Cellular Automata Based S-boxes

Luca Mariot[1], Stjepan Picek[2], Alberto Leporati[1], and Domagoj Jakobovic[3]

[1]Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Viale Sarca 336, 20126 Milano, Italy ,
{luca.mariot, leporati}@disco.unimib.it
[2]Cyber Security Research Group, Delft University of Technology, Mekelweg 2, Delft, The Netherlands ,
stjepan@computer.org
[3]University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, 10000, Zagreb, Croatia ,
domagoj.jakobovic@fer.hr

October 28, 2017

Abstract

The design of Substitution Boxes (S-boxes) with good cryptographic properties represents an interesting problem. In this paper, we investigate how to evolve cellular automata (CA) rules that can be then translated into S-boxes. We first prove some upper bounds for the nonlinearity and differential uniformity of S-boxes generated by CA rules. Next, we employ a heuristic technique called Genetic Programming in order to generate CA based S-boxes with good cryptographic properties. Finally, we use the same heuristic paradigm in order to investigate whether a certain S-box is obtainable by a single CA rule.

Keywords Cellular Automata · S-box · Cryptographic properties · Heuristics

1 Introduction

When designing a block cipher, a common direction is to build the Substitution-Permutation Network (SPN) cipher. Such ciphers usually consist of an exclusive-or operation with the key/subkeys, a linear layer, and a substitution layer [20]. To build the substitution layer, one common option in today's designs is to use one or more Substitution Boxes (S-boxes, vectorial Boolean functions). In order for an S-box to be useful, it needs to fulfill a number of cryptographic properties. In his seminal work on the design of block ciphers, Shannon already introduced the concept of *confusion* that an S-box needs to have [36]. Here, confusion can be defined as the

property that the ciphertext statistics should depend on the plaintext statistics in a manner which is too complicated to be exploited by an attacker. This concept is connected with the cryptographic property of *nonlinearity*. Finding an S-box that is resilient against various attacks is not easy and the problem becomes even more complicated if we consider various sizes of S-boxes that are of practical relevance. As examples, some common occurring S-box sizes are 4×4 (PRESENT [5]), 5×5 (Keccak [4]), and 8×8 (AES [18]). Note that the given examples have the same input and output sizes as with the S-boxes we consider in this paper – i.e., mappings from n bits to n bits.

From the cryptographic properties perspective, the minimum set of criteria one would need to consider when designing S-boxes includes bijectivity, high nonlinearity and low differential uniformity. To obtain such properties, there are several options to consider ranging from mathematical constructions to various heuristics. When discussing mathematical constructions, a typical choice is to use power mappings as is the case, for instance, of the AES S-box (where the inverse power function and affine transformation are used). On the other hand, in heuristic approaches one has at his disposal a number of techniques that in general cannot compete with mathematical constructions, but anyway offer an interesting alternative in specific scenarios (see Section 7 for details).

In this paper, we concentrate on S-boxes constructed with cellular automata (CA) rules. More precisely, a CA-based S-box can be considered as a particular type of a vectorial Boolean function where each coordinate function corresponds to the CA rule applied on a local neighborhood. The best known example of such an S-box is the Keccak sponge construction which is now a part of the SHA-3 standard [4]. There, the authors use a CA rule affecting only three neighborhood positions for each bit, which results in an extremely lightweight definition of the S-box with a small implementation cost, but which also yields suboptimal cryptographic properties. To the best of our knowledge, all the other ciphers using CA rules for the S-box definition actually use that same rule. This is the case of Panama [15], RadioGatún [3], Subterranean [13], and 3Way [17] ciphers. Besides those S-boxes, there are also designs using an S-box that is an affine transformation of the Keccak S-box, such as Ascon [12].

In our investigation, we discuss CA-based S-boxes both from a theoretical and experimental perspective: first, we show bounds one can achieve when using such S-boxes with respect to the nonlinearity and differential uniformity properties, and then we give experimental results for a number of S-box sizes. In order to construct CA-based S-boxes, we employ an *evolutionary computation* (EC) technique called *Genetic Programming* (GP). In particular, we evolve CA rules in the form of Boolean functions, where those rules are actually the description of S-boxes. We emphasize that not all bijective S-boxes can be represented by CA rules, and consequently the number of S-boxes expressible through CA is smaller than the total number of S-boxes of a certain size. If we consider the AES S-box as an example, it is possible to see that this S-box cannot be obtained with a single CA rule. Still, this does not mean there are no S-boxes of that size with the same properties that cannot

be constructed with a single CA rule. On the other hand, there are infinitely many ways how one can represent an S-box with CA rules. For example, by adopting a tree representation for the rule, it suffices to consider the trivial approach where one adds subexpressions that cancel themselves out. Accordingly, the number of CA rules representations is much larger than the number of S-boxes and it is impossible to exhaustively visit them even for small sizes. We finally present a "reverse-engineering" procedure that is able to find CA rules resulting in specific S-boxes. Indeed, if we start with an S-box that can be represented with a single CA rule, the question is whether there exists a shorter rule resulting in the same S-box. Our reverse engineering approach can help in obtaining such shorter rules.

The rest of the paper is organized as follows. In Section 2, we discuss necessary information about S-boxes and cryptographic properties we consider. Section 3 gives background on cellular automata and their connection with S-boxes. Section 4 gives theoretical results – specifically, upper bounds for nonlinearity and differential uniformity attainable by CA-based S-boxes. Section 5 offers experimental results where we investigate how to use heuristics to build CA-based S-boxes as well as to provide enumerations of affine classes for several S-box sizes. In Section 6, we discuss the results and possible future research directions. Section 7 gives a short overview of related work, both from the CA and EC perspective. Finally, in Section 8 we give a brief conclusion.

2 Cryptographic Properties of S-boxes

Let n, m be positive integers, i.e., $n, m \in \mathbb{N}^+$. We denote by $\mathbb{F}_2^n$ 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 Galois field with two elements. Further, for any set S, we denote $S \setminus \{0\}$ by S^*. The usual inner product of a and b equals $a \cdot b = \bigoplus_{i=1}^{n} a_i b_i$ in $\mathbb{F}_2^n$. The addition of elements of the finite field $\mathbb{F}_{2^n}$ is denoted with "+", as usual in mathematics. Often, we identify $\mathbb{F}_2^n$ with $\mathbb{F}_{2^n}$ and if there is no ambiguity, we denote the addition of vectors of $\mathbb{F}_2^n$ when $n > 1$ with "+" as well.

The *Hamming weight* $w_H(a)$ of a vector a, where $a \in \mathbb{F}_2^n$, is the number of non-zero positions in the vector. An (n, m)-function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. An (n, m)-function F can be defined as a vector $F = (f_1, \cdots, f_m)$, where the Boolean functions $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ for $i \in \{1, \cdots, m\}$ 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. Since for every n, there exists a field $\mathbb{F}_{2^n}$ of order 2^n, we can endow the vector space $\mathbb{F}_2^n$ with the structure of that field when convenient. If the vector space $\mathbb{F}_2^n$ is identified with the field $\mathbb{F}_{2^n}$ then we can take $a \cdot b = tr(ab)$ where $tr(x) = x + x^2 + \ldots + x^{2^{n-1}}$ is the *trace function* from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$.

2.1 S-box Representations

A Boolean function f on $\mathbb{F}_2^n$ can be uniquely represented by a truth table (TT), which is a vector $(f(0), ..., f(1))$ that contains the function values of f in lexicographical order with respect to the input entries, i.e., for $a, b \in \mathbb{F}_2^n$, it holds $a \leq b$ if and only if $a_i \leq b_i$ for all $i \in \{1, \cdots, n\}$ [9]. An (n, m) S-box can be represented in the truth table form as a matrix of dimension $2^n \times m$ where each of the m columns represents a coordinate function.

The *Walsh-Hadamard* transform of an (n, m)-function F is (see e.g., [10]):

$$W_{v \cdot F}(\omega) = \sum_{x \in \mathbb{F}_2^n} (-1)^{v \cdot F(x) \oplus \omega \cdot x}, \ \omega, v \in \mathbb{F}_2^m. \tag{1}$$

2.2 S-box Properties

In order to resist linear and differential cryptanalysis attacks, a balanced S-box should ideally have high nonlinearity and low differential uniformity. 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. Balanced (n, n)-functions correspond to bijective S-boxes.

