computers

Article

Genetic Approach to Improve Cryptographic Properties of Balanced Boolean Functions Using Bent Functions

Erol Özçekiç [1,2], Selçuk Kavut [3,*] and Hakan Kutucu [2]

[1] Computer Technologies Department, Vocational School, Balıkesir University, 10145 Balıkesir, Turkey; erolozcekic@balikesir.edu.tr
[2] Software Engineering Department, Faculty of Engineering, Karabük University, 78050 Karabük, Turkey; hakankutucu@karabuk.edu.tr
[3] Computer Engineering Department, Faculty of Engineering, Balıkesir University, 10145 Balıkesir, Turkey
* Correspondence: skavut@balikesir.edu.tr

Abstract: Recently, balanced Boolean functions with an even number n of variables achieving very good autocorrelation properties have been obtained for $12 \leq n \leq 26$. These functions attain the maximum absolute value in the autocorrelation spectra (without considering the zero point) less than $2^{\frac{n}{2}}$ and are found by using a heuristic search algorithm that is based on the design method of an infinite class of such functions for a higher number of variables. Here, we consider balanced Boolean functions that are closest to the bent functions in terms of the Hamming distance and perform a genetic algorithm efficiently aiming to optimize their cryptographic properties, which provides better absolute indicator values for all of those values of n for the first time. We also observe that among our results, the functions for $16 \leq n \leq 26$ have nonlinearity greater than $2^{n-1} - 2^{\frac{n}{2}}$. In the process, our search strategy produces balanced Boolean functions with the best-known nonlinearity for $8 \leq n \leq 16$.

Keywords: absolute indicator; Boolean function; genetic algorithm; nonlinearity

check for updates

Citation: Özçekiç, E.; Kavut, S.; Kutucu, H. Genetic Approach to Improve Cryptographic Properties of Balanced Boolean Functions Using Bent Functions. *Computers* **2023**, *12*, 159. https://doi.org/10.3390/computers12080159

Academic Editors: Ömer Aslan and Refik Samet

Received: 15 July 2023
Revised: 3 August 2023
Accepted: 7 August 2023
Published: 9 August 2023

1. Introduction

In symmetric cryptography, Boolean functions used in a cryptosystem are essential building blocks, and two of their most significant cryptographic properties are nonlinearity and an absolute indicator. Nonlinearity should be high to resist the best affine approximation attacks [1] (in the case of stream ciphers) and linear cryptanalysis [2] (in the case of block ciphers). On the other hand, it is important to achieve a low absolute indicator that, while providing resistance against differential and fault attacks (e.g., [3,4]) on stream ciphers, ensures good diffusion properties. For an even number of variables, optimal Boolean functions in terms of these two properties exist, and such functions are called bent [5,6]; however, since bent functions are not balanced, they cannot be directly used in a cryptosystem. Therefore, it is important to construct balanced Boolean functions with high nonlinearities and low absolute indicators.

Let us denote the maximum nonlinearity of a balanced n-variable Boolean function by $\mathrm{nlb}(n)$. It has been conjectured [7] for an even n that $\mathrm{nlb}(n)$ can be at most $2^{n-1} - 2^{\frac{n}{2}} + \mathrm{nlb}(\frac{n}{2})$, which is still unsettled. In [7], this nonlinearity was achieved by making the all-zero block (of a length $2^{\frac{n}{2}}$) of a normal bent function balanced, which indeed provides the minimum number ($2^{\frac{n}{2}-1}$) of bit changes necessary to make a bent function balanced. However, it is well-known that highly nonlinear Boolean functions exist with better absolute indicator values than those of the construction in [7]. In this direction, it was conjectured [8] for an even n that the maximum absolute indicator of a balanced Boolean function cannot be less than $2^{\frac{n}{2}}$. This conjecture was first disproved in [9] for $n = 14$ by utilizing a heuristic search method beginning with a randomly generated bent function as its starting function. Shortly after that, by performing a steepest-descent-like search method within the class of 10-variable rotation-symmetric Boolean functions, it was also disproved [10] for $n = 10$. Apart from these two search results, theoretical constructions of balanced n-variable Boolean functions with an absolute indicator of less than $2^{\frac{n}{2}}$ were obtained recently in [11,12] for $n \geq 46$ ($n \equiv 2 \mod 4$) and for $n \geq 52$ ($n \equiv 0 \mod 4$), respectively. By suitably modifying the underlying Boolean functions of the constructions in [11,12], specific examples for $n = 18$, 22, and 26 are demonstrated in [11] and, subsequently, using the steepest-descent-like search method, these results are further improved in [12], which provides additional examples for $n = 12$, 14, 16, 20, and 24. Informally speaking, the Boolean functions obtained in [11,12] are constructed by modifying two blocks (with a length of $2^{\frac{n}{2}+1}$ bits) of Dillon's bent function [5].

Evident from the above discussion, balanced Boolean functions with an even number of variables n achieving an absolute indicator of less than $2^{\frac{n}{2}}$ are mostly generated by modifying the bent functions. Here, we perform a genetic algorithm that also exploits a bent function to generate such balanced Boolean functions; however, there are some subtle differences, as follows:

- Our search is realized among the balanced Boolean functions that are at a distance with a length of $2^{\frac{n}{2}-1}$ from a bent function, and, hence, compared to, e.g., [9], we do not need to seek to achieve the balancedness property during the search.
- Though the Boolean functions generated from the construction in [7] have the same distance, our search strategy does not restrict the corresponding bit changes to a specific block of a bent function.
- We do not impose the conditions of a theoretical construction (e.g., [11,12]); rather, our search is more generic, and its search space is all the balanced Boolean functions that are closest to a bent function in terms of the Hamming distance.

Here, we point out that for the balanced n-variable Boolean functions that we consider, a modified hill-climbing method was applied previously in [13] for $n = 8, 10$, and 12. In addition, for these numbers of variables, the bent functions were used in [14] to form the initial population of a hybrid genetic algorithm, which was exploited to construct balanced Boolean functions without restricting their distance to the bent functions. Our approach basically applies a global optimization algorithm to the search space used in [13] for even $n \in [8, 26]$, which yields better cryptographic properties than those obtained in the related literature. More specifically, the absolute indicators that we find improve the best-known values for $n \geq 12$, and the Boolean functions with these absolute indicators achieve nonlinearity of greater than $2^{n-1} - 2^{\frac{n}{2}}$ for $n \geq 16$. Further, the best-known nonlinearities are obtained for $n \leq 16$.

