S-box, SET, Match: A Toolbox for S-box Analysis

Stjepan Picek, Lejla Batina, Domagoj Jakobović, Bariş Ege, Marin Golub

S-box, SET, Match: A Toolbox for S-box Analysis

Stjepan Picek[1,2], Lejla Batina[1], Domagoj Jakobović[2], Barış Ege[1] and Marin Golub[2]

[1] Radboud University Nijmegen, Institute for Computing and Information Sciences (ICIS), Postbus 9010, 6500 GL Nijmegen, The Netherlands
{s.picek, lejla, b.ege}@cs.ru.nl
[2] Faculty of Electrical Engineering and Computing, University of Zagreb Unska 3, Zagreb, Croatia
{domagoj.jakobovic, marin.golub}@fer.hr

Abstract. Boolean functions and substitution boxes (S-boxes) represent the only nonlinear part in many algorithms and therefore play the crucial role in their security. Despite the fact that some algorithms today reuse theoretically secure and carefully constructed S-boxes, there is a clear need for a tool that can analyze security properties of S-boxes and hence the corresponding primitives. This need is especially evident in the scenarios where the goal is to create new S-boxes. Even in the cases when some common properties of S-boxes are known, we believe it is prudent to exhaustively investigate all possible sets of cryptographic properties. In this paper we present a tool for the evaluation of Boolean functions and S-boxes suitable for cryptography.

Keywords: Private-key Cryptography, Boolean functions, S-boxes, Cryptographic Properties

1 Introduction

Boolean functions and S-boxes play important role in a number of stream and block algorithms as the only nonlinear elements. As such, if poorly chosen, Boolean functions or S-boxes can undermine the security of the whole algorithm. To be able to assess the quality of those nonlinear elements a plethora od cryptographic properties was devised over the years. Boolean functions are typically used when the output of the building blocks is one-dimensional, and S-boxes are used in the cases when the output is multi-dimensional.

There are various ways to analyze the security of a block cipher. Besides more traditional linear [1] and differential cryptanalysis [2], the most practical attacks today belong to side-channel analysis (SCA) targeting implementations of cryptography in software and hardware.

It can be concluded that nonlinear elements and their properties are of utmost importance for the security of a cryptographic algorithm as a whole. A natural

question that arises is how to analyze those elements. One can question the need for such analysis since there are nonlinear elements in the literature with good properties. However, it is important to have a tool that can be used for the evaluation of a wider set of properties since in the time of the creation of an S-box not all properties may have been necessarily developed. Second important viewpoint is that researchers sometimes want to develop proprietary S-boxes, or S-boxes with similar functionality but of other sizes than those used before. Although it is not too difficult to check a set of main properties of S-boxes, checking several of those properties can be demanding. It is worth mentioning that many of those cryptographic properties are conflicting, so even the choice of properties that are of interest can pose a problem. In these situations we envision the need for a reliable evaluation tool as such a tool should be one of the essential parts in the evaluation process.

1.1 Related Tools

The lack of publicly available tools that can evaluate S-boxes is somewhat surprising, but this is not due to the fact that those tools would be too difficult to develop. Actually, the most demanding part of the work is to find as many relevant properties as possible. Indeed, every researcher that is interested in S-boxes uses either his own code alone or in combination with some publicly available tools. The main problem is that most of those tools are not publicly available and therefore they are not accessible to wider community. Here we mention tools that can be used to evaluate Boolean functions or S-boxes and that are publicly available.

Boolfun package in R. R is a free software environment for statistical computing and graphics [3]. It works on various UNIX, Windows and Mac OS platforms. Although the default version of R does not have a support for the evaluation of the Boolean functions, it is possible to load a package named $boolfun$ that provides functionalities related to the cryptographic analysis of Boolean functions [4, 5].

Boolean functions in Sage. Sage is a free open-source mathematics software [6]. In Sage there is a module called $BooleanFunctions$ that allows one to study cryptographic properties of Boolean functions. This tool can evaluate most of the relevant cryptographic properties (connected with linear and differential properties) of Boolean functions.

S-boxes in Sage. There is a module called $Sbox$ that allows the algebraic treatment of S-boxes. This module has many options but when considering cryptographic properties it is only possible to calculate difference distribution table and linear approximation matrix.

