Reconstructing an S-box from its Difference Distribution Table

Orr Dunkelman, Senyang Huang

Department of Computer Science, University of Haifa, Haifa, Israel

2020 . 11 . 5

Background and Motivation

Difference Distribution Table (DDT) of an S-box S

Let S be a Boolean function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$

$$\delta(a, b) = \left|\{z \in \mathbb{F}_2^n \middle| S(z \oplus a) \oplus S(z) = b\}\right|.$$

- S-box$\rightarrow$ DDT: Easy
- DDT$\rightarrow$ S-box: Difficult
- The ability to recover the S-box from the DDT of a secret S-box can be used in cryptanalytic attacks.
- Boura et al. [BCJS19] proposed a straightforward guess and determine (GD) algorithm to solve the problem.
- Using the well established relation between the DDT and the linear approximation table (LAT), we devise a new approach to reconstruct an S-box from its DDT.

Linear Approximation Table (LAT) of an S-box S

$$
\begin{aligned}
\lambda(a,b) &= \left|\{x \in \mathbb{F}_2^n \middle| a \cdot x \oplus b \cdot S(x) = 0\}\right| - 2^{n-1} \\
&= \frac{1}{2} \sum_{x \in \mathbb{F}_2^n} (-1)^{a \cdot x \oplus b \cdot S(x)}
\end{aligned}
$$

Walsh-Hadamard Transform

Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{R}$ be a function. $\hat{f}$ denotes its *Walsh-Hadamard transform*, which is equal to:

$$\hat{f}(a, b) = \sum_{x,y} f(x, y)(-1)^{a \cdot x \oplus b \cdot y},$$

where $a \in \mathbb{F}_2^n$, $b \in \mathbb{F}_2^m$ and $a \cdot x$ and $b \cdot y$ are the inner product over the domains $\mathbb{F}_2^n$ and $\mathbb{F}_2^m$, respectively.

Links between an S-box, its DDT and LAT

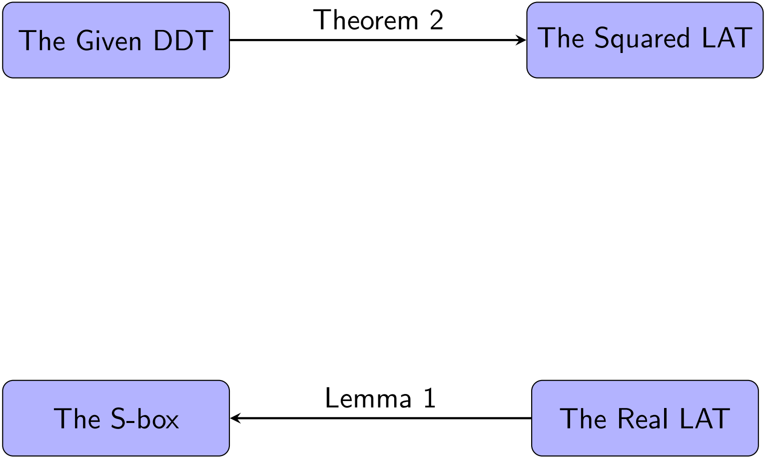
Lemma 1.
([CV95, Lemma 2]) For $(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^m$, *let* $\theta(a, b)$ *be the characteristic function of* S, *i.e.,* $\theta(a, b) = 1$ *if and only if* $S(a) = b$; *otherwise* $\theta(a, b) = 0$. *Then,*

