1. Introduction

As the basic nonlinear components, Boolean functions can achieve the confusion and diffusion for ciphers (pp. 398–399 of [1]). When we apply them to cryptosystems, Boolean functions should embrace excellent cryptographic properties, such as balancedness, correlation immunity, high nonlinearity, high algebraic degree, and high algebraic immunity. However, all such characteristics cannot be optimum at the same time, and trade-offs should be of consideration. Therefore, constructions of Boolean functions with compromise criteria are always a challenging open problem [2, 3].

A metaheuristic is designed to generate an almost best solution to an optimization problem with guidance between local improvement and higher level strategies. In the literature, lots of papers used heuristic algorithms to search for cryptographically important Boolean functions and several long open problems had been solved [4–6]. Hill climbing (HC) and genetic algorithm (GA) were firstly applied to search for highly nonlinear Boolean functions in 1996 [7, 8] by modifying the true table of a Boolean function. Because the HC is a local search procedure and cannot generally achieve global optimization, then global metaheuristics, such as genetic algorithm, simulated annealing, ant colony, and their hybridization with HC, have been presented to improve the solutions between the conflicting cryptographic criteria [9–12]. For example, in 1998, the authors introduced a genetic algorithm combined with HC and found balanced Boolean functions in 6, 8, and 10 variables with nonlinearity greater than \(2^{n-1} - 2^{n/2}\) [9]. However, their method seems not to be valid always for all situations.

Because rotation is a useful operator which can speed up the performance of the ciphers and preserve the security at the same time [13–15], recently, RSBFs have been attracted to be researched because of their advantages in cryptographic algorithms for the simple structure, fast speed, high resource utilization, and their richness of cryptographically significant Boolean functions [16–18]. We call a Boolean function RS if its outputs are invariant under the input of the cyclic shift. Using a steepest-descent-like algorithm, Kavut et al. [4] have searched Boolean functions in 9 variables with nonlinearity 241. This led to solving an almost three-decade-old open problem if there exist Boolean functions of 9 variables whose nonlinearity greater than bent concatenation bound 240. However, this algorithm cannot guarantee the balancedness of Boolean functions. By applying simulated
annealing (SA) to 9-variable RSBFs and with some algebraic
techniques, Liu and Youssef [6] constructed 10-variable
Boolean functions with algebraic degree 7, resiliency degree
2, and nonlinearity 488. This result has answered the open
problem about the existence of such functions in [3]. Mo-
tivated by the previous work, searching for RSBFs with
multiple cryptographical properties should be further
investigated.

In this study, we generalize the traditional genetic al-
gorithm and apply it to search for balanced functions with
high nonlinearity in the class of RSBFs. The experimental
results demonstrate our method can generate excellent
Boolean functions with high nonlinearity, 1-resilient func-
tions, and optimal algebraic immunity. We can also obtain
bent functions which have been applied widely in crypto-
graphy, spread spectrum, coding theory, and combina-
torial design. It shows that these functions have superiority
from the point of practical application in cryptosystems
compared with known ones that are obtained by other
heuristics. We organize this paper as follows. In Section 2, we
introduce some preliminary definitions and useful results.
Section 3 describes the traditional genetic algorithm. Based
on this algorithm, we propose a modification of GA named
GA-reset. We also design an algorithm to generate balanced
RSBFs. A generality of GA-reset is presented when we
pursue the high nonlinearity of RSBF. By combining these
algorithms proposed in this paper, we have obtained ex-
cellent RSBFs with the variables of 8, 10, and 12. We give a
conclusion in Section 4.

