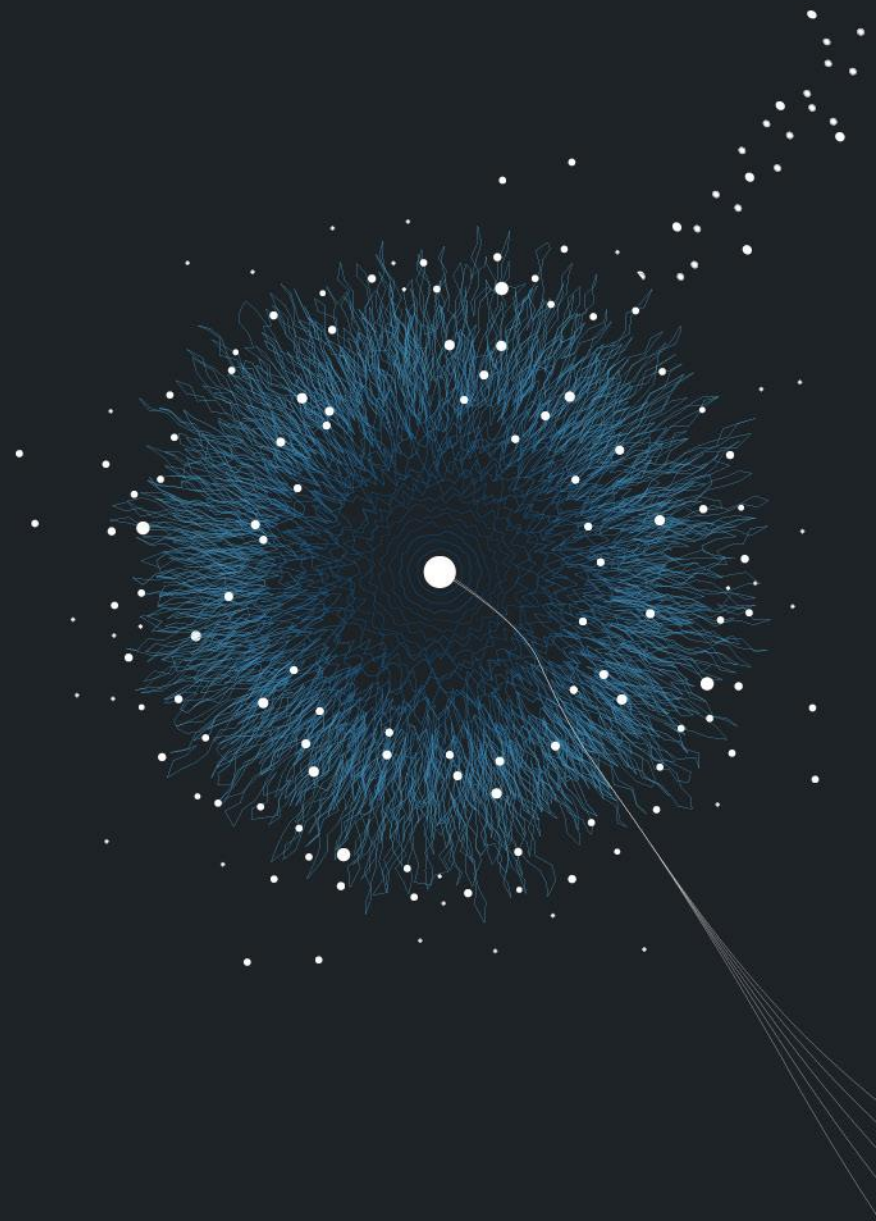
EEMCS - SCS

METAHEURISTIC ALGORITHMS IN CRYPTOGRAPHY

GUEST LECTURE – SEPTEMBER 21, 2026

LUCA MARIOT



UNIVERSITY OF TWENTE.

ABOUT ME

Dr. Luca Mariot
E-Mail: l.mariot@utwente.nl

Assistant Professor in the Semantics, Cybersecurity and Services group (SCS), University of Twente

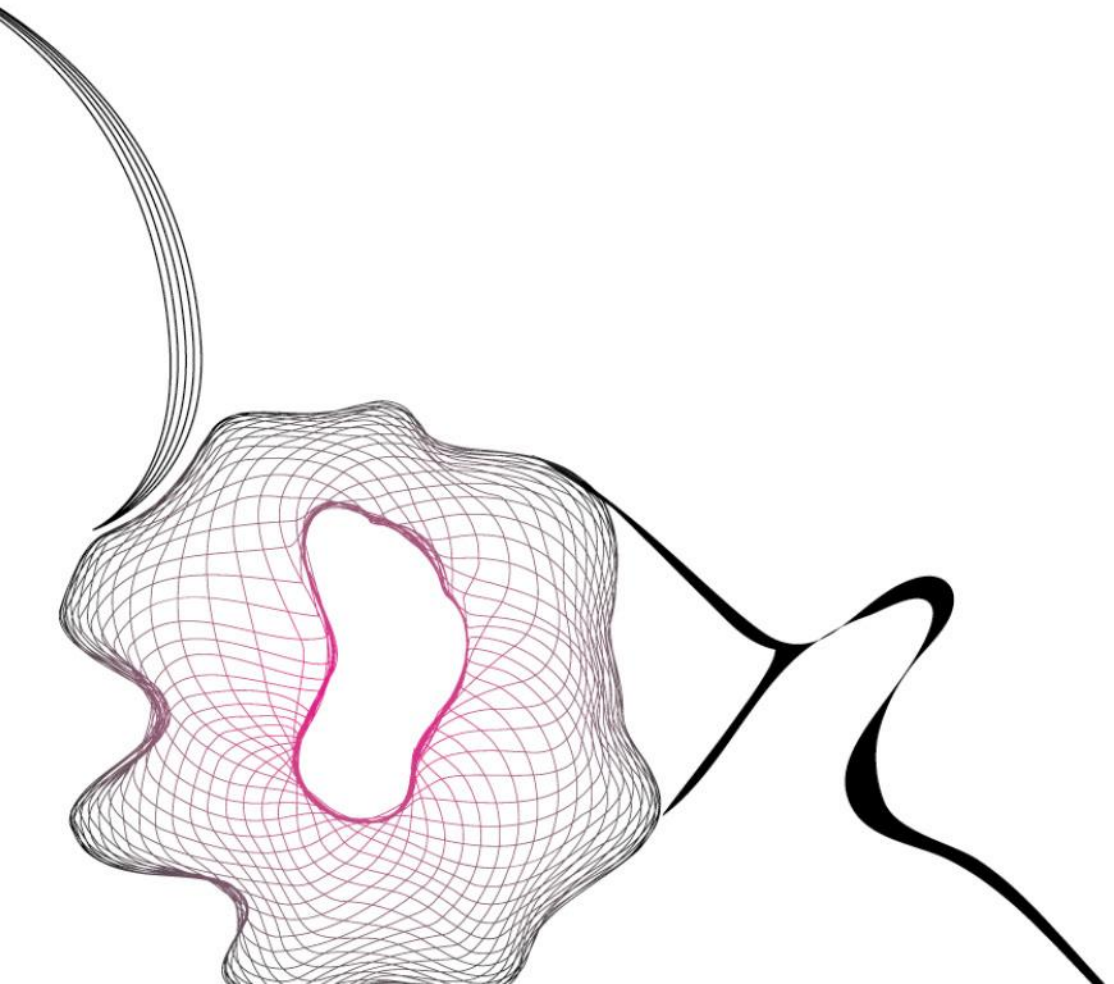
Research Interests:

- Cryptography
- Cellular Automata
- Evolutionary Computing
- Boolean functions
- Machine Learning

Website: https://lucamariot.org

UNIVERSITY OF TWENTE.

SUMMARY

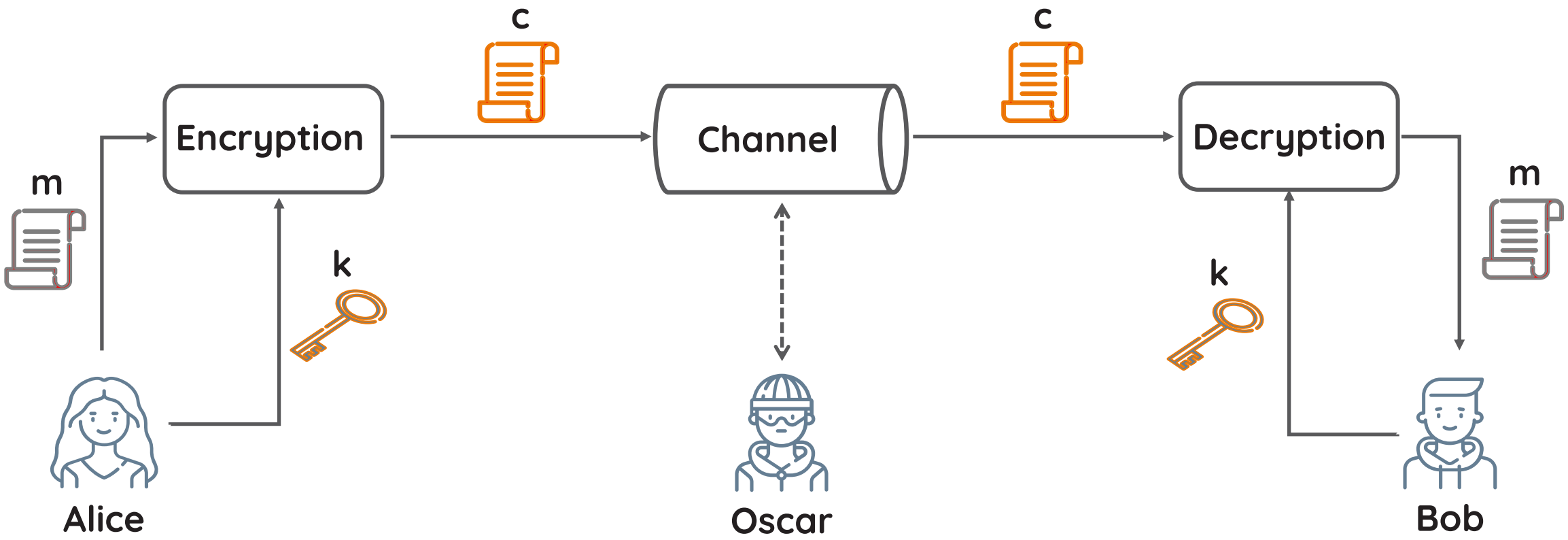


- AI-driven design in symmetric cryptography
- EA for Boolean Functions
- EA for S-boxes
- Neuroevolution for Side-Channel Analysis

AI-DRIVEN DESIGN IN SYMMETRIC CRYPTOGRAPHY

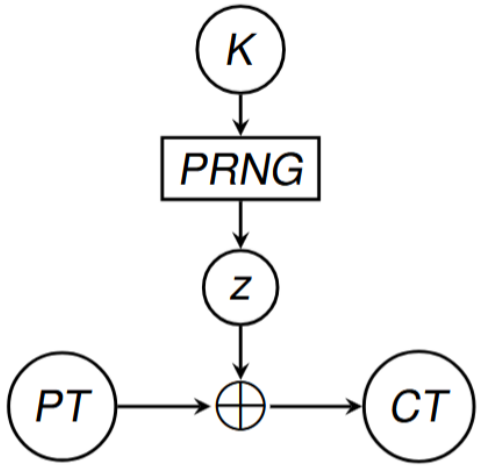
UNIVERSITY
OF TWENTE.

SYMMETRIC-KEY ENCRYPTION SCENARIO

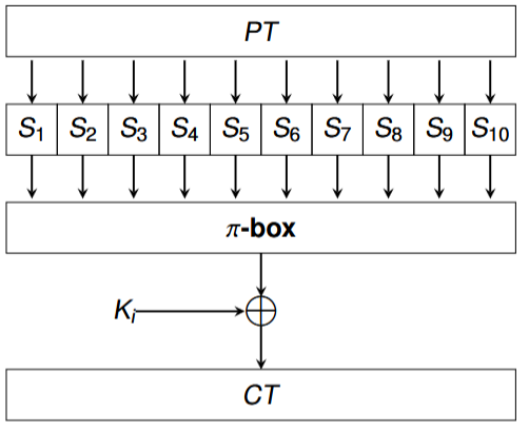


m: plaintext message **k:** encryption/decryption key **c:** ciphertext message

PRIMITIVES IN SYMMETRIC CRYPTOGRAPHY



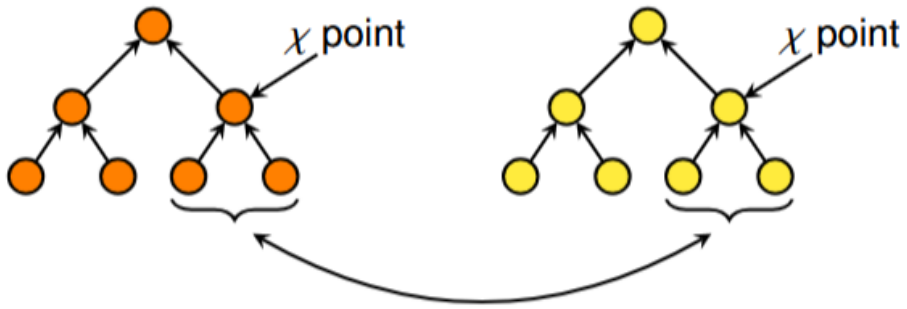
(a) Stream cipher

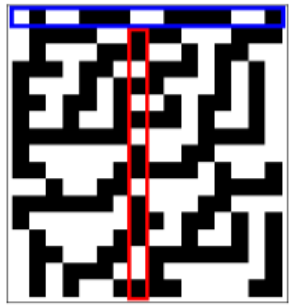


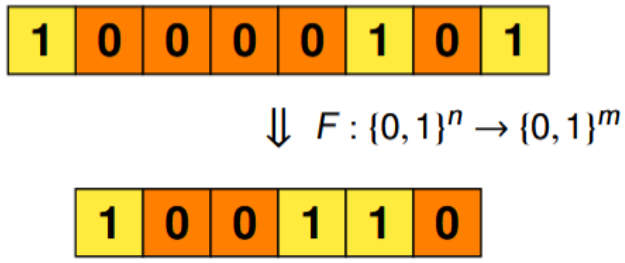
(b) Block cipher

- Symmetric ciphers require several low-level **primitives**:
  - Pseudorandom number generators (PRNG)
  - Boolean functions $f : \{0,1\}^n \to \{0,1\}$
  - S-boxes $F : \{0,1\}^n \to \{0,1\}^n$
  - Diffusion layers, …
- Classic methods to build them: **algebraic constructions** [4]