The *linearity* of F (also called the *spectral radius*) is defined as the maximum linearity of all its component functions $v \cdot F$, where $v \in \mathbb{F}_2^{m*}$ [26, 10]:

$$\mathcal{L}(F) = \max_{\substack{\omega \in \mathbb{F}_2^n \\ v \in \mathbb{F}_2^{m*}}} |W_{v \cdot F}(\omega)|. \tag{2}$$

The *nonlinearity* N_F of an (n, m)-function F equals:

$$N_F = 2^{n-1} - \frac{1}{2}\mathcal{L}(F). \tag{3}$$

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 define the *delta difference table* of F with respect to a and b as:

$$D_F(a, b) = \left\{ x \in \mathbb{F}_2^n : F(x) \oplus F(x \oplus a) = b \right\}. \tag{4}$$

The entry at position (a, b) corresponds to the cardinality of the delta difference table $D_F(a, b)$ and is denoted as $\delta_F(a, b)$. The *differential uniformity* δ_F is then defined as [25]:

$$\delta_F = \max_{\substack{a \in \mathbb{F}_2^{n*} \\ b \in \mathbb{F}_2^m}} \delta_f(a, b). \tag{5}$$

2.3 S-box Bounds

The nonlinearity of any (n, m) function F is bounded above by the so-called *covering radius bound*:

$$N_F \leq 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{6}$$

Functions satisfying the above bound are called *bent*, and they exist only for n even. When $m = n$ a better bound exists. The nonlinearity of any (n, n) function F is bounded above by the so-called *S*idelnikov-Chabaud-Vaudenay bound [11]:

$$N_F \leq 2^{n-1} - 2^{\frac{n-1}{2}}. \tag{7}$$

Bound (7) is an equality if and only if F is an *Almost Bent* (AB) function, by definition of AB functions [10].

Functions that have differential uniformity equal to 2 are called the *Almost Perfect Nonlinear* (APN) functions. Every AB function is also APN, but the converse does not hold in general. AB functions exist only in an odd number of variables, while APN functions also exist for an even number of variables. When discussing the differential uniformity parameter for permutations, the best possible (and known) value is 2 for any odd n and also for $n = 6$. For n even and larger than 6, this is an open question. The differential uniformity value for the inverse function $F(x) = x^{2^n-2}$ equals 4 when n is even and 2 when n is odd.

2.4 Affine Equivalence

Two S-boxes S_1 and S_2 of dimension $n \times n$ are *affine equivalent* if the following equation holds [10]:

$$S_1(x) = B(S_2(A(x) \oplus a)) \oplus b, \tag{8}$$

where A and B are invertible $n \times n$ matrices in $GF(2)$ and $a, b \in \mathbb{F}_2^n$.

Both nonlinearity and differential uniformity are affine invariant properties, which means that an affine transformation of an S-box will not change the values of those properties.

3 Cellular Automata

Cellular Automata (CA) are parallel computational models that have been used to simulate and analyze a wide variety of discrete complex systems in different application domains. A CA is characterized by a lattice of *cells*. During a single time step, each cell in the lattice synchronously updates its state according to a *local rule*, which is applied to the *neighborhood* of the cell. In what follows, we focus on *one-dimensional Boolean cellular automata*, meaning that the lattice is a one-dimensional array, and the state of each cell is binary. The following definition formalizes the two models of CA we address in this work:

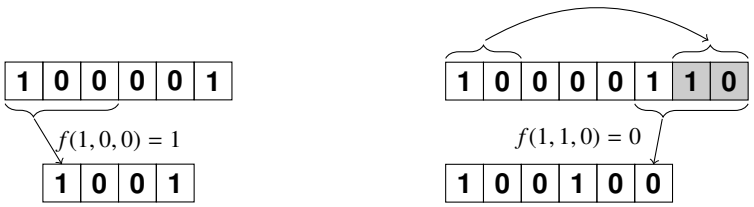
Definition 1. *Let $f : \mathbb{F}_2^d \to \mathbb{F}_2$ and $n \geq d$. We define the following two models of* one-dimensional Boolean *CA with n input cells and* local rule *f:*

- No Boundary CA *(NBCA): $F : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ is defined for all $x \in \mathbb{F}_2^n$ as:*

$$F(x_1, x_2, \cdots, x_n) = (f(x_1, \cdots, x_d), f(x_2, \cdots, x_{d+1}), \cdots, f(x_{n-d+1}, \cdots, x_n)) \ . \tag{9}$$

- Periodic Boundary CA *(PBCA): $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is defined for all $x \in \mathbb{F}_2^n$ as:*

$$F(x_1, x_2, \cdots, x_n) = (f(x_1, \cdots, x_d), \cdots, f(x_{n-d}, \cdots, x_1), \cdots, f(x_n, \cdots, x_{d-1})) \ . \tag{10}$$



(a) No Boundary CA – NBCA (b) Periodic Boundary CA – PBCA

Figure 1: Examples of NBCA and PBCA with local rule 150, defined as $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$

Thus, a CA can be seen as a vectorial Boolean function where each coordinate function f_i corresponds to the local rule f applied to the *neighborhood* $(x_i, \cdots, x_{i+d-1})$. In the no boundary case, this rule is applied just up to the coordinate $n - d + 1$, meaning that the size of the input array shrinks by $d - 1$ cells. In the periodic setting, the CA array is seen as a ring, so that the first cell follows the last one. The remaining $d - 1$ cells are updated by using the first $d - 1$ as their right neighbors. Notice that, since the local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is a Boolean function, it can be represented by a truth table of 2^d bits. In the CA literature, another common way to identify a local rule is by means of its *Wolfram code*, which is basically the decimal representation of the truth table. Figure 1 depicts an example of NBCA and PBCA using rule 150, defined as the XOR of the three neighborhood variables: $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$. The vectorial Boolean function F of a CA is also called the CA *global rule*.

Example 1. *The nonlinear transformation χ used in* Keccak *[4] is a PBCA with $n = 5$ cells and local rule $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ defined as:*

$$f(x_1, x_2, x_3) = x_1 \oplus x_2 x_3 \oplus x_3 \ . \tag{11}$$

The Wolfram code for such rule is 210.

Remark 1. *Note that if the rule used in Keccak is used when the length of the cellular array is even, then the resulting S-box is not bijective. In particular, the S-box is bijective if and only if the size of the CA is odd [16]. Since the Keccak rule has diameter $d = 3$, it results in an optimal S-box of size 3×3, while for the size 5×5 (adopted in the design of Keccak) the resulting S-box has suboptimal cryptographic properties. Naturally, as one would extend the size of this S-box by adding new cells to the CA, the cryptographic properties would become increasingly worse. In Section 4 we formalize this observation by analyzing how the nonlinearity and differential uniformity of a CA are affected by adding new cells, deriving upper bounds for these two cryptographic properties.*

Remark 2. *PBCA with $d = n$ actually correspond to* rotational symmetric S-boxes*, originally introduced in [34].*

We conclude this section by observing that, in the CA literature, the focus is usually on the *iterated behavior* of CA. In particular, the local rule is applied to all cells in parallel for multiple time steps, in order to study the long-term properties of the resulting dynamical system. On the other hand, in this work we only consider the situation where the CA is evolved for just one time step, and investigate the cryptographic properties of the resulting vectorial Boolean functions. This is the same approach used by the designers of the CA-based nonlinear transformation χ used in Keccak [4].

4 Theoretical Findings

In this section, we prove some bounds on the nonlinearity and differential uniformity of S-boxes defined by CA, relating them to the corresponding properties of the underlying local rules. To prove our results, we make use of the following theorem proved by Nyberg, concerning how the nonlinearity and the differential uniformity of an S-box are affected by adding a coordinate function while maintaining fixed the number of input variables [27].

