Evolving DPA-Resistant Boolean Functions

Stjepan Picek[1,2], Lejla Batina[1], and Domagoj Jakobovic[2]

[1] Radboud University Nijmegen, Institute for Computing and Information Sciences
Postbus 9010, 6500 GL Nijmegen, The Netherlands
[2] Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb, Croatia

Abstract. Boolean functions are important primitives in cryptography. Accordingly, there exist numerous works on the methods of constructions of Boolean functions. However, the property specifying the resistance of Boolean functions against Differential Power Analysis (DPA) attacks was until now scarcely investigated and only for S-boxes. Here, we evolve Boolean functions that have higher resistance to DPA attacks than others published before by using two well-known evolutionary computation methods where genetic programming shows best performance.

Keywords: Genetic Algorithms, Genetic Programming, Boolean Functions, Cryptographic Properties, Transparency Order.

1 Introduction

In the area of private-key cryptography, one common division is to block ciphers and stream ciphers [7]. In both areas the nonlinear elements often represent the key part of the algorithm. The usual nonlinear elements are Boolean functions and S-boxes (vectorial Boolean function, i.e. a generalization of a Boolean function). In accordance with their importance, over the years there have been numerous works on constructing those nonlinear elements. Methods used in those works can be divided into algebraic constructions [5], random search [3] and heuristic methods [11] where each method has its drawbacks and benefits. Main advantages of heuristic methods are in a relatively easy addition of cryptographic properties to the evaluation functions and in results comparable with algebraic constructions. From this point on, we will concentrate only on Boolean functions and conduct the research accordingly. Here we note that if a Boolean function has n inputs, then there are 2^{2^n} Boolean functions possible. For n larger than 4 it is impractical to do an exhaustive search. Therefore, generating Boolean functions with desirable cryptographic properties is a hard problem.

1.1 Related Work

Here we mention only a small number of papers where the goal was to find optimal Boolean functions considering certain cryptographic properties.

Simulated annealing is used by McLaughlin and Clark to evolve Boolean functions that have several cryptographic properties with optimal values [10].

Aguirre et al. use evolutionary multiobjective approach to evolve Boolean functions with high nonlinearity [1]. Picek et al. use genetic programming and genetic algorithms to find Boolean functions that have good cryptographic properties [13]. Genetic algorithms are also used by Picek et al. to evolve S-boxes with good DPA (Differential Power Analysis) resistance for the AES case [12].

1.2 Our Contribution

To our best knowledge we are the first to consider a property that concerns the resistance of a Boolean function to side-channel attacks - *transparency order* [14]. Up to now, transparency order has been only vaguely investigated in the case of S-boxes but never for Boolean functions [9]. Furthermore, we give a comparison between some of the currently designed Boolean functions by means of evolutionary computation techniques and a Boolean function created by an algebraic method, used in an actual cryptographic algorithm. In our research we use genetic programming (GP) and genetic algorithms (GAs) to evolve Boolean functions with good transparency order values. Since there are no prior attempts to evaluate the resistance of a Boolean function to side-channel attacks, that makes a fair comparison more difficult.

In Section 2 we give some preliminary information on side-channel attacks and cryptographic properties of Boolean functions. Evolution of Boolean functions with GAs and GP when considering the transparency order is described in Section 3. In Section 4 we present experimental results and discussion about them. Finally, conclusion and future research is given in Section 5.

2 Preliminaries

In this section we present background details about cryptographic properties of Boolean functions and introduce basic notions of side-channel analysis.

2.1 Side-Channel Analysis

Small devices on which cryptographic algorithms are implemented, such as smart cards, have become pervasive in our lives and lots of our security and privacy-sensitive data is stored on those constrained platforms. These devices typically provide unintentional output channels, often called side channels releasing some physical leakage that relates to the operations and/or even data being processed. There are several side channels possible when considering unleashed physical information, such as power consumption or electromagnetic emanation, and we refer an interested reader to [8].

