http://dx.doi.org/10.1145/3075564.3079069

# Design of S-boxes Defined with Cellular Automata Rules

Stjepan Picek[1], Luca Mariot[2], Bohan Yang[1], Domagoj Jakobovic[3], and Nele Mentens[1]

[1]KU Leuven, imec-COSIC, Kasteelpark Arenberg 10 3001 Leuven, Belgium ,
stjepan@computer.org, bohan.yang@esat.kuleuven.be,
nele.mentens@kuleuven.be
[2]Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Viale Sarca 336, 20126 Milano, Italy ,
luca.mariot@disco.unimib.it
[3]University of Zagreb, Unska 3, 10000 Zagreb, Croatia ,
domagoj.jakobovic@fer.hr

**Abstract**

The aim of this paper is to find cellular automata (CA) rules that are used to describe S-boxes with good cryptographic properties and low implementation cost. Up to now, CA rules have been used in several ciphers to define an S-box, but in all those ciphers, the same CA rule is used. This CA rule is best known as the one defining the Keccak $\chi$ transformation. Since there exists no straightforward method for constructing CA rules that define S-boxes with good cryptographic/implementation properties, we use a special kind of heuristics for that – Genetic Programming (GP). Although it is not possible to theoretically prove the efficiency of such a method, our experimental results show that GP is able to find a large number of CA rules that define good S-boxes in a relatively easy way. We focus on the $4 \times 4$ and $5 \times 5$ sizes and we implement the S-boxes in hardware to examine implementation properties like latency, area, and power. Particularly interesting is the internal encoding of the solutions in the considered heuristics using combinatorial circuits; this makes it easy to approximate S-box implementation properties like latency and area a priori.

***Keywords*** Lightweight cryptography, S-boxes, Cellular automata, Genetic programming, Implementation

## 1 Introduction

When studying the design of block ciphers, a common concept is to adhere to Shannon's confusion and diffusion principles [23]. To provide confusion,

one usually uses Substitution boxes (S-boxes). In most cases, those S-boxes are the only nonlinear part of the cipher and they operate on parts of the state. Smaller S-boxes, e.g. those with size $4 \times 4$, are most often employed in lightweight cryptographic ciphers such as PRESENT [5]. On the other hand, $8 \times 8$ S-boxes are found in the AES cipher [16], and subsequently in a number of ciphers that draw inspiration from AES.

Naturally, there exists a clear trade-off between the choices of S-box sizes and properties. S-boxes based on a finite field inversion will have the smallest possible differential probability, the largest possible algebraic degree, and the largest possible nonlinearity (or what is believed to be the largest possible nonlinearity in the case of a bijective $8 \times 8$ S-box). Those properties ensure that the cipher can be secure with a relatively small number of rounds and consequently have good performance However, an S-box of such a size can be troublesome to implement in constrained environments and consequently one uses either smaller optimal S-boxes, large S-boxes with suboptimal cryptographic properties but small implementation cost or constructions of larger S-boxes through smaller S-boxes.

To construct any of the aforementioned S-boxes, algebraic constructions with possible additional affine transformations are usually employed to change the representation of an S-box without affecting its cryptographic properties. This concept can be levied for the $4 \times 4$ size since there it is also possible to use an exhaustive search in order to obtain all optimal S-boxes. Besides that, there have been numerous papers advocating the use of heuristic techniques. Nevertheless, the results are mostly suboptimal with regards to cryptographic properties and they do not describe a proper use case [12, 8, 20]. Finally, there are several ciphers that use cellular automata (CA) rules to describe the S-box. The best known example is the Keccak sponge construction [4] that is now the SHA-3 standard. There, the authors use CA rules affecting only two neighborhood positions for each bit, which results in an extremely lightweight definition of the S-box and a small implementation cost. However, that S-box has suboptimal cryptographic properties (like nonlinearity and differential uniformity), which results in a cipher with more rounds than with optimal S-boxes. As far as the authors know, all the other ciphers using CA rules for the S-box definition actually use the same rule. This is the case in the Panama [13], RadioGatún [3], Subterranean [11], and 3Way [15] ciphers. It is also an interesting consideration (although maybe more in the philosophical domain) whether those ciphers (excluding 3Way) actually use S-boxes, since every output bit depends only on three input bits for the considered CA rule. Taking this into account, the Panama cipher would have a 17-bit S-box and Subterranean would have a 257-bit S-box. We note that since the same local rule is used in different S-box sizes, the cryptographic properties actually degrade with the increase of the S-box size. For instance, when used in $3 \times 3$ S-boxes, both the nonlinearity and differential uniformity equal 2, which is optimal, but if used in $7 \times 7$ S-boxes, both of those properties would have a