Theorem 1. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be an S-box defined by m coordinate functions $f_1, \cdots, f_m : \mathbb{F}_2^n \to \mathbb{F}_2$, and let $g : \mathbb{F}_2^n \to \mathbb{F}_2$. Define $\tilde{F} : \mathbb{F}_2^n \to \mathbb{F}_2^{m+1}$ as follows:*

$$\tilde{F}(x_1, \cdots, x_n) = (f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n), g(x_1, \cdots, x_n)) \ . \tag{12}$$

Then, the following upper bounds hold:

$$N_F(\tilde{F}) \leq \min\{N_F(F), N_F(g)\} \ . \tag{13}$$

$$\frac{1}{2}\delta_F \leq \delta_{\tilde{F}} \leq \delta_F \ . \tag{14}$$

Consider now a CA (either with no boundary or periodic boundary conditions) with n cells and local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$. How do the nonlinearity and differential uniformity of F change by adding a new cell, thus obtaining a new CA $\tilde{F}$ of $n + 1$ cells? Observe that Theorem 1 cannot be directly applied here, because we need to address the case where both a coordinate function *and* an input variable are added to the original CA. We first address this situation for generic S-boxes (i.e., not necessarily defined by a CA rule) in the following result:

Theorem 2. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be an S-box defined by m coordinate functions $f_1, \cdots, f_m : \mathbb{F}_2^m \to \mathbb{F}_2$, and let $g : \mathbb{F}_2^{n+1} \to \mathbb{F}_2$ be a Boolean function defined on $n + 1$ variables. Define $\tilde{F} : \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{m+1}$ as follows:*

$$\tilde{F}(x_1, \cdots, x_n) = (f_1(x_1, \cdots, x_n), \cdots, f_m(x_1, \cdots, x_n), g(x_1, \cdots, x_n, x_{n+1})) \ . \tag{15}$$

Then, $\tilde{F}$ satisfies the following bounds:

$$N_{\tilde{F}} \leq \min\{2 \cdot N_F, N_g\} \ , \tag{16}$$

$$\delta_F \leq \min\{2 \cdot \delta_F, \delta_g\} \ . \tag{17}$$

Proof. We begin by addressing the bound on nonlinearity. We are going to analyze the Walsh-Hadamard transform of $\tilde{F}$ by classifying its component functions as follows:

(i) The $2^m - 1$ component functions that do not select the new coordinate g, i.e., those described by the vectors $\tilde{v} = (v, 0) \in \mathbb{F}_2^{m+1}$, where $v \in \mathbb{F}_2^{m*}$.

(ii) The single component function that just selects g, defined by the vector $(\underline{0}, 1)$ where $\underline{0} \in \mathbb{F}_2^m$.

(iii) Finally, the $2^m - 1$ component functions that select g and whose first m coordinates are not all zeros, defined by the vectors $\tilde{v} = (v, 1) \in \mathbb{F}_2^{m+1}$, where $v \in \mathbb{F}_2^{m*}$.

Consider the component functions of type (i). Let $\tilde{v} = (v, 0) \in \mathbb{F}_2^{m+1}$, where $v \in \mathbb{F}_2^{m*}$. Then, the Walsh-Hadamard transform of $\tilde{v} \cdot \tilde{F}$ computed on $\omega \in \mathbb{F}_2^{n+1}$ equals

$$\begin{aligned} W_{\tilde{v}\cdot\tilde{F}}(\tilde{\omega}) &= \sum_{\tilde{x}\in\mathbb{F}_2^{n+1}} (-1)^{\tilde{v}\cdot\tilde{F}(\tilde{x}) \,\oplus\, \tilde{\omega}\cdot\tilde{x}} = \sum_{(x,x_{n+1})\in\mathbb{F}_2^{n+1}} (-1)^{(v,0)\cdot(F(x),g(x_{n+1})) \,\oplus\, (\omega,\omega_{n+1})\cdot(x,x_{n+1})} = \\ &= \sum_{(x,x_{n+1})\in\mathbb{F}_2^{n+1}} (-1)^{v\cdot F(\tilde{x}) \,\oplus\, \omega\cdot x} \cdot (-1)^{\omega_{n+1}\cdot x_{n+1}} \ . \end{aligned} \tag{18}$$

Let us rewrite the right hand side of Eq. (18) by dividing the sum with respect to the value of x_{n+1}:

$$\begin{aligned} W_{\tilde{v}\cdot\tilde{F}}(\tilde{\omega}) &= \sum_{(x,0)\in\mathbb{F}_2^{n+1}} (-1)^{v\cdot F(v) \,\oplus\, \omega\cdot x} + \sum_{(x,1)\in\mathbb{F}_2^{n+1}} (-1)^{v\cdot F(x) \,\oplus\, \omega\cdot x \,\oplus\, \omega_{n+1}} = \\ &= \sum_{x\in\mathbb{F}_2^n} (-1)^{v\cdot F(x) \,\oplus\, \omega\cdot x} + (-1)^{\omega_{n+1}} \cdot \sum_{x\in\mathbb{F}_2^n} (-1)^{v\cdot F(x) \,\oplus\, \omega\cdot x} \ . \end{aligned} \tag{19}$$

Notice that the two sums in Eq. (19) correspond to the Walsh-Hadamard coefficient $W_{v\cdot F}(\omega)$. Thus, it holds that

$$W_{\tilde{v}\cdot\tilde{F}}(\tilde{\omega}) = \begin{cases} 0 & , \text{if } \omega_{n+1} = 0 \\ 2 \cdot W_{v\cdot F}(\omega) & , \text{if } \omega_{n+1} = 1 \end{cases} \tag{20}$$

Hence, by Eq. (20) we have that the linearity of $\tilde{F}$ will be at least twice the linearity of F, from which it follows that

$$N_{\tilde{F}} \leq 2 \cdot N_F \ . \tag{21}$$

Let us now consider the component of type (ii), i.e., the one defined by $\tilde{v} = (\underline{0}, 1)$. In this case, it is easy to see that $N_{\tilde{v}\cdot\tilde{F}} = N_g$, which yields

$$N_{\tilde{F}} \leq N_g \ . \tag{22}$$

Since the nonlinearity of $\tilde{F}$ is defined as the minimum nonlinearity among all its component functions, by combining Eqs. (21) and (22) we get

$$N_{\tilde{F}} \leq \min\{2 \cdot N_F, N_g\} \ .$$

We finally address the differential uniformity bound. Given $\tilde{a} = (a, a_{n+1}) \in \mathbb{F}_2^{n+1}$ and $\tilde{b} = (b, b_{m+1}) \in \mathbb{F}_2^{m+1}$, the delta difference table of $\tilde{F}$ with respect to $\tilde{a}$ and $\tilde{b}$ is

$$\begin{aligned} D_{\tilde{F}}(a,b) &= \{\tilde{x} = (x, x_{n+1}) \in \mathbb{F}_2^{n+1} : \tilde{F}(\tilde{x} \oplus \tilde{a}) \oplus \tilde{F}(\tilde{x}) = \tilde{b}\} = \\ &= \{\tilde{x} \in \mathbb{F}_2^{n+1} : (F(x \oplus a), g(\tilde{x} \oplus \tilde{a})) \oplus (F(x), g(\tilde{x})) = (b, b_{m+1})\} = \\ &= \{\tilde{x} \in \mathbb{F}_2^{n+1} : [F(x \oplus a) \oplus F(x) = b] \wedge [g(\tilde{x} \oplus \tilde{a}) \oplus g(\tilde{x}) = b_{m+1}]\} = \\ &= \{\tilde{x} \in \mathbb{F}_2^{n+1} : [x \in D_F(a,b)] \wedge [\tilde{x} \in D_g(\tilde{a}, \tilde{b})]\} = \\ &= \{\tilde{x} \in \mathbb{F}_2^{n+1} : x \in D_F(a,b)\} \cap \{(x, x_{n+1}) \in \mathbb{F}_2^{n+1} : \tilde{x} \in D_g(\tilde{a}, \tilde{b})\} = A \cap B \ . \end{aligned} \tag{23}$$

Clearly, from Eq. (23) we have that $B = D_g(\tilde{a}, \tilde{b})$, and thus $|B| = \delta_g(\tilde{a}, \tilde{b})$. On the other hand, for A we obtain $|A| = 2 \cdot |D_F(a,b)| = 2 \cdot \delta_F(a,b)$, since the vectors $\tilde{x}$ in A are constructed by taking all vectors x belonging to $D_F(a,b)$ and by appending to their right a 0 and a 1. Consequently, it holds that

$$\delta_{\tilde{F}}(\tilde{a}, \tilde{b}) = |A \cap B| \leq \min\{2 \cdot \delta_F(a,b), \delta_g(\tilde{a}, \tilde{b})\} \ . \tag{24}$$

Finally, observe that one can construct the delta difference tables of maximum cardinality of $\tilde{F}$ by taking all possible intersections between the delta difference tables of maximum cardinality of F and g. Hence, the differential uniformity of $\tilde{F}$ satisfies

$$\delta_{\tilde{F}} \leq \min\{2 \cdot \delta_F, \delta_g\} \ . \tag{25}$$

□

Of course, the upper bounds given in Eq. (16) are not tight. In fact, the component functions of type (iii) could yield a lower nonlinearity and differential uniformity than those featured by the components of types (i) and (ii) considered in the proof of Theorem 2.

