EEMCS - SCS

NATURE-INSPIRED METAHEURISTICS FOR THE DESIGN OF CRYPTOGRAPHIC PRIMITIVES

LECTURE SERIES ON APPLICATIONS AND THEORY OF NATURE-INSPIRED INTELLIGENT COMPUTING (ATONIIC)

10 APRIL 2025

LUCA MARIOT

UNIVERSITY OF TWENTE.

SUMMARY

- Basics on Symmetric Cryptography
- EA for Boolean Functions
- EA for S-boxes

BASICS ON SYMMETRIC CRYPTOGRAPHY

UNIVERSITY
OF TWENTE.

THE CIA TRIAD IN CRYPTOGRAPHY

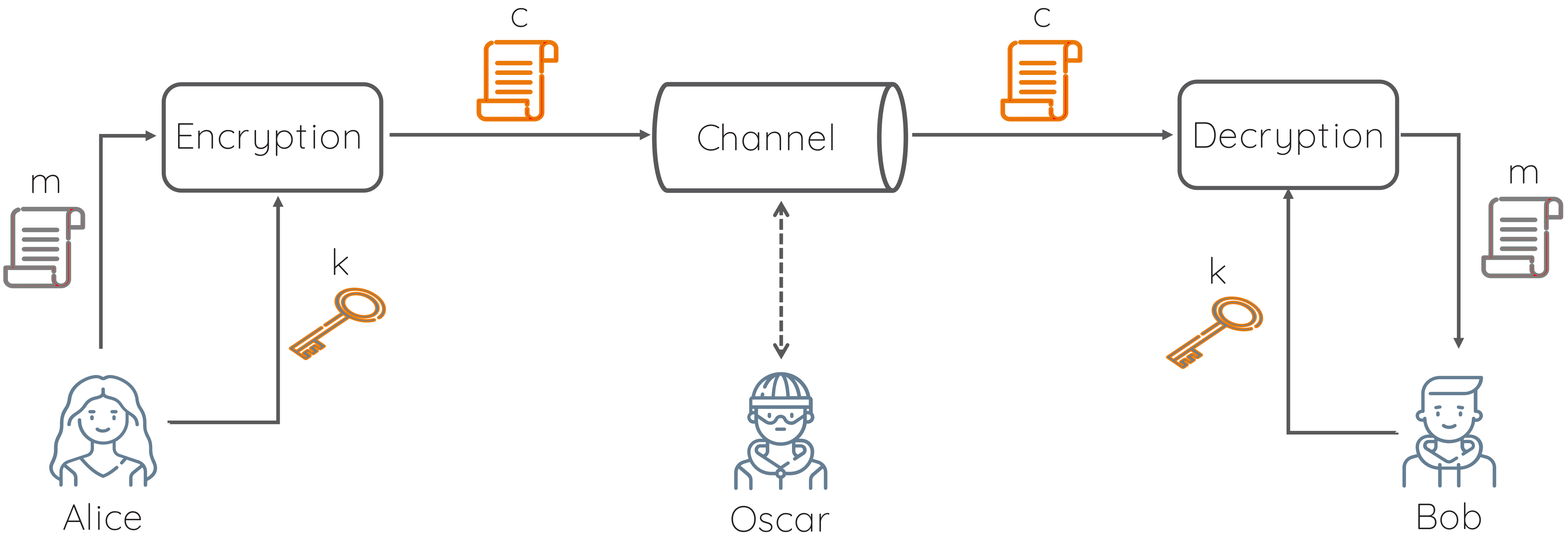
In cryptography, we usually aim to achieve three goals:

Confidentiality	Integrity	Authenticity
Data are protected from unauthorized access/use	Data have not been altered by unauthorized parties	Data have been created by the intended party

Confidentiality: Encryption schemes

Integrity, Authenticity: Authentication codes, Digital signatures

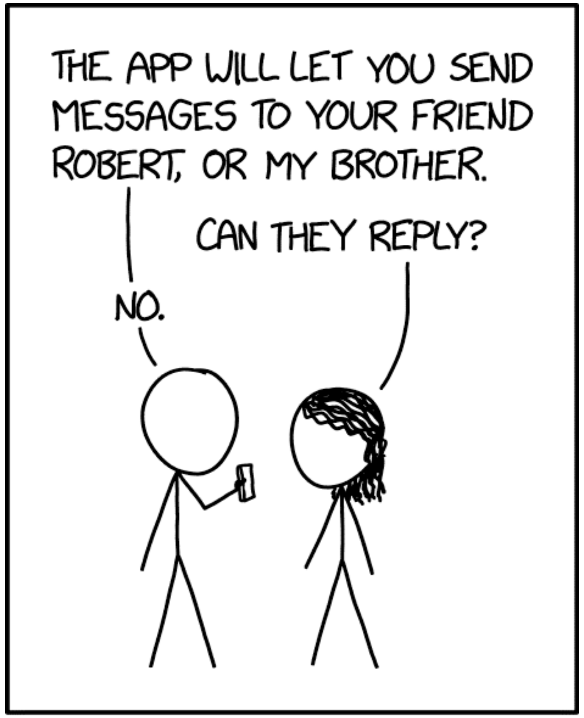
SYMMETRIC-KEY ENCRYPTION SCENARIO



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

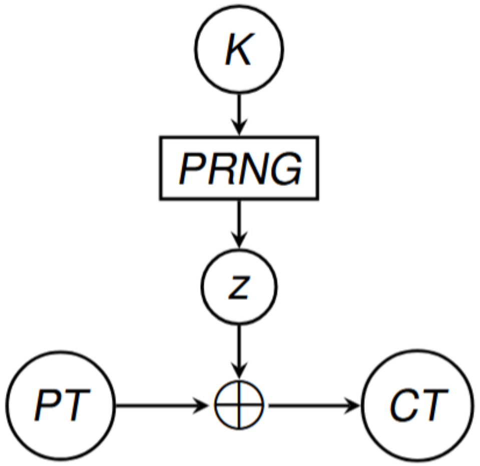
SYMMETRIC-KEY ENCRYPTION SCENARIO

- The same key k is used both for encryption *and* decryption [8]
- The scheme is "secure" as long as Oscar does not know k
- Requires sharing k *before* communicating
- Here, we just assume Alice and Bob shared k somehow

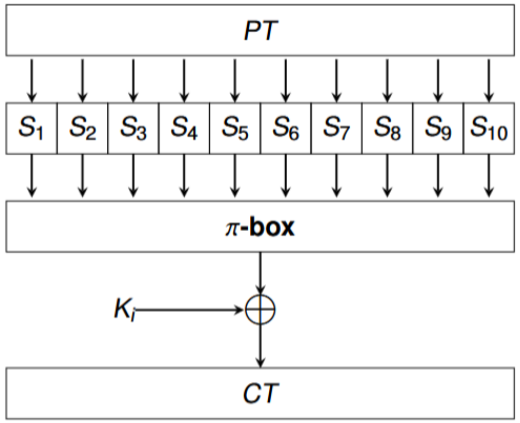


MY NEW SECURE TEXTING APP ONLY ALLOWS PEOPLE NAMED ALICE TO SEND MESSAGES TO PEOPLE NAMED BOB.

PRIMITIVES IN SYMMETRIC CRYPTOGRAPHY



(a) Stream cipher

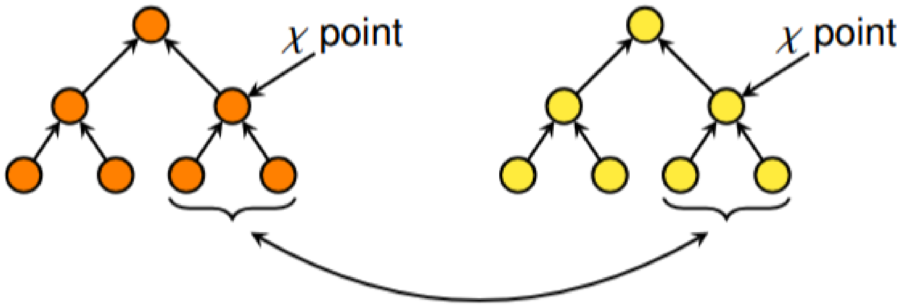


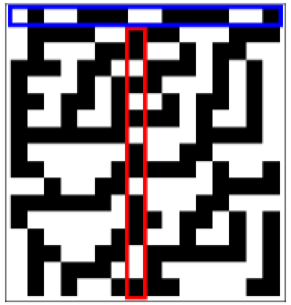
(b) Block cipher

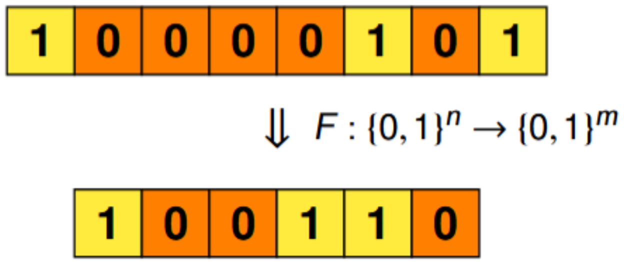
- Symmetric ciphers require several low-level primitives:
  - Pseudorandom number generators (PRNG)
  - Boolean functions $f : \{0,1\}^n \to \{0,1\}$
  - S-boxes $F : \{0,1\}^n \to \{0,1\}^n$
  - Diffusion layers, …
- Classic methods to build them: algebraic constructions [3]

UNIVERSITY OF TWENTE.

