# Algorithmic toolkit for linearization of S-boxes

Alex Biryukov  Philip Tureček  Aleksei Udovenko

Dagstuhl Seminar 26061, February 2nd

DCS and SnT, University of Luxembourg



Work funded by Luxembourg's FNR projects
PQseal (C24/IS/18978392) and CryptoFin (C22/IS/17415825)

# Linearization

**Definition**

An affine map $A : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a vectorial $g$-approximation of $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ if

$$\#\{x \in \mathbb{F}_2^n \mid S(x) = A(x)\} = g$$

# Linearization

**Definition**

An affine map $A : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a vectorial $g$-approximation of $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ if

$$\Pr_x[S(x) = A(x)] = \frac{g}{2^n}$$

# Linearization

**Definition**

An affine map $A : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a vectorial $g$-approximation of $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ if

$$\Pr_x[S(x) = A(x)] = \frac{g}{2^n}$$

**Definition (Vectorial linearity, Chen and Fu 2001)**

$$\mathrm{VecLin}_S = \max_{\substack{\text{aff.} \\ \text{maps } A}} \#\{x \in \mathbb{F}_2^n \mid S(x) = A(x)\}$$

# Cryptanalysis in the Wild

Literature examples (linearize then solve algebraically):

- LowMC [Banik, Barooti, Durak, and Vaudenay 2020 ToSC]
- RAIN/AIMer [Zhang, Wang, Yu, Guo, and Cui 2023 ASIACRYPT]
- AES-like hashing (preimages) [Chen, Guo, List, Shi, and Zhang 2024 EUROCRYPT]

# Cryptanalysis in the Wild

Literature examples (linearize then solve algebraically):

- LowMC [Banik, Barooti, Durak, and Vaudenay 2020 ToSC]
- RAIN/AIMer [Zhang, Wang, Yu, Guo, and Cui 2023 ASIACRYPT]
- AES-like hashing (preimages) [Chen, Guo, List, Shi, and Zhang 2024 EUROCRYPT]

  note: one approximation vs partition

Literature examples (linearize then solve algebraically):

- LowMC [Banik, Barooti, Durak, and Vaudenay 2020 ToSC]
- RAIN/AIMer [Zhang, Wang, Yu, Guo, and Cui 2023 ASIACRYPT]
- AES-like hashing (preimages) [Chen, Guo, List, Shi, and Zhang 2024 EUROCRYPT]

  note: one approximation vs partition
- Keccak (preimages) [Qiao, Song, Liu, and Guo 2017 EUROCRYPT]
  note: linearization on *subspaces*

# Theoretical studies

- [Chen and Fu 2001]
- [Liu, Mesnager, and Chen 2017 CCDS]
- [Carlet 2021 TIT]
- [Ryabov 2023; Ryabov 2024a; Ryabov 2024b ]
- [Courtois, Amiel, and Fonvillars 2024 eprint]
- [Nagy 2025a; Nagy 2025b CCDS, JCTA]

# Theoretical studies

- [Chen and Fu 2001]
- [Liu, Mesnager, and Chen 2017 CCDS]
- [Carlet 2021 TIT]
- [Ryabov 2023; Ryabov 2024a; Ryabov 2024b ]
- [Courtois, Amiel, and Fonvillars 2024 eprint]
- [Nagy 2025a; Nagy 2025b CCDS, JCTA]

Rather loose bounds, relations to diff. uniformity (APN) / Boolean nonlinearity (Bent), ...

# Theoretical studies

- [Chen and Fu 2001]
- [Liu, Mesnager, and Chen 2017 CCDS]
- [Carlet 2021 TIT]
- [Ryabov 2023; Ryabov 2024a; Ryabov 2024b ]
- [Courtois, Amiel, and Fonvillars 2024 eprint]
- [Nagy 2025a; Nagy 2025b CCDS, JCTA]

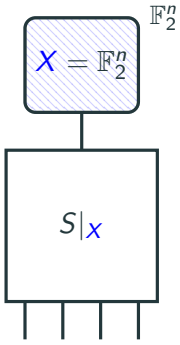
Rather loose bounds, relations to diff. uniformity (APN) / Boolean nonlinearity (Bent), ...

What are the best approximations for 8-bit crypto S-boxes?

# Outline

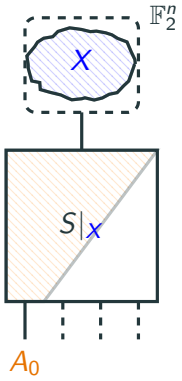
# Plan

# Reduction framework (LAT)



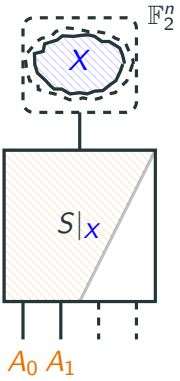

1. Start with $X = \mathbb{F}_2^n$

# Reduction framework (LAT)



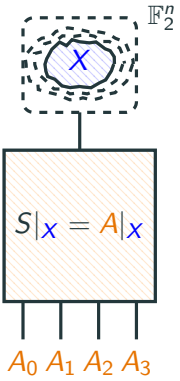

1. Start with $X = \mathbb{F}_2^n$
2. For each output bit $i = 0 \ldots n-1$
   - 2.1 Compute Walsh transform of $S_i$ restricted to inputs $X$
   - 2.2 Choose mask $\alpha$ achieving largest absolute bias
   - 2.3 Reduce $X$ to $\{x \in X \mid \langle \alpha, x \rangle = S_i(x)\}$

# Reduction framework (LAT)



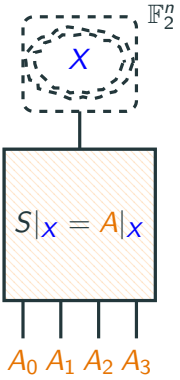

1. Start with $X = \mathbb{F}_2^n$
2. For each output bit $i = 0 \ldots n-1$
   - 2.1 Compute Walsh transform of $S_i$ restricted to inputs $X$
   - 2.2 Choose mask $\alpha$ achieving largest absolute bias
   - 2.3 Reduce $X$ to $\{x \in X \mid \langle \alpha, x \rangle = S_i(x)\}$

# Reduction framework (LAT)




1. Start with $X = \mathbb{F}_2^n$
2. For each output bit $i = 0 \ldots n-1$
   - 2.1 Compute Walsh transform of $S_i$ restricted to inputs $X$
   - 2.2 Choose mask $\alpha$ achieving largest absolute bias
   - 2.3 Reduce $X$ to $\{x \in X \mid \langle \alpha, x \rangle = S_i(x)\}$

# Reduction framework (LAT)




1. Start with $X = \mathbb{F}_2^n$
2. For each output bit $i = 0 \ldots n-1$
   - 2.1 Compute Walsh transform of $S_i$ restricted to inputs $X$
   - 2.2 Choose mask $\alpha$ achieving largest absolute bias
   - 2.3 Reduce $X$ to $\{x \in X \mid \langle \alpha, x \rangle = S_i(x)\}$
3. Interpolate $A$ from $A(x) = S(x)$ for all $x \in X$

# Exhaustive reduction algorithm (LAT)

**Input:** lower bound $B > 0$

1. Start with $X = \mathbb{F}_2^n$
2. For each output bit $i = 0 \dots n-1$
   - 2.1 Compute Walsh transform of $S_i$ restricted to inputs $X$
   - 2.2 For every signed mask $\alpha$
     - 2.2.1 Reduce $X$ to $\{x \in X \mid \langle \alpha, x \rangle = S_i(x)\}$
     - 2.2.2 If $|X| \geq B$, continue with other bits recursively. Otherwise, abort.