2.2 Boolean Functions

A Boolean function is in mathematics usually defined as a mapping from $\{0,1\}^n$ to $\{0,1\}$.

A unique representation of a Boolean function is a truth table (TT) [2]. When the total lexicographical order is assigned, Boolean function f with n inputs has a truth table with 2^n elements, where each element a fulfills $a \in \{0, 1\}$.

A Boolean function is uniquely represented by its Walsh transform which is a real-valued function defined for all $\boldsymbol{\omega} \in \mathbb{F}_2^n$ as [2]

$$W_f\left(\boldsymbol{\omega}\right) = \sum_{\boldsymbol{x} \in \mathbb{F}_2^n} \left(-1\right)^{f(\boldsymbol{x}) \oplus \boldsymbol{x} \cdot \boldsymbol{\omega}}, \tag{1}$$

where $\boldsymbol{x} \cdot \boldsymbol{\omega}$ is the scalar product of vectors $\boldsymbol{x}$ and $\boldsymbol{\omega}$.

Here, $\mathbb{F}_2^n$ is an n-dimensional vector space over Galois field with two elements [2].

The Hamming weight $HW\left(f\right)$ of a Boolean function f is the number of ones in its binary truth table [2].

Boolean function f with n inputs is balanced if its Hamming weight equals 2^{n-1}, i.e. the number of ones equals the number of zeros in the truth table [2].

The nonlinearity NL_f of a Boolean function is its minimum Hamming distance to any affine function and it can be calculated as [2]:

$$NL_f = \frac{1}{2}\left(2^n - max|W_f\left(\boldsymbol{\omega}\right)|\right). \tag{2}$$

In 2005, Prouff introduced a new cryptographic property of S-boxes: **transparency order**, which can be defined for an (n, m)-function as follows [14].

$$T_f = max_{\boldsymbol{\beta} \in \mathbb{F}_2^m}\Big(|m - 2HW(\boldsymbol{\beta})| - \frac{1}{2^{2n} - 2^n} \sum_{\boldsymbol{a} \in \mathbb{F}_2^{n*}} | \sum_{\substack{\boldsymbol{v} \in \mathbb{F}_2^m \\ HW(\boldsymbol{v}) = 1}} (-1)^{\boldsymbol{v} \cdot \boldsymbol{\beta}} W_{D_a f}(\boldsymbol{0}, \boldsymbol{v})|\Big). \tag{3}$$

Here, $W_{D_a f}$ represents Walsh transform of the derivative of f with respect to a vector $a \in \mathbb{F}_2^n$. This property is special since it is related with the resistance of the S-boxes to the differential power analysis attacks where higher transparency order value means lower resistance to DPA attacks [14]. Since Boolean functions are a special form of S-boxes with one output variable (therefore, $m = 1$) maximum (and the worst) possible transparency order for a Boolean function equals 1 [14].

For a Boolean function to be usable in cryptography it needs to be balanced, with high nonlinearity, high algebraic degree and high correlation immunity. For further information about these properties we refer readers to [2]. Furthermore, if one aims at better resistance against side-channel analysis then the transparency order should be as low as possible. However, Prouff showed that minimal transparency equals 0 and is achieved only in case of linear or affine functions which are not suitable for cryptography [14]. Therefore, there are two problems when creating Boolean functions with improved transparency order. The first problem is to determine an *appropriate* level of nonlinearity from a cryptographic perspective. The second problem is to find a Boolean function with at least that level of nonlinearity and with transparency order as good as possible.

3 Evolving Boolean Functions

To capture good cryptographic properties, we need to devise an appropriate fitness function to guide the evolution process. In our experiments, fitness functions incorporate properties mentioned in Sect. 2 (balancedness, nonlinearity and transparency order). There exist more properties, but since there is a trade-off between some of the them we decided to go for as simple as possible fitness function [2].

