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

Natural Computing Models and Techniques for Cryptography

Luca Mariot

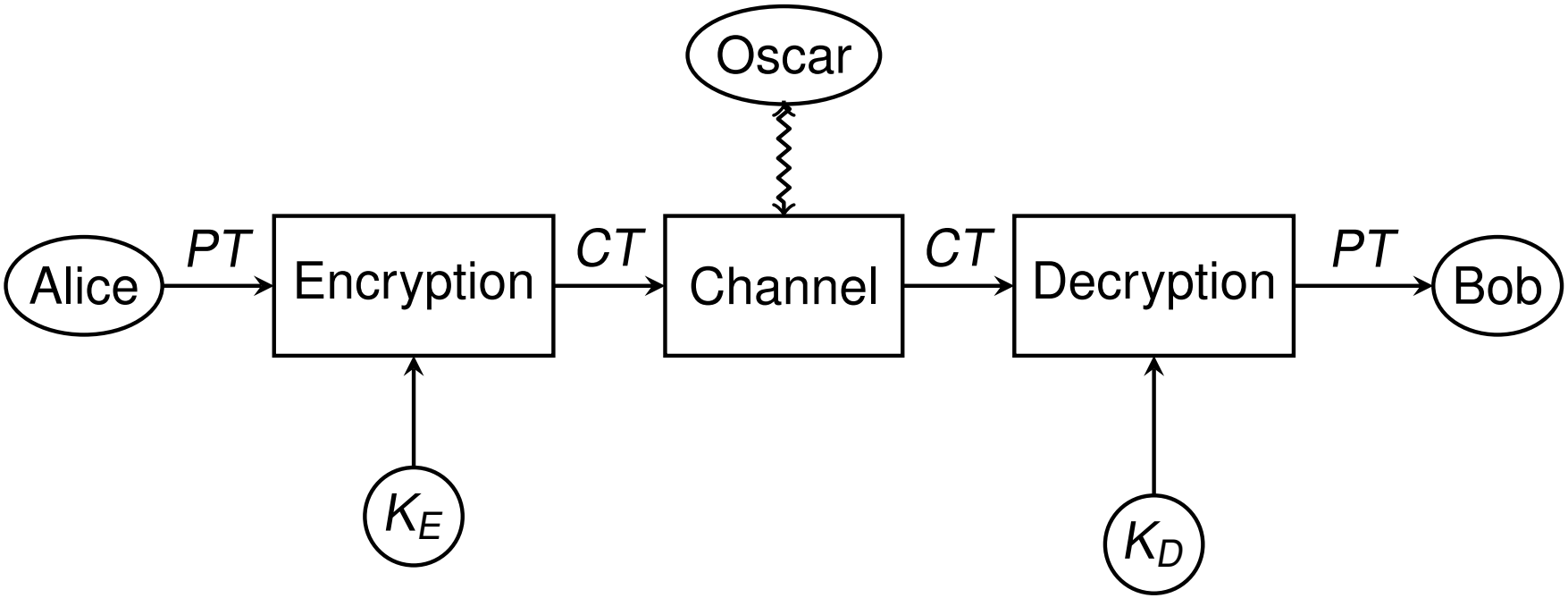
luca.mariot@unimib.it

Milano – December 18, 2019

Part 1: Evolutionary Design of Crytographic Primitives

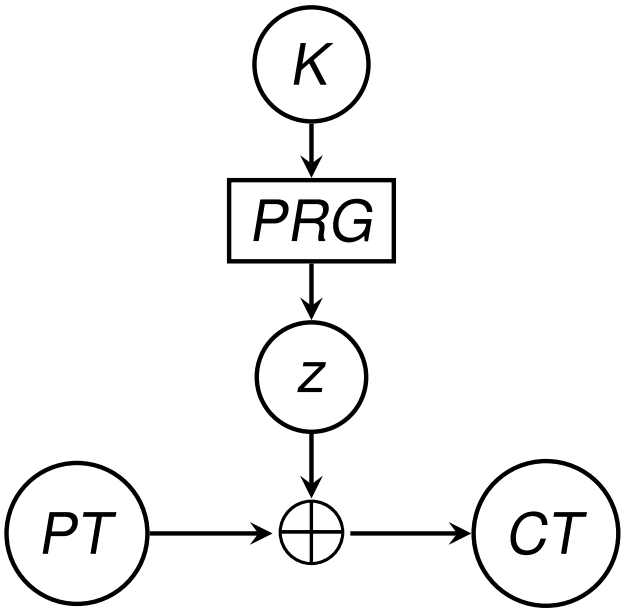
Cryptography

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)

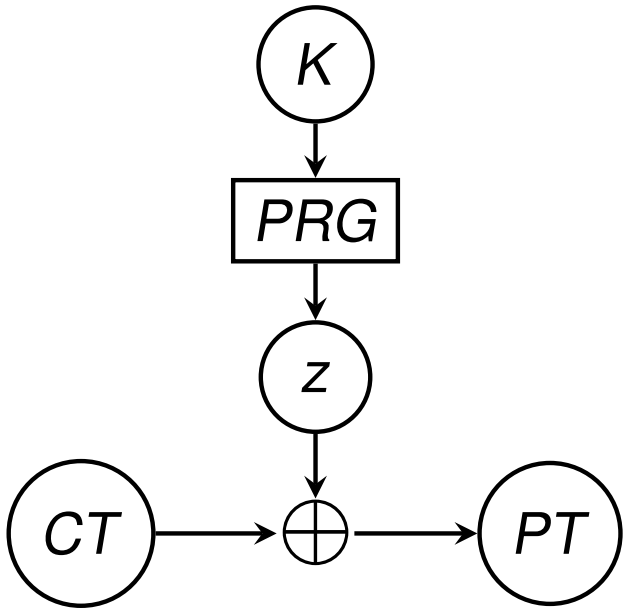


- PT: plaintext
- CT: ciphertext
- K_E: encryption key
- K_D: decryption key

Vernam Stream Cipher



(a) Encryption

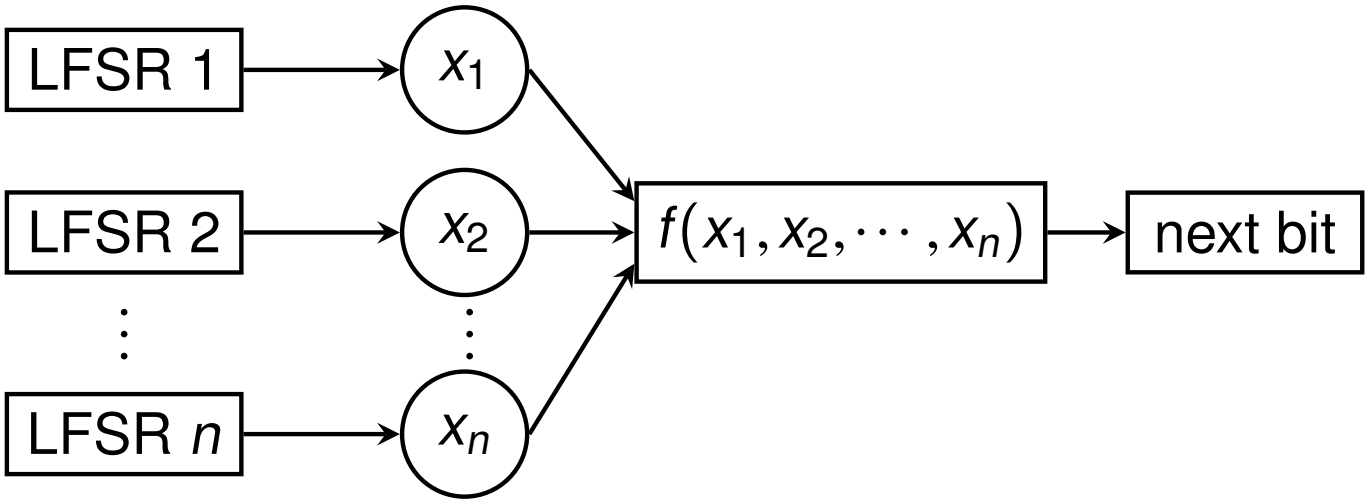


(b) Decryption

- K: secret key
- PRG: Pseudorandom Generator
- z: keystream
- $\oplus$: bitwise XOR
- PT: Plaintext
- CT: Ciphertext