AI-DRIVEN DESIGN OF CRYPTO PRIMITIVES

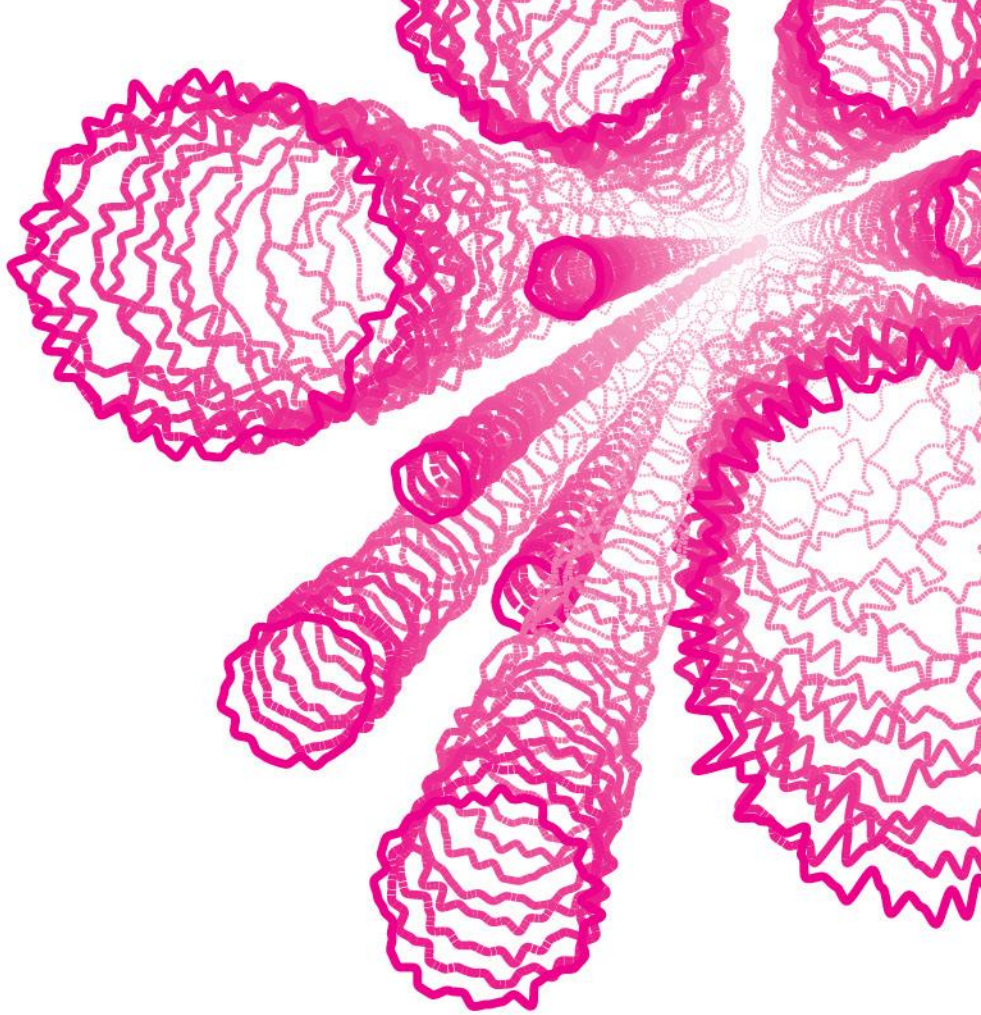






- Algebraic constructions are not flexible
- **AI-driven approach:** support the cipher designer in designing the primitives with:
  - Metaheuristic optimization techniques (Evolutionary Algorithms, …) [6]
  - Nature-inspired computational models (Cellular Automata, …) [11]

EVOLUTIONARY ALGORITHMS FOR BOOLEAN FUNCTIONS

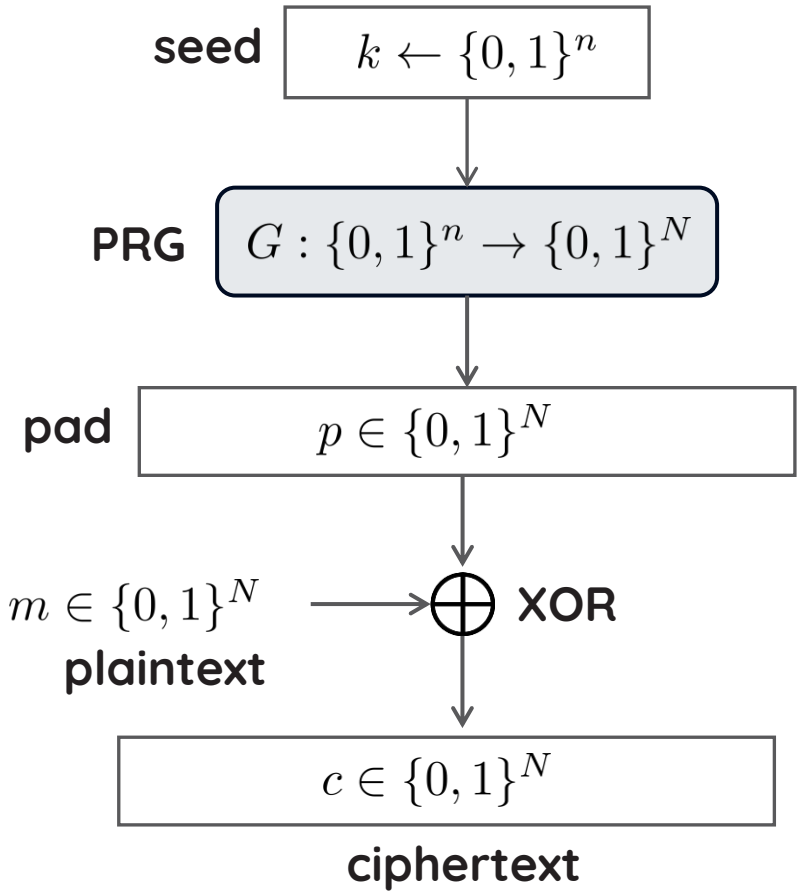


UNIVERSITY OF TWENTE.

STREAM CIPHERS - ENCRYPTION

- **Idea:** Alice encode all messages as stream of bits, $m \in \{0,1\}^N$
- A **Pseudo Random Generator (PRG)** is used to generate a **pad** $p \in \{0,1\}^N$ of the same length of the message [7]
- The seed of the PRG is the **key** $k \leftarrow \{0,1\}^n$
- **Encryption:** Bitwise XOR between message and pad

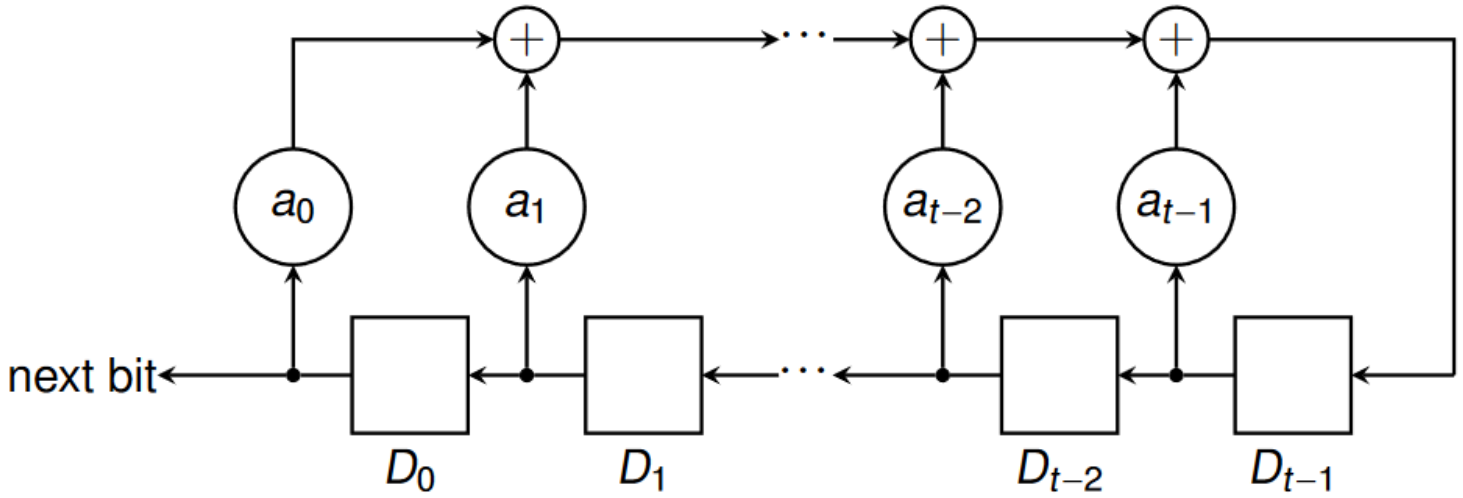
$$c_i := m_i \oplus p_i, \text{ for all } i \in \{1, \cdots, N\}$$



LINEAR FEEDBACK SHIFT REGISTERS (LFSR)

- A device computing a *linear recurring sequence* (LRS)

$$z_i = a_0 \cdot z_{i-t} \oplus a_1 \cdot z_{i-t+1} \oplus \cdots \oplus a_{t-1} \cdot z_{i-1} \qquad a_j, z_j \in \{0, 1\}$$



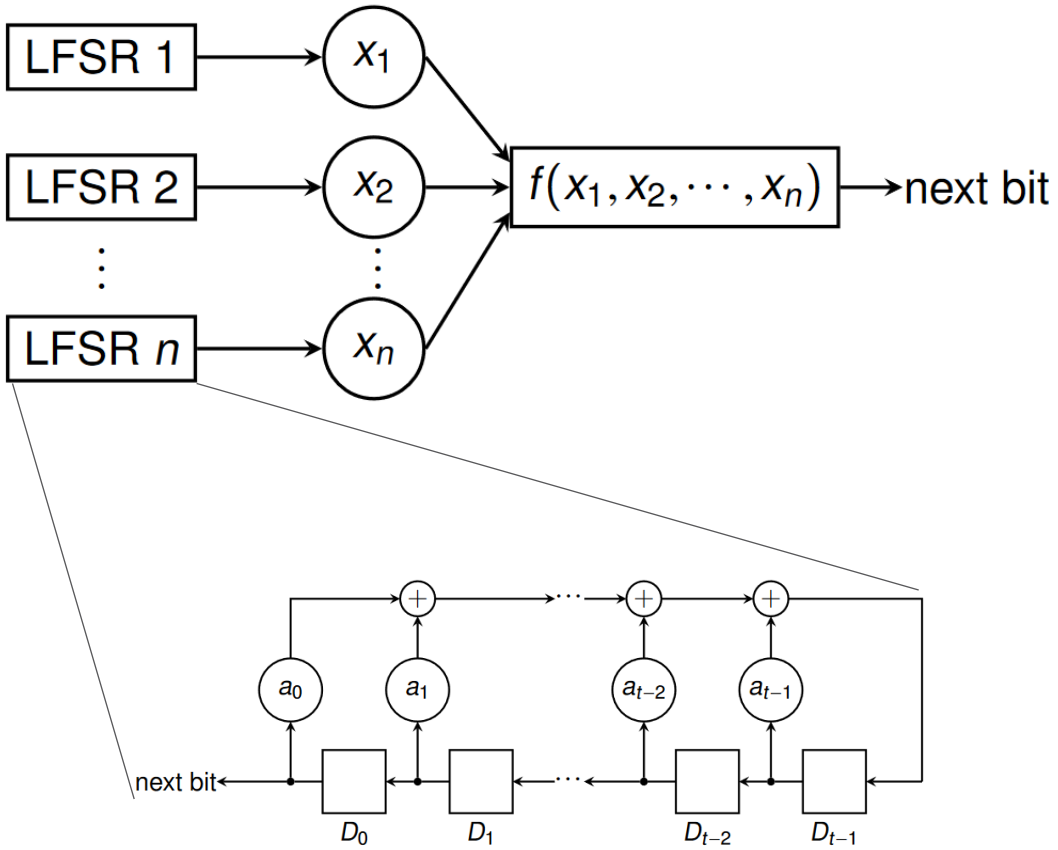
- **Problem:** very weak as a cryptographic PRG [4, 7]

IMPROVING LFSR – COMBINER MODEL FOR PRG

- **Idea:** use n LFSRs instead of one [4]
- LFSRs outputs *combined* using a Boolean function:

$$f : \{0,1\}^n \rightarrow \{0,1\}$$

- Security of the PRG: **cryptographic properties of** $f : \{0,1\}^n \rightarrow \{0,1\}$



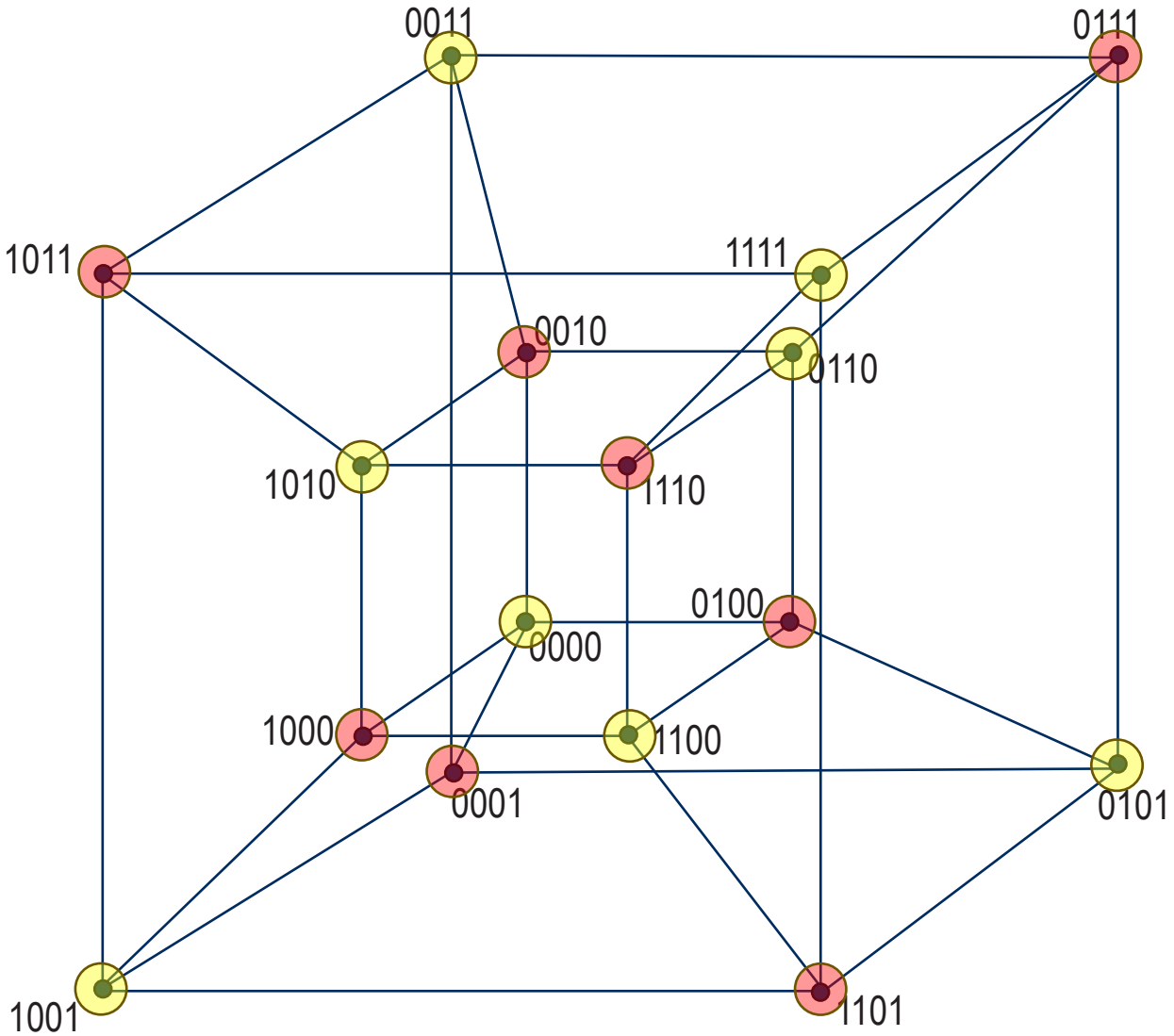
BOOLEAN FUNCTIONS

- A mapping $f : \{0,1\}^n \to \{0,1\}$ represented by a *truth table*

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

- The function must satisfy some properties to resist specific attacks [4]:
  - **Balancedness** (equal number of 0s and 1s)
  - High **Nonlinearity** (Hamming distance from linear functions)
  - High algebraic degree, etc. …