Over the years, researchers have explored different metaheuristic techniques (such as local search, simulated annealing, particle swarm optimization, genetic algorithm, genetic programming, etc.) to design Boolean functions in various dimensions that satisfy multiple cryptographic properties. In the related literature, most of the works consider Boolean functions with numbers of the variables of up to at most 16, and the most commonly sought cryptographic properties to be optimized are balancedness, nonlinearity, and absolute indicator. We refer the reader to [15] and the references therein for a recent survey of metaheuristic algorithms for the design of cryptographic Boolean functions. In the following section, we give a brief background on the cryptographic properties of Boolean functions. For a comprehensive survey and discussion of Boolean functions, we like to refer the reader to [16,17]. Next, in Section 3, we describe the traditional genetic algorithm and present our search strategy, keeping the distance to a bent function unchanged during the search. The details of our search effort are given in Section 4, and then we draw our conclusions in Section 5.

2. Preliminaries

Let $f : GF(2)^n \rightarrow GF(2)$ be an n-variable Boolean function, which is usually defined by its truth table $f = [f(0, \cdots, 0, 0), f(0, \cdots, 0, 1), \ldots, f(1, \cdots, 1, 1)]$ of a length 2^n. The Hamming weight $wt(f)$ of a Boolean function f is the number of ones in its truth table, and the Hamming distance $d(f_1, f_2)$ between two Boolean functions f_1 and f_2, both with the same number of input variables, is the number of places for which f_1 and f_2 differ in their truth tables, i.e., $d(f_1, f_2) = wt(f_1 \oplus f_2)$. If $wt(f) = 2^{n-1}$ for an n-variable Boolean function f, then f is said to be balanced. A Boolean function f with input variables $(x_0, x_1, \ldots, x_{n-1}) \in GF(2)^n$ can be considered as a sum of the products (with all distinct orders) of the variables, i.e., it can be written as a multivariate polynomial over $GF(2)$:

$$\bigoplus_{u \in GF(2)^n} a_u \prod_{i=0}^{n-1} x_i^{u_i}, \tag{1}$$

where the coefficients $a_u \in GF(2)$. This representation is unique and called the algebraic normal form (ANF). The largest number of variables in the product terms with nonzero coefficients is called the algebraic degree of f and is denoted by d_f. The Boolean functions in the form of $g(x) = \omega \cdot x \oplus c$, i.e., those with an algebraic degree of at most one, are referred to as affine functions, where $c \in GF(2)$, $\omega \cdot x = \omega_0 x_0 \oplus \ldots \oplus \omega_n x_n$ is the inner product of $x = (x_0, \ldots, x_{n-1})$ and $\omega = (\omega_0, \ldots, \omega_{n-1}) \in GF(2)^n$. We denote the set of all n-variable affine functions by A_n. An affine function is called a linear function if its constant term is equal to zero, and a Boolean function is called nonlinear if it is not affine.

The Walsh–Hadamard coefficients of an n-variable Boolean function f are the values of the integer-valued function over $GF(2)^n$ given by

$$W_f(\omega) = \sum_{x \in GF(2)^n} (-1)^{f(x) \oplus x \cdot \omega} \text{ for all } \omega \in GF(2)^n \tag{2}$$

and the multi-set $\{W_f(\omega) \mid \omega \in GF(2)^n\}$, where ω runs through $GF(2)^n$ in lexicographic order, is called the Walsh–Hadamard spectrum of f. The nonlinearity NL_f of f is defined as the minimum Hamming distance from all n-variable affine functions, i.e., $NL_f = \min_{g \in A_n} d(f, g)$. As the distance between f and the affine function g can be computed by using the corresponding Walsh–Hadamard coefficient $W_f(\omega)$, NL_f can be expressed in terms of the maximum value in the absolute Walsh–Hadamard spectrum as follows:

$$NL_f = 2^{n-1} - \frac{1}{2} \max_{\omega \in GF(2)^n} \mid W_f(\omega) \mid . \tag{3}$$

The autocorrelation properties of a Boolean function are cryptographically important [8,18]. Let f be an n-variable Boolean function and $d \in GF(2)^n$. The autocorrelation value of f with respect to d is given by $r_f(d) = \sum_{x \in GF(2)^n} (-1)^{f(x) \oplus f(x \oplus d)}$. The maximum absolute autocorrelation value, excluding $r_f(0, \ldots, 0)$, is known as the absolute indicator of f and denoted as

$$\Delta_f = \max_{d \in GF(2)^n, d \neq (0, \ldots, 0)} \mid r_f(d) \mid, \tag{4}$$

which should be small to provide good diffusion properties.

3. Genetic Algorithm

The genetic algorithm (GA) [19], which is based on Darwin's survival of the fittest principle, is the most widely applied evolutionary algorithm in optimization and search problems. The GA is based on the evolutionary process known as natural selection in which any parent, i.e., a pair of individuals, selected from a population produces an offspring. Mutations can occur in this process; however, after mutation, only the fittest individuals survive for the next generation. The general structure of the genetic algorithm used in our study is shown in Figure 1. The initial population in Figure 1 is composed of randomly generated balanced Boolean functions at the closest distance to a fixed bent function. The selection operation (using the elitism approach and the k-tournament mechanism) is applied to the initial population to create the parent population from which the offspring are generated using our crossover strategy given by Algorithm 1. In the elitism approach, the fittest individuals in a population, i.e., the balanced Boolean functions with the best fitness values, survive. In contrast, in the k-tournament method, the fittest one among randomly selected k individuals survives. For a Boolean function f, we use the cost function given in [10,20] as our fitness function here:

$$fitness_f = \sum_{\omega \in GF(2)^n} \left(W_f(\omega)^2 - 2^n \right)^2, \tag{5}$$

which can be considered as a measure proportional to the sum of the squared spectrum deviations from that of a bent function.

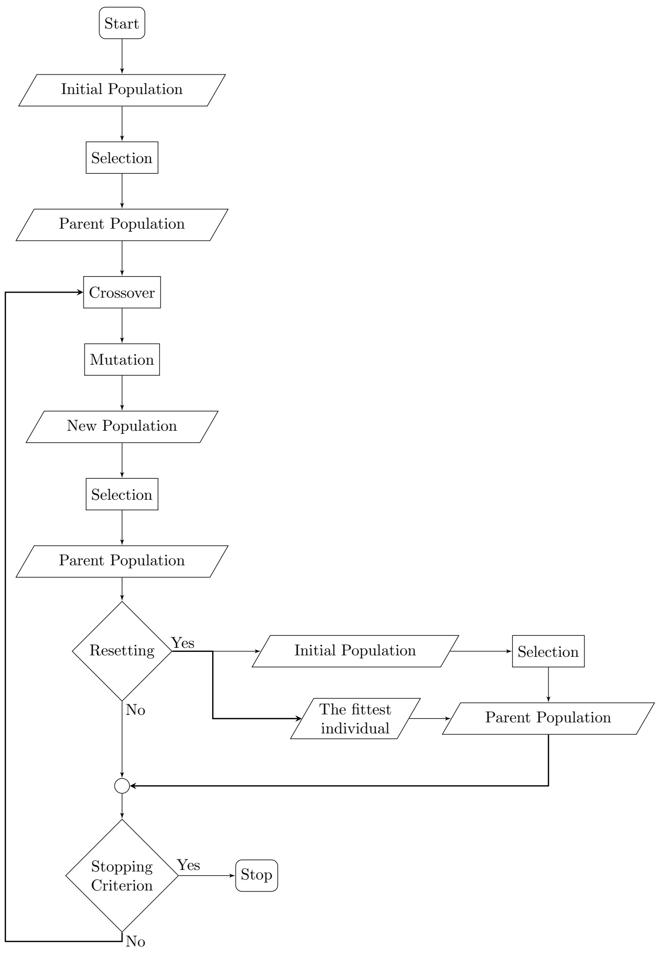


Figure 1. Genetic algorithm structure.