Stream Ciphers: The Combiner Model

- a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ combines the outputs of n LFSR [Carlet10]



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

Boolean Functions - Basic Definitions

Boolean function: a mapping $f : \mathbb{F}_2^n \to \mathbb{F}_2$, where $\mathbb{F}_2 = \{0, 1\}$

- Truth table: vector Ω_f specifying $f(x)$ for all $x \in \mathbb{F}_2$
- Walsh Transform: correlation with the *linear* functions defined as $\omega \cdot x = \omega_1 x_1 \oplus \cdots \oplus \omega_n x_n$

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

Example: $n = 3$, $f(x_1, x_2, x_3) = x_1 \oplus x_3$

(x_1, x_2, x_3)	000	100	010	110	001	101	011	111
Ω_f	0	1	0	1	1	0	1	0
$\hat{F}(\omega)$	0	0	0	0	0	8	0	0

Support of f: $supp(f) = \{100, 110, 001, 011\}$

Correlation Immunity (CI)

- t-th order Correlation Immunity: when fixing any t variables, the restrictions of f have the same Hamming weight
- Walsh characterization:

$$\hat{F}(\omega) = 0 \ \forall \omega : 1 \leq w_H(\omega) \leq t$$

(x_1, x_2, x_3)	000	100	010	110	001	101	011	111
Ω_f	0	1	0	1	1	0	1	0
$\hat{F}(\omega)$	0	0	0	0	0	8	0	0

$$\Downarrow$$

$$\hat{F}(101) = 8 \Rightarrow f \text{ is 1-CI, but NOT 2-CI}$$

Applications:

- Resistance to correlation attacks in the combiner model
- Side-channel countermeasures

Constructions of CI Boolean Functions

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

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

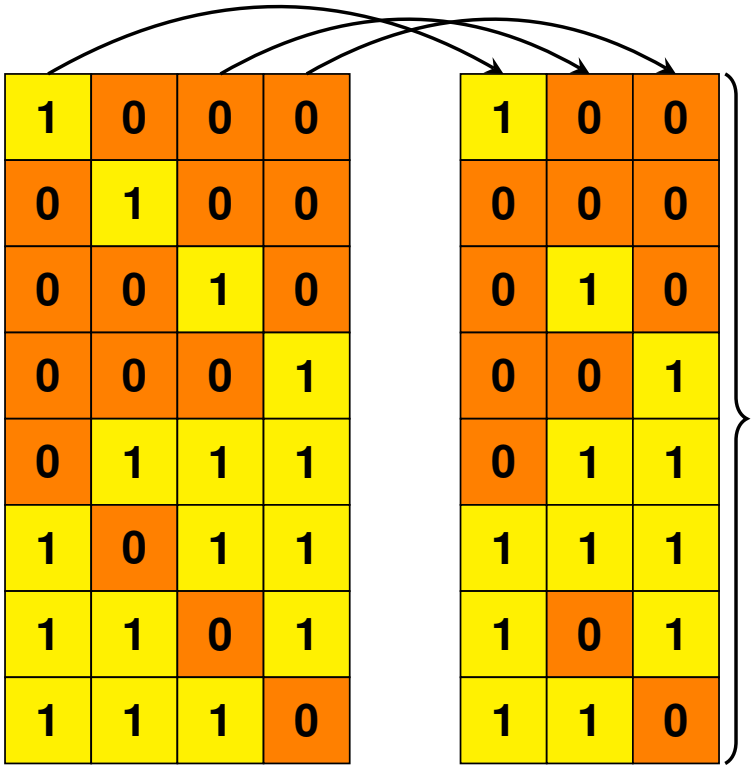
- $\Rightarrow$ too huge for exhaustive search when $n > 5$!

In practice, one usually resorts to:

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

A different angle: Orthogonal Arrays (OA)

- (N,k,t,λ) **Orthogonal Array**: $N \times k$ binary matrix A such that each binary t-uple occurs λ times in each $N \times t$ submatrix.



Example: OA $(8,4,3,1)$

$\Rightarrow$ Each 3-bit vector $(x_1,x_2,x_3) \in \{0,1\}^3$ appears once in the submatrix with columns 1, 3, 4

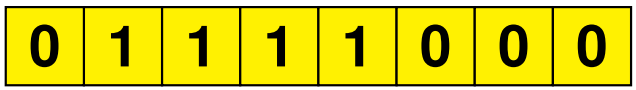
- The rows of a (N,k,t,λ) OA are the **support** of a t-CI function
- **Goal**: Construct OA with Evolutionary Algorithms (EA)

