A Search for Additional Structure: The Case of Cryptographic S-boxes

Claude Carlet[1], Marko Djurasevic[2], Domagoj Jakobovic[2], Stjepan Picek[3]

[1] Department of Informatics, University of Bergen, Norway
and Department of Mathematics, Universities of Paris VIII and XIII, France.
claude.carlet@gmail.com

[2] Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb, Croatia
{marko.durasevic,domagoj.jakobovic}@fer.hr

[3] Cyber Security Research Group, Delft University of Technology,
Mekelweg 2, Delft, The Netherlands
s.picek@tudelft.nl

Abstract. We investigate whether it is possible to evolve cryptographically strong S-boxes that have additional constraints on their structure. We investigate two scenarios: where S-boxes additionally have a specific sum of values in rows, columns, or diagonals and the scenario where we check that the difference between the Hamming weights of inputs and outputs is minimal. The first case represents an interesting benchmark problem, while the second one has practical ramifications as such S-boxes could offer better resilience against side-channel attacks.
We explore three solution representations by using the permutation, integer, and cellular automata-based encoding. Our results show that it is possible to find S-boxes with excellent cryptographic properties (even optimal ones) and reach the required sums when representing S-box as a square matrix. On the other hand, for the most promising S-box representation based on trees and cellular automata rules, we did not succeed in finding S-boxes with small differences in the Hamming weights between the inputs and outputs, which opens an interesting future research direction. Our results for this scenario and different encodings inspired a mathematical proof that the values reached by evolutionary algorithms are the best possible ones.

1 Introduction

S-boxes are functions with an important role in cryptography as they are the only source of nonlinearity for many cryptographic algorithms. Without S-boxes with good cryptographic properties, many designs would be easy to break by running cryptanalyses [1,2]. A common option to obtain cryptographically strong S-boxes is to use algebraic constructions [3]. Still, there are alternatives to algebraic constructions. If the construction constraints are not too difficult, the random search can work well. Besides algebraic constructions and random search, various heuristic techniques showed their potential [4]. Heuristics play an important role, especially in the cases where one requires S-boxes with cryptographic properties that are not obtainable by the known algebraic constructions (note, that there are not too many known algebraic constructions [3]). We explore whether we can design cryptographically strong S-boxes (with good cryptographic properties) that have additional properties of structure. We consider two settings: obtaining cryptographic S-boxes that have an arrangement of elements that result in 1) the same sum for every row, column, or diagonal, or 2) the minimal difference between the Hamming weights of the inputs and outputs.

Finding additional structure in S-boxes while keeping very good mandatory cryptographic features is not an easy task. We do not know if there exists such structure, or even if it does, whether it occurs for various S-box sizes. Next, if we assume there is additional structure, there is no prior knowledge about how difficult it is to reach it with heuristics. What is more, we may not be able to find such solutions depending on the selection of solution encoding or fitness functions. Finally, it is not known if there are trade-offs between cryptographic properties and the properties denoting additional structure.

Exploring the S-boxes depicted as square matrices with sums of rows, columns, or diagonals equal to a specific value does not have practical cryptographic applications to the best of our knowledge. This is because what we search for is not affine invariant, contrary to many notions in cryptography [3]. Still, we consider it an interesting benchmark problem as now, for smaller S-box sizes, we can easily obtain optimal cryptographic properties with heuristics, while for larger S-boxes, heuristics cannot give results close to algebraic constructions. We consider this additional constraint an interesting bridging option to allow more fine-grained evaluation of heuristics for S-box construction. By imposing additional constraints, we enable heuristics to reduce the search space size. Note that this structure constraint requires that an S-box is also a magic square [5].

The second constraint ensuring that the Hamming weights of S-box inputs (x) and outputs ($F(x)$) are as close as possible, has more practical ramifications. S-boxes are common targets for side-channel attacks [6]. In such attacks, one needs to consider an appropriate leakage model that the cryptographic device follows. A common model is the Hamming weight model, where the power consumption is proportional to the Hamming weight of a processed value. Minimizing the difference in the Hamming weights could potentially make the S-box more resilient against side-channel attacks, as explained in more detail in Section 2.