NONLINEARITY – GEOMETRIC INTUITION



- **Example**: $n = 2$ variables (truth tables of $2^2 = 4$ bits, 16 functions in total)
- Linear functions and complements:

$a_1x_1 \oplus a_2x_2$	TT	TT $\oplus 1$
0	0000	1111
x_1	0011	1100
x_2	0101	1010
$x_1 \oplus x_2$	0110	1001

- Hamming distance from linear functions: **1**, e.g.: $f(x_1, x_2) = x_1x_2$

WALSH TRANSFORM

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

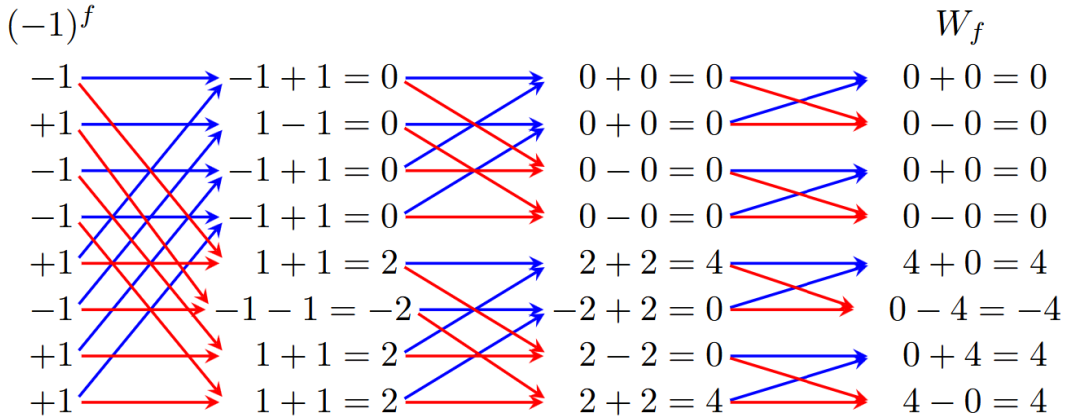
- Measures correlation with *linear* functions:

$$L_a(x) = a \cdot x = a_1 x_1 \oplus \ldots \oplus a_n x_n,$$

- Can be computed in $\mathcal{O}(n2^n)$ steps (FFT-like algorithm)

- Nonlinearity is equal to:

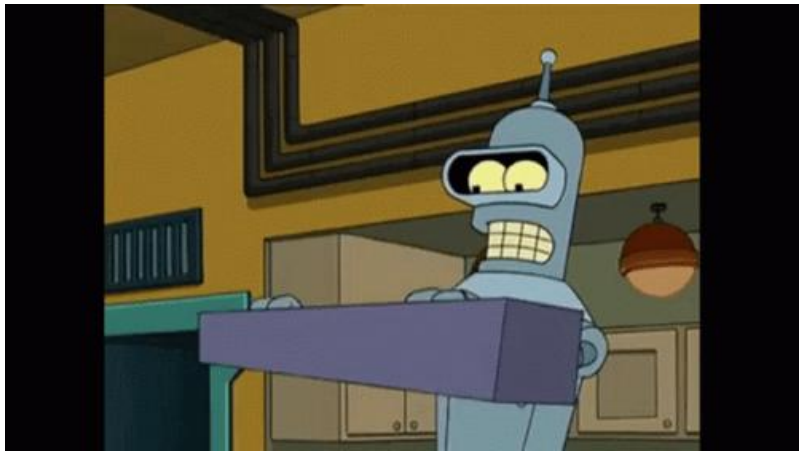
$$nl_f = 2^{n-1} - \frac{1}{2} \max_{a \in \mathbb{F}_2^n} |W_f(a)|$$



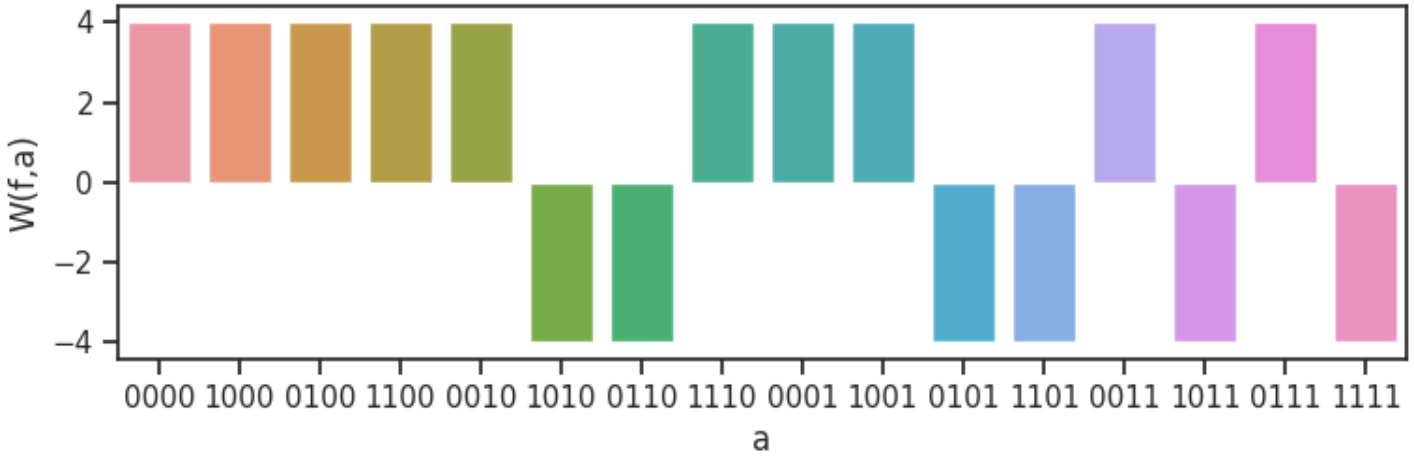
Example of Fast Walsh Transform calculation with 3-variable functions

UNIVERSITY OF TWENTE.

BENT FUNCTIONS



- Only two spectral values: $W_f(a) \in \{-2^{\frac{n}{2}}, +2^{\frac{n}{2}}\}$
- **PRO**: highest possible nonlinearity $nl_f = 2^{n-1} - 2^{\frac{n}{2}-1}$
- **CONS**: unbalanced, exist only for n even [4]



Example of Walsh spectrum of a bent function in 4 variables

OPTIMIZATION PROBLEM

- Given $n \in \mathbb{N}$, how do we fill the table so that f is balanced and highly nonlinear?

(x_1, x_2, x_3)	000	001	010	011	100	101	110	111
$f(x_1, x_2, x_3)$	?	?	?	?	?	?	?	?

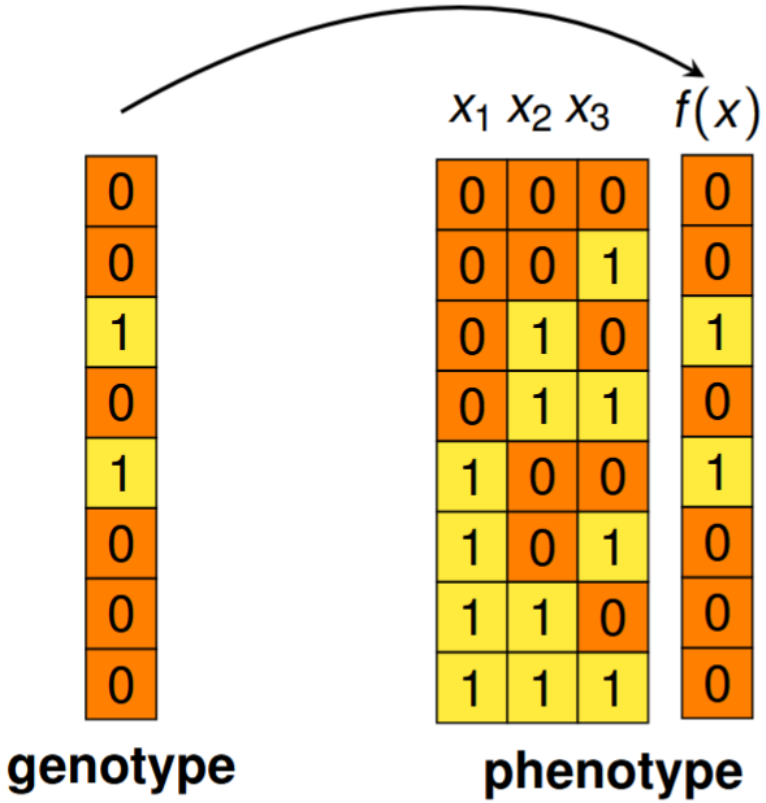
$$f^* = \text{argmax}_{f:\{0,1\}^n \to \{0,1\}}(BAL(f) + NL(f))$$

- The truth table has size 2^n so there are 2^{2^n} combinations

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

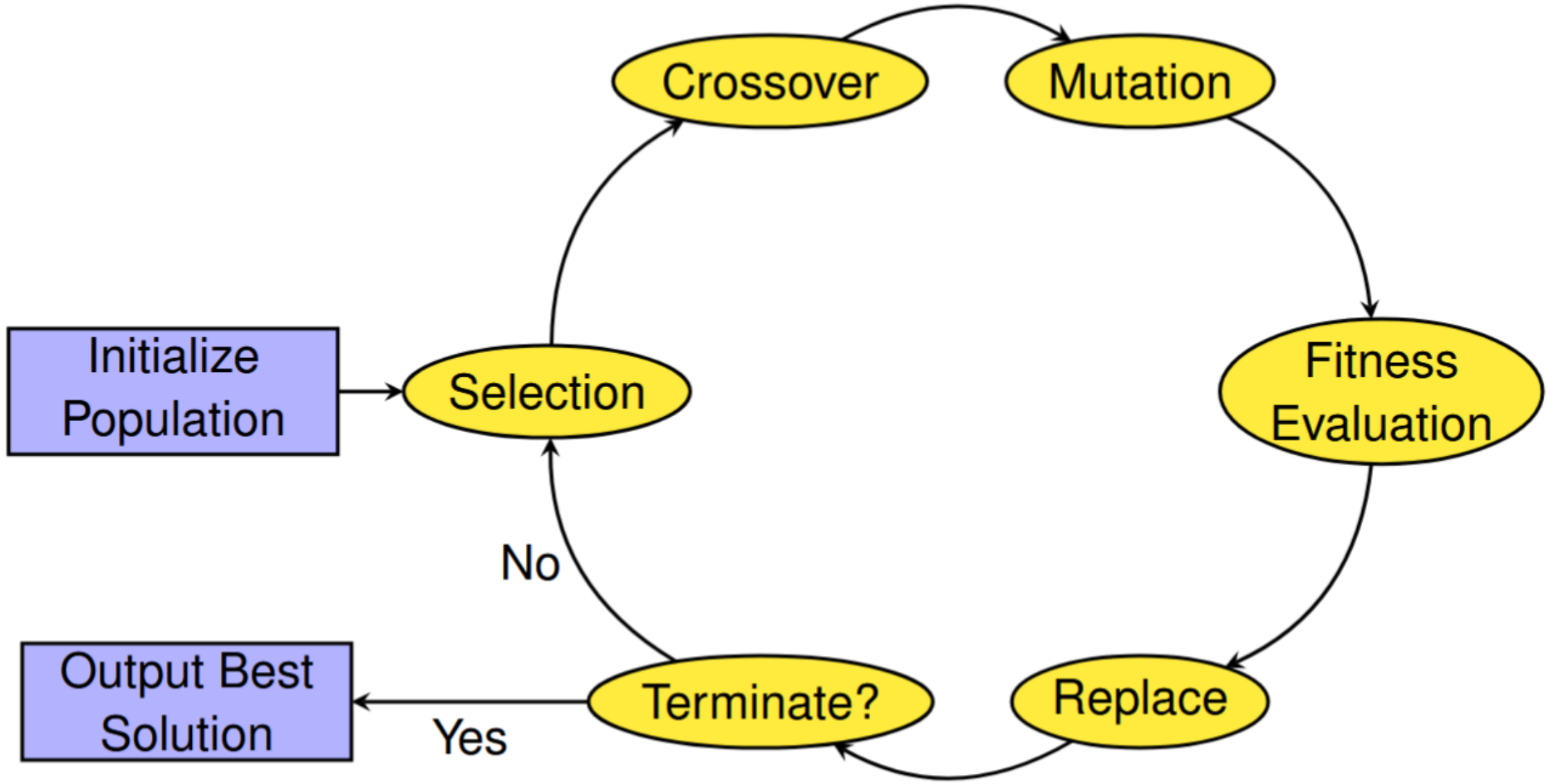
- Exhaustive search is already unfeasible for $n > 5$!

