

# A new approach to evolution of bijective S-boxes with AI based swap predictions

Terézia Gurbalová   **Pavol Zajac**<sup>1</sup>

CECC 2025, Budapest, Hungary

---

<sup>1</sup>his work was in part supported by the Slovak Research and Development Agency under the Contract no. APVV-23-0292, and in part by grant VEGA 1/0105/23.

# Outline

---

- ① S-boxes
- ② Problem definition and methodology
- ③ Predictor results
- ④ Evolution results
- ⑤ Conclusions

# Our object of study

- Our focus are bijective  $n$ -bit S-boxes:

$$S : \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$$

- Our aim is to optimize differential properties of  $S$ :
  - difference distribution table (DDT)

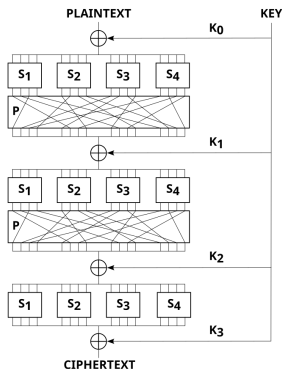
$$D_{a,b} = |\{x \in \mathbb{Z}_2^n; S(x) + S(x+a) = b\}|$$

- differential uniformity:

$$\delta = \max \{D_{a,b}; a \neq 0\}$$

- multiplicity of the differential uniformity value:

$$\nu = |\{D_{a,b} = \delta\}|$$



# Classical approaches

---

- Mathematical constructions: base  $S$  on a specific function, such as  $S(x) = Ax^{-1} + b$  (AES)
- Random S-boxes: generate and test properties
- S-box evolution: hill-climbing, genetic algorithms, ...

# S-box evolution

---

- We use a simplified model of S-box evolution:
  - ① Start with random S-box  $S$
  - ② Repeat:
    - ① Update S-box definition:  $S \mapsto S_{new}$
    - ② If  $S_{new}$  is better than  $S$ , replace  $S$  with  $S_{new}$
- S-box update is limited to a single swap of 2 values (in S-box vector of values)

$$\begin{array}{l}
 S \\
 S_{new}
 \end{array}
 \left| \begin{array}{cccccccccccccccc}
 13 & 4 & 10 & 3 & 14 & 0 & 8 & 6 & 7 & 15 & 5 & 11 & 2 & 9 & 12 & 1 \\
 13 & \mathbf{14} & 10 & 3 & \mathbf{4} & 0 & 8 & 6 & 7 & 15 & 5 & 11 & 2 & 9 & 12 & 1
 \end{array} \right| \begin{array}{l}
 \delta = 6 \\
 \delta = 4
 \end{array}$$

# S-box evolution

---

- We use a simplified model of S-box evolution:
  - ① Start with random S-box  $S$
  - ② Repeat:
    - ① Update S-box definition:  $S \mapsto S_{new}$
    - ② If  $S_{new}$  is better than  $S$ , replace  $S$  with  $S_{new}$
- S-box update is limited to a single swap of 2 values (in S-box vector of values)

## Problem

*Is it possible to predict suitable swap value with high accuracy using AI methods?*

# S-box evolution with AI

---

## Problem

*Is it possible to predict suitable swap value with high accuracy using AI methods?*

- Objective: Construct AI (black-box) predictor:

**INPUT** vector of values of S-box  $S$

**OUTPUT** swap values  $x_1, x_2$

- Let  $S_{new}(x_1) = S(x_2)$ ,  $S_{new}(x_2) = S(x_1)$   $S_{new}(x) = S(x)$  for  $x \notin \{x_1, x_2\}$
- Predictor **succeeds** if  $\delta_S > \delta_{S_{new}}$  or ( $\delta_S = \delta_{S_{new}}$  and  $\nu_S > \nu_{S_{new}}$  )
- **Accuracy** : ratio of successful predictions to total number of predictions