The *balancedness* property is the only one which is always strict as a part of the fitness function, since unbalanced Boolean functions are not appropriate for cryptography. The balancedness property (BAL) is used as a penalty, and presented in pseudo-code it calculates as

```
**if** $(HW\,(TT) > \frac{2^n}{2})$ **then**
    $BAL = \frac{2^n - HW(TT)}{HW(TT)} \cdot X$
**else**
    $BAL = \frac{HW(TT)}{2^n - HW(TT)} \cdot X$
**end if**
```

where we experimentally find that X = - 5 scales well for Boolean functions with $n = 8$ inputs. Balancedness is rated gradually, so that a balanced function receives the value 1, and unbalanced functions receive a negative value corresponding to the level of unbalancedness in the range r, where $r \in [-1275, -5]$. Minimum value is given when the number of ones is either 0 or 2^n.

A balanced Boolean function has an upper bound on *nonlinearity* as given in [2]:

$$NL_f = 2^{n-1} - 2^{\frac{n}{2}-1}. \quad (4)$$

This bound can be achieved only when n is even. Functions that have maximal nonlinearity are called *bent functions*, but they are not appropriate for use in cryptography since they are not balanced. Therefore, we expect to find functions with nonlinearity below 120 for $n = 8$ case.

As mentioned in Section 2, the worst possible *transparency order* equals 1. Based on the results from Mazumdar et al. who show 0.1 decrease in transparency order over the AES example as 8×8 vectorial Boolean function, we conclude that it is possible to expect the improvement in transparency order of around 1/8th of 0.1 which amounts to 0.0125 [9]. Therefore, on the basis of the data from above, we see that the nonlinearity is in range $[0, 120]$ and transparency is in range $[0, 1]$ for a balanced Boolean functions with 8 inputs.

3.1 Fitness Functions

We present here the two settings we are interested in from the experimental perspective. In the first setting the goal is to find as high as possible nonlinearity value and for that nonlinearity level the best corresponding transparency order. This is represented with the following fitness function:

$$fitness_1 = BAL + NL_f + (1 - T_f). \quad (5)$$

Given Equation (5), the optimization problem considers the *maximization* of the fitness function. Note that, once the balancedness is achieved, the main driving force of fitness is the nonlinearity, with values in excess of 100. Since the nonlinearity NL_f can only assume integer values, and transparency T_f assumes values in $[0, 1]$, it is clear that transparency is used as a secondary objective, to be able to perform selection of different solutions with the same level of nonlinearity.

Since there are three different properties to optimize, one possible approach would be the use of multiobjective optimization. We do not use that approach for several reasons; first of all, the balancedness is an absolute requirement and should not be included as an independent measure. On the other hand, we have to differentiate the unbalanced solutions in the evolution and allow them to improve, so a part of the fitness function (BAL) is devised to produce the greatest penalty. The second most important property is nonlinearity, for which we want it to be as high as possible. Since we do not know in advance what nonlinearity is achievable, the algorithm should be driven to find the maximum value, regardless of the other properties. Only when this is evolved, we need to find solutions with the transparency order as good as possible. Since the nonlinearity can only assume a fixed number of levels, the search can be adjusted to suit those values, which is investigated in the following setting.

In the second setting the goal is to find low transparency order values for a certain nonlinearity level. Here we reiterate that the lowest possible transparency order is achieved for linear and affine functions, but since they are not appropriate for the use in cryptography we do not consider them as a viable choice [2]. Because it is difficult to say what would be an appropriate nonlinearity value, we decided to set several levels as a constraint. Therefore, we look for the best transparency order with a target minimum nonlinearity level, NL_t. We set 4 levels of nonlinearity, at the values of 86, 92, 98 and 104. Fitness function for the second set of experiments is defined as:

$$fitness_2 = BAL + (1 - T_f) - pos\left(2 \times (NL_t - NL_f)\right), \tag{6}$$

where the function $pos(x)$ returns x if $x > 0$ and zero otherwise.

