Reconstructing an S-box from its Difference Distribution Table

Orr Dunkelman and Senyang Huang(✉)

Department of Computer Science, University of Haifa, Israel.
shuang@campus.haifa.ac.il

Abstract.
In this paper we study the problem of recovering a secret S-box from its difference distribution table (DDT). While being an interesting theoretical problem on its own, the ability to recover the S-box from the DDT of a secret S-box can be used in cryptanalytic attacks where the adversary can obtain the DDT (e.g., in Bar-On et al.'s attack on GOST), in supporting theoretical analysis of the properties of difference distribution tables (e.g., in Boura et al.'s work), or as a tool for developing an S-box with a unique differential trapdoor.
We show that using the well established relation between the DDT and the linear approximation table (LAT), one can devise an algorithm different from the guess-and-determine algorithm proposed by Boura et al. Moreover, we show how to exploit this relation, and embed the knowledge obtained from it in the guess-and-determine algorithm, and we discuss when our new method gives better results than the simple guess and determine attack.

Keywords: S-box, DDT, LAT, the sign determination problem

1 Introduction

Differential cryptanalysis, introduced by Biham and Shamir [7], has transformed the field of cryptanalysis and offered attacks against multiple symmetric-key primitives (and a few public-key ones). An essential component in estimating the probability of a differential characteristic is the *Difference Distribution Table* of an S-box. This table is easy to compute when the S-box is given (in time $O(2^{2n})$ for n-bit S-box). However, the inverse problem of deducing the S-box from a given DDT, was mostly left unstudied.

At first, this problem looks like a theoretical problem of very limited practical interest. However, efficient reconstruction of the S-box from the DDT is a useful tool in several cases. First, several cryptanalytic attacks on secret S-boxes constructions (such as GOST [16] and Blowfish [22]) may have access to the difference distribution table rather than the S-box itself. For example, in Bar-On et al.'s slide attack on GOST [6], the adversary can learn the DDT, and needs to deduce the secret S-box from it.

Another line of research that will enjoy such efficient reconstruction algorithms is the study of the theoretical properties of DDTs. A recent work by Boura et al. [11] studied a theoretical question — can two different S-boxes, that do not satisfy some trivial relation, share the same DDT. As part of this work, a guess-and-determine algorithm for the reconstruction of the S-box was introduced and used. While being practical for small S-boxes, this algorithm's running time was not analyzed for the general case, and it seems that for large S-boxes it may be impractical.

A different application of such a reconstruction algorithm is in the study of embedding backdoors in S-boxes. Consider a malicious designer who wishes to construct a large secret

S-box which has a differential-based backdoor known to him only. Such a designer may decide to first pick a DDT with many low entries, besides one (or a few) entries of relatively high value. Then, the adversary could use the reconstruction algorithm to find the actual S-box. This line of research thus supports the reverse engineering of "suspicious" S-boxes, e.g., the work by Biryukov and Perrin in [8].

In this paper we study the reconstruction problem in a different prospective from that of [11]. Instead of performing a guess-and-determine of the possible S-boxes, we rely on the well established relation between the DDT of an S-box and the Sbox's LAT [9, 10, 13]. We show that using this relation, it is possible to transform the DDT into multiple linear approximation tables,[1] each of which is offering an S-box (that can be easily computed relying on the Walsh-Hadamard transform).

After analyzing the basic idea, we show how to use the knowledge obtained using this approach in the guess-and-determine algorithm. The combination offers superior results to the previous ones, resulting in an algorithm which is (to the best of our knowledge) the best in solving the construction problem.

Finally, we test different types of S-box recovery cases, checking the time complexities of the actual inversion. These tests were performed on different sizes of S-boxes. We compare our method with the simple guess-and-determine algorithm and discuss in which cases our new method provides better performance than the simple guess and determine attack.

This paper is organized as follows. In Section 2, we discuss the preliminary of the reconstruction problem, including the DDT and LAT, the previous works on the relation between an Sbox, its DDT and its LAT. The notations used in this paper are also introduced. In Section 3, a new problem is introduced and we propose an algorithm to solve it. With the knowledge obtained by solving the new problem, the guess-and-determine algorithm in [11] is improved in Section 4. Then, our approach is implemented on the different S-boxes and some special Boolean functions. In Section 5 we compare the performances of our method with the guess-and-determine algorithm in [11]. In Section 6, we conclude this paper.