**Output:** all affine approximations of $S$ of size at least $B$

# Exhaustive reduction algorithm (LAT) summary

- exhaustive: guarantee to find the approximation if it exists

1

# Exhaustive reduction algorithm (LAT) summary

- exhaustive: guarantee to find the approximation if it exists
- larger approximations are easier to find

1

# Exhaustive reduction algorithm (LAT) summary

- exhaustive: guarantee to find the approximation if it exists
- larger approximations are easier to find
- feasible for 8-bit S-boxes with best known bounds:
  - $B = 18 + 1$ (AES) takes 870 core-hours[1]
  - $B = 15 + 1$ (best known $g$) takes 20 736 core-hours (quadratic $\Rightarrow$ 81 core-hours)
  - $B = 26 + 1$ (SNOW-3G) takes 27 core-hours

[1] AMD EPYC 7H12 @ 3.3GHz 128 cores

# Exhaustive reduction algorithm (LAT) summary

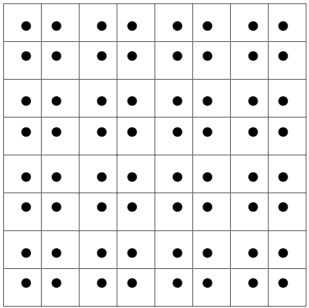
- exhaustive: guarantee to find the approximation if it exists
- larger approximations are easier to find
- feasible for 8-bit S-boxes with best known bounds:
  - $B = 18 + 1$ (AES) takes 870 core-hours[1]
  - $B = 15 + 1$ (best known $g$) takes 20 736 core-hours (quadratic $\Rightarrow$ 81 core-hours)
  - $B = 26 + 1$ (SNOW-3G) takes 27 core-hours

[1] AMD EPYC 7H12 @ 3.3GHz 128 cores

# Plan

# Greedy extension idea (DDT)

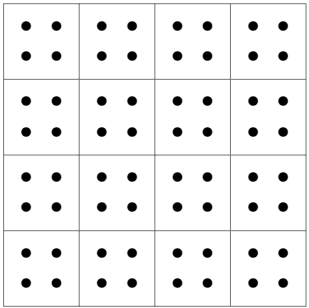
Input space: $(\mathbb{F}_2^n)$



- write $A(x) = L(x) + c$

# Greedy extension idea (DDT)

Input space: $(\mathbb{F}_2^n)$

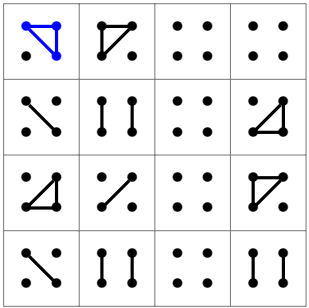


- write $A(x) = L(x) + c$

- every *cell* is a coset of the subspace of known values of $L$

# Greedy extension idea (DDT)

Input space: $(\mathbb{F}_2^n)$



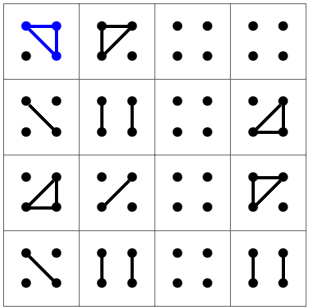
- write $A(x) = L(x) + c$

- every *cell* is a coset of the subspace of known values of $L$

- edge $x_i - x_j$ when
  $S(x_i) \oplus S(x_j) = A(x_i) \oplus A(x_j) = L(x_i \oplus x_j)$

# Greedy extension idea (DDT)

Input space: $(\mathbb{F}_2^n)$



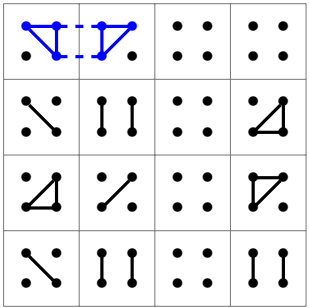
- write $A(x) = L(x) + c$

- every *cell* is a coset of the subspace of known values of $L$

- edge $x_i - x_j$ when
  $S(x_i) \oplus S(x_j) = A(x_i) \oplus A(x_j) = L(x_i \oplus x_j)$

# Greedy extension idea (DDT)

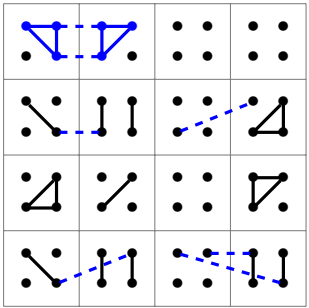
Input space: $(\mathbb{F}_2^n)$



- write $A(x) = L(x) + c$
- every *cell* is a coset of the subspace of known values of $L$
- edge $x_i - x_j$ when $S(x_i) \oplus S(x_j) = A(x_i) \oplus A(x_j) = L(x_i \oplus x_j)$
- greedily merge with largest clique

# Greedy extension idea (DDT)

Input space: $(\mathbb{F}_2^n)$

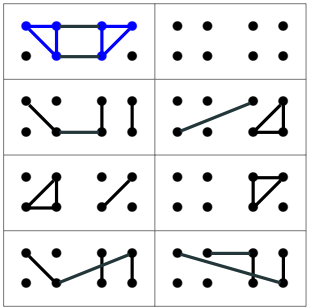


- write $A(x) = L(x) + c$
- every *cell* is a coset of the subspace of known values of $L$
- edge $x_i - x_j$ when $S(x_i) \oplus S(x_j) = A(x_i) \oplus A(x_j) = L(x_i \oplus x_j)$
- greedily merge with largest clique
- find new edges (using DDT):

$$S(x) \oplus S(x \oplus \delta_x) = \delta_y = A(x) \oplus A(x \oplus \delta_x) = L(\delta_x)$$

# Greedy extension idea (DDT)

Input space: $(\mathbb{F}_2^n)$

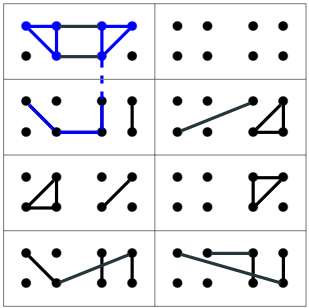


- write $A(x) = L(x) + c$
- every *cell* is a coset of the subspace of known values of $L$
- edge $x_i - x_j$ when $S(x_i) \oplus S(x_j) = A(x_i) \oplus A(x_j) = L(x_i \oplus x_j)$
- greedily merge with largest clique
- find new edges (using DDT):

$$S(x) \oplus S(x \oplus \delta_x) = \delta_y = A(x) \oplus A(x \oplus \delta_x) = L(\delta_x)$$

# Greedy extension idea (DDT)

Input space: $(\mathbb{F}_2^n)$



- write $A(x) = L(x) + c$
- every *cell* is a coset of the subspace of known values of $L$
- edge $x_i - x_j$ when $S(x_i) \oplus S(x_j) = A(x_i) \oplus A(x_j) = L(x_i \oplus x_j)$
- greedily merge with largest clique
- find new edges (using DDT):

$$S(x) \oplus S(x \oplus \delta_x) = \delta_y = A(x) \oplus A(x \oplus \delta_x) = L(\delta_x)$$

# Greedy extension algorithm (DDT) summary

- heuristic (greedy/step) - no guarantees

# Greedy extension algorithm (DDT) summary

- heuristic (greedy/step) - no guarantees
- optimized, very fast: $\mathcal{O}(2^n)$ DDT lookups and XORs

# Greedy extension algorithm (DDT) summary

- heuristic (greedy/step) - no guarantees
- optimized, very fast: $\mathcal{O}(2^n)$ DDT lookups and XORs
- good *randomization*: large space for initial choices

# Greedy extension algorithm (DDT) summary

- heuristic (greedy/step) - no guarantees
- optimized, very fast: $\mathcal{O}(2^n)$ DDT lookups and XORs
- good *randomization*: large space for initial choices
- almost all our best approximations were found using this algorithm