To the best of our knowledge, there are no previous works that consider evolving cryptographically strong S-boxes with additional constraints on the arrangement of elements to result in a specific sum, or to minimize the differences in the Hamming weights. In the rest of this paper, whenever talking about S-boxes, we consider those that are cryptographically strong (details are given in Section 2). While there is not much previous work on exploring additional structure in S-boxes, there are works that use evolutionary algorithms to evolve S-boxes. For example, Clark et al. used the principles from the evolutionary design of Boolean functions to evolve S-boxes for sizes up to 8×8 [7]. Picek et al. developed an improved fitness function that enables evolutionary algorithms to find higher nonlinearity values for several S-box sizes [8]. Mariot et al. investigated the genetic programming approach to evolve cellular automata rules that are then used to generate S-boxes [9]. The results obtained with GP used to evolve CA rules outperform any other metaheuristics for sizes 5×5 up to 7×7. Jakobovic et al. presented a fitness landscape analysis for S-boxes of various sizes [10]. Finally, Picek et al. used evolutionary algorithms to find S-boxes that have good cryptographic properties but also good side-channel resilience [11].

In this paper, we explore how evolutionary algorithms can be used to construct S-boxes with additional structure. Toward that goal, we first define several S-box sizes we investigate with both single-objective and multi-objective approaches. Our results indicate that it is possible to construct S-boxes with sums of rows and columns equal to a specific value. We find a more difficult condition when adding the constraint on the sum of diagonals. We could not find any such S-box with optimal cryptographic properties but also having the sum of all rows, columns, and diagonals equal to a specific value (thus, producing S-box that is also a magic square).

Our experiments show it is possible to obtain an S-box with optimal cryptographic properties and a small difference between the Hamming weights of inputs and outputs. Then, our experimental results inspire mathematical research proving that the nonlinearity must be equal or smaller than the sum of differences of the Hamming weights. Finally, our results show that the solution encoding (tree-based) that works the best for cryptographic properties performs the worst for these additional, structure-based properties.

2 Background

Let n, m be positive integers. $\mathbb{F}_2^n$ is the n-dimensional vector space over $\mathbb{F}_2$ and by $\mathbb{F}_{2^n}$ the finite field with 2^n elements. The set of all n-tuples of elements in the field $\mathbb{F}_2$ is denoted by $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the finite field with two elements. For any set S, we denote $S \backslash \{0\}$ by S^*. The addition of elements of the finite field $\mathbb{F}_{2^n}$ is represented with "+" while the inner product of a and b equals $a_1 x_1 + \ldots + a_n x_n$.

2.1 S-boxes – Representations and Properties

An S-box (Substitution box) is a mapping F from n bits into m bits (thus, S-box is an (n, m) function). An (n, m)-function F can be defined as a vector $F = (f_1, \ldots, f_m)$, where the Boolean functions $f_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ for $i \in \{1, \ldots, m\}$ are called the coordinate functions of F. The component functions of an (n, m)-function F are all the linear combinations of the coordinate functions with non all-zero coefficients. For every n, there exists a field $\mathbb{F}_{2^n}$ of order 2^n, so we can endow the vector space $\mathbb{F}_2^n$ with the structure of that field when convenient. In Table 1, we give the best known/possible results for two relevant cryptographic properties and S-box dimensions we consider.

An (n, m)-function F is balanced if it takes every value of $\mathbb{F}_2^m$ the same number 2^{n-m} of times. If a function F is balanced, then it is a permutation (the function is bijective, i.e., $n = m$).