GENETIC ALGORITHMS (GA)

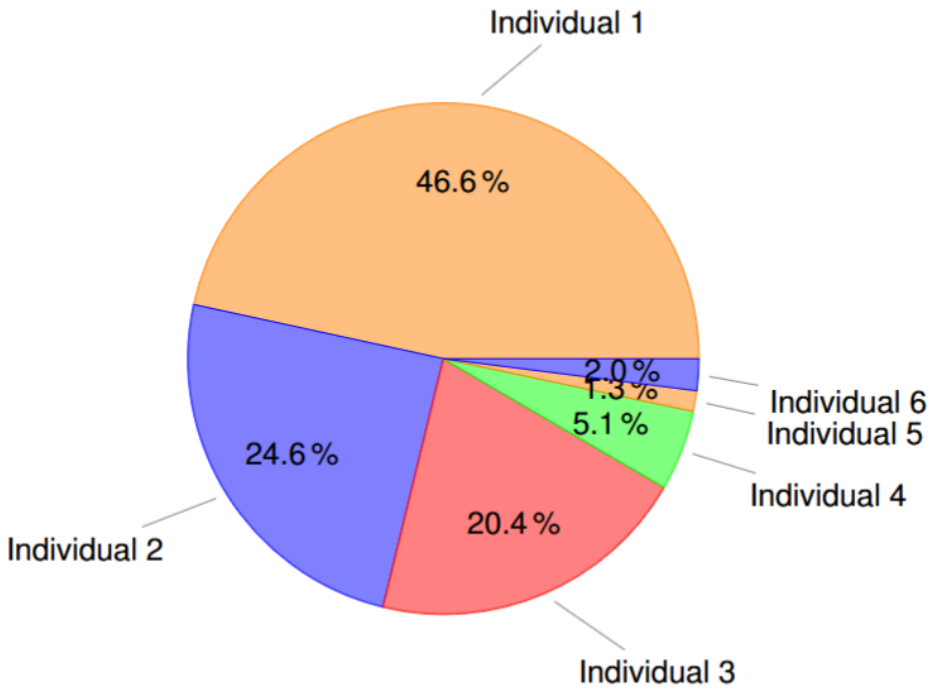


- Introduced by John Holland (1975)
- Optimization algorithms loosely based on evolutionary principles
- **GA genotype**: fixed-length bitstrings
- **phenotype**: truth table of f [14]

THE EA LOOP


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

SELECTION

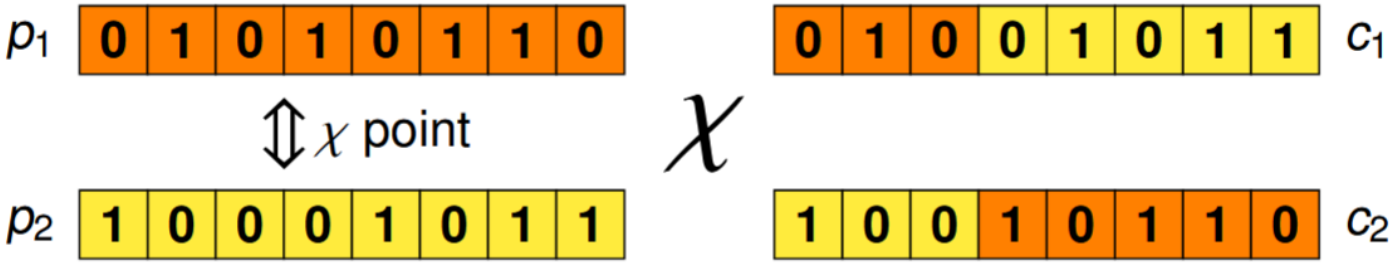


- **Roulette-Wheel:** selection probability proportional to individual's fitness
- **Tournament:** select the fittest individual from a random sample of t individuals
- In general: tournaments work better

CROSSOVER

- **Idea:** recombine the genes of two parents (**Exploitation**)

GA Example: One-Point Crossover

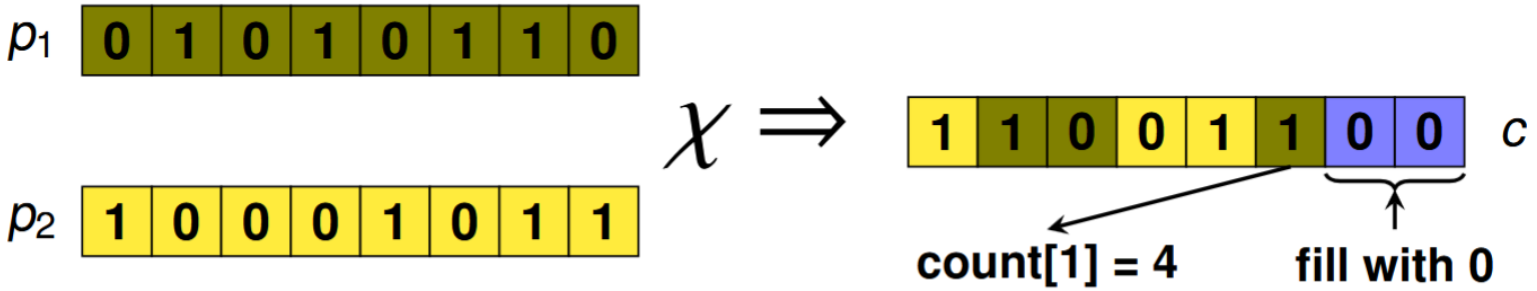


- **Problem:** how dow we ensure balancedness for cryptography?
- We could optimize it in the fitness function…

BALANCED CROSSOVERS

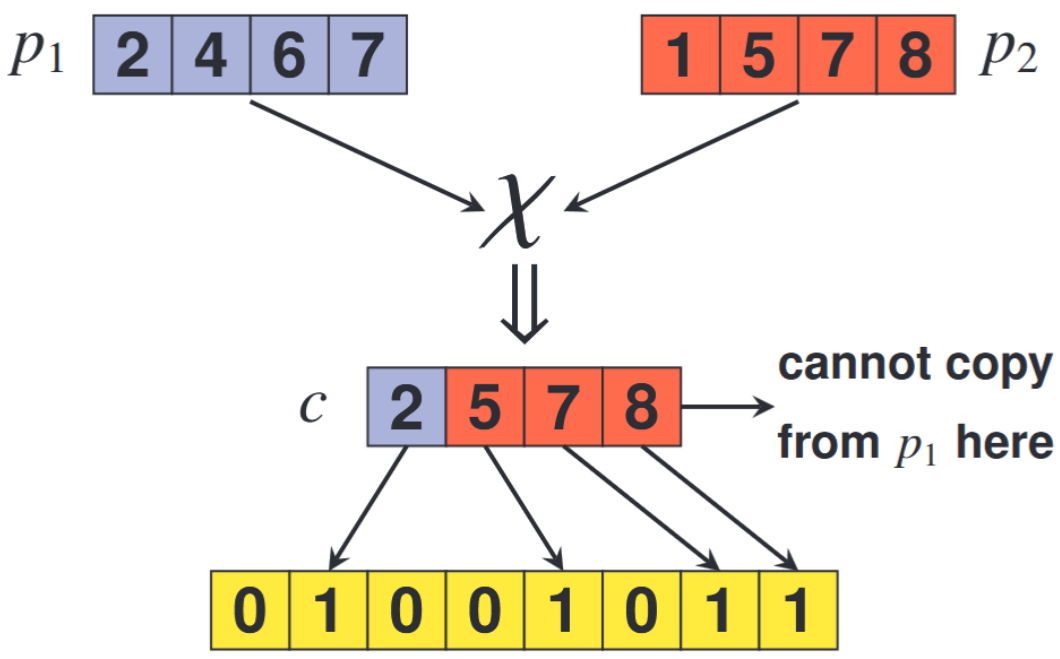
- **Idea:** use **counters** to keep track of the numbers of 1s in the child [9,]

GA Example: Counter-based Crossover



- If we start from balanced parents, we get balanced children

BALANCED CROSSOVERS



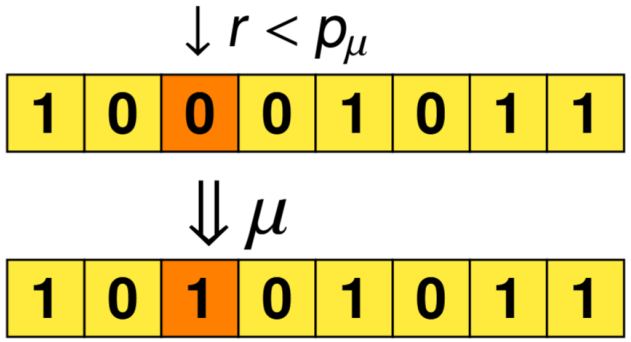
GA Example: Map-of-Ones Crossover

- **Other variant:** encode the positions of the 1s in the bitstring [9, 14]
- Randomly copy from the first or the second parent, going left to right
- Make sure there are no *repeated* values in the offspring

MUTATION

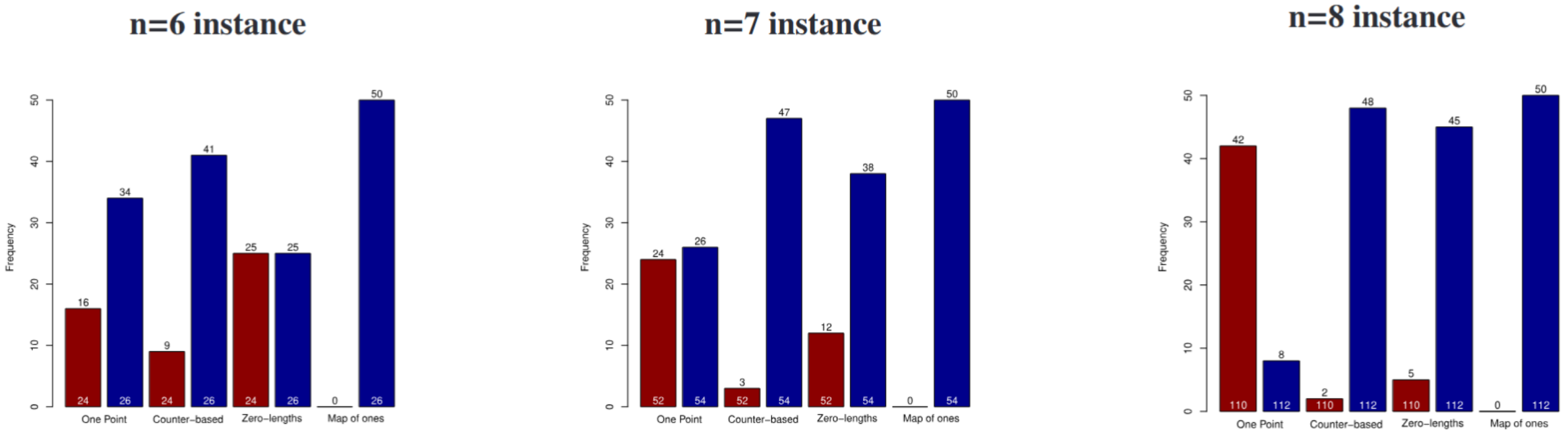
- **Idea:** introduce new "genetic material" in the offspring (**Exploration**)

GA Example: Bit-flip mutation



- For balancedness: randomly **swap** some bits instead of flipping them

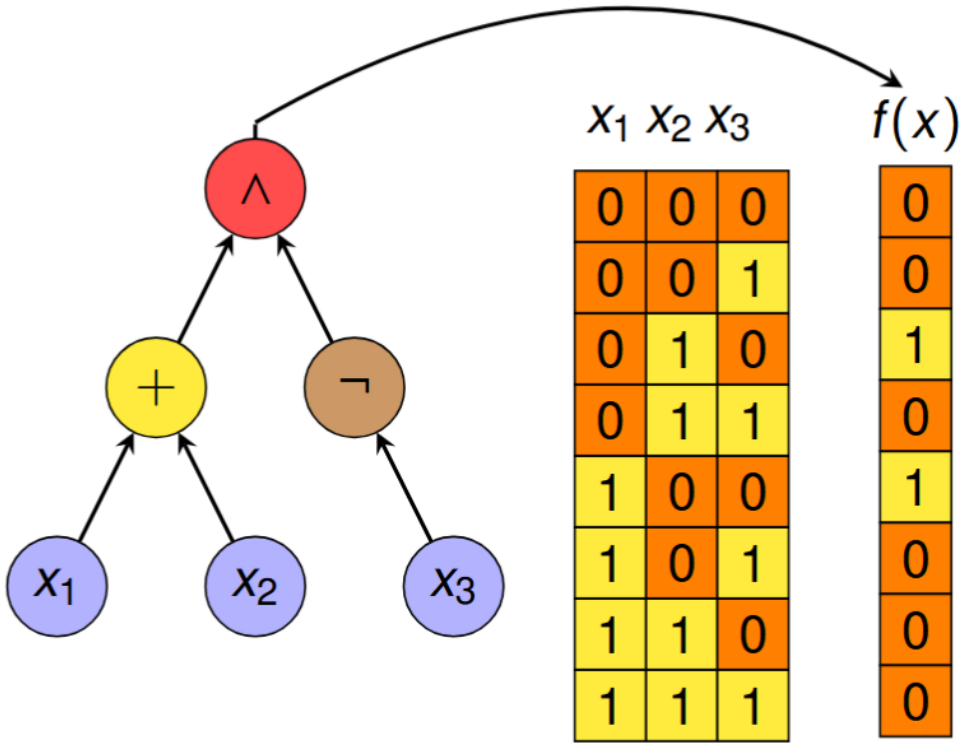
EXPERIMENTAL COMPARISON



In general: balanced operators perform better than one-point crossover, Map-of-Ones is the operator with the best success rate [9]

UNIVERSITY OF TWENTE.