$$\hat{\lambda}(a, b) = 2^{m+n-1}\theta(a, b).$$

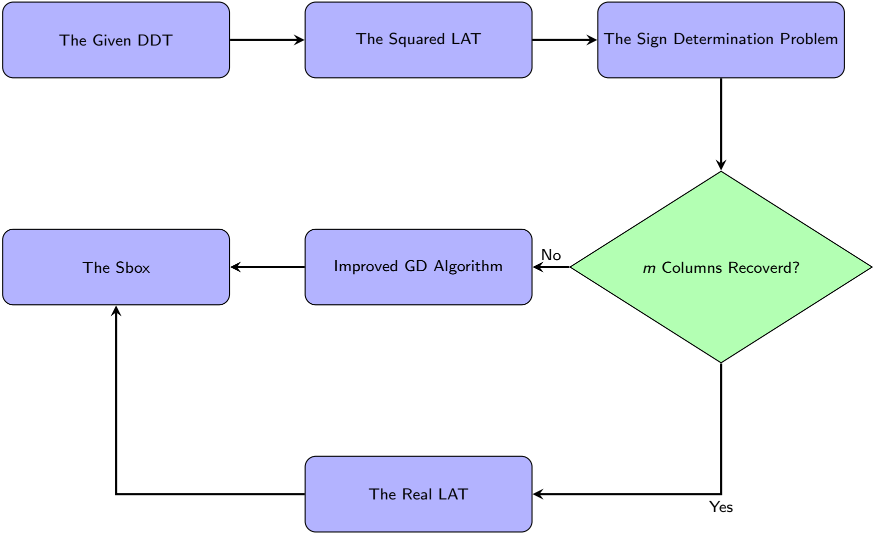
Theorem 2.
([BN13, CV95, DGV95]) For all $(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^m$,

1. $\hat{\delta}(a, b) = 4\lambda^2(a, b)$,
2. $4\widehat{\lambda^2}(a, b) = 2^{m+n}\delta(a, b)$,

where $\widehat{\lambda^2}(a, b)$ *is the Walsh-Hadamard transform of* $\lambda^2(a, b)$, *the squared LAT.*


The Given DDT
Theorem 2
The Squared LAT
The S-box
Lemma 1
The Real LAT

$$
\begin{tikzcd}
\text{The Given DDT} \arrow[r, "\text{Theorem 2}"] & \text{The Squared LAT} \arrow[d, dash, dashed] \\
\text{The S-box} & \text{The Real LAT} \arrow[l, "\text{Lemma 1}"']
\end{tikzcd}
$$


The Given DDT
The Squared LAT
The Sign Determination Problem
m Columns Recoverd?
No
Improved GD Algorithm
The Sbox
Yes
The Real LAT

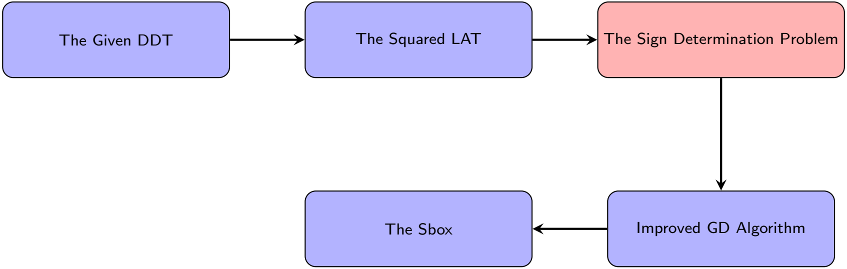
The Sign Determination Problem

Definition 3.
We define the † notion as follows:

$$\vec{v}^{\dagger} = (|v_0|, \ldots, |v_{\ell-1}|)^T,$$

where $\vec{v} = (v_0, \ldots, v_{\ell-1})^T$ and $|\cdot|$ is the absolute value of a number.

Definition 4.
Given $\vec{\lambda}_b^{\dagger}$ where $1 \leq b < 2^m$, the *sign determination problem* of the b-th column in an LAT is the problem of recovering $\vec{\lambda}_b$ from $\vec{\lambda}_b^{\dagger}$, i.e., determining the signs of $\lambda(a, b), 0 \leq a < 2^n$.



- The Linear Relation between $\vec{\lambda}_b$ and $\vec{s}_b$
- Solving the System of Linear Equations $H_n\vec{x} = \vec{y}$
- Basic Algorithm
- Improved Algorithm

The Linear Relation between $\vec{\lambda}_b$ and $\vec{s}_b$

Theorem 5.

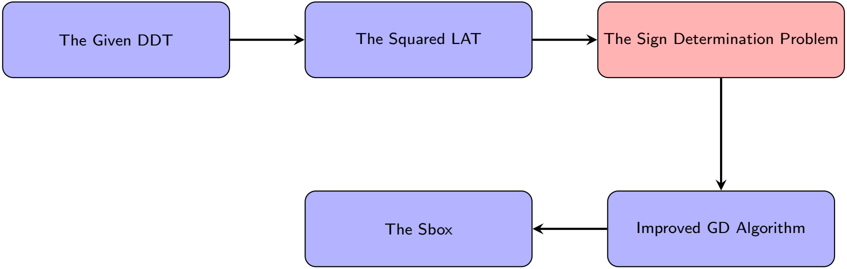
For any b-th column of the linear approximation table (for $0 \leq b < 2^m$), the following formula holds

$$H_n \vec{s}_b = 2\vec{\lambda}_b.$$

Definition 6.