2. Preliminaries
2.1. Boolean Functions. Let $GF(2)^n$ be the $n$-dimensional
vector space over the finite field $GF(2) = \{0, 1\}$. Denote by $\phi$
the addition operation over $GF(2)$. Let 0 and 1 be the all-
zero vector and the all-one vector of $GF(2)^n$, respectively.
An $n$-variable Boolean function $f(x)$ can be represented
uniquely as an $n$-variable polynomial, called its algebraic
normal form (ANF). An $n$-variable Boolean function $f(x)$,
where $x = (x_0, x_1, \ldots, x_{n-1}) \in GF(2)^n$, is a mapping from
$GF(2)^n$ to $GF(2)$, which can be represented uniquely as an
$n$-variable polynomial, called its algebraic normal form
(ANF):

$$f(x_0, x_1, \ldots, x_{n-1}) = \bigoplus_{u \in GF(2)^n} \lambda_u \left( \prod_{i=0}^{n-1} x_i^{u_i} \right), \quad \lambda_u \in GF(2).$$

(1)

The algebraic degree is defined as the number of variables
in the highest order product term with nonzero coefficient.
A Boolean function is said to be affine if its degree does not
exceed 1. The set of all $n$-variable affine functions is denoted
by $A_n$. We call a function nonlinear if it is not in $A_n$. The
Hamming weight $w_H(f)$ of a binary vector $x \in GF(2)^n$ is the
number of its nonzero coordinates, and the Hamming weight
$w_H(f)$ of a Boolean function $f$ is the size of its support
$\{x \in GF(2)^n : f(x) = 1\}$. If $w_H(f) = 2^n - 1$, we call
$f(x)$ balanced. Let $\mathcal{B}_n$ denote the set of Boolean functions
of $n$ variables. Given $x, w \in GF(2)^n$ with $x = (x_0, x_1, \ldots, x_{n-1})$,
w = $(w_0, w_1, \ldots, w_{n-1})$, let $w \cdot x$ be an inner product in
$GF(2)^n$, for instance, the usual inner product
$w_0x_0 + w_1x_1 + \cdots + w_{n-1}x_{n-1}$. Then, the Walsh
coefficients for a Boolean function $f(x) \in \mathcal{B}_n$ are the values of the real
valued function over $GF(2)^n$ defined by

$$W_f(w) = \sum_{x \in GF(2)^n} (-1)^{f(x) \cdot w \cdot x}, \quad \text{for all } w \in GF(2)^n.$$  

(2)

The Walsh spectrum of the Boolean function $f$ is the set
of all the Walsh coefficients $W_f(w)$.

For convenience, we use $W_f$ instead of $W_f(0)$. It is easy
to derive the following elementary identity:

$$W_f = 2^n - 2w_H(f),$$

(3)

and the well-known formula (see Th. 2.17, p.13, of [2])

$$N_f = 2^{n-1} - \frac{1}{2} \max_{w \in GF(2)^n} |W_f(w)|.$$  

(4)

Definition 1. Let $n$ be even. A Boolean function on $GF(2)^n$
is called a bent function if and only if its Walsh transform satisfies

$$W_f(w) = \pm 2^{n/2}, \quad \text{for all } w \in GF(2)^n.$$  

(5)

Definition 2. A Boolean function on $GF(2)^n$ is called
t-resilient if and only if its Walsh coefficients satisfied

$$W_f(w) = 0, \quad \text{for } 0 \leq w_H(w) \leq t, \quad w \in GF(2)^n.$$  

(6)

From Xiao–Messay theorem [19], the algebraic degree of a
t-resilient function $f \in \mathcal{B}_n$ is at most $n - t - 1$. Let
$x = (x_0, x_1, \ldots, x_{n-1}) \in GF(2)^n$. For $0 \leq i, k \leq n - 1$, we
define the left $k$-cyclic shift operator $\rho_k^n$ acting on $x_i$ as

$$\rho_k^n(x_i) = x_{(i+k) \mod n} \quad \text{(that is, } \rho_k^n \text{ permutes the indices of co-
ordinates of } x).$$

We can extend the definition of $\rho_k^n$ on tuples as follows:

$$\rho_k^n(x_0, x_1, \ldots, x_{n-1}) = (\rho_k^n(x_0), \rho_k^n(x_1), \ldots, \rho_k^n(x_{n-1})).$$  

(7)

Let $G_n$ be the cyclic group of the permutation

$$\{\rho_k^n : 0 \leq k \leq n - 1\},$$

and we denote by

$$G_n(x_0, \ldots, x_{n-1}) = \{\rho_k^n(x_0, \ldots, x_{n-1}) : 0 \leq k \leq n - 1\},$$

(8)

the orbit of $(x_0, x_1, \ldots, x_{n-1})$ under the action of $G_n$.