The Walsh-Hadamard transform of an (n, m)-function F is (see, e.g., [3]):

$$W_F(a, v) = \sum_{x \in \mathbb{F}_2^m} (-1)^{v \cdot F(x) + a \cdot x}, \; a, v \in \mathbb{F}_2^m. \quad (1)$$

The nonlinearity nl_F of an (n, m)-function F equals the minimum nonlinearity of all its component functions $v \cdot F$, where $v \in \mathbb{F}_2^{m*}$ [3,12]:

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

Let F be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$ with $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^m$. We write:

$$D_F(a, b) = \{x \in \mathbb{F}_2^n : F(x) + F(x + a) = b\}. \quad (3)$$

The entry at the position (a, b) corresponds to the cardinality of the delta difference table $D_F(a, b)$ and we write it as $\delta_F(a, b)$. The differential uniformity δ_F is then defined as [13]:

$$\delta_F = \max_{a \neq 0, b} \delta_F(a, b). \quad (4)$$

Table 1: Best known values for bijective S-boxes. For 8×8, we give the best known results while for smaller sizes, we give the optimal values. For bijective S-boxes (and in $\mathbb{F}_2$), both nonlinearity and differential uniformity can be even values only. The worst possible values are 0 for nonlinearity (i.e., the S-box is linear), and 2^n for differential uniformity.

Property	3×3	4×4	5×5	6×6	7×7	8×8
nl_F	2	4	12	24	56	112
δ_F	2	4	2	2	2	4

2.2 Side-channel Attacks

Besides various cryptanalysis techniques, another category of attacks on cryptographic targets is side-channel attacks (SCAs). In such attacks, one does not aim at the weakness of the algorithm, but on the weaknesses of implementation [6]. For instance, one could observe the power consumption of a device while running the cryptographic algorithm, and compare it with hypotheses for every possible key (or, more precisely, subkey). When a particular statistic technique (for instance, Pearson correlation) shows the best absolute value of correlation, we assume that the key hypothesis is correct, and we use it to break the target. For this to work, we need to assume the leakage model in which a device leaks, where one of the standard models is the Hamming weight model. If the input and output of the S-box have the same Hamming weights, this will result in several equally likely key hypotheses (they will have the same correlation). As such, the side-channel attack in this leakage model would become somewhat more difficult to succeed.

2.3 Magic Squares

A magic square is a $d \times d$ square grid that consists of distinct positive integer values in the range $[1, d^2]$. The sum of values in every row, column, and diagonal is equal. That sum value is called the magic constant of the magic square. There is no strict requirement on the value of the magic constant. Usually, the sum is determined as the sum of all elements occurring in the magic square and divided by the number of cells on each side (d), i.e., $d(d^2 + 1)/2$.

3 Experiments

We consider S-boxes of sizes $n \times n$ only. As a consequence, our S-boxes can be bijective, but there is no constraint on this for all encodings. The smallest S-box size we work with is 3×3, while the largest is 8×8 (the smallest and largest practically used S-box sizes). The set of experiments is divided into two groups. The first group of experiments deals with the evolution of S-boxes with a constraint that is imposed on the sums of its rows, columns, and diagonals. More precisely, we can depict an S-box as a square matrix. Then, each row, column, or diagonal in that matrix must sum up to the same value (i.e., an S-box is magic square). As an example, let us consider an S-box of size 4×4 that has 16 elements. We can depict them in a matrix of size 4×4 where we fill it with S-box values column by column. Finally, we can easily check the sum of each row, column, or diagonal in such a matrix. To ensure that it is possible to obtain such S-boxes, we place the following constraints in our experiments:

1. We consider S-boxes of even dimensions only, and more precisely, dimensions 4, 6, and 8. Recall from Section 2 that S-boxes are defined as elements of finite fields where the underlying field is the finite field with two elements, i.e., $\mathcal{F}_2$. As such, the number of elements in an S-box equals 2^n, where n is the size of an S-box. Simultaneously, the number of elements in a magic square is equal to d^2, where d is the size of the magic square. For every odd dimension, the number of elements in a magic square with size d is not the same as the number of elements in a finite field of size 2^n, regardless of the fact if $d = n$.
2. Commonly, the elements in a magic square are denoted from 1 to d^2 while the elements in an S-box with elements from 0 to $2^n - 1$. This does not represent a problem as in the finite field; all elements are calculated *modulo* 2^n, which means that 0 and 2^n (d^2) represent the same values.

