University of Milano-Bicocca
Department of Informatics, Systems and Communications

Boolean Functions, S-Boxes and Evolutionary Algorithms

Luca Mariot

luca.mariot@unimib.it

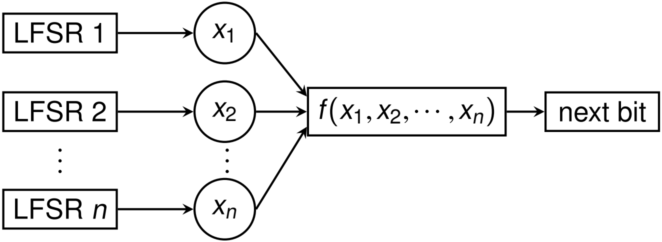
De Cifris Athesis Local Seminar

Trento – December 16, 2019

Part 1: Boolean Functions and S-Boxes

Stream Ciphers: The Combiner Model

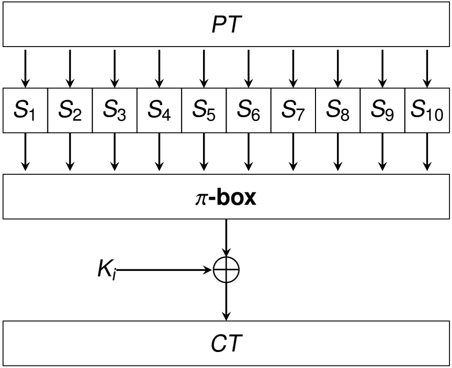
- a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ combines the outputs of n Linear Feedback Shift Registers (LFSR) [Carlet10]



- Security of the combiner $\Leftrightarrow$ cryptographic properties of f

Block Ciphers: Substitution-Permutation Network

Round function of a SPN cipher:



- $S_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ are S-boxes providing confusion
- Security of confusion layer $\Leftrightarrow$ cryptographic properties of S_i

Boolean Functions - Basic Representations

- Truth table: vector Ω_f specifying $f(x)$ for all $x \in \mathbb{F}_2$

(x_1, x_2, x_3)	000	100	010	110	001	101	011	111
Ω_f	0	1	1	1	1	0	0	0

- Algebraic Normal Form (ANF): Sum (XOR) of products (AND) over the finite field $\mathbb{F}_2$

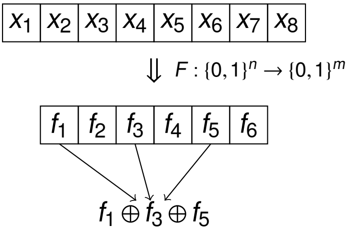
$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \oplus x_1 \oplus x_2 \oplus x_3$$

- Walsh Transform: correlation with the *linear* functions defined as $\omega \cdot x = \omega_1 x_1 \oplus \cdots \oplus \omega_n x_n$

$$\hat{F}(\omega) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus \omega \cdot x}$$

S-boxes – Representation

- Substitution Box (S-box, or (n, m)-function): a mapping $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ defined by m coordinate functions $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$



- Component functions $v \cdot F$: non-trivial linear combinations of the coordinate functions f_i

Design Criteria

Several properties to consider for thwarting attacks, e.g.:

A **Boolean function** used in the combiner model should:

- be balanced
- have high algebraic degree d
- have high nonlinearity $nl(F)$
- be resilient of high order t

A (n, n)**-function** used in the SPN paradigm should

- be balanced ($\Leftrightarrow$ bijective)
- have high nonlinearity N_F
- have low differential uniformity δ_F

Bounds and Trade-offs

Most of these properties cannot be satisfied simultaneously!

Bounds for **Boolean functions**:

- *Covering Radius*: $nl(f) \leq 2^{n-1} - 2^{\frac{n}{2}-1}$ (met by bent functions)
- *Siegenthaler*: $d \leq n - t - 1$
- *Tarannikov*: $nl(f) \leq 2^{n-1} - 2^{t+1}$

Bounds for **S-Boxes**:

- *Covering Radius*: $N_F \leq 2^{n-1} - 2^{\frac{n}{2}-1}$ (met by bent functions)
- *Sidelnikov-Chabaud-Vaudenay*: $N_F \leq 2^{n-1} - 2^{\frac{n-1}{2}}$ (met by AB functions)
- *Differential Uniformity*: $\delta_F \geq 2$ (met by APN functions)

Constructions of good Boolean Functions and S-Boxes

- Number of Boolean functions of n variables: 2^{2^n}

n	3	4	5	6	7	8
2^{2^n}	256	65536	$4.3 \cdot 10^9$	$1.8 \cdot 10^{19}$	$3.4 \cdot 10^{38}$	$1.2 \cdot 10^{77}$

- $\Rightarrow$ too huge for exhaustive search when $n > 5$!