AI-DRIVEN DESIGN OF CRYPTO PRIMITIVES







- Algebraic constructions are not flexible
- AI-driven approach [13]: support the cipher designer in designing the primitives with:
  - Metaheuristic optimization techniques (Evolutionary Algorithms, …)
  - Nature-inspired computational models (Cellular Automata, …)

UNIVERSITY OF TWENTE.

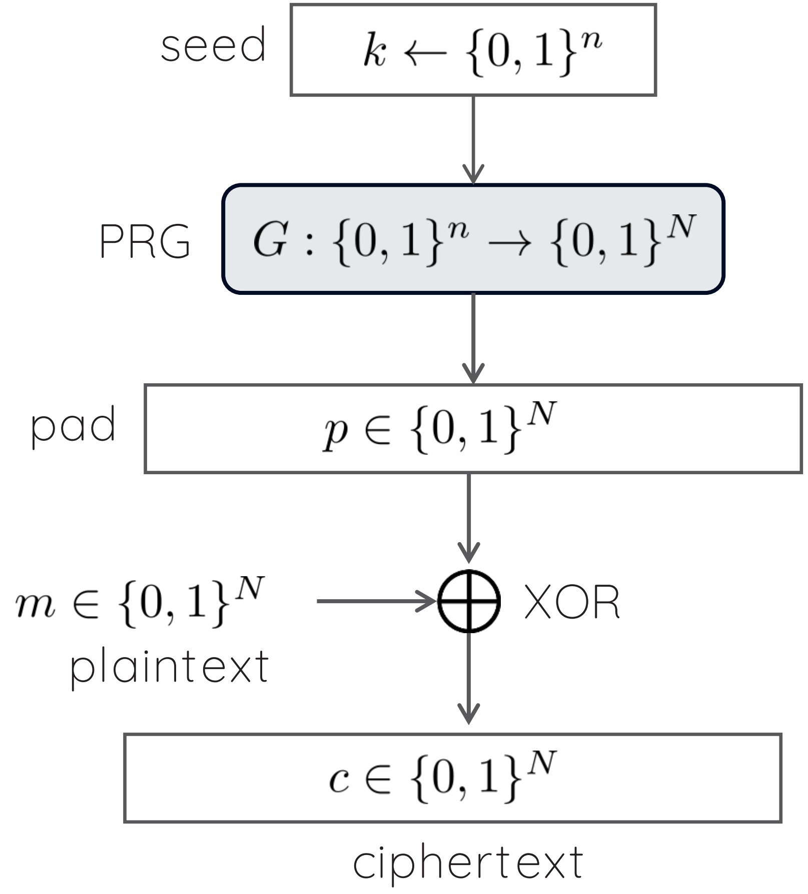
EVOLUTIONARY ALGORITHMS FOR BOOLEAN FUNCTIONS

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 [8]
- The seed of the PRG is the key $k \leftarrow \{0,1\}^n$
- Encryption: Bitwise XOR between message and pad

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

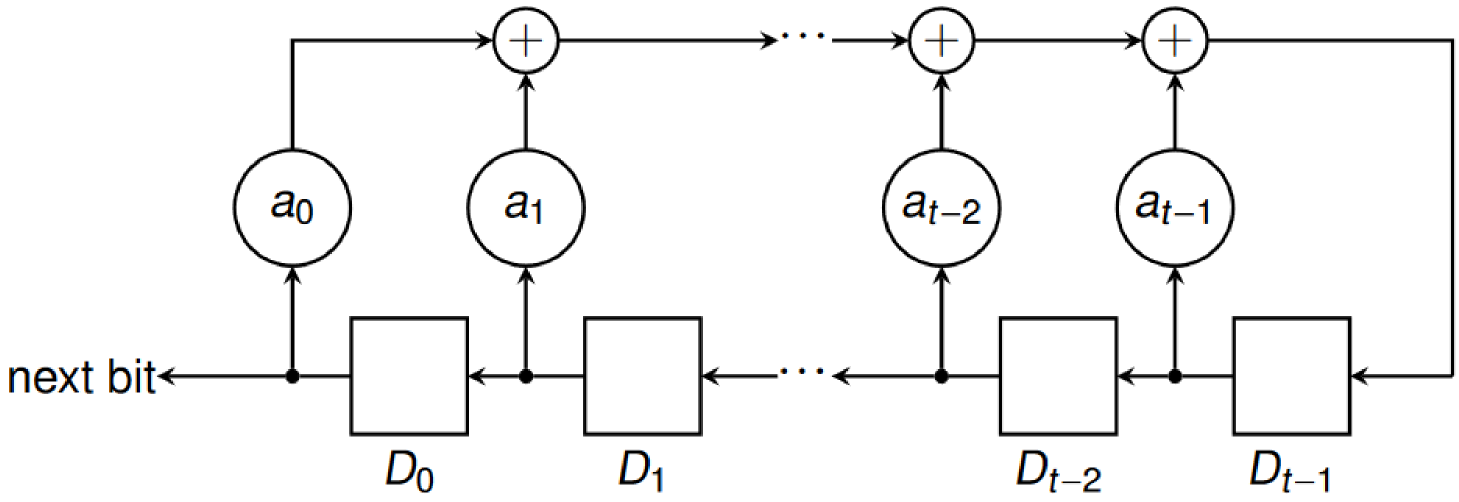


UNIVERSITY OF TWENTE.

LINEAR FEEDBACK SHIFT REGISTERS (LFSR)

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

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



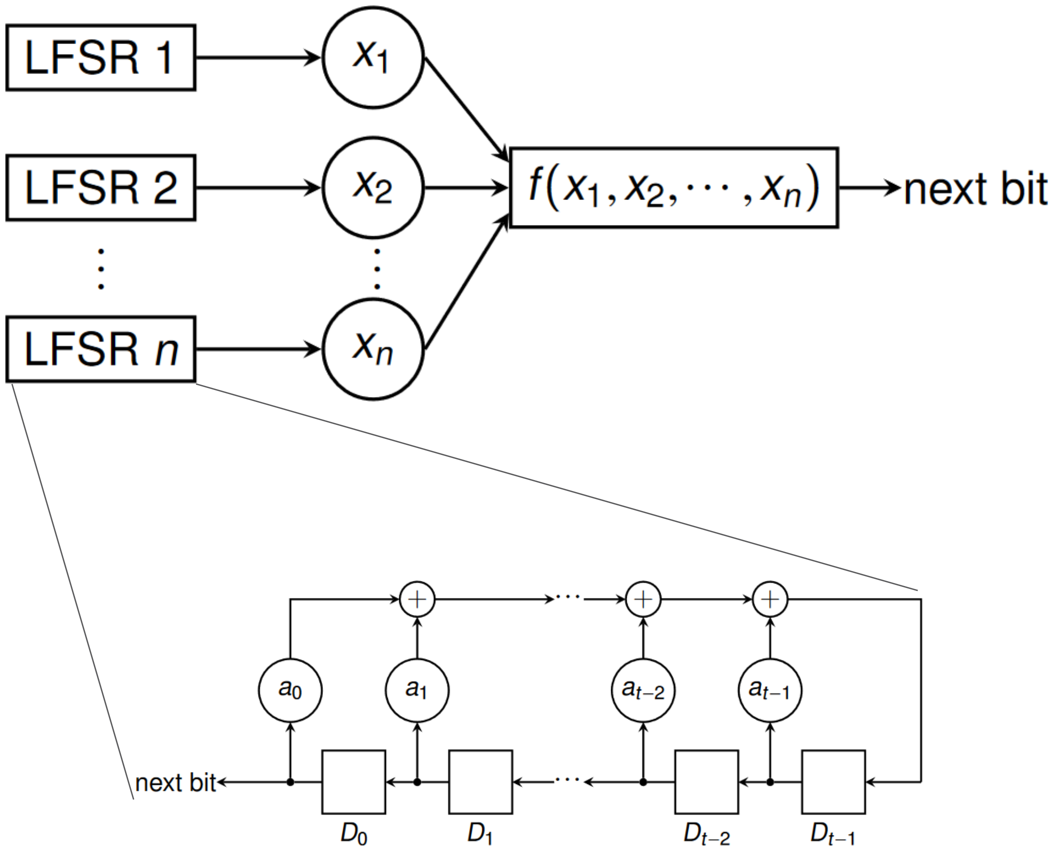
- Problem: very weak as a cryptographic PRG [6]

IMPROVING LFSR – COMBINER MODEL FOR PRG

- Idea: use n LFSRs instead of one [3]
- LFSRs outputs *combined* using a Boolean function:

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

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



UNIVERSITY OF TWENTE.