Before turning our attention to the CA case, we still need one more preliminary result about how the nonlinearity and differential uniformity of a Boolean function change by adding dummy variables:

Lemma 1. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function with nonlinearity N_f and differential uniformity δ_f. Given $t \in \mathbb{N}$, define $\tilde{f} : \mathbb{F}_2^{n+t} \to \mathbb{F}_2$ as follows:*

$$\tilde{f}(x_1, \cdots, x_n, x_{n+1}, \cdots, x_{n+t}) = f(x_1, \cdots, x_n) \ . \tag{26}$$

Then, the following equalities hold:

$$N_{\tilde{f}} = 2^t \cdot N_f \ \ , \delta_{\tilde{f}} = 2^t \cdot \delta_f \ \ . \tag{27}$$

Proof. We proceed by induction on t.

For $t = 1$, one can easily see that $\tilde{f}$ is a special case of the vectorial function $\tilde{F}$ considered in Theorem 2 with $m = 1$, with the difference that no new output coordinates are added. Hence, the Walsh-Hadamard transform of $\tilde{f}$ is described by Eq. (20), which yields $N_{\tilde{f}} = 2 \cdot N_f$. On the other hand, for $\tilde{a} = (a, a_{n+1}) \in \mathbb{F}_2^{n+1}$ and $b \in \mathbb{F}_2$, the delta difference table of $\tilde{f}$ is

$$D_{\tilde{f}}(\tilde{a}, b) = \{(x, x_{n+1}) \in \mathbb{F}_2^{n+1} : f(x \oplus a) \oplus f(x) = b\} \ \ ,$$

from which it follows that $\delta_{\tilde{f}}(\tilde{a}, b) = 2 \cdot \delta_f(a, b)$, and thus $\delta_{\tilde{f}} = 2 \cdot \delta_f$.

Next, assume that $t > 1$, and consider the case $t + 1$, with $f' : \mathbb{F}_2^{n+t} \to \mathbb{F}_2$ indicating the function truncated at $n + t$ variables. Then, by induction hypothesis the following equalities are satisfied:

$$\begin{aligned} N_F(f') &= 2^t \cdot N_F(f) \ \ , \\ \delta_{f'} &= 2^t \cdot \delta_f \ \ . \end{aligned}$$

Similarly to the case $t = 1$, the Walsh-Hadamard coefficients of $\tilde{f} : \mathbb{F}_2^{n+t+1} \to \mathbb{F}_2$ are as in Eq. (20), from which one obtains

$$N_{\tilde{f}} = 2 \cdot N_{f'} = 2 \cdot 2^t \cdot N_f = 2^{t+1} \cdot N_f \ \ . \tag{28}$$

Finally, the delta difference table $D_{\tilde{f}}(\tilde{a}, b)$ is again constructed by appending a 0 and a 1 to all vectors in $D_{f'}(a, b)$. Hence, the equality $\delta_{\tilde{f}}(\tilde{a}, b) = 2 \cdot \delta_{f'}(a, b)$ holds for all $\tilde{a} = (a, a_{n+t+1}) \in \mathbb{F}_2^{n+t+1}$, from which it finally follows that

$$\delta_{\tilde{f}} = 2 \cdot \delta_{f'} = 2 \cdot 2^t \delta_f = 2^{t+1} \cdot \delta_f \ \ .$$

□

Leveraging on the above results, we can now prove upper bounds on the nonlinearity and differential uniformity of S-boxes defined by CA, both in the no boundary and the periodic settings:

Theorem 3. *Let $f : \mathbb{F}_2^d \to \mathbb{F}_2$ and $n \geq d$. Then, the NBCA and PBCA $\tilde{F}$ with n input cells and local rule f satisfy the following bounds:*

$$N_{\tilde{F}} \leq 2^{n-d} \cdot N_f \tag{29}$$

$$\delta_{\tilde{F}} \leq 2^{n-d} \cdot \delta_f \ \ . \tag{30}$$

Proof. We first address the no boundary case. Let $\tilde{F} : \mathbb{F}_2^n \to \mathbb{F}_2^{n-d+1}$ be a NBCA with local rule f. We proceed by induction on $m = n - d + 1$.

For $m = 2$, we can apply Theorem 2 by setting $F = f$ and $g : \mathbb{F}_2^{d+1} \to \mathbb{F}_2$ defined as follows:

$$g(x_1, x_2, \cdots, x_{d+1}) = f(x_2, \cdots, x_{d+1}) \ .$$

Thus, Theorem 2 yields that

$$\begin{aligned} N_{\tilde{F}} &\leq \min\{2 \cdot N_f, N_g\} \ , \\ \delta_{\tilde{f}} &\leq \min\{2 \cdot \delta_f, \delta_g\} \ . \end{aligned}$$

Additionally, by Lemma 1 we know that

$$\begin{aligned} N_g &= 2 \cdot N_f \ , \\ \delta_g &= 2 \cdot \delta_f \ . \end{aligned}$$

Since $m - 1 = n - d + 1 - 1 = 1$, the three bounds are satisfied in the base case.

Next, let us assume that $m > 2$ and consider the case $m+1$, with $\tilde{F} : \mathbb{F}_2^{n+1} \to \mathbb{F}_2^{m+1}$ being the NBCA with $n + 1$ cells. In particular, define $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ as the NBCA with n cells, and $g : \mathbb{F}_2^{n+1} \to \mathbb{F}_2$ as $g(x_1, \cdots, x_{n+1}) = f(x_{n-d}, \cdots, x_{n+1})$. Again, Theorem 2 gives us that

$$\begin{aligned} N_{\tilde{F}} &\leq \min\{2 \cdot N_F, N_g\} \ , \\ \delta_{\tilde{F}} &\leq \min\{2 \cdot \delta_F, \delta_g\} \ . \end{aligned}$$

while by Lemma 1 we obtain

$$\begin{aligned} N_g &= 2^m \cdot N_f \ , \\ \delta_g &= 2^m \cdot \delta_f \ . \end{aligned}$$

Remarking that $m + 1 = n - d + 1$, by induction hypothesis one finally gets

$$\begin{aligned} N_{\tilde{F}} &\leq 2^m \cdot N_f = 2^{n-d} \cdot N_f \ , \\ \delta_{\tilde{F}} &\leq 2^m \cdot \delta_f = 2^{n-d} \cdot \delta_f \ , \end{aligned}$$

which concludes the proof for the NBCA case. Finally, for the periodic case it just suffices to observe that the PBCA is constructed by adding $n - d$ coordinate functions to the NBCA $\tilde{F}$ without extending the number of input variables, where the new coordinates always coincide with the local rule f applied on the rightmost and leftmost $d - 1$ cells. Hence, Theorem 1 can be applied here, from which one deduces that the same bounds for nonlinearity and differential uniformity also hold for the PBCA case. □

Tables 2a and 2b report the best nonlinearity values respectively reachable by PBCA as given by Theorem 3, for various values of d and n, and by generic bijective (n, n)-functions. Table 2a is lower triangular because the bound of Theorem 3 is

(a) PBCA S-Boxes

CA size n \ Rule size d	3	4	5	6	7
3	2	–	–	–	–
4	4	4	–	–	–
5	8	8	12	–	–
6	16	16	24	24	–
7	32	32	48	48	56

(b) Generic (n, n)-functions

$n \times n$	N_F
3×3	2
4×4	4
5×5	12
6×6	24
7×7	56

Figure 2: Best attainable nonlinearity values for PBCA S-boxes and generic bijective S-boxes up to $n = 7$ variables.

meaningful only if $n \geq d$. For the maximum nonlinearity of N_f of the local rule we considered the quadratic bound, since it is known to be optimal for balanced Boolean functions of sizes up to $d = 7$ variables [9]. By comparing Tables 2a and 2b one can see that the only case where CA are able to reach the same best values as generic (n, n)-functions is when $d = n$, i.e. the rotation-symmetric case which corresponds to the diagonal of Table 2a. This also explains from a theoretical point of view why the nonlinearity of the CA χ used in Keccak is suboptimal with respect to the Sidelnikov-Chabaud-Vaudenay bound, since the neighborhood size of the rule is $d = 3$ while $n = 5$. The χ rule is, however, optimal with respect to the nonlinearity bound given in Theorem 3.

