On Using Genetic Algorithms for Intrinsic Side-Channel Resistance: The Case of AES S-Box

No Author Given

No Institute Given

Abstract. Finding balanced S-boxes with high nonlinearity and low transparency order is a difficult problem. The property of transparency order is important since it specifies the resilience of an S-box against differential power analysis. Many S-boxes used today have high transparency order and are therefore intrinsically more susceptible to side-channel analysis. Better values for transparency order and hence improved side-channel security often imply less in terms of nonlinearity. Therefore, it is impossible to find an S-box with all optimal values. However, well-balanced S-boxes with low transparency order and relatively high nonlinearity present a valuable alternative when implementing symmetric ciphers on embedded devices like smart cards. Currently, there are no algebraic procedures that can give the preferred and complete set of properties for an S-box. In this paper, we employ evolutionary algorithms to find S-boxes with desired cryptographic properties. Specifically, we conduct experiments for the 8×8 S-box case as used in the AES standard. The results of our experiments proved the feasibility of finding S-boxes with the desired properties in the case of AES. In addition, the results of our preliminary side-channel analysis on different versions of "improved" S-boxes show the method effective in terms of increasing DPA resistance.

Keywords: Private-key Cryptography, Block Ciphers, S-Box, Transparency Order, Genetic Algorithms, Side-channel Analysis

1 Introduction

Block cipher algorithms are vulnerable to various kinds of cryptanalysis. Besides more traditional linear [1] and differential cryptanalysis [2], the most popular attacks today belong to side-channel analysis (SCA) targeting actual implementations of cryptography in software or hardware. SCA relies on the physical leakages from the actual implementation and its efficiency is much greater than the one of linear or differential cryptanalysis [3]. Through various countermeasures such as numerous masking and hiding schemes [4, 5] it is possible to make the algorithm more resilient to SCA. However, this comes with a substantial cost increase due to the increase of memory requirements and the decrease of performance of the algorithm implemented.

In the design process of block ciphers one usually follows principles of diffusion and confusion as introduced by Shannon [6]. The amount of confusion in an algorithm is measured with the nonlinearity property of nonlinear parts of algorithm e.g. Boolean functions and S-boxes. Considering side-channel security, Prouff [7] defines the transparency order property that characterizes the resistance of S-boxes to the SCA or more precisely to differential power analysis (DPA) [8]. However, there are still no algebraic methods available to design S-boxes that have good transparency order and adequately high nonlinearity. Since the worst transparency order is obtained in the case when bent functions are used (as bent functions obtain maximal nonlinearity [9]) it implies the fact that nonlinearity and transparency order are conflicting criteria.

When random generation is used for S-boxes, the results have reduced nonlinearity when compared to many specially constructed S-boxes. This suggests that the transparency order will be improved. However, this improvement in transparency order value results in deterioration of other properties. Therefore, random generation does not present a viable choice in the generation of S-boxes with good transparency order.

In this paper we follow this intuition but we also do better in terms of making good trade-offs among all the properties. We use evolutionary computation techniques to evolve S-boxes with good transparency order and acceptable nonlinearity values. More precisely, this work makes the first step in using this powerful method on a very practical cryptographic problem. We aim at finding better alternatives for S-boxes, as used in block ciphers or other symmetric cryptographic primitives, in terms of improving their resistance against side-channel analysis without too much deteriorating the security of S-boxes. Our general goal is to come up with an evolutionary computation framework for finding "proper" S-boxes that is both, effective and efficient. Naturally, a design method that does not favour special methods (e.g. algebraic based) also has several downsides. The most obvious bottleneck is the inability to store S-box in a format different from a lookup table. From that perspective it is unlikely that the new S-boxes can be used in every environment. However, on platforms with sufficient area for the lookup tables and where resilience against SCA is of great importance, we are confident that our method can be a viable alternative. In this case, as table lookups are also susceptible to cache attacks, a cache-timing resistant lookup table could be used e.g. as presented by Bernstein [10].

1.1 Related Work

Relevant previous works use evolutionary computation in different scenarios related to the design of cryptographic primitives. Here, we mention two important cases: evolving Boolean functions and evolving S-boxes.

Evolving Boolean functions. Burnett et al. construct two simple heuristic methods to evolve balanced Boolean functions with good nonlinearity and autocorrelation properties [11]. Also, Aguirre et al. use multi-objective evolutionary technique to evolve balanced Boolean functions with high nonlinearity [12].

Further examples where evolutionary techniques are used to evolve Boolean functions with cryptographic properties are given in the works of Millan et al. [13] and Jacob et al. [14]. In each of these papers, some evolutionary technique to evolve Boolean functions with good cryptographic properties is used. Main difference is in the set of properties of interest. However, one should note that evolving S-boxes is much harder than evolving Boolean functions since the solution space is larger in the case of S-boxes.

Evolving S-boxes. Clark et al. use the principles from the evolutionary design of Boolean functions to evolve S-boxes with desired cryptographic properties [15]. They used simulated annealing technique coupled with hill-climbing algorithm to evolve bijective S-boxes with high nonlinearity. On the other hand, Burnett et al. use heuristic method to generate MARS-like S-boxes [16]. Such an approach generates S-boxes that adhere to the all requirements and is computationally fast.

The two papers mentioned above represent successful applications of metaheuristics to the creation of S-boxes. However, it is difficult to compare those works with ours since they did not evolve 8×8 S-boxes nor they investigated the transparency order property.

Mazumdar et al. design rotation symmetric S-boxes with high nonlinearity and DPA resistance [17]. Furthermore, they employ those S-boxes in several hardware implementations and show that their S-boxes have better DPA resistance than the AES S-box.

Our work is the first one to use the techniques of evolutionary computation in finding cryptographically strong S-boxes that feature also improved side-channel resilience. More details on our contributions are given below.