Though any bent function can be used, we randomly select a Maiorana–McFarland (MM)-type bent function [21] from which the balanced Boolean functions are generated to form the initial population whose size is set to 1000 in our search. In our experiments, any bent function that we generate has a Hamming weight of $2^{n-1} - 2^{\frac{n}{2}-1}$, i.e., the number of zeros is $2^{\frac{n}{2}}$ more than the number of ones in its truth table. Next, we compute the fitness value of each individual within the initial population and then apply the elitism approach by selecting 100 of those with the best fitness values. Subsequently, out of the selected individuals, 40 are chosen as the parent population using the k-tournament method, where we take $k = 3$.

After generating the parent population, the algorithm enters the generation loop, which begins with a crossover operation. There are various crossover methods, such as one-point, two-point, uniform, and partial fit, which are used to obtain the offspring from the parent population. Here, we utilize the uniform crossover as given by Algorithm 1 in which the offspring produced from the mating of any two individuals (belonging to the parent population) that are at a distance of $2^{\frac{n}{2}-1}$ from an n-variable bent function keeps the same distance to that bent function. In Algorithm 1, the functions f, g, p_1, and p_2 represent the bent function, the offspring, the first individual, and the second individual, respectively. The algorithm starts by checking each pair of the bits (belonging to the same position) of p_1 and p_2 and assigns their value to the offspring if both are equal. Then, as given by the next two steps, any position at which the bent function f is zero and the first individual p_1 (the second individual p_2) is one is assigned to I_1 (resp., I_2). The crossover operation is completed by assigning 0 to the randomly selected half of the positions in $J_1 \cup J_2$, and 1 to the other half, where J_1 and J_2 are the sets formed by removing the positions belonging to both I_1 and I_2 from I_1 and I_2, respectively. In the GA that we perform, the offspring is generated by using Algorithm 1 from each possible pair of individuals within the parent population. In other words, we take the crossover probability as one. It is to be noted that the offspring g is balanced and has the distance $2^{\frac{n}{2}-1}$ to the bent function f.

Algorithm 1 Uniform crossover operation

```
Input: $f, p_1, p_2$
Output: $g$
1: $g(x) = p_1(x)$ for all $GF(2)^n$ such that $p_1(x) = p_2(x)$
2: $I_1 = \{x \in GF(2)^n : f(x) = 0 \;\&\; p_1(x) = 1\}$
3: $I_2 = \{x \in GF(2)^n : f(x) = 0 \;\&\; p_2(x) = 1\}$
4: $J_1 = I_1 \backslash (I_1 \cap I_2)$
5: $J_2 = I_2 \backslash (I_1 \cap I_2)$
6: $g(x) = 1$ for all $x \in J_1 \cup J_2$
7: $g(x) = 0$ for a randomly selected half of the positions $x \in J_1 \cup J_2$
```

The mutation is applied to the offspring obtained from the crossover in order to ensure diversity. It is realized by flipping a randomly chosen pair of 0 and 1 in the truth table of an offspring such that the bent function is 0 at the corresponding positions of these two bits, which implies that the distance (to the bent function) remains the same after the mutation operation. Since the search space grows super-exponentially as the value n of the variables increases, high mutation probabilities may hinder exploration near the parents. Therefore, in our search, the probability of any offspring being mutated is taken as $2/2^n$, which decreases as n increases. The offspring obtained after the mutation operation are added to the parent population (selected previously from the initial population). Then, a new parent population is formed by applying the selection operation mentioned earlier to the current population.

As the last stage of the loop, to provide additional diversity to the parent population, we apply a resetting step that is a slightly modified version of the one suggested in [22]. An optional resetting step is applied in [22], which keeps the fittest individual and randomly generates the remainder of the parent population if there is no improvement in the best fitness value for a number of generations. Here, we apply the resetting step by considering the fitness values of all individuals in the parent population of each generation. More specifically, if those values are the same for a generation, then one of them is kept, and the other 39 individuals are generated by forming a new population of size 1000 randomly and then applying the selection operation to them. Moreover, in this step, the GA outputs the fittest individual within the parent population of the current generation. The loop continues until the stopping criterion is met, which is the maximum number of generations, and, throughout our experiments, it varies between 100 and 20,000.

3.1. Tuning Phase

The performance of the GA depends upon fine-tuning parameters, such as the size of the initial population, the size of the parent population, and the number of individuals selected from the initial population. We conduct a parameter-tuning phase for 8-, 10-, and 12-variable Boolean functions to determine the values (aforementioned in Section 3) of these parameters by performing a large number of experiments. More precisely, for each number of variables, we experiment with initial population sizes of 500, 1000, 1500, and 2000; parent population sizes of 20, 40, 60, and 80; and selection sizes of 50, 100, 150, and 200. After omitting the combinations for which the parent population size is greater than the selection size, there are 56 combinations left, and the parameter-tuning phase has a stopping criteria of 20,000 iterations for each combination. The set of the best-obtained parameters is used for all Boolean functions considered in this paper.

Since our aim is to optimize the nonlinearity of balanced Boolean functions, the combinations have been evaluated in terms of their corresponding nonlinearity results. Clearly, when a combination is evaluated in terms of the average value obtained, a bad solution can lead to the neglect of good solutions. Therefore, instead of average values, we consider the best values found to compare the combinations. For eight-variable Boolean functions, we observe that every combination of parameters reaches the best-known nonlinearity value of 116. For 10- and 12-variable cases, 11 and 4 combinations obtain the best-known nonlinearities of 492 and 2010, respectively. These combinations are given in Table 1, where a combination is represented by a triplet (initial population size, selection size, parent population size). As can be seen, the best nonlinearity value is achieved for all initial populations with sizes 500, 1000, 1500, and 2000 in the case of 10 variables and for initial populations with sizes 1000 and 2000 in the case of 12 variables. Since the population sizes of 1000 and 2000 provide the best nonlinearity value for both cases, we opted for the size of 1000. Looking at the parent population sizes, it is seen that the best nonlinearity value of 492 is obtained for 10-variable Boolean functions with parent population sizes 20 and 40, while for 12-variable Boolean functions, the best nonlinearity value of 2010 is obtained with a parent population size of 40. Hence, we chose the parent population size of 40, which yields the best nonlinearity value for both the 10- and 12-variable cases. Finally, since the best nonlinearity value is achieved with a selection size of 100 for both 10- and 12-variable scenarios, we chose the selection size to be 100.