VBF library. For the sake of completeness we also list VBF (Vector Boolean Functions) library. Alverez-Cubero and Zufiria presented their tool for analyzing vectorial Boolean functions from cryptographic perspective that possibly could calculate various properties of S-boxes [7]. [1]

[1] We write possibly since although this library should be publicly available, we could not find it anywhere for download.

1.2 Our Contribution

In this paper we present our software SET (S-box Evaluation Tool) that can be used to evaluate Boolean functions and S-boxes. SET can be used to analyze a wide range of properties of Boolean functions or S-boxes that are relevant for the cryptographic assessment. Naturally, this set of properties is not complete, and we plan to add new properties during the further development. Main contribution of this paper is in providing the tool for the analysis of a wide set of cryptographic properties of S-boxes or Boolean functions. In fact, as far as we know, this tool deals with the largest set of cryptographic properties that is publicly known. Furthermore, since the code is available under the GNU General Public License it is easy for other researchers to add or change the existing code.

The remainder of this paper is organized as follows: In Section 2 we give short introduction to representations and cryptographic properties of Boolean functions and S-boxes related with SET tool. In Section 3 we give the details of our tool. Finally, in Section 4 we conclude the paper.

2 Representations and Cryptographic Properties

In this section we give necessary information about the representations and cryptographic properties of Boolean functions and S-boxes. Due to the lack of space, we do not give formulas or explain the role of each property in the security of an algorithm, but rather refer to the additional literature where those information are available. When trying to classify all properties into several groups we could follow different avenues. One classification could be made on the relation to a type of cryptanalysis - properties related to differential cryptanalysis, linear cryptanalysis, algebraic/cube cryptanalysis or side-channel attacks. Another classification could be related to the properties that reflect propagations of differences induced by S-box and to properties that reflect algebraic structures of S-boxes [8]. However, we decided to classify properties on the basis of the representation we use to calculate the property.

An S-box ((n, m)-function) is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. If m equals 1 then the function is called a Boolean function. Boolean functions $f_i, i \in \{1, ..., m\}$ are coordinate functions of F where every Boolean function has n variables.

2.1 Representations

A Boolean function f on $\mathbb{F}_2^n$ can be uniquely represented by a **truth table** (TT), which is a vector $(f(\mathbf{0}), ..., f(\mathbf{1}))$ that contains the function values of f, ordered lexicographically [9]. **Polarity truth table** (PTT) is a vector containing functions values of $(-1)^f$ [9]. **Walsh transform** is a second unique representation of a Boolean function that measures the similarity between $f(\boldsymbol{x})$ and the linear function $\boldsymbol{a} \cdot \boldsymbol{x}$ [4,9]. The inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{x}$ is denoted as $\boldsymbol{a} \cdot \boldsymbol{x}$ and equals $\boldsymbol{a} \cdot \boldsymbol{x} = \oplus_{i=1}^{n} a_i x_i$ where " $\oplus$ " is addition modulo 2.

Third unique representation of an Boolean function f on $\mathbb{F}_2^n$ is by means of a polynomial in $\mathbb{F}_2[x_0, ..., x_{n-1}]/(x_0^2 - x_0, ..., x_{n-1}^2 - x_{n-1})$. This form is called **algebraic normal form** (ANF) [9].

Autocorrelation function of a Boolean function f on $\mathbb{F}_2^n$ is real-valued function that does not uniquely determine a Boolean function [9].

Truth table of an (n, m)-function F equals the concatenation of truth tables of all coordinate functions. Walsh transform [10], algebraic normal form [10] and autocorrelation function [9] of an (n, m)-function F is the collection of all respective values of the component functions of F.

2.2 Cryptographic Properties

Next we enumerate cryptographic properties of Boolean functions and S-boxes that can be calculated with SET. With each of the properties we list the references where an interested reader can find definitions and formulas. First block of citations refers to Boolean functions and second one to S-boxes. Informally, this list can be also regarded as the dependency tree where one needs to obtain respective representation before the property listed in the group can be calculated.

Truth table based.

- Algebraic immunity [9] [10].
- DPA Signal-to-noise ration SNR (DPA) (only for S-boxes) [11].

Walsh transform based.

- Balancedness [9] [12].
- Nonlinearity [9] [13].
- Bias of nonlinearity (only for Boolean functions) [9].
- Correlation immunity [9] [14].
- Resilience [4] [14].

Algebraic normal form based.

- Algebraic degree [14] [8, 9].

Autocorrelation function based.

- Global avalanche criterion (GAC) [9,15] [14,16]. GAC property consists of absolute indicator and sum-of-square indicator.

