Artificial Intelligence and Security Lab
Cyber Security Research Group
Delft University of Technology

Artificial Intelligence methods for the design of cryptographic primitives

Luca Mariot

L.Mariot@tudelft.nl

AICrypt@EUROCRYPT 2021

Zagreb, October 16, 2021

Outline

Intro – AI in symmetric crypto

AI-based optimization methods in cryptography

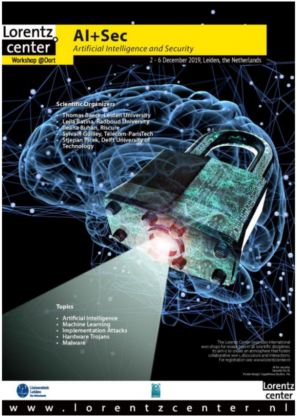
AI-based computational models in cryptography

Conclusions

Broader reference

This talk is based on the chapter:

L. Mariot, D. Jakobovic, T. Bäck, J. Hernandez-Castro: *Artificial Intelligence Methods in Cryptography*. AI+Sec: Artificial Intelligence and Security. Springer (forthcoming)



Outline

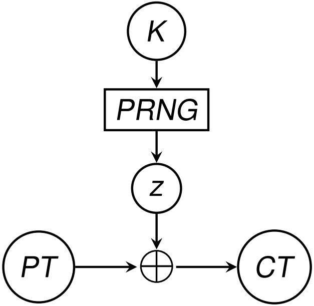
Intro – AI in symmetric crypto

AI-based optimization methods in cryptography

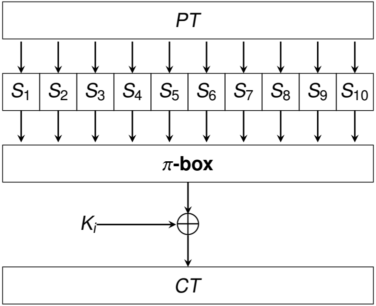
AI-based computational models in cryptography

Conclusions

Primitives in symmetric crypto



(a) Stream cipher



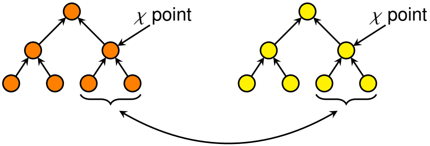
(b) Block cipher

Symmetric ciphers require several low-level primitives, such as:

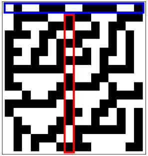
- Pseudorandom number generators (PRNG)
- Boolean functions $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and S-boxes
- Permutation (diffusion) layers, ...

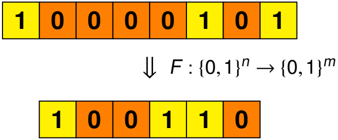
AI approaches to design symmetric primitives

- "Traditional" approach: ad-hoc and algebraic constructions to choose primitives with specific security properties
- "AI" approach: support the designer in choosing the primitives using AI methods/models from the following domains:
  - Optimization (Evolutionary algorithms, swarm intelligence...)



  - Computational models (cellular automata, neural networks...)





Outline

Intro – AI in symmetric crypto

AI-based optimization methods in cryptography

AI-based computational models in cryptography

Conclusions

Combinatorial Optimization

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

Minimization:

$$x^* = argmin_{x \in S}\{fit(x)\}$$

Maximization:

$$x^* = argmax_{x \in S}\{fit(x)\}$$

- Heuristic optimization algorithm: iteratively **tweaks** a set of candidate solutions using *fit* to drive the search

Evolutionary Algorithms (EA) – Genetic Algorithms (GA)