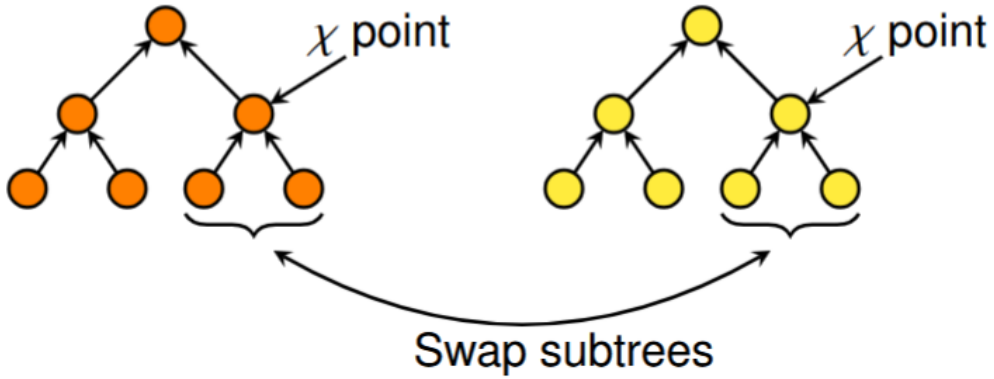
GENETIC PROGRAMMING (GP)



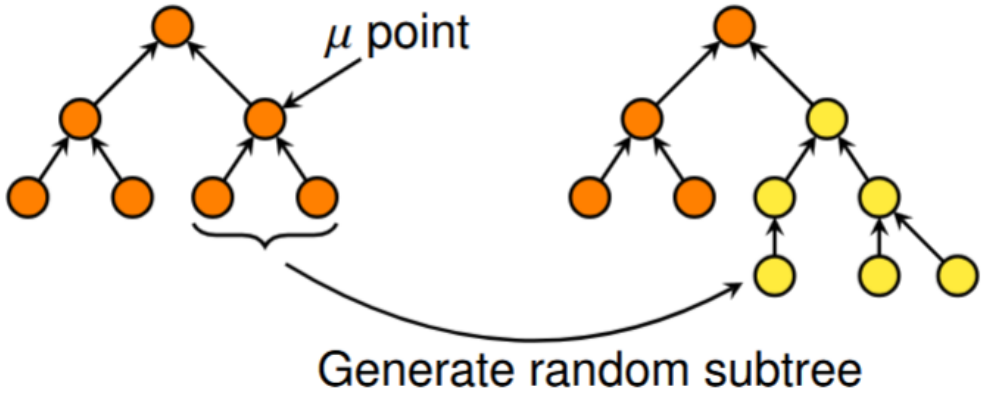
- **Idea**: evolve computer programs to solve specific tasks [8]
- **GP Genotype**: a syntactic tree
  - Leaf nodes: input variables
  - Internal nodes: operators (e.g. AND, OR, NOT, XOR, …)
- **Phenotype:** evaluate the tree for all possible assignments of the leaf nodes

GP CROSSOVER & MUTATION

Subtree Crossover



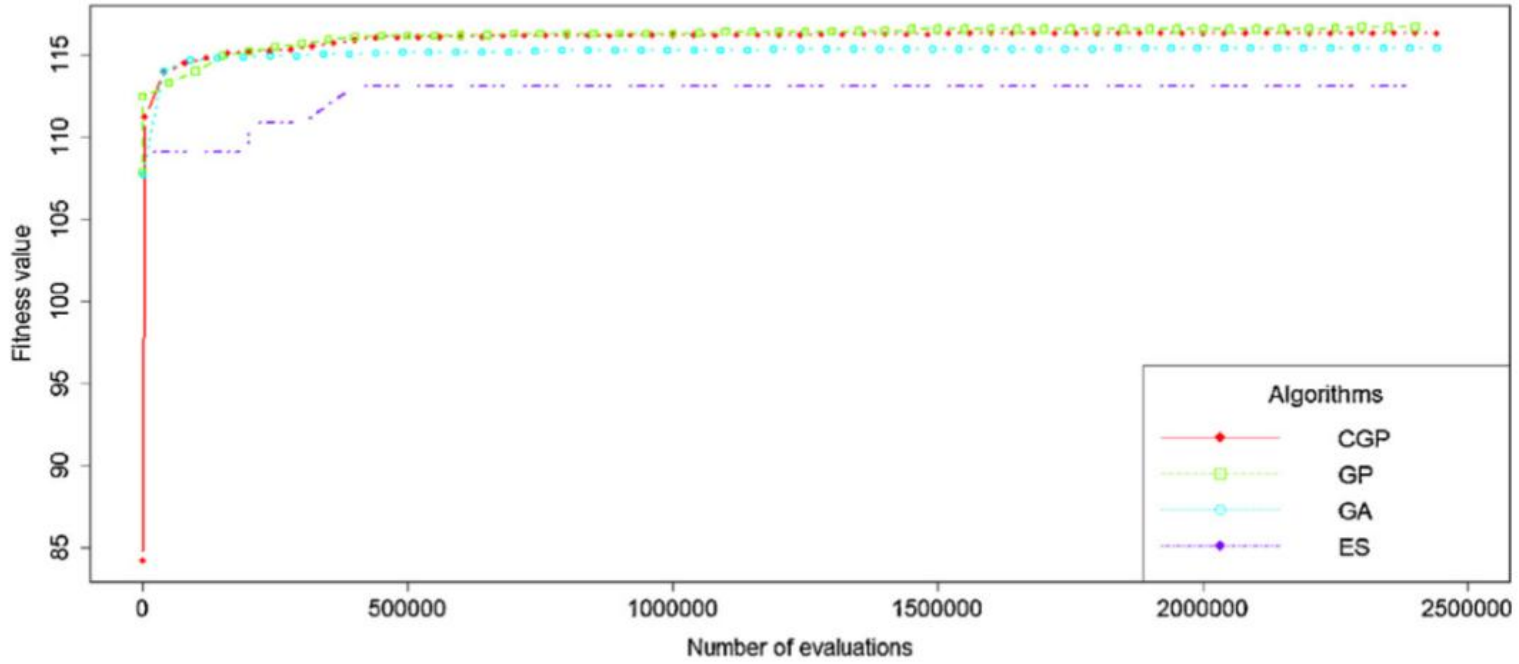
Subtree Mutation



- **Subtree crossover:** select random internal node and swap the two subtrees
- **Subtree mutation:** select random node and generate new random subtree
- **In general:** not possible to preserve the balancedness of the Boolean function
- Difference between the syntax of trees and the semantics of truth tables

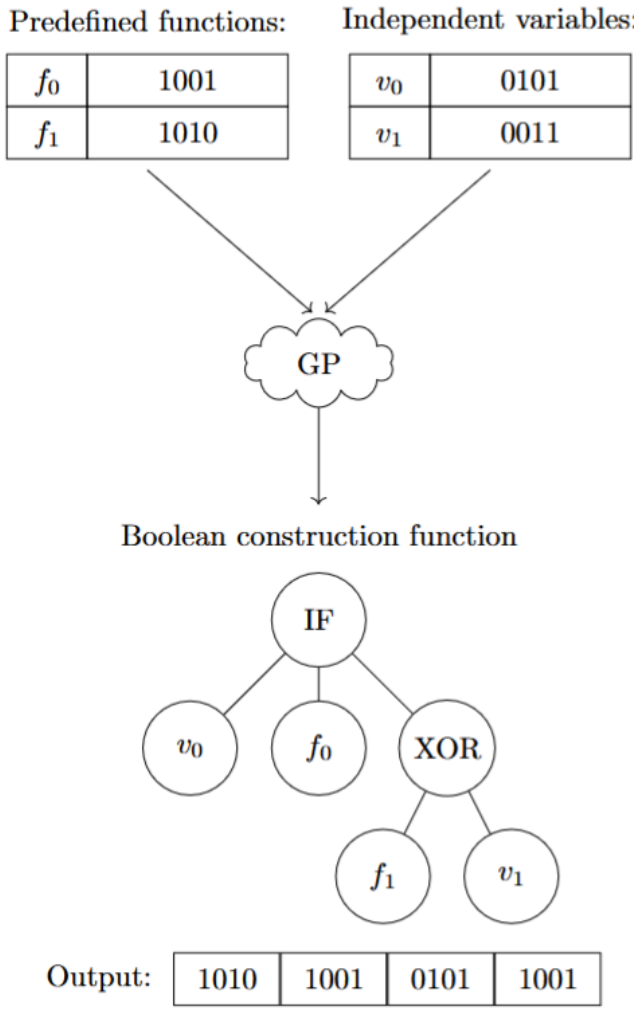
GP VS. GA PERFORMANCES

- GA with balanced operators explores a smaller search space than GP



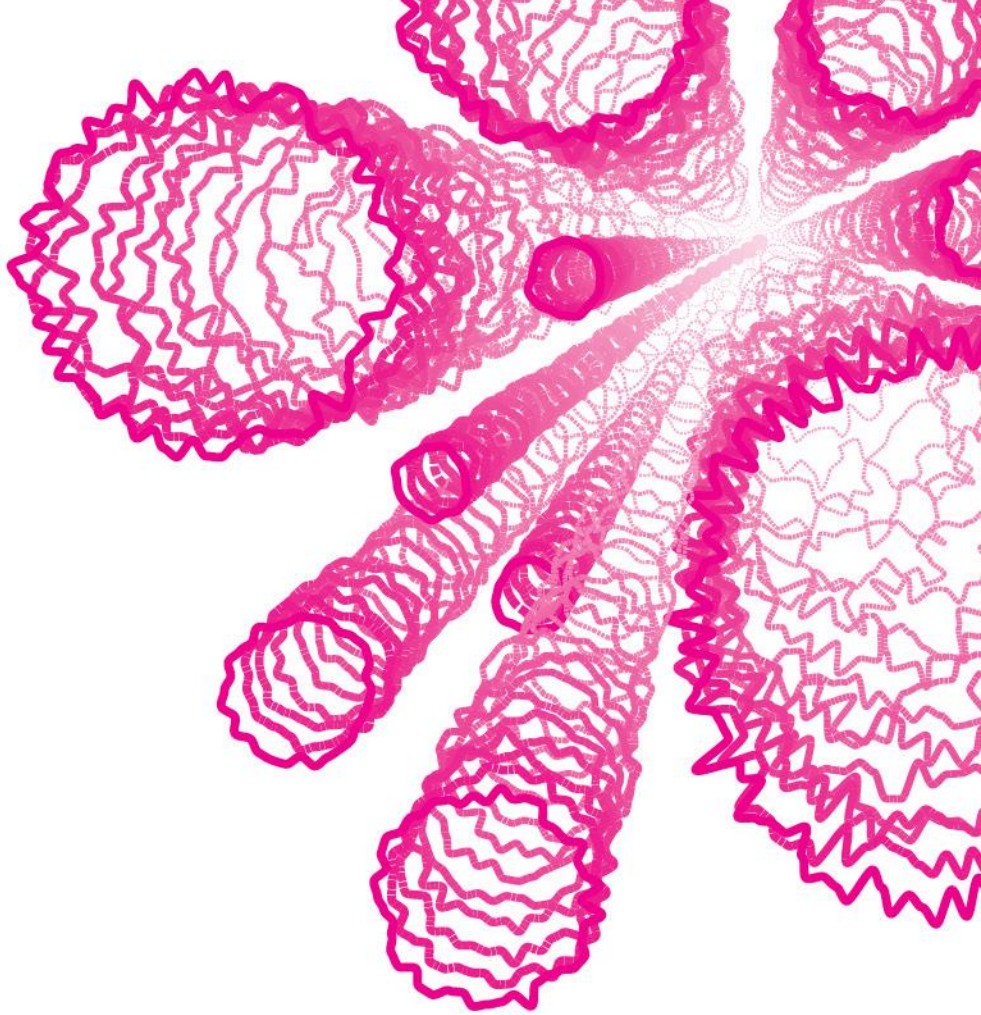
- **Nevertheless:** GP usually fares better than (balanced) GA [6, 12, 17, 19]

EVOLVING ALGEBRAIC CONSTRUCTIONS WITH GP



- **Idea:** Do not evolve functions directly, but rather their algebraic constructions [3, 10, 20]
- Use Boolean minimizers to interpret the constructions
- **Research Question**: Does GP obtain new constructions?
- **Finding**: GP always generates the same known construction, but in a very complicated way [20]

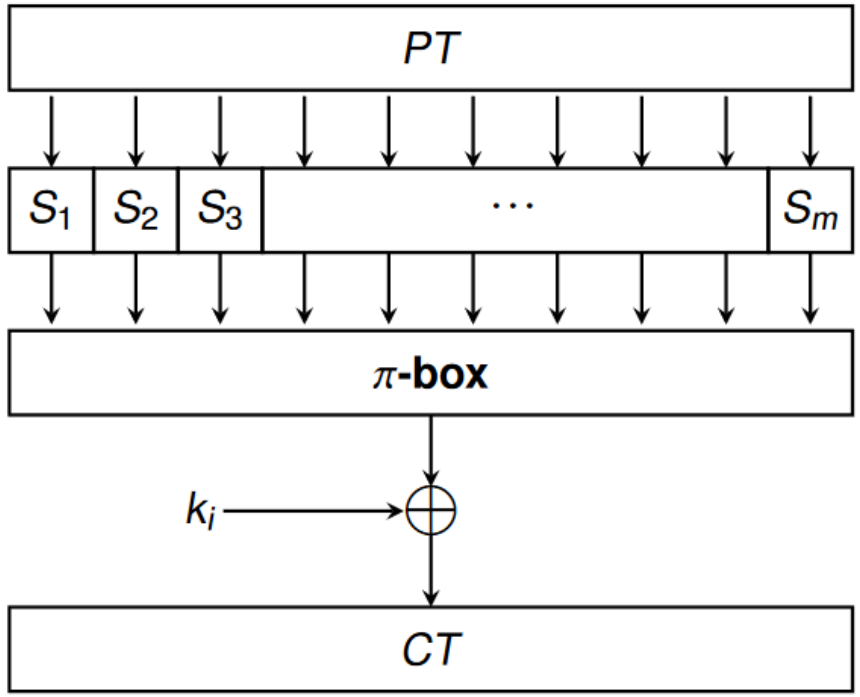
EVOLUTIONARY ALGORITHMS FOR S-BOXES



UNIVERSITY OF TWENTE.