Let $H_0 = (1)$, then the Hadamard matrix H_i can be represented as

$$H_i = \begin{pmatrix} H_{i-1} & H_{i-1} \\ H_{i-1} & -H_{i-1} \end{pmatrix}, i \geq 1.$$



- The Linear Relation between $\vec{\lambda}_b$ and $\vec{s}_b$
- Solving the System of Linear Equations $H_n\vec{x} = \vec{y}$
- Basic Algorithm
- Improved Algorithm

Solving the System of Linear Equations $H_n\vec{x} = \vec{y}$

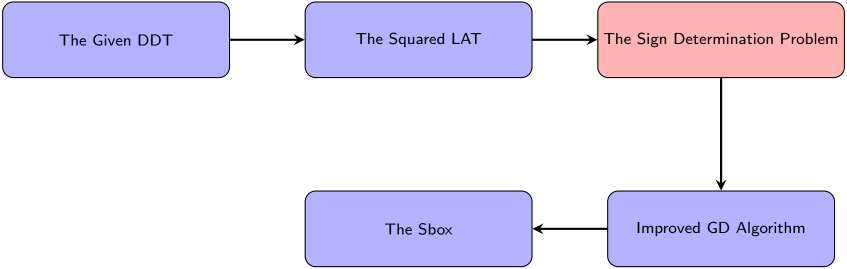
$$
(H_n, \vec{y}) = \left(\begin{array}{cc|c} H_{n-1} & H_{n-1} & \vec{y}^{[0,2^{n-1}-1]} \\ H_{n-1} & -H_{n-1} & \vec{y}^{[2^{n-1},2^n-1]} \end{array}\right)
$$

$$
\Rightarrow \left(\begin{array}{cc|c} H_{n-1} & 0 & (\vec{y}^{[0,2^{n-1}-1]} + \vec{y}^{[2^{n-1},2^n-1]})/2 \\ 0 & H_{n-1} & (\vec{y}^{[0,2^{n-1}-1]} - \vec{y}^{[2^{n-1},2^n-1]})/2 \end{array}\right)
$$

$$
\vdots
$$

$$
\Rightarrow \left(\begin{array}{ccc|c} H_0 & \cdots & 0 & \vec{x}[0] \\ & \ddots & & \vdots \\ 0 & \cdots & H_0 & \vec{x}[2^n - 1] \end{array}\right).
$$

Apply the elementary transformation to the independent subproblems by n times.



- The Linear Relation between $\vec{\lambda}_b$ and $\vec{s}_b$
- Solving the System of Linear Equations $H_n\vec{x} = \vec{y}$
- Basic Algorithm
- Improved Algorithm

Basic Algorithm

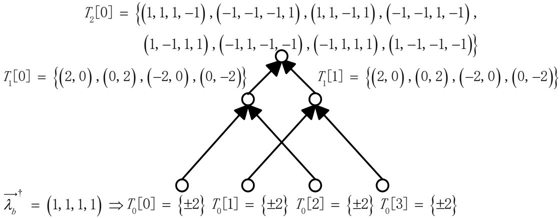
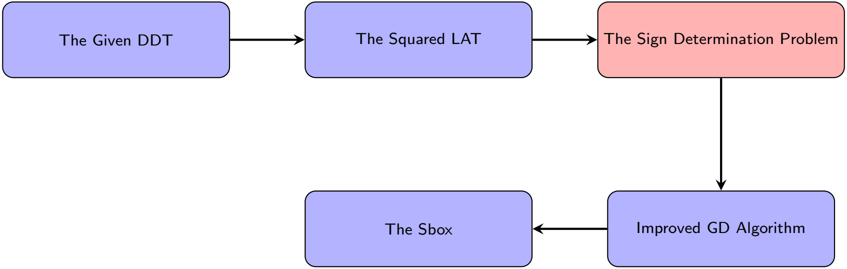


Figure 1: The Tree Structure for $n = 2$

- Apply the idea of solving the system of linear equations $H_n\vec{x} = \vec{y}$ to reduce the problem into two independent subproblems.
- The possible i-th constraint of subproblems is stored as a vector.
- A *full set* contains all the possible i-th constraints.