1.2 Our Contribution

When using evolutionary computation techniques a special caution is required as evolutionary algorithms are not magic-solvers for any kind of problem. They can help in finding viable solutions but to have something feasible or in this case suitable for real-life applications, all the conditions have to be taken into account and treated specifically. Wolpert and Macready introduce the "No Free Lunch" theorem and prove that there is no single best algorithm for every problem [18]. Of course, this theorem is only applicable when we possess no knowledge about the problem at hand. With a careful choice of evolutionary computation technique and with adequate settings, evolutionary computation can be used to solve various real-world problems.

Here we need to reiterate that evolutionary computation should not be regarded as the best possible method for solving this problem (or any problem). Rather, it present a set of generic algorithms that can be successfully applied to many problems.

In this paper, we use evolutionary computation technique, specifically genetic algorithm, to evolve S-boxes with low transparency order and relatively high nonlinearity values. To be able to do that, we experiment with several versions of evolutionary computation techniques to find the best one. Also we present

simple, yet effective fitness function we use to find new S-boxes. The experiments prove that evolutionary algorithms are a viable option in evolving S-boxes with low transparency order and high nonlinearity. In addition, we show the results of practical experiments that confirm our findings. For this purpose, we use power consumption traces derived from a programmable smart card on which our new improved S-boxes are implemented. More precisely, we conduct two different types of the experiments. The first type are experiments to evolve S-boxes, and the second type are the experiments to evaluate the resistance of evolved S-boxes to DPA attacks. To avoid the confusion, for the former experiments we use the name evolutionary experiments and for the latter side-channel experiments.

The remainder of this paper is organized as follows: In Sect. 2 we survey necessary information about evolutionary computation and cryptographic properties of S-boxes. In Sect. 3 our evolutionary computation experimental setup and the results are presented. Sect. 4 contains a discussion about the implementation of evolved S-boxes and our first results from side-channel analysis. Finally, in Sect. 5 we conclude the paper.

2 Preliminaries

Here we give necessary information about side-channel analysis, cryptographic properties of S-boxes and evolutionary computation.

2.1 Side-channel Analysis and DPA

Small cryptographic devices, such as smart cards, RFID tags etc. have become pervasive in our lives and lots of our security and privacy-sensitive data is stored on those constrained platforms. Cryptographic algorithms used to preserve the security and privacy of their users are typically implemented in software or hardware on those physical devices that interact with and are influenced by their environments. These devices provide unintentional output channels, often called side channels. Sometimes, these types of information leakages may be linked either to the types of operations that the cryptographic algorithm is performing, or to the data, i.e., the keys being processed. This makes the leakages explorable by the adversary trying to extract the secret key as she is always looking for shortcuts in cryptanalysis. Considering the physical information explored there are several side channels possible. The best known and most commonly used side-channel is power consumption. If the adversary is able to collect many power consumption traces, she can use powerful statistical and mathematical methods to recover the key from the time series data. In this case the attack performed is called Differential Power Analysis (DPA).

Side-channel attacks are the main security threat for smart cards since the first academic publications by Kocher et al. [8, 19]. Different sources of side-channel data, such as electromagnetic emanation [20, 21], timing [19], sound, and temperature have been used for successful side-channel attacks (for a general overview see e.g. [5]).

2.2 Cryptographic Properties of S-boxes

Here we present the properties that are used in evaluation of S-boxes by evolutionary algorithms. Other relevant cryptographic properties are calculated a posteriori and presented in Sect. 3.4. Some details about those properties are given in Appendix A.
The addition modulo 2 is denoted as " $\oplus$ ". The inner product of vectors $\bar{a}$ and $\bar{b}$ is denoted as $\bar{a} \cdot \bar{b}$ and equals $\bar{a} \cdot \bar{b} = \oplus_{i=1}^{n} a_i b_i$.
An (n, m)-function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ [7]. Such a function F is called S-box or vectorial Boolean function. If m equals 1 then the function is called Boolean function. Boolean functions f_i, where $i \in \{1, ..., m\}$ are coordinate functions of F and every Boolean function has n variables. Hamming weight HW of a vector $\bar{a}$, where $\bar{a} \in \mathbb{F}_2^n$, is the number of non-zero positions in the vector.

An (n, m)-function is called balanced if it takes every value of $\mathbb{F}_2^m$ the same number 2^{n-m} of times [22]. Balanced (n, n)-functions are permutations on $\mathbb{F}_2^n$.

Nonlinearity N_F of an (n, m)-function F equals minimum nonlinearity of all non-zero linear combinations $\bar{b} \cdot F$, where $\bar{b} \neq 0$, of its coordinate functions f_i [3].

$$N_F = 2^{n-1} - \frac{1}{2} max_{\substack{\bar{a} \in \mathbb{F}_2^n \\ \bar{v} \in \mathbb{F}_2^{m*}}} |W_F(\bar{a}, \bar{v})| \tag{1}$$

Here, $W_F(\bar{a}, \bar{v})$ represents Walsh transform of F [7].

$$W_F(\bar{a}, \bar{v}) = \sum_{\bar{x} \in \mathbb{F}_2^n} (-1)^{\bar{v} \cdot F(\bar{x}) \oplus \bar{a} \cdot \bar{x}} \tag{2}$$

In 2005, Prouff introduced a new cryptographic property of S-boxes: transparency order [7] which can be defined for a (n, m)-function as follows.

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

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 unlike the ones known up to that time (with the exception of SNR (DPA) (F) property [23]) since it is related with the resistance of the S-boxes to the DPA attacks. According to Prouff, transparency order has an upper bound of m for an (n, m)-function. This bound is achieved if every coordinate function f_i is bent function. In the case F is an affine function, then the transparency order is zero. The higher the transparency order value is, the lower is the S-box resistance to the DPA attacks [7]. Since bent functions have maximum nonlinearity, we can see that high nonlinearity and low transparency order are conflicting criteria. Carlet also showed that some S-boxes with very high nonlinearity have very bad transparency orders [3].