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

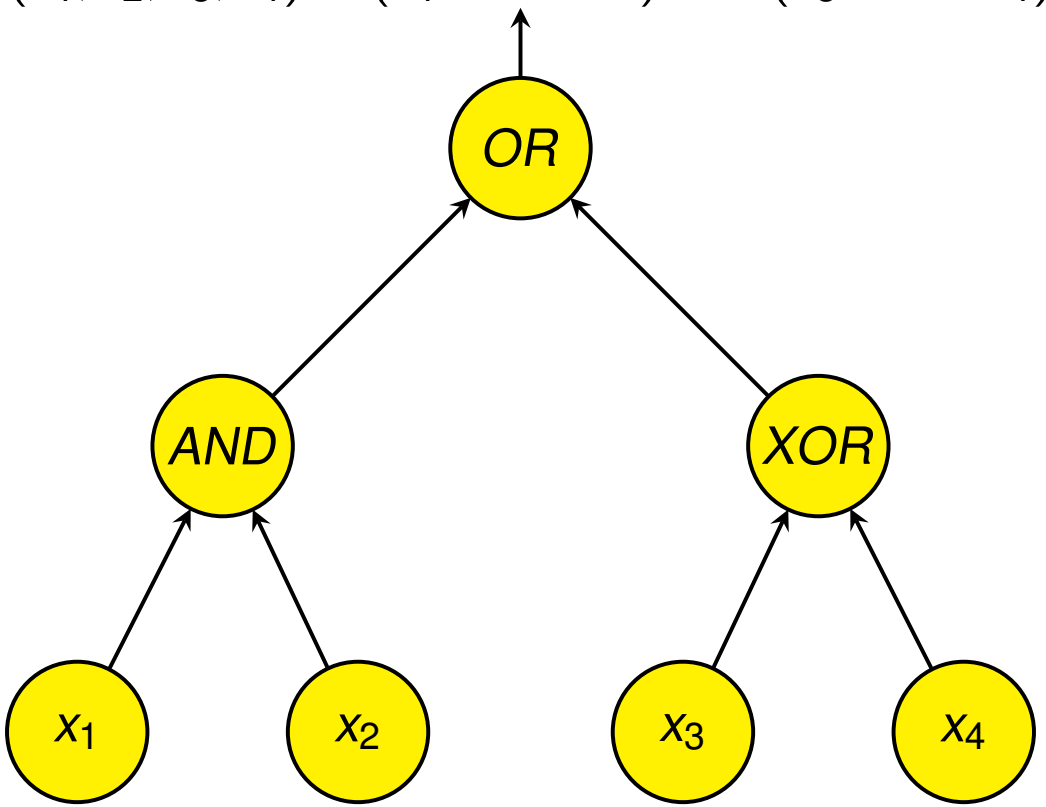


$$\Downarrow$$

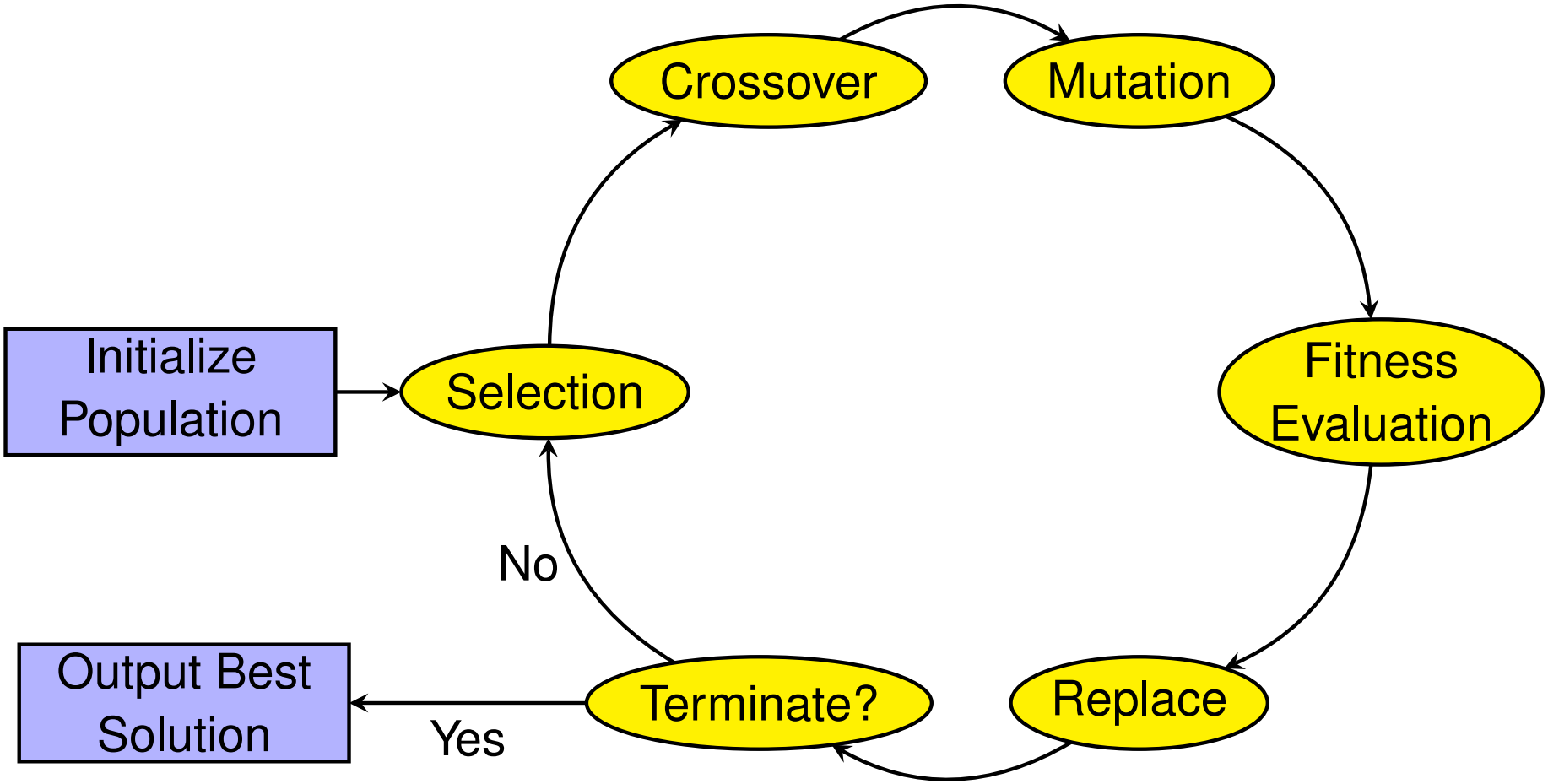
$$f(x_1, x_2, x_3) = x_1 \cdot x_2 \oplus x_1 \oplus x_2 \oplus x_3$$

- **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, ...)

