# Design of S-boxes Defined with CA Rules

CF 2017 / Mal-IoT – Siena

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

[1] KU Leuven, imec-COSIC, Belgium

[2] DISCo, Università degli Studi Milano - Bicocca, Italy

[3] University of Zagreb, Croatia

luca.mariot@disco.unimib.it

May 15, 2017

# Introduction

- S-boxes are crucial components in block ciphers
- Cellular Automata (CA) represent an interesting framework for designing S-boxes
- Most known example of CA-based S-box: $\chi$ transform, used for instance in KECCAK [Keccak11]
- Goal: Find CA rules which induce S-boxes with good cryptographic and implementation properties

# Boolean Functions and S-boxes

- Boolean function: a mapping $f : \mathbb{F}_2^n \to \mathbb{F}_2$, where $\mathbb{F}_2 = \{0, 1\}$
- $(n, m)$– function (or S-box): a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$
- Each output coordinate of $F$ is described by a *coordinate function* $F_i : \mathbb{F}_2^n \to \mathbb{F}_2$
- Component function: given $v \in \mathbb{F}_2^m \setminus \{\underline{0}\}$ and $x \in \mathbb{F}_2^n$,

$$v \cdot F = v_1 \cdot F_1(x) \oplus \cdots \oplus v_m \cdot F_m(x)$$

where $\cdot$ is the logical AND while $\oplus$ is the XOR

# Cryptographic Properties of $(n, m)$–Functions (1/2)

- Balancedness: for each output $y \in \mathbb{F}_2^m$, exactly $2^{n-m}$ input values map to $y$ under $F$
- Balanced $(n, n)$–functions $\Leftrightarrow$ bijective S-boxes
- Walsh Transform of $F$:

$$W_F(a, v) = \sum_{x \in \mathbb{F}_2^m} (-1)^{v \cdot F(x) \oplus a \cdot x}, \; a \in \mathbb{F}_2^n, \; v \in \mathbb{F}_2^m \setminus \{\underline{0}\}.$$

- Nonlinearity: minimum Hamming distance of $F$ from all affine functions:

$$N_F = 2^{n-1} - \frac{1}{2} \max_{a \in \mathbb{F}_2^n, \; v \in \mathbb{F}_2^m \setminus \{\underline{0}\}} \left| W_F(a, v) \right|.$$

# Cryptographic Properties of $(n, m)$–Functions (2/2)

- $F$ is $\delta$-Differential Uniform iff:

$$|\{x \in \mathbb{F}_2^n : F(x \oplus a) \oplus F(x) = b\}| \leq \delta, \ \forall a \in \mathbb{F}_2^n \setminus \{\underline{0}\}, \ b \in \mathbb{F}_2^m$$

- Algebraic Degree: maximum algebraic degree of the component functions of $F$
- The Branch Number of $F$ is defined as

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

where $HW$ denotes the Hamming weight

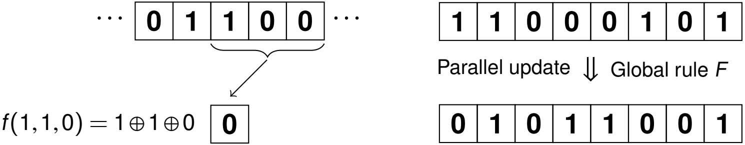
# Cellular Automata (CA)

- A $(n, n)$–function $F$ defined by a local rule $f : \mathbb{F}_2^{\delta} \to \mathbb{F}_2$ with $\delta \leq n$, such that

$$F(x_1, \cdots, x_n) = (f(x_1, \cdots, x_\delta), f(x_2, \cdots, x_{\delta+1}), \cdots, f(x_n, \cdots, x_{\delta-1}))$$

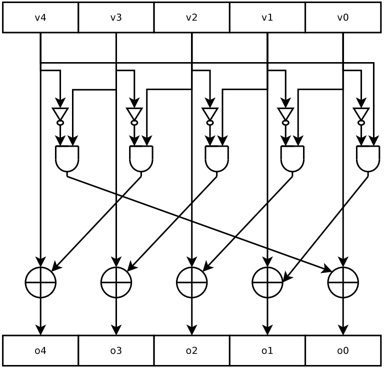
- The local rule is applied to the neighborhood of size $\delta$ of each input cell with periodic boundary conditions

Example: $n = 8$, $\delta = 3$, $f(x_i, x_{i+1}, x_{i+2}) = x_i \oplus x_{i+1} \oplus x_{i+2}$