3.2 Algorithms, Representations and Parameters

As previously mentioned there are several ways to uniquely represent Boolean functions. For GA we decided to represent the individuals as strings of bits where values are truth tables of functions, and GP individuals as trees of Boolean primitives which are then evaluated according to the truth table they produce. We use two selection mechanisms, namely steady-state with tournament operator (SST) and generational with roulette-wheel (RW) selection. In the first one, 3 solutions are selected at random and the worst among them is replaced by the offspring of the remaining two. The offspring is mutated with a given probability (0.3).

The RW selection uses the fitness proportional roulette-wheel operator applied to the whole population to select the new generation of survivors, to which crossover and mutation are then applied; in this scenario, the offspring replaces the parents. The same two selections are applied both to GA and GP. All the employed methods are a part of the Evolutionary Computation Framework (ECF) [6].

In the experiments we initialize population with random individuals where we do not require that the solutions are balanced. That is opposite from what is usually done [4, 11], but we expect that the evolution process would benefit from that additional diversity.

In an effort to find the differences in the performance of the algorithms, we also compare the best algorithms with the *balanced random search* algorithm. Balanced random search uses random sampling of solutions, but only considering balanced Boolean function as a solution candidate, as opposed to a purely random choice of solutions. Stopping criterion for the random search algorithm is 200 000 evaluations.

GA Variations. For GA representation, mutation is selected uniformly at random between simple mutation, where a single bit is inverted, and mixed mutation, which randomly shuffles the bits in a randomly selected subset. Additionally, we use a *balanced mutation* that preserves balancedness of the solution by changing 2 bits of the individual if it is already balanced.

For all mutation operators we experiment also with an *adaptive mutation rate*. In the beginning, the mutation is given as a fixed probability (0.3), but as the evolution starts to stagnate (i.e. no improvement in the best solution), the mutation probability raises. The probability is increased linearly with the number of generations without improvement, until it reaches a predefined maximum level (0.8) after a given maximum number of generations without improvement has passed. If a new best solution is found, the mutation rate is reset to initial value. The crossover operators used in GA are one-point and uniform crossover, performed at random for each new offspring.

GP Variations. Of the modifications in the previous section, with GP we employ only the adaptive mutation rate, in the same manner as for the GA. The function set for genetic programming in all the experiments is OR, NOT, XOR, AND, IF, and terminals correspond to 8 Boolean variables. Genetic programming has maximum tree depth of 11. For the Boolean functions we are interested only in XOR and AND operators, but it is quite easy to transform the function from one notation to the other.

Common Parameters. Parameters that are in common for every round of the experiments are the following: the size of Boolean function is 8 (the size of the truth table is 256) and the population size is 500. In the roulette-wheel selection the crossover probability is 0.5 and the ratio of the probability of survival of the best and the worst individual is scaled to 10. Mutation probability for non-adaptive variations is set to 0.3 per individual. The parameters above are the result of a combination of a small number of preliminary experiments and our experience with similar problems; no thorough parameter tuning has been performed.

Unlike in the traditional optimization case, our main goal is not to compare different approaches we implement, but rather to find the best possible solutions. In that case, we do not limit the algorithms with a fixed number of evaluations (and compare the averages), but allow the algorithms to run as long as something useful might occur. That is why the stopping criterion is set to a given number of generations without improvement, rather than a fixed maximum number, which is a reasonable criterion for researchers trying to find an adequate solution.

Further information regarding experimental setup is listed when needed.

4 Results and Discussion

When presenting the best achieved results, we also compare them with some of the existing results from the literature. For previously published Boolean functions, we use the following notation: for the Boolean function from the Burnett et al. paper we use the name Function_1 [4], from the work by McLaughlin and Clark we abbreviate Boolean function with Function_2 [10]. For the Boolean function from the work by Picek et al. we abbreviate it with Function_3 [13], and finally, for Rakaposhi Boolean function we abbreviate it with Rakaposhi [5].