BOOLEAN FUNCTIONS

- A mapping $f : \{0,1\}^n \rightarrow \{0,1\}$ 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 [3]:
  - Balancedness (equal number of 0s and 1s)
  - High Nonlinearity (Hamming distance from linear functions)
  - High algebraic degree, etc. …

UNIVERSITY OF TWENTE.

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
$f(x_1, x_2, x_3)$	?	?	?	?	?	?	?	?

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

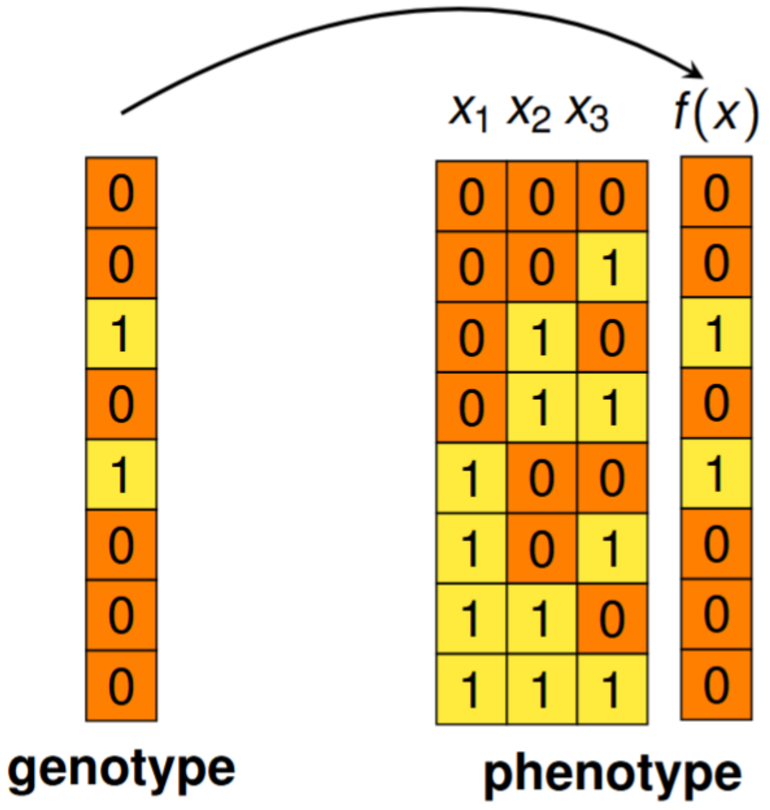
- The truth table has size 2^n so there are 2^{2^n} combinations

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 $n > 5$!

UNIVERSITY OF TWENTE.

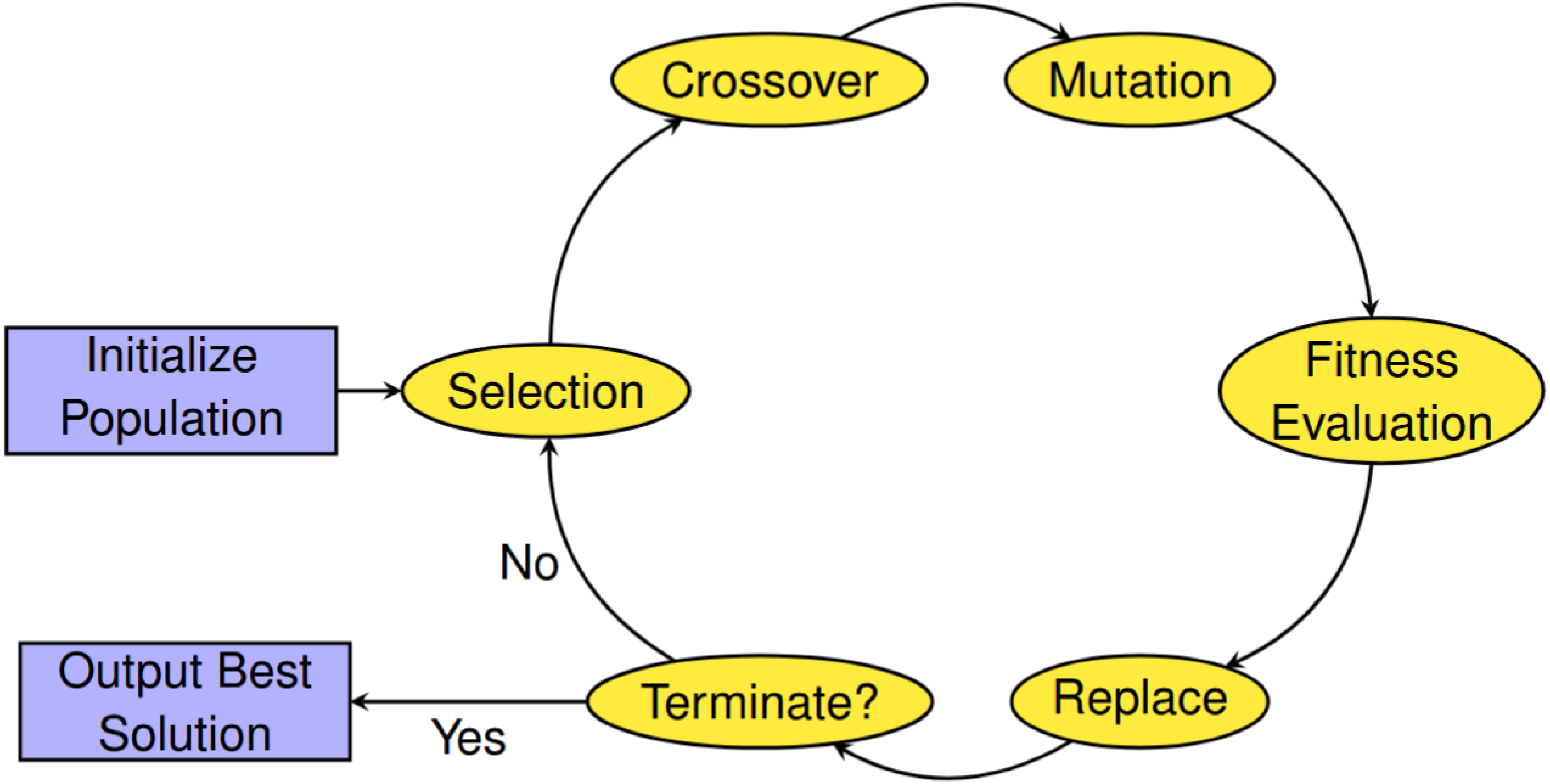
GENETIC ALGORITHMS (GA)



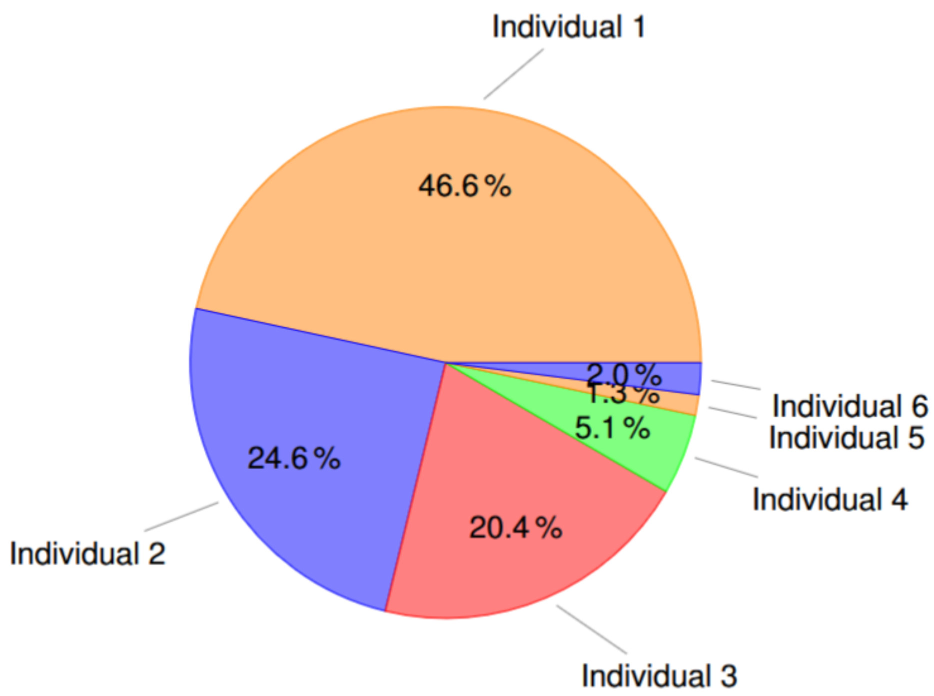
- Introduced by John Holland (1975)
- Optimization algorithms loosely based on evolutionary principles
- GA genotype: fixed-length bitstrings
- phenotype: truth table of f [5]

UNIVERSITY OF TWENTE.