Lookup table based.

- Propagation characteristics [17] [9].
- Difference distribution table DDT (only for S-boxes) [9, 18].
- Linear approximation table LAT (only for S-boxes) [9].
- Differential delta uniformity (only for S-boxes) [10, 19].
- Robustness to differential cryptography (only for S-boxes) [2, 20].
- Fixed points and opposite fixed points (only for S-boxes) [21].
- Branch number (only for S-boxes) [21, 22].
- Transparency order [12, 23].

These properties and corresponding functions represent only a part of available functions in SET. Full list is available in extended version of the paper published in IACR database as well as in website containing the source code of the tool (`http://sidesproject.wordpress.com/`).

3 SET Tool

SET (S-box Evaluation Tool) is a tool for the analysis of cryptographic properties of Boolean functions and S-boxes that is written in ANSI C code. All the properties mentioned in Section 2 can be evaluated with SET. The tool supports arbitrary input and output sizes where those sizes can differ. In the process of the development of the tool there are some challenges that had to be addressed. First and the most important one was which properties to include. We tried to add the properties that cover the most important cryptanalysis types. Therefore, we added the properties related with differential, linear, algebraic/cube cryptanalysis as well as the properties related with side-channel attacks. Some of those properties are not often used but we consider important to be able to collect as much data on S-boxes or Boolean functions as possible. Next, it was necessary to decide on the format of the tool, i.e. using a command line application, application with a GUI, or a C library. There was also an option to include it in some other tool that have support for S-boxes or Boolean functions, like Sage or R.

Currently, the program comes in two versions: first one is a stand-alone command line tool with a fixed user interface and the second version is a C library. However, since the source code is also available, users are able to customize the program for their needs. SET is available on SIDES project website: `http://sidesproject.wordpress.com/`.

We give description here of the stand-alone program since it encompasses all the available functionalities. In the stand-alone tool user cannot choose which properties will be calculated, but instead all the properties are calculated. It is possible to choose whether to display the results to the screen, file or both. When the program starts, a user needs to enter the basic parameters: input and output sizes and a file name. There is an option to start the program with the command line arguments (input size, output size and file name) so it can be used in a script. If the program is called with command line arguments then all results are stored to text files and are not displayed on the screen.

A file that contains a Boolean function must be defined in the truth table form and binary format. For the S-boxes case, text file must contain a lookup table of decimal or hexadecimal values with a tabular delimiter. The program can recognize between those two formats so the user does not need to provide any additional information about the format. When the program saves data to the file, the name is a concatenation of word "stats_" and input file name. Since various representations of S-box can grow large rather fast, program writes in separate file for each representation - Walsh transform, autocorrelation function, algebraic normal form and truth table, as well as for difference distribution table DDT and linear approximation table LAT. Files naming convention is "walsh_", "ac_", "anf_", "tt_", "ddt_", "lat_" + input name of the file, respectively. Program can also output S-box coordinate functions values for each property.

The code is written with the first objective being that each of the properties or representations can be calculated separately. Although one can expect that the performance is the key objective of the tool, in our opinion the option to calculate

wide set of properties is of even greater importance. Performance becomes more important in the case when one needs to calculate the properties of big S-box (e.g. 16×16) or when one uses the program as a script to go through a number of S-boxes. To improve the speed of tool execution, all small functions that are often called are inlined. Furthermore, since Hamming weight is often used, instead of calling it every time we call it at the beginning and store results in lookup table for faster execution. To aid researchers in examining to code or adding new functionalities, every function is commented. Additionally, as a part of the source code package there is an extensive documentation about all functions, logic behind them and instructions on how to use them.

3.1 Time and Memory Complexity

Once the input file is loaded and transformed in the truth table form, the evaluation process starts. Most of the properties are calculated through the coefficients of the Walsh spectrum, autocorrelation spectrum and algebraic normal form. Since the formulas are the same for Boolean functions and S-boxes (only difference is that the calculation is repeated for every linear combination of the coordinate Boolean functions in the case of an S-box) we write here only complexities for Boolean function case. Computational complexity for calculating the algebraic normal form and Walsh spectrum is of order $\mathrm{O}\left(n \cdot 2^n\right)$. We use Fast Walsh transform when calculating Walsh spectrum. Autocorrelation function has computational complexity $\mathrm{O}\left(2^{2 \cdot n}\right)$.