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

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

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

0	1	1	1	1	0	0	0

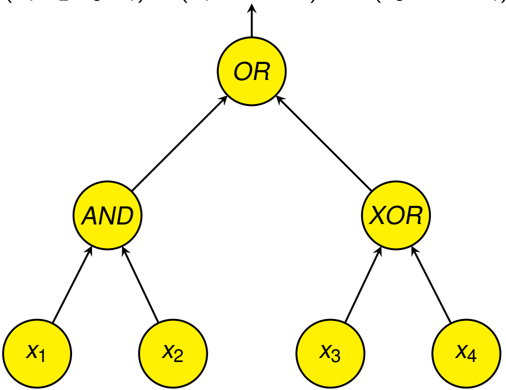
$$\Downarrow$$

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

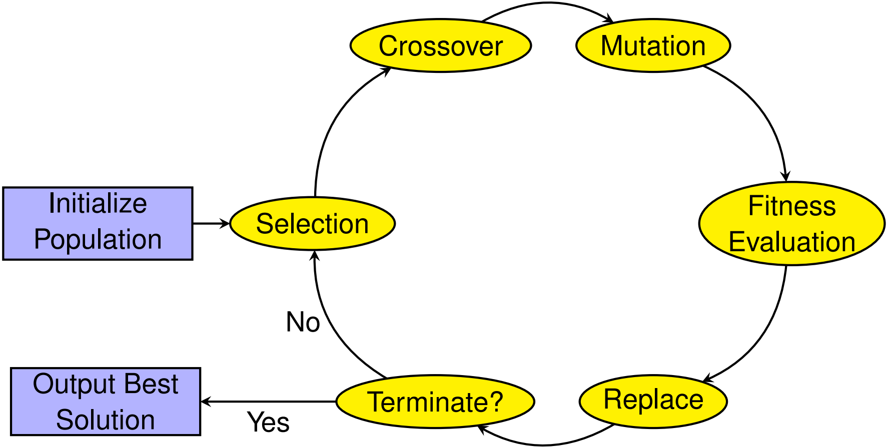
Evolutionary Algorithms (EA) – Genetic Programming (GP)

- **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, ...)

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



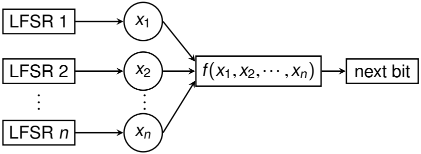
The EA Loop


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

EA in symmetric cryptography

Several design steps can be cast as combinatorial optimization problems, such as the search of:

- Boolean functions $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ for stream ciphers



- S-Boxes $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ for block ciphers

Possible advantages of using EA for this search:

- Diversity of solutions, due to the "blindness" of EA
- Flexibility of EA (optimizing several properties at once)

Design Criteria

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

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

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

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

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

Constructions of good Boolean Functions and S-Boxes

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

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

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

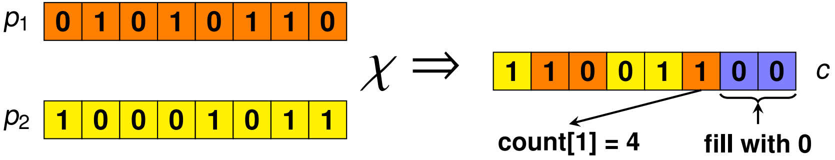
In practice, one usually resorts to:

- **Algebraic constructions** (*Maiorana-McFarland, Rothaus,...*) [Carlet21]
- **Combinatorial optimization techniques**
  - *Simulated Annealing* [Clark04]
  - *Evolutionary Algorithms* [Millan98, Picek16]
  - *Swarm Intelligence* [M15] [Mariot15], ...

Evolving Boolean Functions with GA

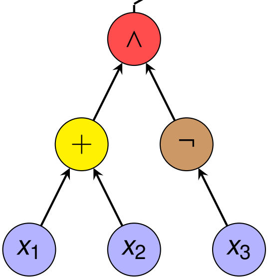
- GA encoding: represent the truth tables as 2^n-bit strings
- Fitness function: combines nonlinearity, algebraic degree, correlation-immunity
- Specialized crossover and mutation operators for preserving balancedness

Crossover Idea: Use *counters* to keep track of the multiplicities of zeros and ones [Millan98, Manzoni20]



Evolving Boolean Functions with GP

- The truth table is synthesized from a GP tree:

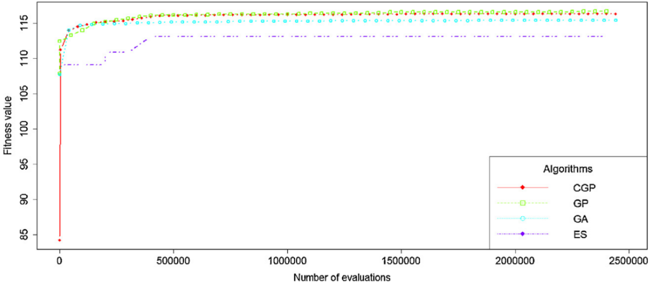


x_1	x_2	x_3	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

- Difficult to enforce constraints on balancedness with crossover and mutation

Results – Comparisons between GA and GP

- GP and its variants generally fares better than GA on optimizing Boolean functions [P16]



