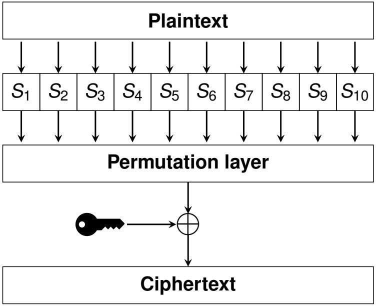
Evolving Boomerang Uniformity in Cryptographic S-boxes

Marko Djurasevic, Domagoj Jakobovic, **Luca Mariot**, Sihem Mesnager, Stjepan Picek

l.mariot@utwente.nl

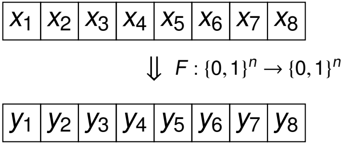
EvoAPPS 2023 – Brno, April 12, 2023

S-boxes in symmetric crypto



(a) Substitution-Permutation Network (SPN)

Zoom in on a **S-box** S_i:



(b) S-box S_i

Main properties of an S-box [C21]:

- **invertibility**
- High **nonlinearity**
- Low **differential uniformity**

Boolean Functions - Basic Definitions

- $\mathbb{F}_2 = \{0,1\}$, $\mathbb{F}_2^n$: n-dimensional vector space over $\mathbb{F}_2$
- n-variable *Boolean function*: mapping $f : \mathbb{F}_2^n \to \mathbb{F}_2$
- Common representation: *truth table* Ω_f

Example with $n = 3$:

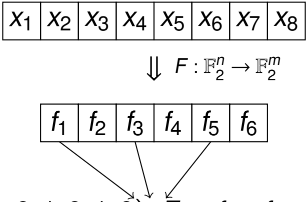
(x_1, x_2, x_3)	000	001	010	011	100	101	110	111
Ω_f	0	1	1	0	1	0	1	0

- *Scalar product* of $v, x \in \mathbb{F}_2^n$:

$$v \cdot x = \bigoplus_{i=1}^{n} v_i x_i = v_1 x_1 \oplus \cdots \oplus v_n x_n$$

S-Boxes: General definitions

- Substitution Box (S-box): vectorial mapping $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$
- Component functions $v \cdot F : \mathbb{F}_2^n \to \mathbb{F}_2$ linear combinations of the *output coordinates* $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$



$$(1,0,1,0,1,0) \cdot F = f_1 \oplus f_3 \oplus f_5$$

- In SPN ciphers, one uses S-boxes with $m = n$ [C21]

Balancedness

- F is balanced iff the component $v \cdot F$ is balanced for all $v \in \mathbb{F}_2^n$.
- When $m = n$, balanced S-boxes are *invertible*.
- Example: $n = m = 3$, the 3-WAY S-box

(x_1, x_2, x_3)	000	001	010	011	100	101	110	111
$F(x)$	000	101	110	001	011	010	100	111

$$\Downarrow$$

F is balanced (bijective)

Differential Uniformity

- *delta difference table* of F with respect to $a, b \in \mathbb{F}_2^n$:

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

- *differential uniformity* [N99]:

$$\delta_F = \max_{\substack{a \in \mathbb{F}_2^n \setminus \{0\} \\ b \in \mathbb{F}_2^m}} \#\Delta_F(a,b)$$

- Low $\delta_F \Rightarrow$ better resistance to *differential cryptanalysis*.
- *APN functions*: $\delta_F = 2$ (minimum possible)

Differential Uniformity – Example

- Example: $n = m = 3$

(x_1, x_2, x_3)	000	001	010	011	100	101	110	111
$F(x)$	000	101	110	001	011	010	100	111

$$\Downarrow$$

$\delta_F(a, b)$	000	001	010	011	100	101	110	111
001	0	2	0	2	0	2	0	2
010	0	0	0	0	2	2	2	2
011	0	2	0	2	2	0	2	0
100	0	0	2	2	0	0	2	2
101	0	2	2	0	0	2	2	0
110	0	0	2	2	2	2	0	0
111	0	2	2	0	2	0	0	2

$\Rightarrow$ differential uniformity of F: $\delta_f = 2$

Boomerang Uniformity

- Additional property related to *boomerang attacks*
- Represent $x \in \mathbb{F}_2^n$ as element in the extension field $\mathbb{F}_{2^n}$
- *Boomerang Connectivity Table*:

$$T_F(a,b) = \{x \in \mathbb{F}_{2^n} : F^{-1}(F(x)+a) + F^{-1}(F(x+b)+a) = b\}$$

- *Boomerang uniformity* [B18]:

$$\beta_F = \max_{a,b \neq 0} \#T_F(a,b).$$

- $\delta_F = 2 \Leftrightarrow \beta_F = 2$

Constructions of good 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 $n > 5$!