# Plan

# Crypto S-boxes

| $n$ | S-box | $\delta$ | Lin | deg | $\mathrm{VecLin}_S$ (Out of 256) |
|---|---|---|---|---|---|
| 8 | $S_{15}$ (APN)[2] | 2 | 32 | 2 | 15 |
| | $x^3$ (APN) | 2 | 32 | 2 | 16 |
| | AES | 4 | 32 | 7 | 18 |
| | Kalyna $\pi_2$, $\pi_3$ | 8 | 48 | 7 | 19 |
| | BelT | 8 | 52 | 7 | 20 |
| | Kuznyechik | 8 | 56 | 7 | 20 |
| | Anubis | 8 | 68 | 7 | 21 |
| | Skipjack | 12 | 56 | 7 | 21 |
| | Scream | 8 | 64 | 6 | 22 |
| | SNOW-3G SQ | 8 | 64 | 5 | 26 |
| | iScream | 16 | 64 | 6 | 27 |
| | Safer | 128 | 92 | 7 | 30 |
| | $\mathrm{Skinny}_8$ | 64 | 128 | 6 | 47 |
| | CSS | 128 | 256 | 4 | 81 |

[2]Found in [Beierle and Leander 2022]

# Monomials

| | $n$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S$ | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $x^3$ | 6 | 7 | 9 | 11 | 16 | 19 | 25 | 28 | 36 | 39 | 55 | 51 |

# Monomials

| | $n$ | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S$ | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $x^3$ | 6 | 7 | 9 | 11 | 16 | 19 | 25 | 28 | 36 | 39 | 55 | 51 |
| $x^{2^n-2}$ | 7 | 7 | 10 | 12 | 18 | 19 | 34 | 30 | 66 | 41 | 130 | 50 |

**Proposition ([Zhang, Wang, Yu, Guo, and Cui 2023 ASIACRYPT])**

*For n even, the finite field inverse map can be linearized on $\geq 2^{n/2} + 2$ points, based on the decomposition:*

$$x^{2^n-2} = \left(x^{2^{n/2}+1}\right)^{2^{n/2}-2} \cdot x^{2^{n/2}}$$

# Super-Sboxes

| $n$ | Super-Sbox | Key | VecLin $\geq$ |
|---|---|---|---|
| 16 | Midori64 | (0,0,0,0) | 4096 |
| | Midori64 | (C,C,C,C) | 4096 |
| | Midori64 | (E,2,1,9) | 576 |
| | Midori64 | (0,A,0,F) | 1280 |
| | SKINNY64 | (0,0,0,0) | 1054 |
| | SKINNY64 | (9,8,2,0) | 1046 |
| | LED | (0,0,0,0) | 117 |
| | LED | (5,A,5,9) | 114 |
| | MISTY-FI | (0,0,0,0) | 50 |
| | MISTY-FI | (0,8,0,e) | 50 |
| 32 | AES | (00,00,00,00) | 60[3] |

[3] Ongoing work with Alex Biryukov and Luca Bonamino.

# Super-Sboxes

| $n$ | Super-Sbox | Key | VecLin $\geq$ |
|---|---|---|---|
| 16 | Midori64 | (0,0,0,0) | 4096 |
| | Midori64 | (C,C,C,C) | 4096 |
| | Midori64 | (E,2,1,9) | 576 |
| | Midori64 | (0,A,0,F) | 1280 |
| | SKINNY64 | (0,0,0,0) | 1054 |
| | SKINNY64 | (9,8,2,0) | 1046 |
| | LED | (0,0,0,0) | 117 |
| | LED | (5,A,5,9) | 114 |
| | MISTY-FI | (0,0,0,0) | 50 |
| | MISTY-FI | (0,8,0,e) | 50 |
| 32 | AES | (00,00,00,00) | $60$[3] |

involut.$\Rightarrow$ fixed points ($A = \mathrm{Id}$)

[3] Ongoing work with Alex Biryukov and Luca Bonamino.

# Super-Sboxes

| $n$ | Super-Sbox | Key | VecLin $\geq$ |
|---|---|---|---|
| 16 | Midori64 | (0,0,0,0) | 4096 |
| | Midori64 | (C,C,C,C) | 4096 |
| | Midori64 | (E,2,1,9) | 576 |
| | Midori64 | (0,A,0,F) | 1280 |
| | SKINNY64 | (0,0,0,0) | 1054 |
| | SKINNY64 | (9,8,2,0) | 1046 |
| | LED | (0,0,0,0) | 117 |
| | LED | (5,A,5,9) | 114 |
| | MISTY-FI | (0,0,0,0) | 50 |
| | MISTY-FI | (0,8,0,e) | 50 |
| 32 | AES | (00,00,00,00) | $60$[3] |

involut.$\Rightarrow$ fixed points ($A = \mathrm{Id}$)

still quite large

[3] Ongoing work with Alex Biryukov and Luca Bonamino.

# Super-Sboxes

| $n$ | Super-Sbox | Key | VecLin $\geq$ | |
|---|---|---|---|---|
| 16 | Midori64 | (0,0,0,0) | 4096 | involut.$\Rightarrow$ fixed points ($A = \mathrm{Id}$) |
| | Midori64 | (C,C,C,C) | 4096 | |
| | Midori64 | (E,2,1,9) | 576 | still quite large |
| | Midori64 | (0,A,0,F) | 1280 | |
| | SKINNY64 | (0,0,0,0) | 1054 | weak linear layer |
| | SKINNY64 | (9,8,2,0) | 1046 | |
| | LED | (0,0,0,0) | 117 | |
| | LED | (5,A,5,9) | 114 | |
| | MISTY-FI | (0,0,0,0) | 50 | |
| | MISTY-FI | (0,8,0,e) | 50 | |
| 32 | AES | (00,00,00,00) | $60$[3] | |

[3] Ongoing work with Alex Biryukov and Luca Bonamino.

# Super-Sboxes

| $n$ | Super-Sbox | Key | VecLin $\geq$ | |
|---|---|---|---|---|
| 16 | Midori64 | (0,0,0,0) | 4096 | involut.$\Rightarrow$ fixed points ($A = \mathrm{Id}$) |
| | Midori64 | (C,C,C,C) | 4096 | |
| | Midori64 | (E,2,1,9) | 576 | still quite large |
| | Midori64 | (0,A,0,F) | 1280 | |
| | SKINNY64 | (0,0,0,0) | 1054 | weak linear layer |
| | SKINNY64 | (9,8,2,0) | 1046 | |
| | LED | (0,0,0,0) | 117 | small but 2$\times$ random |
| | LED | (5,A,5,9) | 114 | |
| | MISTY-FI | (0,0,0,0) | 50 | |
| | MISTY-FI | (0,8,0,e) | 50 | |
| 32 | AES | (00,00,00,00) | $60$[3] | |

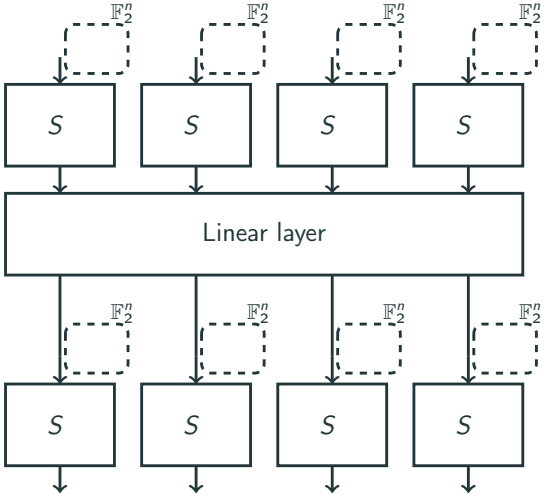
[3] Ongoing work with Alex Biryukov and Luca Bonamino.