Table 1. The combinations yielding the best-known nonlinearities for 10- and 12-variable Boolean functions, where a combination is represented by the triplet (initial population size, selection size, parent population size).

# of Variables	Combinations
10	(500,100,20) (500,100,60) (1000,50,20) (1000,100,40) (1000,200,40) (1500,100,20) (1500,150,80) (1500,200,40) (2000,50,20) (2000,100,20) (2000,100,40)
12	(1000,100,40) (1000,150,40) (2000,100,60) (2000,200,40)

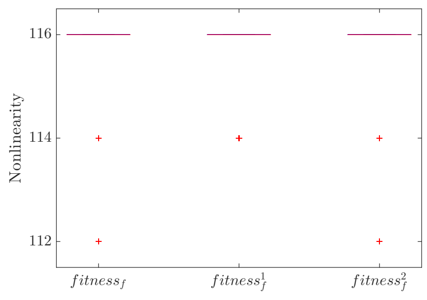
3.2. Performance Evaluation

The choice of the fitness function, along with the tuning of the parameters, plays a crucial role in the performance of the GA and the quality of solutions. In literature, Clark's cost/fitness function [23] is one of the most widely used ones for evolving Boolean functions; however, it has parameters that need to be tuned experimentally. As our fitness function ($fitness_f$ given by Equation (5)) does not require any experimental parameter, here, we evaluate the performance of our algorithm using two different fitness functions (given below) that do not involve experimental parameters.

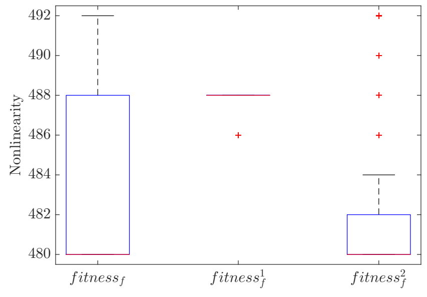
$$fitness_f^1 = NL_f \tag{6}$$

$$fitness_f^2 = \frac{1}{2^{4n}} \sum_{\omega \in GF(2)^n} W_f(\omega)^4 \tag{7}$$

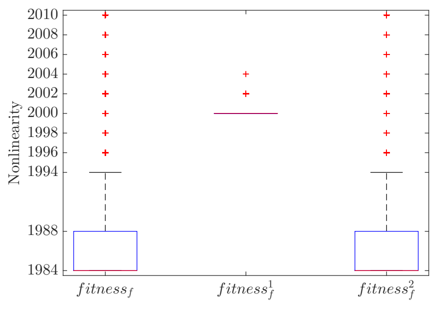
While the fitness function $fitness_f^1$ is simply the nonlinearity of f, the other fitness function $fitness_f^2$ is Gowers second-order norm of f which, by comparing it with Clark's fitness function, has been shown in [14] to be efficient for reaching the optimal solutions. In Figure 2, for the 8-, 10-, and 12-variable cases, the comparison results of the mentioned three fitness functions are presented in boxplot form. The boxplots shown in Figure 2a–c (Figure 2d–f) represent the nonlinearity (resp. the absolute indicator) distribution of 400,000 balanced Boolean functions generated by running the GA 20 times with 20,000 iterations each.



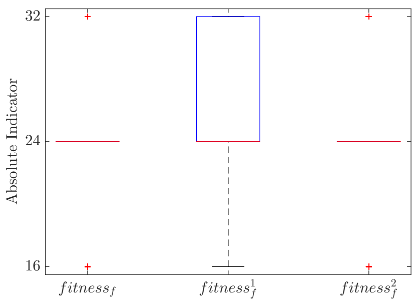
(**a**)



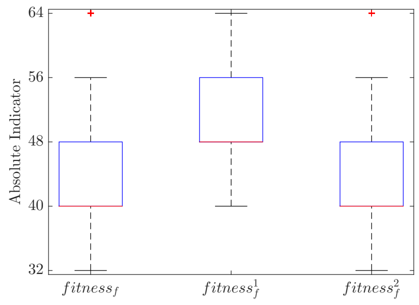
(**b**)



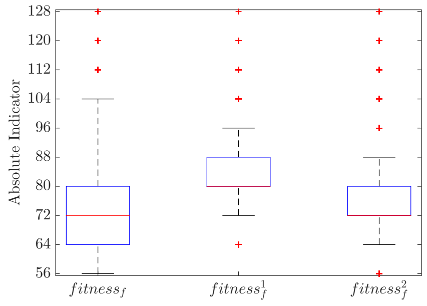
(**c**)



(**d**)



(**e**)



(**f**)

Figure 2. Boxplot distributions produced by different cost functions for 8-, 10, and 12-variable Boolean functions, where (**a**–**c**) [(**d**–**f**)] show nonlinearity (resp. absolute indicator) distributions, respectively.

From Figure 2a, it is seen that, when $fitness_f$ or $fitness_f^2$ is used, the boxplot distributions are the same, and the GA obtains nonlinearity values of 112, 114, and 116. Additionally, we observe that the GA with $fitness_f^1$ does not produce the lowest nonlinearity (112) among them. Figure 2b shows that the GA with $fitness_f$ or $fitness_f^2$ reaches the nonlinearity value of 492 (with the former one producing this nonlinearity more frequently), and the best-achieved nonlinearity by the GA with $fitness_f^1$ is 488. In the case of 12 variables, as can be seen from Figure 2c, the boxplot distributions indicating that the highest achieved nonlinearity value is 2010 are the same for $fitness_f$ and $fitness_f^2$; however, the best-achieved nonlinearity value is 2004 for $fitness_f^1$.

Considering the boxplot distributions in Figure 2d–f, we see that both $fitness_f$ and $fitness_f^2$ provide the same boxplots with lowest absolute indicator values of 16 and 32 for the cases of 8- and 10-variable Boolean functions, respectively. In addition, we observe that $fitness_f$ and $fitness_f^2$ outperforms $fitness_f^1$ for the 10- and 12-variable cases by producing better absolute indicator values of 32 and 56, respectively. However, for the 8-variable case, the GA with $fitness_f^1$ produces an absolute indicator value of 16 more frequently compared to the GA with $fitness_f$ or $fitness_f^2$, and, similarly, for the 12-variable case, the GA with $fitness_f$ produces an absolute indicator value of 56 more frequently compared to the GA with $fitness_f^2$.