Even worse with S-boxes:

- # of $n \times n$ S-boxes: 2^{n2^n}
- # of $n \times n$ invertible S-boxes: $(2^n)!$

In practice

- **Algebraic Constructions** [Me20, C21]
- **Metaheuristics** [C02, M22]

Solutions Encoding – Integer

Integer Encoding:

0	**2**	**3**	**5**	**2**	**1**	**3**	**6**

$\Downarrow$

(x_1, x_2, x_3)	000	001	010	011	100	101	110	111
$F(x)$	000	010	011	101	010	001	011	110

- **Advantages:** straightforward representation, can use classic operators (one-point crossover, flip mutation...)
- **Disadvantages:** does not ensure bijectivity

Solutions Encoding – Permutation

Permutation Encoding:

0	**5**	**6**	**1**	**3**	**2**	**4**	**7**

$$\Downarrow$$

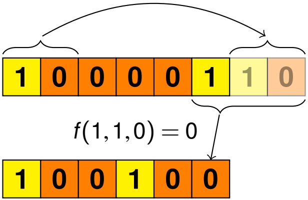
(x_1, x_2, x_3)	000	001	010	011	100	101	110	111
$F(x)$	000	101	110	001	011	010	100	111

- **Advantages:** ensure bijectivity
- **Disadvantages:** use of specialized operators to preserve the permutation (PMX/CX crossover, etc.)

Solutions Encoding – Cellular Automata

- Each coordinate updates its state $s \in \{0,1\}$ by evaluating a local rule $f : \mathbb{F}_2^d \to \mathbb{F}_2$ on itself and the $d-1$ cells on its right

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

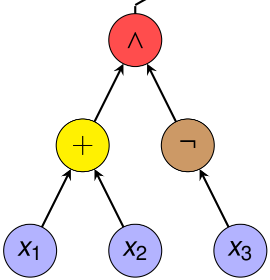


- **Advantages:** Focus just on optimizing the local rule (a Boolean function) [P17, P17b, M19b]
- **Disadvantages:** does not ensure bijectivity

CA rule evolution with GP

- The truth table $f(x)$ of the rule is synthesized from the tree

Example: $n = 3$, $f : \{0,1\}^3 \to \{0,1\}$



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

Fitness functions

First fitness: just minimize

$$fitness_1 = \beta_f$$

- Used only with permutation encoding

Second fitness: minimize

$$fitness_2 = \begin{cases} 2^n + BAL & \text{if } BAL > 0 \\ \beta, & \text{otherwise.} \end{cases}$$

- Used with integer and CA encoding
- BAL: *balancedness penalty* (# of missing outputs)

Experimental Settings

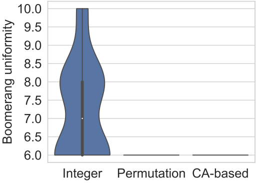
Optimization approaches:

- **Single-objective** (minimize only β_F)
- **Multi-objective** (min δ_F and β_F) $\Rightarrow$ NSGA-II [D02]
- **Random Search** as a baseline

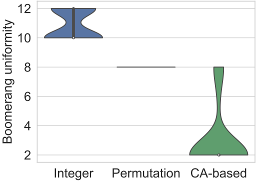
EA Parameters:

- Instances: $n = 4, 5, 6, 7, 8$
- Fitness Evals.: 500 000
- Breeding: Steady-state
- Population size: 500
- GP tree depth: n
- Tournament size: 3
- Mutation rate: 0.7
- Independent Runs: 30

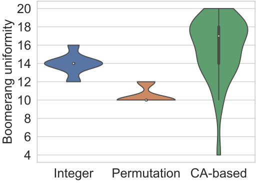
Results – Single-objective



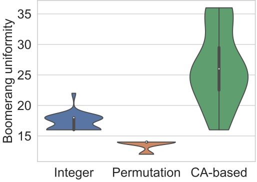
4×4



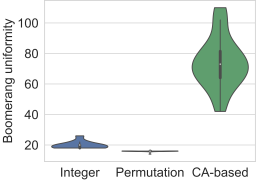
5×5



6×6



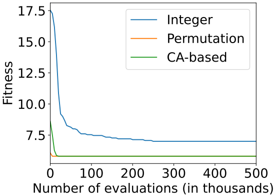
7×7



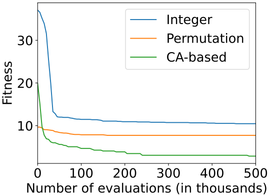
8×8

Main Finding: Permutation has lowest fitness across all sizes, CA finds best solutions up to 6×6