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

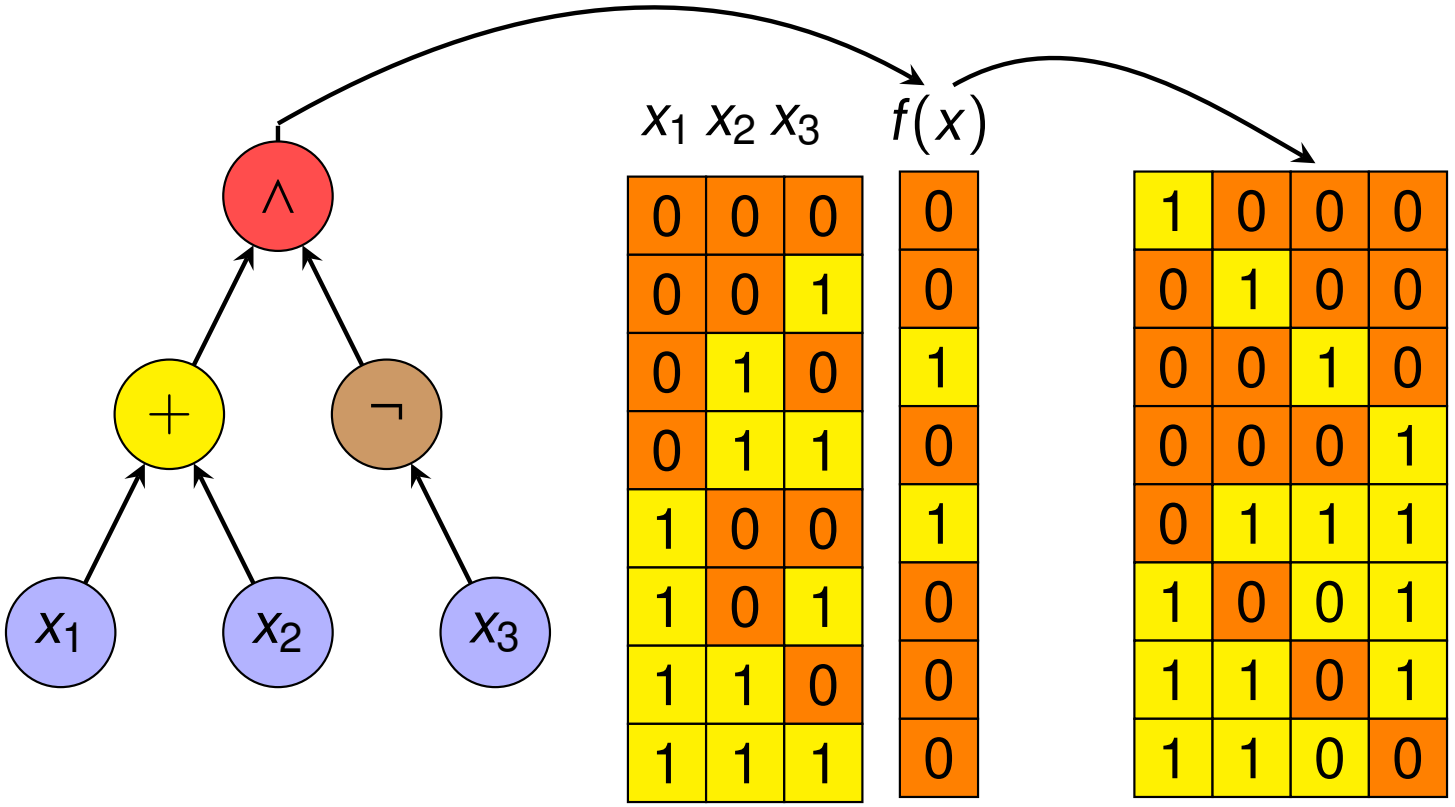


The EA Loop


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

Solutions Encoding

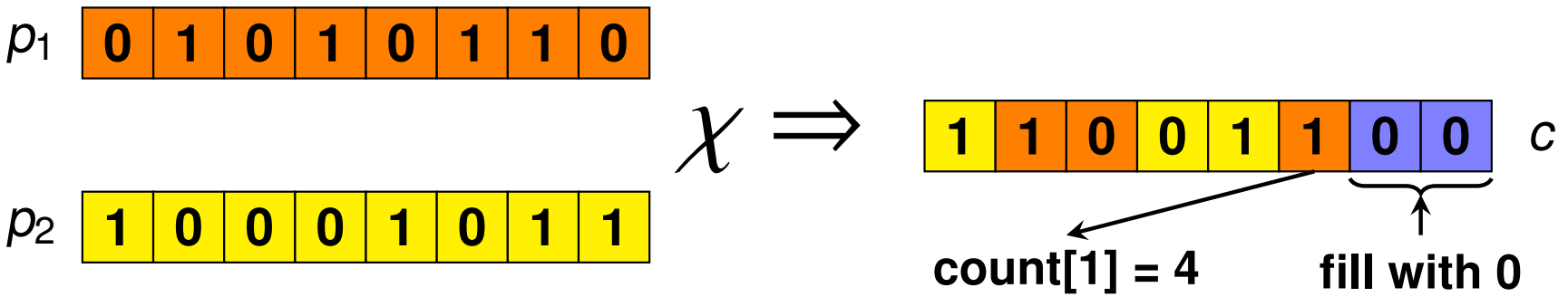
- Each column is the truth table of a n-variable Boolean function
- For GP, the truth table is synthesized from the tree of the individual



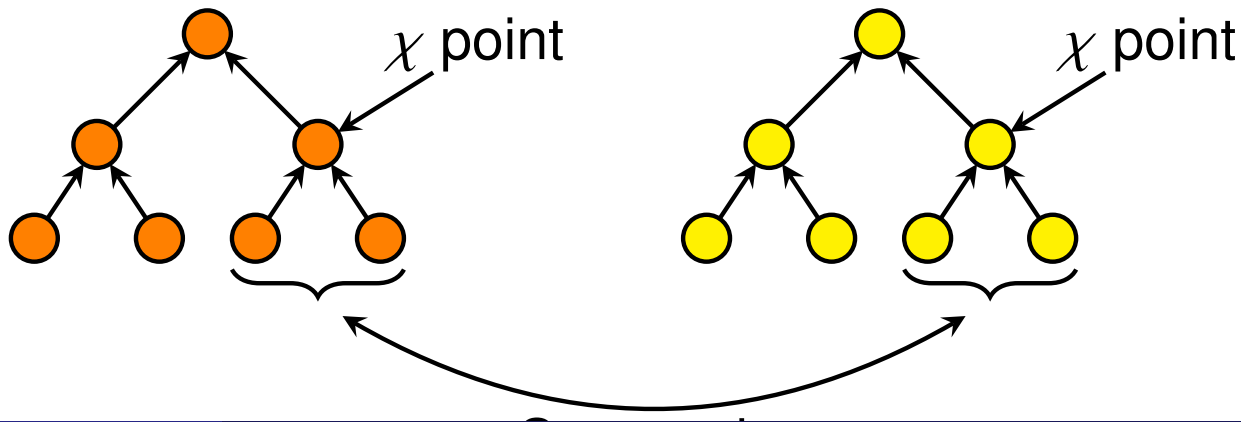
- Crossover and mutation are applied **column-wise**

Crossover Operators

- **Remark**: Each column of an OA must be balanced
- **GA Crossover Idea:** Use *counters* to keep track of the multiplicities of zeros and ones [Millan98]

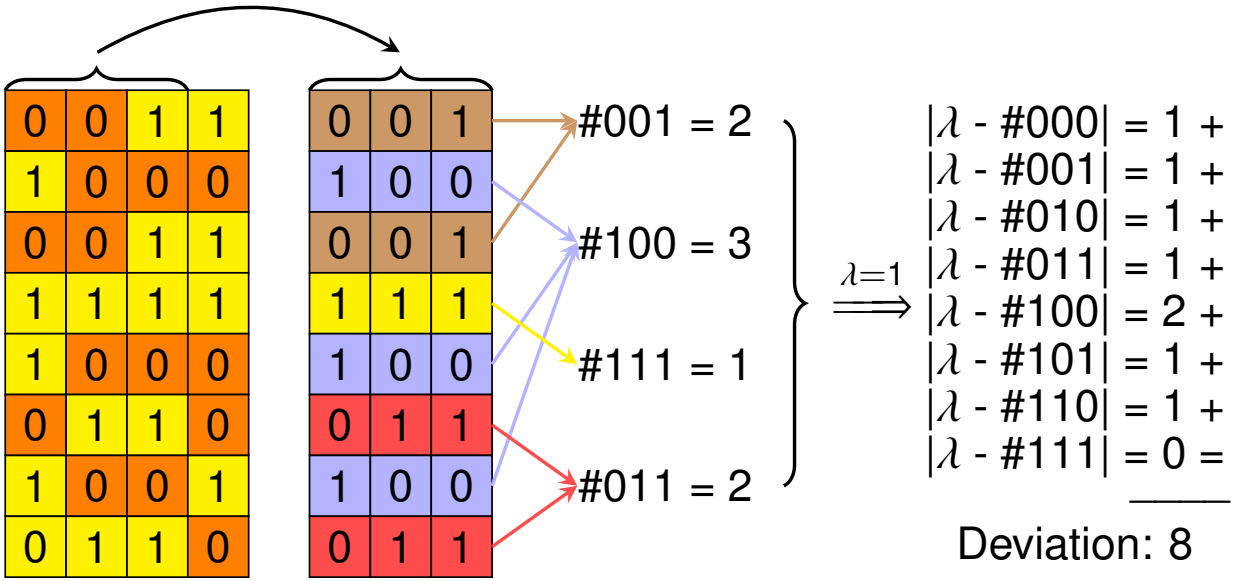


- **For GP**: Use standard subtree crossover



Fitness Function

Idea: *minimize* in each $N \times t$ submatrix the number of occurrences of each t-uple deviating from λ



Fitness function: L^p distance between vector $(\lambda, \cdots, \lambda)$ and the vector of deviations for each submatrix

$$fit_p(A) = \sum_{S \text{ Submatrix}} \left(\sum_{x \in \{0,1\}^t} |\lambda - \#x|^p \right)^{\frac{1}{p}}$$

Experimental Setting

- Problem instances considered:

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
n	3	3	3	3	3	4	4	5	5	6
N	8	8	8	8	16	16	16	32	32	64
k	4	4	5	7	8	8	15	16	31	32
t	2	3	2	2	2	3	2	3	2	3
λ	2	1	2	2	4	2	4	4	8	8

- **GP function set:** AND, OR, XOR, XNOR (2 arguments), NOT (1 argument), IF (3 arguments)
- **Population sizes:** 500 (GP), 50 (GA)
- **Common parameters:** 500 000 fitness evaluations, 30 independent runs for each instance

Results

- **Main finding:** GP outperforms by far GA wrt success rate

	GA					GP				
Exp.	min	avg	std	max	time (s)	min	avg	std	max	time (s)
l_1	0	0	0	0	< 1	0	0	0	0	< 1
l_2	0	0	0	0	< 1	0	0	0	0	< 1
l_3	0	0	0	0	< 1	0	0	0	0	< 1
l_4	0	0.533	1.38	4	7	0	0	0	0	1
l_5	0	2.333	1.75	6	38	0	0	0	0	1
l_6	0	39.96	10.9	57.41	110	0	0.565	3.09	16.97	13
l_7	52	65.4	6.41	80	147	0	0.533	2.03	8	48
l_8	1174	1266	43.4	1349	1995	0	83.72	41.5	135.8	1212
l_9	654	684	14.5	714	1 125	0	32	13.9	64	692
l_{10}	-	-	-	-	-	18812	19159	116	19355	15308