# The KECCAK $\chi$ transform

- Local rule: $f(x_1, x_2, x_3) = x_1 XOR((NOT(x_2 AND x_3)))$
- Invertible (balanced) for every odd size $n$ of the CA [Daemen94]
- Used in KECCAK with $n = 5$, resulting in an S-box with $N_F = 8$ and $\delta = 8$ [Keccak11]



# Problem Statement

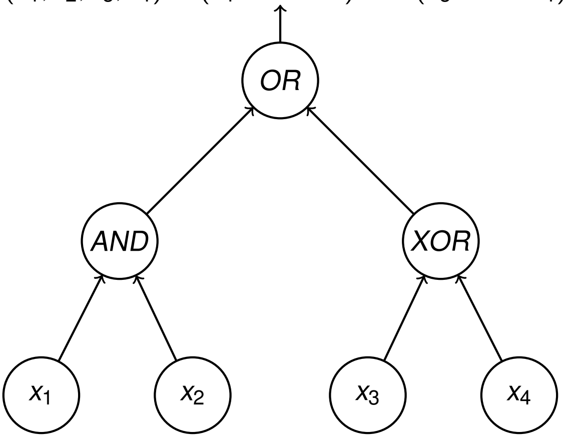
- Goal: Find CA of length $n$ and local rule of size $\delta = n$ having cryptographic properties equal to or better than those of other real-world S-boxes (e.g. KECCAK [Keccak11], ...)
- Considered S-boxes sizes: from $n = 4$ to $n = 8$
- With CA, exhaustive search is possible up to $n = 5$
- But we are also interested in implementation properties!
- $\Rightarrow$ Using tree encoding, exhaustive search is already unfeasible for $n = 4$
- We adopted an evolutionary heuristic – Genetic Programming

# Genetic Programming (GP)

- Optimization method inspired by evolutionary principles, introduced by Koza [Koza93]
- Each candidate solution (individual) is represented by a tree
  - Terminal nodes: input variables
  - Internal nodes: Boolean operators (AND, OR, NOT, XOR, ...)
- New solutions are created through genetic operators like tree crossover and subtree mutation applied to a population of candidate solutions
- Optimization is performed by evaluating the new candidate solutions wrt a fitness function

# GP Tree Encoding – Example

$$f(x_1, x_2, x_3, x_4) = (x_1\ AND\ x2)\ OR\ (x_3\ XOR\ x_4)$$



# Fitness Function

- Main cryptographic properties: balancedness ($BAL = 0$ if $F$ is balanced, $-1$ otherwise), nonlinearity $N_F$ and $\delta$-uniformity $\delta_F$
- Implementation properties: weight $w_I$ defined by GE measure (# of equivalent NAND gates)
  - *NAND* and *NOR* gates: $w_I = 1$
  - *XOR* gate: $w_I = 2$
  - *IF* gate: $w_I = 2.33$
  - *NOT* gate: $w_I = 0.667$
  - *area_penalty*: weighted sum of all operators in a solution
- Fitness function used:

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

where $\Delta_{BAL,0} = 1$ if $F$ is balanced, 0 otherwise

# Experimental Setup

- Problem instance / CA size: $n = 4$ up to $n = 8$
- Maximum tree depth: equal to $n$
- Genetic operators: simple tree crossover, subtree mutation
- Population size: 500
- Stopping criterion: 500000 fitness evaluations
- Parameters determined by initial tuning phase on $n = 5$ case

# Results – Crypto Properties

| $n$ | $N_F$ | $deg_F$ | $deg_F^{-1}$ | $\delta_F$ | $b_F$ | Rule |
|---|---|---|---|---|---|---|
| 4×4 | 4 | 3 | 3 | 4 | 2 | $IF(((x_4\ NOR\ x_2)\ XOR\ x_1),\ x_3,\ x_2)$ |
| 5×5 | 8 | 2 | 3 | 8 | 2 | $((x_3\ NOR\ NOT(x_5))\ XOR\ x_2)$ |
| 5×5 | 8 | 2 | 3 | 4 | 2 | $((x_5\ NAND\ (x_3 XOR x_1))\ XOR\ x2)$ |
| 5×5 | 12 | 2 | 3 | 2 | 2 | $(IF(x_2, x_3, x_5)\ XOR\ (x_1\ NAND\ NOT(x_4)))$ |