THE EA LOOP


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

SELECTION

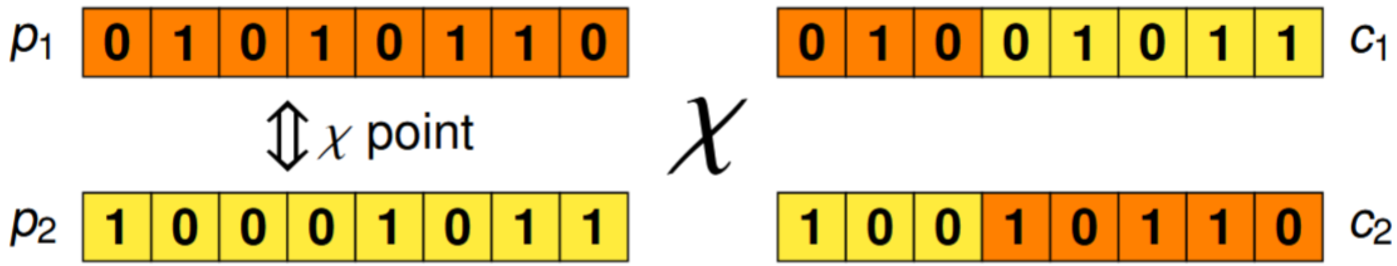


- Roulette-Wheel: selection probability proportional to individual's fitness
- Tournament: select the fittest individual from a random sample of t individuals
- In general: tournaments work better

CROSSOVER

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

GA Example: One-Point Crossover

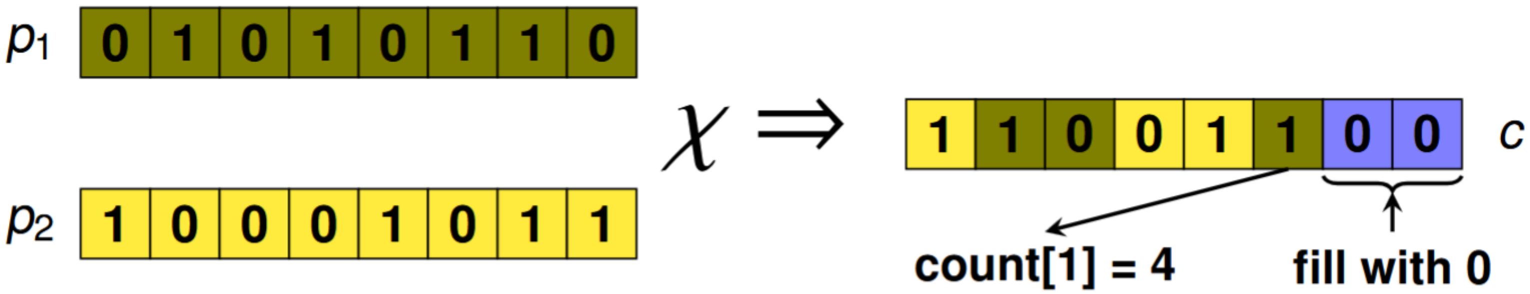


- Problem: how dow we ensure balancedness for cryptography?
- We could optimize it in the fitness function…

BALANCED CROSSOVERS

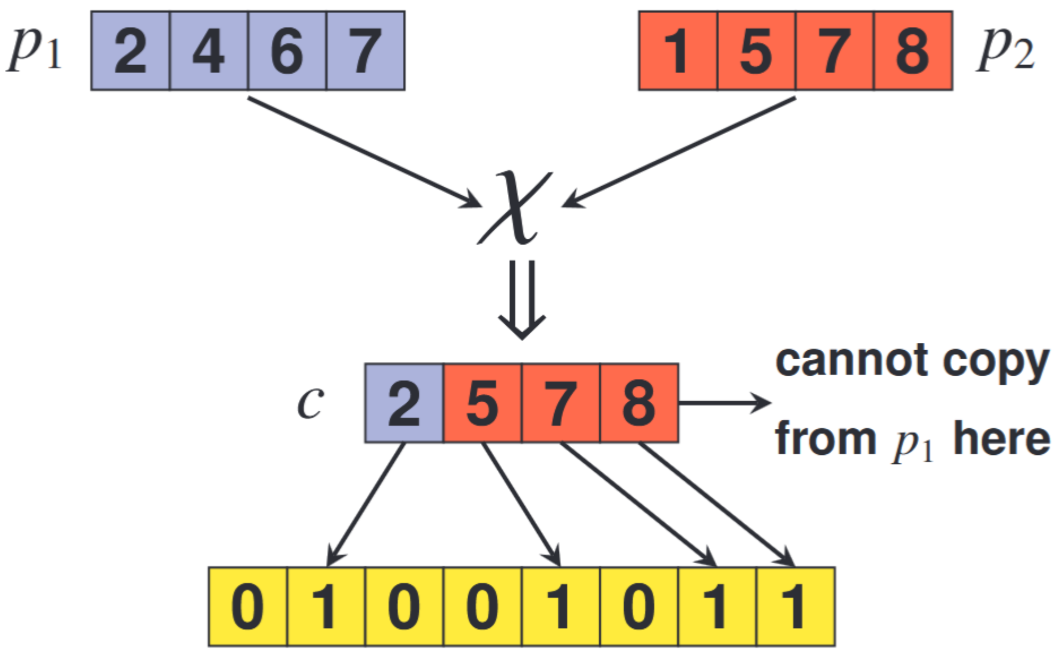
- Idea: use counters to keep track of the numbers of 1s in the child [10, 16]

GA Example: Counter-based Crossover



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

BALANCED CROSSOVERS



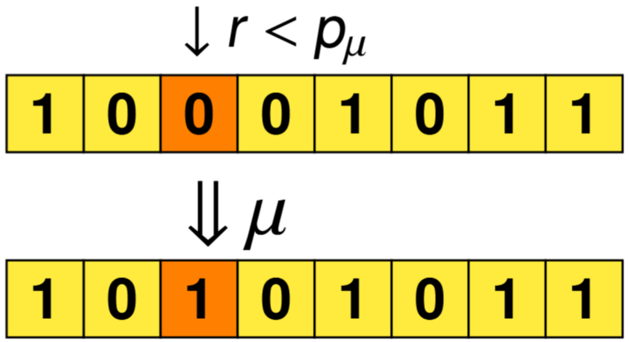
GA Example: Map-of-Ones Crossover

- Other variant: encode the positions of the 1s in the bitstring [10]
- 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)

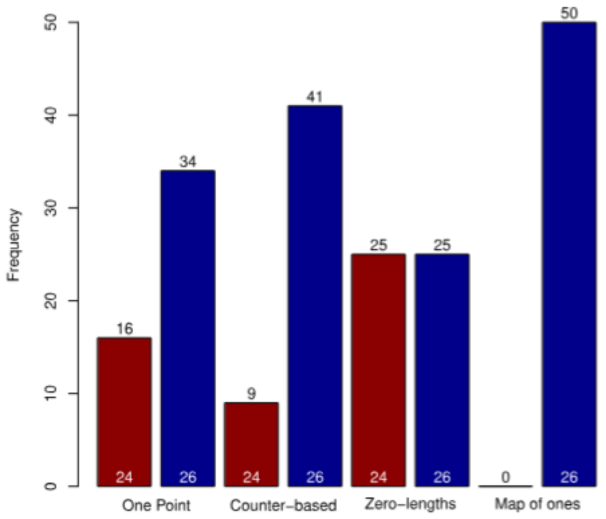
GA Example: Bit-flip mutation



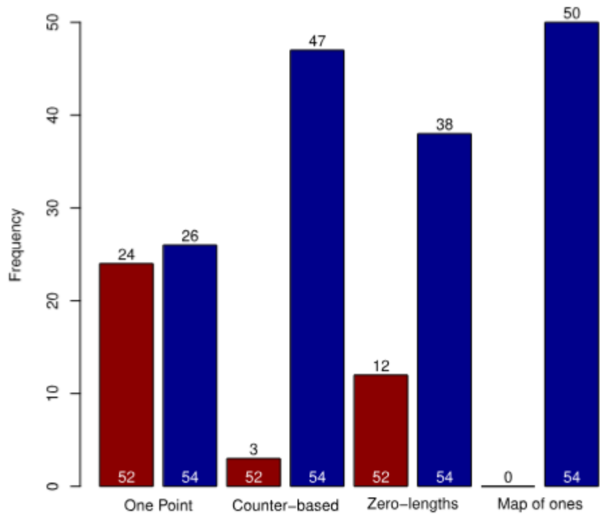
- For balancedness: randomly swap some bits instead of flipping them