5 Experimental Validation

5.1 Genetic Programming Approach

Genetic Programming (GP) is an evolutionary algorithm in which the data structures that undergo optimization are computer programs [2]. Although GP has a history longer than 50 years, its full acceptance is due to the work of John Koza at the beginning of the 1990s in which he formalized the idea of employing chromosomes on the basis of tree data structures. Since the aim of GP is to automatically generate new programs, each individual of a population represents a computer program [2], where the most common used form are symbolic expressions representing parse trees. A parse tree (syntax tree) is an ordered, rooted tree that represents the syntactic structure of a string according to some context-free grammar. A tree can represent a mathematical expression, a rule set or a decision tree, for instance.

The building elements in a tree-based GP are functions (inner nodes) and terminals (leaves, problem variables) where both functions and terminals are known as primitives. For further information about GP, we refer interested readers to [21, 32].

In our approach, the task of GP is to evolve a Boolean function of n variables, in the form of a tree. This function will be used as a CA local rule, so each GP individual presents a CA rule that needs to be evaluated. In this process, we assume

the following: the state of a CA is represented with a periodic one-dimensional binary array of size n. The elements of the binary array are used as Boolean variables in a GP tree (GP terminals), where the variable c_0 denotes the value that is being updated. The variables $c_1, \ldots, c_{n-1}$ denote the cells to the right of the current cell. In our experiments, the neighborhood of a cell is formed by the cell itself and the $n-1$ cells to its right, so each value in the current state can be used in a local update rule, which corresponds to the case of rotation-symmetric S-boxes (i.e. $d = n$). As discussed in the previous section, this choice is motivated by the fact that only in this case the CA nonlinearity bound given by Theorem 3 coincides with the Sidelnikov-Chabaud-Vaudenay bound.

The function set consists of several Boolean primitives necessary to represent any Boolean function. We use the following function set: NOT, which inverts its single argument, XOR, NAND, NOR, each of which takes two input arguments. Additionally, we use the function IF, which takes three arguments and returns the second one if the first one evaluates to *true*, and the third one otherwise. This function corresponds to the multiplexer gate (MUX).

A candidate Boolean function obtained with GP is evaluated in the following manner: all the possible 2^n input states are considered, and for each state the same rule is applied in parallel to each of the variables to determine the next state. The obtained global rule represents a candidate S-box that is then evaluated according to the desired cryptographic criteria.

In the evolution process, GP uses a 3-tournament selection, where the worst of three randomly selected individuals is eliminated. A new individual is then created by applying crossover to the remaining two individuals from the tournament. The new individual is then mutated with a probability of 0.5. We note that we use the mutation probability to select whether an individual would be mutated or not, and the mutation operator is executed only once on a given individual; e.g., if the mutation probability is 0.5, then on average 5 out of every 10 new individuals will be mutated and one mutation will be performed on each of those 5 individuals. This procedure is illustrated in Algorithm 1.

The variation operators are simple tree crossover, uniform crossover, size fair, one-point, and context preserving crossover (selected at random), and subtree mutation [33]. All our experiments suggest that having a maximum tree depth equal to the size of S-box is sufficient (i.e., tree depth equals n, which is the number of Boolean variables). The initial population is created at random and every experiment is repeated 50 times.

In order to examine the influence of the GP parameters, we conduct a tuning phase for the stopping criterion and the population size. The starting set of parameters was tested on S-boxes of size $n = 6$, with population 500, 1 000, and 2 000, for which 30 runs were executed. Although there were no significant differences, the best results in terms of average fitness value and the number of optimal solutions were obtained with a population size of 2 000, which we used in the subsequent experiments. Finally, the stopping criterion is set to 2 000 000 evaluations, since no change of the best solution was detected afterwards.

Algorithm 1 Genetic Programming evolution

repeat
  randomly select 3 individuals;
  remove the worst of 3 individuals;
  child = crossover (remaining two individuals);
  perform mutation on *child*, with given individual mutation probability;
  generate S-box using *child* Boolean function
  evaluate S-box
  assign fitness to *child*
  insert *child* into population;
until stopping criteria reached

5.2 Fitness Function

We require that the evolved S-boxes are balanced, with high nonlinearity, and low differential uniformity. We design our fitness function to be a two-stage function. First, the balancedness is verified, and if an S-box is balanced, we assign it a value of 0, otherwise the value equals -1; this is denoted with the label *BAL*. Only if the S-box is balanced, we calculate the nonlinearity and differential uniformity, which is subtracted from the value 2^n, since we aim to minimize the value of that property. The goal is to **maximize** the value:

$$fitness = BAL + \Delta_{BAL,0}\left(N_F + \left(1 - \frac{nMinN_F}{2^n}\right) + (2^n - \delta_F)\right). \tag{31}$$

Here, $\Delta_{BAL,0}$ represents the Kronecker delta function that equals one when the function is balanced (i.e., $BAL = 0$) and zero otherwise. $nMinN_F$ represents the number of occurrences of the current value of N_F. Since the difference in neighboring levels of nonlinearity is always at least 2 (we consider only bijective S-boxes), the part of the expression including $nMinN_F$ acts as a secondary criterion which is effectively being minimized and assumes values in the range $[0, 1]$.

5.3 Results

In Table 1, we give statistical results for our heuristic search. The results are averaged over the best obtained values for each run. Column *T_max* denotes the theoretical maximal value of the fitness function. These results serve as an indicator on the performance of our search technique.

As can be seen in the table, for dimensions 4×4 and 5×5 the problem is easy. Indeed, for the former, all the experiments finished with the optimal value, and for the latter, most of the runs reached the optimal solution. For the size 6×6, we observe a larger standard deviation, but the obtained value is still high. For the size 7×7, we see that the problem becomes difficult, although there are a few runs reaching the optimal value. Finally, for the size 8×8, we see the results do not even come close to the optimal value.

Table 1: Statistical results and comparison.

S-box size	T_max	GP			N_F	δ_F
		Max	Avg	Std dev		
4×4	16	**16**	16	0	4	4
5×5	42	**42**	41.73	1.01	12	2
6×6	86	**84**	80.47	4.72	24	4
7×7	182	**182**	155.07	8.86	56	2
8×8	364	318	281.87	13.86	82	20

Table 2: Tree sizes, Eq. (31)

S-box size	Min	Max	Avg	Std dev
4×4	**8**	103	48.77	25.03
5×5	**6**	67	26.27	11.93
6×6	**9**	82	35.13	18.71
7×7	**6**	64	30.63	14.77
8×8	**15**	119	68.7	28.93

Observe that for sizes 4×4, 5×5, and 7×7 we obtained S-boxes with the best possible values of nonlinearity and differential uniformity. For 6×6 size our best solution is close to the optimal one. Actually, the difference between our solution and the optimal one is in the differential uniformity property where our solution has value 4 and the optimal value equals 2 [6]. We subsequently investigated the optimal 6×6 S-box and found that it is not possible to obtain it with a single CA rule (see Sec. 5.4), which means that our approach results in the best possible obtainable values for the 6×6 size.

In Table 2 we display statistics for tree sizes for every S-box dimension. Note that the values are averaged over all runs and not only over those that resulted in S-boxes with optimal values. The two most interesting sizes seem to be 4×4 and 8×8. In the former case it appears to be easy to obtain optimal values (note that in all runs we obtained optimal solutions as given in Table 1) and therefore GP relatively easily founds even longer rules that result in optimal S-boxes. In the latter case, we see that even finding solutions that result in bijective S-boxes with suboptimal properties requires on average long trees. We consider this to be one of the reasons for relative lack of success for the 8×8 size, i.e., to obtain better solutions one would need much larger trees and consequently much longer evolution time.