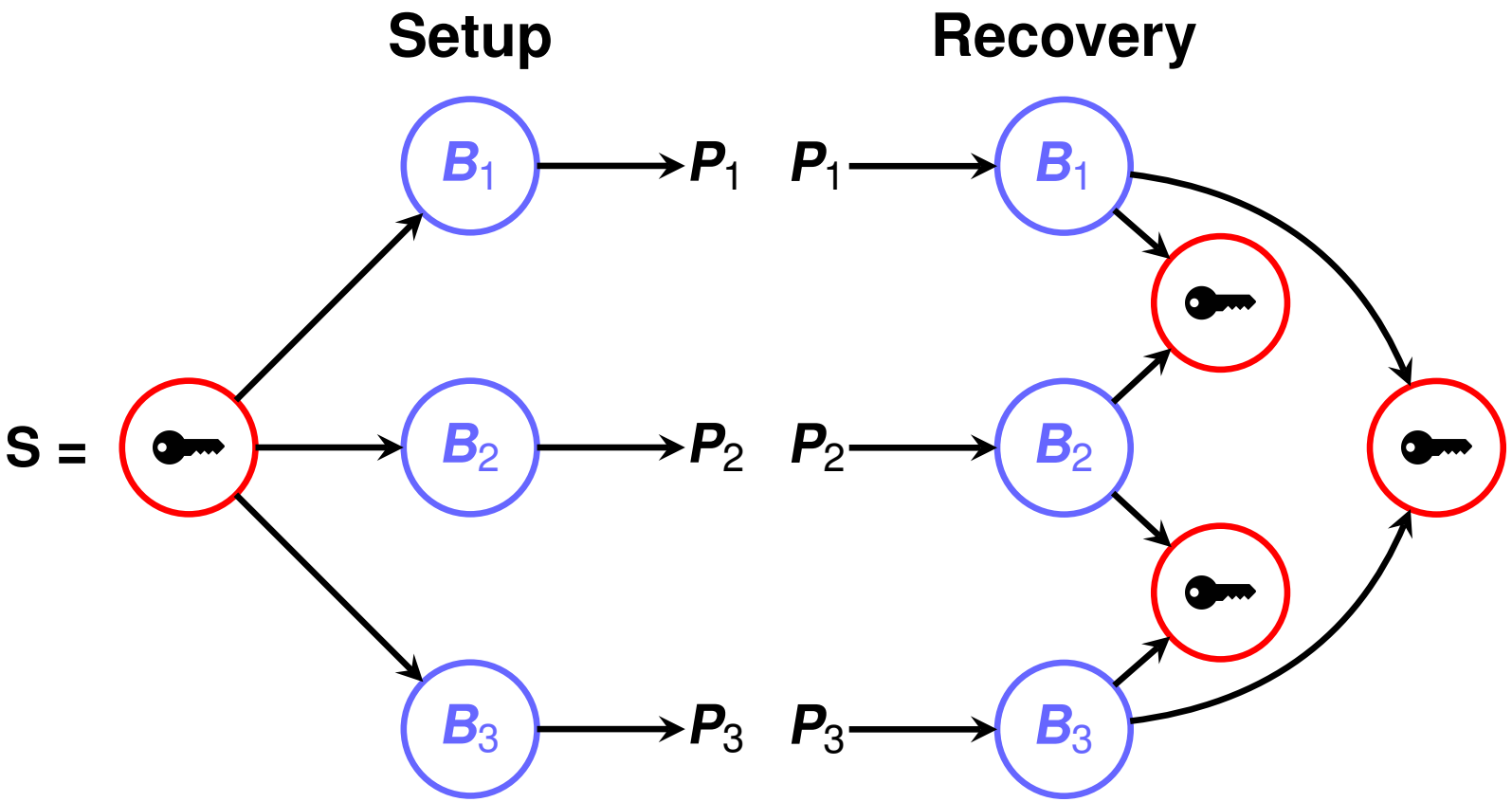
- Results published in PPSN 2018 [Mariot18a]

Part 2: Secret Sharing Schemes from Cellular Automata

Secret Sharing Schemes (SSS)

(k,n) Threshold Secret Sharing Scheme: a procedure enabling a dealer to share a secret S among n players so that at least k players out of n can recover S [Shamir79]

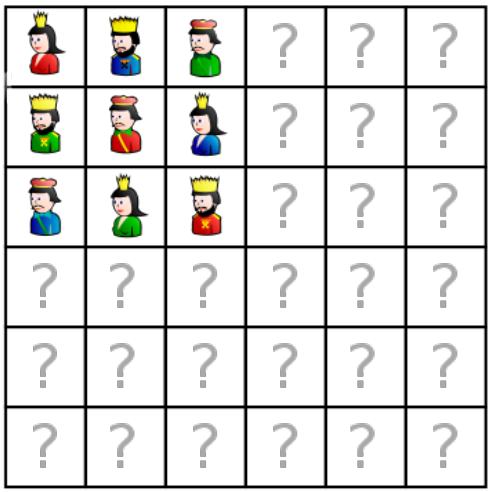
Example: $(2,3)$–scheme



Euler's 36 Officers Problem

« *A very curious question [...] revolves around arranging 36 officers to be drawn from 6 different ranks and also from 6 different regiments so that they are ranged in a square so that in each line (both horizontal and vertical) there are 6 officers of different ranks and different regiments.* »

L. Euler, *Sur une nouvelle espèce de quarrés magiques*, 1782



Latin Squares

Definition

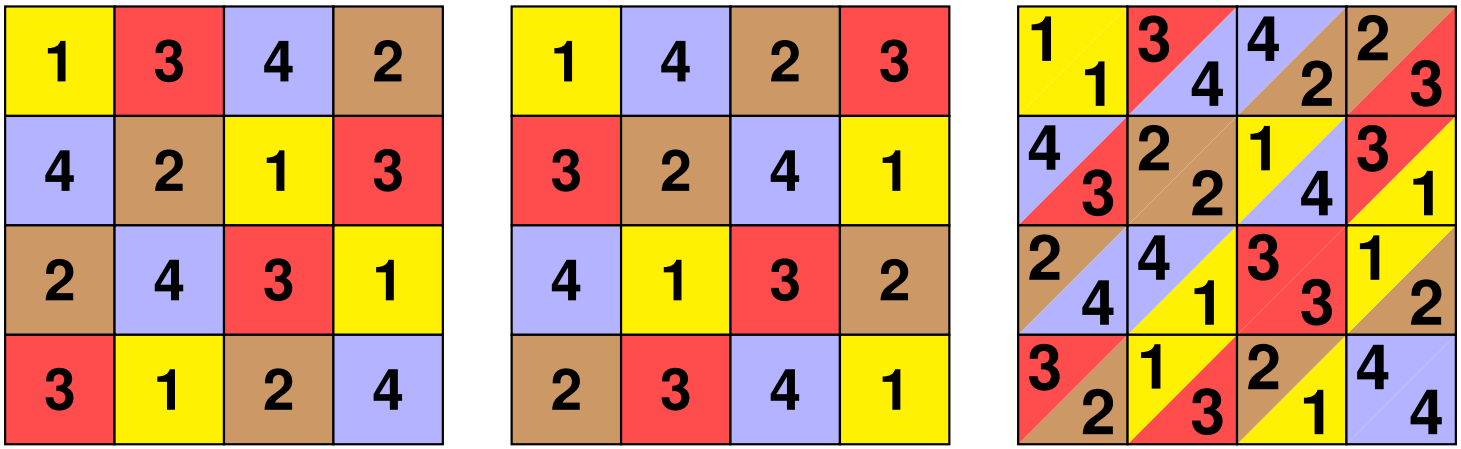
A *Latin square* of order N is a $N \times N$ matrix L such that every row and every column are permutations of $[N] = \{1, \cdots, N\}$

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

Orthogonal Latin Squares (OLS)

Definition

Two Latin squares L_1 and L_2 of order N are *orthogonal* if their superposition yields all the pairs $(x, y) \in [N] \times [N]$.