It is obvious that $G_n(x_1, x_2, \ldots, x_n)$ generates a partition
of the vector space $GF(2)^n$. It is shown in [20] that the
number of orbits of $GF(2)^n$ is exactly

$$g_n = \frac{1}{n!} \sum_{t \in \mathbb{Z}} \phi(t) 2^{nt},$$  

(9)

where $\phi$ is Euler’s function.

2.2. Related Work. The term genetic algorithm (GA) was
first used by John Holland in 1995 based on Darwinian
evolution theory. Followed by Spillman [21] and Clark [22],
it was shown that GA has been successfully applied in
cryptanalysis of classical ciphers and modern ciphers [8, 9, 23, 24]. In evolution of Boolean functions, Millan et al. [7, 8] firstly applied GA to find Boolean functions with high nonlinearity. By introducing a resetting step, they combined GA with HC and obtained balanced Boolean functions with high nonlinearity [8]. However, most of the previous work applied several fitness functions to obtain Boolean functions with multiple cryptographical criteria. In this study, we will show that one can obtain cryptographically strong Boolean function by using the fitness function defined in step (2) of Algorithm 1.

GAs is inspired by bio-operators such as mutation, crossover, and selection. It usually starts from a sample of individuals which is generated randomly. In each iteration, there is an iterative process with the sample, which is called a generation. In a genetic algorithm, the sample with candidate solutions (i.e., individuals) is expected to evolve toward better solutions. For more details, see [25].

3. Searching for Cryptographically Strong RSBFs

We represent the individuals as truth tables of Boolean functions. However, when the search space is restricted to the class of RSBFs, each orbit indicates a gene and the length of the crossover is equal to the number of the orbits. If a bit in the truth table of an RSBF is changed, then it means that all outputs corresponding to an orbit should be changed to obtain another RSBF. Take 10-variable RSBFs as an example. There are \(g_{10} = 108\) orbits of \(GF(2)^{10}\); we list them in Table 1.

The genetic algorithm searching for RSBFs is designed as Algorithm 1.

Remark 1. The function in step (2) was first proposed by [26] to measure the cryptographical stability of a Boolean function. Kavut et al. make use of it in their steepest-descent-like iterative algorithm and find RSBFs in 9 variables with nonlinearity 241 [4]. Because this fitness function mimics the squared distance of a Boolean function with even number of variables to bent functions in terms of Walsh spectra, therefore, we can expect a highly nonlinear RSBF with the minimum of it. By experiments, we found that when the initial population size is 30, the efficiency of the algorithm and the scale of the solutions have the best trade-off.

3.1. A Modification of GA-Rest. In the previous algorithm, the “child” solution produced by the “parents” solutions with the genetic operators, crossover and mutation, is generally not a balanced RSBF. Therefore, we improve it as Algorithm 2. Let \(p_1\) and \(p_2\) be the parent \(n\)-variable balanced RSBFs and \(d(p_1, p_2)\) be the Hamming distance of \(p_1\) and \(p_2\). Denote by \(c\) the child bred by \(p_1\) and \(p_2\). Let \(n_1\) be the number of 1 of the truth table of \(c\) restricted to the indexes such that the parents bits are different. The objective of the algorithm is to generate a balanced RSBF such that \(n_1 = d(p_1, p_2)\). Note that all entries corresponding to an orbit should be changed to obtain another RSBF if one bit of the truth table of an RSBF is complemented.