From the boxplot distributions in Figure 2, one can infer that though the GA with $fitness_f$ provides more frequent generations of the nonlinearity value of 492 for the 10-variable case and the absolute indicator value of 56 for the 12-variable case, it exhibits a similar performance to the GA with $fitness_f^2$ when we consider only the best-achieved results. Further, both of these generate better nonlinearity and absolute indicator values than the GA with $fitness_f^1$ for the 10- and 12-variable cases.

The averages and standard deviations of the distributions of nonlinearity and absolute indicator values, which are represented with boxplots in Figure 2, are given in Table 2. As can be seen, for the cases of 10 and 12 variables, the average nonlinearity (absolute indicator) value obtained for $fitness_f$ is greater (resp. less) than that obtained for $fitness_f^2$. Additionally, compared to $fitness_f^2$, $fitness_f$ gives larger standard deviations for the nonlinearity and absolute indicator distributions. Therefore, for these cases, the statistical parameters provided by $fitness_f$ seem to be better than those provided by $fitness_f^2$. In the eight-variable case, for both nonlinearity and absolute indicator values, though the standard deviations corresponding to $fitness_f$ are larger than those corresponding to $fitness_f^2$, $fitness_f$ gives worse average values than $fitness_f^2$. When we consider the average nonlinearity (absolute indicator) values found for $fitness_f^1$, we observe that they are better (resp. worse) than the other averages of the nonlinearity (resp. absolute indicator) distributions. It should be noticed that since $fitness_f^1$ gives very small standard deviations (with the exception of the distribution of the absolute indicator values in the eight-variable case), the nonlinearity and absolute indicator values reached by $fitness_f$ or $fitness_f^2$ can be seen as less likely to be achieved with $fitness_f^1$.

Table 2. The averages (μ) and standard deviations (σ) of the boxplot distributions (given in in Figure 2) of (**a**) nonlinearity values and (**b**) absolute indicator values for n-variable Boolean functions, where $n = 8, 10$, and 12.

(**a**)

		$n = 8$	$n = 10$	$n = 12$
$fitness_f$	μ	115.2670	483.4514	1987.4868
	σ	1.4142	4.8963	7.0278
$fitness_f^1$	μ	115.9977	487.8096	2000.1845
	σ	0.0684	0.5870	0.5789
$fitness_f^2$	μ	115.3356	482.5152	1986.9790
	σ	1.3560	4.3490	6.1503

Table 2. *Cont.*

(b)				
		$n = 8$	$n = 10$	$n = 12$
$fitness_f$	μ	24.6387	43.6811	76.0864
	σ	2.1718	5.2957	12.6712
$fitness_f^1$	μ	27.8896	49.7244	81.5849
	σ	3.9986	4.3848	6.3888
$fitness_f^2$	μ	24.6021	43.6920	76.1885
	σ	2.1139	4.9239	11.1301

4. Search Effort

The GA explained in the section above is performed by setting the experimental parameter values as summarized in Table 3. The computer system that we utilize has the following specifications: Intel® Core™ i5-9400F CPU (2.9 GHz, 9 MB cache, 6 cores), 16 GB RAM, and Windows 10 Pro 64-bit operating system. The GA is implemented in C programming language, and the typical consumed times for 100 generations are given in Table 4 for all the values of n of the variables we consider. We used 20 computers (each with the given specification) that work with all of the cores for about six months to obtain our results. All truth tables of our best-achived results can be found in [24]. Among them, we have provided the truth tables of the Boolean functions with numbers of the variables 8, 10, and 12 in Appendix A.

Table 3. Experimental parameter values.

Parameter	Value
Initial population size	1000
Number of individuals left after the elitisim approach	100
Number of individuals participating in the tournament (k)	3
Parent population size	40
Crossover probability	1
Mutation probability (n is the number of input variables)	$2/2^n$
Maximum number of generations	100 to 20,000

Table 4. Consumed times for 100 generations and # of balanced functions.

n	Consumed Time	$N_{\text{balanced}}(n)$
8	3 s	2^{41}
10	5 s	2^{100}
12	10 s	2^{234}
14	32 s	2^{536}
16	1 min 52 s	2^{1204}
18	7 min 33 s	2^{2668}
20	34 min 56 s	2^{5853}
22	2 h 37 min 52 s	$2^{12,735}$
24	14 h 18 min 10 s	$2^{27,524}$
26	1 d 18 h 50 min 36 s	$2^{59,150}$

Clearly, the number $N_{\text{balanced}}(n)$ of the balanced n-variable Boolean functions generated by flipping $2^{\frac{n}{2}-1}$ bits of a single bent function is $\binom{2^{n-1}+2^{\frac{n}{2}-1}}{2^{\frac{n}{2}-1}}$, which is computed as given in Table 4. Let $N_{\text{bent}}(n)$ be the number of n-variable bent functions. We know from [25] that $N_{\text{bent}}(8) = 99{,}270{,}589{,}265{,}934{,}370{,}305{,}785{,}861{,}242{,}880$ ($\approx 2^{106}$), and it is unknown for $n > 8$. Then, the search space is of the size $N_{\text{bent}}(n) \times \binom{2^{n-1}+2^{\frac{n}{2}-1}}{2^{\frac{n}{2}-1}}$, which gives $\approx 2^{147}$ in the case of $n = 8$.

To better understand the efficiency of our search strategy, let us consider in particular the case of eight-variable Boolean functions. Using an exhaustive search, we enumerate all balanced Boolean functions with the best possible nonlinearity obtained from a randomly selected MM-type bent function f whose truth table is given as follows using hexadecimal notation:

```
3C3C5A5A33CC669900000FF066665AA5 0F0F3CC369693333699655555AA00FF
```

As a result, we find that the number of those balanced functions is 6,346,334,872, and their classification is given in Table 5, where the cryptographic properties are denoted by the triplet (nonlinearity; absolute indicator; algebraic degree). As can be seen from Table 5, the best possible profile is $(116; 16; 7)$, and there are 34,336 balanced Boolean functions with that profile. Noting that $\lim_{z\to\infty}(1-1/z)^z = 1/e$, one may obtain such a function in approximately $\binom{136}{8}/34{,}336$ ($> 2^{26}$) trials in a random search with a probability of $1 - 1/e$ ($> 1/2$). We then perform the GA by using the bent function f (to form the initial population) and setting the maximum number of generations to 20,000. In 50 runs, we observe that 4 of the 10^6 (=20,000×50 $< 2^{20}$) generation outputs have the best profile, which shows the superiority of our method over a random search.

Table 5. Classification of balanced Boolean functions with nonlinearity 116 in the neighborhood of the bent function f.