2.3 Evolutionary Computation and Evolutionary Algorithms

Evolutionary computation (EC) is a subfield of computation intelligence area that draws inspiration from the process of natural evolution [24]. In that process it adheres to the theory of Darwinian evolution. In accordance with that, it is natural that researchers try to develop algorithms that are based on evolutionary process.

Evolutionary algorithms (EAs) are population based optimization algorithms that use biology inspired mechanisms to refine a set of solution candidates iteratively [25]. From this definition we can see that all evolutionary algorithms have some common underlying idea behind them. Therefore, it is possible to give basic evolutionary scheme that is common to all evolutionary algorithms.

Algorithm 1 Basic Evolutionary Algorithm

```
Input : Parameters of the algorithm
Output : Optimal solution set
t ← 0
P(0) ← CreateInitialPopulation
while TerminationCriterion do
  t ← t + 1
  P'(t) ← SelectMechanism (P(t − 1))
  P(t) ← VariationOperators(P'(t))
end while
Return OptimalSet(P)
```

There are many techniques that are used in evolutionary algorithms. Well known examples are Genetic Algorithms (GA), Genetic Programming (GP) and Evolutionary Strategies (ES).

For further information about evolutionary algorithms we refer to [24–27].

Genetic Algorithms. Genetic algorithms are a subclass of evolutionary algorithms where the elements of the search space S are arrays of elementary types [25]. Today, genetic algorithms represent evolutionary technique that has been successfully applied to various optimization problems. Usual variation operators are mutation and crossover (recombination) operators. Mutation operators are operators that use one parent to create one child by applying randomised changes to parent. Mutation depends on the mutation rate p_m which determines the probability that a change will occur within individual. Recombination operators work on two or more parents to create offspring from the information contained within parent solutions. Recombination is usually applied probabilistically according to a crossover rate p_c. Besides variation operators, it is necessary to decide about selection method. Today, the k-tournament selection method is widely used for this purpose [25].

3 Experimental Settings and Results

In all our evolutionary experiments we use the genetic algorithm that follows basic algorithm as presented in Section 2.3.

The goal is to evolve balanced bijective S-boxes with high nonlinearity and low transparency order. We experiment with the 8×8 size S-box as this is the size of AES S-box which represents the standard for block ciphers.

3.1 Fitness Function, Representation and Parameters

Maximization of the value of a fitness function is the objective in all evolutionary experiments. Fitness function represents definition of the problem to solve with evolutionary algorithm. For fitness function we use a combination of balancedness, nonlinearity and transparency order properties. Since we require that the solutions are balanced, we do not add balancedness to the fitness function. Rather, we set it as a constraint that needs to be fulfilled to evaluate the fitness value of an individual.

Our fitness function equals the sum of nonlinearity (Nl) and transparency order (Tr) properties values. Since the transparency order value should be as low as possible, we subtract the value obtained from the upper bound value for transparency order.

This fitness function can be easily extended to contain more properties that are of relevance to the evolutionary experiments.

$$fitness = max\left[Nl + (m - Tr)\right] \tag{4}$$

In the genetic algorithm we use permutation representation of solutions.

When using permutation representation, the problem of finding good S-boxes can be informally treated as an special instance of travelling salesman problem (TSP) [28, 29]. In TSP the objective is to find the optimal path between all the cities in the map (or more generally, objective is to decide on the order of values). Here, we wanted to find the optimal path between values in S-box lookup tables. When regarded as a TSP, we can conclude the problem is hard since there is $256! - 2$ possible solutions (we neglect solutions where the output of the lookup table is the same as the input and where AES S-box is a solution). In the permutation representation, S-box is represented with decimal values between 0 and 255 (256 distinct values) where each of those values is one entry for S-box lookup table.

Parameters for the evolutionary algorithm are following: the size of (m, n)-function is 8×8, number of independent runs for each evolutionary experiment is 30 and the population size is 100. Tournament size in steady-state tournament selection is equal to 3. Mutation probability is set to 0.3 per individual. This mutation rate is chosen on a basis of a small set of tuning experiments where it showed the best results on average.

3.2 Time and Memory Complexity

The genetic algorithm used in the evolutionary experiments takes constant extra space as it needs only the space to store the S-box in bitstring format for the evaluation process. In the case that time is constrained, it is possible to speed up the evolutionary process. Currently, dominant time in the fitness evaluation is calculating the transparency order. In our code, that calculation lasts approximately 90% of complete evaluation time. By implementing e.g. the algorithm presented by Fan et al. fitness function could be calculated faster [30].

Evaluating time complexity of evolutionary algorithm is impractical since it depends on the number of generations until termination and number of individuals in the population. It is relevant only to consider time complexity of evaluation function (fitness function) for one individual. In fitness calculation the dominant part is the transparency order computation. Our implementation has time complexity of $\mathrm{O}\left(2^{3 \cdot n}\right)$, where n is the number of inputs for each coordinate function of F.

3.3 Evolutionary Process

Next, we give detailed description of genetic algorithm we use.

Once the parameters of GA are set, we can start with the generation of the initial population. Initial population is created by randomly setting each value from 0 to 255 as outputs of a lookup table where inputs are in lexicographical order. When the initial population is generated, genetic algorithm starts with the evolution process. First step is that it chooses k possible solutions and from those k solutions destroys the $k-2$ worst solutions (this selection method also ensures elitism i.e. the best solutions are propagated to the next generation). From the 2 remaining parents we obtain $k-2$ offspring solutions via variation operators. Goodness of a solution is evaluated with the fitness function as presented in Equation (4).

For variation operators we use 3 mutation operators and 3 crossover operators (we chose the operators that are among the most common ones in use today). It is important to state that in permutation representation operators can not create a duplicate of a solution. We use insert mutation, inversion mutation and toggle mutation. Mutation operator works by randomly moving alleles within a solution, where the probability that some position will change its value equals p_m. For crossover operators we use partially mapped crossover (PMX), position based crossover (PBX) and order crossover (OX). Crossover operators for permutation based representation must transmit as much as possible of information contained within parents while maintaining the permutation. Operators are selected uniformly at random between all operators within a class of operators (mutation and crossover). Further informations about variation operators can be found in Appendix B and [24]. Such new created offspring solutions together with the parents form new generation in the evolution process.