The second group of experiments will focus on evolving S-boxes where the difference in the Hamming weights for every pair of S-box input and output is minimal. Unlike the previous scenario, where the experiments were constrained to only even-sized S-boxes, now, there is no such constraint. Consequently, for this set of experiments, we test S-boxes from 3×3 up to 8×8 size.

3.1 Experimental Setup

The first set of experiments consider the case in which the additional constraints are placed on the sum of elements in rows, columns, and diagonals. The second round of experiments, in addition to cryptographic properties, also places the constraint on the difference in Hamming weights between the inputs and outputs of the S-box. The experiments evaluate both the bijective and non-bijective S-boxes. For that purpose, three solution representations are applied, out of which the first two are used with a genetic algorithm (GA) and the third with genetic programming (GP):

1. The integer genotype, consisting of a vector of integer values of size 2^n and values in the range $[0, 2^n - 1]$.
2. The permutation genotype of size 2^n, where each value in $[0, 2^n - 1]$ occurs only once.
3. The tree genotype, which represents the transition function of cellular automata (CA).

In our experiments, the integer genotype is used to evolve non-bijective S-boxes, while the permutation and tree genotype are used to evolve bijective S-boxes. The integer representation is a super-set of the permutation representation, and it is possible, although very unlikely, to obtain bijective S-boxes with integer genotype. The integer genotype mutation selects a random position in the vector and assigns it a new value in the defined range. The crossover operators for integer vector include a simple one-point and two-point recombination, as well as an averaging crossover which defines all the elements of the child vector as an arithmetic mean of the corresponding values in the parent vectors. For the permutation genotype, we use three mutation operators and five crossover operators where we chose among the most common ones in practice. The mutation operators are insert mutation, inversion mutation, and swap mutation [14]. We used partially mapped crossover (PMX), position based crossover (PBX), order crossover (OX), uniform like crossover (ULX), and cyclic crossover [15]. For each new individual, an operator is selected uniformly at random between all operators within a class (both mutation and crossover).

Finally, we use the tree representation, where GP is used to evolve a suitable cellular automata function in the form of a tree. The input bits of the S-box are used as terminal nodes of the tree, where their number is equal to n. The function set consists of Boolean primitives with 1) two inputs: NOT, which inverts its argument, XOR, AND, OR, NAND, and XNOR, and 2) three inputs: IF (it takes three arguments and returns the second one if the first evaluates to *true*, and the third one otherwise). The evolved function is used as a transition that, based on the input bits of the current state, calculates the output bits which act as the next state. For details about the cellular automata approach for S-box evolution, we refer readers to [9]. The recombination operators for this representation are simple tree crossover, uniform crossover, size fair, one-point, and context preserving crossover [16] (selected at random) and subtree mutation.

Both GA and GP use a steady-state tournament algorithm with tournament size $k = 3$ (select three individuals and remove the worst one, from the remaining two make an offspring and mutate it), and with individual mutation probability of 30%. All parameters were selected after a tuning phase, where the selected combinations showed good performance. The experiments for each considered configuration were executed 30 times (independent runs). The algorithms optimize a single-objective function, which is defined as a linear combination of individual criteria, as defined in the next section. In addition to the single-objective case, the multi-objective case using the well known NSGA-II algorithm [17] was also tested. Since the results achieved in the multi-objective case were equally good or worse than in the single-objective case, these results are not further discussed in the paper.