(a) L_1 (b) L_2 (c) (L_1, L_2)

A set of n pairwise orthogonal Latin squares is denoted as n-MOLS

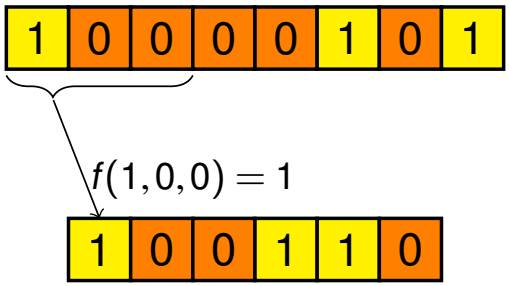
Remark: $(2, n)$–scheme $\Leftrightarrow$ set of n-MOLS

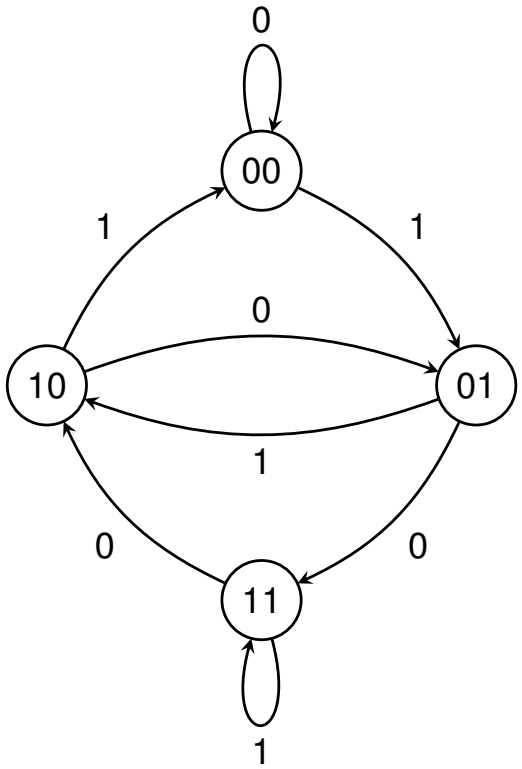
One-Dimensional Cellular Automata (CA)

Definition

One-dimensional CA: triple $\langle m, n, f \rangle$ where $n \in \mathbb{N}$ is the number of cells on a one-dimensional array, $n \in \mathbb{N}$ is the neighborhood and $f : \{0,1\}^n \to \{0,1\}$ is the local rule.

Example: $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ (Rule 150)





Latin Squares through Bipermutive CA (1/2)

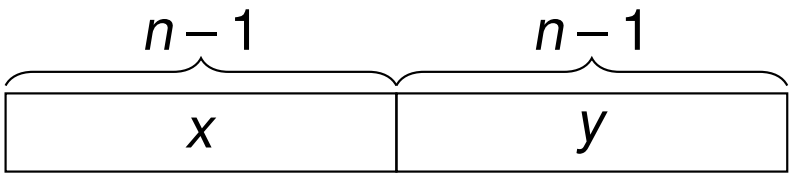
- Idea: determine which CA induce orthogonal Latin squares
- Bipermutive CA: local rule f is defined as

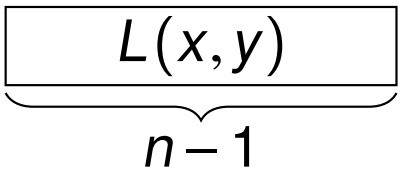
$$f(x_1, \cdots, x_n) = x_1 \oplus \varphi(x_2, \cdots, x_{n-1}) \oplus x_n$$

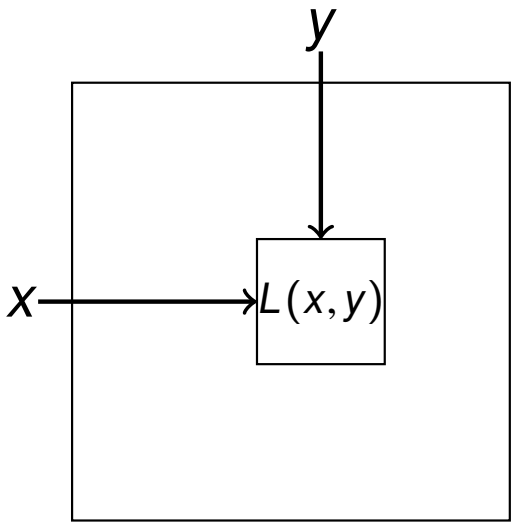
- $\varphi : \{0,1\}^{n-2} \to \{0,1\}$: generating function of f

Lemma ([Mariot16])

Let $\langle 2(n-1), n, f \rangle$ be a CA with bipermutive rule. Then, the global rule F generates a Latin square of order $N = 2^{n-1}$

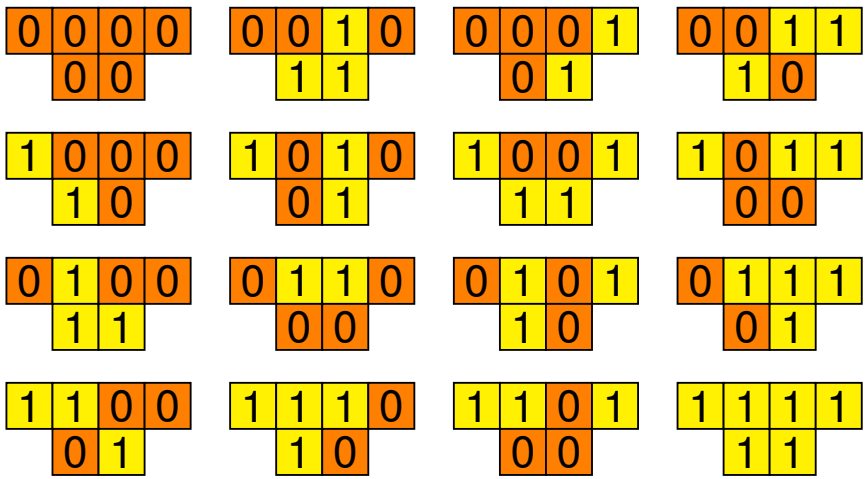






Latin Squares through Bipermutive CA (2/2)

- Example: CA $\langle 4, 1, f \rangle$, $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ (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}

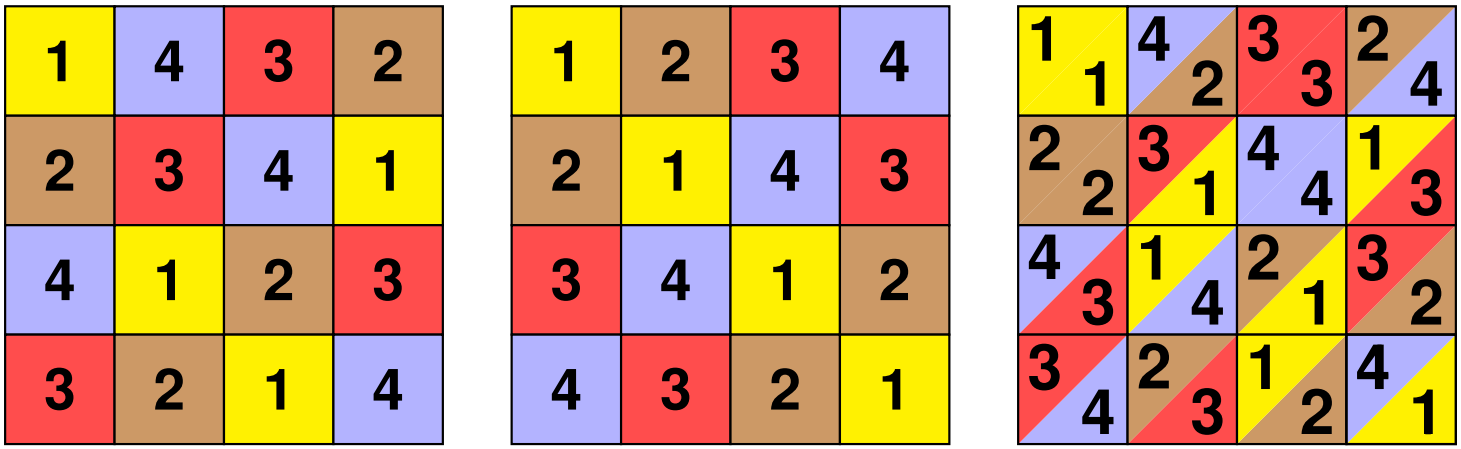
Mutually Orthogonal Cellular Automata (MOCA): set of n bipermutive CA generating n-MOLS