In practice, one usually resorts to:

- **Algebraic constructions** (*Maiorana-McFarland, Rothaus,...*) [Carlet10]
- **Combinatorial optimization techniques**
  - *Simulated Annealing* [Clark04]
  - *Evolutionary Algorithms* [Millan98]
  - *Swarm Intelligence* [Mariot15b], ...

Part 2: Combinatorial Optimization and Evolutionary Algorithms

Combinatorial Optimization

- Combinatorial Optimization Problem: map $\mathcal{P} : \mathcal{I} \to \mathcal{S}$ from a set $\mathcal{I}$ of *problem instances* to a family $\mathcal{S}$ of *solution spaces*
- $S = \mathcal{P}(I)$ is a finite set equipped with a *fitness function* $fit : S \to \mathbb{R}$, giving a score to candidate solutions $x \in S$
- Optimization goal: find $x^* \in S$ such that:

Minimization:	**Maximization:**
$x^* = argmin_{x \in S}\{fit(x)\}$	$x^* = argmax_{x \in S}\{fit(x)\}$

- Heuristic optimization algorithm: iteratively tweaks a (set of) candidate solution(s) using *fit* to drive the search

Hill Climbing and Simulated Annealing

- Let $d_S : S \times S \to \mathbb{R}$ be a distance over the solution space S, and assume there is a minimum distance $d_m \in \mathbb{R}$ such that $d_S(x, x') \geq d_m$ for all $x, x' \in S$.
- Neighborhood of a solution $x \in S$:

$$N(x) = \{y \in S : \ \forall z \in S \ d_S(z, x) \geq d_S(y, x)\}$$

- Hill Climbing: always choose y in $N(x)$ with better fitness
- Simulated Annealing: acceptance probability defined as:

$$P_a = \begin{cases} 1 & , \quad \text{if } f(x) < f(y) \ [f(x) > f(y)] \\ e^{-\left(\frac{|f(y)-f(x)|}{T}\right)} & , \quad \text{if } f(x) \geq f(y) \ [f(x) \leq f(y)] \end{cases}$$

Temperature T updated as $T \leftarrow \alpha T$, where $\alpha \in (0, 1)$.

Genetic Algorithms (GA) – Genetic Programming (GP)

Optimization algorithms loosely based on evolutionary principles, introduced respectively by **J. Holland** (1975) and **J. Koza** (1989)

- Work on a **coding** of the candidate solutions
- Evolve in parallel a **population** of solutions.
- **Black-box optimization**: use only the fitness function to optimize the solutions.
- Use **Probabilistic operators** to evolve the solutions

GA Encoding: Typically, an individual is represented with a fixed-length bitstring

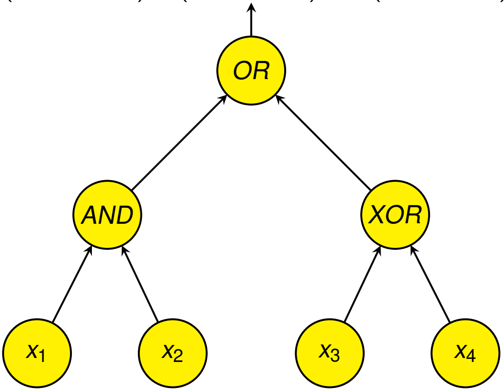
0	**1**	**1**	**1**	**1**	**0**	**0**	**0**