$(NL_f; \Delta_f; d_f)$	#
$(116, 16, 6)$	22,720
$(116; 16; 7)$	34,336
$(116; 24; 6)$	1,006,138,800
$(116; 24; 7)$	1,906,929,536
$(116; 32; 6)$	2,090,955,784
$(116; 32; 7)$	1,342,253,696

We now construct, as an example, a balanced eight-variable Boolean function with nonlinearity 116, absolute indicator 16, and algebraic degree 7. Let $\ell_\omega(x) = \omega \cdot x$, for all $x \in GF(2)^4$, be a linear function determined by $\omega \in GF(2)^4$, and let $f = [\ell_{\omega^{(1)}}\ \ell_{\omega^{(2)}}\ \cdots\ \ell_{\omega^{(16)}}]$ be an MM-type bent function that can be seen as a concatenation of the distinct linear functions determined by $\omega^{(1)}, \omega^{(2)}, \ldots, \omega^{(16)}$. Consider the bent function f formed by the linear functions $\ell_{\omega^{(i)}}$ corresponding to the $\omega^{(i)}$ vectors, $i = 1, 2, \ldots, 16$, given in Table 6. Flipping the eight bits $\ell_{\omega^{(5)}}(1,0,1,0)$, $\ell_{\omega^{(5)}}(1,1,1,1)$, $\ell_{\omega^{(6)}}(0,0,0,0)$, $\ell_{\omega^{(6)}}(1,1,0,0)$, $\ell_{\omega^{(16)}}(0,1,0,0)$, $\ell_{\omega^{(16)}}(0,1,0,1)$, $\ell_{\omega^{(16)}}(0,1,1,0)$, and $\ell_{\omega^{(16)}}(0,1,1,1)$ of the bent function f yields the balanced eight-variable Boolean function with the profile (116; 16; 7).

Table 6. The $\omega^{(i)}$ vectors to generate the bent function f.

$\omega^{(1)} = (0,1,1,0)$	$\omega^{(5)} = (0,0,0,0)$	$\omega^{(9)} = (0,1,0,0)$	$\omega^{(13)} = (1,1,1,1)$
$\omega^{(2)} = (0,1,0,1)$	$\omega^{(6)} = (1,1,0,0)$	$\omega^{(10)} = (1,1,1,0)$	$\omega^{(14)} = (0,0,0,1)$
$\omega^{(3)} = (1,0,1,0)$	$\omega^{(7)} = (0,0,1,1)$	$\omega^{(11)} = (0,1,1,1)$	$\omega^{(15)} = (1,0,0,1)$
$\omega^{(4)} = (1,0,1,1)$	$\omega^{(8)} = (1,1,0,1)$	$\omega^{(12)} = (0,0,1,0)$	$\omega^{(16)} = (1,0,0,0)$

5. Conclusions

In Table 7, denoting the cryptographic properties by the triplet (nonlinearity; absolute indicator; algebraic degree), we summarize and compare our best-achieved results with the present best results in [9–12] for even $n \in [8, 26]$. As can be seen, by applying the GA to the balanced n-variable Boolean functions produced by flipping the minimum number ($2^{\frac{n}{2}-1}$) of bits of a bent function, we can achieve better absolute indicators (less than $2^{\frac{n}{2}}$), which were unknown before, for all $n = 12, 14, \ldots, 26$. In addition, we find that the corresponding nonlinearity values exceed $2^{n-1} - 2^{\frac{n}{2}}$ for $n = 16, 18, \ldots, 26$. The results also demonstrate examples of balanced Boolean functions with the best-known nonlinearity that are at the closest distance to a bent function for $n = 8, 10, \ldots, 16$. To the best of our knowledge, there is no nontrivial lower bound on the absolute indicator values, and we only know that the absolute indicator of a balanced Boolean function can be as low as eight. Hence, though the results given in Table 7 improve the absolute indicator values in [12], it is quite possible that one can achieve further improvements with sufficient computational power, especially for a large number of variables. On the other hand, as a long-standing open problem, it is still unknown whether there exists a balanced Boolean function with an even number n of variables that achieves nonlinearity of greater than $2^{n-1} - 2^{\frac{n}{2}} + \mathrm{nlb}(\frac{n}{2})$ for $n \geq 8$, and, hence, it seems elusive to achieve this nonlinearity. For instance, it is unknown whether there exists a balanced eight-variable Boolean function with nonlinearity of 118.

Table 7. Comparison of our best-achieved profiles with those in the literature, where the profile of f is defined as the triplet $(NL_f; \Delta_f; d_f)$.

Reference	$n = 8$	$n = 10$	$n = 12$	$n = 14$
[9]	$(116; 16; 7)$	$(488; 40; 9)$	$(2002; 64; 11)$	$(8102; 104; 13)$ $(8104; 112; 13)$
[10]	–	$(488; 24; 9)$	–	–
[12]	–	–	$(1996; 56; 11)$	$(8106; 96; 13)$
[13]	$(116; 24; 7)$	$(488; 40; 9)$	$(2002; 72; 11)$	–
[14]	$(116; 24; -)$	$(488; 40; -)$	$(2002; 72; -)$	–
Our results	$(116; 16; 7)$	$(480; 24; 6)$ $(492; 40; 9)$	$(1984; 40; 7)$ $(1984; 48; 11)$ $(2010; 80; 11)$	$(8064; 88; 13)$ $(8064; 88; 13)$ $(8120; 144; 12)$

Reference	$n = 16$	$n = 18$	$n = 20$
[11]	-	(130,664; 480; 17)	-
[12]	(32,604; 160; 15)	(130,762; 312; 17)	(523,688; 600; 19)
Our results	(32,516; 152; 15) (32,628; 288; 14)	(130,752; 256; 17)	(523,676; 400; 19)

Reference	$n = 22$	$n = 24$	$n = 26$
[11]	(2,095,484; 1880; 21)	-	(33,547,436; 7856; 25)
[12]	(2,096,020; 1224; 21)	(8,386,392; 2360; 23)	(33,550,064; 4584; 25)
Our results	(2,095,958; 592; 21)	(8,386,318; 904; 23)	(33,549,996; 1360; 25)

Here, we mainly consider two of the most important cryptographic properties of balanced Boolean functions, which are nonlinearity and the absolute indicator. It should be noted that in order for a Boolean function to be used in a cryptosystem, one should also consider the other cryptographic properties, such as resiliency, algebraic immunity, and fast algebraic immunity. We think that it is possible to exploit our search strategy to optimize these properties as well by increasing the distance to the bent functions and designing the fitness function accordingly. However, since there is a trade-off among those cryptographic properties and improving one property may lead to the deterioration of another property, it seems quite probable to deterioriate our best-found cryptographic properties in Table 7 while optimizing the other cryptographic properties. For instance, when compared to [26], where two classes of one-resilient Boolean functions with the best-known absolute indicator values are constructed, one can see that the absolute indicator values presented in Table 7 are better.