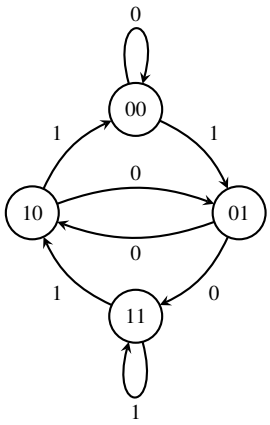


Figure 3: De Bruijn graph associated to CA rule of Keccak.

5.4 Reverse Engineering of CA-based S-boxes

Up to now we considered a problem where for a given S-box size we want to find a CA rule that maps to an S-box that is as good as possible with regards to the cryptographic properties. Next, we change our objective and assume that we already have an S-box and we want to obtain its CA rule representation. There are two obvious reasons why one would want to do this. The first reason is to check whether a certain S-box is expressible with a CA rule. The second reason is to obtain a combinatorial circuit representation of an S-box (in the case that the S-box can be represented with a CA rule). The first objective can be reached with another technique that we briefly explain.

Given the truth table description of an S-box, the task of determining the local rule of the corresponding CA can be determined using the De Bruijn graph representation [37]. The *De Bruijn graph* associated to a CA with local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is a directed graph $G = (V, E)$ where $|V| = 2^{d-1}$. In particular, each vertex in G is labeled with a binary vector of length $d - 1$. An edge from vertex $a \in V$ to $b \in V$ exists if and only if a and b *overlap* respectively on the last and the first $d - 2$ coordinates. For example, for $d = 3$ the De Bruijn graph has an edge from $a = 01$ to $b = 10$ since a and b have a 1 respectively in the last and in the first position. A CA local rule is represented over the De Bruijn graph as a labeling of the edges, i.e., a function $l : E \to \{0, 1\}$. Hence, in the example above the labeling of $(01, 10)$ would be the result of the local rule applied to the input 010. Figure 3 depicts the De Bruijn graph representation of the CA rule χ used in Keccak.

To check if a given S-box of length n can be expressed using a CA rule with diameter $d < n$, one could start from a De Bruijn graph with 2^{d-1} vertices and iteratively label the edges by reading the entries in the truth table of the S-box. As soon as an inconsistency is found (i.e., an edge gets more than one label), one knows that the S-box is not representable with a CA of diameter d. On the other hand, if after reading the whole S-box each edge has a unique label, then the De Bruijn

graph of a CA rule implementing that S-box is obtained.

As an example, we consider the APN function in dimension 6 [6]. Considering the last occurring value that equals 22, we see that this S-box cannot be generated with a CA rule. This is due to the fact that for the input 63 (111111 in binary) the output equals 22 (010110), which means that the local rule is not consistent because it assigns different values to the cells 1, 3, and 6 (value 0) and to the cells 2, 4, and 5 (value 1).

We note that the above procedure cannot help us to reach the second objective, i.e., to find a combinatorial representation of a given S-box. Additionally, this problem is much more difficult since there exist many circuits mapping to the same truth table and there is no easy way to determine the smallest circuit. In this case we can use a regression process in order to find combinatorial circuits for a given S-box. Here we apply the same evolutionary algorithm (GP) with the same functions and terminals as before. Note that the evolution is now guided with the fitness function that describes *the difference* between the S-box obtained by a CA rule, and the one given as an input parameter. The design of the objective function is such that the truth table output of the current CA rule is compared with the truth table of the given S-box.

Rather than only counting all the bits in which the two differ, we employ a two-stage fitness as in the previous section. In the first stage the number of differing bits is minimized, which is the primary objective. Only if the difference is zero, we add a term devoted to minimizing the size of the resulting CA rule (enforcing parsimony). This term is defined so it is inversely proportional to the size of the GP individual, in this case corresponding to a CA rule. The final fitness function in this case equals:

$$fitness_{re} = nErrors + \Delta_{nErrors,0}\left(\frac{treeSize}{maxTreeSize}\right), \tag{32}$$

where $nErrors$ denotes the number of differing bits, $treeSize$ is the actual tree size, and $maxTreeSize$ is the maximum size that the tree may assume given the maximum tree depth and the number of arguments of the GP functions. Note that this fitness measure is minimized; the correct CA rule will have a fitness in the range $[0, 1]$ which in that case depends only on the expression size.

In our experiments, we use as input S-boxes the previously evolved solutions with the best obtained properties. That way, we can be sure that the S-box can be represented with a CA rule, while trying to find an implementation with a smaller complexity. In Table 3 we give results for each S-box size. Column *Original size* gives the size of the target S-box used in the regression, and the other columns give statistics for the obtained results (here, column Min represents the best obtained solution). Note that we randomly selected target input S-boxes among those with the best obtained properties from the previous section.

We see that for all presented sizes our procedure is able to find much shorter CA rules than used in the original case. Therefore, this makes our methodology a viable option when the goal is to implement the S-box obtained via a CA rule

Table 3: Reverse engineering approach, Eq. (32)

S-box size	Original size	New size			
		Max	Min	Avg	Std dev
4×4	77	26	**11**	13.96	3.36
5×5	27	30	**9**	15.32	6.13
6×6	26	31	**13**	20.11	5.34
7×7	23	42	**13**	22.19	8.99

Table 4: Results for exhaustive search

n	Number of CA-based S-boxes	Number of bijective S-boxes	Number of optimal S-boxes
3	256	36	12
4	65 536	1 536	512
5	4 294 967 296	22 500 002	2 880

in hardware, since a shorter rule will mean smaller gate count and consequently a smaller area. As an interesting fact, we note that we also tried this approach with the Keccak S-box. Among obtained solutions there were several occurrences of the exact same CA rule as used in Keccak. When working with the 8×8 S-box size, our regression technique was unable to find any correct rule corresponding to the given S-box. We note that we experimented with an S-box originally obtained with a CA rule consisting of 177 primitives, which is a much longer rule when compared with the sizes where our approach found correct rules.

5.5 Equivalence Classes

In this section, we concentrate on S-boxes up to 5×5 size, i.e., those that can be exhaustively checked. First, in Table 4 we give results for sizes 3×3, 4×4, and 5×5. As it can be seen, from the corpus of possible CA-based S-boxes, only a fragment is bijective. Additionally, from the bijective S-boxes again only a small part is optimal. Here, by optimal we consider to have the largest possible nonlinearity and the smallest possible differential uniformity. We emphasize that this is the total number of CA-based S-boxes since other S-boxes that are affine equivalent to these cannot be obtained with a single CA rule.

For sizes larger than 5×5, an exhaustive search is not possible. Still, a simple estimation can be made. The total number of CA-based S-boxes equals the number of Boolean functions of the corresponding size, i.e., 2^{2^n}. Next, the number of balanced Boolean functions of size n equals $\binom{2^n}{2^{n-1}}$, which also represents a trivial upper bound on the number of bijective CA-based S-boxes. As an example, for size 5×5, the number of bijective S-boxes obtainable with a single CA rule forms only 26.7% of possible balanced Boolean functions of size 5.