Source: *S. Picek, D. Jakobovic, J. Miller, L. Batina, M. Cupic: Cryptographic Boolean Functions: One Output, Many Design Criteria.* Appl. Soft Comp. 40 (2016) 635–653

- Similar results to traditional algebraic constructions

Outline

Intro – AI in symmetric crypto

AI-based optimization methods in cryptography

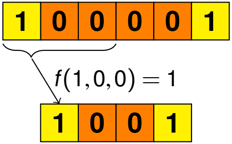
AI-based computational models in cryptography

Conclusions

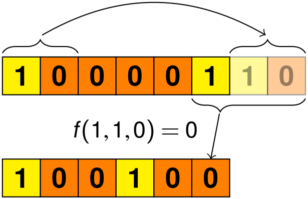
Cellular Automata

- One-dimensional Cellular Automaton (CA): a discrete parallel computation model composed of a finite array of n cells

Example: $n = 6$, $d = 3$, $\omega = 0$, $f(s_i, s_{i+1}, s_{i+2}) = s_i \oplus s_{i+1} \oplus s_{i+2}$ (rule 150)



No Boundary CA – NBCA

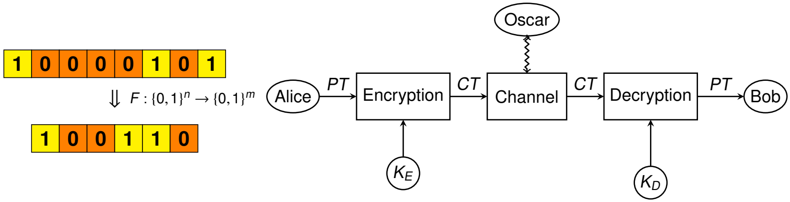


Periodic Boundary CA – PBCA

- Each cell updates its state $s \in \{0, 1\}$ by applying a local rule $f : \{0, 1\}^d \to \{0, 1\}$ to itself, the ω cells on its left and the $d - 1 - \omega$ cells on its right

Motivations

General Research Goal: Investigate cryptographic primitives defined by Cellular Automata

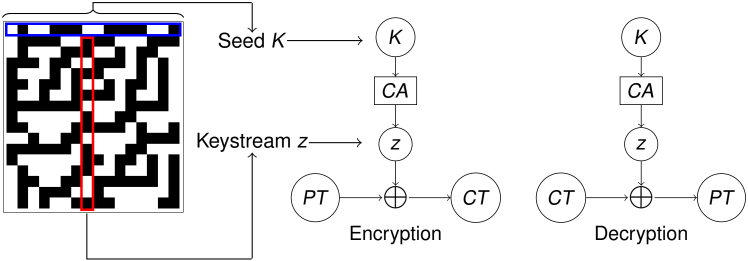


Why CA, anyway?

1. **Security from Complexity**: CA can yield very complex dynamical behaviors, depending on the local rule
2. **Efficient implementation**: Leverage CA parallelism and locality for lightweight cryptography

CA-based Crypto History: Wolfram's PRNG

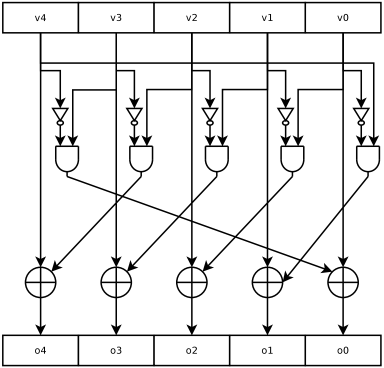
- CA-based Pseudorandom Generator (PRG) [Wolfram86]: central cell of rule 30 CA used as a stream cipher keystream



- Security claims based mainly on statistical/empirical tests
- This CA-based PRNG was later shown to be vulnerable, improvements by choosing larger local rules [Leporati14]

Real world CA-Based Crypto: Keccak χ S-box

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



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

Problem Statement

- Goal: Find PBCA of length n and diameter $d = n$:
  - with cryptographic properties on par with those of other real-world ciphers [Mariot19]
  - with low implementation cost [Picek17]
- Considered S-boxes sizes: from $n = 4$ to $n = 8$
- Genetic Programming to address this problem
- **Fitness function**: optimize *both* crypto (nonlinearity, differential uniformity) and implementation properties (GE measure)

Results

Table: Statistical results and comparison.