3.2 Fitness Functions

In addition to the cryptographic properties, the fitness function has to include an additional term to ensure that the algorithm can evolve S-boxes of the desired structure. When considering structures that have the sums on row, column, and diagonal elements equal to some predefined value, we use:

$$\begin{aligned} msq = \sum_{i}^{n} \left| mc - \sum_{j}^{n} sq[i][j] \right| + \sum_{i}^{n} \left| mc - \sum_{j}^{n} sq[i][j] \right| + \\ \left| mc - \sum_{i}^{n} sq[i][i] \right| + \left| mc - \sum_{i}^{n} sq[i][n-i] \right| . \end{aligned} \tag{5}$$

Here, mc represents the magic constant, i.e., the number to which the elements in rows, columns, and diagonals need to be equal to when summed up, while sq represents a square consisting of elements of the S-box. The msq property calculates the distance for all relevant elements (rows, columns, diagonals) from the predefined constant. The previous equation is the most general version, which considers the sums on all the relevant elements of the square. Depending on the experiment, some elements in that expression will not be calculated when not all relevant elements need to adhere to the specified sum constraint. Although the constant to which the rows, columns, and diagonals need to be equal to can be freely specified, we use values to which the elements sum up in magic squares of the corresponding sizes [18]. For these constants, we know it is possible to construct a square with the requested structure, whereas choosing a random value could mean it would not be possible to obtain a square of numbers where the sum of elements equals the selected number. For the 4×4 S-box, the constant is 34. For the 6×6 S-box, it equals 260, and for the 8×8 S-box, it equals 2 056.

The fitness function to be minimized is defined as:

$$f = msq - \alpha \frac{nl_F}{nl_{best}} - \beta \frac{(\delta_{worst} - \delta_F)}{\delta_{worst}}, \tag{6}$$

where nl_{best} represents the best-known values for nonlinearity as defined in Table 1, δ_{worst} is the worst possible value for differential uniformity equal to 2^n with n representing the number of inputs in the S-box, while α and β represent scaling factors for the cryptographic properties. In the experiments, the emphasis is put on the sq property in a way that the other two properties are normalized and scaled with additional factors. This is because we want to see whether there are S-boxes that adhere to the additional structures, but still have good cryptographic properties.

Based on preliminary experiments, the values for both α and β were set to the value of 0.5, which results in the *msq* property being treated as a primary objective, while the cryptographic properties are the secondary. In this way, the algorithm will always prefer a solution with a better structure, while the other two properties will then force the algorithm to search for those solutions that have better cryptographic properties.

The second set of experiments places a constraint on the difference in the Hamming weights of the inputs and outputs of the S-box. This property is calculated as:

$$HWD = |w_H(x) - w_H(F(x))|, \tag{7}$$

where x represents an input value to the S-box, $F(x)$ is the S-box function that transforms the input, and w_H is the function returning the number of ones in the argument. HWD is simply defined as the absolute difference between the Hamming weights of the inputs in the S-box and outputs from the S-box. In this case, the fitness function which has to be minimized is:

$$f = HWD - \alpha\frac{nl_F}{nl_{best}} - \beta\frac{(\delta_{best} - \delta_F)}{\delta_{best}}. \tag{8}$$

If the HWD property would be optimized primarily by giving a smaller weight to the cryptographic properties, then the algorithms would always obtain the optimal solution for this property. This would lead to poor results for cryptographic properties, and as such, the evolved S-box would not have much use. When focusing on S-boxes that have such a structure, the primary focus is placed on evolving S-boxes with good cryptographic properties and then adjusting the solutions to conform to the desired structure. After executing some preliminary tests with different weight values, the α and β coefficients are set to 10, which we evaluated to be enough for the algorithm to focus primarily on cryptographic properties, and only then on the difference of the Hamming weights.

3.3 Results

Table 2 shows the results obtained for the experiments in which additional constraints are placed on the sum of elements in rows, columns, and diagonals. For each configuration, the S-box with the best fitness was selected, and the individual properties that constitute the fitness function are denoted. The *constrained elements* column denotes for which elements of the S-box the sum was calculated to be of the specified constant. For the size of 4×4, the algorithm obtained S-boxes that mostly have good cryptographic properties. For S-boxes in which the sum in rows or columns had to be equal to a certain sum, it was almost trivial for the algorithm to find the square with the optimal cryptographic properties. Still, when it is enforced that the sum on more elements (e.g., rows and columns) has to be equal to the desired constant, we found optimal solutions for the permutation encoding only. Interestingly, although the tree representation obtained S-boxes with optimal cryptographic properties, it was unable to obtain S-boxes that adhered to any of the additionally placed constraints.