The size of the full sets in the intermediate layers grows so fast!



- The Linear Relation between $\vec{\lambda}_b$ and $\vec{s}_b$
- Solving the System of Linear Equations $H_n\vec{x} = \vec{y}$
- Basic Algorithm
- Improved Algorithm

Improved Algorithm

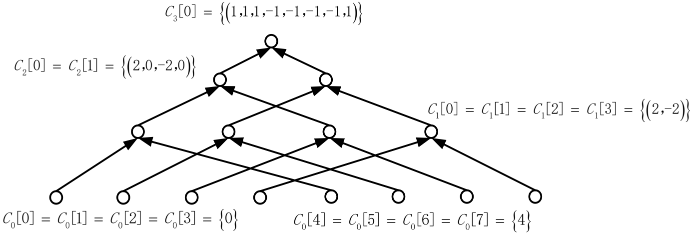


Figure 2: The Tree Structure for a Sign Determination Problem

- The symmetric structure of the full set
- Only record the representatives of the equivalence classes in the *compact set*.
- The compact representation reduces both time and memory complexity.

Algorithm 1: Constructing $M_{\vec{u},\vec{w}}$ from $\vec{u} \in C_\ell[i]$ and $\vec{w} \in C_\ell[i + 2^{n-\ell-1}]$

procedure $\textsc{ConstructSet}(\vec{u}, [\vec{w}]^+, J)$
    $M_{\vec{u},\vec{w}} = [\vec{w}]^+$
    for all integers $j \in J$ **do**
        Find $\pi^\ell_{j_0}, \ldots, \pi^\ell_{j_{p-1}}$ such that $\vec{u} = \pm\pi^\ell_{j_{p-1}} \circ \cdots \circ \pi^\ell_{j_0}(\vec{u})$
        for all the distinct vectors $\vec{e}, \vec{f}$ in $M_{\vec{u},\vec{w}}$ **do**
            if $\vec{e} = \pm\pi^\ell_{j_{p-1}} \circ \cdots \circ \pi^\ell_{j_0}(\vec{f})$ **then**
                $M_{\vec{u},\vec{w}} = M_{\vec{u},\vec{w}} \setminus \{\vec{f}\}$
            end if
        end for
    end for
    return $M_{\vec{u},\vec{w}}$
end procedure

In this way, the compact set $C_{\ell+1}[i]$ is indeed constructed by combining $\vec{u} \in C_\ell[i]$ and $\vec{v}$ in each $M_{\vec{u},\vec{w}}$.

Algorithm 2: Improved Algorithm for Solving the Sign Determination Problem

---

Input: $\vec{\lambda}_b^{\dagger}$;
Output: $F = \{\vec{u} | H_n \vec{u} = 2\vec{\lambda}_b, \vec{u}[0] = 1\}$
for each integer $i \in [0, 2^n - 1]$ **do**
    $C_0[i] = \{2\lambda^{\dagger}(i, b)\}$ ▷ Initialization
end for
$C_n[0] = \text{LAYER}(C_0, 0)$
Construct the full set $F_n[0]$ from $C_n[0]$.
return $F = \{\vec{u} | \vec{u} \in F_n[0], \vec{u}[0] = 1\}$.

procedure $\text{LAYER}(C_\ell, \ell)$;
    for each integer $i \in [0, 2^{n-\ell-1} - 1]$ **do**
        if there are no vectors in $C_\ell[i]$ or $C_\ell[i + 2^{n-\ell-1}]$ **then**
            return There exist no S-boxes corresponding to the given DDT!
        end if
        $C_{\ell+1}[i] = \emptyset$
        Randomly pick a vector from $C_\ell[i]$ and compute $J = \{j \big| C_\ell[i]$ is j-symmetric, $0 \le j < \ell\}$
        for each $\vec{w}$ in $C_\ell[i + 2^{n-\ell-1}]$ **do**
            for each $\vec{u}$ in $C_\ell[i]$ **do**
                $M = \text{CONSTRUCTSET}(\vec{u}, [\vec{w}]^+, J)$
                for each $\vec{v}$ in M **do**

```
        r⃗ = E_ℓ(u⃗, v⃗)
        if ℓ < n then
            if every entry in r⃗ is even and [−2^{n−ℓ−1}, 2^{n−ℓ−1}] then
                C_{ℓ+1}[i] = C_{ℓ+1}[i] ∪ {r⃗}
            else
                Discard r⃗
            end if
        else
            if every entry in r⃗ is 1 or −1 then          ▷ when ℓ = n
                C_n[i] = C_n[i] ∪ {r⃗}
            else
                Discard r⃗
            end if
        end if
      end for
    end for
  end for
end for
if ℓ < n then
    Layer(C_{ℓ+1}, ℓ + 1)
else
    return C_n[0]
end if
end procedure
```