Here we emphasize that Boolean function in Rakaposhi cipher is obtained through algebraic construction, more specifically the finite field inversion method. Rakaposhi stream cipher is an example of a modern, state-of-the-art stream cipher that uses Boolean function as a nonlinear element. The algorithm names presented in tables are abbreviated in the following way: after the abbreviation of the algorithm we write in the subscript the distinguishing properties of the algorithm: *SST* represents steady state tournament selection, *RW* roulette wheel selection, *balanced* means balanced mutation operator and *variable* represents variable mutation rate.

With the objective to find the best individuals, the stopping criterion is set to 50 generations without improvement for all algorithm variations and the number of independent runs is 400. In Table 1 we give the best result for each of the experiments conducted, as well as for random search algorithm and Boolean functions from related works. Here, evolutionary algorithms use fitness function as defined by Equation (5). All solutions are balanced so we did not specifically write that property.

Table 1. Best Boolean functions, $fitness_1$

Algorithm	NL_f	T_f	Algorithm	NL_f	T_f
GP_{SST}	**116**	**0.962**	$GP_{SST,variable}$	**112**	**0.919**
$GP_{RW,variable}$	112	0.965	GA_{SST}	112	0.934
$GA_{SST,balanced,variable}$	112	0.931	$GA_{SST,balanced}$	112	0.938
$GA_{RW,balanced}$	112	0.935	*Random search*	110	0.934
Function_1	100	0.927	*Function_2*	116	0.969
Function_3	116	0.976	*Rakaposhi*	112	0.946

Based on the related work, we choose nonlinearity levels of 112 and 116 as the most interesting ones. The best (lowest) transparency results for those nonlinearity levels are in bold style.

When optimizing the second fitness function (6), we concentrate on 4 predefined nonlinearity levels. Here we do not compare this results with literature since previous works did not investigate transparency property (and therefore those works can have better nonlinearity in general, but when looking at both of those properties related works have worse results). In this case we are interested in the lowest transparency value that can be obtained with a given minimum nonlinearity. For $fitness_2$ equation the best results are presented in Table 2.

Table 2. Best Boolean functions, $fitness_2$

Algorithm	86	92	98	104
$GP_{SST,variable}$	0.774	0.815	0.866	0.898
$GA_{SST,balanced,variable}$	0.78	0.817	0.822	0.866
Random Boolean	0.887	0.894	0.905	0.915

The space of possible Boolean functions is huge and therefore it is impractical to do an exhaustive search for Boolean functions with the number of inputs relevant in cryptography. However, the space of Boolean functions with "good" cryptographic properties is also large. In such a large space it is difficult to find Boolean functions with excellent cryptographic properties. This is an obvious example of the convergence of the algorithm towards the local optima. In an attempt to search beyond those local optima we employ different algorithms and modifications.

In the experiments we use two different selection methods where we expected that the roulette-wheel selection should be the best one, because preliminary results (also the results from other researchers) showed that all algorithms display very quick convergence. However, algorithms with the steady-state tournament selection consistently found the best solutions among all the algorithms.

A simple fitness function, that includes only a subset of the desired properties, has the advantage that there are no conflicts between variables, and some high quality properties inherently mean that other properties will also be good. In our previous experiments, preliminary results show that it is not trivial to combine many properties in a single fitness value because of varying magnitudes (scaling issues) of different property values.

Statistical analysis (not presented in the paper) shows that all GA variations give similar results with smaller standard deviation than in GP variations. We also conducted pairwise comparison between GP algorithm with steady-state tournament selection and all other algorithms where the results show there are no statistically significant differences. However, genetic programming with steady-state tournament produced the single best result when considering the nonlinearity value. For a nonlinearity level 112 the best result was achieved with

$GP_{SST,variable}$ algorithm. In this case we can also see that this Boolean function has better transparency order value than the Rakaposhi Boolean function.