For S-boxes of size 6×6, the algorithm demonstrated an interesting behavior. When the permutation and integer genotypes were applied, the algorithm had no problem with optimizing the additional constraints, since it evolved S-boxes that satisfied these constraints in all cases. On the other hand, in any of the experiments, were we able to obtain an S-box with optimal cryptographic properties. It is not surprising that the algorithm could not reach optimal cryptographic properties since this is a difficult task for EA when using this type of encoding. Additionally, the integer genotype achieved inferior results compared to the permutation genotype. The cellular automata rules evolved by GP demonstrated the opposite behavior since there, we found S-boxes with optimal cryptographic properties, but which did not satisfy the additional constraints.

For the 8×8 S-boxes, the algorithm exhibited difficulties in obtaining S-boxes with an additional structure. For the permutation genotype, the algorithm evolved S-boxes that satisfy only the most simple constraint in which either the row or column sums are equal to the desired constant. For the integer genotype, the algorithm had fewer problems and obtained the desired structure for each constraint, which is expected since, in that case, it is much easier to construct an S-box with the desired structure. The S-boxes constructed by the permutation genotype have better cryptographic properties than those constructed by the integer genotype. The obtained objects have relatively good cryptographic properties, which are in line with the results for EA and S-boxes of that size [8]. The results obtained by the GP evolved CA are quite poor since this representation was neither able to evolve S-boxes with good cryptographic properties, nor which satisfied the constraints that were additionally placed upon the S-boxes for its structure.

It is not always easy to discern what are strong cryptographic properties as that depends on the whole cipher and not only the S-box part. For all tested sizes, it is possible to find S-boxes whose structure adheres to certain constraints. For the S-box size of 4×4, GA obtained S-boxes both with the desired structure and optimal cryptographic properties. This was not possible in GP's case since it was not possible to find an optimal S-box even for the least restrictive structure constraint. The cryptographic properties are further away from the optimal values for larger sizes, but they are still relatively good. For 6×6 and 8×8 and permutation encoding, cryptographic properties are the same for the most restrictive structure in which the elements in rows, columns, and diagonals have to sum up to a certain value, and in the structure where this is not required for the diagonal elements.

Table 3 presents the results obtained when, in addition to optimizing the cryptographic properties, the difference in the Hamming weights is used as an additional constraint. For the S-box size of 3×3, the algorithm obtained the same value for the integer and permutation genotypes. By additionally using exhaustive search, it was also proven that this is the optimal solution that can be

Table 2: Best results obtained with additional constraints imposed on the sums of rows, columns, and diagonals, single-objective optimization.

S-box size	Genotype	Constrained elements	msq	nl_F	δ_F
4×4	Integer	Rows	0	4	2
4×4	Integer	Columns	0	4	4
4×4	Integer	Rows and columns	0	3	4
4×4	Integer	Rows, columns, and diagonals	0	2	6
4×4	Permutation	Rows	0	4	4
4×4	Permutation	Columns	0	4	4
4×4	Permutation	Rows and columns	0	4	4
4×4	Permutation	Rows, columns, and diagonals	0	2	8
4×4	Tree	Rows	6	4	4
4×4	Tree	Columns	6	4	4
4×4	Tree	Rows and columns	26	4	4
4×4	Tree	Rows, columns, and diagonals	49	4	4
6×6	Integer	Rows	0	20	8
6×6	Integer	Columns	0	19	8
6×6	Integer	Rows and columns	0	7	8
6×6	Integer	Rows, columns, and diagonals	0	7	10
6×6	Permutation	Rows	0	20	6
6×6	Permutation	Columns	0	20	6
6×6	Permutation	Rows and columns	0	18	8
6×6	Permutation	Rows, columns, and diagonals	0	18	8
6×6	Tree	Rows	88	24	4
6×6	Tree	Columns	88	24	4
6×6	Tree	Rows and columns	220	24	4
6×6	Tree	Rows, columns, and diagonals	422	24	4
8×8	Integer	Rows	0	33	12
8×8	Integer	Columns	0	29	12
8×8	Integer	Rows and columns	0	18	12
8×8	Integer	Rows, columns, and diagonals	0	11	12
8×8	Permutation	Rows	0	100	8
8×8	Permutation	Columns	0	100	10
8×8	Permutation	Rows and columns	2	96	10
8×8	Permutation	Rows, columns, and diagonals	4	96	10
8×8	Tree	Rows	476	86	28
8×8	Tree	Columns	5 760	84	36
8×8	Tree	Rows and columns	8 440	80	32
8×8	Tree	Rows, columns, and diagonals	10 169	82	26