value equal to 32, which is far from optimal.

In this paper, we focus on the investigation of cellular automata rules that are able to produce optimal S-boxes with a low implementation cost. In order to achieve that, we employ a heuristic technique known as Genetic Programming (GP) to evolve CA rules – definitions of S-boxes. Genetic Programming evolves tree structures that consist of logical operations and therefore offer a natural mapping to combinatorial circuits which makes it possible to estimate the area of the generated S-boxes. Thanks to the maximum tree depth parameter of GP, we are also able to limit the latency of the S-boxes. Finally, to evaluate the power consumption, we use a posteriori setup where we batch large numbers of CA rules to find which one coincides with the smallest power consumption.

In our analysis, we focus on S-boxes of sizes $4 \times 4$ and $5 \times 5$. While larger S-boxes with optimal cryptographic properties can be also designed by using our methodology, the initial obtained results show that such CA rules are usually too large to be of real interest when implemented in hardware. We are aware that one could argue that there are already enough knowledge and tools to design smaller S-boxes, but as far as we know, there is nothing more than the Keccak rule when considering CA rules for the construction of S-boxes. We emphasize that the way the CA rule is implemented, i.e., one rule for each bit of the input, make CA rules a very interesting technique but one that leads to large implementations for larger sizes. Indeed, if a single rule needs only 5 $GE$, the overhead for $5 \times 5$ S-boxes is 25 $GE$.

Additionally, we remark that the number of S-boxes of a certain size defined by CA is much smaller than the total number of S-boxes of that size, since a CA S-box is described just by the Boolean function of its local rule. This means that there are $2^{2^n}$ CA S-boxes of size $n \times n$, a space which could be exhaustively searched up to $n = 5$. However, since we use tree encoding to measure the implementation cost (see Sec. 5), the number of CA rules representations is much larger than the number of S-boxes and thus impossible to exhaustively visit even for smaller sizes, which motivates the use of heuristic techniques such as GP.

## 2 Cryptographic Properties of S-boxes

Let $n, m$ be positive integers, i.e., $n, m \in \mathbb{N}^+$. The set of all $n$-tuples of elements in the field $\mathbb{F}_2$ is denoted as $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the Galois field with two elements. The inner product of two vectors $a$ and $b$ equals $a \cdot b = \bigoplus_{i=1}^{n} a_i b_i$ and "$\bigoplus$" represents addition modulo two. An $(n, m)$-function is any mapping $F$ from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. A $(n, m)$-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. The following results are well-known in the theory of Boolean functions and S-boxes and can be found for instance in [9, 10, 18, 19].

The Walsh-Hadamard transform of an $(n, m)$-function $F$ equals:

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

An $(n, m)$-function $F$ is *balanced* (BAL) if it takes every value of $\mathbb{F}_2^m$ the same number $2^{n-m}$ of times.

