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

Cryptography by Cellular Automata

Luca Mariot

luca.mariot@disco.unimib.it

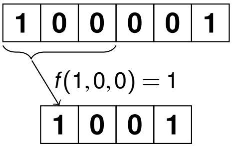
Zagreb – November 14, 2017

Context (1/2): Cellular Automata

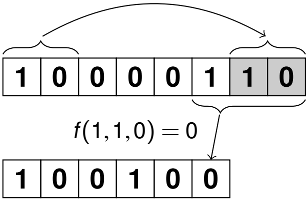
- One-dimensional Cellular Automaton (CA): a discrete parallel computation model composed of a finite array of n cells
- Each cell updates its state $s \in \{0,1\}$ by applying a local rule $f : \{0,1\}^d \to \{0,1\}$ to itself and the $d-1$ cells to its right

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: $\Omega(f) = 01101001 \to$ Rule 150



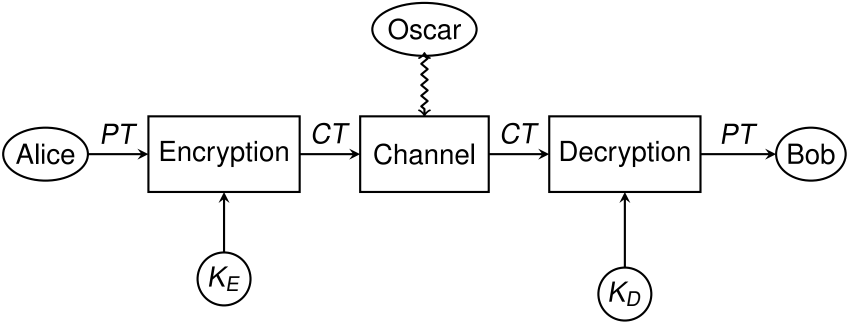
No Boundary CA – NBCA



Periodic Boundary CA – PBCA

Context (2/2): Cryptography

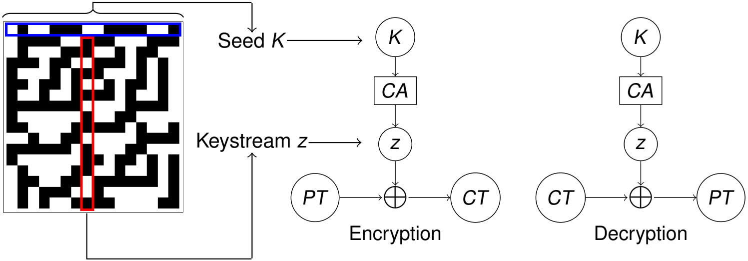
Basic Goal of Cryptography: Enable two parties (Alice and Bob, A and B) to securely communicate over an insecure channel, even in presence of an opponent (Oscar, O)



- PT: plaintext
- CT: ciphertext
- K_E: encryption key
- K_D: decryption key

CA-based Crypto History: Wolfram's PRNG

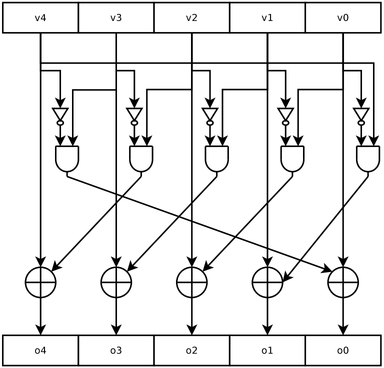
- General Idea: exploit the emergent complexity of CA to design cryptosystems satisfying confusion and diffusion criteria [Shannon49]
- CA-based Pseudorandom Generator (PRG) [Wolfram86]: central cell of rule 30 CA used as a stream cipher keystream



- This CA-based PRNG was later shown to be vulnerable [Meier91]

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 [Daemen94]



- Used as a PBCA with $n = 5$ in the KECCAK specification of SHA-3 standard [Keccak11]

Motivations

Research Goal: investigate the cryptographic properties and the combinatorial designs induced by CA to realize significant cryptographic schemes

What do we mean by "significant"?