Remark 2. Note that the complementing truth table of a Boolean function does not change its nonlinearity. The check in Step 2 of Algorithm 2 is to ensure that only the parents who are close to each other are allowed to breed. The checks in Steps 9 and 12 are used to force the child RSBF to be balanced. Experimental results show that these modifications are benefit for obtaining better solutions.

3.2. Results and Discussion. Denote by \((n, m, d, N_f, AI)\) the profile of a Boolean function as the number of its input variable, resiliency order, algebraic degree, nonlinearity, and algebraic immunity. Particularly, we denote by \((n, -1, d, N_f, AI)\) and by \((n, 0, d, N_f, AI)\) the unbalanced functions and the balanced functions, respectively. In this section, we perform the traditional GA and GA-rest to search RSBFs with 10 variables to determine which algorithm is better. By programming the traditional GA, the highest nonlinearity of the RSBFs achieved is 484, and it is balanced, but it is not resilient. Meantime, we have obtained many RSBFs with \((10, 1, -1, 480, 5)\). We present its truth table in a hexadecimal format.

By programming GA-reset, we have found balanced RSBFs with nonlinearity 486, which are higher than the results generated by the traditional GA. And also, we have obtained RSBFs with \((10, 1, 8, 484, 5)\). The following is one of the examples:

\[
\begin{align*}
E895932C75B1A1C07E32C0FD3D0A8C546EB94B & \text{58E1BF540E7F7551B8C51 90E13307C99DF620CF32C1} \\
F2D09B9AF 776455AD9A2A86A33653CED1E076468754 & \text{BD460E4B1F02DA9592E72EBF52D 4954BFA0B49F053F} \\
AF3400896DAD8E93E7A29742675B39E9C19CD0806 & \text{C6C8 6CE1B3D65B08A8A657AD402F383469.}
\end{align*}
\]

We collect the best results of the two algorithms in Table 2.

The results show that though the efficiency of GA-reset is lower than GA, the solutions obtained are better than that of GA. It seems that the efficiency of GA-reset and converge of the solutions is in a reasonable trade-off between them.

3.3. Searching for RSBF with High Nonlinearity. If we extend the “generic operators” in Algorithm 1 to all pairs of the current generations, then the algorithm can converge quickly to bent functions. In searching for the class of 10-variable RSBFs, it takes \(145^934\) to obtain bent functions. Most of them are with optimal algebraic immunity. That constructing bent function with optimal algebraic immunity has always been an open problem until the method present in [27]. This shows that the new algorithm converges to global optimal solutions targeting nonlinearity. Together with Algorithm 2, we can search balanced RSBFs with the highest nonlinearity compared with the known algorithms. We state it in Algorithm 3.

Remark 3. It shows that though the efficiency is lower than previous algorithms, we can generate a bent RSBF within an acceptable time. Together with Algorithm 2, we can generate balanced RSBFs with strongly cryptographical properties.
Table 1: The orbits of $GF(2)^{10}$.

<table>
<thead>
<tr>
<th>The length of the orbit</th>
<th>The number of the orbits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>99</td>
</tr>
</tbody>
</table>

Algorithm 1: GA-reset.

(1) **Initialization.** To be balanced.
(2) **Fitness function.** Let the fitness function be
Fitness = $-\sum_{w \in GF(2)^n} (W^j_i w - 2^n)^2$
(3) **Genetic operators**

- **Two-point crossover.** The two crossover points (potential solution) are chosen randomly on the parent truth tables of the rotation symmetric functions (RSTTs). All bits between these two points are swapped between the parents, rendering two child RSTTs.
- **Mutation.** The purpose of mutation in GAs is introducing diversity. According to the mutation probability $p_m$, we chose some orbits of the RSTT and complete it. We checked the efficiency of the algorithm for the mutation probability of 0.2, 0.1, and 0.05 and found that when $p_m = 0.05$, the output is optimum.
- **Selection.** The fitness function is evaluated for each individual and then the fitness values are normalized. For the $k$th individual with fitness value $f_k$, then its probability of being selected is $p_i = f_i / \sum_{j=1}^{N} f_j$, where $N$ is the number of individuals in the population. Compute the cumulative probability distribution $F_k = \sum_{i=1}^{k} p_i$ and generate a uniform random number $\xi_k$ ranging in $[0, 1)$, then, the $k$th individual is selected if $F_{k-1} \leq \xi_k < F_k$.