2 Background and Notations

Throughout the paper we discuss S-boxes with n-bit inputs and m-bit outputs, i.e., $n \times m$ S-box. When $m = n$, we refer to the S-box as an n-bit S-box. We treat the S-box as a vectorial Boolean function, i.e., $S(x) = (S^{m-1}(x), \ldots, S^0(x))$.

After recalling the definitions of the difference distribution table and the linear approximation table, the previous work on the relation between the two tables is revisited. Then the notations which are used in this paper are described.

2.1 Difference Distribution Table and Linear Approximation Table

The difference distribution table (DDT) of an S-box counts the number of cases when the input difference of a pair is a and the output difference is b (see [7]).[2] For an input difference $a \in \mathbb{F}_2^n$ and an output difference $b \in \mathbb{F}_2^m$, the entry $\delta(a, b)$ of the Sbox's DDT is:

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

We follow the work of Boura et al. in [11] and call two S-boxes $S_0(x)$ and $S_1(x)$ DDT-equivalent if they have the same DDT. In [11], a DDT-equivalence class is called *trivial* when its size matches the lower-bound given in Property 1:

[1]As we later discuss in Section 2.2, this relation allows for recovering the absolute values of the entries of the LAT, and the signs of the different entries are to be determined.

[2]We note that [7] also discusses DDTs that store these pairs. However, for the sake of this work we use the common DDT that stores only the number of pairs.

Property 1. Let S be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$ and let ℓ denote the dimension of its linear space, i.e., of the space formed by all linear structures of S. Then, the DDT-equivalence class of S contains the $2^{m+n-\ell}$ distinct functions of the form

$$x \to S(x \oplus c) \oplus d, \qquad c \in \mathbb{F}_2^n, \quad d \in \mathbb{F}_2^m.$$

The differential uniformity is an important characteristic for analysing the resistance to differential cryptanalysis (see [21]). The differential uniformity of an S-box $S(x)$ is defined as

$$\max_{a \in \mathbb{F}_2^n \setminus \{0^n\}, b \in \mathbb{F}_2^m} \delta(a,b).$$

The lowest possible value for the differential uniformity of a function from $\mathbb{F}_2^n$ into itself is two and the functions with differential uniformity two are called almost perfect nonlinear (APN). As we discuss in Section 5, it is harder to reconstruct an APN function from its DDT.

The linear approximation table (LAT) of an S-box is used to derive approximate linear relations between input bits and output bits of the S-box (see [19]). For any input mask $a \in \mathbb{F}_2^n$ and any output mask $b \in \mathbb{F}_2^m$, the LAT entry is defined as

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

where $a \cdot X$ and $b \cdot S(X)$ are the inner product over $\mathbb{F}_2$. It could also be seen from Equation 1 that given the S-box's LAT, the adversary can reconstruct the underlying S-box by solving a system of linear equations (see Lemma 2 in [13]).

The *nonlinearity* of a Boolean function f from $\mathbb{F}_2^n$ to $\mathbb{F}_2$ is the minimum Hamming distance between f and the set of affine functions. In our case, for each $b \in \mathbb{F}_2^m$, the nonlinearity of $b \cdot S(x)$ is

$$nf(b \cdot S(x)) = 2^{n-1} - \max_{a \in \mathbb{F}_2^n \setminus \{0^n\}} |\lambda(a,b)|.$$

When for all a, $|\lambda(a,b)|$ in the b-th column is equal to $2^{n/2-1}$, $b \cdot S(x)$ is called a *bent function* (see [12]).

2.2 Links between an S-box, its Difference Distribution Table and its Linear Approximation Table

We now revisit the relation between DDTs and LATs observed in [9, 10, 13]. To do so, we start with the Walsh-Hadamard transform. Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{R}$ be a function, $\hat{f}$ is denoted as the *Walsh-Hadamard transform*:

$$\forall (a,b) \in \mathbb{F}_2^n \times \mathbb{F}_2^m, \hat{f}(a,b) = \sum_{x,y} f(x,y)(-1)^{a \cdot x \oplus b \cdot y},$$

where the sum is evaluated over the reals and $a \cdot x$ and $b \cdot y$ are the inner product over the domain $\mathbb{F}_2^n$ and $\mathbb{F}_2^m$, respectively. Theorem 1 obtained in [10, 13] shows that the entries of DDT and LAT are linked to each other through the Walsh-Hadamard transform.

Theorem 1. *([10, 13]) 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 transform of* $\lambda^2(a,b)$.

The first conclusion in the theorem above implies that a squared entry in the LAT can be deduced from the given DDT. Hence, in order to recover the S-box from the DDT, we need to determine the signs of the entries in the LAT as we discuss in Section 3.