1. **Secure**: Satisfying strong security properties
2. **Efficient**: Leveraging CA parallelism for efficient hardware-oriented cryptography

Main focus: Security aspect

Summary of Contributions

Research lines investigated up to now:

- **Line 1**: CA cryptographic properties
  - Bounds on the nonlinearity and *differential uniformity* of CA-based S-boxes
  - CA Cryptographic properties optimization through Genetic Programming (GP)
- **Line 2**: Secret sharing schemes based on CA
  - Orthogonal Latin Squares (OLS) from linear CA
  - Evolutionary search of nonlinear CA generating OLS

Research Line 1: CA cryptographic properties

CA-based cipher design

Design principle: the CA used in cryptographic primitives must satisfy certain properties, to thwart particular attacks

State of the art, up to now:

$$\cdots \boxed{0}\underbrace{\boxed{1}\boxed{1}\boxed{0}}\boxed{0} \cdots$$

$$\downarrow f : \{0,1\}^d \to \{0,1\}$$

$$\boxed{0}$$

- Focus on CA local rules, viewed as Boolean functions
- Rationale: choose rule f with best crypto properties

Our approach:

$$\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{1}$$

$$\Downarrow F : \{0,1\}^n \to \{0,1\}^m$$

$$\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{1}\boxed{0}$$

- Some attacks cannot be formalized in a local way
- Idea: Analyze the CA global rule as a S-box

Research Line 1: CA cryptographic properties

Contribution 1: Bounds on the nonlinearity and differential uniformity of CA-based S-boxes

Nonlinearity of Boolean Functions

- Linear Boolean function $L_\omega : \{0,1\}^n \rightarrow \{0,1\}$:

$$L_\omega(x) = \omega \cdot x = \omega_1 x_1 \oplus \cdots \oplus \omega_n x_n$$

- Nonlinearity of $f : \{0,1\}^n \rightarrow \{0,1\}$: minimum Hamming distance of f from the set of all linear functions:

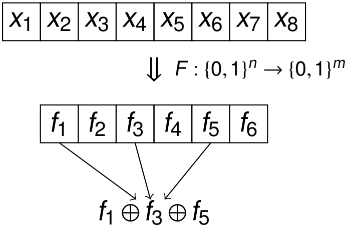
$$N_f = 2^{n-1} - \frac{1}{2}(|W_{max}(f)|)$$

where $W_{max}(f)$ is the maximum absolute value of the Walsh transform of f:

$$W_f(\omega) = \sum_{x \in \{0,1\}^n} (-1)^{f(x) \oplus \omega \cdot x}$$

Nonlinearity of S-boxes

- A Substitution Box (S-box) is a mapping $F : \{0,1\}^n \rightarrow \{0,1\}^m$ defined by m coordinate functions $f_i : \{0,1\}^n \rightarrow \{0,1\}$
- The component functions $v \cdot F : \{0,1\}^n \rightarrow \{0,1\}$ for $v \in \{0,1\}^m$ of F are the linear combinations of the f_i



- The nonlinearity of a S-box F is defined as the minimum nonlinearity among all its component functions
- S-boxes with high nonlinearity allow to resist to linear cryptanalysis attacks

Differential Uniformity of S-boxes

- delta difference table of F wrt a, b:

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

- Given $\delta_F(a,b) = |D_F(a,b)|$, the differential uniformity of F is:

$$\delta_F = \max_{\substack{a \in \{0,1\}^{n*} \\ b \in \{0,1\}^m}} \delta_F(a,b).$$

- S-boxes with low differential uniformity are able to resist differential cryptanalysis attacks

Nonlinearity and Differential Uniformity of CA S-Boxes)

- We proved the following upper bounds for NBCA and PBCA:

Theorem

The nonlinearity and differential uniformity of the S-box F of an n-cell NBCA or PBCA with local rule $f : \{0,1\}^d \to \{0,1\}$ satisfy

$$N_F \leq 2^{n-d} \cdot N_f$$

$$\delta_F \leq 2^{n-d} \cdot \delta_f$$

- **Remark**: This explains why adding cells to a CA makes the cryptographic properties of the S-box worse (see e.g. KECCAK)

Research Line 1: CA cryptographic properties

Contribution 2: CA Cryptographic properties optimization through Genetic Programming (GP)

(Joint work with Stjepan Picek and Domagoj Jakobovic)

Problem Statement

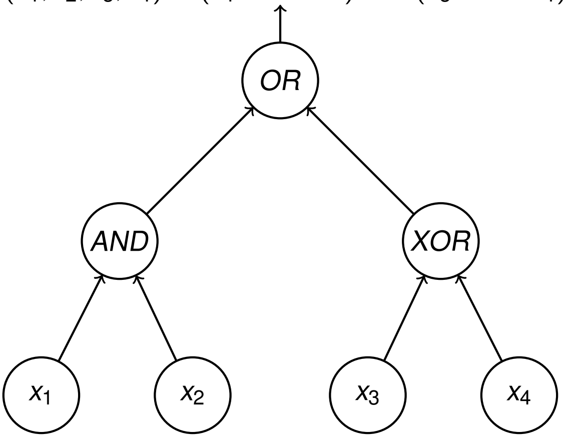
- Goal: Find PBCA of length n and diameter $d = n$ having cryptographic properties equal to or better than those of other real-world S-boxes (e.g. KECCAK, ...)
- Considered S-boxes sizes: from $n = 4$ to $n = 8$
- Using tree encoding, exhaustive search is already unfeasible for $n = 4$
- We adopted an 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

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

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

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

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 – Crypto Properties

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

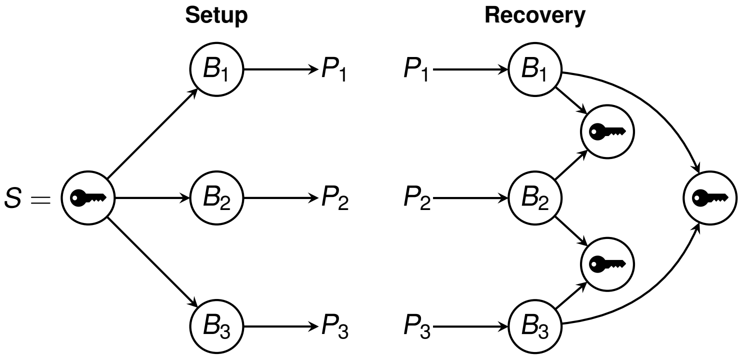
Summary of Contributions

Research Line 2: CA-based secret sharing schemes

Secret Sharing Schemes

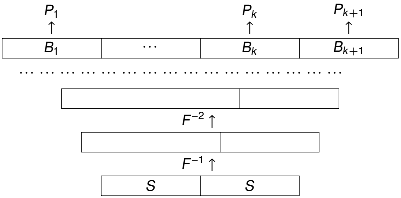
- Secret sharing scheme (SSS): a procedure enabling a dealer to share a secret S among a set $\mathcal{P}$ of n players
- (k, n) threshold SSS: at least k players to recover S

Example: $(2, 3)$–scheme

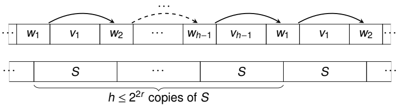


State of the art CA-based SSS

- All CA-based SSS (e.g. [Mariot14]) have a sequential threshold, where shares must be *adjacent*



(a) Sequential threshold CA SSS



(b) Period of spatially periodic preimage

- **Question**: Is it possible to design a CA-based threshold SSS without adjacency constraint?