# Super-Sboxes

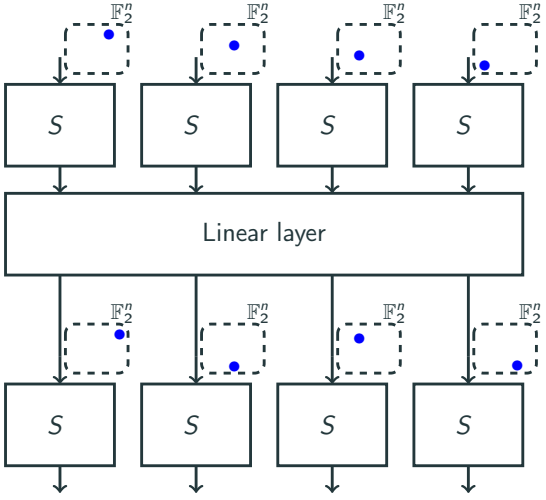
| $n$ | Super-Sbox | Key | VecLin $\geq$ | |
|---|---|---|---|---|
| 16 | Midori64 | (0,0,0,0) | 4096 | involut.$\Rightarrow$ fixed points ($A = \mathrm{Id}$) |
| | Midori64 | (C,C,C,C) | 4096 | |
| | Midori64 | (E,2,1,9) | 576 | still quite large |
| | Midori64 | (0,A,0,F) | 1280 | |
| | SKINNY64 | (0,0,0,0) | 1054 | weak linear layer |
| | SKINNY64 | (9,8,2,0) | 1046 | |
| | LED | (0,0,0,0) | 117 | small but 2$\times$ random |
| | LED | (5,A,5,9) | 114 | |
| | MISTY-FI | (0,0,0,0) | 50 | same as random |
| | MISTY-FI | (0,8,0,e) | 50 | |
| 32 | AES | (00,00,00,00) | 60[3] | |

[3] Ongoing work with Alex Biryukov and Luca Bonamino.

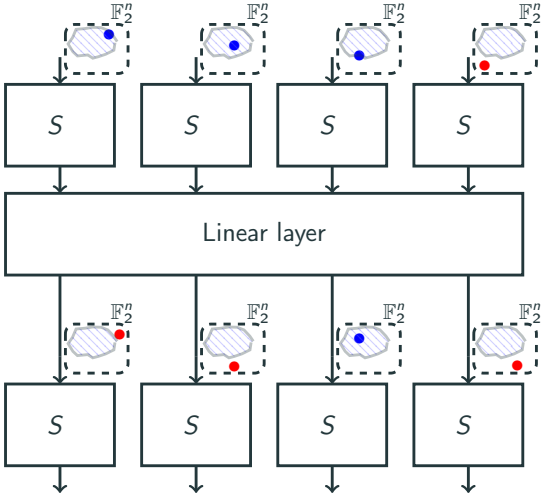
# Cryptanalysis options



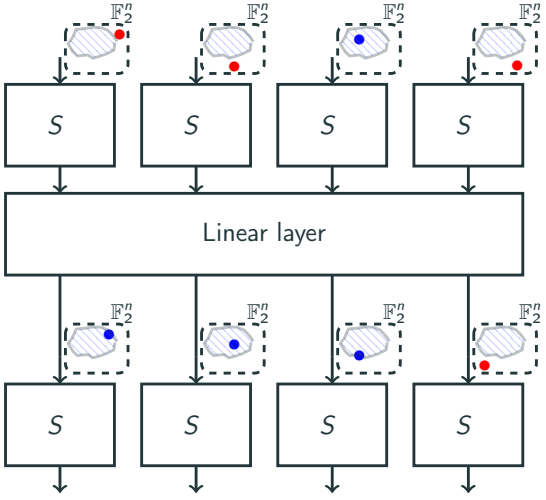
# Cryptanalysis options



# Cryptanalysis options

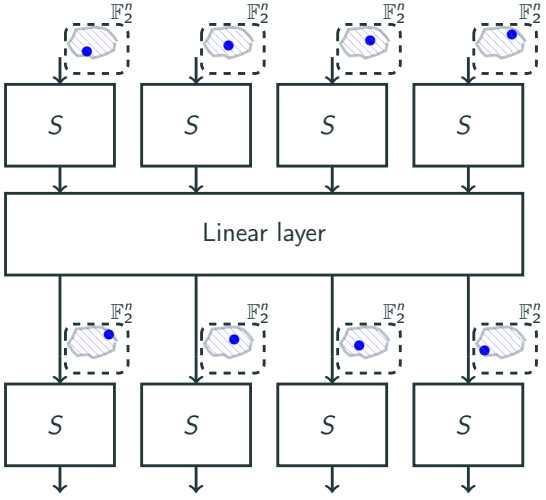


# Cryptanalysis options



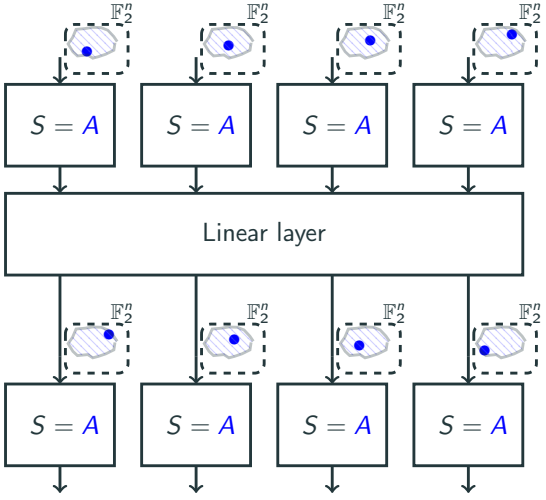

1. Randomize instance&solution

# Cryptanalysis options



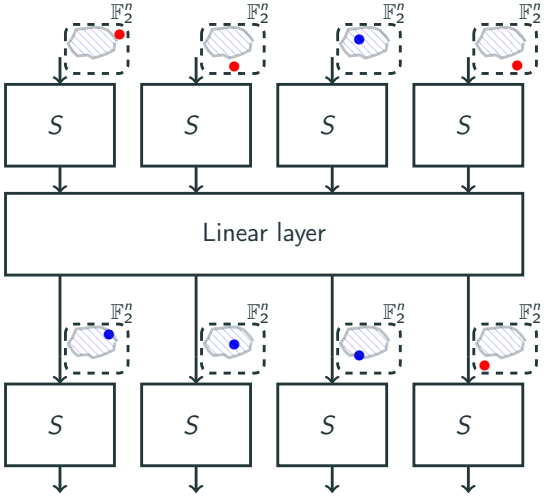

1. Randomize instance&solution

# Cryptanalysis options



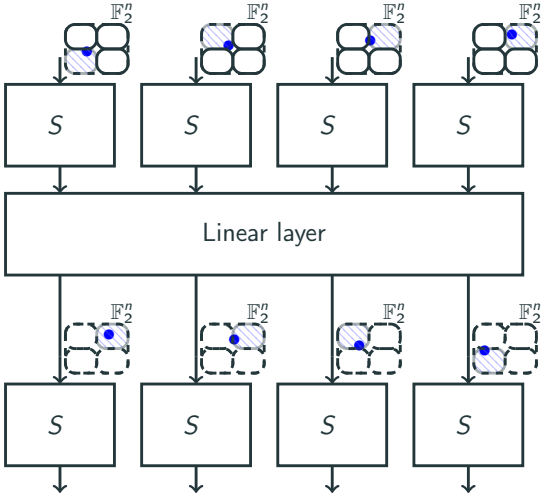

1. Randomize instance&solution

# Cryptanalysis options



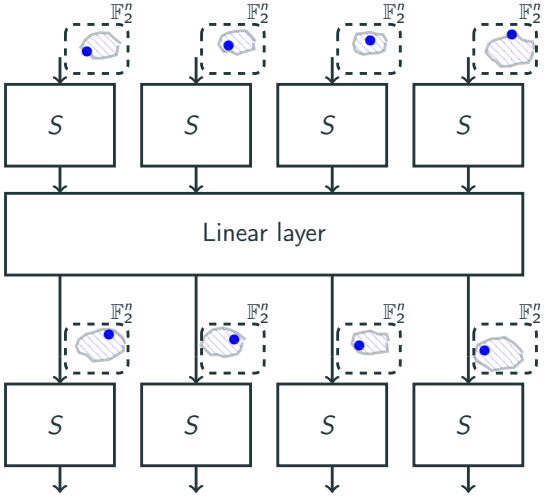

1. Randomize instance&solution
2. Randomize approximation

# Cryptanalysis options




1. Randomize instance&solution
2. Randomize approximation
   - *Partition* when available (deterministic)

# Cryptanalysis options




1. Randomize instance&solution
2. Randomize approximation
   - *Partition* when available (deterministic)
   - *Covering* can be more efficient (probabilistic)!