$$\Downarrow$$

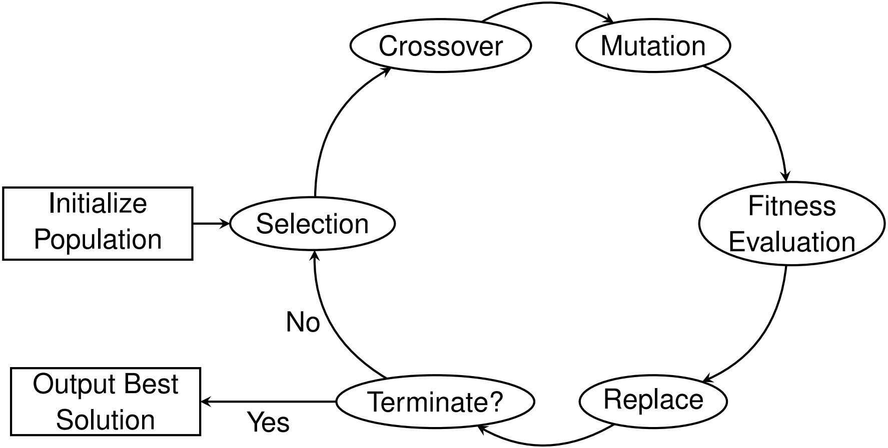
$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \oplus x_1 \oplus x_2 \oplus x_3$$

- **GP Encoding**: an individual is represented by a tree
  - Terminal nodes: input variables of a program
  - Internal nodes: operators (e.g. AND, OR, NOT, XOR, ...)

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



The EA Loop

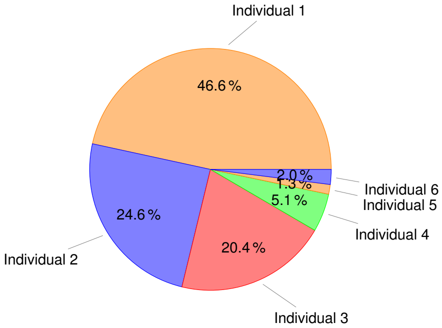


Initialize
Population
Selection
Crossover
Mutation
Fitness
Evaluation
Replace
Terminate?
No
Yes
Output Best
Solution

Selection

Roulette-Wheel Selection (RWS): the probability of selecting an individual is proportional to its fitness

Tournament Selection (TS): Randomly sample *t* individuals from the population and select the fittest one.



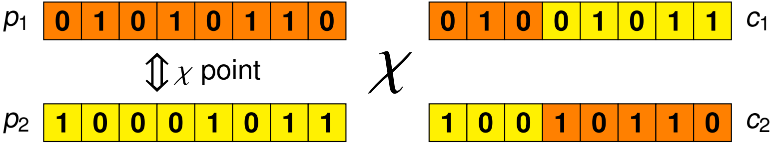
Generational Breeding: Draw as many pairs as population size

Steady-State Breeding: Select only a single pair

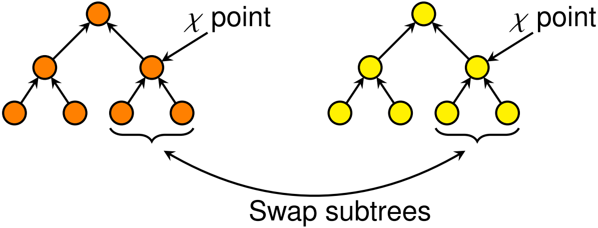
Crossover

Idea: Recombine the genes of two parents individuals to create the offspring (Exploitation)

GA Example: One-Point Crossover



GP Example: Subtree Crossover



Mutation

Idea: Introduce new genetic material in the offspring (Exploration)

GA Example: Bit-flip mutation

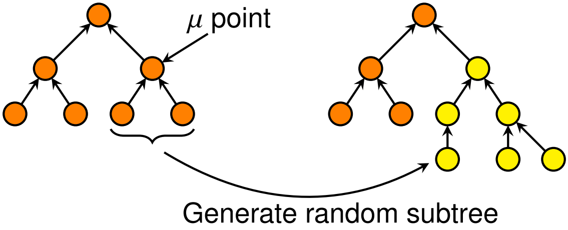
$\downarrow r < p_\mu$

1	**0**	**0**	**0**	**1**	**0**	**1**	**1**

$\Downarrow \mu$

1	**0**	**1**	**0**	**1**	**0**	**1**	**1**

GP Example: Subtree mutation



Replacement and Termination

- **Elitism**: keep the best individual from the previous generation
- **Termination**: several criteria such as budget of fitness evaluations, solutions diversity, ...