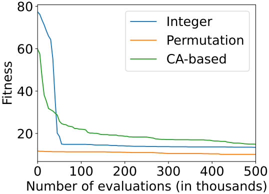
Results – Convergence



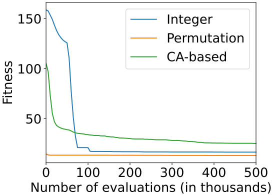
4×4



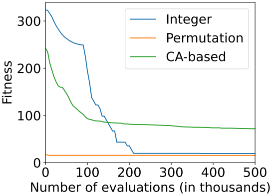
5×5



6×6



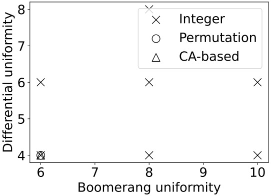
7×7



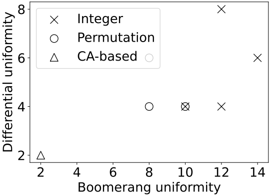
8×8

Main Finding: CA fitness still improving even towards the end of the optimization run

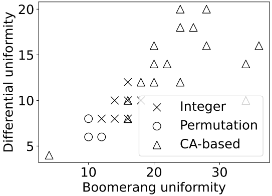
Results – Multi-Objective



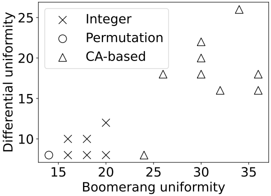
4×4



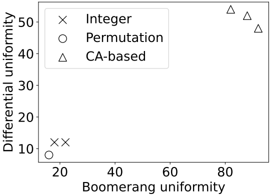
5×5



6×6



7×7



8×8

Main Finding: CA dominating for 5×5 and 6×6, Permutation for 7×7 and 8×8

Conclusions and Future Works

Conclusions:

- No single encoding working best across all instances!
- Optimal solutions found only up to 5×5
- For 6×6, optimal β (but for non-APN functions)

Future work:

- Multi-objective optimization of nonlinearity and boomerang uniformity
- Incorporate domain-specific knowledge in the EA [Ma20]

References

- [B18] Boura, C., Canteaut, A.: On the boomerang uniformity of cryptographic Sboxes. IACR Transactions on Symmetric Cryptology 2018(3): 290–310 (2018)
- [C21] Carlet, C.: Boolean functions for cryptography and coding theory. Cambridge University Press (2021)
- [C02] Clark, J.A., Jacob, J.L., Stepney, S.: The design of s-boxes by simulated annealing. In: Proceedings of CEC, vol. 2, pp. 1533–1537 (2004)
- [D02] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002)
- [M22] Mariot, L., Jakobovic, D., Bäck, T., Hernandez-Castro, J.: Artificial Intelligence for the Design of Symmetric Cryptographic Primitives. Security and Artificial Intelligence 2022, pp. 3-24 (2022)
- [M19] Mariot, L., Picek, S., Leporati, A., Jakobovic, D.: Cellular automata based S-boxes. Cryptography and Communications 11(1):41–62 (2019)
- [Ma20] Manzoni, L., Mariot, L., Tuba, E.: Balanced crossover operators in Genetic Algorithms. Swarm Evol. Comput. 54: 100646 (2020)
- [Me20] Mesnager, S., Tang, C., Xiong, M.: On the boomerang uniformity of quadratic permutations. Des. Codes Cryptogr. 88(10): 2233–2246 (2020)
- [N99] Nyberg, K.: Perfect Nonlinear S-Boxes. In: Proceedings of EUROCRYPT ’91, pp. 378–386 (1991)
- [P17] Picek, S., Mariot, L., Yang, B., Jakobovic, D., Mentens, N.: Design of S-boxes defined with cellular automata rules. Conf. Computing Frontiers 2017: 409–414 (2017)
- [P17b] Picek, S., Mariot, L., Leporati, A., Jakobovic, D.: Evolving s-boxes based on cellular automata with genetic programming. In: Proceedings of GECCO 2017, pp. 251–252 (2017)