MOCA by Linear CA

- Bipermutive Linear rule: $f(x) = x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_{n-1} x_{n-1} \oplus x_n$
- Associated polynomial: $f \mapsto P_f(X) = a_1 + a_2 X + \cdots + a_n X^{n-1}$

Theorem ([Mariot19])

A set of bipermutive linear CA are MOCA if and only if their associated polynomials are pairwise coprime

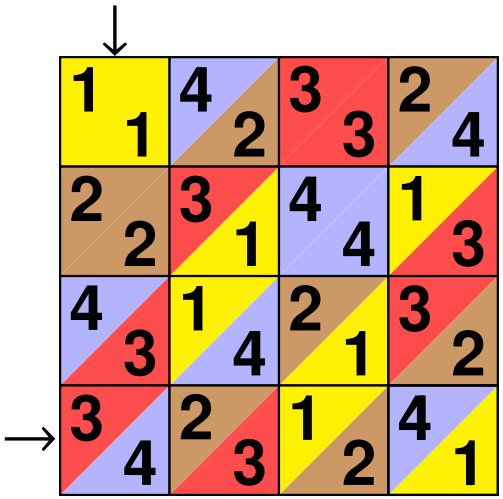


(a) Rule 150 (b) Rule 90 (c) Superposition

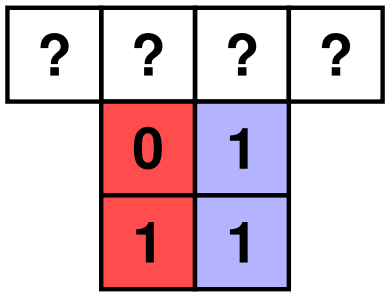
Figure: $P_{150}(X) = 1 + X + X^2$, $P_{90}(X) = 1 + X^2$ (coprime)

Inversion Problem in OCA [Mariot18b]

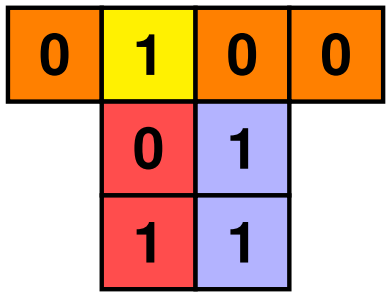
- **Input**: A pair $w, z \in \{0,1\}^{n-1}$ of final configurations
- **Output**: The unique preimage x generating w, z under the action of two OCA



(a) rule 90-150



(b) Input

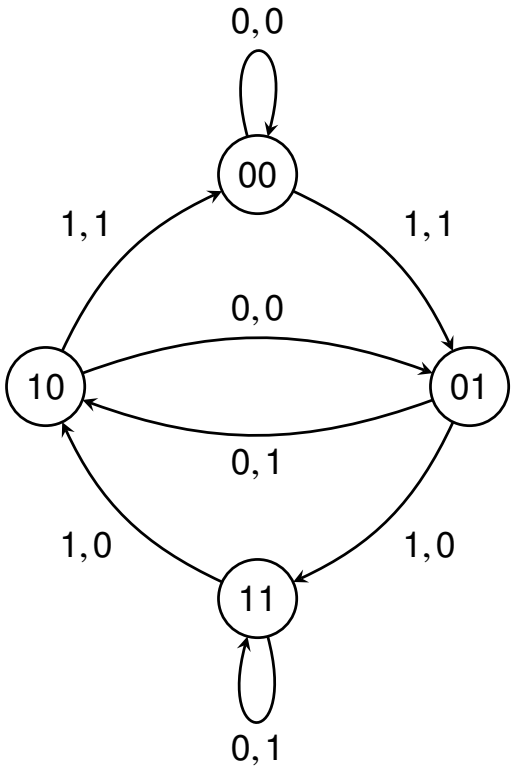


(c) Output

Coupled De Bruijn Graph

Idea: Walk on the De Bruijn graph labelled with both rules until a matching path is found.

(x_1, x_2, x_3)	f_{90}	f_{150}
000	0	0
100	1	1
010	0	1
110	1	0
001	1	1
101	0	0
011	1	0
111	0	1

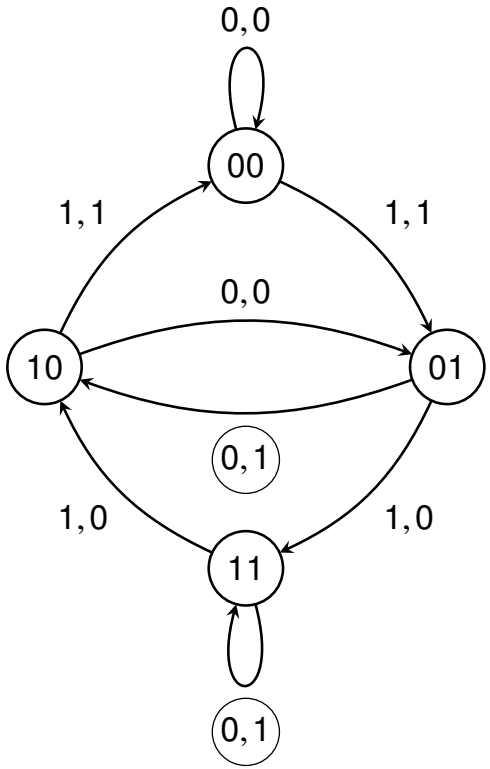


?	?	?	?
	0	1	
	1	1	

Coupled De Bruijn Graph

Idea: Walk on the De Bruijn graph labelled with both rules until a matching path is found.

(x_1, x_2, x_3)	f_{90}	f_{150}
000	0	0
100	1	1
010	0	1
110	1	0
001	1	1
101	0	0
011	1	0
111	0	1

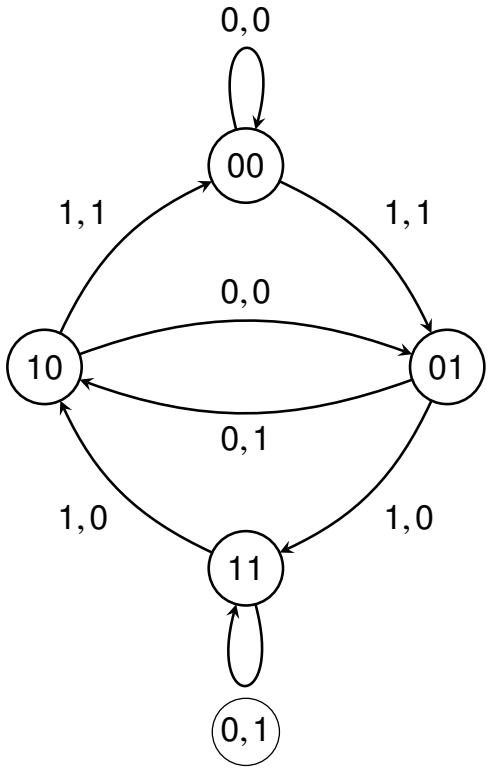


?	?	?	?
	0	1	
	1	1	

Coupled De Bruijn Graph

Idea: Walk on the De Bruijn graph labelled with both rules until a matching path is found.