SPN BLOCK CIPHERS

- Round function: composed of **confusion layer** and **diffusion layer** [7]
- Confusion: Substitution boxes (**S-boxes**) of the form $S_i : \{0,1\}^n \rightarrow \{0,1\}^n$
- Diffusion: Permutation box (**P-box**) of the form $\pi : \{0,1\}^b \rightarrow \{0,1\}^b$
- Result is bitwise XORed with round key
- All S-boxes and P-boxes are invertible



S-BOXES

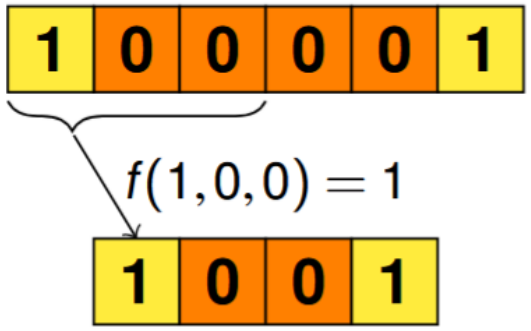
- Vectorial mapping $F : \{0,1\}^n \rightarrow \{0,1\}^m$
- Defined by m *coordinate Boolean functions* $f_i : \{0,1\}^n \rightarrow \{0,1\}$

(x_1, x_2, x_3)	000	001	010	011	100	101	110	111
$dec(x_1, x_2, x_3)$	0	1	2	3	4	5	6	7
$F(x_1, x_2, x_3)$	0	5	6	1	3	2	4	7
$f_1(x_1, x_2, x_3)$	0	1	1	0	0	0	1	1
$f_2(x_1, x_2, x_3)$	0	0	1	0	1	1	0	1
$f_3(x_1, x_2, x_3)$	0	1	0	1	1	0	0	1

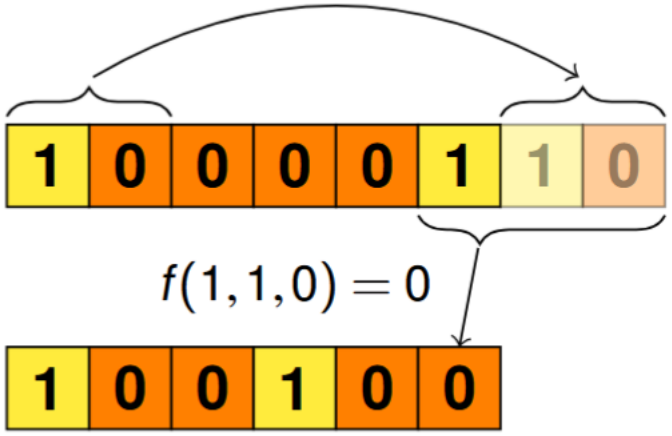
- Again, an S-box needs to satisfy some properties to resist specific attacks (balancedness, nonlinearity, differential uniformity, …) [4]

CELLULAR AUTOMATA (CA)

- Discrete parallel computation model composed of a finite array of *cells*



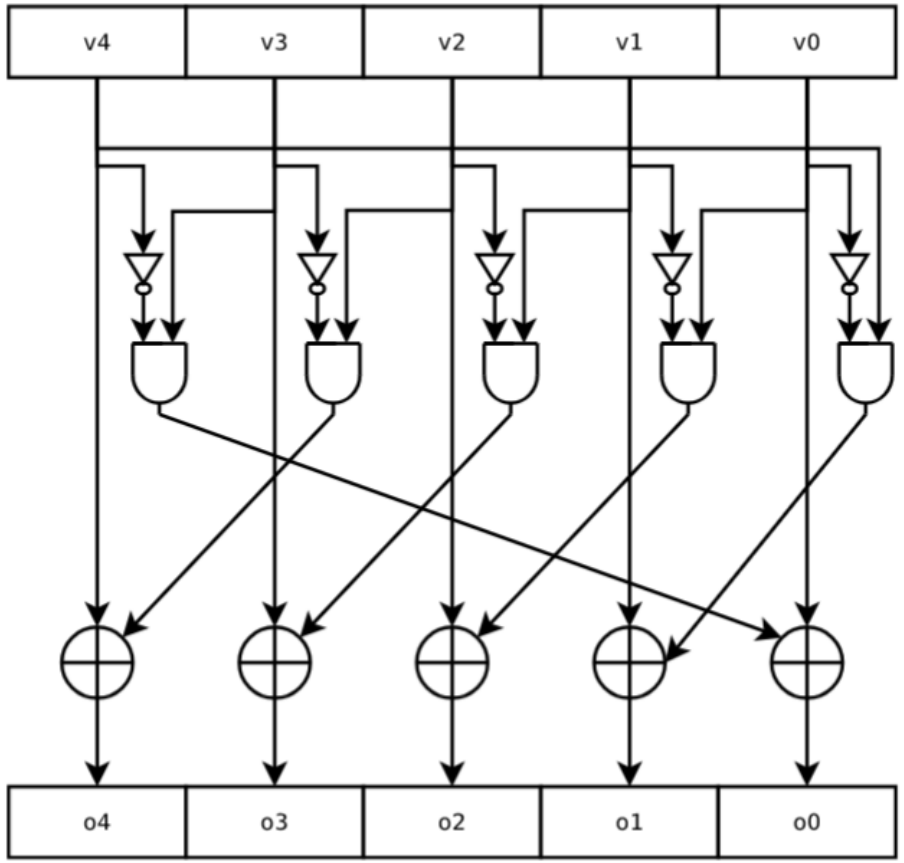
No Boundary CA – NBCA



Periodic Boundary CA – PBCA

- Each cell updates its state by applying a **local rule** $f : \{0,1\}^d \rightarrow \{0,1\}$ on itself and the d-1 right neighboring cells [13]

CA-BASED CRYPTO: KECCAK χ MAPPING

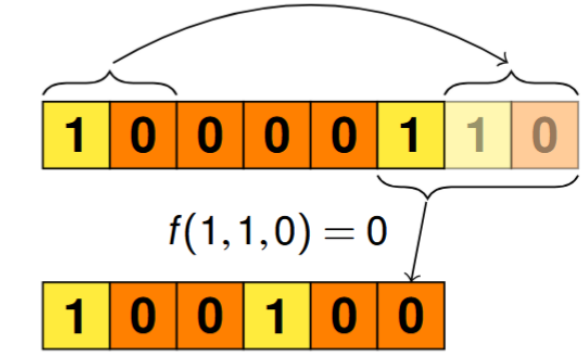


- Local rule of diameter d=3:

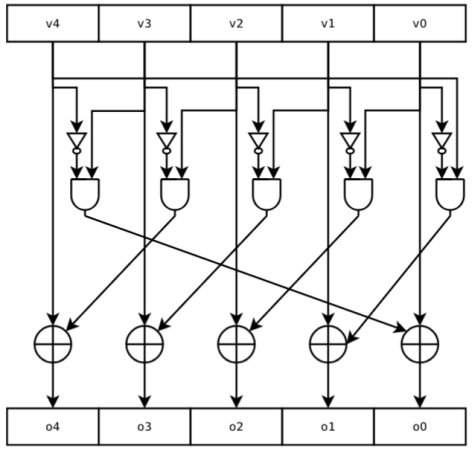
$$\chi(x_1, x_2, x_3) = x_1 \oplus (1 \oplus (x_2 \cdot x_3))$$

- Invertible PBCA for every odd size of the CA array [5]
- Used as a 5x5 S-box in the *Keccak* specification of the SHA-3 standard [2]
- What about larger diameters?

OPTIMIZATION OF CA-BASED S-BOXES PROPERTIES



Periodic Boundary CA – PBCA



- **Advantage:** optimize only the CA local rule, which is single-output Boolean function of d variables
- **Goal:** find PBCA of length n and diameter d = n with:
  - good cryptographic properties [13]
  - low implementation cost [18]
- GP optimizing both crypto (nonlinearity, differential uniformity) and implementation properties (GE measure)

UNIVERSITY OF TWENTE.

RESULTS – CRYPTO PROPERTIES

- Except for n=8, GP obtains CA-based S-boxes with optimal cryptographic properties [18]

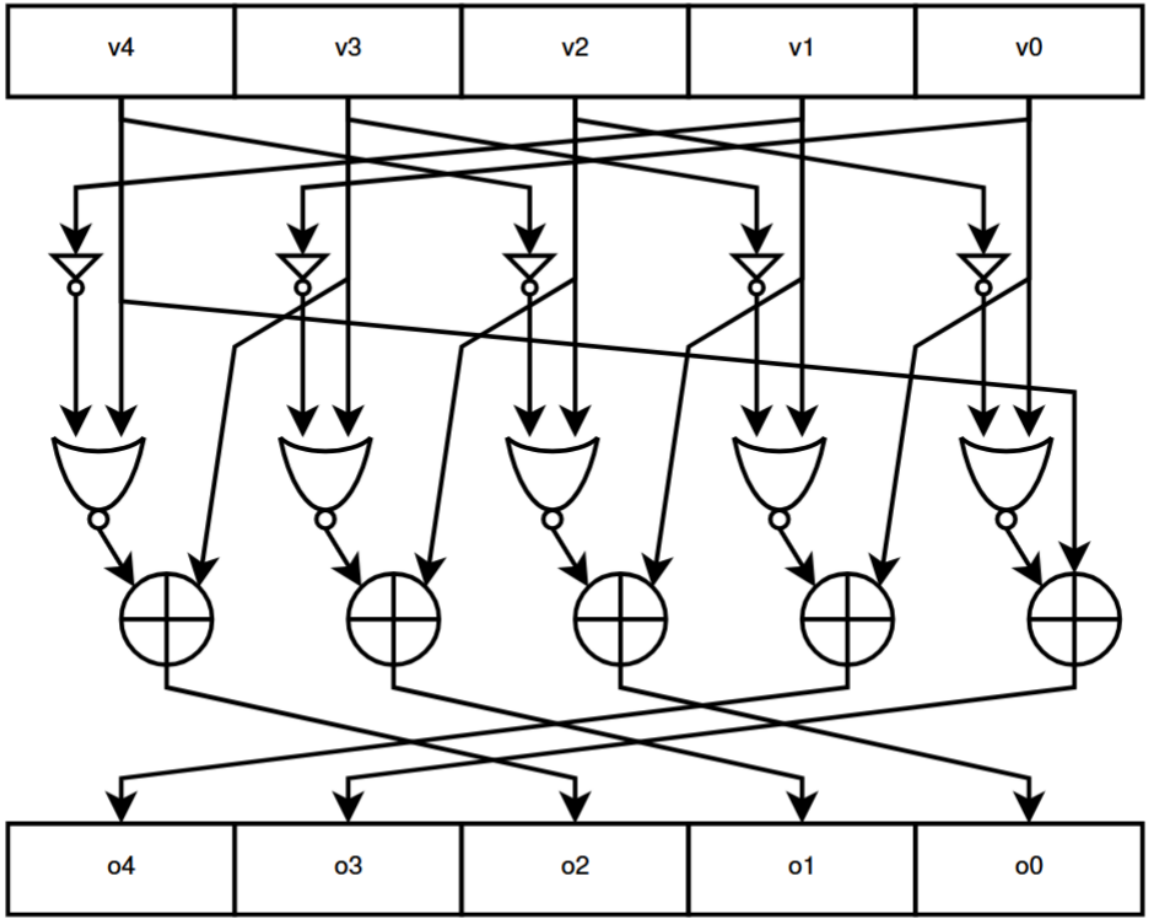
S-box size	T_max	GP			N_F	δ_F
		Max	Avg	Std dev		
4×4	16	**16**	16	0	4	4
5×5	42	**42**	41.73	1.01	12	2
6×6	86	**84**	80.47	4.72	24	4
7×7	182	**182**	155.07	8.86	56	2
8×8	364	318	281.87	13.86	82	20

RESULTS – IMPLEMENTATION PROPERTIES

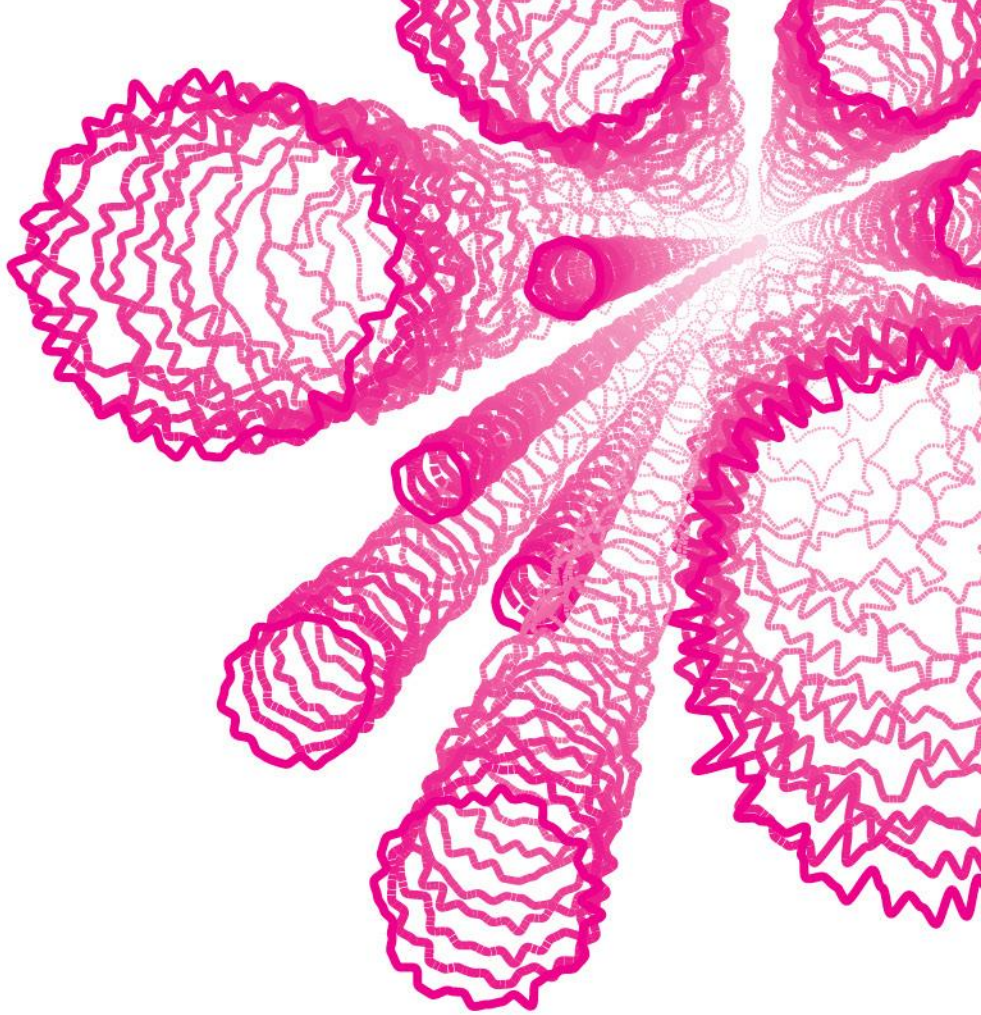
- GE measure of CA-based S-boxes on par with Keccak S-box [18]

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 CA-BASED S-BOX EVOLVED BY GP



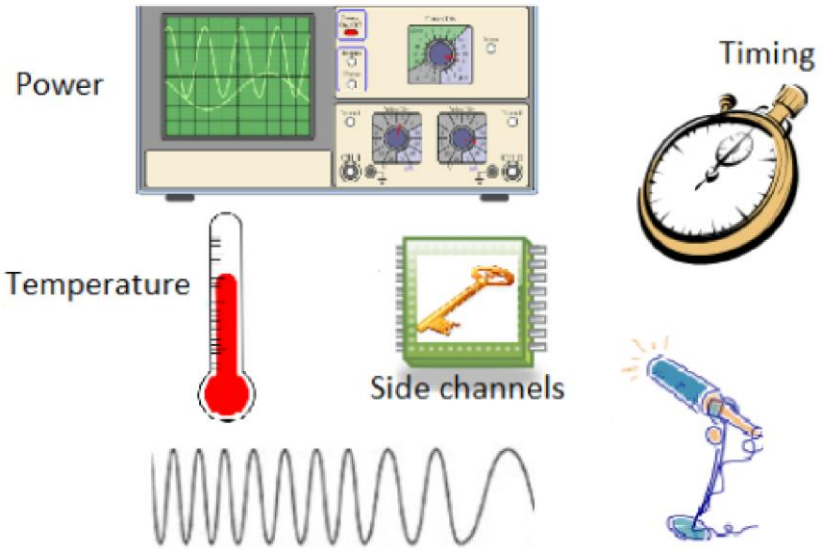
NEUROEVOLUTION FOR SIDE-CHANNEL ANALYSIS

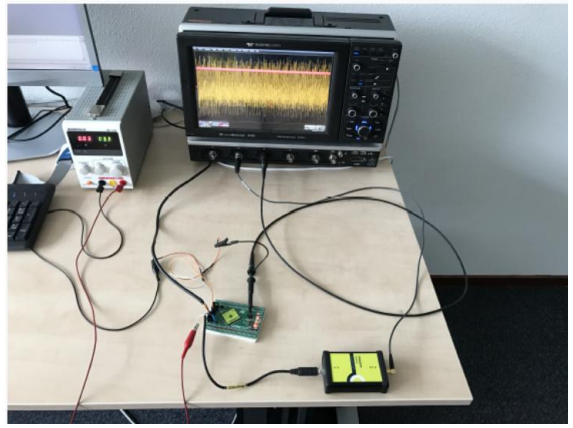


UNIVERSITY
OF TWENTE.