The *nonlinearity* $N_F$ of an $(n, m)$-function $F$ equals the minimum nonlinearity of all its component functions $v \cdot F$, where $v \in \mathbb{F}_2^m \setminus \{\underline{0}\}$:

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

The nonlinearity of an S-box should be as high as possible in order to avoid *linear attacks*. The maximal nonlinearity for any $(n, n)$ function $F$ is bounded above by:

$$N_F \leq 2^{n-1} - 2^{\frac{n-1}{2}}. \tag{3}$$

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

$$D(a, b) = |\{x \in \mathbb{F}_2^n : F(x + a) + F(x) = b\}|. \tag{4}$$

The entry at the position $(a, b)$ corresponds to the cardinality of $D(a, b)$ and is denoted as $\delta(a, b)$. The *differential uniformity* $\delta_F$ is then defined as:

$$\delta_F = \max_{a \neq 0, b} \delta(a, b). \tag{5}$$

In order to withstand *differential attacks*, the differential uniformity of an S-box needs to be as low as possible.

The algebraic degree $deg_F$ of $(n, m)$-function $F$ is the maximum algebraic degree of all component functions [10]. As a cryptographic criterion, the degree of an S-box should be as high as possible in order to thwart *higher-order differential attacks*.

The *branch number* $b_F$ of a function $F$ is defined as [16]:

$$b_F = \min_{a, b \neq a} (HW(a \oplus b) + HW(F(a) \oplus F(b))). \tag{6}$$

## 3 Cellular Automata

A cellular automaton (CA) is a parallel computational model that has 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.

For the purposes of our work, we consider only the case in which each cell is described by a binary state, 0 or 1, leading to the following definition: An infinite *cellular automaton* (CA) is a quadruple $A = \langle c, \delta, \omega, f \rangle$ where $c$ is a bi-infinite array of *cells*, each of which takes a value in $\mathbb{F}_2$, $\delta \in \mathbb{N}$ is the *diameter*, $\omega \in \mathbb{Z}$ is the *offset* and $f : \mathbb{F}_2^\delta \to \mathbb{F}_2$ is the *local rule*. In particular, the next state of each cell $c_i$ with $i \in \mathbb{Z}$ is determined by applying in parallel the rule $f$ to the neighborhood $(c_{i-\omega}, \cdots, c_{i-\omega+\delta-1})$.

Notice that, since the local rule $f : \mathbb{F}_2^\delta \to \mathbb{F}_2$ is a Boolean function, it can be represented by a truth table of $2^\delta$ bits. The *global rule* $F : \mathbb{F}_2^\mathbb{Z} \to \mathbb{F}_2^\mathbb{Z}$ of an infinite CA $\langle c, \delta, \omega, f \rangle$ is the function mapping the current state of the bi-infinite array $c$ to its next configuration. An important property characterizing the global rules of CA is their *shift invariance*. Denoting by $\sigma$ the function which shifts all values in a bi-infinite configuration one place to the left, any CA global rule commutes with $\sigma$, i.e. $F(\sigma(x)) = \sigma(F(x)$ for all $x \in \mathbb{F}_2^\mathbb{Z}$.

For practical applications, CA can obviously be implemented using only finite arrays, which leads to the problem of updating the cells at the boundaries. One of the most commonly adopted policies are *periodic boundary conditions*, where the finite cells array is viewed as a ring with the last cell preceding the first one. In this case, we denote a finite CA by a quintuple $\langle c, n, \delta, \omega, f \rangle$, where $n \in \mathbb{N}$ indicates the length of the cellular array. The global rule $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is thus defined for all possible values $x \in \mathbb{F}_2^\mathbb{Z}$ of the state array $c$ as $F(x) = (f(x_{-\omega}, \cdots, x_{\delta-\omega}), \cdots, f(x_{n-1-\omega}, \cdots, x_{n-\omega+\delta-2}))$ where all indices are computed modulo $n$. As a consequence, the global rule of a finite CA is a vectorial Boolean function of $n$ inputs and $n$ outputs, where for all $i \in \{0, \cdots, n-1\}$ the $i$-th coordinate function $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ corresponds to the local rule $f$ applied to the neighborhood of cell $c_i$. As an example, consider the rule $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ and the finite configuration $c = 01001$. The application of the global rule $F : \mathbb{F}_2^5 \to \mathbb{F}_2^5$ to $c$ using periodic boundary conditions changes the CA state to $F(c) = 01111$.