# AI predictor

---

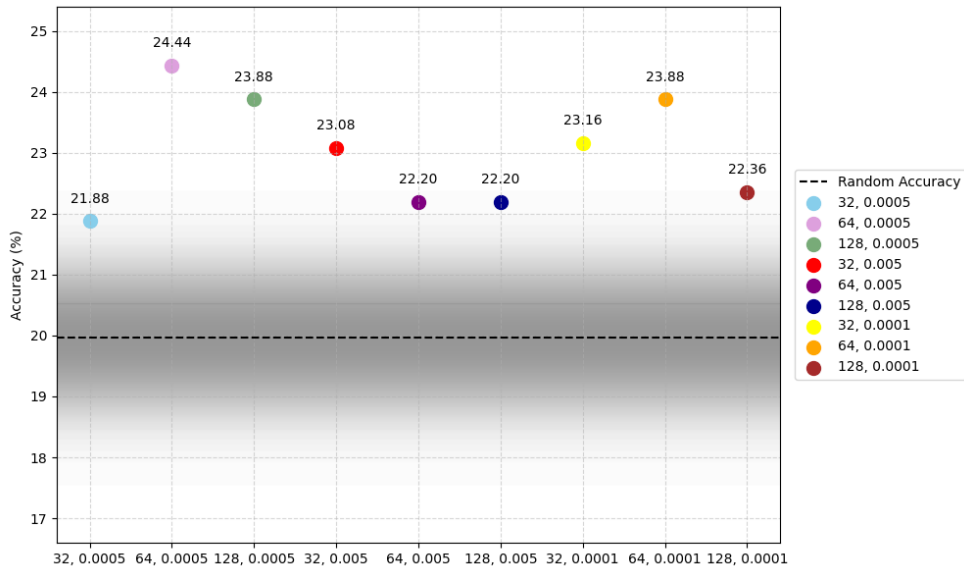
- AI model is trained on positive examples (unique S-boxes + improving swaps)
  - obtained with a hill climbing method.
  - split into 90% training set, 10% validation set
- Two main approaches:
  - ① CNN : convolution neural network architecture
  - ② Seq2seq : based on LSTM neural network architecture
- Model results are probability (softmax) distributions over choice of  $x_1, x_2$  for given input S-box

## Best results for CNN

batch_size	learning_rate	$8 \times 8$	duration for $8 \times 8$
32	0.0005	21.88%	189 s
<b>64</b>	<b>0.0005</b>	<b>24.44%</b>	<b>156 s</b>
128	0.0005	23.88%	136 s
32	0.005	23.08%	197 s
64	0.005	22.20%	159 s
128	0.005	22.20%	147 s
32	0.0001	23.16%	189 s
64	0.0001	23.88%	163 s
128	0.0001	22.36%	145 s

Table: DA\_accuracy and time of training for different batch\_size and learning\_rate of the model.

# Predictor results



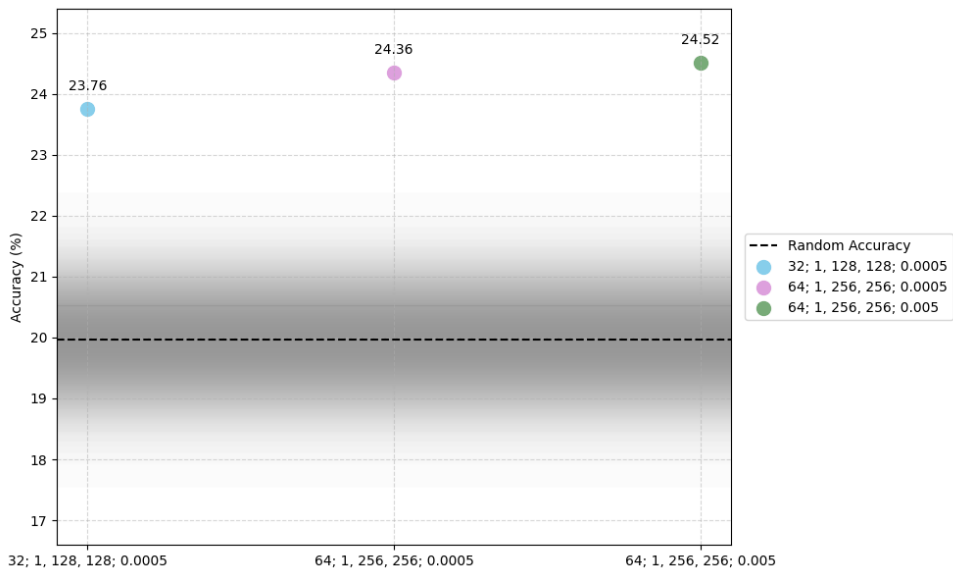
## Best results for Seq2Seq model

---

batch_size	layers, enc, dec	learning_rate	$8 \times 8$	duration $8 \times 8$
32	1, 128, 128	0.0005	23.76%	101 s
64	1, 256, 256	0.0005	24.36%	1156 s
<b>64</b>	<b>1, 256, 256</b>	<b>0.005</b>	<b>24.52%</b>	<b>1226 s</b>

**Table:** DA\_\_accuracy for combined best configurations on  $8 \times 8$  S-boxes.

# Predictor results



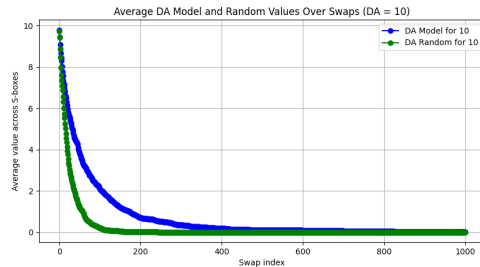
## Using AI predictor in S-box evolution