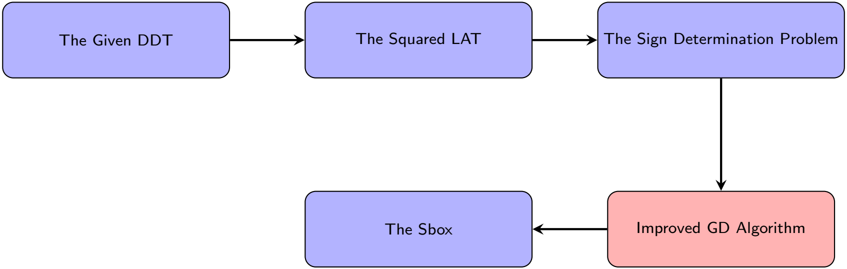
For some cases, the size of the compact sets still grows very fast!

Heuristic Threshold

- A threshold H on the number of internal vectors can be preset heuristically with respect to the accessible memory of the attacker.
- We call a column in the absolute LAT *good* if it can be recovered under the threshold H applying Algorithm 2; otherwise *bad*.
- According to our experiments with input size n between 8 and 14, the solutions for the good columns contains at most two equivalence classes.

Complexity Analysis of Algorithm 2

- The memory complexity of Algorithm 2 is $O(H \cdot n^2 2^n + n2^{2n})$ bits.
- The upper bound of the time complexity is $O(H^2 2^{3n})$.



- The Matching Phase for k Independent Good Columns
- Improved Guess-and-determine Algorithm

The Matching Phase for k Independent Good Columns

Definition 7.
The c_0-th, ..., the c_{k-1}-th columns in the LAT where $0 \leq c_0 < \cdots < c_{k-1} < 2^m$ are *independent columns* if the binary representations of $c_0, \ldots, c_{k-1}$ are linearly independent over $\mathbb{F}_2^m$.

Theorem 8.

For any $0 \leq b, c < 2^n$,

$$\vec{\lambda}_{b \oplus c} = 2H_n \cdot \vec{s}_b \odot \vec{s}_c,$$

where $\vec{s}_b \odot \vec{s}_c$ *is the Hadamard product of these vectors, i.e.*
$\vec{s}_b \odot \vec{s}_c = (\vec{s}_b[0] \cdot \vec{s}_c[0], \ldots, \vec{s}_b[2^n - 1] \cdot \vec{s}_c[2^n - 1])^T$.

Algorithm 3: The Matching Phase Given k Good Columns

Input: the index set of the good columns $C = \{c_0, \ldots, c_{k-1}\}$, the corresponding solution sets $V_0, \ldots, V_{k-1}$ and the squared LAT;

Output: $c_0 S(x), \ldots, c_{k-1} S(x)$;

for each $i \in [0, k-2]$ **do**

  if $i = 0$ **then**

    for each $\vec{u} \in \{\vec{u}_0, \ldots, \vec{u}_p\}$ and $\vec{v} \in V_1$ **do**

      $\vec{w} = 1/2H_n \cdot (\vec{u} \odot \vec{v})$

      if $\vec{w}^\dagger = \vec{\lambda}^\dagger_{c_i \oplus c_{i+1}}$ **then**

        $\vec{p}_0 = \vec{u}, \vec{p}_1 = \vec{v}$

        break $\triangleright$ this line is to be removed if the DDT-equivalence class is nontrivial.