Author Contributions: Conceptualization, H.K. and S.K.; methodology, H.K. and S.K.; software, E.Ö. and S.K.; validation, E.Ö., S.K. and H.K.; formal analysis, H.K. and S.K.; investigation, E.Ö., S.K. and H.K.; resources, E.Ö. and S.K.; data curation, E.Ö. and S.K.; writing—original draft preparation, E.Ö., S.K. and H.K.; writing—review and editing, E.Ö. and S.K.; visualization, E.Ö.; supervision, H.K. and S.K.; project administration, H.K. and S.K.; funding acquisition, E.Ö., S.K. and H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The programming code supporting the results presented in this manuscript is available in the GitHub repository, https://github.com/Selcuk-kripto/imp_AC (accessed on 8 August 2023).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

For 8, 10, and 12 variables, we provide the truth tables of the balanced Boolean functions with the cryptographic properties in Table 7. All truth tables for our results given in this paper are available at [24].

$(116; 16; 7)$:

```
3C3C5A5A33CC669900218FF866665AA50F0F3CC3696933336996555555AA0FFF
```

$(480; 24; 6)$:

```
3C3C3C3C7969969655555555AB5A55A333333330F0F0F1F5A5A5A5A33CCDC33
4F0FF0F0669966993373CCCC00FF00FF0FF0F04F55AA55AA3CC37CC30FF00FF0
696969696E6666660000FFFF699E69965AA55AA555AAAA5D00FFFF0069969E69
020000003C3CC3C35557AAAA666699995A5AA5A73CC3C33C66999B6633CC33CC
```

$(492; 40; 9)$:

```
0000FFFF55AA55AA00FF00FF0F0F0F0F3333333355555555AA55AA5669966 99
00FFFF0006008000 3333CCCC0FF00FF055AAAA5566E6F9995555AAAA66999966
3CC33CC32FF4F99F696969693CC3C33C3C3C3C3C79976BD65A5AA5A533CCCC33
696996960F0FF0F06996966933CC33CC5A5A5A5A3C3CC3C36666666665AA5A55A
```

$(1984; 40; 7)$:

```
6666666666766666555555555555755555AAAA55AA5555AA6969696969696969
5555AAAA5555AAAA0F0FF0F0F0F00F0F55AAAA5555AABA550FF00FF0F02FF00F
66679999999966663CCCE3333CCCC333333CCCC3333CCCC00FF00FFFF00FF00
00FFFF00FF0000FF33CC33CC33CC33CC3333CDCCCCCC3333000200000000000000
0FF0F00F0FF0F08F3C3CC3C37C3CC3C35AA55AA5A55AA55A5A5AA5A5A5A55A5A
69699696969669693C3C3C3C3C3C3C3C69969669E996966900FFFF0000FFFF40
6699996E99666699 6D966996699669960000000FFFFFFFF0F0F0F0FF0F0F0F0
3CC3C33C3CC3C33C69966996966996695D555555AAAAAAAA5555AAAEAAAA5555
3CC33CC33CC33CD35A5A5A5A7A5A5A5A3CC33CC3C33CC33C6666666699999999
69969669966969963333333CCCCCCCC0FF00FF01FF00FF069699696696996B6
5AA5A55BA55A5AA56B69696996969696 3C3C3C3CC3C3C3C30000FFFF0000FFFF
```

```
5A5AA5A55A5AA5A5669966999966996601 00FFFFFFFF00006699669B66996699
0F0F0F0F8F0F0F33CCCC33CC3373CC3C3CC3C3C3C33C3C55AA55AA55AA55AA
6666999966669999 0F0FF0F00F0FF0F000FF00FF00FF80FF3333333333733333
5AAD5AA55AA55AA566999D666699996 60FF0F00FF00F0FF03CC3C33CC33C3CC3
33CC33CCCC33CC335A5A5A5AA5A5A5A55AA5AD5A5AA5A55A55AE55AAAA55AA55
```

$(1984; 48; 11)$:

```
0F0FF0F0F0F02F0F6969696969696969 33CCCC33CC3337CC0FF0F00FF00F0FF0
3CC3C33CCB3C3CC30F0FF0F00F0FF0F055555555BAAAAAAA0FF00FF00FF00FF0
666699B99999666600FF00FF00FF00FF5555AAAE5555AAAA3C3C3C3CC3C3C3C3
0008FFFF0000FFFF3333CCCCCCCC33335AB5A55A5AA5A55A00FF00FFFF00FF00
01FFFF00FF0000FF0FF0F00F0FF0F00FDAA55AA5A55AA55A6996966969969669
6699D96699666699 33CC33CC33CC33CC5AA5A75AA55A5AA55A5A5A5A5A5A5A5A
66666666666766663CC3C33C3CC3C33C3CC33CC3C3BCC33C3C3CC3C33C3CC3C3
666699996666 99D90FF00FF0F00FF00F5555AAAAAAAA555700FFFF0000FFFF00
55AAAA55AA5555AA7C3C3C3C3C3C3C3C66996699669966996B69969696966969
5A5A5A5AA5A5A5A55AA55BA55AA55AA533CC33CCCC33CC3333333333333333 33
0F0F0F0FF0F0F0F06666666699D999995555555555555555B5AA5A5A5A75A5A
69966996966996696969969669699697 5A5AA5A55A5AA5A566999966669999E6
6996966996696996000000000000000080033CCCC3333CCCC330000FFFFFFFF1000
55AA55AAAA55AA5566996699B966996633333333CCCCCCCC699669966D966996
00000000FFFFFFFF55AAAA5D55AAAA553333CCCC3333CCCC3C3CC3D3C3C33C3C
69696969969696960F2F0F0F0F0F0F0F3CC33CC33CC33CC355AE55AA55AA55AA
```

$(2010; 80; 11)$:

```
0F0FF0F00F0FF0F000000000FFFFFFFF33333333CCCCCCCC5AA55AA5A55AA55A
33F7CCFE3733FDCC3333333333333333FCCC7BEC33FBCC3C3CC3C33C3CC3C3
0FF0F00FF00F0FF00F0F0FF0F0F0F0696996966969969669969669966 96996
33CCCC3333CCCC333C3C3C3CC3C3C3C30F0FF0F0F0F00F0F00FFFF0000FFFF00
55555555AAAAAAAA5AA55AA55AA55AA55A5A5A5A5A5A5A5A6996699696699669
0FF0F00F0FF0F00F3C3C3C3C3C3C3C3C5555AAAAAAAA555500FF00FF00FF00FF
6666666699999999 55AAAA55AA5555AA5A5AA5A5A5A55A5A3CC3C33CC33C3CC3
6996966969969669666666666666666655AA55AAAA55AA555A5A5A5AA5A5A5A5
5AA5A55A5AA5A55A0FF00FF0F00FF00F00FFFF00FF0000FF3C3CC3C3C3C33C3C
69969696966969690000FFFF0000FFFF69696969969696965A5AA5A55A5AA5A5
33CC33CC33CC33CC66669999666699996699996699666699 3CC3C33C3CC3C33C
66999966669999665555AAAA5555AAAA3333CCCCCCCC333333CC33CCCC33CC33
5AA5A55AA55A5AA5669966999966996600 00FFFFFFFF00005555555555555555
8CFF21FFFF00FF2355AAAA5555AAAA552000080000310080666699999999 6666
69696969696969690F0F0F0F0F0F0F0F3CC33CC33CC33CC355AA55AA55AA55AA
3CC33CC3C33CC33C66996699669966990FF00FF00FF00FF06996699669966996
```