Evolution process lasts until the stopping criterion is met, here the stopping criterion is a certain number of generations without improvement of the best solution.

3.4 Genetic Algorithm Results

Several examples of S-boxes are given in Table 1. First two S-boxes should be regarded as benchmarks, since first one is the AES S-box and second one is a randomly created S-box.

S-boxes 1 to 5 are examples of evolved S-boxes. Additionally, we give values for the following cryptographic properties: algebraic degree (deg) [9,31], correlation immunity (CI) [9,31], signal-to-noise ratio (SNR) [23], global avalanche criterion (GAC) - absolute indicator (Δ_F) and sum-of-square indicator (σ_F) [31,32], differential δ-uniformity (δ-uniformity) [22, 33] and strict avalanche criterion (SAC) [31, 34].

Lookup tables of S-boxes 1 to 5 in hexadecimal format are given in Appendix C.

Table 1. Cryptographic Properties of S-boxes

S-box	Nl	Tr	SNR	δ-unif.	Δ_F	σ_F
AES S-box	112	7.86	9.599	4	32	133120
Random S-box	92	7.804	10.001	6	96	257152
S-box 1	100	7.716	8.686	6	104	245632
S-box 2	98	7.358	5.825	6	104	341248
S-box 3	98	7.41	6.034	6	112	370816
S-box 4	100	7.53	5.44	6	104	298624
S-box 5	98	7.50	6.547	6	112	356224

All the S-boxes enumerated in the table are balanced so we did not write that property in the table. Also, all the S-boxes have algebraic degree equal to 7. We omitted correlation immunity property from the table since it must be 0 as evident by Siegenthaler's inequality [9]. Further, none of the S-boxes satisfy SAC property so we also omitted it from the table.

In Fig. 1 we displayed some results of evolutionary experiments. Circles represent one million random S-boxes results, the plus symbol represents AES S-box, the diamond symbol represents evolved S-box 1, and finally, the triangle symbol represents evolved S-box 2.

Distribution of the random S-boxes values is also shown in Table 2.

As evident from Table 1 and Fig. 1, finding S-boxes with low transparency order and high nonlinearity is hard. Low nonlinearity value does not ensures low transparency order. In fact, it is easy to find S-boxes with nonlinearity below 90 and with transparency order comparable to that of AES S-box. Since we could not find any S-box with nonlinearity level the same as in AES case and with significantly lower transparency order, we opted to find S-boxes with nonlinearity lower than in AES, but also with transparency order significantly lower than in

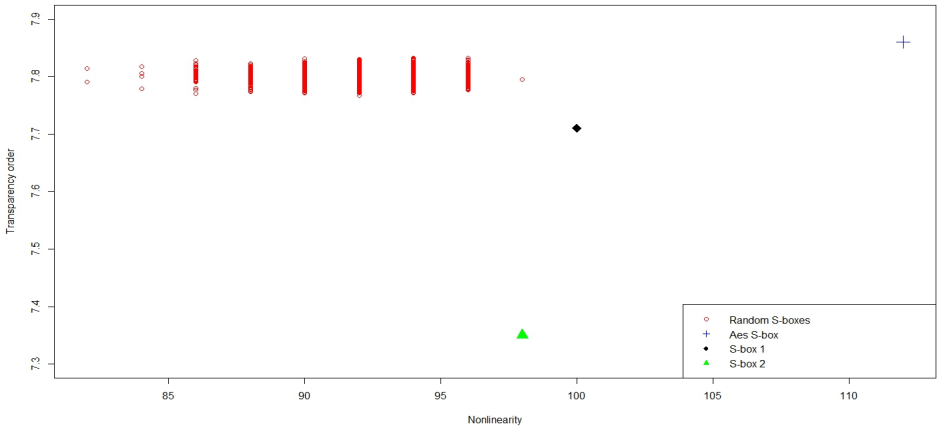

Fig. 1. Nonlinearity versus transparency order for S-boxes

AES case. Since the evolved S-boxes must be implemented through lookup tables while they have lower transparency value and higher GAC, transparency order must be low enough to justify it.

Here we can also make distinction between two different hard problems, one is finding as low as possible transparency order value while maintaining adequate nonlinearity level, and second problem is to find S-boxes with nonlinearity value between 100 and 112 while having low transparency order values. In the case of evolved S-boxes 1 and 2 from Table 1, we consider S-box 2 to be much better since its nonlinearity is only slightly lower while its transparency order value is significantly lower than in the case of S-box 1.

S-box 2 was evolved in 2325th generation of genetic algorithm which took 96000 seconds and S-box 4 was evolved in 124th generation and that took 4600 seconds. In the evolution process we used a cluster of computers where the average machine is Pentium with 2.6 GHz and 2 GB RAM. The algorithm was implemented in C++ programming language.

Table 2. Distribution of random S-boxes values

Property	Max	Min	Mean	Std. dev.
Nonlinearity	98	82	92.65	2.18
Transparency order	7.83	7.77	7.8	0.01

4 Implementing Evolved S-boxes and DPA Resistance

Using an S-box which has evolved in a way that is explained in previous sections of the work can be a quite challenging task when area constrained devices are concerned. Since the S-box is not generated through algebraic methods but evolutionary methods, the only way to implement these S-boxes is by using lookup tables (LUTs). When smart cards are considered, a software implementation of AES would make use of lookup tables. Therefore one of the most important platforms, where side-channel analysis is considered as a major threat, can be strengthened by using one of the proposed S-boxes at virtually no additional cost. Here, one can argue that it is not entirely area friendly to implement a lookup table for an 8×8 S-box, it should also be considered that side-channel resistance always comes at cost. Moreover, an additional drawback in hardware implementations would be the limited choice of options if a masking scheme is required together with the proposed S-boxes [35]. Therefore, the suitability of using such S-boxes in hardware implementations remains to be considered.