      end if

    end for

  else

    for each $\vec{v} \in V_{i+1}$ **do**

      $\vec{w} = 1/2H_n \cdot (\vec{p}_i \odot \vec{v})$

      if $\vec{w}^\dagger = \vec{\lambda}^\dagger_{c_i \oplus c_{i+1}}$ **then**

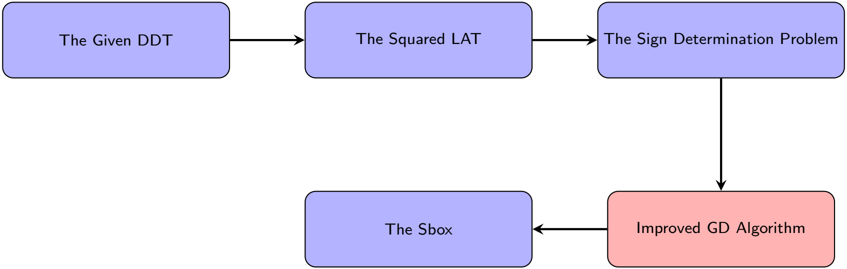
        $\vec{p}_{i+1} = \vec{v}$

        break $\triangleright$ this line is to be removed if the DDT-equivalence class is nontrivial.

      end if

    end for

```
    end if
end for
Deduce c_0 S(x), ..., c_{k-1} S(x) from p⃗_0, ..., p⃗_{k-1}
return c_0 S(x), ..., c_{k-1} S(x).
```



- The Matching Phase for k Independent Good Columns
- Improved Guess-and-determine Algorithm

Algorithm 4: Improved Guess-and-determine Algorithm

Input: $c_0, \ldots, c_{k-1}$, $c_0 S(x), \cdots, c_{k-1}S(x)$ and the given DDT
Output: one representative in the DDT-equivalence class
$\vec{s}$ is initialized as a vector of 2^m zeros.
$\textsc{ImprovedGD}(\vec{s}, 1)$
return $\vec{s}$
procedure $\textsc{ImprovedGD}(\vec{s}, i)$
  if $i < 2^m$ **then**
    $\mathcal{L} = \bigcap_{0 \leq j < i} \{x \oplus \vec{s}\,[j] \| x \in \mathcal{R}_{i \oplus j}, c_0 S(i) = c_0 \cdot x, \cdots, c_{k-1} S(i) = c_{k-1} \cdot x\}$
  else
    if the DDT of $\vec{s}$ matches the given DDT **then**
      return $\vec{s}$
    end if
  end if
  if $L \neq \emptyset$ **then**
    for each $x \in \mathcal{L}$ **do**
      $\vec{s}\,[i] = x$
      $\textsc{ImprovedGD}(\vec{s}, i + 1)$
    end for
  else
    return There exist no S-boxes corresponding to the given DDT!
  end if
end procedure

Complexity Analysis of the GD Phase

The expected time complexity of Algorithm 4 is

$$T_{n,m}(k) = 2^{m+1} P_{n,m}^{DDT} \sum_{i=0}^{2^n-2} W_i(k),$$

$$W_i = \begin{cases} 2^{(m-k)i} (P_{n,m}^{DDT})^{\frac{i^2+i}{2}} & ,0 \leq i \leq K, \\ 1 & ,K < i < 2^n, \end{cases}$$

where K is the smallest positive integer such that $2^{(m-k)i} (P_{n,m}^{DDT})^{\frac{i^2+i}{2}} < 1$.

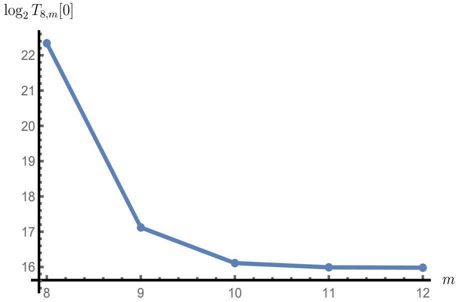


Figure 3: $\log_2 T_{8,m}(0)$ for 8-bit input S-box with different sizes of output

- Increasing the size of the output of the S-box (i.e., m) makes the reconstruction process easier.