EXPERIMENTAL COMPARISON

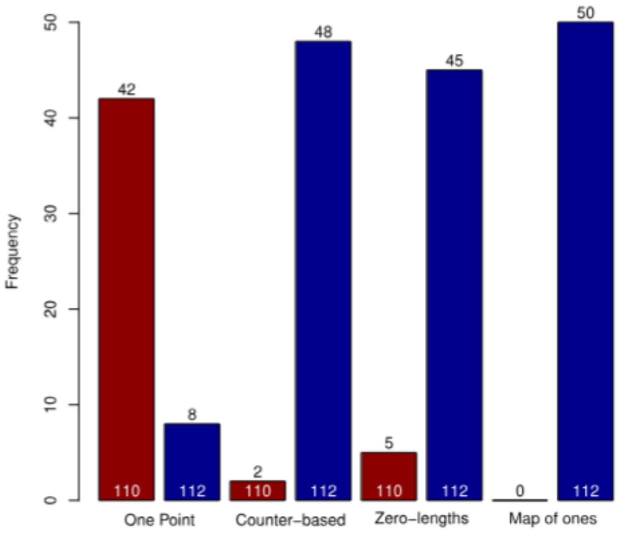
n=6 instance



n=7 instance



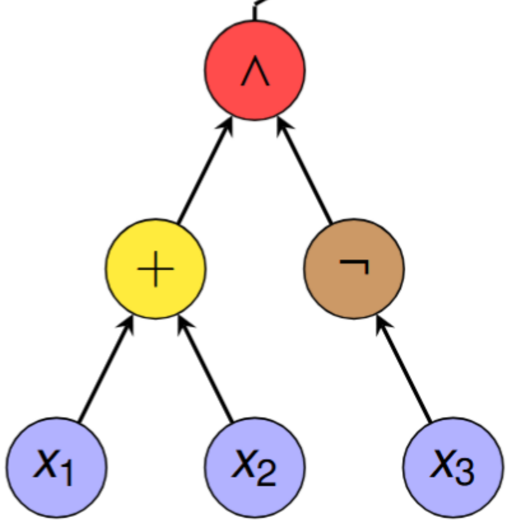
n=8 instance



In general: balanced operators perform better than one-point crossover, Map-of-Ones is the operator with the best success rate [11]

UNIVERSITY OF TWENTE.

GENETIC PROGRAMMING (GP)

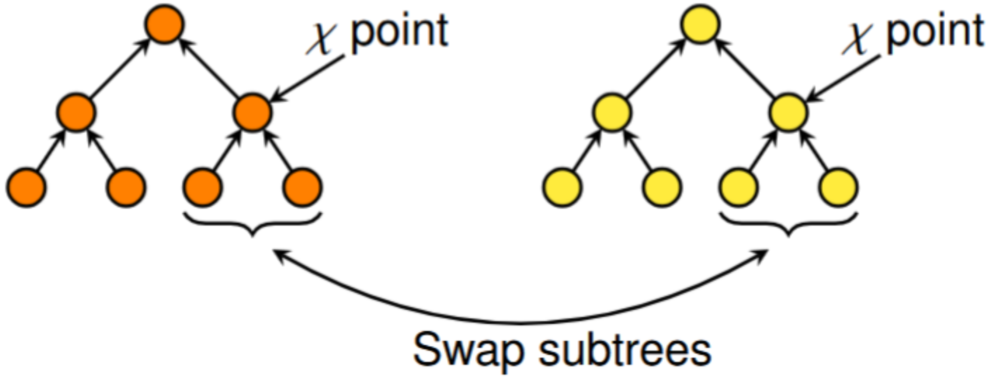


x_1	x_2	x_3	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

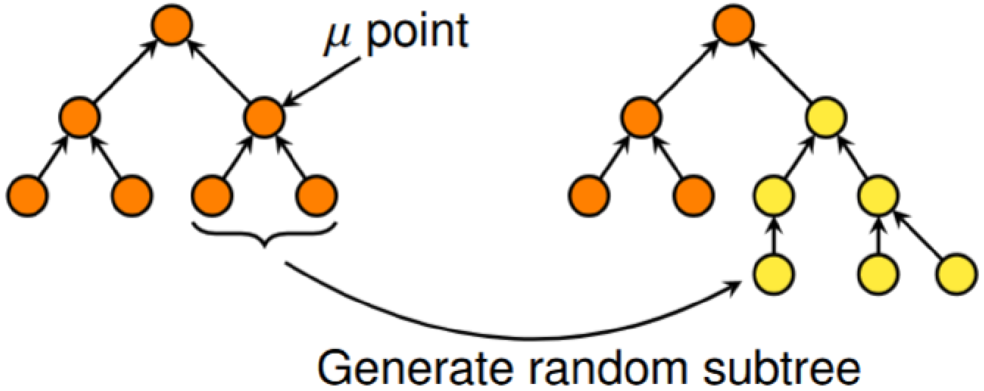
- Idea: evolve computer programs to solve specific tasks [9]
- GP Genotype: a syntactic tree
  - Leaf nodes: input variables
  - Internal nodes: operators (e.g. AND, OR, NOT, XOR, …)
- Phenotype: evaluate the tree for all possible assignments of the leaf nodes