Aside from evaluating the cryptographic properties of the proposed S-boxes, we also evaluated side-channel resistance of software implementation of the new S-boxes. We implemented the new S-boxes on a smart card with an ATMega163 microcontroller. The measurements were collected with a PC oscilloscope at 250 million samples per second sampling rate. A straightforward software implementation of AES is modified to use the proposed S-boxes in side-channel experiments. For running the attacks, the output of the `SubBytes` operation is targeted and Hamming weight model is used to estimate the power consumption. The power estimation for each key candidate is checked for fitness with the actual power measurements through Pearsonn correlation. The experiment is repeated for 10 different keys selected at random, and the success rate is computed following the methodology proposed by Standaert et al. [36]. The results of our analysis are presented in Figure 2. The analysis is done on an AES implementation which processes the 16 S-box lookups in a random order for each execution of the code. This way, the noise level is increased and therefore the effect of the transparency order is more visible. Practical experiments are done for two new S-boxes: one with the lowest transparency order values we have managed to obtain, and another with the highest nonlinearity and the lowest transparency order for that nonlinearity. It is evident from the figure that using S-boxes with lower transparency order values results in an immediate improvement over the AES S-box in terms of side-channel resistance. However, it is evident that more experiments and more detailed analysis are needed to evaluate properly the impact of transparency order as defined by Prouff.

We further observe that the effect of transparency order is less obvious when the level of noise is low. More precisely, the correlation values obtained for the incorrect key guesses increase when S-boxes with lower transparency order values are used. This suggests that more experiments with high level of noise and other countermeasures are interesting for future studies.

When comparing the results obtained in this research with those of Mazumdar et al. [17] we can note that their S-boxes have slightly higher nonlinear-

ity (102 compared to ours 98 or 100) but our S-boxes have significantly lower transparency order values (7.76 compared to 7.35). Based on the results from their hardware implementations on an FPGA board we expect that our S-boxes would be viable alternatives improving even further side-channel resistance in those settings.

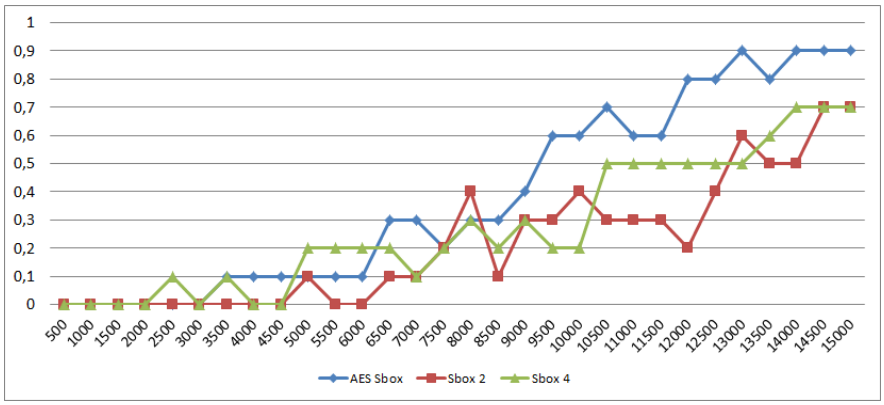

Fig. 2. Ranking of the correct key byte with number of traces

5 Conclusion

In this work we promote for the first time the use of GAs for evolving S-boxes with improved side-channel resistance. Our approach shows potential in both creation of S-boxes as well as in the evaluation. However, we are aware of the difficulties that lookup table approach could pose. Nevertheless, we do believe that our results have practical values. In general, one can consider the results as a proof of existence of S-boxes with desired properties. We expect that the results can be optimized further but the main goal was to find S-boxes with high nonlinearity and low transparency order. In this research we used generic GA but the results can be improved by employing custom made GAs, some other evolutionary algorithms like Estimation of Distribution Algorithm, or even to go outside EA area using Swarm Intelligence algorithms and use algorithms like Particle Swarm Optimization.

Above all, this paper promotes evolutionary computation as a serious tool for tackling some hard problems in cryptography i.e. evolving S-boxes and their transparency order. In near future, we expect many more attempts in using those mature optimization techniques on cryptographic and other security-related problems.

References

1. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques. EUROCRYPT'92, Berlin, Heidelberg, Springer-Verlag (1993) 81–91
2. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology. CRYPTO '90, London, UK, UK, Springer-Verlag (1991) 2–21
3. Carlet, C.: On highly nonlinear S-boxes and their inability to thwart DPA attacks. In: Proceedings of the 6th international conference on Cryptology in India. INDOCRYPT'05, Berlin, Heidelberg, Springer-Verlag (2005) 49–62
4. Akkar, M.L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems. CHES '01, London, UK, UK, Springer-Verlag (2001) 309–318
5. 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, NJ, US1A (2007)
6. Shannon, C.: Communication theory of secrecy systems. Bell System Technical Journal **28**(4) (1949) 656–715
7. Prouff, E.: DPA Attacks and S-Boxes. In: Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers. Volume 3557 of Lecture Notes in Computer Science., Springer (2005) 424–441
8. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In Wiener, M., ed.: Advances in Cryptology: Proceedings of CRYPTO'99. Number 1666 in Lecture Notes in Computer Science, Springer-Verlag (1999) 388–397
9. Braeken, A.: Cryptographic Properties of Boolean Functions and S-Boxes. PhD thesis, Katholieke Universiteit Leuven (2006)
10. Bernstein, D.J.: Cache-timing attacks on AES (2004) http://cr.yp.to/papers.html#cachetiming.
11. Burnett, L., Millan, W., Dawson, E., Clark, A.: Simpler methods for generating better Boolean functions with good cryptographic properties. Australasian Journal of Combinatorics **29** (2004) 231–247
12. 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'07. (2007) 749–756
13. Millan, W., Clark, A., Dawson, E.: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. In: Advances in Cryptology - EUROCRYPT '98. (1998) 489–499
14. Clark, J.A., Jacob, J.L., Stepney, S., Maitra, S., Millan, W.: Evolving Boolean Functions Satisfying Multiple Criteria. In: Progress in Cryptology - INDOCRYPT 2002. (2002) 246–259
15. Clark, J.A., Jacob, J.L., Stepney, S.: The design of S-boxes by simulated annealing. New Generation Computing **23**(3) (September 2005) 219–231
16. Burnett, L., Carter, G., Dawson, E., Millan, W.: Efficient Methods for Generating MARS-Like S-Boxes. In: Proceedings of the 7th International Workshop on Fast Software Encryption. FSE '00, London, UK, UK, Springer-Verlag (2001) 300–314