# Plan

# Cryptanalysis tool: probabilistic covering


| Approx. Weight | Size |
|---|---|
| | |

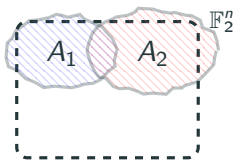
| Input | Pr[$x$ is covered by] $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
|---|---|---|---|---|---|
| | | | | | |

# Cryptanalysis tool: probabilistic covering


| Approx. | Weight | Size |
|---|---|---|
| $A_1$ | 0.18 | 115 |

| Input | Pr[$x$ is covered by] $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
|---|---|---|---|---|---|
| $0\ldots00$ | 0.18 | | | | 0.18 |
| $0\ldots01$ | 0.18 | | | | 0.18 |
| $0\ldots10$ | | | | | |
| $\vdots$ | | | | | |
| $1\ldots11$ | | | | | |

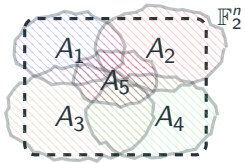
# Cryptanalysis tool: probabilistic covering




| Approx. | Weight | Size |
|---|---|---|
| $A_1$ | 0.18 | 115 |
| $A_2$ | 0.15 | 102 |

| Input | Pr[$x$ is covered by] $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
|---|---|---|---|---|---|
| 0...00 | 0.18 | | | | 0.18 |
| 0...01 | 0.18 | 0.15 | | | 0.33 |
| 0...10 | | 0.15 | | | 0.15 |
| ⋮ | | | | | |
| 1...11 | | 0.15 | | | 0.15 |

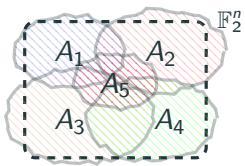
# Cryptanalysis tool: probabilistic covering




| Approx. | Weight | Size |
|---|---|---|
| $A_1$ | 0.18 | 115 |
| $A_2$ | 0.15 | 102 |
| $A_3$ | 0.23 | 120 |
| $A_4$ | 0.20 | 99 |
| $A_5$ | 0.24 | 109 |
| Avg. | | 109.56 |

| | Pr[$x$ is covered by] | | | | |
|---|---|---|---|---|---|
| Input | $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
| 0...00 | 0.18 | | | 0.24 | 0.42 |
| 0...01 | 0.18 | 0.15 | | | 0.33 |
| 0...10 | | 0.15 | | | 0.15 |
| $\vdots$ | | | | | |
| 1...11 | | 0.15 | | 0.24 | 0.39 |
| | | | | | $\rightarrow$(avg.$\lvert A_i \rvert$)/$2^n$ |

# Cryptanalysis tool: probabilistic covering

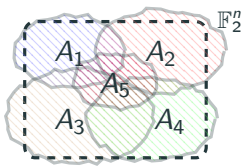



| Approx. | Weight | Size |
|---|---|---|
| $A_1$ | 0.18 | 115 |
| $A_2$ | 0.15 | 102 |
| $A_3$ | 0.23 | 120 |
| $A_4$ | 0.20 | 99 |
| $A_5$ | 0.24 | 109 |
| Avg. | | 109.56 |

| | Pr[$x$ is covered by] | | | | |
|---|---|---|---|---|---|
| Input | $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
| 0...00 | 0.18 | | | 0.24 | 0.42 |
| 0...01 | 0.18 | 0.15 | | | 0.33 |
| 0...10 | | 0.15 | | | 0.15 |
| $\vdots$ | | | | | |
| 1...11 | | 0.15 | | 0.24 | 0.39 |
| | | | | | $\rightarrow$(avg.$\lvert A_i \rvert$)/$2^n$ |

- Choose weights: maximize avg.$|A_i|$ and uniformity (Linear Programming)

# Cryptanalysis tool: probabilistic covering

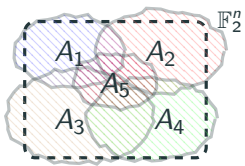



| Approx. | Weight | Size |
|---|---|---|
| $A_1$ | 0.18 | 115 |
| $A_2$ | 0.15 | 102 |
| $A_3$ | 0.23 | 120 |
| $A_4$ | 0.20 | 99 |
| $A_5$ | 0.24 | 109 |
| Avg. | | 109.56 |

| | Pr[$x$ is covered by] | | | | |
|---|---|---|---|---|---|
| Input | $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
| 0...00 | 0.18 | | | 0.24 | 0.42 |
| 0...01 | 0.18 | 0.15 | | | 0.33 |
| 0...10 | | 0.15 | | | 0.15 |
| $\vdots$ | | | | | |
| 1...11 | | 0.15 | | 0.24 | 0.39 |
| | | | | | $\rightarrow$(avg.$\lvert A_i \rvert$)$/2^n$ |

- Choose weights: maximize avg.$|A_i|$ and uniformity (Linear Programming)
- Effective approximation size 109.56 vs naive enumeration $2^n/5 = 51.2$

# Cryptanalysis tool: probabilistic covering

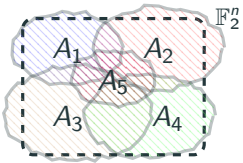



| Approx. | Weight | Size |
|---|---|---|
| $A_1$ | 0.18 | 115 |
| $A_2$ | 0.15 | 102 |
| $A_3$ | 0.23 | 120 |
| $A_4$ | 0.20 | 99 |
| $A_5$ | 0.24 | 109 |
| Avg. | | 109.56 |

| | Pr[$x$ is covered by] | | | | |
|---|---|---|---|---|---|
| Input | $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
| 0...00 | 0.18 | | | 0.24 | 0.42 |
| 0...01 | 0.18 | 0.15 | | | 0.33 |
| 0...10 | | 0.15 | | | 0.15 |
| $\vdots$ | | | | | |
| 1...11 | | 0.15 | | 0.24 | 0.39 |
| | | | | | $\rightarrow$(avg.$\lvert A_i \rvert$)$/2^n$ |

- Choose weights: maximize avg.$|A_i|$ and uniformity (Linear Programming)
- Effective approximation size 109.56 vs naive enumeration $2^n/5 = 51.2$
- Imbalance in probabilities can be mitigated by #sboxes

# Cryptanalysis tool: probabilistic covering

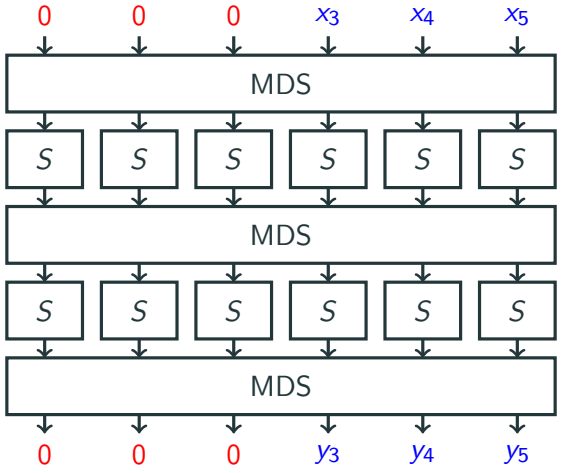



| Approx. | Weight | Size |
|---|---|---|
| $A_1$ | 0.18 | 115 |
| $A_2$ | 0.15 | 102 |
| $A_3$ | 0.23 | 120 |
| $A_4$ | 0.20 | 99 |
| $A_5$ | 0.24 | 109 |
| Avg. | | 109.56 |

| | Pr[$x$ is covered by] | | | | |
|---|---|---|---|---|---|
| Input | $A_1$ | $A_2$ | $\cdots$ | $A_5$ | Pr[cov.] |
| 0...00 | 0.18 | | | 0.24 | 0.42 |
| 0...01 | 0.18 | 0.15 | | | 0.33 |
| 0...10 | | 0.15 | | | 0.15 |
| $\vdots$ | | | | | |
| 1...11 | | 0.15 | | 0.24 | 0.39 |
| | | | | | $\rightarrow$(avg.$\lvert A_i \rvert$)$/2^n$ |

- Choose weights: maximize avg.$|A_i|$ and uniformity (Linear Programming)
- Effective approximation size 109.56 vs naive enumeration $2^n/5 = 51.2$
- Imbalance in probabilities can be mitigated by #sboxes
- Quadratic S-boxes: perfect covering using derivative shifts!