obtained for this size. On the other hand, the algorithm did not obtain the optimal value by using the tree representation. This further shows that the evolved CA, although very powerful with dealing with only cryptographic properties, exhibits difficulties when S-boxes of certain structures are being evolved. For all other S-box sizes, the permutation and integer genotypes achieve a similar performance, which seems to demonstrate that both can be used for the considered problem. The only significant difference happens for 8×8 S-box, where the integer genotype obtained a slightly better result for the nonlinearity property, but it also obtained a much worse value for the difference in Hamming weights.

In many cases, the obtained difference in Hamming weights between inputs and outputs is equal to the obtained nonlinearity value. This inspires us to ask a question of whether we found a lower bound on the sum of the difference of the Hamming weights. More precisely, whether for bijective S-boxes, $\sum_{x \in F_2^n} |w_H(F(x)) - w_H(x)| \geq min_{u,v \in F_2^n, \epsilon \in F_2, v \neq 0} d_H(v \cdot F, u \cdot x + \epsilon)$? It turns out this is true and can be mathematically proven. We have $\sum_{x \in F_2^n} |w_H(F(x)) - w_H(x)| \geq d_H(v \cdot F(x), u \cdot x + \epsilon)$ when $u = v$ equals the all-1 vector and $\epsilon = 0$, where d_H denotes the Hamming distance. Indeed, for each x such that $v \cdot F(x) \neq u \cdot x$, we have $w_H(F(x)) \neq w_H(x)$. Observe this is true as the nonlinearity of an S-box is the minimal nonlinearity of all its components. Then, $\sum_{x \in F_2^n} |w_H(F(x)) - w_H(x)|$ is at least the distance between the component function $v \cdot F(x)$ and the linear function $u \cdot x$. *A fortiori* it is at least the nonlinearity. To the best of our knowledge, the characterization between the maximal nonlinearity and the differences between the Hamming weights of input/output pairs is new. As such, we see that EAs not only managed to reach the optimal results for several S-box sizes, but they also inspired the new characterization of the differences of the Hamming weights concerning nonlinearity.

The CA evolved by GP once again demonstrates its superiority when considering only cryptographic properties. Nevertheless, the results obtained for the difference in Hamming weights are poor compared to the results for the other two genotypes. Thus, CA again does not seem to be fit to handle the evolution of S-boxes with additional structure. Although it could be said that this representation achieves poor results because for the S-boxes with better cryptographic properties it is not even possible to obtain good values for the difference in Hamming weights, the results obtained for the S-box of size 8×8 disprove this, since, for that size, GP obtained poor results for both the cryptographic properties and the difference in Hamming weights. It simply seems that the restricted search space of CA is quite suitable for evolving S-boxes of good cryptographic properties (up to the size of 8×8), but it exhibits problems when the S-boxes have to include an additional structure in them.

4 Conclusions and Future Work

The results we obtain suggest that the problem of evolving S-boxes with additional structure can be rather difficult. This should not come as a surprise as we know that S-boxes' design is a difficult task for evolutionary algorithms. The choice of genotype significantly influences the algorithm's ability to obtain S-boxes with additional structure. We managed to find cryptographically optimal S-boxes with sums of rows and columns equal to a specified constant. We also managed to obtain cryptographically optimal S-boxes with a small difference in the Hamming weights between inputs and outputs. We mathematically prove

Table 3: Best results obtained by optimizing the difference in Hamming weights simultaneously with cryptographic properties.