(4) **Resetting.** As in [9], we add the resetting step to the traditional GAs. That is, if the fitness of the best solution cannot be improved after a number of iterations, then we retain the best solution and randomly generate $N - 1$ balanced RSBFs.
(5) **Termination.** Because of the randomness of GAs, there must be enough iterations so that the solutions can be convergent. Thus, we assign the number of iterations to be 100,000.

Algorithm 2: Generation of a balanced RSBF.

Input: Two RSBFs
Output: A balanced RSBF
(1) int $n_1 = 0, k = 0$
(2) if $d(p_1, p_2) > 2^{n-1}$ then
(3) for $i = 1$ to $g_n$ do
(4) $p_1[i] = p_1[i] + 1$;
(5) end for
(6) end if
(7) for $i = 1$ to $g_n$ do
(8) if $p_1[i] = p_2[i]$ then
(9) $c[i] = p_1[i]$;
(10) else
(11) if $n_1 = d(p_1, p_2)/2$ then
(12) $c[i] = 0$
(13) else
(14) if $n_1 + d(p_1, p_2) - k = d(p_1, p_2)/2$ then
(15) $c[i] = 1$;
(16) else $c[i]$ is randomly equal to 0 or 1.
(17) $k = k + 1$;
(18) if $c[i] = 1$ then
(19) $n_1 = n_1 + g_n$;
(20) end if
(21) end if
(22) end if
(23) end if
(24) end for return $c$. 

Algorithm 2: Generation of a balanced RSBF.
In the remainder of this section, we apply Algorithm 3 to search for 8, 10, and 12-variable RSBFs and the maximum number of the iterations 100,000. We get RSBFs with \((12, -1, 10, 1998, 6)\). They cannot be linearly transformed to balanced functions since there is no zero in their Walsh spectrum. However, we find RSBFs which are \((12, 0, 10, 1996, 6)\) and there are 76 zeroes in its Walsh spectrum, which can be linearly transformed to 1-resilient functions[10]. We also find RSBFs which are \((12, 1, 10, 1992, 6)\) and one of Algorithms 2 and 3 of the examples is in Section Appendix.

We collect and compare the known results obtained by heuristics in Table 3.

### Conclusion

Rotation symmetric Boolean functions have an advantage in cryptosystems since they can be described lightly. In this study, we search for balanced RSBF with excellent cryptographic properties by designing heuristic algorithms. The experimental results have proved that there is a reasonable trade-off between the efficiency of our algorithms and the convergence of the RSBFs. Bent functions can also be generated by the algorithm. By programming the algorithms shown in this study, we have obtained excellent RSBFs with the variables of 8, 10, and 12. This strategy is shown to be significantly superior to some known algorithms.

### Appendix

The truth table of \((12, 1, 10, 1992, 5)\) is in the hexadecimal format.

<table>
<thead>
<tr>
<th>Method</th>
<th>8</th>
<th>10</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2^{n/2}−2^{n/2})</td>
<td>118</td>
<td>480</td>
<td>1984</td>
</tr>
<tr>
<td>GA&amp;HC[8, 9]</td>
<td>116</td>
<td>486</td>
<td>1976</td>
</tr>
<tr>
<td>Algorithms 2 and 3</td>
<td>116</td>
<td>488</td>
<td>1996</td>
</tr>
</tbody>
</table>

| Table 3: Comparison of the nonlinearity of balanced Boolean functions. |
Data Availability

The data used to support the findings of this study have been uploaded to github (https://github.com/kistoday/cryptographically-significant-rotation-symmetric-boolean-functions).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Nature Science Foundation of China, under Grant no. 61872381.

References