When considering 3×3 size, we give details in Table 5. In Table 6 we give

Table 5: Equivalence classes of bijective 3×3 S-boxes

Class	Representative	Number of S-boxes	Optimal
0	0,1,2,3,4,5,6,7	6	No
1	0,1,2,3,4,5,7,6	6	No
2	0,1,2,3,4,6,7,5	12	No
3	0,1,2,4,3,6,7,5	12	Yes

Table 6: Equivalence classes of bijective 4×4 S-boxes

Class	Representative	Number of S-boxes	Optimal
0	F,D,B,9,7,5,3,1,E,C,A,8,6,4,2,0	16	No
1	0,1,2,3,4,5,6,7,8,9,A,B,C,D,F,E	32	No
3	0,1,2,3,4,5,6,7,8,9,A,B,D,E,F,C	32	No
4	0,1,2,3,4,5,6,7,8,9,A,B,D,C,F,E	16	No
6	0,1,2,3,4,5,6,7,8,9,A,C,B,D,F,E	32	No
9	0,1,2,3,4,5,6,7,8,9,A,C,D,E,B,F	64	No
41	0,1,2,3,4,5,7,6,8,A,9,C,B,F,E,D	128	No
193	0,1,2,3,4,5,8,A,6,C,7,F,D,B,9,E	128	No
270	0,1,2,3,4,6,8,B,5,C,9,D,E,A,7,F	128	Yes (G_4)
272	0,1,2,3,4,6,8,B,5,C,D,7,9,F,A,E	128	Yes (G_6)
273	0,1,2,3,4,5,8,A,6,C,7,F,E,B,9,D	128	No
278	0,1,2,3,4,6,8,B,5,C,D,7,A,F,9,E	128	Yes (G_5)
279	0,1,2,3,4,5,8,A,6,B,C,7,D,F,E,9	128	No
281	0,1,2,3,4,5,7,8,6,9,A,C,F,B,D,E	128	No
282	0,1,2,3,4,6,8,B,5,C,D,7,F,9,E,A	128	Yes (G_3)
288	0,1,2,3,4,5,6,7,8,9,C,E,F,B,D,A	32	No
289	0,1,2,3,4,5,6,7,8,9,C,E,B,F,D,A	64	No
291	0,1,2,3,4,5,7,6,8,A,9,B,C,F,E,D	64	No
294	0,1,2,3,4,5,6,7,8,9,B,A,E,F,D,C	32	No

details about equivalence classes of 4×4 S-boxes that are CA-based and bijective. For the 4×4 S-box size, Leander and Poschmann defined optimal S-boxes as those being bijective, with maximal nonlinearity (equal to 4), and minimal differential uniformity (again equal to 4) [22]. There are in total 16 non equivalent classes of S-boxes with such properties (denoted $G_0, \ldots, G_{15}$).

6 Discussion and Future Work

On the basis of the presented results one can observe that for dimensions from 4×4 up to 7×7 it is possible to find CA rules that result in S-boxes with very good cryptographic properties. Here, only 6×6 size remains suboptimal but the optimal solution as given by Dillon [6] (from the cryptographic perspective) is not even attainable with a single CA rule.

When considering 8×8 size, the obtained CA rules result in suboptimal S-boxes

which are even far from the results obtainable with heuristics using permutation encoding (see e.g., [28]). We see that the average sizes of 8×8 rules are significantly larger when compared with the other investigated dimensions, which certainly results in a more complicated evolution process. Besides the goal of evolving CA rules resulting in S-boxes with good cryptographic properties, we also changed the paradigm and looked for CA rules that map to a specific S-box. Again, results for sizes 4×4 up to 7×7 are very good, where up to the 6×6 size we have 100% success rate in obtaining correct rules. For the 7×7 size, that percentage equals 96.7%, which is still an excellent result. As with the previous goal, size 8×8 presents an unsurmountable obstacle where our approach did not succeed in obtaining any equivalent CA rule.

There are several options for future work. The most obvious one is concentrating on the 8×8 size and trying to improve the values of cryptographic properties. Naturally, in this paper we concentrated only on a small set of cryptographic properties and one could include in the fitness function other relevant properties like algebraic degree. As it can be seen in Table 2a, for the sizes 4×4 and 6×6 the best obtainable nonlinearity still equals the quadratic bound respectively when $d = 3$ and $d = 5$. Hence, it would be interesting to investigate whether in these cases our heuristic approach based on GP is still able to evolve S-boxes with optimal nonlinearity with $d = n - 1$, i.e. where the local rule depends on all input variables except one. From the theoretical side, another possible direction for future research is to investigate lower bounds on the nonlinearity and differential uniformity of CA S-boxes based on specific subclasses of local rules, such as plateaued Boolean functions. We note that this question has already been investigated in Mariot et al. [23] for permutive local rules. A local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is called *permutive* if it is defined as $f(x_1, \cdots, x_{d-1}, x_d) = g(x_1, \cdots, x_{d-1}) \oplus x_d$, where g is a function of $d - 1$ variables. Computer searches performed on small input size suggest that permutive rules always satisfy with equality the bound on nonlinearity given in Theorem 3. An example of permutive rule is the function χ used in the Keccak S-box. However, the authors of [23] later observed a mistake in the proof of this fact, and they are currently investigating either how to fix it or to disprove it.

The experiments presented in this paper focused on evolving CA as nonlinear elements for the confusion phase of a block cipher. Another interesting perspective would be to investigate the use of CA also for the diffusion phase. Since linear diffusion layers are often implemented in the literature using MDS linear codes, a possible venue for future research in this context is to optimize through GP the implementation cost of the MDS matrices arising from linear CA.

Recall that a (n, k, t) binary linear cyclic code C is a k-dimensional subspace of the vector space $\mathbb{F}_2^n$ such that each pair of vectors (called *codewords*) is at the Hamming distance at least t, which is also closed under cyclic shifts. On the other hand, a CA is called *linear* if its local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ is defined as an XOR of a subset of cells in the neighborhood. Consider now a NBCA of length $n = m + d - 1$ equipped with a linear local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$. In this case, the global rule of F is

defined by a $m \times (m + d - 1)$ *transition matrix* M_F of the following form:

$$M_F = \begin{pmatrix} a_1 & \cdots & a_d & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & a_1 & \cdots & a_d & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & a_1 & \cdots & a_d \end{pmatrix}. \tag{33}$$

The vectorial Boolean function defined by such CA is determined by the matrix-vector multiplication $y = M_F x^\top$.

One can notice that the transition matrix in Eq. (33) actually has the same form as the generator matrix of a cyclic code (see [24]). Hence, an interesting idea would be to use linear CA to implement MDS cyclic codes for diffusion layers in block ciphers. This would require first to characterize the systematic generator matrix S_F of the cyclic codes induced by a linear CA with transition matrix M_F as defined in Eq. (33). Consequently, one could employ the non-systematic part of S_F as a MDS matrix to implement a linear diffusion layer. It is known that these MDS matrices are not sparse [1]. Thus, a possible future work in this context could be to optimize through GP the implementation cost of the MDS matrices arising from linear CA.

Finally, it would be interesting to see how CA rules can be integrated into unbalanced MISTY constructions as presented by Canteaut et al. [8]. Since the aim there is to construct lightweight S-boxes of larger sizes, a procedure to obtain building blocks (i.e., smaller S-boxes) could be beneficial.

7 Related Work

Most of the block ciphers based on the dynamics of cellular automata focus on the use of *reversible* CA (RCA). A CA is reversible if its global rule $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is bijective and the inverse $G = F^{-1}$ is again the global rule of a CA. In a CA-based block cipher, the idea is to represent a block of plaintext as the initial configuration of the CA. The global rule is then applied for a certain number of steps to obtain the encrypted block. For decryption, the inverse global rule is applied for the same number of steps starting from the ciphertext block to recover the plaintext.

The first block cipher based on cellular automata was proposed by Gutowitz [19]. In particular, for the substitution phase *block CA* were used to ensure the invertibility of the resulting S-box. In a block CA, the local rule does not determine the next state of a single cell, but rather the state of a block of adjacent cells. The cellular array is partitioned in blocks of equal length, and then a permutation is applied to each block in parallel. In the next step, the partition is shifted one cell to the right with periodic boundary conditions.

A second type of CA which have been used for block ciphers are *second-order CA*, where the state of a cell is determined by XOR-ing its previous state with the result of the local rule. Hence, the configuration at time $t - 1$ can be computed by knowing both the configurations at time t and $t + 1$. Seredynsky et al. investigated

second-order CA as S-boxes, by assessing the *avalanche properties* of several rules with diameter $d = 5, 7$ and array lengths $n = 32, 64$ [35].

Another interesting kind of CA for block ciphers are the so-called *complementing landscapes cellular automata* (CLCA), where the state of a cell is flipped if and only if a pattern belonging to a specific landscape occurs in the surrounding cells. Daemen et al. studied CLCA for designing block ciphers, discovering the rule χ used in Keccak [16, 4]. In particular, This rule induces an invertible CA if the length n of the cellular array is odd.