A finite CA with periodic boundary conditions corresponds to an infinite CA with the same local rule, but restricted only to *periodic configurations*, i.e. configurations $x \in \mathbb{F}_2^\mathbb{Z}$ such that $x_{i+h} = x_i$ for a certain $h$ and for all $i \in \mathbb{Z}$.

We conclude this section by mentioning the class of *reversible* CA (RCA), which is particularly interesting for cryptographic applications. Formally, an infinite CA $A = \langle c, \delta, \omega, f \rangle$ is reversible if its global rule $F : \mathbb{F}_2^\mathbb{Z} \to \mathbb{F}_2^\mathbb{Z}$ is bijective and the inverse $G = F^{-1}$ is again the global rule of an infinite CA. In practice, as Richardson [22] proved, the reversibility of an infinite CA is characterized just by the bijectivity of its global rule. Since an infinite RCA $A = \langle c, \delta, \omega, f \rangle$ is clearly reversible over the set of periodic configurations, it

follows that the global rule of the corresponding finite CA $A' = \langle c, n, \delta, \omega, f \rangle$ is a bijective $S$-box. The converse is however not true: a local rule $f : \mathbb{F}_2^\delta \to \mathbb{F}_2$ may give rise to a finite CA whose global rule is invertible only for certain array lengths, but the associated infinite CA is not reversible.

# 4 Related Work

The first block cipher based on cellular automata was proposed by Gutowitz [17]. The design was based on the adoption of *permutive* rules for the diffusion phase, in particular, where the value of the leftmost or rightmost variable is XORed with a function of the remaining variables, and *block reversible* CA for the confusion phase, where a permutation is applied on smaller blocks of the cipher state which are then shifted in the next time steps.

Gutowitz's design focused on the *iterated behaviour* of the CA, where the global rule is applied for several time steps in order to generate the ciphertext. Another perspective is to consider the S-box corresponding only to one CA iteration. This approach has been mainly investigated with respect to the class of *complementing landscapes cellular automata* (CLCA), where the local rule complements the state of the current cell if the values of the surrounding cells belong to a specific pattern. In [14], CLCA for block cipher design were studied distinguishing between *locally* and *globally* invertible rules. In particular, all locally invertible CLCA turn out to be *involutions*, while globally invertible CLCA are invertible only over certain sets of periodic configurations. In particular, the transformation $\chi$ used in Keccak [4] is invertible only over configurations of odd lengths.

From the evolutionary computation (EC) perspective, we mention only some characteristic approaches that are all using permutation encoding. Clark et al. used the principles from the evolutionary design of Boolean functions to evolve S-boxes with the desired cryptographic properties [12]. They used simulated annealing (SA) coupled with the hill climbing algorithm to evolve bijective S-boxes with high nonlinearity. Burnett et al. used a heuristic method to generate MARS-like S-boxes [8]. 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 several evolutionary algorithms to evolve S-boxes and discussed how to obtain permutation-based encoding with those algorithms [20].

# 5 Experimental Setup and Results

## 5.1 Genetic Programming Approach

Genetic Programming (GP) is an evolutionary algorithm in which the data structures that undergo optimization are computer programs. Since the

aim of GP is to automatically generate new programs, each individual of a population represents a computer program, where the most common 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). Both functions and terminals are known as primitives.

Consider the Keccak construction that uses the CA rule $\chi$ on an array of length $n = 5$ for the nonlinear part. This rule can be represented as:

$$c_i(t+1) = c_i(t)\ XOR\ ((NOT(c_{i+1}(t)))\ AND\ c_{i+2}(t)), 0 \leq i < 5 \text{ and } t \in \mathbb{N}. \quad (7)$$

The above rule is applied to the current state in step $t$ to produce the next state in step $t+1$. Note that Eq. (7) is in fact a Boolean function which defines a CA local rule with offset $\omega = 0$.