References

1. Ding, C.; Xiao, G.; Shan, W. *The Stability Theory of Stream Ciphers*; Springer: Berlin, Heidelberg, Germany, 1991.
2. Matsui, M. Linear cryptanalysis method for DES cipher. In Proceedings of the Eurocrypt'93, Lofthus, Norway, 23–27 May 1993; Springer: Berlin/Heidelberg, Germany, 1994; LNCS, Volume 765, pp. 386–397.
3. Aumasson, J.-P.; Fischer, S.; Khazaei, S.; Meier, W.; Rechberger, C. New features of Latin dances: Analysis of Salsa, ChaCha, and Rumba. In Proceedings of the Fast Software Encryption, Lausanne, Switzerland, 10–13 February 2008; Springer: Berlin/Heidelberg, Germany, 2008; LNCS, Volume 5086, pp. 470–488.
4. Banik, S.; Maitra, S.; Sarkar, S. A differential fault attack on the Grain family of stream ciphers. In Proceedings of the Cryptographic Hardware and Embedded Systems, Leuven, Belgium, 9–12 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; LNCS, Volume 7428, pp. 122–139.
5. Dillon, J.F. Elementary Hadamard Difference Sets. Ph.D. Dissertation, University of Maryland, College Park, MD, USA, 1974.
6. Rothaus, O.S. On "bent" functions. *J. Combin. Theory Ser. A* **1976**, *20*, 300–305. [CrossRef]
7. Dobbertin, H. Construction of bent functions and balanced Boolean functions with high nonlinearity. In Proceedings of the Fast Software Encryption, Leuven, Belgium, 14–16 December 1994; Springer: Berlin/Heidelberg, Germany, 1995; LNCS, Volume 1008, pp. 61–74.
8. Zhang, X.M.; Zheng, Y. GAC—The criterion for global avalanche characteristics of cryptographic functions. *J. Univers. Comp. Sci.* **1995**, *1*, 316–333.
9. Burnett, L.; Millan, W.; Dawson, E.; Clark, A. Simpler methods for generating better Boolean functions with good cryptographic properties. *Aust. J. Comb.* **2004**, *29*, 231–248.
10. Kavut, S.; Maitra, S.; Yücel, M.D. Search for Boolean Functions With Excellent Profiles in the Rotation Symmetric Class. *IEEE Trans. Inf. Theory* **2007**, *53*, 1743–1751. [CrossRef]

11. Tang, D.; Maitra, S. Construction of n-variable ($n \equiv 2 \mod 4$) balanced Boolean functions with maximum absolute value in autocorrelation spectra $< 2^{\frac{n}{2}}$. *IEEE Trans. Inf. Theory* **2018**, *64*, 393–402.
12. Kavut, S.; Maitra, S.; Tang, D. Construction and search of balanced Boolean functions on even number of variables towards excellent autocorrelation profile. *Des. Codes Cryptogr.* **2019**, *87*, 261–276. [CrossRef]
13. Izbenko, Y.; Kovtun, V.; Kuznetsov, A. The design of Boolean functions by modified hill climbing method. In Proceedings of the Sixth International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 27–29 April 2009; IEEE: Manhattan, NY, USA, 2009; pp. 356–361.
14. Behera, P.K.; Gangopadhyay, S. An improved hybrid genetic algorithm to construct balanced Boolean function with optimal cryptographic properties. *Evol. Intel.* **2022**, *15*, 639–653. [CrossRef]
15. Djurasevic, M.; Jakobovic, D.; Mariot, L.; Picek, S. A survey of metaheuristic algorithms for the design of cryptographic Boolean functions. *Cryptogr. Commun.* **2023**. [CrossRef]
16. Carlet, C. *Boolean Functions for Cryptography and Error Correcting Codes*; Cambridge University Press: Cambridge, UK, 2021.
17. Cusick, T.W.; Stănică, P. *Cryptographic Boolean Functions and Applications*; Elsevier: Amsterdam, The Netherlands; Academic Press: Cambridge, MA, USA, 2009.
18. Preneel, B.; Leekwijck, W.V.; Linden, L.V.; Govaerts, R.; Vandewalle, J. Propagation characteristics of Boolean functions. In Proceedings of the Eurocrypt'90, Aarhus, Denmark, 21–24 May 1990; Springer: Berlin/Heidelberg, Germany, 1991; LNCS, Volume 473, pp. 161–173.
19. Holland, J.H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
20. Kavut, S.; Yücel, M.D. Improved cost function in the design of Boolean functions satisfying multiple criteria. In Proceedings of the Indocrypt'03, New Delhi, India, 8–10 December 2003; Springer: Berlin/Heidelberg, Germany, 2003; LNCS, Volume 2904, pp. 121–134.
21. McFarland, R.L. A family of noncyclic difference sets. *J. Combin. Theory Ser. A* **1973**, *15*, 1–10. [CrossRef]
22. Millan, W.; Clark, A.; Dawson, E. Heuristic design of cryptographically strong balanced Boolean functions. In Proceedings of the Eurocrypt'98, Espoo, Finland, 31 May–4 June 1998; Springer: Berlin/Heidelberg, Germany, 1998; LNCS, Volume 1403, pp. 489–499.
23. Clark, J.A.; Jacob, J.L.; Stepney, S.; Maitra, S.; Millan, W. Evolving Boolean functions satisfying multiple criteria. In Proceedings of the Indocrypt 2002, Hyderabad, India, 16–18 December 2002; Springer: Berlin/Heidelberg, Germany, 2002; LNCS, Volume 2551, pp. 246–259.
24. Balanced Boolean Functions on Even Number of Variables Achieving Absolute Indicator Less Than $2^{n/2}$. Available online: https://github.com/Selcuk-kripto/imp_AC (accessed on 14 July 2023).
25. Langevin, P.; Leander, G. Counting all bent functions in dimension eight 99270589265934370305785861242880. *Des. Codes Cryptogr.* **2011**, *59*, 193–205. [CrossRef]
26. Li, Y.; Kan, H.; Peng, J.; Tan, C.H. SAO 1-resilient functions with lower absolute indicator in even variables. *IEEE Access* **2020**, *8*, 222377–222384. [CrossRef]