GP CROSSOVER & MUTATION

Subtree Crossover



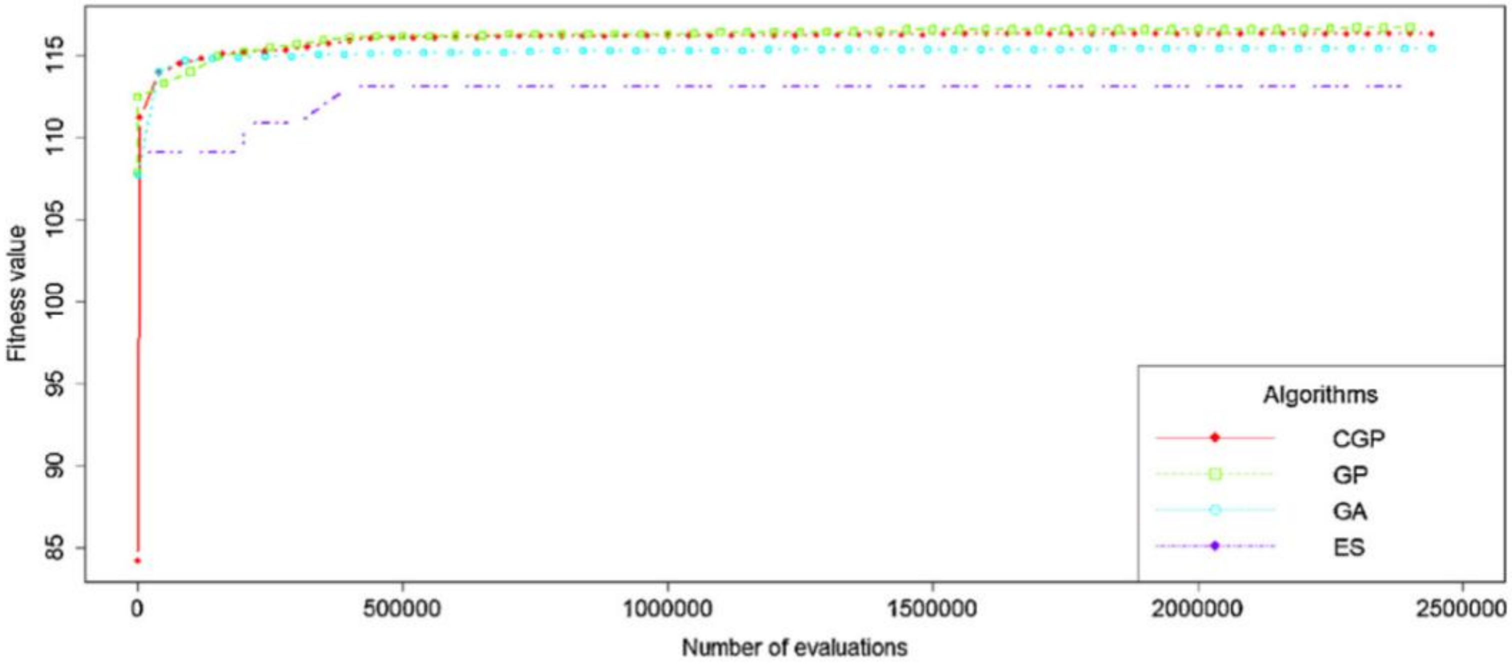
Subtree Mutation



- 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

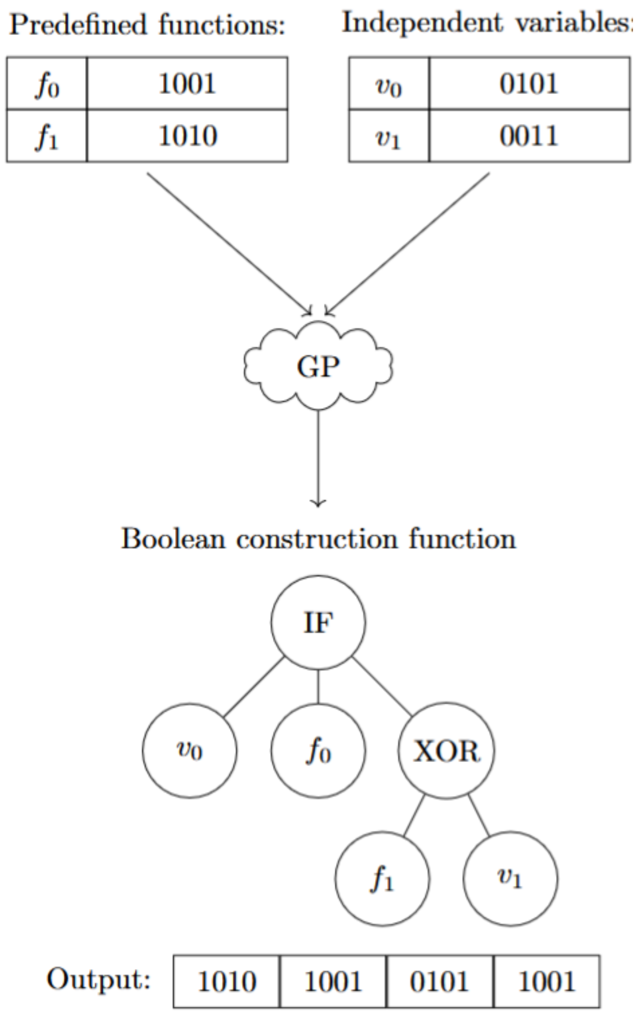
- GA with balanced operators explores a smaller search space than GP



- Nevertheless: GP usually fares better than (balanced) GA [14, 17, 18, 19]

UNIVERSITY OF TWENTE.

EVOLVING ALGEBRAIC CONSTRUCTIONS WITH GP



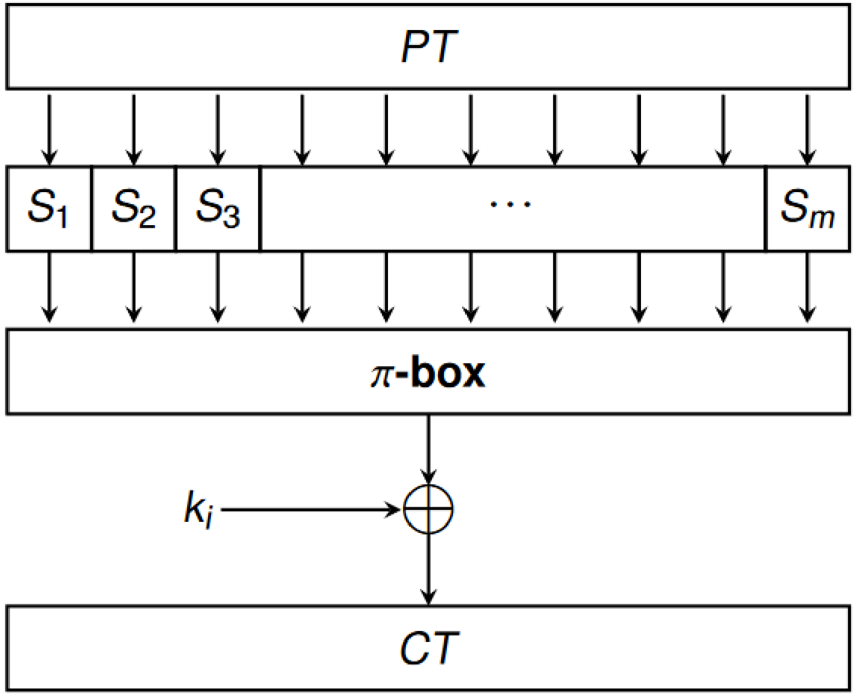
- Idea: Do not evolve functions directly, but rather their algebraic constructions [8]
- Use Boolean minimizers to interpret the constructions
- Research Question: Does GP obtain new constructions?
- Finding: GP always generates the same known construction, but in a very complicated way

EVOLUTIONARY ALGORITHMS FOR S-BOXES

UNIVERSITY
OF TWENTE.

SPN BLOCK CIPHERS

- Round function: composed of confusion layer and diffusion layer [6]
- Confusion: Substitution boxes (S-boxes) of the form $S_i : \{0,1\}^n \to \{0,1\}^n$
- Diffusion: Permutation box (P-box) of the form $\pi : \{0,1\}^b \to \{0,1\}^b$
- Result is bitwise XORed with round key
- All S-boxes and P-boxes are invertible



S-BOXES

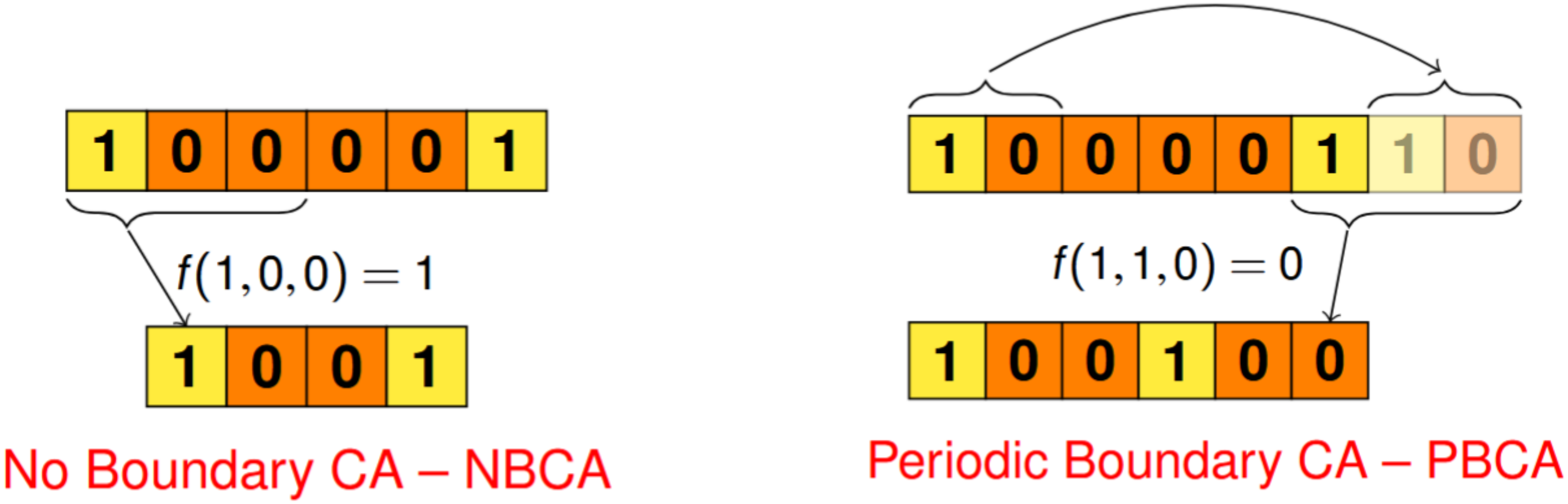
- Vectorial mapping $F : \{0,1\}^n \to \{0,1\}^m$
- Defined by m *coordinate Boolean functions* $f_i : \{0,1\}^n \to \{0,1\}$

(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, …) [3]

CELLULAR AUTOMATA (CA)