# Application to CICO problem



# Plan

# Application to CICO problem

**Table:** Lower bound $g_{LB}$ on the approximation size $g$ needed for linearization attack

**Time complexity:** factor $t^{-\#\text{sboxes}}$ over exhaustive search, where $g = t \cdot g_{LB}$

| Rounds | $g_{LB}(n)$ |
|---|---|
| 1 | $2^{n/2} = \sqrt{2^n}$ |
| 2 | $2^{3n/4}$ |
| 3 | $2^{5n/6}$ |
| 4 | $2^{7n/8}$ |
| $R$ | $2^{n\left(1-\frac{1}{2R}\right)}$ |

# Application to CICO problem

**Table:** Lower bound $g_{LB}$ on the approximation size $g$ needed for linearization attack

**Time complexity:** factor $t^{-\#\text{sboxes}}$ over exhaustive search, where $g = t \cdot g_{LB}$

| Rounds | $g_{LB}(n)$ | $n = 4$ | $n = 8$ | $n = 16$ | $n = 32$ | $n = 64$ |
|---|---|---|---|---|---|---|
| 1 | $2^{n/2}$ | 4 | 16 | $2^8$ | $2^{16}$ | $2^{32}$ |
| 2 | $2^{3n/4}$ | 8 | 64 | $2^{12}$ | $2^{24}$ | $2^{48}$ |
| 3 | $2^{5n/6}$ | 10.1 | 102 | $2^{13.3}$ | $2^{26.7}$ | $2^{53.3}$ |
| 4 | $2^{7n/8}$ | 11.3 | 128 | $2^{14}$ | $2^{28}$ | $2^{56}$ |
| $R$ | $2^{n\left(1-\frac{1}{2R}\right)}$ | | | | | |

# Application to CICO problem

**Table:** Lower bound $g_{LB}$ on the approximation size $g$ needed for linearization attack

**Time complexity:** factor $t^{-\#\text{sboxes}}$ over exhaustive search, where $g = t \cdot g_{LB}$

| Rounds | $g_{LB}(n)$ | $n = 4$ | $n = 8$ | $n = 16$ | $n = 32$ | $n = 64$ |
|---|---|---|---|---|---|---|
| 1 | $2^{n/2}$ | 4 | 16 | $2^8$ | $2^{16}$ | $2^{32}$ |
| 2 | $2^{3n/4}$ | 8 | 64 | $2^{12}$ | $2^{24}$ | $2^{48}$ |
| 3 | $2^{5n/6}$ | 10.1 | 102 | $2^{13.3}$ | $2^{26.7}$ | $2^{53.3}$ |
| 4 | $2^{7n/8}$ | 11.3 | 128 | $2^{14}$ | $2^{28}$ | $2^{56}$ |
| $R$ | $2^{n\left(1-\frac{1}{2R}\right)}$ | | | | | |

Example: $\text{SKINNY}_8$ S-box: $16 (R = 1) < \ g = 47 \ < 64 (R = 2)$

# Application to CICO problem

**Table:** Lower bound $g_{LB}$ on the approximation size $g$ needed for linearization attack

**Time complexity:** factor $t^{-\#\text{sboxes}}$ over exhaustive search, where $g = t \cdot g_{LB}$

| Rounds | $g_{LB}(n)$ | $n = 4$ | $n = 8$ | $n = 16$ | $n = 32$ | $n = 64$ |
|---|---|---|---|---|---|---|
| 1 | $2^{n/2}$ | 4 | 16 | $2^8$ | $2^{16}$ | $2^{32}$ |
| 2 | $2^{3n/4}$ | 8 | 64 | $2^{12}$ | $2^{24}$ | $2^{48}$ |
| 3 | $2^{5n/6}$ | 10.1 | 102 | $2^{13.3}$ | $2^{26.7}$ | $2^{53.3}$ |
| 4 | $2^{7n/8}$ | 11.3 | 128 | $2^{14}$ | $2^{28}$ | $2^{56}$ |
| $R$ | $2^{n\left(1-\frac{1}{2R}\right)}$ | | | | | |

Example: $\text{SKINNY}_8$ S-box: $16(R = 1) < g = 47 < 64(R = 2)$

Experimentally verified: $2^{15.3}$ complexity on $\text{CICO}_{3,3}(1\text{R})$ vs $2^{24}$ exhaustive search

# Plan

# Conclusions

**On linearizations of S-boxes:**

- new powerful algorithms (heuristic or exhaustive)
- results for crypto S-boxes, monomials, SuperSboxes
- covering technique and generic CICO linearization

# Conclusions

**On linearizations of S-boxes:**

- new powerful algorithms (heuristic or exhaustive)
- results for crypto S-boxes, monomials, SuperSboxes
- covering technique and generic CICO linearization

**Open questions**

- understanding monomial approximations
- more concrete cryptanalysis ideas

# References i


- Banik, Subhadeep, Khashayar Barooti, F. Betül Durak, and Serge Vaudenay (2020). **"Cryptanalysis of LowMC instances using single plaintext/ciphertext pair".** In: *IACR Trans. Symm. Cryptol.* 2020.4, pp. 130–146. issn: 2519-173X. doi: 10.46586/tosc.v2020.i4.130-146.
- Beierle, Christof and Gregor Leander (2022). **"New Instances of Quadratic APN Functions".** In: *IEEE Transactions on Information Theory* 68.1, pp. 670–678. doi: 10.1109/TIT.2021.3120698.
- Carlet, Claude (2021). **"Bounds on the Nonlinearity of Differentially Uniform Functions by Means of Their Image Set Size, and on Their Distance to Affine Functions".** In: *IEEE Transactions on Information Theory* 67.12, pp. 8325–8334. doi: 10.1109/TIT.2021.3114958.

- Chen, Lusheng and Fangwei Fu (2001). **"ON THE NONLINEARITY OF MULTI-OUTPUT BOOLEAN FUNCTIONS".** In: *Acta Scientificiarum Naturalium University Nankaiensis* 34.4, pp. 28–33. url: https://www.alljournals.cn/view_abstract.aspx?pcid=01BA20E8BA813E1908F3698710BBFEFEE816345F465FEBA5&cid=96E6E851B5104576C2DD9FC1FBCB69EF&jid=2F663905325EBF0C8638CFC155B7BC91&aid=A4F31984B5D18D95&yid=14E7EF987E4155E6.
- Chen, Shiyao, Jian Guo, Eik List, Danping Shi, and Tianyu Zhang (May 2024). **"Diving Deep into the Preimage Security of AES-Like Hashing".** In: *EUROCRYPT 2024, Part I*. Ed. by Marc Joye and Gregor Leander. Vol. 14651. LNCS. Springer, Cham, pp. 398–426. doi: 10.1007/978-3-031-58716-0_14.

# References iii


- Courtois, Nicolas T., Frédéric Amiel, and Alexandre Bonnard de Fonvillars (2024). ***On Maximum Size Simultaneous Linear Approximations in Ascon and Keccak and Related Translation and Differential Properties.*** Cryptology ePrint Archive, Report 2024/802. url: https://eprint.iacr.org/2024/802.
- Liu, Jian, Sihem Mesnager, and Lusheng Chen (2017). **"On the nonlinearity of S-boxes and linear codes".** In: *Cryptogr. Commun.* 9.3, pp. 345–361. doi: 10.1007/S12095-015-0176-Z.
- Nagy, Gábor P. (2025a). **"On the minimum Hamming distance between vectorial Boolean and affine functions".** In: *Cryptography and Communications* 17.6, pp. 1703–1720. issn: 1936-2455. doi: 10.1007/s12095-025-00808-4. url: https://doi.org/10.1007/s12095-025-00808-4.