We use the same approach with GP, where the task is to evolve a Boolean function of $n$ variables, in the form of a tree, which is used as a CA local rule. Similarly to the Keccak rule $\chi$, in our experiments we assume that the offset of the finite CA is $\omega = 0$. Hence the neighborhood of a cell is formed by the cell itself and the $n-1$ cells to its right. Unlike $\chi$, however, we also assume that the length of the CA equals the number of variables of the CA local rule. 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 represented as GP terminals, where the terminal $c_0$ denotes the value that is being updated. The terminals $c_1, ..., c_{n-1}$ denote the cells to the right of the current cell.

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 bits to determine the next state. The obtained global rule represents a candidate S-box that is then evaluated according to the fitness function. The function set consists of several Boolean primitives necessary to represent any Boolean function. Here, we use the following function set: NOT, which inverts its argument, XOR, NAND, NOR, each of which take 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 represents the multiplexer gate (MUX). In the evolution process, GP uses a 3-tournament selection, where the worst of the 3 randomly selected individuals is eliminated. A new individual is then created by applying crossover to the remaining two from the tournament. The new individual is then mutated with a probability of 0.5.

The variation operators are simple tree crossover and subtree mutation [21]. In simple tree crossover, randomly selected branches are exchanged

between two parent trees to create offspring while subtree mutation selects a node in the parse tree and replaces the branch at that node by a randomly generated branch. All our experiments suggest that having a maximum tree depth equal to the size of S-box (i.e., $n$) is sufficient. The initial population is created uniformly at random and every experiment is repeated 100 times.

In order to examine the influence of the GP parameters, we conducted a tuning phase for the stopping criterion and the population size. The starting set of parameters was tested on S-boxes of size $n = 5$, with population 200, for which 100 runs were executed. Based on these results, the stopping criterion was set to 500 000 evaluations since no change of the best solution was detected afterwards. Furthermore, we investigated population sizes 100, 200, 500, and 1 000 on the same problem size; although there were no significant differences, the best results were obtained with a population size of 500, which we used in the following experiments.

### 5.2 Fitness Function

We try to find $(n, n)$ S-boxes that possess the minimal necessary properties to be used in real world ciphers. Therefore, we want the evolved S-boxes to be balanced, with high nonlinearity, and low differential uniformity. We note that those are the standard minimum properties one should consider, although there are other properties that are important, but out of the scope of this work.

With the goal of finding balanced S-boxes that have as high as possible nonlinearity and as low as possible differential uniformity, we use a two-stage fitness function. First, the balancedness is verified, and if an S-box is balanced, we give it a value of zero, 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. Additionally, to reduce the implementation complexity of the evolved S-box, we aim to reduce both the number of elements in the CA rule and the approximate circuit area. Therefore, for every function that may be used in a GP individual, we define an *implementation weight* using the $GE$ measure, which stands for Gate Equivalent (i.e., the number of equivalent NAND gates in the specified technology). This weight reflects the relative area of those functions as follows: the weights of NAND and NOR gates are set to 1, the XOR weight is 2, the weight of IF is 2.33 and the weight of NOT equals 0.667 (note that the weights can be easily modified to reflect different hardware properties). Finally, we aim to **maximize** the resulting value:

$$fitness = BAL + \Delta_{BAL,0}(N_F + (2^n - \delta_F)) + 1/area\_penalty. \quad (8)$$

Here, $\Delta_{BAL,0}$ represents the Kronecker delta function that equals one when the function is balanced (i.e., $BAL = 0$) and zero otherwise. Further-

http://dx.doi.org/10.1145/3075564.3079069

more, we give equal weights to both $N_F$ and differential uniformity since our experiments show there is no statistically significant difference between those two approaches and no weight factors means less tuning. The *area_penalty* term is the summed weight of all the used functions. We opted not to use some sort of a multi-objective approach since we consider balancedness as a constraint and we are not interested in solutions if they are not balanced.

### 5.3 Experimental Results

In Table 1 we give the best obtained values for the cryptographic properties for both S-box sizes. In the column *Rule*, we give the rule that defines the specific S-box. Note that for the $5 \times 5$ size we are able to find AB function. Besides the cryptographic properties used in the fitness function, we also give results for the branch number ($b_F$), algebraic degree ($deg_F$), and the algebraic degree of the inverse S-box ($deg_F^{-1}$). Note that the results for $b_F$, $deg_F$, and $deg_F^{-1}$ could be improved by including them into fitness function, but we leave that for the future work.