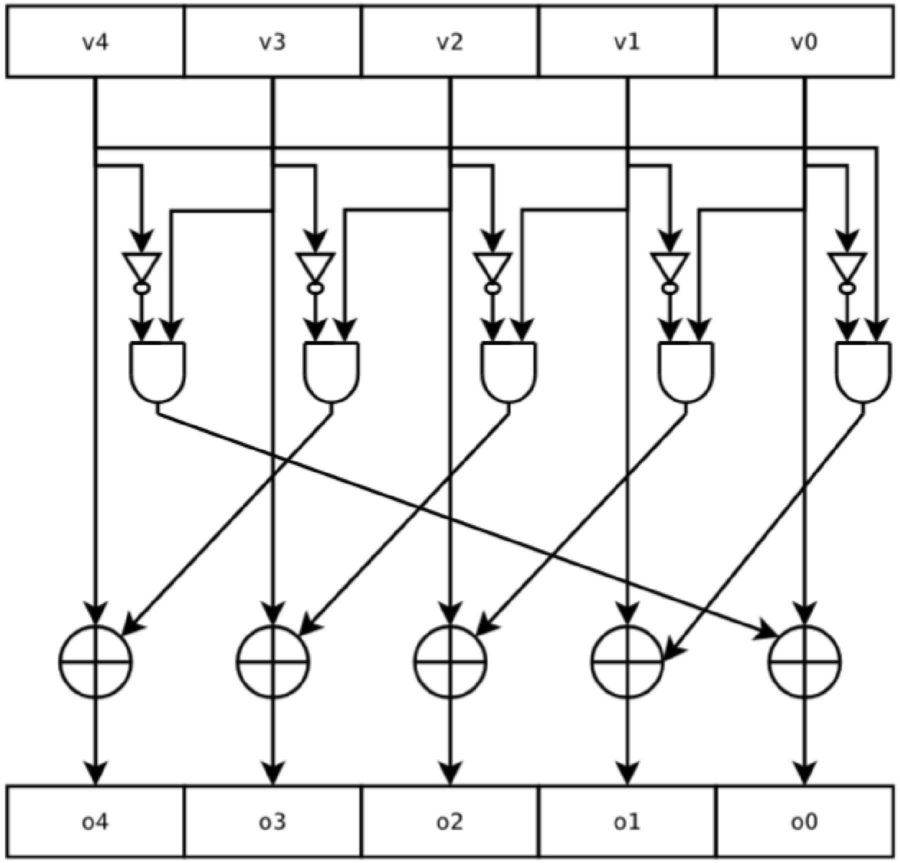
- Discrete parallel computation model composed of a finite array of *cells*



- Each cell updates its state by applying a local rule $f : \{0,1\}^d \to \{0,1\}$ on itself and the d-1 right neighboring cells [15]

UNIVERSITY OF TWENTE.

CA-BASED CRYPTO: KECCAK χ MAPPING

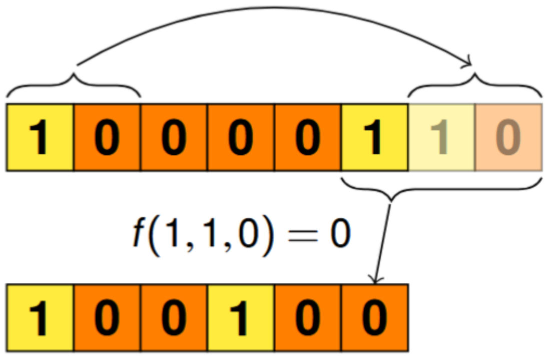


- Local rule of diameter d=3:

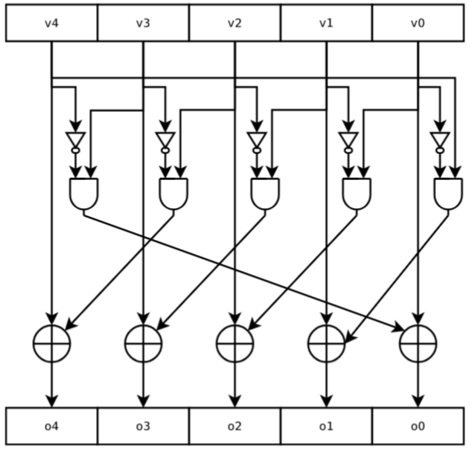
$$\chi(x_1, x_2, x_3) = x_1 \oplus (1 \oplus (x_2 \cdot x_3))$$

- Invertible PBCA for every odd size of the CA array [4]
- Used as a 5x5 S-box in the *Keccak* specification of the SHA-3 standard [1]
- 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 [15]
  - low implementation cost [18]
- GP optimizing both crypto (nonlinearity, differential uniformity) and implementation properties (GE measure)

RESULTS – CRYPTO PROPERTIES

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

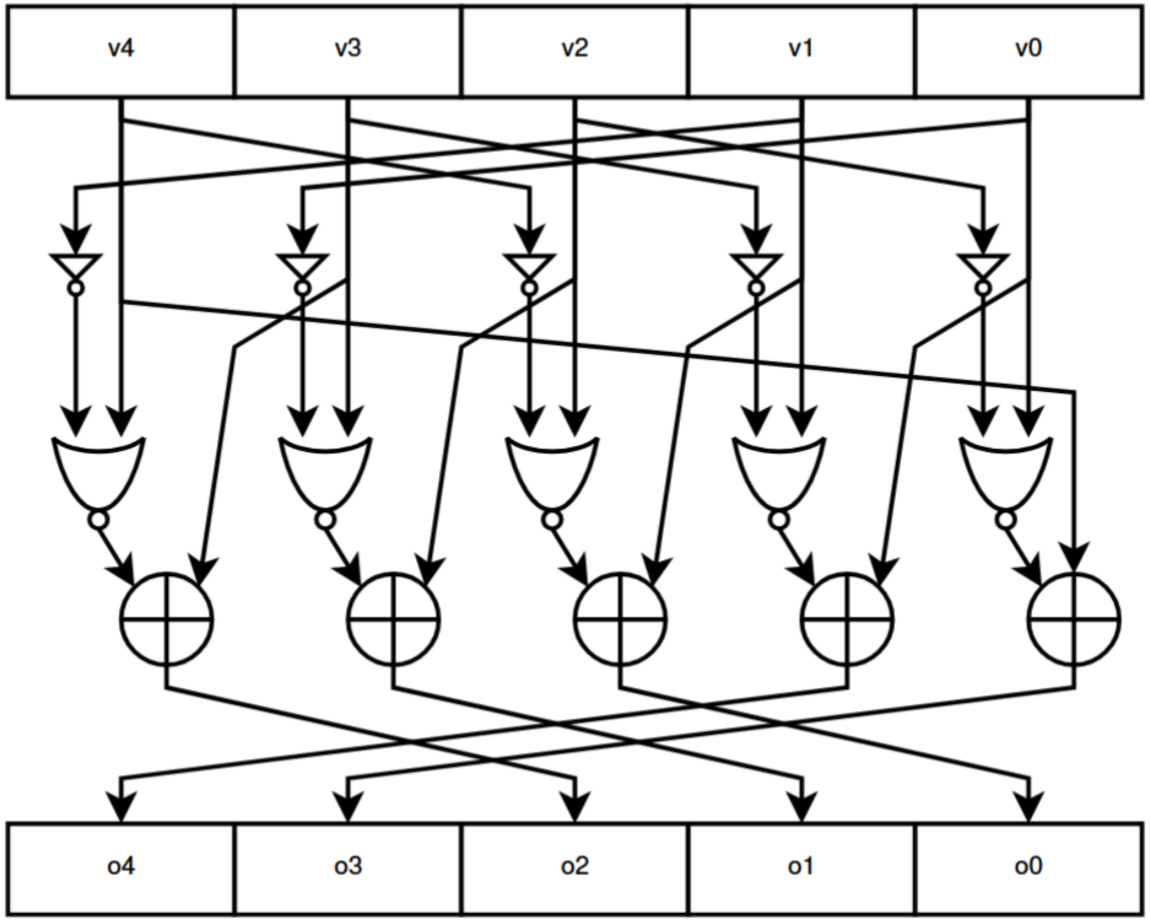
S-box size	T_max	GP			N_F	δ_F
		Max	Avg	Std dev		
4×4	16	**16**	16	0	4	4
5×5	42	**42**	41.73	1.01	12	2
6×6	86	**84**	80.47	4.72	24	4
7×7	182	**182**	155.07	8.86	56	2
8×8	364	318	281.87	13.86	82	20

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

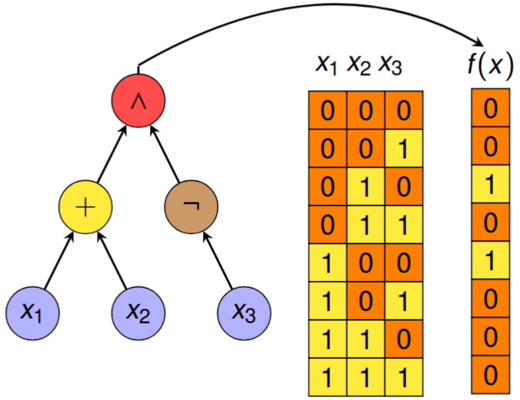


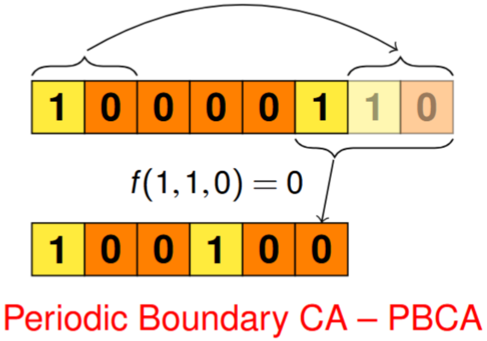
UNIVERSITY
OF TWENTE.

