 234   | 190        | 83         | 60           | 150       |
| 252   | 149        | 83         | 60           | 150       |
| 205   | 46         | 83         | 60           | 150       |
| 61    | 140        | 83         | 60           | 150       |
| 198   | 142        | 83         | 60           | 150       |
| 169   | 103        | 83         | 60           | 150       |
| 83    | 174        | 83         | 60           | 150       |
| 14    | 186        | 83         | 60           | 150       |
| 43    | 172        | 83         | 60           | 150       |
| 220   | 48         | 83         | 60           | 150       |
| 219   | 248        | 83         | 60           | 150       |
| 170   | 215        | 83         | 60           | 150       |
| 97    | 221        | 83         | 60           | 150       |
| 139   | 243        | 83         | 60           | 150       |

1] are randomly selected for the individual \( p \) to be mutated, where \( p = p_0, \ldots, p_{c_1}, \ldots, p_{c_2}, \ldots, p_{2^n-1} \). Then, the genes between two positions \( (c_1, c_2) \) in individual \( p \) are arranged in an inverted order to obtain a new \( p = p_0, \ldots, p_{c_2}, \ldots, p_{c_1}, \ldots, p_{2^n-1} \).

IV. EXPERIMENTAL SETUP AND RESULTS

This paper uses the new genetic algorithm and the traditional genetic algorithm to search for \( 8 \times 8 \) S-boxes with low differential uniformity, high nonlinearity, and low boomerang uniformity.

A. EXPERIMENTAL SETUP

For the traditional genetic algorithm and the new genetic algorithm, we run 30 experiments, respectively. Except for the different genetic operators, the other parameters of the two genetic algorithms are the same. The parameter values are determined based on experience and experimental feedback. The population size \( N \) is 256. The tournament size \( \beta \) is set to 3. Different crossover probabilities and mutation probabilities have no significant impact on the search of traditional genetic algorithm. The higher the crossover probability and the mutation probability for our new genetic algorithm, the better. Therefore, we set these two parameters to relatively large values. Crossover probability \( p_c = 0.9 \), and mutation probability \( p_m = 0.1 \). The maximum number of iterations is determined by observing the output of experimental results, and \( \text{MAX}=400 \).

B. EXPERIMENTAL RESULTS AND ANALYSIS

In our work, in addition to the differential uniformity and nonlinearity, we also consider the boomerang uniformity. Table 3 shows the distribution of these three properties in the initial population and the end population. The data 16#13 in Table 3 means that the number of differential uniformity \( \delta_p = 16 \) in the initial population is 13.

TABLE 3. The property distribution of S-boxes in the population.

| Distribution | Initial population | End population |
|--------------|--------------------|----------------|
| Best \( \delta_p \) | 16/13              | 6/4            |
| Maximum \( \delta_p \) | 64/91              | 12/9          |
| Best \( \mathcal{L}_p \) | 96/86              | 10/8          |
| Maximum \( \mathcal{L}_p \) | 64/120             | 94/10         |
| Best \( \beta_p \) | 52/1               | 10/8          |
| Maximum \( \beta_p \) | 256/186            | 20/87         |

As can be seen from Table 3, for the initial population created by the unbalanced Feistel structure, the differential uniformity, nonlinearity, and boomerang uniformity are concentrated at 64, 64, and 256, respectively. The best differential uniformity, nonlinearity, and boomerang uniformity in
the initial population are 16, 96 and 52, respectively. The best S-box created by the unbalanced Feistel structure is given in Table 1. It can be seen that the properties of the initial population created by the unbalanced Feistel structure are not ideal, especially the boomerang uniformity. At the end of the iteration, the differential uniformity, nonlinearity, and boomerang uniformity of the population obtained by our new genetic algorithm are concentrated at 12, 94, and 20, respectively. At this time, the best values of differential uniformity, nonlinearity, and boomerang uniformity in the population are 6, 108, and 10. At the end of the population, there are 119 $\delta_p \leq 10, 47 N_p \geq 96$ and 9 $\beta_p \leq 16$. According to the comparison in Table 3, on the whole, the new genetic algorithm improves the properties of S-boxes created by the unbalanced Feistel structure.

In Table 4, S-box1-S-box4 are generated by our new genetic algorithm, and S-box5 is generated by the traditional genetic algorithm. It can be seen from Table 4 that the S-box1 and S-box2 generated by our new genetic algorithm have the best cryptographic properties: the lowest differential uniformity 6, the highest nonlinearity 108, and the lowest boomerang uniformity 10. Table 2 shows the lookup table of S-box1.

Table 5 compares the cryptographic properties of S-boxes generated in different ways. The values of random S-box are the expected values of differential uniformity, nonlinearity, and boomerang uniformity given in [33]–[34]. As can be seen from Table 5, the S-box generated by the new genetic operator has better properties than the random S-box. Moreover, the S-box generated by the new genetic operator is comparable with those generated by other methods.