After the evolution, the best obtained results are implemented in hardware to examine their performance. We use a clock frequency of 10 MHz because the dynamic power and the cell leakage power have similar orders of magnitude for this frequency for the technology used in this paper. This enables us to optimize both shares of the power at the same time. Furthermore, for a fixed clock frequency and computation time, optimizing for energy is the same as optimizing for average power. The power consumption of the S-boxes is estimated by means of simulation. In the first step of our simulation setup, an S-box is generated in the style of a look-up table (LUT). A Matlab (R2014b) script is used to generate the HDL description of the S-box (Verilog file *S-box.v*). For logic synthesis, we use a standard cell approach using the NANGATE 45 nm open cell library (PDKv1_3_v2010_12). Synopsys Design Compiler (I-2013.12) is used to produce the gate-level netlist and the delay file (*.sdf*). The standard method for estimating the power consumption using the Synopsys tool chain is based on the random switching activity of the internal nodes. While this approach may be suitable for first-order estimation, it does not give realistic application-specific data. In order to obtain a more realistic estimation, one needs to use a real testbench to approximate the switching activity for each gate. We use a testbench that goes through all possible $n \times (n-1)$ input transitions of the S-box. Modelsim SE PLUS 6.6d is used to simulate the wave file (*.vcd*) containing the switching activity of all nodes. This file is then converted to an activity file (.saif) using vcd2saif (D-2010.06-SP2). Finally, Design Compiler is used to estimate the power consumption. Similarly, the area cost estimation is based on the netlist before placement and routing. The area consumption of the S-box is estimated by the Synopsys tool chain and represented with the unit *GE*.

Table 1: The best obtained properties.

| Size | $N_F$ | $deg_F$ | $deg_F^{-1}$ | $\delta_F$ | $b_F$ | Rule |
|---|---|---|---|---|---|---|
| 4 × 4 | 4 | 3 | 3 | 4 | 2 | IF(((v3 NOR v1) XOR v0), v2, v1) |
| 5 × 5 | 8 | 2 | 3 | 8 | 2 | ((v2 NOR NOT(v4)) XOR v1) |
| 5 × 5 | 8 | 2 | 3 | 4 | 2 | ((v4 NAND (v2 XOR v0)) XOR v1) |
| 5 × 5 | 12 | 2 | 3 | 2 | 2 | (IF(v1, v2, v4) XOR (v0 NAND NOT(v3))) |

We emphasize that we present rules only for sizes up to $5 \times 5$ due to the lack of space (but also the implementation cost of such rules). For larger sizes, our technique is still able to find S-boxes with optimal cryptographic properties, but the logical complexity is high (for example, for $7 \times 7$ the shortest CA rule we found requires 31 $GE$ which makes the whole S-box 217 $GE$ large, a result which is much larger than the smallest known implementation for the AES S-box of 128 $GE$ [6]).

In Table 2 we present the implementation costs for a number of reference S-boxes as well as S-boxes defined with CA rules. The column $DPow$ denotes the dynamic power and the column $LPow$ denotes the cell leakage power.

In Figure 1 we depict the rule obtained with our approach that results in a $5 \times 5$ S-box with the same cryptographic properties (and similar implementation cost) as Keccak.