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

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

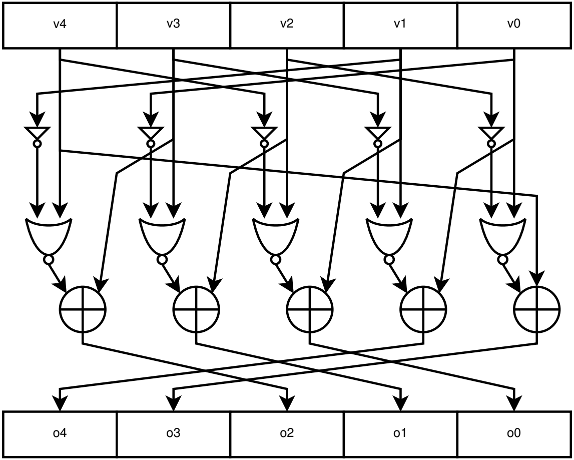
A Posteriori Analysis – Implementation Properties, $n = 5$

Table: Power is in *nW*, area in *GE*, and latency in *ns*. *DPow*: dynamic power, *LPow*: cell leakage power

Size	5×5	Rule	Keccak				
DPow.	321.684	LPow:	299.725	Area:	17	Latency:	0.14
Size	5×5	Rule	((v2 NOR NOT(v4)) XOR v1)				
DPow.	324.849	LPow:	308.418	Area:	17	Latency:	0.14
Size	5×5	Rule	((v4 NAND (v2 XOR v0)) XOR v1)				
DPow.	446.782	LPow:	479.33	Area:	24.06	Latency:	0.2
Size	5×5	Rule	(IF(v1, v2, v4) XOR (v0 NAND NOT(v3)))				
DPow.	534.015	LPow:	493.528	Area:	26.67	Latency:	0.17

- Results on par with the Keccak χ S-box

Example of Optimal CA S-box found by GP



Outline

Intro – AI in symmetric crypto

AI-based optimization methods in cryptography

AI-based computational models in cryptography

Conclusions

Conclusions and Perspectives

Summing up:

- Up to now, AI-based methods and models can help in solving certain specific design problems for symmetric ciphers.
- Many more open directions remain!

Open questions:

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

References

- [Keccak11] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche: The Keccak reference. (January 2011). `http://keccak.noekeon.org/`
- [Carlet21] C. Carlet: Boolean functions for cryptography and coding theory. Cambridge University Press (2021)
- [Clark04] J. Clark, J. Jacob, S. Maitra, P. Stanica: Almost Boolean Functions: The Design of Boolean Functions by Spectral Inversion. Computational Intelligence 20(3): 450-462 (2004)
- [Leporati14] A. Leporati and L. Mariot: Cryptographic properties of bipermutive cellular automata rules. J. Cell. Autom. 9(5-6):437–475 (2014)
- [Manzoni20] L. Manzoni, L. Mariot, E. Tuba: Balanced crossover operators in Genetic Algorithms. Swarm Evol. Comput. 54: 100646 (2020)
- [Mariot15] L. Mariot, A. Leporati: Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications. In: GECCO 2015 (Companion): 1425-1426. ACM (2015)
- [Mariot19] L. Mariot, S. Picek, A. Leporati, and D. Jakobovic. Cellular automata based S-boxes. Cryptography and Communications 11(1):41–62 (2019)
- [Millan98] W. Millan, J. Clark, E. Dawson: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. Proceedings of EUROCRYPT 1998, pp. 489-499 (1998)
- [Picek16] 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)
- [Picek17] S. Picek, L. Mariot, B. Yang, D. Jakobovic, N. Mentens: Design of S-boxes defined with cellular automata rules. Conf. Computing Frontiers 2017: 409-414 (2017)
- [Wolfram86] S. Wolfram. Cryptography with cellular automata. In CRYPTO '85, pp. 429–432 (1986)