In summary, the new genetic algorithm has successfully improved the properties of the S-boxes created by the unbalanced Feistel structure. Moreover, the S-boxes generated by the new genetic algorithm have better properties than those generated by the traditional genetic algorithm, demonstrating the effectiveness and superiority of the new genetic algorithm in searching S-boxes.

V. CONCLUSIONS

In this paper, a genetic algorithm is used to improve the properties of the S-boxes created by the Feistel structure. New genetic operators are designed for the genetic algorithm to generate $8 \times 8$ S-boxes with low differential uniformity, high nonlinearity, and low boomerang uniformity. It is the first time that a genetic algorithm has been used to improve the boomerang uniformity of the S-box. Experimental results show that the new genetic algorithm successfully improves the properties of the S-boxes created by the Feistel structure. The S-boxes generated by the new genetic algorithm have better properties than those generated by the traditional genetic algorithm, which shows the effectiveness and superiority of the new genetic algorithm in generating S-boxes. In the future, genetic algorithms can be used to generate S-boxes of different sizes. On the other hand, other new genetic operators can be designed to generate S-boxes with better performances.

REFERENCES

[1] D. Wagner, “The Boomerang Attack”, in Fast Software Encryption, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, Vol.1636, 1999, pp.156–170.
[2] B. Eli, D. Orr and K. Nathan, “Related-Key Boomerang and Rectangle Attacks”, in Advances in Cryptology – EUROCRYPT 2005, Springer, Berlin, Heidelberg, 2005, pp.507–525.
[3] H. Seokhie, K. Jongsung, L. Sangjin, and P. Bart, “Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192”, in Fast Software Encryption, Springer, Berlin, Heidelberg, 2005, pp.386–393.
[4] O. Dunkelman, N. Keller and J. Kim, “Related-Key Rectangle Attack on the Full SHACAL-1”, in Selected Areas in Cryptography, Lecture Notes in Computer Science, Springer, Montreal, Canada, Vol.4356, 2006, pp.28–44.
[5] K. Jongsung, H. Seokhie and P. Bart, “Related-Key Rectangle Attacks on Reduced AES-192 and AES-256”, in Fast Software Encryption, Springer, Berlin, Heidelberg, 2007, pp.225–241.
[6] C. Carlos, H. Tao, P. Thomas, S. Yu, and S. Ling, “Boomerang Connectivity Table: A New Cryptanalytic Tool”, in Advances in Cryptology – EUROCRYPT 2018, Springer International Publishing, Cham, 2018, pp.683–714.
[7] C. Boura and A. Canteaut, “On the Boomerang Uniformity of Cryptographic Sboxes”, IACR Transactions on Symmetric Cryptology, Springer, vol. 2018, no. 3, pp.290–310, Sept. 2018.
[8] N. Kaisa, “Differentially uniform mappings for cryptography”, in Advances in Cryptology – EUROCRYPT ’93, Springer, Berlin, Heidelberg, 1994, pp.55–64.
[9] C. Carlet, “Vectorial Boolean Functions for Cryptography”, Boolean Models and Methods in Mathematics, Cambridge University Press, Cambridge, pp.396–470, 2010.
[10] K. Li, L. Qu, B. Sun and C. Li, “New Results About the Boomerang Uniformity of Permutation Polynomials”, IEEE Transactions on Information Theory, Vol.65, No.11, pp.7542-7553, 2019.
[11] S. Mesnager, C.M. Tang and M.S. Xiong, “On the boomerang uniformity of (quadratic) permutations over $F_{2^n}$”, Designs, Codes and Cryptography, Vol.80, 2020, DOI:10.1007/s10623-020-00439-x.
[12] S. Mesnager, C.M. Tang and M.S. Xiong, “On the boomerang uniformity of $F_{2^n}$”, Designs, Codes and Cryptography, Vol.80, 2020, DOI:10.1007/s10623-020-00439-x.
[13] Y. P. Wang, Q. Wang, and W. G. Zhang, “Boomerang uniformity of normalized permutation polynomials of low degree”, Applicable Algebra in Engineering, Communication and Computing, pp.307-322, 2020, DOI:10.1007/s00200-020-00431-1.
[14] K. Li, C. Li, T. Helleseth, and L. Qu, “Cryptographically strong permutations from the butterfly structure”, Designs Codes and Cryptography, Vol.89, No.3, 2021, DOI:10.1007/s10623-020-00837-5.
MAN KANG received the B.S. degree from Hebei GEO University, Shijiazhuang, China, in 2013, and the M.S. degree from the Beijing University of Chemical Technology, Beijing, China, in 2017. She is currently pursing the Ph.D. degree at the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. Her research interests include cryptography and information security.

MINGSHENG WANG received the Ph.D. degree from Beijing Normal University, Beijing, China, in 1994. He is currently a Professor with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing. His research interests include computational algebra, cryptography, and information security.