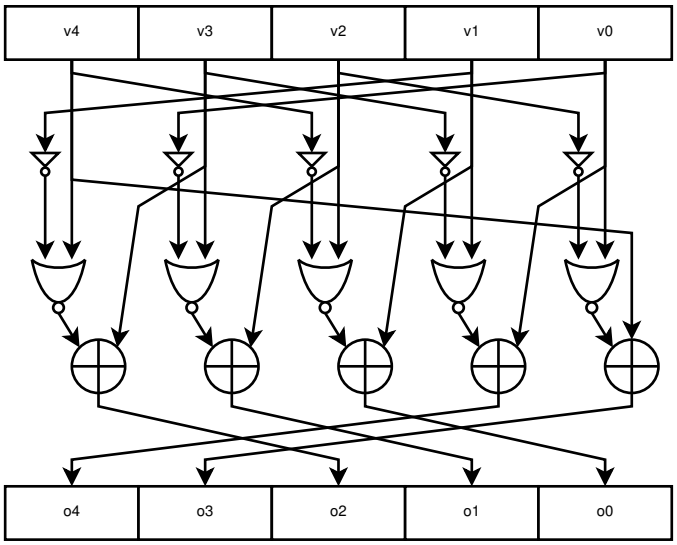


Figure 1: $5 \times 5$ rule with optimal nonlinearity and differential uniformity.

Table 2: Implementation results, power is in $nW$, area in $GE$, and latency in $ns$.

| Size | $4 \times 4$ | Rule | PRESENT | | |
|---|---|---|---|---|---|
| DPow. | 470.284 | LPow: | 430.608 | Area: | 22.67 | Latency:0.27 |
| Size | $4 \times 4$ | Rule | Piccolo | | |
| DPow. | 222.482 | LPow: | 215.718 | Area: | 12 | Latency:0.25 |
| Size | $4 \times 4$ | Rule | IF(((v3 NOR v1) XOR v0), v2, v1) | | |
| DPow. | 242.52 | LPow: | 337.47 | Area: | 16.67 | Latency:0.14 |
| Size | $5 \times 5$ | Rule | Keccak | | |
| DPow. | 321.684 | LPow: | 299.725 | Area: | 17 | Latency:0.14 |
| Size | $5 \times 5$ | Rule | ((v2 NOR NOT(v4)) XOR v1) | | |
| DPow. | 324.849 | LPow: | 308.418 | Area: | 17 | Latency:0.14 |
| Size | $5 \times 5$ | Rule | ((v4 NAND (v2 XOR v0)) XOR v1) | | |
| DPow. | 446.782 | LPow: | 479.33 | Area: | 24.06 | Latency:0.2 |
| Size | $5 \times 5$ | Rule | (IF(v1, v2, v4) XOR (v0 NAND NOT(v3))) | | |
| DPow. | 534.015 | LPow: | 493.528 | Area: | 26.67 | Latency:0.17 |

### 5.4 Discussion and Future Work

The results suggest that the most natural size for CA rules is $5 \times 5$ since the implementation properties have the smallest relative overhead. This is due to the fact that we always require at least several logical functions to build a CA rule for an $n$-bit S-box. With a small modification (e.g., adding just one logical gate) it is possible to reach optimal S-boxes for size $n + 1$. Naturally, for smaller S-box sizes, one could do an exhaustive search (as done for the $4 \times 4$ size), but translating such obtained S-boxes to CA rules is far from trivial for several reasons. The first reason is that not all S-boxes can be expressed with only a single CA rule. The second reason stems from the fact that each coordinate function (i.e., each Boolean function) can be expressed with a number of different CA rules.

Larger sizes of S-boxes, e.g., $7 \times 7$ and $8 \times 8$ result in relatively long rules (inefficient from the implementation perspective), so we refrain from giving a detailed analysis. Recall that when applying a single rule for a number of times, even if that rule is relatively efficient (e.g., small), the total implementation cost of the S-box can be high.

On a more general level, we notice that using the MUX gate (i.e., IF function) results in much smaller optimal S-boxes compared to cases where MUX is not allowed. This makes it worthwhile to use MUX gates, even though they have a somewhat larger area compared to the other gates we use. As already said, generating CA rules in the form of depth constrained

http://dx.doi.org/10.1145/3075564.3079069

trees has the additional advantage that we can control the maximal latency of the circuit, but naturally this control is somewhat coarse and there are many possible latency values one can reach for the same tree depth.

For the area results we follow the approach where we try to minimize the number of gates that constitute a CA rule as well as to use the "cheapest" gates based on weight factors expressed in terms of $GE$. Still, we can give only an approximation of results since the synthesis process can result in some differences.