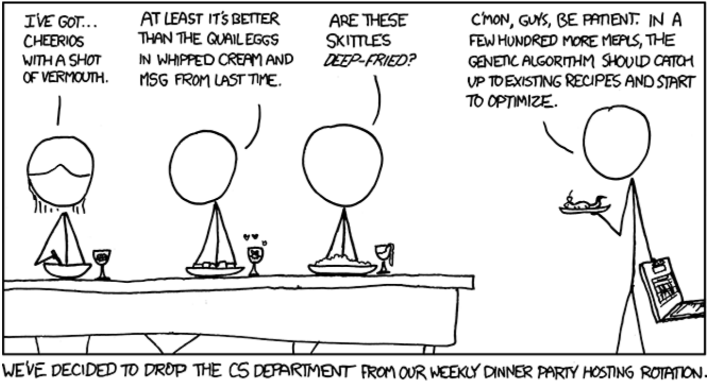


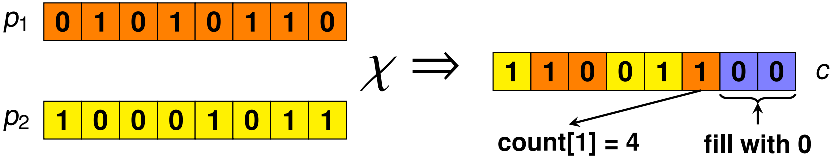
Image credit: `https://xkcd.com/720/`

Part 3: Evolving Boolean Functions and S-Boxes

Direct Search of Boolean Functions [Millan98]

- GA encoding: represent the truth tables as 2^n-bit strings
- Fitness function measuring nonlinearity, algebraic degree, and deviation from correlation-immunity
- Specialized crossover and mutation operators for preserving balancedness

Crossover Idea: Use *counters* to keep track of the multiplicities of zeros and ones



- GP has better performance than GA with direct search [Picek16]

Spectral Inversion [Clark04]

- Applying the Inverse Walsh Transform to a generic spectrum yields a pseudoboolean function $f : \mathbb{F}_2^n \to \mathbb{R}$

$$\mathcal{S}_f = (0, -4, -2, 2, 2, 4, 4, -2)$$

$$\Downarrow \hat{F}^{-1}$$

$$\Omega_{\hat{f}} = (0, 0, 0, -1, 0, -1, 2)$$

- New objective: minimize the deviation of Walsh spectra which satisfy the desired cryptographic constraints
- Heuristic techniques proposed for this optimization problem:
  - Clark et al. [Clark04]: Simulated Annealing (SA)
  - Mariot and Leporati [Mariot15a]: Genetic Algorithms (GA)

Plateaued Functions

- Our GA evolves spectra of plateaued functions
- A (pseudo)boolean function f is plateaued if its Walsh spectrum takes only three values: $-W_M(f)$, 0 and $+W_M(f)$, with $W_M(f) = 2^r$

$$\mathcal{S}_f = (0,0,0,0,-4,4,4,4) \Rightarrow \text{plateaued}$$

- Motivations:
  - Simple combinatorial representation of candidate solutions, determined by a single parameter $r \geq n/2$
  - Plateaued functions reach both Siegenthaler's and Tarannikov's bounds

Chromosome Encoding

- Resiliency Constraint: ignore positions with at most t ones

x	$\underline{000}$	$\underline{100}$	$\underline{010}$	110	$\underline{001}$	101	011	111
$\mathcal{S}_f$	0	0	0	−4	0	4	4	4

- The chromosome c is the permutation of the spectrum in the positions with more than t ones:

x	110	101	011	111
c	−4	4	4	4

- The multiplicities of 0, $-W_M(f)$ and $+W_M(f)$ in the permutation depend on plateau index r

Fitness Function

- Given $\hat{f} : \mathbb{F}_2^n \to \mathbb{R}$, the nearest boolean function $\hat{b} : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined for all $x \in \mathbb{F}_2^n$ as:

$$\hat{b}(x) = \begin{cases} +1 & , \text{ if } \hat{f}(x) > 0 \\ -1 & , \text{ if } \hat{f}(x) < 0 \\ +1 \text{ or } -1 \text{ (chosen randomly)} & , \text{ if } \hat{f}(x) = 0 \end{cases}$$

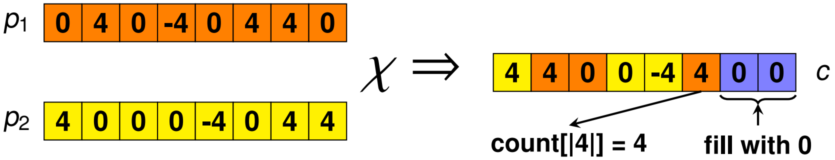
- Objective function proposed in [Clark04]:

$$obj(f) = \sum_{x \in \mathbb{F}_2^n} (\hat{f}(x) - \hat{b}(x))^2$$

- Fitness maximised by GA [Mariot15a]: $fit(f) = -obj(f)$

Genetic Operators

- Crossover between two Walsh spectra p_1, p_2 must preserve the multiplicities of $-W_M(f)$, 0 and $+W_M(f)$
- Idea: Adapt Millan et al.'s counter-based crossover [Millan98]