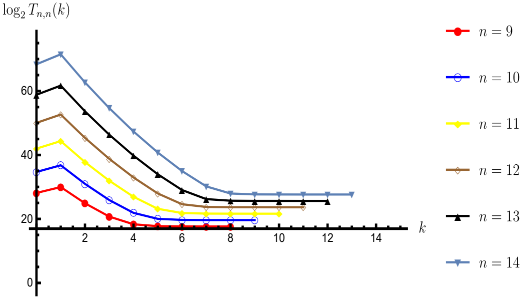


Figure 4: $\log_2 T_{n,n}(k)$ for random n-bit S-box with different k

- The original GD algorithm ($k = 0$) quickly becomes impractical with the size of S-box growing.
- To optimize the original GD algorithm, the attacker should find at least two independent good columns.
- When the number of good columns grows, the effect of reducing the search space of the GD phase becomes less significant.

Experiment Results

Three types of Boolean functions:

- Random S-boxes
- Specific S-boxes of Existing Ciphers
- 4-differential uniformity S-boxes and APN functions

A single core of an Intel(R) Xeon(R) E5-2620 v3 CPU @ 2.40GHz of 64GB memory.

Random S-boxes

n	k	Min (s)	Max (s)	Average (s)	Median (s)	Standard Deviation	Method
8	0	8.01×10^{-4}	0.07	0.01	0.01	0.01	GD algorithm
8	2	0.03	0.11	0.05	0.05	0.01	Our Approach
9	0	0.01	1.70	0.49	0.05	0.42	GD algorithm
9	3	0.39	0.70	0.50	0.49	0.06	Our Approach
10	0	0.88	159.94	45.80	38.83	36.0	GD algorithm
10	3	4.98	6.74	5.48	5.45	0.32	Our Approach
11	0	86.97	2.56×10^4	8.20×10^3	7.00×10^3	6.26×10^3	GD algorithm
11	4	43.61	94.68	58.23	57.00	11.34	Our Approach
12	0	3.88×10^4	8.73×10^6	3.66×10^6	4.17×10^6	2.17×10^6	GD algorithm
12	4	584.22	1437.26	962.33	925.08	167.38	Our Approach
13	0	5.72×10^7	3.90×10^9	1.83×10^9	1.96×10^9	9.90×10^8	GD algorithm
13	6	6.68×10^3	1.22×10^4	8.07×10^3	8.04×10^3	878.56	Our Approach
14	0	1.90×10^8	1.09×10^{12}	4.79×10^{11}	4.78×10^{11}	2.88×10^{11}	GD algorithm
14	6	6.93×10^4	8.81×10^4	7.52×10^4	7.39×10^4	4.07×10^3	Our Approach

Table 1: The Statistical Data for The Instances

- 4.79×10^{11}s are approximately 15178.9 years and 7.52×10^4s are less than one day.

Random S-boxes

n	k	Min (s)	Max (s)	Average (s)	Median (s)	Standard Deviation	Method
8	0	8.01×10^{-4}	0.07	0.01	0.01	0.01	GD algorithm
8	2	0.03	0.11	0.05	0.05	0.01	Our Approach
9	0	0.01	1.70	0.49	0.05	0.42	GD algorithm
9	3	0.39	0.70	0.50	0.49	0.06	Our Approach
10	0	0.88	159.94	45.80	38.83	36.0	GD algorithm
10	3	4.98	6.74	5.48	5.45	0.32	Our Approach
11	0	86.97	2.56×10^4	8.20×10^3	7.00×10^3	6.26×10^3	GD algorithm
11	4	43.61	94.68	58.23	57.00	11.34	Our Approach
12	0	3.88×10^4	8.73×10^6	3.66×10^6	4.17×10^6	2.17×10^6	GD algorithm
12	4	584.22	1437.26	962.33	925.08	167.38	Our Approach
13	0	5.72×10^7	3.90×10^9	1.83×10^9	1.96×10^9	9.90×10^8	GD algorithm
13	6	6.68×10^3	1.22×10^4	8.07×10^3	8.04×10^3	878.56	Our Approach
14	0	1.90×10^8	1.09×10^{12}	4.79×10^{11}	4.78×10^{11}	2.88×10^{11}	GD algorithm
14	6	6.93×10^4	8.81×10^4	7.52×10^4	7.39×10^4	4.07×10^3	Our Approach

Table 2: The Statistical Data for The Instances

- Our approach is much more stable than GD algorithm.