Research Line 2: CA-based secret sharing schemes

Contribution 1: Generating Orthogonal Latin Squares (OLS) through Linear CA

Latin squares and threshold SSS

- A *Latin square* (LS) is a $N \times N$ matrix where each row and each column permutes $[N] = \{1, \cdots, N\}$
- $L_1, \cdots, L_n$ are *mutually orthogonal* (n-MOLS) if their pairwise superposition yields all the pairs $(x, y) \in [N] \times [N]$

1	3	4	2
4	2	1	3
2	4	3	1
3	1	2	4

(a) L_1

1	4	2	3
3	2	4	1
4	1	3	2
2	3	4	1

(b) L_2

1,1	3,4	4,2	2,3
4,3	2,2	1,4	3,1
2,4	4,1	3,3	1,2
3,2	1,3	2,1	4,4

(c) (L_1, L_2)

Remark: n-MOLS $\Leftrightarrow$ $(2, n)$ threshold SSS

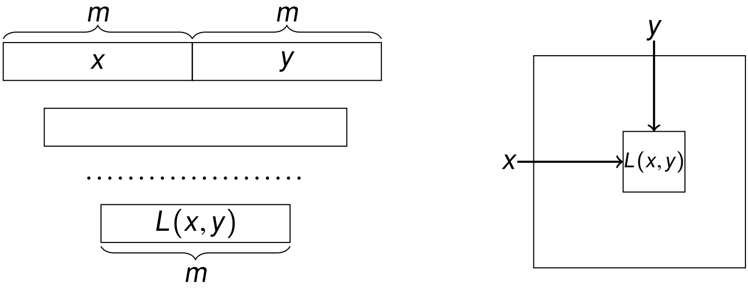
Latin Squares through Bipermutive CA (1/2)

- Idea: determine which CA induce orthogonal Latin squares
- Bipermutive CA: local rule f is defined as

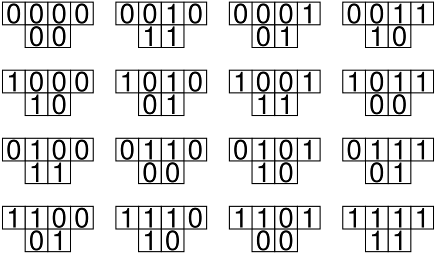
$$f(x_1, \cdots, x_{2r+1}) = x_1 \oplus g(x_2, \cdots, x_{2r}) \oplus x_{2r+1}$$

Lemma

Let F be a m-cell bipermutive NBCA with diameter d s.t. $(d-1)|m$. Then, the CA generates a Latin square of order $N = 2^m$



- Example: CA $\langle \mathbb{F}_2, 4, 1, f \rangle$, $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ (Rule 150)
- Encoding: $00 \mapsto 1, 10 \mapsto 2, 01 \mapsto 3, 11 \mapsto 4$



(a) Rule 150 on 4 bits

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(b) Latin square L_{150}

Linear CA

- Local rule: linear combination of the neighborhood cells

$$f(x_1,\cdots,x_d) = a_1x_1 \oplus \cdots \oplus a_dx_d \ \ , \ a_i \in \mathbb{F}_2$$

- Associated polynomial:

$$f \mapsto \varphi(X) = a_1 + a_2X + \cdots + a_dX^{d-1}$$

- Global rule: $m \times (m+d-1)$ $(d-1)$-diagonal transition matrix

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

$$x = (x_1,\cdots,x_n) \mapsto M_Fx^\top$$

Orthogonal Latin Squares by Linear CA

Theorem

Let F, G be linear bipermutive NBCA. The Latin squares induced by F and G are orthogonal if and only if $P_f(X)$ and $P_g(X)$ are coprime

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(a) Rule 150

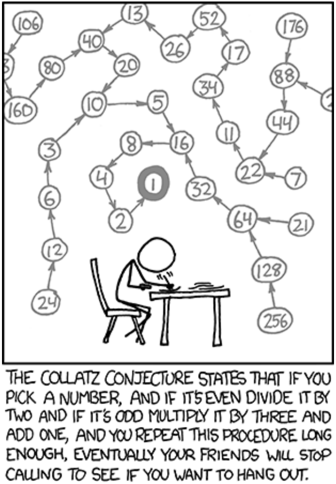
1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