(x_1, x_2, x_3)	f_{90}	f_{150}
000	0	0
100	1	1
010	0	1
110	1	0
001	1	1
101	0	0
011	1	0
111	0	1

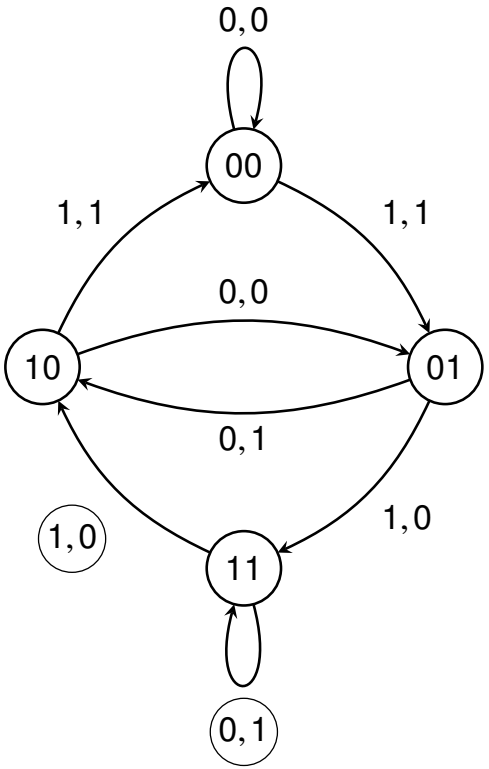


1	**1**	**?**	**?**
	0	**1**	
	1	**1**	

Coupled De Bruijn Graph

Idea: Walk on the De Bruijn graph labelled with both rules until a matching path is found.

(x_1, x_2, x_3)	f_{90}	f_{150}
000	0	0
100	1	1
010	0	1
110	1	0
001	1	1
101	0	0
011	1	0
111	0	1

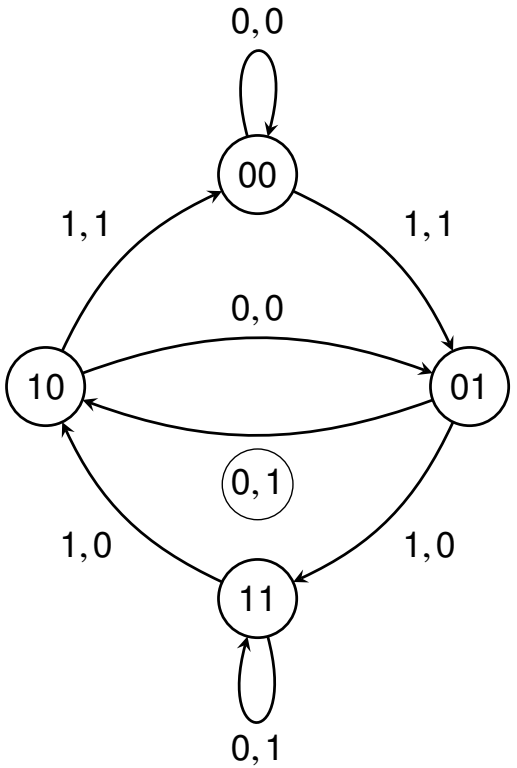


1	**1**	**?**	**?**
	0	**1**	
	1	**1**	

Coupled De Bruijn Graph

Idea: Walk on the De Bruijn graph labelled with both rules until a matching path is found.

(x_1, x_2, x_3)	f_{90}	f_{150}
000	0	0
100	1	1
010	0	1
110	1	0
001	1	1
101	0	0
011	1	0
111	0	1

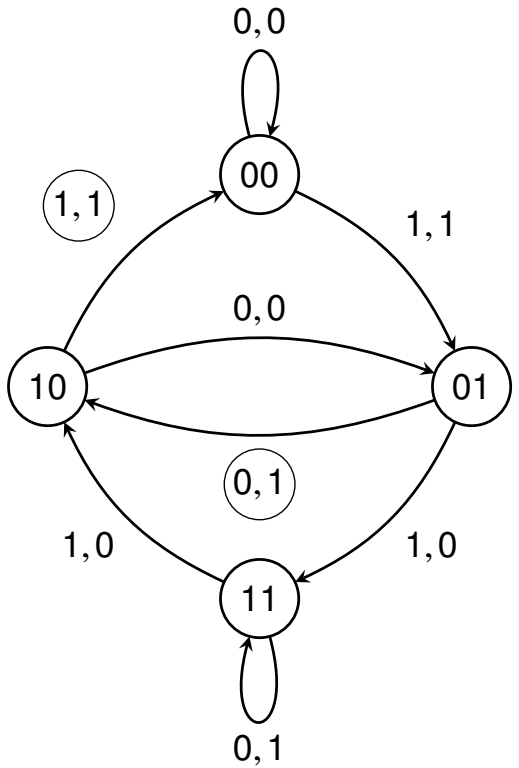


0	**1**	**?**	**?**
	0	**1**	
	1	**1**	

Coupled De Bruijn Graph

Idea: Walk on the De Bruijn graph labelled with both rules until a matching path is found.

(x_1, x_2, x_3)	f_{90}	f_{150}
000	0	0
100	1	1
010	0	1
110	1	0
001	1	1
101	0	0
011	1	0
111	0	1



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

Inversion Algorithm

INVERT-OCA($G_{DB}(f,g), w, z$)

V := VERTEX($G_{DB}(f,g)$)
E := EDGES($G_{DB}(f,g)$)
l := LABELS($G_{DB}(f,g)$)
c := NIL
while $e \in \{(v_1,v_2) \in E : \ l(v_1,v_2) = (w_1,z_1)\}$ AND $c = NIL$ **do**
  c := DFS-MOD(V, E, l, v_1, w, z)
end while
return c

Theorem

Given two OCA rules $f,g : \{0,1\}^n \to \{0,1\}$ and two final configurations $w,z \in \{0,1\}^{n-1}$, algorithm INVERT-OCA *returns the preimage $x \in \{0,1\}^{2(n-1)}$ of w,z in $O(n \cdot 2^n)$ steps*

Conclusions and Future Directions

Conclusions

Evolutionary Design of Crytographic Primitives

- Evolutionary Algorithms (EA) represent an interesting method to design CI Boolean functions via Orthogonal Arrays
- GP seems the best heuristic for this problem, although it explores a larger solution space

Secret Sharing Schemes from Cellular Automata

- We designed an algorithm to implement the recovery phase of a $(2, n)$-SSS based on linear CA
- Drawback: the De Bruijn graph size is exponential in the CA diameter

Future Directions

Evolutionary Design of Crytographic Primitives

- Investigate why GP has better performances than GA, using fitness landscape analysis
- New fitness functions based on the Walsh transform
- Incremental approach to evolve OA, one column at a time

Secret Sharing Schemes from Cellular Automata

- Adapt the Invert-OCA algorithm to work with the Algebraic Normal Form of the local rule
- Explore the sets of MOCA using the Invert-OCA algorithm

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)
- [Mariot19] Mariot, L., Gadouleau, M., Formenti, E., Leporati, A.: Mutually orthogonal latin squares based on cellular automata. Des. Codes Cryptogr. (2019) doi:10.1007/s10623-019-00689-8
- [Mariot18a] Mariot, L., Picek, S., Jakobovic, D., Leporati, A.: Evolutionary Search of Binary Orthogonal Arrays. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.): PPSN 2018 (I). LNCS vol. 11101, pp. 121–133. Springer (2018)
- [Mariot18b] Mariot, L., Leporati, A.: Inversion of Mutually Orthogonal Cellular Automata. In: Mauri, G., El Yacoubi, S., Dennunzio, A., Nishinari, K., Manzoni, L. (eds.): ACRI 2018. LNCS vol. 11115, pp. 364–376. Springer (2018)
- [Mariot16] Mariot, L., Formenti, E., Leporati, A.: Constructing Orthogonal Latin Squares from Linear Cellular Automata. In: Exploratory papers of AUTOMATA 2016. CoRR abs/1610.00139 (2016)
- [Millan98] Millan, W., Clark, J., Dawson, E.: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. EUROCRYPT 1998: pp. 489–499 (1998)
- [Shamir79] Shamir, A.: How to share a secret. Commun. ACM 22(11):612–613 (1979)