# References iv


- Nagy, Gábor P. (2025b). **"Sidon sets, thin sets, and the nonlinearity of vectorial Boolean functions".** In: *Journal of Combinatorial Theory, Series A* 212, p. 106001. issn: 0097-3165. doi: https://doi.org/10.1016/j.jcta.2024.106001. url: https://www.sciencedirect.com/science/article/pii/S0097316524001407.
- Qiao, Kexin, Ling Song, Meicheng Liu, and Jian Guo (2017). **"New Collision Attacks on Round-Reduced Keccak".** In: *EUROCRYPT 2017, Part III*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10212. LNCS. Springer, Cham, pp. 216–243. doi: 10.1007/978-3-319-56617-7_8.
- Ryabov, V.G. (Oct. 2023). **"Nonlinearity of APN functions: comparative analysis and estimates".** In: *[Prikladnaya Diskretnaya Matematika]* 61, pp. 15–27. doi: 10.17223/20710410/61/2. url: https://www.mathnet.ru/rus/pdm810.

- Ryabov, V.G. (2024a). **"Distance between vectorial Boolean functions and affine analogues (following the Eighth International Olympiad in Cryptography)".** In: *[Mat. Vopr. Kriptogr.]* 15.1, pp. 127–142. doi: 10.4213/mvk465. url: https://www.mathnet.ru/eng/mvk466.
- — (2024b). **"Nonlinearity of vectorial functions over finite fields".** In: *[Diskr. Mat.]* 36.2, pp. 50–70. doi: 10.4213/dm1822. url: https://www.mathnet.ru/eng/dm1822.
- Zhang, Kaiyi, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui (Dec. 2023). **"Algebraic Attacks on Round-Reduced Rain and Full AIM-III".** In: *ASIACRYPT 2023, Part III*. Ed. by Jian Guo and Ron Steinfeld. Vol. 14440. LNCS. Springer, Singapore, pp. 285–310. doi: 10.1007/978-981-99-8727-6_10.

# Greedy extension algorithm (DDT)

$\mathsf{DDT}_{\mathsf{sol}} \leftarrow \{(\delta_x, \delta_y) \mapsto \{x \mid S(x) + S(x + \delta_x) = \delta_y\}\}$
$\mathsf{LS}_x = \{0\} \subseteq \mathbb{F}_2^n$
$\mathsf{LS} = \{(0, 0)\} \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^m$
cliques $\leftarrow \{\{x\} : x \in \mathbb{F}_2^n\}$
**for** $d \in \{0, \ldots, n-1\}$ **do**
    $(a, a') \leftarrow$ representatives of two biggest cliques from different cosets ($(a + a') \notin \mathsf{LS}_x$)
    $\delta_x \leftarrow a + a'$
    $\delta_y \leftarrow S(a) + S(a')$
    **for all** $v \in \mathsf{LS}$ **do**
        $(\tilde{\delta_x}, \tilde{\delta_y}) \leftarrow v + (\delta_x, \delta_y)$
        **for all** $x \in \mathsf{DDT}_{\mathsf{sol}}[\tilde{\delta_x}, \tilde{\delta_y}]$ **do**
            Merge cliques containing $x$ and $x + \tilde{\delta_x}$
        **end for**
    **end for**
    $\mathsf{LS} \leftarrow \mathsf{LS} \cup ((\delta_x, \delta_y) + \mathsf{LS})$
    $\mathsf{LS}_x \leftarrow \mathsf{LS}_x \cup (\delta_x + \mathsf{LS}_x)$
**end for**
$A \leftarrow$ affine map interpolated from the largest clique
**return** $A$

# Greedy extension algorithm (DDT)

$\mathsf{DDT}_{\mathsf{sol}} \leftarrow \{(\delta_x, \delta_y) \mapsto \{x \mid S(x) + S(x + \delta_x) = \delta_y\}\}$
$\mathsf{LS}_x = \{0\} \subseteq \mathbb{F}_2^n$
$\mathsf{LS} = \{(0, 0)\} \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^m$
cliques $\leftarrow \{\{x\} : x \in \mathbb{F}_2^n\}$
**for** $d \in \{0, \ldots, n-1\}$ **do**
    $(a, a') \leftarrow$ representatives of two biggest cliques from different cosets $((a + a') \notin \mathsf{LS}_x)$
    $\delta_x \leftarrow a + a'$
    $\delta_y \leftarrow S(a) + S(a')$
    **for all** $v \in \mathsf{LS}$ **do**  $|\mathsf{LS}| = 1, 2, 4, 8, \ldots, 2^d, \ldots, 2^{n-1}$
        $(\tilde{\delta_x}, \tilde{\delta_y}) \leftarrow v + (\delta_x, \delta_y)$
        **for all** $x \in \mathsf{DDT}_{\mathsf{sol}}[\tilde{\delta_x}, \tilde{\delta_y}]$ **do**
            Merge cliques containing $x$ and $x + \tilde{\delta_x}$
        **end for**
    **end for**
    $\mathsf{LS} \leftarrow \mathsf{LS} \cup ((\delta_x, \delta_y) + \mathsf{LS})$
    $\mathsf{LS}_x \leftarrow \mathsf{LS}_x \cup (\delta_x + \mathsf{LS}_x)$
**end for**
$A \leftarrow$ affine map interpolated from the largest clique
**return** $A$

# Greedy extension algorithm (DDT)

$\mathsf{DDT}_{\mathsf{sol}} \leftarrow \{(\delta_x, \delta_y) \mapsto \{x \mid S(x) + S(x + \delta_x) = \delta_y\}\}$
$\mathsf{LS}_x = \{0\} \subseteq \mathbb{F}_2^n$
$\mathsf{LS} = \{(0, 0)\} \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^m$
cliques $\leftarrow \{\{x\} : x \in \mathbb{F}_2^n\}$
**for** $d \in \{0, \ldots, n-1\}$ **do**
    $(a, a') \leftarrow$ representatives of two biggest cliques from different cosets $((a + a') \notin \mathsf{LS}_x)$
    $\delta_x \leftarrow a + a'$
    $\delta_y \leftarrow S(a) + S(a')$
    **for all** $v \in \mathsf{LS}$ **do**  $|\mathsf{LS}| = 1, 2, 4, 8, \ldots, 2^d, \ldots, 2^{n-1}$
        $(\tilde{\delta_x}, \tilde{\delta_y}) \leftarrow v + (\delta_x, \delta_y)$
        **for all** $x \in \mathsf{DDT}_{\mathsf{sol}}[\tilde{\delta_x}, \tilde{\delta_y}]$ **do**  $\mathcal{O}(1)$ expected values
            Merge cliques containing $x$ and $x + \tilde{\delta_x}$
        **end for**
    **end for**
    $\mathsf{LS} \leftarrow \mathsf{LS} \cup ((\delta_x, \delta_y) + \mathsf{LS})$
    $\mathsf{LS}_x \leftarrow \mathsf{LS}_x \cup (\delta_x + \mathsf{LS}_x)$
**end for**
$A \leftarrow$ affine map interpolated from the largest clique
**return** $A$

# Greedy extension algorithm (DDT)

$\mathsf{DDT}_{\mathsf{sol}} \leftarrow \{(\delta_x, \delta_y) \mapsto \{x \mid S(x) + S(x + \delta_x) = \delta_y\}\}$
$\mathsf{LS}_x = \{0\} \subseteq \mathbb{F}_2^n$
$\mathsf{LS} = \{(0, 0)\} \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^m$
cliques $\leftarrow \{\{x\} : x \in \mathbb{F}_2^n\}$
**for** $d \in \{0, \ldots, n-1\}$ **do**
    $(a, a') \leftarrow$ representatives of two biggest cliques from different cosets $((a + a') \notin \mathsf{LS}_x)$
    $\delta_x \leftarrow a + a'$
    $\delta_y \leftarrow S(a) + S(a')$
    **for all** $v \in \mathsf{LS}$ **do**  $|\mathsf{LS}| = 1, 2, 4, 8, \ldots, 2^d, \ldots, 2^{n-1}$
        $(\tilde{\delta_x}, \tilde{\delta_y}) \leftarrow v + (\delta_x, \delta_y)$
        **for all** $x \in \mathsf{DDT}_{\mathsf{sol}}[\tilde{\delta_x}, \tilde{\delta_y}]$ **do**  $\mathcal{O}(1)$ expected values
            Merge cliques containing $x$ and $x + \tilde{\delta_x}$  using union-find data structure
        **end for**
    **end for**
    $\mathsf{LS} \leftarrow \mathsf{LS} \cup ((\delta_x, \delta_y) + \mathsf{LS})$
    $\mathsf{LS}_x \leftarrow \mathsf{LS}_x \cup (\delta_x + \mathsf{LS}_x)$
**end for**
$A \leftarrow$ affine map interpolated from the largest clique
**return** $A$