Random S-boxes

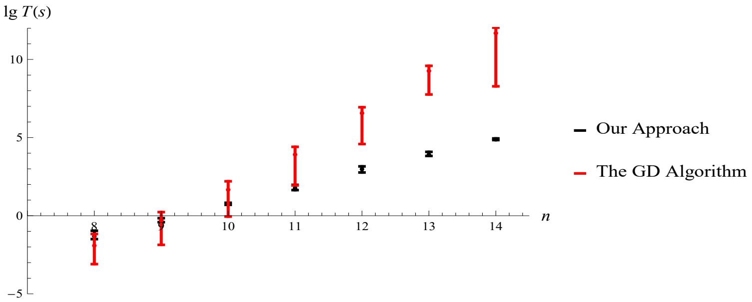


Figure 5: The Running Time on Random S-boxes

- The advantage of our approach over the GD algorithm sharply increases when the size of the S-box grows.

Random S-boxes

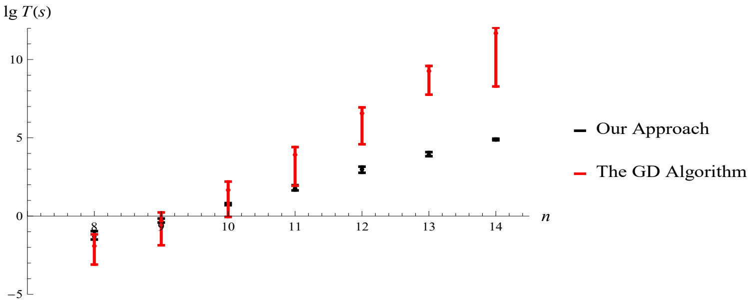


Figure 6: The Running Time on Random S-boxes

- When the input size of S-boxes is larger than 11, our approach is better in all cases.

Specific S-boxes of Existing Ciphers

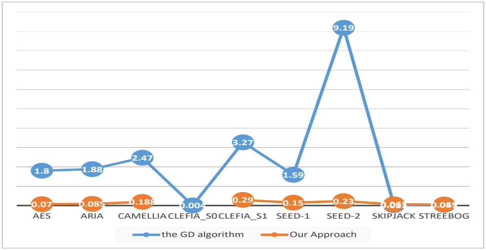


Figure 7: The Running Time on Specific S-boxes

- No good column is found in the S-box $S0$ of CLEFIA.
- Our approach is better: AES, ARIA, SEED, Camellia, and $S1$ of CLEFIA.
- GD algorithm is better: Streebog, Skipjack and $S0$ of CLEFIA.

4-differential uniformity S-boxes and APN functions

- It is difficult to find good columns in the absolute LAT of the S-boxes with low differential uniformity.
- It is also hard to find good columns in the absolute LAT of APN functions.

Conclusion and Open Problem

- We presented a new algorithm for reconstructing an S-box from its DDT. The new algorithm is more efficient than the guess-and-determine algorithm proposed by Boura et al. in [BCJS19], for random S-boxes starting at the size of 10 bits, it outperforms the previous GD algorithm by several orders of magnitude.
- The new algorithm can be useful to explore problems related to DDTs.
- Another related open problems are the problems of reconstructing an S-box from its *Boomerang Connectivity Table*, introduced in [CHP^{+}18] and its *Differential-Linear Connectivity Table*, introduced in [BODKW19], respectively.

Thank you for your attention!

Christina Boura, Anne Canteaut, Jérémy Jean, and Valentin Suder.
Two Notions of Differential Equivalence on Sboxes.
Des. Codes Cryptography, 87(2-3):185–202, 2019.

Céline Blondeau and Kaisa Nyberg.
New links between differential and linear cryptanalysis.
In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 388–404. Springer Berlin Heidelberg, 2013.

Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman.
DLCT: A New Tool for Differential-Linear Cryptanalysis.
In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, volume 11476 of *Lecture Notes in Computer Science*, pages 313–342, Cham, 2019. Springer Berlin Heidelberg.

Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song.
Boomerang Connectivity Table: A New Cryptanalysis Tool.
In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, volume 10821 of *Lecture Notes in Computer Science*, pages 683–714, Cham, 2018. Springer Berlin Heidelberg.

Florent Chabaud and Serge Vaudenay.
Links between Differential and Linear Cryptanalysis.
In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer Berlin Heidelberg, 1995.

Joan Daemen, René Govaerts, and Joos Vandewalle.
Correlation matrices.
In Bart Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pages 275–285. Springer Berlin Heidelberg, 1995.