17. Mazumdar, B., Mukhopadhyay, D., Sengupta, I.: Design and implementation of rotation symmetric S-boxes with high nonlinearity and high DPA resilience. In: Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on. (2013) 87–92
18. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation **1**(1) (April 1997) 67–82
19. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In Koblitz, N., ed.: Advances in Cryptology: Proceedings of CRYPTO'96. Number 1109 in Lecture Notes in Computer Science, Springer-Verlag (1996) 104–113
20. Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smard Cards. In Attali, I., Jensen, T.P., eds.: Smart Card Programming and Security (E-smart 2001). Volume 2140 of Lecture Notes in Computer Science., Springer-Verlag (2001) 200–210
21. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic Analysis: Concrete Results. In Ç.K. Koç, Naccache, D., Paar, C., eds.: Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Number 2162 in Lecture Notes in Computer Science, Springer-Verlag (2001) 255–265
22. Crama, Y., Hammer, P.L.: Boolean Models and Methods in Mathematics, Computer Science, and Engineering. 1st edn. Cambridge University Press, New York, NY, USA (2010)
23. Guilley, S., Pacalet, R.: Differential Power Analysis Model and Some Results. In: In proceedings of CARDIS 2004, Kluwer Academic Publishers (2004) 127–142
24. Eiben, A.E., Smith, J.E. In: Introduction to Evolutionary Computing. Springer-Verlag, Berlin Heidelberg New York, USA (2003)
25. Weise, T. In: Global Optimization Algorithms Theory and Application. (2009) http://www.it-weise.de/.
26. Dumitrescu, D., Lazzerini, B., Jain, L.C., Dumitrescu, A. In: Evolutionary Computation. CRC Press, Florida, USA (2000)
27. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs (3rd ed.). Springer-Verlag, London, UK, UK (1996)
28. Boese, K.D.: Cost Versus Distance In the Traveling Salesman Problem. Technical report (1995)
29. Whitley, D., Hains, D., Howe, A.: Tunneling between optima: partition crossover for the traveling salesman problem. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. GECCO '09, New York, NY, USA, ACM (2009) 915–922
30. Fan, L., Zhou, Y., Feng, D.: A Fast Implementation of Computing the Transparency Order of S-Boxes. In: Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for. (2008) 206–211
31. Burnett, L.D.: Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. PhD thesis, Queensland University of Technology (2005)
32. Zhang, X., Zheng, Y.: GAC-the criterion of global avalanche characteristics of cryptographic functions. Journal of Universal Computer Science **1**(5) (1995) 316–333
33. Nyberg, K.: Perfect nonlinear s-boxes. In: Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings. Volume 547 of Lecture Notes in Computer Science., Springer (1991) 378–386

34. Forre, R.: The Strict Avalanche Criterion: Spectral Properties of Boolean Functions and an Extended Definition. In: Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings. Volume 403 of Lecture Notes in Computer Science., Springer (1988) 450–468
35. Goubin, L., Patarin, J.: DES and Differential Power Analysis (The "Duplication" Method). In: Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems. CHES '99, London, UK, UK, Springer-Verlag (1999) 158–172
36. Standaert, F.X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In Joux, A., ed.: Advances in Cryptology - EUROCRYPT 2009. Volume 5479 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 443–461
37. Syswerda, G.: Schedule optimization using genetic algorithms. In: Handbook of Genetic Algorithms. (1991) 332–349

A Appendix A

Algebraic degree deg(f) of a Boolean function f, is defined as the number of variables in the largest product term of the functions' algebraic normal form (ANF) having a non-zero coefficient [31]. Algebraic degree $deg(F)$ of (n, m)-function F is the minimum value of all non-zero linear combinations of the coordinate functions of F [9]:

$$deg(F) = max_{\bar{b} \in \mathbb{F}_2^m} deg(\bar{b} \cdot F) \tag{5}$$

Global avalanche criterion [32] consists of absolute indicator and sum-of-square indicator. Both criteria of (n, m)-function F equals maximum value of all non-zero linear combinations of the coordinate functions of F [31]. Absolute indicator equals:

$$\Delta_F = max_{\bar{b} \in \mathbb{F}_2^{m*}} |r(\bar{b} \cdot F)| \tag{6}$$

where $r(\bar{b} \cdot F)$ is autocorrelation spectra.

Sum-of-square indicator equals:

$$\sigma_F = \sum_{\bar{b} \in \mathbb{F}_2^m} r(\bar{b} \cdot F)^2 \tag{7}$$

Function F satisfies strict avalanche criterion [9,31,34] if all linear combinations of the coordinate functions f_i satisfies SAC.

Function F is correlation immune of order t - CI(t) if all linear combinations of coordinate functions are correlation immune of order t. Correlation immunity equals

$$W_F(\bar{a}, \bar{v}) = 0, \text{ for } 0 \leq HW(\bar{a}) \leq t \tag{8}$$

DPA Signal-to-Noise Ratio is alongside transparency order, second property that characterize resilience of S-boxes to DPA attack [23].

$$SNR(DPA)(F) = n \cdot 2^{2 \cdot m} \left(\sum_{\bar{k}} \left(\sum_{i=0}^{m-1} \widehat{(-1)^{F_i}}(\bar{k}) \right) \right) \tag{9}$$

where $\widehat{f}(\bar{k})$ equals

$$\widehat{f}(\bar{k}) = \sum_{\bar{x}} (-1)^{\bar{x}\cdot\bar{k}} f(\bar{x}) \tag{10}$$

Differential delta uniformity δ represents the largest value in the difference distribution table without counting the value 2^n in the first row [2, 22, 33].