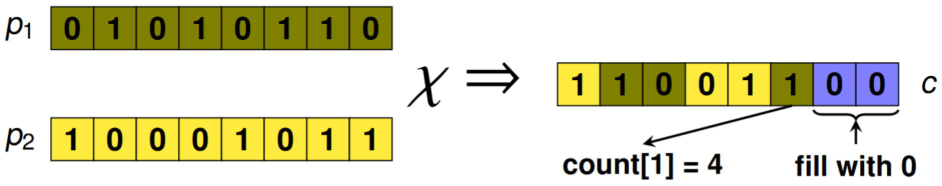
WRAPPING UP

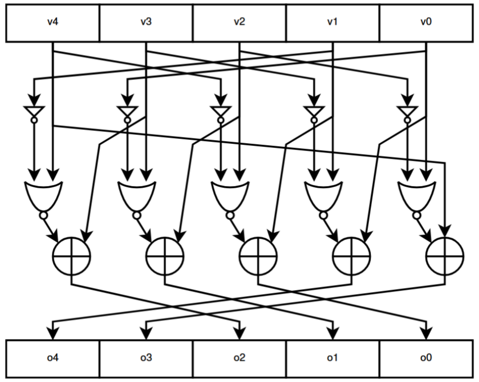
CONCLUSIONS AND FUTURE DIRECTIONS

- EA are more flexible than algebraic constructions for building crypto primitives
- Usually, GP fares better than GA [5]
- Future directions:
  - Fitness landscape analysis of the spaces of Boolean functions and S-boxes [6, 7]
  - Develop the use of GP to evolve algebraic constructions [2, 12, 20]



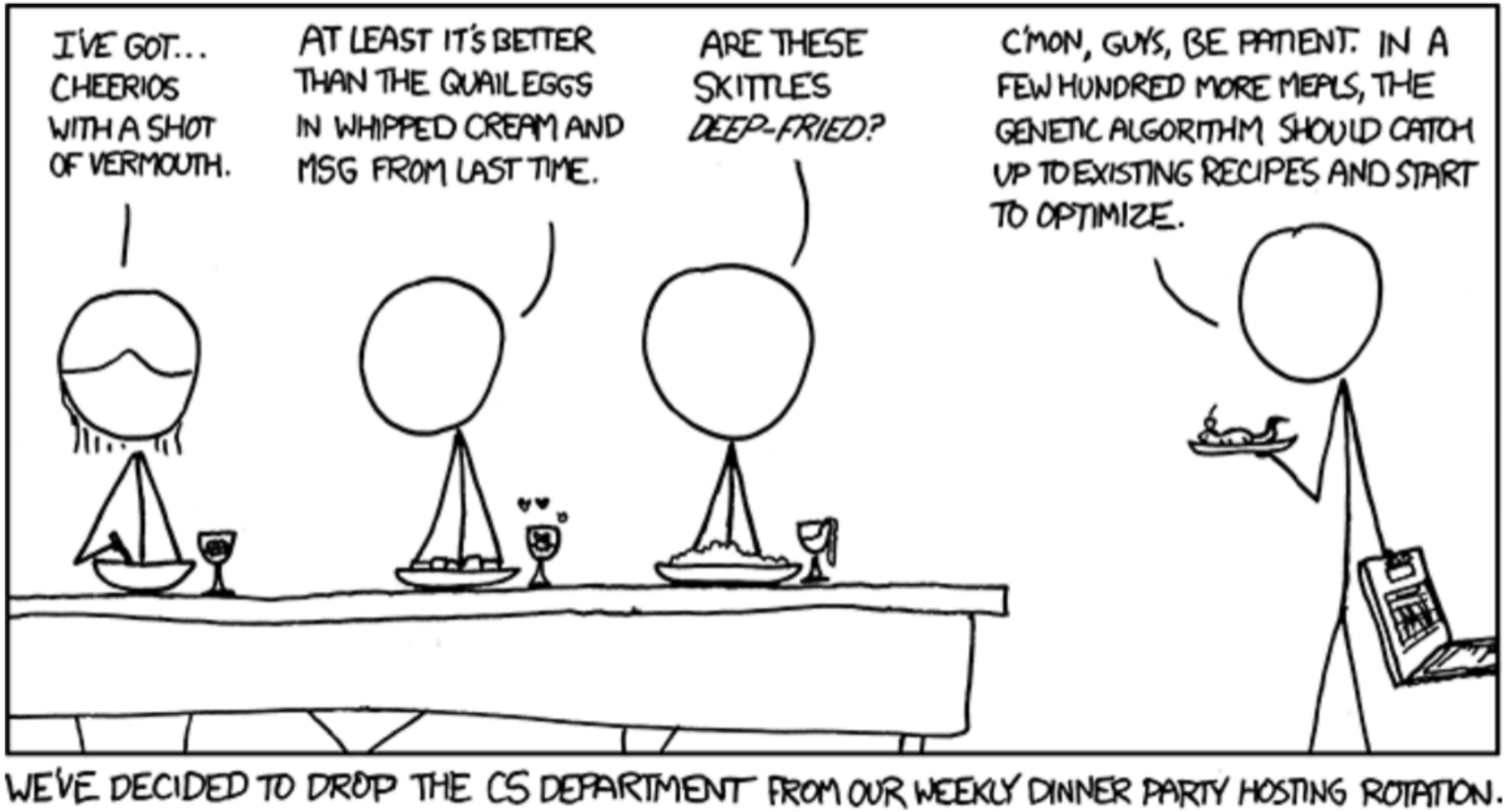






UNIVERSITY OF TWENTE.

THANKS!
QUESTIONS?



UNIVERSITY
OF TWENTE.

REFERENCES

1. G. Bertoni, J. Daemen, M. Peeters, G. Van Assche: The Keccak reference (2011)
2. 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)
3. C. Carlet: Boolean functions for cryptography and coding theory. Cambridge University Press (2021)
4. J. Daemen, R. Govaerts, J. Vandewalle: Invertible shift-invariant transformations on binary arrays. Appl. Math. Comput. 62:259–277 (1994)
5. 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)
6. D. Jakobovic, S. Picek, M.S.R. Martins, M. Wagner: Toward more efficient heuristic construction of Boolean functions. Appl. Soft Comput. 107:107327 (2021)
7. D. Jakobovic, S. Picek, M.S.R. Martins, M. Wagner: A characterisation of S-box fitness landscapes in cryptography. Proc. of GECCO 2019, pp. 285-293 (2019)
8. J. Katz, Y. Lindell: Introduction to Modern Cryptography. Third Edition, CRC Press (2021)
9. J.R. Koza: Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems (Vol. 34). Stanford, CA: Stanford University, Department of Computer Science (1990)
10. L. Manzoni, L. Mariot, E. Tuba: Balanced crossover operators in Genetic Algorithms. Swarm Evol. Comput. 54: 100646 (2020)
11. L. Manzoni, L. Mariot, E. Tuba: Does constraining the search space of GA always help?: the case of balanced crossover operators. Companion Proc. of GECCO 2019, pp. 151-152 (2019)
12. 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)
13. 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)
14. L. Mariot, D. Jakobovic, A. Leporati, S. Picek: Hyper-bent Boolean Functions and Evolutionary Algorithms. Proc. of EuroGP 2019, pp. 262-277 (2019)
15. L. Mariot, S. Picek, A. Leporati, D. Jakobovic: Cellular automata based S-boxes. Cryptogr. Commun. 11(1): 41-62 (2019)
16. W. Millan, J. Clark, E. Dawson: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. Proc. of EUROCRYPT 1998, pp. 489–499 (1998)
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)