S-box size	Genotype	HWD	nl_F	δ_F
3×3	Integer	2	2	2
4×4	Integer	6	4	2
5×5	Integer	10	10	4
6×6	Integer	22	22	6
7×7	Integer	46	46	8
8×8	Integer	156	102	10
3×3	Permutation	2	2	2
4×4	Permutation	4	4	4
5×5	Permutation	10	10	4
6×6	Permutation	20	20	6
7×7	Permutation	46	46	8
8×8	Permutation	102	100	8
3×3	Tree	6	2	2
4×4	Tree	8	4	4
5×5	Tree	20	12	2
6×6	Tree	60	24	4
7×7	Tree	84	48	8
8×8	Tree	224	84	30

that the smallest sum of differences cannot be smaller than the nonlinearity value, which is a previously unknown result, inspired by evolutionary algorithms experiments and observations. In Figure 1, we give examples of two evolved 4×4 S-boxes for scenarios 1 and 2, respectively. In future work, we plan to explore whether we can obtain magic S-boxes if we consider some other patterns, e.g., looking at broken rows/columns/diagonals. Besides the difference in Hamming weights, we also plan to consider the Hamming distance metric.

7	6	10	11
1	15	2	16
14	8	9	3
12	5	13	4

(a) S-box with optimal cryptographic properties where the elements in rows and columns sum up to the value 34.

0	8	2	4
10	6	5	13
1	3	12	9
14	7	11	15

(b) S-box with optimal cryptographic properties where the difference of Hamming weights equals 4.

Fig. 1: Examples of 4×4 S-boxes with additional structure.

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.: Boolean Functions for Cryptography and Coding Theory. Cambridge University Press (2020)
4. Picek, S., Mariot, L., Yang, B., Jakobovic, D., Mentens, N.: Design of S-boxes Defined with Cellular Automata Rules. In: Proceedings of the Computing Frontiers Conference. CF'17, New York, NY, USA, ACM (2017) 409–414
5. Chabert, J.L. In: Magic Squares. Springer Berlin Heidelberg, Berlin, Heidelberg (1999) 49–81
6. 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, USA (2007)
7. 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
8. Picek, S., Cupic, M., Rotim, L.: A new cost function for evolution of s-boxes. Evolutionary Computation **24**(4) (2016) 695–718 PMID: 27482748.
9. Mariot, L., Picek, S., Leporati, A., Jakobovic, D.: Cellular automata based s-boxes. Cryptography and Communications **11**(1) (Jan 2019) 41–62
10. Jakobovic, D., Picek, S., Martins, M.S.R., Wagner, M.: A characterisation of s-box fitness landscapes in cryptography. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO '19, New York, NY, USA, ACM (2019) 285–293
11. 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 '14, New York, NY, USA, ACM (2014) 13–18
12. Nyberg, K.: On the construction of highly nonlinear permutations. In Rueppel, R., ed.: Advances in Cryptology - EUROCRYPT' 92. Volume 658 of LNCS. Springer Berlin Heidelberg (1993) 92–98
13. 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 LNCS., Springer (1991) 378–386
14. Eiben, A.E., Smith, J.E., et al.: Introduction to evolutionary computing. Volume 53. Springer (2003)
15. Kellegöz, T., Toklu, B., Wilson, J.: Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem. Applied Mathematics and Computation **199**(2) (2008) 590–598
16. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk` (2008) (With contributions by J. R. Koza).
17. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. Trans. Evol. Comp **6**(2) (April 2002) 182–197

18. Abbott, S.: Most-perfect pandiagonal magic squares: their construction and enumeration, by kathleen ollerenshaw and david brée. pp. 172. £19.50. 1998. isbn 0 905091 06 x. (institute of mathematics and its applications, 16 nelson st, southend-on-sea, ss1 1ef). The Mathematical Gazette **82**(495) (1998) 535–536