3.2 Program Code Examples

In this section we first show some source code of a small program and after that we show the output of that program.

```
#include <sat.h>
int main (int argc, char *argv[]){
prepare (name, argc, argv);
truth_table (tt);
linear_combinations (tt, ll);
walsh_transform (ll, wt);
is_balanced (wt);
printf ("Nonlinearity is %d\n", nonlinearity(wt));
printf ("Correlation immunity is %d\n", correlation_immunity(wt));
printf ("Transparency order is %f\n", get_transparency_order());
printf ("Delta uniformity is %d\n", get_delta_uniformity());
free_all();
}
```

When writing the program, first it is necessary to call the function *prepare* $(name, argc, argv)$ where this function sets the main variables and lookup table values for Hamming weight. After that, we simply call functions for the properties we want to calculate. At the end, function $free\_all()$ is called to free dynamically allocated arrays and matrices.

If we call this program with the AES S-box input and output sizes and the name of the file where the lookup table is, i.e.

```
SAT.exe 8 8 c:/AES.txt
```

as the output we get

```
                        SET - S-box Evaluation Toolbox
Name of the file: c:/AES.txt
Input size M is 8.
Output size N is 8.
S-box is balanced.
Nonlinearity is 112.
Correlation immunity is 0.
Transparency order is 7.860.
Delta uniformity is 4.
```

3.3 Results and Speed of Execution

In Table 1 we give a few examples for different sizes of Boolean functions and S-boxes and their execution times in milliseconds (parameter $Time$). Fields In and Out represent the number of input and output variables of S-boxes or Boolean functions and field δ represents differential uniformity. We display only a subset of available properties, but field $Time$ represents total execution time (when calculating all available properties) in miliseconds. The calculations are done on a system with Intel i5 3230M processor and 4 Gb of RAM. The tests are conducted with operating systems Debian 3.13 and Windows versions 7 and 8. It is possible to analyze larger Boolean functions or S-boxes, but the execution speed significantly goes down as expected.

Table 1. SET execution examples

Function	In	Out	Nonlinearity	Degree	δ	GAC	Time [ms]
Boolean 1	8	1	112	7	N/A	32, 133120	0.01
Boolean 2	10	1	456	9	N/A	168, 3182848	0.55
Boolean 3	12	1	1942	11	N/A	312, 48707584	450
PRESENT [24]	4	4	4	3	4	16, 1024	1
DES 3 [8]	6	4	16	5	8	48, 24064	1
AES [21]	8	8	112	7	4	32, 133120	650

It is hard to compare the execution speed of the SET tool with related tools since we offer more functionalities and therefore total time of execution for our program can be longer, depending on the number of properties that are tested. Nevertheless, in Table 2 we give execution times for some of the functions when calculated with SET and other some tools mentioned in Section 1.1. The test

environment is the same as the one mentioned previously (only Debian operating system) and the times are given for the 8×1 Boolean function (as used in Rakaposhi algorithm [25]) and for 8×8 S-box (AES algorithm). We emphasize that this comparison should not be regarded as in-depth analysis of the performances ot the tools, but rather as a indication of their respective execution times. In the analysis we do not take into account specificities of each tool, e.g. precomputed values stored in memory but rather we are interested only in total time from the call of each function to the display of corresponding result. In accordance with that, presented times should serve only as a guideline.

For R, we use *rbenchmark* library and function *benchmark* [26]. For Sage, we use *timeit* function and for SET we use function *clock_gettime*. Execution times represent average times over 100 runs in microseconds. Additionally, we give necessary input arguments for functions when using SET tool.

Table 2. SET functions

Name	Argument	R	Sage		SET	
			Boolean	S-box	Boolean	S-box
walsh_transform	truth table	50	0.69	N/A	1.17	2720
autocorrelation	truth table	N/A	0.8	N/A	1.3	2980
algebraic_normal	truth table	50	3160	N/A	10.02	4501
LAT	lookup table	N/A	N/A	75.6E3	N/A	1.1E5
DDT	lookup table	N/A	N/A	434E3	N/A	441
nonlinearity	walsh spectrum	70	3.31	N/A	0.66	199
correlation_immunity	walsh spectrum	40	2.5	N/A	6.6	1768
absolute_indicator	autocorrelation	N/A	27.4	N/A	0.64	192
sum_of_square_indicator	autocorrelation	N/A	2.6	N/A	0.82	259
algebraic_degree	alg. normal form	40	N/A	N/A	0.62	166
algebraic_immunity	truth table	40	2E5	N/A	12	5.2E5
propagation_characteristics	lookup table	N/A	N/A	N/A	1.58	1.58
num_fixed_points	lookup table	N/A	N/A	N/A	N/A	0.6
num_opposite_fixed_points	lookup table	N/A	N/A	N/A	N/A	0.68
snr_dpa	truth table	N/A	N/A	N/A	N/A	4019
branch_number	lookup table	N/A	N/A	N/A	N/A	277
transparency_order	lookup table	N/A	N/A	N/A	1218.7	6105