From the EC perspective, we mention only several characteristic approaches, all of which use the permutation encoding. Clark et al. used the principles from the evolutionary design of Boolean functions to evolve S-boxes with the desired cryptographic properties for sizes up to 8×8 [14]. Burnett et al. used a heuristic method to generate MARS-like S-boxes [7]. With their approach, they were able to generate a number of S-boxes of appropriate sizes that satisfy all the requirements placed on a MARS S-box. Picek et al. used Cartesian Genetic Programming and Genetic Programming to evolve S-boxes and discussed how to obtain permutation based encoding with those algorithms [31]. Picek et al. presented an improved fitness function with which EC is able to find higher nonlinearity values for a number of S-box sizes [28]. Picek et al. also discussed how to use genetic programming to evolve cellular automata rules that in turn can be used to generate S-boxes with good cryptographic properties [29]. Finally, Picek et al. used the same genetic paradigm to evolve CA rules to be used in S-boxes but where the goal is not only cryptographic properties but also implementation perspective [30]. Interestingly, the results obtained in these two papers, where GP is used to evolve CA rules, outperform any other solutions obtained with heuristics for sizes 5×5 up to 7×7.

8 Conclusions

In this paper, we approach the problem of designing S-boxes with good cryptographic properties from a completely new perspective – we use cellular automata rules that are then mapped to S-boxes. We first show upper bounds for the nonlinearity and differential uniformity achievable by CA, both in the no boundary and periodic boundary settings. We then apply a heuristic approach to evolve CA rules inducing S-boxes with good cryptographic properties. Our approach shows great potential where this is the first time that evolutionary computation (or more generally, heuristic) techniques are able to find optimal S-boxes for sizes larger than 4×4. Our approach transforms a problem that has been up to now of extreme difficulty into a simpler problem for certain S-box sizes. Naturally, since not all S-boxes can be represented with a cellular automaton rule, our technique cannot be used to design all optimal S-boxes of the corresponding size. We are confident that the corpus of obtainable functions is still large enough to give a sufficient diversity in future block cipher designs. We note that the GP approach also offers easy handling of the resulting S-box latency and area when implemented in hardware,

which makes our methodology even more usable.

Finally, we show how one can use GP in order to "reverse-engineer" an S-box. There, we use the regression approach to find the shortest CA rule resulting in a specific S-box. This approach has interesting ramifications from two aspects: fast checking whether an S-box is expressible through CA rules and obtaining different rules (and consequently their sizes) resulting in a specific S-box.

References

[1] D. Augot and M. Finiasz. Direct construction of recursive MDS diffusion layers using shortened BCH codes. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 3–17, 2014.

[2] T. Bäck, D. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, 2000.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Radiogatún, a belt-and-mill hash function. *IACR Cryptology ePrint Archive*, 2006:369, 2006.

[4] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. The Keccak reference, January 2011. `http://keccak.noekeon.org/`.

[5] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '07, pages 450–466, Berlin, Heidelberg, 2007. Springer-Verlag.

[6] K. A. Browning, J. F. Dillon, M. T. McQuistan, and A. J. Wolfe. An APN permutation in dimension six. *Finite Fields: theory and applications*, pages 33–42, 2010.

[7] L. Burnett, G. Carter, E. Dawson, and W. Millan. Efficient Methods for Generating MARS-Like S-Boxes. In *Proceedings of the 7th International Workshop on Fast Software Encryption*, FSE '00, pages 300–314, London, UK, UK, 2001. Springer-Verlag.

[8] A. Canteaut, S. Duval, and G. Leurent. Construction of Lightweight S-Boxes Using Feistel and MISTY Structures. In O. Dunkelman and L. Keliher, editors, *Selected Areas in Cryptography - SAC 2015: 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 373–393, Cham, 2016. Springer International Publishing.

[9] C. Carlet. Boolean Functions for Cryptography and Error Correcting Codes. In Y. Crama and P. L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 257–397. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[10] C. Carlet. Vectorial Boolean Functions for Cryptography. In Y. Crama and P. L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 398–469. Cambridge University Press, New York, USA, 1st edition, 2010.

[11] F. Chabaud and S. Vaudenay. Links between differential and linear cryptanalysis. In A. De Santis, editor, *Advances in Cryptology — EUROCRYPT '94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, 1994 Proceedings*, pages 356–365. Springer Berlin Heidelberg, 1995.

[12] F. M. Christoph Dobraunig, Maria Eichlseder and M. Schläffer. Ascon, 2014. CAESAR submission, `http://ascon.iaik.tugraz.at/`.

[13] L. Claesen, J. Daemen, M. Genoe, and G. Peeters. Subterranean: A 600 Mbit/sec cryptographic VLSI chip. In *Computer Design: VLSI in Computers and Processors, 1993. ICCD '93. Proceedings., 1993 IEEE International Conference on*, pages 610–613, Oct 1993.

[14] J. A. Clark, J. L. Jacob, and S. Stepney. The design of S-boxes by simulated annealing. *New Generation Computing*, 23(3):219–231, Sept. 2005.

[15] J. Daemen and C. S. K. Clapp. Fast Hashing and Stream Encryption with PANAMA. In *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, pages 60–74, 1998.

[16] J. Daemen, R. Govaerts, and J. Vandewalle. Invertible shift-invariant transformations on binary arrays. *Applied Mathematics and Computation*, 62(2):259 – 277, 1994.

[17] J. Daemen, R. Govaerts, and J. Vandewalle. A new approach to block cipher design. In R. Anderson, editor, *Fast Software Encryption: Cambridge Security Workshop Cambridge, U. K.,1993 Proceedings*, pages 18–32, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[18] J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[19] H. Gutowitz. Cryptography with dynamical systems. In *Cellular Automata and Cooperative Systems*, pages 237–274. Springer, 1993.

[20] L. R. Knudsen and M. Robshaw. *The Block Cipher Companion.* Information Security and Cryptography. Springer, 2011.

[21] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[22] G. Leander and A. Poschmann. On the Classification of 4 Bit S-Boxes. In C. Carlet and B. Sunar, editors, *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computer Science*, pages 159–176. Springer Berlin Heidelberg, 2007.

[23] L. Mariot and A. Leporati. A cryptographic and coding-theoretic perspective on the global rules of cellular automata. *Natural Computing*, 2017.

[24] R. J. McEliece. *Theory of Information and Coding*. Cambridge University Press, New York, NY, USA, 2nd edition, 2001.

[25] K. Nyberg. 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*, pages 378–386. Springer, 1991.

[26] K. Nyberg. On the construction of highly nonlinear permutations. In R. Rueppel, editor, *Advances in Cryptology - EUROCRYPT' 92*, volume 658 of *Lecture Notes in Computer Science*, pages 92–98. Springer Berlin Heidelberg, 1993.

[27] K. Nyberg. S-boxes and round functions with controllable linearity and differential uniformity. In *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, pages 111–130, 1994.

[28] S. Picek, M. Cupic, and L. Rotim. A New Cost Function for Evolution of S-boxes. *Evolutionary Computation*, 2016.

[29] S. Picek, L. Mariot, A. Leporati, and D. Jakobovic. Evolving S-boxes Based on Cellular Automata with Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 251–252, New York, NY, USA, 2017. ACM.

[30] S. Picek, L. Mariot, B. Yang, D. Jakobovic, and N. Mentens. Design of S-boxes Defined with Cellular Automata Rules. In *Proceedings of the Computing Frontiers Conference*, CF'17, pages 409–414, New York, NY, USA, 2017. ACM.

[31] S. Picek, J. F. Miller, D. Jakobovic, and L. Batina. Cartesian Genetic Programming Approach for Generating Substitution Boxes of Different Sizes. In *GECCO Companion '15*, pages 1457–1458, New York, NY, USA, 2015. ACM.

[32] R. Poli, W. B. Langdon, and N. F. McPhee. *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.

[33] R. Poli, W. B. Langdon, and N. F. McPhee. *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).

[34] V. Rijmen, P. S. L. M. Barreto, and D. L. G. Filho. Rotation symmetry in algebraically generated cryptographic substitution tables. *Inf. Process. Lett.*, 106(6):246–250, 2008.

[35] M. Seredynski and P. Bouvry. Block encryption using reversible cellular automata. In *Cellular Automata, 6th International Conference on Cellular Automata for Research and Industry, ACRI 2004, Amsterdam, The Netherlands, October 25-28, 2004, Proceedings*, pages 785–792, 2004.

[36] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[37] K. Sutner. De bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991.