As evident form the results, we found Boolean functions with better transparency order values but the improvements are rather small. One could ask if such small improvements make a difference. We consider this to be true, since we did not expect obtaining a big difference if the analogy with S-boxes (where 0.1 is a significant improvement) is valid. Furthermore, side-channel analysis of stream ciphers is more difficult and therefore even small improvements are relevant. Naturally, further investigation of the relevance of the transparency order property is needed in order to put these results in proper perspective. Our new Boolean functions present viable choice for future implementations since they are offering improvement in the properties while not bringing additional area or speed drawbacks.

5 Conclusions and Future Work

Finding Boolean functions with improved DPA resistance is a difficult problem, not only because of the huge search space, but also due to the lack of previous work. As far as we know, we are the first to experiment with the transparency order property for Boolean functions and additionally to use evolutionary algorithms to find suitable functions. Our experiments showed that it is possible to find Boolean functions with better transparency order than that in reference work. From cryptographic perspective, genetic programming achieve the highest nonlinearity level with an improved transparency order value. However, from the evolutionary point of view, the results show no statistically significant difference with respect to the genetic algorithm. Due to the lack of prior work on this topic, results obtained here should be also regarded as a baseline for future research.

As future research directions we plan to experiment with other algorithms like Cartesian GP or Estimation of Distribution. Initial experiments give promising results.

Acknowledgments. This work was supported in part by the Technology Foundation STW (project 12624 - SIDES), The Netherlands Organization for Scientific Research NWO (project ProFIL 628.001.007) and the ICT COST action IC1204 TRUDEVICE.

References

1. Aguirre, H., Okazaki, H., Fuwa, Y.: An evolutionary multiobjective approach to design highly non-linear boolean functions. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2007, pp. 749–756 (2007)
2. Braeken, A.: Cryptographic Properties of Boolean Functions and S-Boxes. PhD thesis, Katholieke Universiteit Leuven (2006)
3. Burnett, L.: Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. PhD thesis, Faculty of Information Technology, Queensland University of Technology (2005)

4. Burnett, L., Millan, W., Dawson, E., Clark, A.: Simpler methods for generating better boolean functions with good cryptographic properties. Australasian Journal of Combinatorics 29, 231–247 (2004)
5. Cid, C., Kiyomoto, S., Kurihara, J.: The RAKAPOSHI Stream Cipher. In: Qing, S., Mitchell, C.J., Wang, G. (eds.) ICICS 2009. LNCS, vol. 5927, pp. 32–46. Springer, Heidelberg (2009)
6. Jakobovic, D., et al.: Evolutionary computation framework (December 2013), `http://gp.zemris.fer.hr/ecf/`
7. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman and Hall/CRC, Boca Raton (2008)
8. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Advances in Information Security. Springer-Verlag New York, Inc., Secaucus (2007)
9. Mazumdar, B., Mukhopadhay, D., Sengupta, I.: Constrained Search for a Class of Good Bijective S-Boxes with Improved DPA Resistivity. IEEE Transactions on Information Forensics and Security PP(99), 1 (2013)
10. McLaughlin, J., Clark, J.A.: Evolving balanced boolean functions with optimal resistance to algebraic and fast algebraic attacks, maximal algebraic degree, and very high nonlinearity. Cryptology ePrint Archive, Report 2013/011 (2013), `http://eprint.iacr.org/`
11. Millan, W.L., Clark, A.J., Dawson, E.: Heuristic design of cryptographically strong balanced boolean functions. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 489–499. Springer, Heidelberg (1998)
12. Picek, S., Ege, B., Batina, L., Jakobovic, D., Chmielewski, L., Golub, M.: On Using Genetic Algorithms for Intrinsic Side-channel Resistance: The Case of AES S-box. In: Proceedings of the First Workshop on Cryptography and Security in Computing Systems, CS2 2014, pp. 13–18. ACM, New York (2014)
13. Picek, S., Jakobovic, D., Golub, M.: Evolving Cryptographically Sound Boolean Functions. In: GECCO (Companion), pp. 191–192 (2013)
14. Prouff, E.: DPA Attacks and S-Boxes. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 424–441. Springer, Heidelberg (2005)