(b) Rule 90

1,1	4,2	3,3	2,4
2,2	3,1	4,4	1,3
4,3	1,4	2,1	3,2
3,4	2,3	1,2	4,1

(c) Superposition

Figure : $P_{150}(X) = 1 + X + X^2$, $P_{90}(X) = 1 + X^2$ (coprime)



https://xkcd.com/710/

- Number of coprime polynomial pairs of degree n and nonzero constant term:

$$a(n) = 4^{n-1} + a(n-1) = \\ = \frac{4^{n-1}-1}{3} = \\ = 0, 1, 5, 21, 85, \ldots$$

- This sequence corresponds to OEIS A002450, which has several other interpretations (e.g. Collatz conjecture, ...)

Research Line 2: CA-based secret sharing schemes

Contribution 2: Evolutionary search of nonlinear CA generating OLS

(Joint work with Stjepan Picek and Domagoj Jakobovic)

Motivations and Goals

- Construction of OLS solved for linear CA [Mariot16]
- MOLS arising from nonlinear constructions have relevance in cheater-immune Secret Sharing Schemes [Tompa88]

Goal: Design OLS based on CA by evolving pairs of nonlinear bipermutive local rules through GA and GP

Twofold motivation:

- **Theoretical:** Understand the mathematical structure of the space of nonlinear CA-based OLS
- **EC perspective:** Source of new problems for evolutionary algorithms

Search Space Size

- Number of Boolean functions of n variables: $\mathcal{F}_n = 2^{2^n}$
- Bipermutive rules of size $n \Leftrightarrow$ Generating functions of size $n-2$ (which are $\mathcal{F}_{n-2} = 2^{2^{n-2}}$)
- Pairs of bipermutive rules of size n: $\mathcal{B}_n = 2^{2^{n-1}} = \mathcal{F}_{n-1}$

n	3	4	5	6	7	8
$\mathcal{B}_n$	16	256	65536	$\approx 4.3 \times 10^9$	$\approx 1.8 \cdot 10^{19}$	$\approx 3.4 \cdot 10^{38}$
$N \times N$	4×4	8×8	16×16	32×32	64×64	128×128
$\#OLS$	8	72	1704	533480	?	?

Remark: Exhaustive enumeration possible up to $n = 6$

Fitness Functions (1/2)

- $\#rep(L_1, L_2)$: Number of occurrences of each pair (except the first one) in the superposition of Latin squares L_1 and L_2

1	3	4	2
4	2	1	3
2	4	3	1
2	3	4	1

(a) L_1

1	4	3	2
2	3	4	1
4	1	2	3
3	2	1	4

(b) L_2

4,1	1,4	2,3	3,2
3,2	2,3	1,4	4,1
1,4	4,1	3,2	2,3
2,3	3,2	4,1	1,4

(c) $\#rep(L_1, L_2) = 12$

- Let φ, γ be the generating functions of two bipermutive CA, and let L_φ, L_γ be the associated Latin squares

First fitness function: minimize $fit_1(\varphi, \gamma) = \#rep(L_\varphi, L_\gamma)$

Fitness Functions (2/2)

- **Remark**: fit_1 does not consider the nonlinearity of φ and γ!
- Nonlinearity penalty factor:

$$NlPen(\varphi,\gamma) = \begin{cases} 0 \ , & \text{if } Nl(\varphi) > 0 \ \ AND \ \ Nl(\gamma) > 0 \\ 1 \ , & \text{if } Nl(\varphi) = 0 \ \ XOR \ \ Nl(\gamma) = 0 \\ 2 \ , & \text{if } Nl(\varphi) = 0 \ \ AND \ \ Nl(\gamma) = 0 \end{cases}$$

Second fitness function: minimize

$$fit_2(\varphi,\gamma) = \#rep(L_\varphi, L_\gamma) + NlPen(\varphi,\gamma) \cdot N^2$$