---

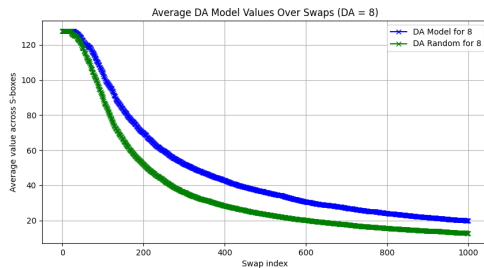
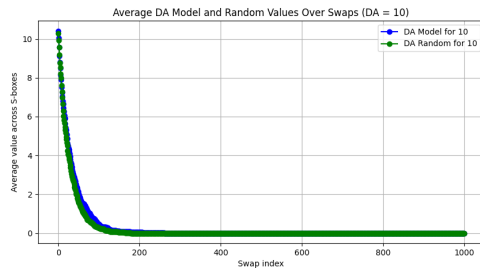
Prediction becomes deterministic for given  $S$ : how can we use predictor, if prediction fails?

- Use list of predictions  $[(x_1, x_2)]$  based on softmax score:
  - computed independently for  $x_1, x_2$ ,
  - joint probability softmax for pairs  $(x_1, x_2)$ ,
- Alternate strategy: use random affine transformation of input S-box

# Use separate softmax results for evolution



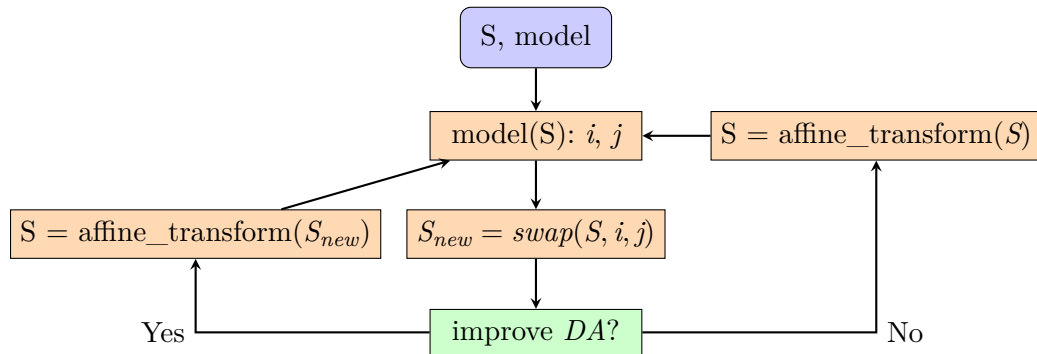
# Use joint softmax results for evolution



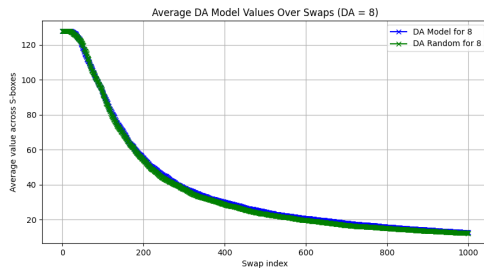
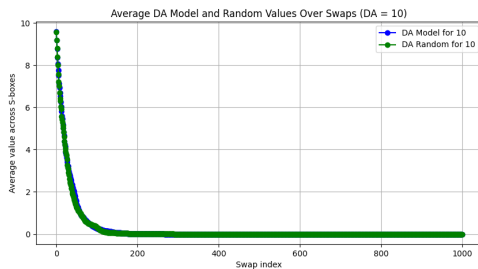
## Using AI predictor in S-box evolution

Prediction becomes deterministic for given  $S$ : how can we use predictor, if prediction fails?

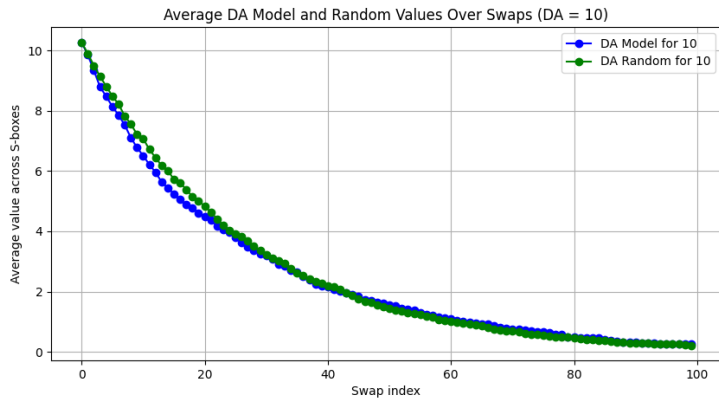
- Alternate strategy: use random affine transformation of input S-box



# Affine randomization + best predictor



# Affine randomization + best predictor



## Summary

---

- We have created AI predictor for S-box improving swaps.
- AI predictor is better than random guess in a single-guess scenario.
- AI predictor cannot outperform random guessing in longer evolution scenario (repeated guessing).
  
- Open problem: What is the explanation for experimental results?
- Can we improve the results with more/better learning examples?

Thank you for your attention. Questions? Comments?