3.4 Further Work

There are several avenues that we plan to investigate in the further development of SET. First one is to add more relevant cryptographic properties (e.g. basic AI and graph AI properties of S-boxes, k-normality of Boolean functions). The following would be to make further improvements in the execution speed of the code where the goal is to be able to efficiently work with the S-boxes of sizes up to 16×16. We are also considering to make a module for Sage where we would include all the functionalities of our program.

4 Conclusion

S-boxes or Boolean functions represent important part of many algorithms. Often, some well researched and widely known S-boxes are reused (or at least the idea behind them is reused - as in the case in Rakaposhi Boolean function that uses the same irreducible polynomial as AES [25]). However, often there is a need to use new, previously not considered S-boxes or Boolean functions. Sometimes that is just due to new required sizes (as in lightweight cryptography) and sometimes it is due to the aim of having a proprietary algorithm. In any case, we consider this tool as a valuable asset to cryptographic researchers.

Acknowledgements

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.

The authors would like to thank Antonio De La Piedra for his help with the speed performance tests of different tools.

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. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. (2013) ISBN 3-900051-07-0.
4. Lafitte, F.: The boolfun Package: Cryptographic Properties of Boolean Functions. (2013)
5. Lafitte, F., Heule, D.V., Hamme, J.V.: Cryptographic Boolean Functions with R. The R Journal **3**(1) (jun 2011) 44–47
6. Stein, W.A., et al.: Sage Mathematics Software (Version 5.10). The Sage Development Team. (2013) `http://www.sagemath.org`.
7. Alvarez-Cubero, J., Zufiria, P.: A c++ class for analysing vector boolean functions from a cryptographic perspective. In: Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on. (July 2010) 1–9
8. Knudsen, L.R., Robshaw, M.: The Block Cipher Companion. Information Security and Cryptography. Springer (2011)
9. Braeken, A.: Cryptographic Properties of Boolean Functions and S-Boxes. PhD thesis, Katholieke Universiteit Leuven (2006)
10. 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)

11. Guilley, S., Pacalet, R.: Differential Power Analysis Model and Some Results. In: In proceedings of CARDIS 2004, Kluwer Academic Publishers (2004) 127–142
12. 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
13. 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
14. Burnett, L.D.: Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. PhD thesis, Queensland University of Technology (2005)
15. Zhang, X., Zheng, Y.: GAC-the criterion of global avalanche characteristics of cryptographic functions. Journal of Universal Computer Science **1**(5) (1995) 316–333
16. 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
17. Preneel, B., Van Leekwijck, W., Van Linden, L., Govaerts, R., Vandewalle, J.: Propagation characteristics of Boolean functions. In: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology. EUROCRYPT '90, New York, NY, USA, Springer-Verlag New York, Inc. (1991) 161–173
18. Heys, H.M.: A Tutorial on Linear and Differential Cryptanalysis. Technical report (2001)
19. 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
20. Seberry, J., Zhang, X.M., Zheng, Y.: Systematic Generation of Cryptographically Robust S-boxes (Extended Abstract) . In: In Proceedings of the First ACM Conference on Computer and Communications Security. (1993) 172–182
21. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2002)
22. Saarinen, M.J.: Cryptographic Analysis of All 4 x 4-Bit S-Boxes. In Miri, A., Vaudenay, S., eds.: Selected Areas in Cryptography. Volume 7118 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 118–133
23. 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
24. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems. CHES '07, Berlin, Heidelberg, Springer-Verlag (2007) 450–466
25. Cid, C., Kiyomoto, S., Kurihara, J.: The RAKAPOSHI Stream Cipher. In Qing, S., Mitchell, C., Wang, G., eds.: Information and Communications Security. Volume 5927 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009) 32–46
26. Kusnierczyk, W.: rbenchmark: Benchmarking routine for R. (2012)