- The N^2 scaling factor balances the range of $\#rep(L_\varphi, L_\gamma)$, which is $\{0, \cdots, N^2\}$

GA Encoding: Single Bitstring

- Let $\varphi, \gamma : \{0,1\}^{n-2} \rightarrow \{0,1\}$ be a pair of generating functions, with 2^{n-2}-bit truth tables $\Omega(\varphi), \Omega(\gamma)$, and let $||$ denote concatenation

First GA encoding: $enc_1(\varphi, \gamma) = \Omega(\varphi) || \Omega(\gamma)$

Example:

$$\varphi(x_1, x_2, x_3) = x_1 \oplus x_3 \Rightarrow \Omega(\varphi) = (0,1,0,1,1,0,1,0)$$
$$\gamma(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 \Rightarrow \Omega(g) = (0,1,1,0,1,0,0,1)$$
$$enc_1(\varphi, \gamma) = (0,1,0,1,1,0,1,0,0,1,1,0,1,0,0,1)$$

- Classic GA variation operators like one-point crossover and bit-flip mutation are applied in this case

GA & GP Encodings: Double Bitstring/Double Tree

- Idea: Keep the generating functions separated and evolve them independently

  Second GA encoding: $enc_2(\varphi,\gamma) = (\Omega(\varphi),\Omega(\gamma))$

- We use the same idea for GP: the genotype is composed of the two trees $T(\varphi)$ and $T(\gamma)$ representing φ and γ

  GP encoding: $enc_{GP}(\varphi,\gamma) = (T(\varphi),T(\gamma))$

- Classic GA and GP variations operators are applied independently on each of the two components

GA Encoding: Balanced Quaternary Strings (1/2)

Definition

$f, g : \{0,1\}^n \to \{0,1\}$ are pairwise balanced (PWB) if

$$\left|(f,g)^{-1}(0,0)\right| = \left|(f,g)^{-1}(1,0)\right| =$$
$$= \left|(f,g)^{-1}(0,1)\right| = \left|(f,g)^{-1}(1,1)\right| = 2^{n-2}$$

Example:

- $f(x_1, x_2, x_3) = x_1 \oplus x_3$ (Rule 90)
- $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ (Rule 150)

$$\Omega(f) = (0,1,0,1,1,0,1,0) \ ,$$
$$\Omega(g) = (0,1,1,0,1,0,0,1) \ .$$

Each of the pairs $(0,0), (1,0), (0,1), (1,1)$ occurs $2^{3-2} = 2$ times

GA Encoding: Balanced Quaternary Strings (2/2)

- Experimental observations on exhaustive search:
  - Two bipermutive CA generate OLS $\Rightarrow$ the local rules are PWB
  - Generating functions are PWB $\Rightarrow$ the local rules are PWB

Third GA encoding: $enc_3(\varphi,\gamma)$ is a quaternary string of length 2^{n-2} where each number from 1 to 4 occurs 2^{n-4} times

Example: $n = 5, (0,0) \mapsto 1, (1,0) \mapsto 2, (0,1) \mapsto 3, (1,1) \mapsto 4$

$$\Omega(\varphi) = (0,1,0,1,1,0,1,0)$$
$$\Omega(\gamma) = (0,1,1,0,1,0,0,1)$$
$$enc_3(\varphi,\gamma) = (1,4,3,2,4,1,2,3)$$

- Balancedness-preserving variation operators for GA:
  - Crossover: use counters to keep track of the multiplicities of the 4 values in the offspring
  - Mutation: use a swap-based operator

Experimental settings

Common Parameters:

- Problem instances: rules of $n = 7$ and $n = 8$ variables
- Termination condition: 300000 fitness evaluations
- Each experiment is repeated over 50 independent runs
- Selection operator: steady-state with 3-tournament operator

GA Parameters:

- Population size: 30 individuals
- Crossover and mutation probabilities: $p_c = 0.95$, $p_m = 0.2$

GP Parameters:

- Boolean operators: AND, OR, XOR, XNOR, NOT, IF
- Population size: 500 individuals
- Mutation probability: $p_m = 0.5$

Results

- (GA, n, enc_i): GA experiment with CA rules of n variables and encoding enc_i, fitness function fit_1
- (GP, n, fit_i): GP experiment with CA rules of n variables and encoding enc_{GP}, fitness function fit_i

Exp.	avg fit	std fit	#opt	#lin	#nlin
$(GA, 7, enc_1)$	520.32	360.16	12/50	0	12
$(GA, 7, enc_2)$	565.44	389.03	15/50	0	15
$(GA, 7, enc_3)$	392.64	328.47	18/50	0	18
$(GA, 8, enc_1)$	4165.44	604	1/50	0	1
$(GA, 8, enc_2)$	4222.16	125.03	0/50	0	0
$(GA, 8, enc_3)$	4696.48	135.51	0/50	0	0
$(GP, 7, fit_1)$	0	0	50/50	50	0
$(GP, 7, fit_2)$	0	0	50/50	0	50
$(GP, 8, fit_1)$	0	0	50/50	47	3
$(GP, 8, fit_2)$	0	0	50/50	0	50

Discussion

For GP:

- GP always manages to converge to an optimal solution
- ... but under fit_1, all solutions found are linear!
- Possible explanation: GP first converges to linear pairs (since it has the XOR operator), then OLS are easily found

On the other hand, for GA:

- GA converged just once for $n = 8$ and the performances for $n = 7$ are worse than GP
- ... but all solutions found are nonlinear, even under fit_1

Conclusions

Wrapping up

We investigated two applications of CA to cryptography, namely:

- Design of CA-based S-boxes:
  - Study of the bounds on nonlinearity and differential uniformity of S-boxes generated through CA
  - Evolutionary search of CA-based S-boxes with good crypto properties through GP
- Design of CA-based Secret Sharing Schemes:
  - Characterization of OLS generated by linear CA
  - Evolutionary search of nonlinear CA generating OLS

Future developments

Research Line 1:

- Consider CA with respect to cryptographic properties related to other kinds of attacks (algebraic attacks, ...)
- Prove lower bounds on the nonlinearity of CA induced by specific classes of rules (bipermutive rules, plateaued functions, ...)

Research Line 2:

- Investigate the behavior of GP in evolving CA generating OLS
- Generalize to higher thresholds (via orthogonal arrays)

References

- [Keccak11] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G. 2011. The Keccak reference. `http://keccak.noekeon.org/` (2011)
- [Daemen94] Daemen, J., Govaerts, R., Vandewalle, J. An efficient nonlinear shift-invariant transformation. In Proceedings of the 15th Symposium on Information Theory in the Benelux, pp. 108-115 (1994)
- [Koza93] J. R. Koza: Genetic programming – on the programming of computers by means of natural selection. Complex adaptive systems, MIT Press 1993
- [Mariot16] Mariot, L., Formenti, E., Leporati, A.: Constructing Orthogonal Latin Squares from Linear Cellular Automata. In: Exp. Proceedings of *AUTOMATA 2016*
- [Mariot14] Mariot, L., Leporati, A.: Sharing Secrets by Computing Preimages of Bipermutive Cellular Automata. In: Proceedings of *ACRI 2014*
- [Meier91] Meier, W., Staffelbach, O. Analysis of Pseudo Random Sequence Generated by Cellular Automata. In EUROCRYPT, Vol. 91, pp. 186-200 (1991)
- [Shannon49] Shannon, C. E. Communication theory of secrecy systems. Bell Labs Technical Journal, 28(4), 656-715 (1949)
- [Tompa88] Tompa, M., Woll, H.: How to share a secret with cheaters. J. Cryptology 1(2), 133–138 (1988)
- [Wolfram86] Wolfram, S.: Random Sequence Generation by Cellular Automata. Adv. Appl. Math. 7(2), 123–169 (1986)