B Appendix B

Insert Mutation. In this mutation operator, two positions inside an individual are randomly chosen. Then one of those values is moved to the adjacent position of the other value. Other values are shuffled to make the room for the moved value [24].

Inverse Mutation. In inverse mutation, two position are randomly chosen and then all the values between those two positions are shuffled to be in reverse order [24].

Toggle Mutation. In toggle mutation, two positions are randomly chosen and the values on those positions are exchanged.

PMX Crossover. First, two crossover positions are chosen randomly, and the segment between them from the first parent is copied to the offspring. Then, starting from the first crossover position check elements in that segment of second parent that have not been copied. For each of those elements i, check the offspring to see what elements j has been copied in its place from first parent.Place those values i into the positions occupied j in parent 2. If the place occupied by j in parent 2 has already been occupied in the offspring by an element k, put i in the position occupied by k in parent 2. After all the elements in crossover segment are finished, the rest of the offspring is filled from parent 2 [24].

PBX Crossover. In this operator first the values in random positions from the first parent are copied to the same positions in the offspring. Next, values from the second parent that are not present in the offspring are copied to it starting from the beginning of the offspring [37].

OX Crossover. Two crossover positions are chosen at random, and the segment between those positions is copied from the first parent to the offspring. Starting from the second crossover point in the second parent, copy unused values to the offspring in the order they appear in the second parent, wrapping around at the end of te list [24].

C Appendix C

S-box 1 = (34, 8b, f6, 7f, d, 2e, d3, e4, 6f, 59, 10, 97, 3e, 3f, f2, 53, 43, 7b, 57, 86, 18, dd, c4, 50, 9, 63, 7a, 5c, 5f, 21, 66, 68, 99, f4, 14 , 4a, 60, 4d, ea, 85, cf, ca, 17, fc, eb, 8, bb, e5, b1, 5b, 9e, 40, ef, 48, 1e, aa, 51, d6, 1f, b, 75, 49, a9, 2d, f, 41, 67, 82, 8d, 1d, 55, de, ad, 46, c6, f9, a7, 26, 4, 3c, 22, 6c, 47, e7, c2, a8, cd, 3a, d0, c5, e1, c, 3d, c1, 7, 65, 2b, e2, 19, e0, ac, 3, c3, da, 6b, 0, bf, b2, be, ce, 83, 91, 28, d5, 9d, 52, e, cc, 4f, 80, 64, c8, a, 15, d8, 32, 8a, 54, 25, b6, 2f, 69, 20, 6a, bd, ab, 90, 12, a6, 4e, a4, 7d, 79, 81, 29, d4, b0, df, 6d, b8, 24, b5, 74, a5, 1, e3, cb, e8, 87, 1a, 9f, 8e, b7, f5, 39, 9c, 16, b4, d2, 31, 96, 92, 9a, ec, 35, 8c, 36, 4b, 98, c0, af, ba, d1, 61, 89, 38, ed, 5e, 13, 4c, d7, f7, f0, d9, db, 77, 33, 73, 94, a3, fe, 78, e9, 56, 9b, f1, a0, 42, 30, 2c, f3, dc, 8f, 37, 72, 95, 7e, 5d, 70, fb, bc, 1c, 45, c7, 5a, 1b, c9, e6, 2, 44, 3b, 62, f8, b9, 7c, 76, 58, ee, 5, 6, b3, 11, 6e, 27, fd, fa, 71, 84, a1, a2, 23, 2a, 88, 93, ff, ae)

S-box 2 = (61, de, f8, 1a, 30, 0, a6, 9f, 25, 7b, 88, 16 , 34, 7e, 6b, 1d, 9c, 6e, c5, 5f, 73, 49, 65, bf, dc, 36, 6d, 64, f6, d, 11, ca, f5, 5, 1f, cc, 7d, e3, 82, fc, 5e, 20, d3, bd, cd, 42, 45, ba, f7, af, 52, 80, c, b9, be, 2, e, cf, 76, 91, 2a, 6c, c4, 2b, 68, c6, d2, 1b, c0, 5c, a4, 7f, f0, eb, 2c, 8c, a8, 3b, 4b, 5a, 12, 48, e0, 27, 8d, 74, c3, 63, b5, 39, c2, 46, df, f4, 8a, 40, 26, 32, 43, 67, 47, 99, 71, 8, 3, 70, c7, 31, 66, 4c, 55, 23, 56, 3f, 6, 7a, bc, 33, 4a, d1, 5d, 44, b, f2, 59, 41, ef, b8, 72, 35, 92, fd, ed, 93, f, a0, 78, 8e, d7, 9d, 4d, c1, 97, b1, d5, a9, 9a, 84, 15, fa, fb, f1, 53, c9, 57, 24, 19, 60, 9e, a7, a1, c8, 94, 3a, 6a, 18, 83, e7, a5, ff, e9, 62, 7, 1c, ea, 4e, 38, a, ad, b6, 22, 2d, e6, 7c, b3, bb, 96, d4, e4, ab, 69, dd, 87, a2, 8b, 10, 86, ae, 6f, 9, cb, e2, ec, b0, 50, ac, fe, 1, d9, a3, 58, b2, d6, 21, db, e8, ee, e1, 75, 3e, da, 81, d8, b4, 79, 4, 13, 1e, 9b, 51, 95, 77, b7, 5b, f9, 3c, 90, 2e, 2f, 54, 28, d0, e5, 17, 89, 4f, 98, 37, 85, 3d, aa, ce, 8f, f3, 29, 14)