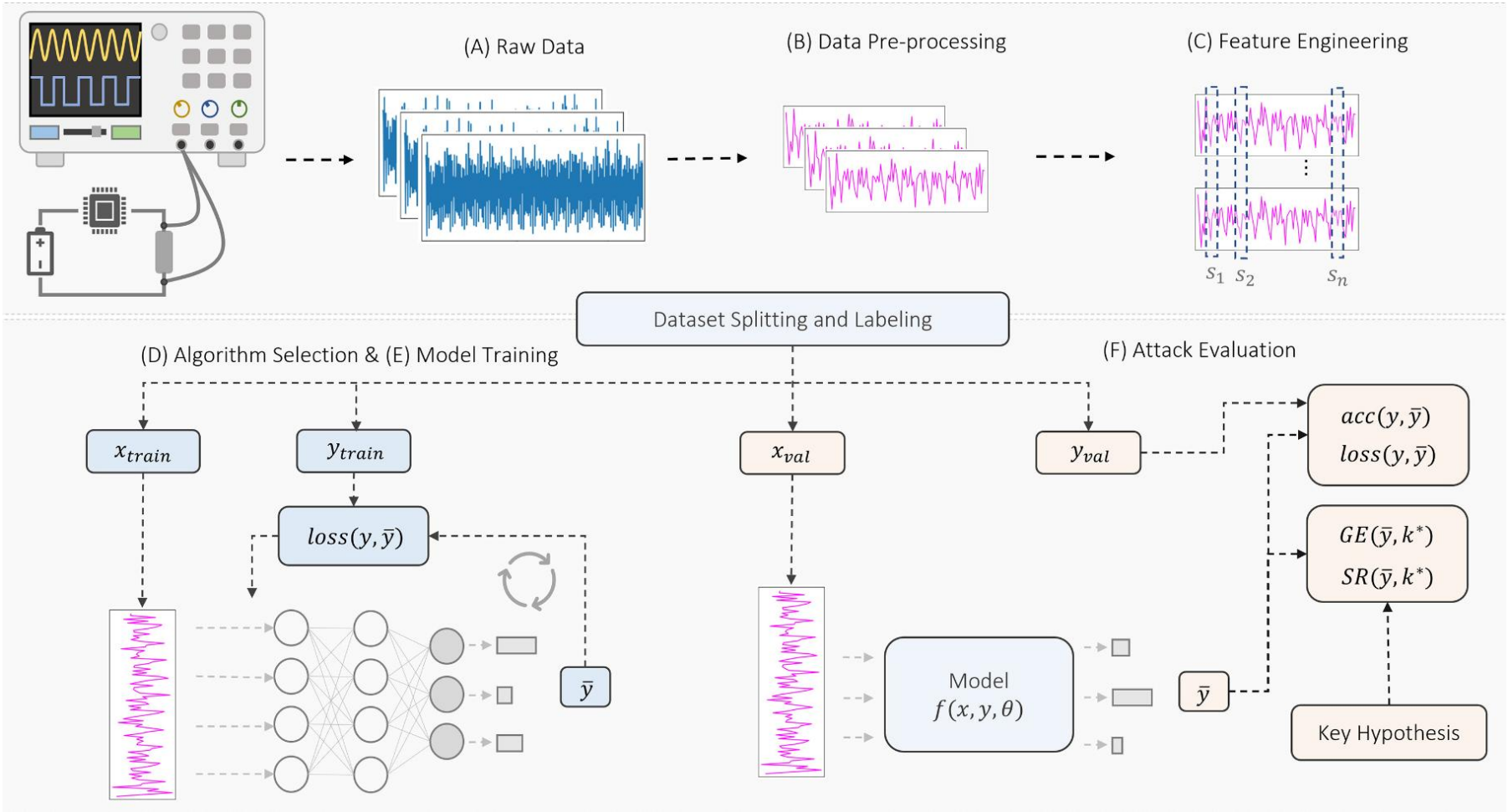
SIDE-CHANNEL ANALYSIS (SCA)

- **Idea:** Attack the *physical implementation* of a cipher, instead of its *mathematical structure*
- Exploits various physical leakages (timing delays, EM emanations, power consumption, …)
- **Profiling SCA:** the adversary estimates the leakage on a clone device, then exploited to extract secret information in the attack phase
- **Deep Learning-based SCA (DL-SCA):** use a DL model to estimate the leakage [16]





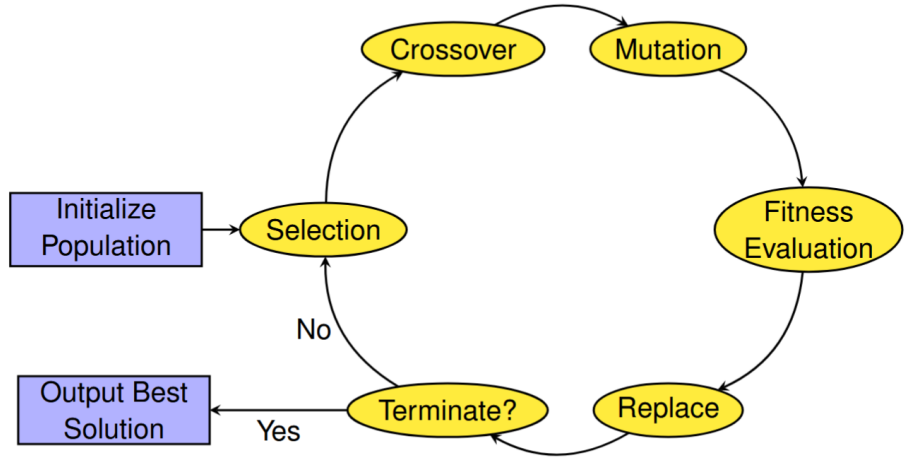
SIDE-CHANNEL ATTACKS

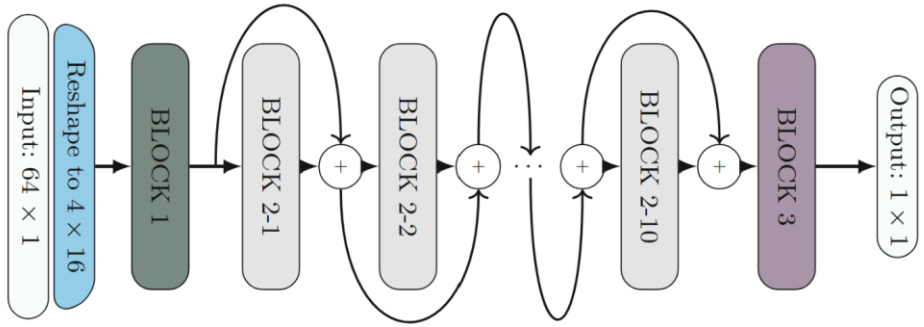


DL-SCA pipeline [16]

NEUROEVOLUTION

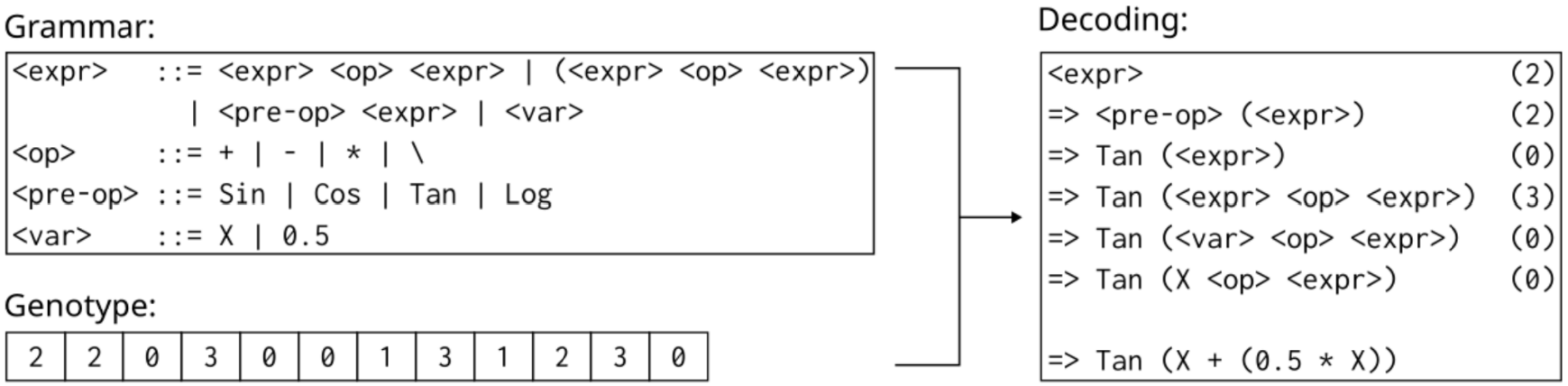
- Most of the models used in DL-SCA (CNN, ResNets, etc.) have been manually designed
- **Question:** is it possible to *automate* the design of neural networks for DL-SCA?
- **Neuroevolution:** employ EA to optimize the *architecture* of a neural network [NE]
- Many different approaches (NEAT [1], Grammatical Evolution [15], GA [19], ...)





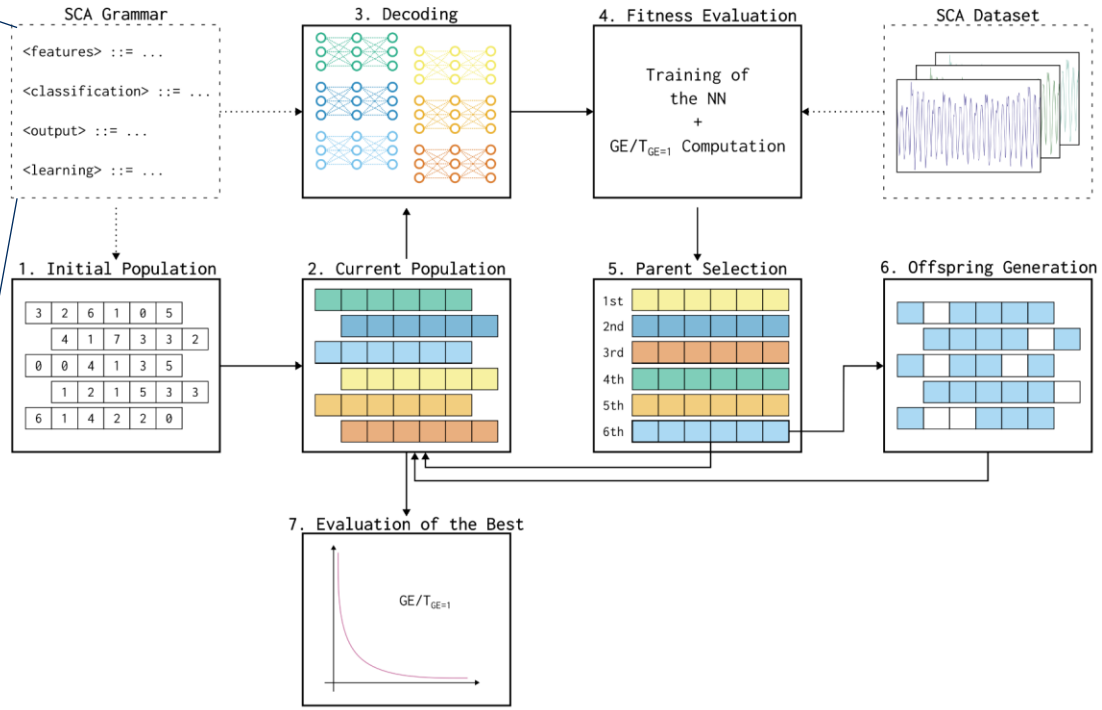
GRAMMATICAL EVOLUTION

- **Genotype:** string of integer numbers indexing the production rules
- **Evaluation:** apply production rules from left to right (with recursion if needed)
- Evolution works by evolving the genotype as a string of integers



SCA DRIVEN BY GRAMMATICAL EVOLUTION (SCAGE)

```
<features> ::= <convolution>
             | <pooling>
             | <batch-norm>
<convolution> ::= layer:conv1d [num-filters,int,1,4,512]
    ↪ [filter-shape,int,1,2,80] [stride,int,1,30,50]
    ↪ <activation-function>
<pooling> ::= <pool-type> [kernel-size,int,1,2,20] [stride,int,1,2,20]
<pool-type> ::= layer:pool-avg1d | layer:pool-max1d
<batch-norm> ::= layer:batch-norm
<classification> ::= <fully-connected> | <dropout>
<fully-connected> ::= layer:fc <activation-function> <regularizer>
    ↪ [num-units,int,1,10,1000]
<regularizer> ::= <regularizer-type> [regrate,float,1,0.00001,0.05]
<regularizer-type> ::= reg:l1 | reg:l2 | reg:none
<dropout> ::= layer:dropout [rate,float,1,0.05,0.5]
<activation-function> ::= act:relu | act:selu
<output> ::= layer:fc act:softmax num-units:256 reg:none
<learning> ::= <optimizer> [lr,float,1,0.0001,0.001] <early-stop>
    ↪ [batch_size,int,1,100,1000] epochs:100
<optimizer> ::= learning:adam | learning:rmsprop
<early-stop> ::= [early_stop,int,1,5,20]
```



SCAGE iteration diagram and grammar [15]

UNIVERSITY OF TWENTE.