- Mutation: swap two random positions in the chromosome with different values
- Selection operators adopted:
  - Roulette-Wheel (*RWS*)
  - Deterministic Tournament (*DTS*)

Experimental Settings

Common parameters:

- Number of variables $n = 6, 7$ and plateau index $r = 4$

(n, m, d, nl)	$\|0_{res}\|$	$\|0_{add}\|$	$\|-W_M(f)\|$	$\|+W_M(f)\|$
$(6, 2, 3, 24)$	22	26	6	10
$(7, 2, 4, 56)$	29	35	28	36

GA-related parameters:

- Population size $N = 30$
- max generations $G = 500000$
- GA runs $R = 500$
- Mutation probability $p_\mu = 0.05$
- Tournament size $tsize = 3$

SA-related parameters:

- Inner loops $MaxIL = 3000$
- Moves in loop $MIL = 5000$
- SA runs $R = 500$
- Initial temperatures $T = 100, 1000$
- Cooling parameter: $\alpha = 0.95, 0.99$

Results

Statistics of the best solutions found by our GA and SA over $R = 500$ runs.

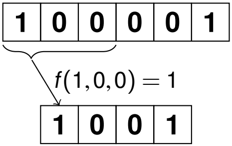
n	Stat	$GA(RWS)$	$GA(DTS)$	$SA(T_1, \alpha_1)$	$SA(T_2, \alpha_2)$
6	avg_o	14.08	13.02	19.01	19.03
	min_o	0	0	0	0
	max_o	16	16	28	28
	std_o	5.21	6.23	4.89	4.81
	$\#opt$	$\boxed{60}$	$\boxed{93}$	$\boxed{11}$	$\boxed{10}$
	avg_t	83.3	79.2	79.1	79.4
7	avg_o	53.44	52.6	45.09	44.85
	min_o	47	44	32	27
	max_o	58	59	63	57
	std_o	2.40	2.77	4.39	4.18
	$\#opt$	0	0	0	0
	avg_t	204.2	204.5	180.3	180.2

Cellular Automata S-boxes

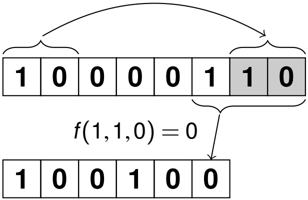
- One-dimensional Cellular Automaton (CA): a discrete parallel computation model composed of a finite array of n cells
- Each cell updates its state $s \in \{0,1\}$ by applying a local rule $f : \{0,1\}^d \rightarrow \{0,1\}$ to itself and the $d-1$ cells to its right

Example: $n = 6$, $d = 3$, $f(s_i, s_{i+1}, s_{i+2}) = s_i \oplus s_{i+1} \oplus s_{i+2}$,

Truth table: $\Omega(f) = 01101001 \rightarrow$ Rule 150



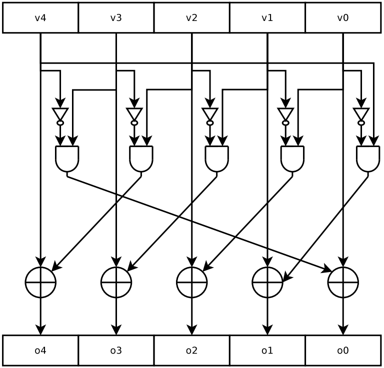
No Boundary CA – NBCA



Periodic Boundary CA – PBCA

CA-Based Crypto History: Keccak χ S-box

- Local rule: $\chi(x_1, x_2, x_3) = x_1 \oplus (1 \oplus (x_2 \cdot x_3))$ (rule 210)
- Invertible for every odd size n of the CA



- Used in the Keccak specification of SHA-3 standard

Problem Statement

- Goal: Find PBCA of length n and diameter $d = n$:
  - with cryptographic properties on par with those of other real-world ciphers [Mariot19]
  - with low implementation cost [Picek17]
- Considered S-boxes sizes: from $n = 4$ to $n = 8$
- Using tree encoding, exhaustive search is already unfeasible for $n = 4$
- We adopted Genetic Programming to address this problem