- for $n = 4$ and $n = 5$, we obtained CA rules inducing S-boxes with optimal crypto properties
- for $n > 5$, GP finds S-boxes with optimal cryptographic properties up to $n = 7$, but with too high implementation costs

# A Posteriori Analysis – Implementation Properties, $n = 4$

Table: Power is in *nW*, area in *GE*, and latency in *ns*. *DPow*: dynamic power, *LPow*: cell leakage power

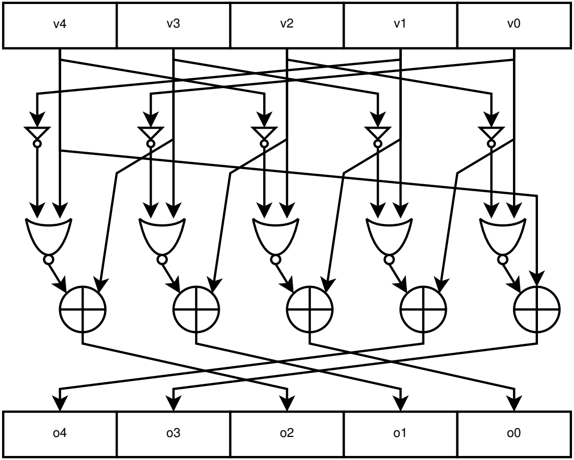
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Size | $4 \times 4$ | Rule | PRESENT [Present07] | | | | |
| DPow. | 470.284 | LPow: | 430.608 | Area: | 22.67 | Latency: | 0.27 |
| Size | $4 \times 4$ | Rule | Piccolo [Piccolo11] | | | | |
| DPow. | 222.482 | LPow: | 215.718 | Area: | 12 | Latency: | 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 |

# A Posteriori Analysis – Implementation Properties, $n = 5$

Table: Power is in *nW*, area in *GE*, and latency in *ns*. *DPow*: dynamic power, *LPow*: cell leakage power

| | | | | | | |
|---|---|---|---|---|---|---|
| Size | 5×5 | Rule | Keccak [Keccak11] | | | |
| DPow. | 321.684 | LPow: | 299.725 | Area: | 17 | Latency:0.14 |
| Size | 5×5 | Rule | ((v2 NOR NOT(v4)) XOR v1) | | | |
| DPow. | 324.849 | LPow: | 308.418 | Area: | 17 | Latency:0.14 |
| Size | 5×5 | Rule | ((v4 NAND (v2 XOR v0)) XOR v1) | | | |
| DPow. | 446.782 | LPow: | 479.33 | Area: | 24.06 | Latency:0.2 |
| Size | 5×5 | Rule | (IF(v1, v2, v4) XOR (v0 NAND NOT(v3))) | | | |
| DPow. | 534.015 | LPow: | 493.528 | Area: | 26.67 | Latency:0.17 |

# Example of Optimal CA S-box found by GP



# Conclusions

- We used Genetic Programming to evolve CA rules generating S-boxes with good cryptographic properties and low implementation cost
- From the cryptographic standpoint, GP is able to find S-boxes with optimal properties up to size $n = 7$
- For the implementation cost, the best evolved S-boxes are similar to those already published in the literature up to $n = 5$ (e.g. KECCAK)
- For $n > 5$, the implementation cost gets worse

# Future Work

- The main avenue for future research is to improve the implementation costs of the solution evolved by GP
- A couple of ideas to achieve this goal:
  - Use power analysis with an a priori approach (include it in th fitness)
  - Use switching technique (different CA rules are used on different input variables)
- Other future direction: improve cryptographic properties for the $n > 5$ case

# References


- [Keccak11] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. 2011. The Keccak reference. (January 2011). http://keccak.noekeon.org/
- [Present07] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. 2007. PRESENT: An Ultra-Lightweight Block Cipher. CHES 2007: 450–466.
- [Daemen94] Joan Daemen, Rene Govaerts, and Joos Vandewalle. 1994. Invertible shift-invariant transformations on binary arrays. Appl. Math. Comput. 62, 2 (1994), 259 – 277
- [Koza93] J. R. Koza: Genetic programming – on the programming of computers by means of natural selection. Complex adaptive systems, MIT Press 1993
- [Piccolo11] K. Shibutani, T. Isobe, H. Hiwatari, Ai Mitsuda, T.Akishita, T. Shirai: Piccolo: An Ultra-Lightweight Blockcipher. CHES 2011: 342–357