S-box 3 = (c5, 69, f4, be, 20, e, f2, 6d, e9, 15, 98, ca, f8, 96, a1, 64, a4, 24, e1, 59, fa, 5f, c1, 8e, 90, a6, 73, 38, 92, 1b, 47, 9f, 35, d3, 52, ef, 54, ab, 75, b1, 22, 93, 9c, c7, e0, 4a, fc, cb, 88, c4, 31, f3, 11, 4b, 34, 6f, 74, 23, 40, 3e, 3a, eb, c, 8b, 18, cf, 3b, ae, 61, f0, a8, 7, 10, fd, 5a, f5, 65, ee, 84, 6b, 32, 26, 82, e7, 14, 4d, 30, 4f, 80, a2, a0, 68, 78, d5, bd, ff, 85, 7c, d1, 3d, b3, 7e, ac, c8, 5, 8d, a, bb, 72, 5d, 87, 9e, 42, 2d, 62, 1d, 6c, b, 66, ba, f1, f6, 21, cc, c0, 5e, 91, 5b, 50, 9b, 86, e2, 3f, e3, 4e, 89, 28, 83, b2, c6, 79, 9, b9, de, 9d, 51, 1a, 7d, a9, e6, 39, 77, 7f, da, 97, ed, 2e, e5, 2f, 1, a7, 8c, 76, 6a, 43, 95, 36, dc, d7, 58, b5, db, fb, df, a3, 5c, 7b, d0, e8, 94, a5, d6, f9, 12, 17, 19, bf, 2c, 6, cd, f7, 63, ea, 2, 25, 8, af, 44, 71, dd, 53, b7, 46, 2a, 55, d, 57, 99, 6e, ad, 7a, 13, 27, e4, b6, 0, d9, 48, c9, 1f, b4, fe, 49, 4, b0, 45, 3c, ce, 1c, c3, b8, c2, ec, 67, f, 8f, 1e, 8a, 70, 81, 16, 33, 56, 3, 37, aa, 4c, 41, d8, 9a, 2b, d2, 29, 60, bc, d4)

S-box 4 = (4b, 3a, 21, 85, b2, 18, 6e, fd, 69, e5, 5b, 27, ff, 39, 62, 8a, da, 50, 75, de, 4, 3c, e0, a4, cd, 9e, fc, 9f, f9, e7, 5c, 63, 52, 48, 51, 31, a7, 3b, 5a, 90, a8, fa, b3, 73, ee, 67, 4d, ce, 16, 49, 15, d0, 2, be, 99, 94, ac, ec, 3d, a5, 9d, a3, b8, 92, 98, 19, 71, 5f, 1, 1e, 8e, e4, a6, e3, f2, 30, e9, c0, 76, 46, f, 83, b4, 6c, 7, 12, d2, 38, f1, d5, c, 4f, fb, a1, 2e, bf, 97, e, 86, b5, 82, 5e, cc, e6, aa, 81, b7, d, 79, 43, 58, c4, 6a, 96, 84, 7b, 1c, 9b, e1, 95, e8, ea, 7a, 37, f5, 44, 3, 33, bb, 66, 3f, f3, 64, 6f, 6d, ab, 8, ca, 8f, 2d, c7, 14, 40, ba, ad, ed, 25, 47, 7f, 87, dd, f0, 54, 4c, 7c, a9, d6, b6, a, cb, d1, c5, 2b, 65, af, 7d, fe, a2, 9, c6, 53, 6b, f8, 70, 42, 5d, db, 88, f4, ef, 20, f7, eb, 5, c3, d8, 7e, b1, 13, 17, 2c, 2f, d3, 8b, 23, 59, bc, 57, 9a, c8, 26, c1, 41, b0, d9, d4, dc, 3e, cf, 68, 10, f6, 36, 61, 91, 80, 78, 34, 1a, 4e, bd, 74, 6, b9, c2, 8c, 24, 89, 45, 29, 1b, 56, 0, c9, 32, 35, 2a, 55, 9c, 4a, 8d, 60, 1d, 1f, a0, 93, e2, 22, b, df, d7, ae, 77, 72, 28, 11)

S-box 5 = (db, 59, a9, f7, c0, fb, d, 85, b5, 15, 9c, 26, 21, 77, 34, 1a, 56, 2d, 7b, e4, 81, 63, 53, 44, 1f, 16, 4e, 38, 66, c2, 0, b4, ab, d5, 90, f5, 4b, 8d, 5f, ed, 76, a2, 24, d2, 5b, f6, 7d, b3, 9e, d7, 8f, c, 47, 6f, b7, ef, a0, ba, e8, e, cd, 30, 19, a6, 52, a8, 9b, 3a, 50, 6a, 32, 3d, 8, 1d, ee, b1, a1, f1, 45, 9d, 18, 2a, 12, f0, 29, eb, 33, 40, cb, 49, f9, 60, 69, cf, dd, 5d, 1e, a7, 1b, 78, 93, a5, 5e, e1, 4d, 8b, 5c, 91, ff, aa, 6b, bd, 92, e6, 39, 6, 2f, 7, 73, 2c, 68, f3, 8c, df, dc, fd, 48, 27, d1, bc, 97, 6e, 84, 67, f, c8, 55, 82, 79, b2, 64, bb, b8, 13, 8a, 4c, c7, f4, a3, c9, 86, e7, 88, 7a, 46, 42, a4, 51, 7c, d6, ce, ca, b9, 7f, d9, f8, d0, 4f, 22, 11, 28, b6, ae, 2b, 57, 61, 94, d4, 5, a, 31, b, e2, ac, 70, 17, ad, 3b, fc, 36, 35, 14, 10, bf, fa, 3e, e9, 74, 83, 8e, ec, 98, af, 1, e5, 3, 9, 87, be, e0, 95, 25, 23, 6d, 96, 72, d8, 89, 6c, c6, 2e, c3, cc, 54, 2, 9a, b0, 7e, 9f, 62, 71, 75, 3c, 41, 1c, da, ea, 99, e3, c5, f2, 80, 20, fe, 37, 3f, 5a, 4a, c4, 43, c1, d3, 58, de, 65, 4)