Fitness Functions – Cryptographic properties

- Considered cryptographic properties:
  - balancedness/invertibility ($BAL = 0$ if F is balanced, -1 otherwise)
  - nonlinearity N_F
  - differential uniformity δ_F
- First Fitness function maximized:

$$fitness_1 = BAL + \Delta_{BAL,0}\left(N_F + \left(1 - \frac{nMinN_F}{2^n}\right) + (2^n - \delta_F)\right)$$

where $\Delta_{BAL,0} = 1$ if F is balanced and 0 otherwise, $nMinN_F$: number of occurrences of the current value of nonlinearity

Fitness Functions – Implementation properties

- 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
- Second Fitness function maximized:

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

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: 2000
- Stopping criterion: 2000000 fitness evaluations
- Parameters determined by initial tuning phase on $n = 6$ case

Results

Table: 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

- From $n = 4$ to $n = 7$, we obtained CA rules inducing S-boxes with optimal crypto properties
- Only for $n = 8$ the performances of GP are consistently worse wrt to the theoretical optimum

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

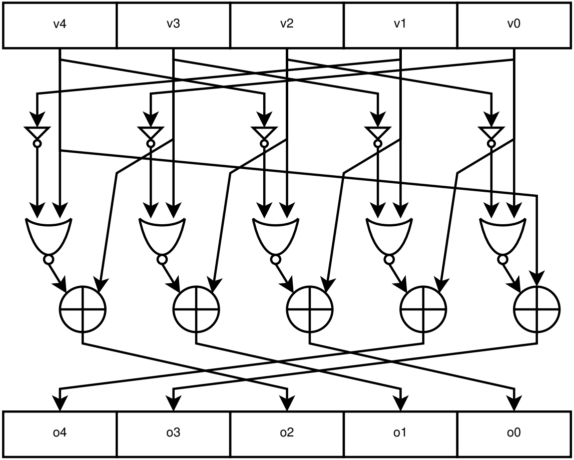
Size	4×4	Rule	PRESENT				
DPow.	470.284	LPow:	430.608	Area:	22.67	Latency:	0.27
Size	4×4	Rule	Piccolo				
DPow.	222.482	LPow:	215.718	Area:	12	Latency:	0.25
Size	4×4	Rule	IF(((v3 NOR v1) XOR v0), v2, v1)				
DPow.	242.52	LPow:	337.47	Area:	16.67	Latency:	0.14

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			
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 and Perspectives

Summing up:

- The design of Boolean functions and S-boxes with good properties is a hard optimization problem
- Evolutionary Algorithms (EA) represent an interesting method to search for optimal Boolean functions and S-boxes both crypto-wise and implementation-wise

Open questions:

- take into account other properties (e.g. algebraic immunity, ...)
- Have a better understanding of which algorithm works best to evolve a Boolean function/S-box with certain properties (using e.g. fitness landscape analsysis)
- Apply EA to other optimization problems in symmetric crypto (e.g. round constants selection)

References

[Carlet10] Carlet, C., Boolean functions for cryptography and error correcting codes. Boolean models and methods in mathematics, computer science, and engineering, vol. 2, pp. 257–397 (2010)

[Clark04] Clark, J., Jacob, J., Maitra, S., Stanica, P.: Almost Boolean Functions: The Design of Boolean Functions by Spectral Inversion. Computational Intelligence 20(3): 450-462 (2004)

[Millan98] Millan, W., Clark, J., Dawson, E.: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. EUROCRYPT 1998: 489-499

[Mariot15a] Mariot, L., Leporati, A.: A Genetic Algorithm for Evolving Plateaued Cryptographic Boolean Functions. In: Proceedings of *TPNC 2015*: 33-45 (2015)

[Mariot15b] Mariot, L., Leporati, A.: Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications. In: GECCO 2015 (Companion): 1425-1426. ACM (2015)

[Mariot19] Mariot, L. Picek, S., Leporati, A., Jakobovic, D.: Cellular Automata Based S-Boxes. *Cryptography and Communications* 11(1): 41-62 (2019)

[Picek16] Picek, S., Jakobovic, D., Miller, J.F., Batina, L., Cupic, M.: Cryptographic Boolean functions: One output, many design criteria Appl. Soft Comput. 40: 635-653 (2016)

[Picek17] Picek, S., Mariot, L., Yang, B., Jakobovic, D., Mentens, N.: Design of S-boxes defined with cellular automata rules. Conf. Computing Frontiers 2017: 409-414 (2017)