SCAGE RESULTS ON ASCAD DATASET

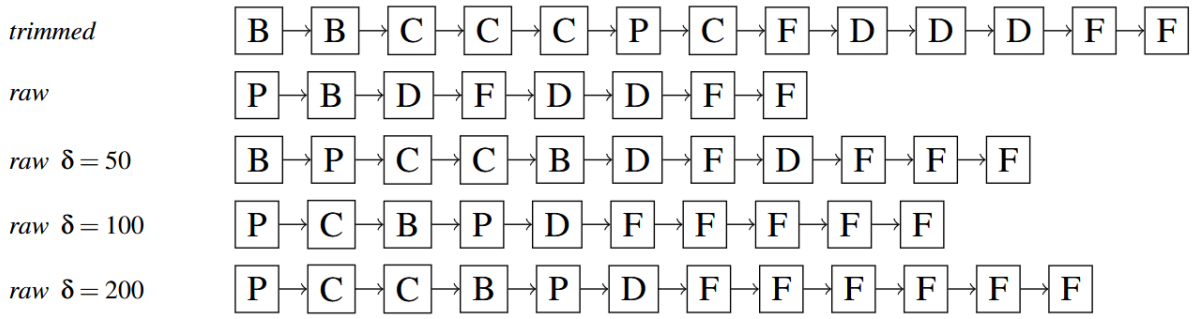
Dataset	Average $T_{GE=1}$		Search Success Rate	
	SCAGE	RS	SCAGE	RS
trimmed	115	792	100%	20%
raw $\delta_{max} = 0$	25	648.33	70%	1.2%
raw $\delta_{max} = 50$	168	x	16.6%	0.0%
raw $\delta_{max} = 100$	4	x	3.3%	0.0%
raw $\delta_{max} = 200$	58	x	6.6%	0.0%

SCAGE vs. Random Search

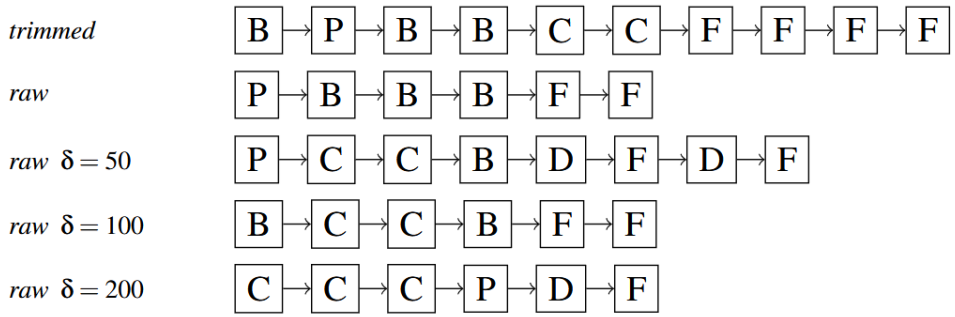
Dataset	$T_{GE=1}$					
	SCAGE	RS	[26]	[12]	[2]	[20]
trimmed	23	85	78	–	120	–
raw $\delta_{max} = 0$	1	1	1	5	–	–
raw $\delta_{max} = 50$	1	2	–	–	–	33
raw $\delta_{max} = 100$	1	x	73	5	–	251
raw $\delta_{max} = 200$	1	x	–	4	–	44

SCAGE vs. SoTA in DL-SCA

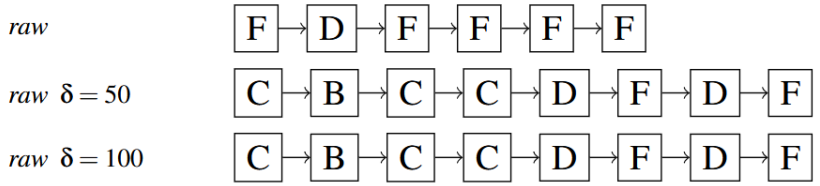
SCAGE – EVOLVED NEURAL NETWORKS



(a) ASCADf NN architectures



(b) ASCADr NN architectures

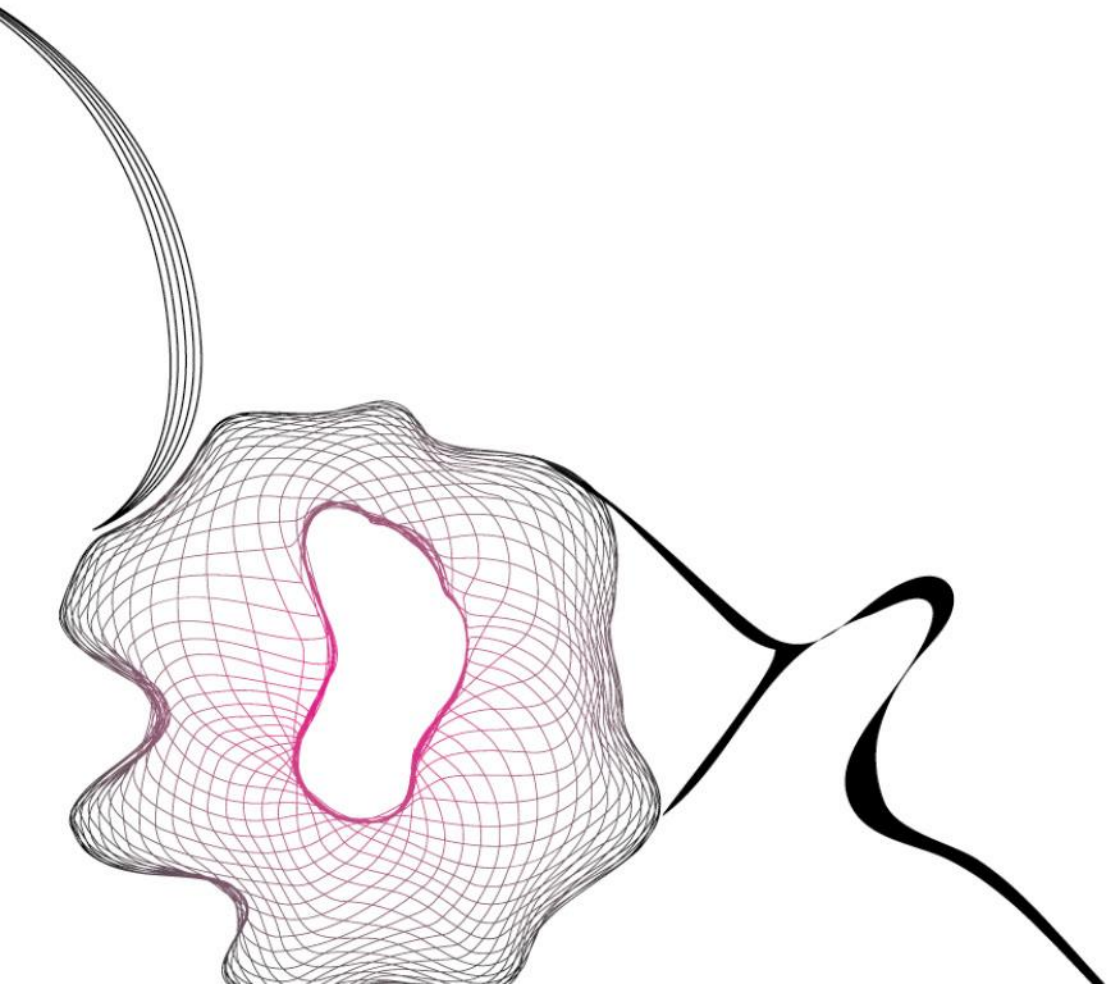


(c) CHES CTF 2018 NN architectures

- Architectures of some of the best neural networks discovered by SCAGE in various scenarios [15]
- C stands for convolutional layer, P for pooling, B for batch normalization, F for fully connected, and D for dropout.

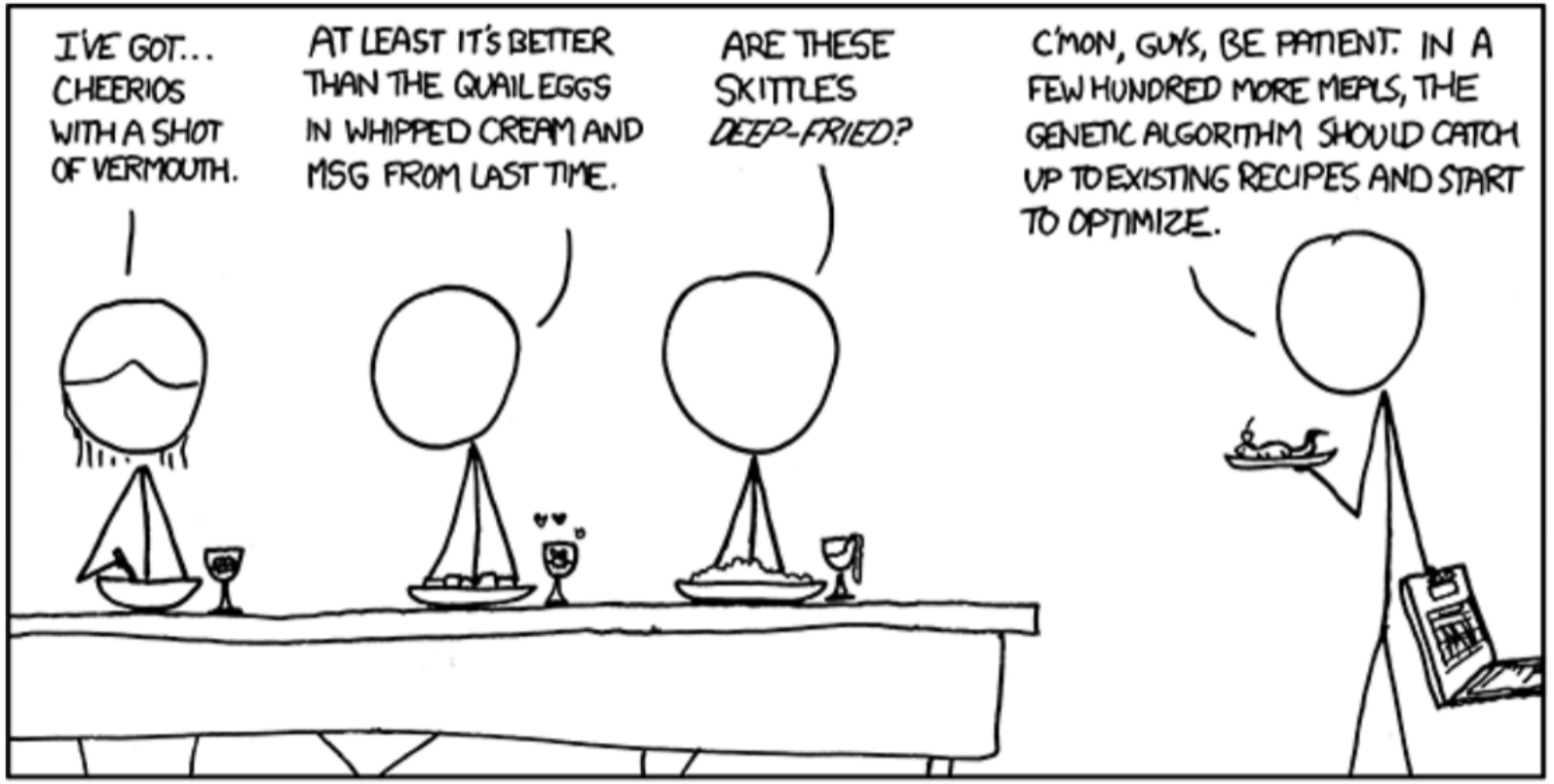
UNIVERSITY OF TWENTE.

SUMMARY



- AI-driven design in symmetric cryptography
- EA for Boolean Functions
- EA for S-boxes
- Neuroevolution for Side-Channel Analysis

THANKS!
QUESTIONS?



REFERENCES

1. R. Y. Acharya, F. Ganji, D. Forte: InfoNEAT: Information Theory-based NeuroEvolution of Augmenting Topologies for Side-channel Analysis. CoRR abs/2105.00117 (2021)
2. G. Bertoni, J. Daemen, M. Peeters, G. Van Assche: The Keccak reference (2011)
3. C. Carlet, M. Djurasevic, D. Jakobovic, L. Mariot, S. Picek: Evolving constructions for balanced, highly nonlinear boolean functions. Proc. of GECCO 2022, pp. 1147–1155 (2022)
4. C. Carlet: Boolean functions for cryptography and coding theory. Cambridge University Press (2021)
5. J. Daemen, R. Govaerts, J. Vandewalle: Invertible shift-invariant transformations on binary arrays. Appl. Math. Comput. 62:259–277 (1994)
6. M. Djurasevic, D. Jakobovic, L. Mariot, S. Picek: A survey of metaheuristic algorithms for the design of cryptographic Boolean functions. Cryptogr. Commun. 15(6): 1171-1197 (2023)
7. J. Katz, Y. Lindell: Introduction to Modern Cryptography. Third Edition, CRC Press (2021)
8. J. R. Koza, M. A. Keane, J. P. Rice: Performance improvement of machine learning via automatic discovery of facilitating functions as applied to a problem of symbolic system identification. Proc. of ICNN 1993, pp. 191–198 (1993)
9. L. Manzoni, L. Mariot, E. Tuba: Balanced crossover operators in Genetic Algorithms. Swarm Evol. Comput. 54: 100646 (2020)
10. L. Mariot, M. Saletta, A. Leporati, L. Manzoni: Heuristic search of (semi-)bent functions based on cellular automata. Nat. Comput. 21(3): 377-391 (2022)
11. L. Mariot, D. Jakobovic, T. Bäck, J.C. Hernandez-Castro: Artificial Intelligence for the Design of Symmetric Cryptographic Primitives. Security and Artificial Intelligence, pp. 3-24 (2022)
12. L. Mariot, D. Jakobovic, A. Leporati, S. Picek: Hyper-bent Boolean Functions and Evolutionary Algorithms. Proc. of EuroGP 2019, pp. 262-277 (2019)
13. L. Mariot, S. Picek, A. Leporati, D. Jakobovic: Cellular automata based S-boxes. Cryptogr. Commun. 11(1): 41-62 (2019)
14. W. Millan, J. Clark, E. Dawson: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. Proc. of EUROCRYPT 1998, pp. 489–499 (1998)
15. M. Napoli, A. Leporati, S. Picek, L. Mariot: Mind the Grammar: Side-Channel Analysis driven by Grammatical Evolution. IACR Cryptol. ePrint Arch. 2025: 698 (2025)
16. S. Picek, G. Perin, L. Mariot, L. Wu, L. Batina: SoK: Deep Learning-based Physical Side-channel Analysis. ACM Comput. Surv. 55(11): 227:1-227:35 (2023)
17. S. Picek, K. Knezevic, L. Mariot, D. Jakobovic, A. Leporati: Evolving Bent Quaternary Functions. Proc. of CEC 2018, pp. 1–8 (2018)
18. S. Picek, L. Mariot, B. Yang, D. Jakobovic, N. Mentens: Design of S-boxes Defined with Cellular Automata Rules. Proc. of CF 2017, pp. 409-414 (2017)
19. S. Picek, D. Jakobovic, J. F. Miller, L. Batina, M. Cupic: Cryptographic Boolean functions: One output, many design criteria. Appl. Soft Comput. 40: 635-653 (2016)
20. S. Picek, D. Jakobovic: Evolving Algebraic Constructions for Designing Bent Boolean Functions. Proc. of GECCO 2016, pp. 781-788 (2016)
21. F. Schijlen, L. Wu, L. Mariot: NASCTY: Neuroevolution to Attack Side-channel Leakages Yielding Convolutional Neural Networks. Mathematics 11(12): 2616 (2024)

UNIVERSITY OF TWENTE.