# Greedy extension algorithm (DDT)

$\mathsf{DDT}_{\mathsf{sol}} \leftarrow \{(\delta_x, \delta_y) \mapsto \{x \mid S(x) + S(x + \delta_x) = \delta_y\}\}$
$\mathsf{LS}_x = \{0\} \subseteq \mathbb{F}_2^n$
$\mathsf{LS} = \{(0, 0)\} \subseteq \mathbb{F}_2^n \times \mathbb{F}_2^m$
cliques $\leftarrow \{\{x\} : x \in \mathbb{F}_2^n\}$
**for** $d \in \{0, \ldots, n-1\}$ **do**
    $(a, a') \leftarrow$ representatives of two biggest cliques from different cosets $((a + a') \notin \mathsf{LS}_x)$  using heap
    $\delta_x \leftarrow a + a'$
    $\delta_y \leftarrow S(a) + S(a')$
    **for all** $v \in \mathsf{LS}$ **do**  $|\mathsf{LS}| = 1, 2, 4, 8, \ldots, 2^d, \ldots, 2^{n-1}$
        $(\tilde{\delta_x}, \tilde{\delta_y}) \leftarrow v + (\delta_x, \delta_y)$
        **for all** $x \in \mathsf{DDT}_{\mathsf{sol}}[\tilde{\delta_x}, \tilde{\delta_y}]$ **do**  $\mathcal{O}(1)$ expected values
            Merge cliques containing $x$ and $x + \tilde{\delta_x}$  using union-find data structure
        **end for**
    **end for**
    $\mathsf{LS} \leftarrow \mathsf{LS} \cup ((\delta_x, \delta_y) + \mathsf{LS})$
    $\mathsf{LS}_x \leftarrow \mathsf{LS}_x \cup (\delta_x + \mathsf{LS}_x)$
**end for**
$A \leftarrow$ affine map interpolated from the largest clique
**return** $A$

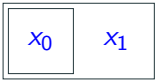
# Extension framework (DDT)

domain of $A$, $\subseteq \mathbb{F}_2^n$


choose $x_0$, set $A(x_0) = S(x_0)$

# Extension framework (DDT)

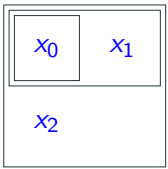
domain of $A$, $\subseteq \mathbb{F}_2^n$




choose $x_0$, set $A(x_0) = S(x_0)$
choose $x_1$, set $A(x_1) = S(x_1)$

# Extension framework (DDT)

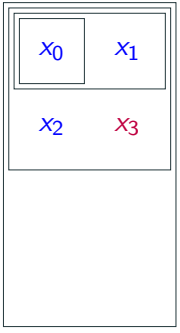
domain of $A$, $\subseteq \mathbb{F}_2^n$




choose $x_0$, set $A(x_0) = S(x_0)$
choose $x_1$, set $A(x_1) = S(x_1)$
choose $x_2$, set $A(x_2) = S(x_2)$

# Extension framework (DDT)

domain of $A$, $\subseteq \mathbb{F}_2^n$

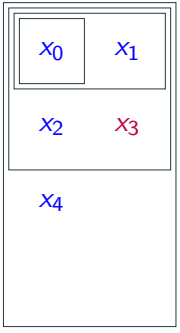



choose $x_0$, set $A(x_0) = S(x_0)$
choose $x_1$, set $A(x_1) = S(x_1)$
choose $x_2$, set $A(x_2) = S(x_2)$

$$A(x_3) = A(x_0 + x_1 + x_2)$$
$$= A(x_0) + A(x_1) + A(x_2) \stackrel{?}{=} S(x_3)$$

# Extension framework (DDT)

domain of $A$, $\subseteq \mathbb{F}_2^n$




choose $x_0$, set $A(x_0) = S(x_0)$

choose $x_1$, set $A(x_1) = S(x_1)$

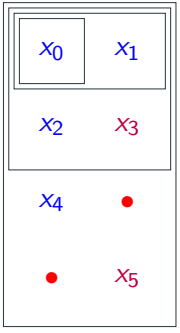
choose $x_2$, set $A(x_2) = S(x_2)$

$$A(x_3) = A(x_0 + x_1 + x_2)$$
$$= A(x_0) + A(x_1) + A(x_2) \stackrel{?}{=} S(x_3)$$

choose $x_4$, set $A(x_4) = S(x_4)$

# Extension framework (DDT)

domain of $A$, $\subseteq \mathbb{F}_2^n$

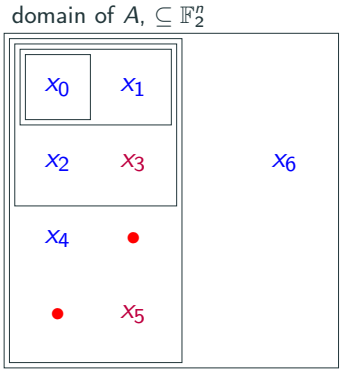



choose $x_0$, set $A(x_0) = S(x_0)$
choose $x_1$, set $A(x_1) = S(x_1)$
choose $x_2$, set $A(x_2) = S(x_2)$

$$A(x_3) = A(x_0 + x_1 + x_2)$$
$$= A(x_0) + A(x_1) + A(x_2) \stackrel{?}{=} S(x_3)$$

choose $x_4$, set $A(x_4) = S(x_4)$
$A(x_5) \stackrel{?}{=} S(x_5)$ for free

# Extension framework (DDT)




choose $x_0$, set $A(x_0) = S(x_0)$

choose $x_1$, set $A(x_1) = S(x_1)$

choose $x_2$, set $A(x_2) = S(x_2)$

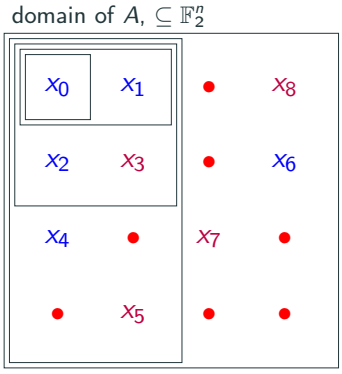
$$A(x_3) = A(x_0 + x_1 + x_2)$$
$$= A(x_0) + A(x_1) + A(x_2) \stackrel{?}{=} S(x_3)$$

choose $x_4$, set $A(x_4) = S(x_4)$

$A(x_5) \stackrel{?}{=} S(x_5)$ for free

choose $x_6$, set $A(x_6) = S(x_6)$

# Extension framework (DDT)




choose $x_0$, set $A(x_0) = S(x_0)$

choose $x_1$, set $A(x_1) = S(x_1)$

choose $x_2$, set $A(x_2) = S(x_2)$

$$A(x_3) = A(x_0 + x_1 + x_2)$$
$$= A(x_0) + A(x_1) + A(x_2) \stackrel{?}{=} S(x_3)$$

choose $x_4$, set $A(x_4) = S(x_4)$

$A(x_5) \stackrel{?}{=} S(x_5)$ for free

choose $x_6$, set $A(x_6) = S(x_6)$

$A(x_7) \stackrel{?}{=} S(x_7)$ for free

$A(x_8) \stackrel{?}{=} S(x_8)$ for free