For example, the rules $IF(v0, ((v3\ NOR\ v1)\ XOR\ v2), v1)$ and $IF(((v3\ NOR\ v1)\ XOR\ v0), v2, v1)$ have the same number and type of gates, but the area is equal to $18.05GE$ for the former and equal to $16.67GE$ for the latter. This difference stems from the fact that after the synthesis process the first rule actually uses 14 gates and the second one only 13 gates.

We only used power analysis as an a posteriori approach where we tested the obtained S-boxes for their dynamic and cell leakage power. Accordingly, the results are not comparable with for instance those obtained for the MIDORI cipher [1], but are better than for PRESENT. The implementation costs of larger S-boxes (i.e., $7 \times 7$ and larger) is omitted here since it has been shown that the most power efficient size is $4 \times 4$ [2]. We note that it would also be possible to run heuristics where the obtained CA rules would be immediately investigated for their power consumption which would result in better results than those presented here. To introduce even more diversity into CA rules, we could use the switching technique [7]. There, we can exchange one or more coordinate functions (corresponding to a CA rule) with a new rule. In that way, we could improve the cryptographic properties of S-boxes, find S-boxes with the same cryptographic properties but smaller or more power efficient when implemented or just obtain S-boxes not possible to design with only a single rule. For instance, we can take the Keccak rule and use it on the first 4 input bits. However, on the last input bit we would then use the rule $(v0\ XOR\ (v4\ NOR\ NOT(v3)))$ which enables us to obtain an S-box with the same cryptographic properties but utilizing a different set of input bits (see Eq. (7)).

## 6 Conclusions

In this paper, we use Genetic Programming to evolve CA rules that define S-boxes. The results show that our approach is able to generate a large number of rules, resulting in S-boxes that vary from having good cryptographic properties to being optimal, all with low implementation cost. We emphasize $5 \times 5$ as the best size, since it seems to offer the best trade-off between the minimal number of gates necessary to define a CA rule and the number of gates needed to define an optimal $5 \times 5$ S-box. Indeed, our best result for the $5 \times 5$ size has an area of 26 $GE$, which is 9 $GE$ more than the Keccak

S-box, but 10 *GE* less than the PRIMATEs S-box, which is an example of an S-box with the same cryptographic properties.

Furthermore, we are able to generate a large number of rules where we can set the desired values of cryptographic properties which gives the potential designer more choice when selecting appropriate S-boxes. Our technique can also be applied for larger S-box sizes where we are still able to find S-boxes with optimal values of nonlinearity and differential uniformity but then the implementation costs can be significantly higher. Finally, we are confident that some of our CA rules can offer interesting options for future designs of block ciphers.

## Acknowledgments

This work has been supported in part by Croatian Science Foundation under the project IP-2014-09-4882. In addition this work was supported in part by the Research Council KU Leuven: C16/15/058.

## References

[1] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni. Midori: A Block Cipher for Low Energy (Extended Version). Cryptology ePrint Archive, Report 2015/1142, 2015.

[2] S. Banik, A. Bogdanov, and F. Regazzoni. Exploring Energy Efficiency of Lightweight Block Ciphers. 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 178–194, Cham, 2016. Springer International Publishing.

[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.

[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] J. Boyar and R. Peralta. A Small Depth-16 Circuit for the AES S-Box. In D. Gritzalis, S. Furnell, and M. Theoharidou, editors, *Information Security and Privacy Research: 27th IFIP TC 11 Information Security*

*and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, 2012. Proceedings*, pages 287–298. Springer Berlin Heidelberg, 2012.

[7] L. Budaghyan, C. Carlet, and G. Leander. Constructing new APN functions from known ones. *Finite Fields and Their Applications*, 15(2):150 – 159, 2009.

[8] 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.

[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] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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).

[22] D. Richardson. Tessellations with local transformations. *Journal of Computer and System Sciences*, 6(5):373 – 388, 1972.

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