2.3 Notations

In the LAT, we denote the b-th column of the LAT by $\vec{\lambda}_b$, where $0 \leq b < 2^m$. In addition, we use $\lambda^\dagger(a,b)$ to denote the absolute value of the element $\lambda(a,b)$ of the LAT. We extend the $\dagger$ notion to vectors as follows:

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

where $\vec{v} = (v_0, \ldots, v_{\ell-1})$ and $|\cdot|$ is the absolute value of a number. We define the absolute LAT of the vectorial Boolean function S as $(\vec{\lambda}_0^\dagger, \cdots, \vec{\lambda}_{2^m-1}^\dagger)$. We also define $\vec{s}_b$ as $((-1)^{b\cdot S(0)}, \ldots, (-1)^{b\cdot S(2^m-1)})$.

Let $\vec{u} = (u_0, \ldots, u_{\ell-1})$ and $\vec{v} = (v_0, \ldots, v_{\ell-1})$ be two vectors. Then, the *Hadamard product* of these vectors is denoted by $\vec{u} \odot \vec{v} = (u_0 \cdot v_0, \ldots, u_{\ell-1} \cdot v_{\ell-1})$.

Finally, we define for a vector $\vec{v} = (v_0, \ldots, v_{\ell-1})$, a partial vector $\vec{v}^{\,[x,y]} = (v_x, \ldots, v_y)$, $0 \leq x < y < \ell$.

3 The Sign Determination Problem

As suggested in Section 2.2, given the DDT, we can easily compute $\lambda^\dagger(a,b)$. To recover the S-box we just need to determine the signs of the entries. We define the sign determination problem as follows:

Definition 1. *Given $\vec{\lambda}_b^\dagger$, the **sign determination problem** of the b-th column in an LAT is the problem of recovering $\vec{\lambda}_b$ from $\vec{\lambda}_b^\dagger$, for $1 \leq b < 2^m$.*

We introduce a new method for solving the sign determination problem. To achieve this purpose, the linear relation between $\vec{\lambda}_b$ and $\vec{s}_b$ is studied in Section 3.1. Based on this property, an algorithm for solving the sign determination problem is discussed in Section 3.2.

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

Property 2. For any b-th column in a linear approximation table, $0 \leq b < 2^m$, the following formula holds

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

where H_n is the Hadamard matrix of order 2^n.

Proof. From the definition of LAT, the following equality is obtained straightforwardly.

$$\begin{aligned}\sum_{a\in\mathbb{F}_2^n} 2(-1)^{a\cdot p}\lambda(a,b) &= \sum_{a\in\mathbb{F}_2^n}\sum_{x\in\mathbb{F}_2^n} (-1)^{a\cdot p}\cdot(-1)^{a\cdot x\oplus b\cdot S(x)} \\ &= \sum_{x\in\mathbb{F}_2^n}\sum_{a\in\mathbb{F}_2^n} (-1)^{b\cdot S(x)}(-1)^{a\cdot x\oplus a\cdot p}.\end{aligned}$$

From Proposition 7 in [12], if $x = p$, $\sum_{a\in\mathbb{F}_2^n} (-1)^{a\cdot x \oplus a\cdot p} = 2^n$; otherwise the sum is zero.

$$\sum_{a\in\mathbb{F}_2^n} 2(-1)^{a\cdot p}\lambda(a,b) = (-1)^{b\cdot S(p)}2^n, \quad 0 \le p < 2^n. \tag{2}$$

$$\Rightarrow \sum_{a\in\mathbb{F}_2^n} (-1)^{p\cdot a}\lambda(a,b) = (-1)^{b\cdot S(p)}\cdot 2^{n-1}, \quad 0 \le p < 2^n. \tag{3}$$

$$\Rightarrow \qquad H_n\vec{\lambda}_b = 2^{n-1}\vec{s}_b \tag{4}$$

As $H_n \cdot H_n = 2^n I_{2^n}$, where I_{2^n} is the identity matrix of order 2^n, this formula can also be written as $H_n\vec{s}_b = 2\vec{\lambda}_b$. Remark that when $p = 0$ in Equation 3, it follows that $\sum_{a\in\mathbb{F}_2^n} \lambda(a,b) = \pm 2^{n-1}$. □

Note that the assignments of the b-th column are related to the linear combination of the components of $S(x)$. Let $b = b_{2^m-1}\dots b_0$ be the binary representation of b. It implies that $b\cdot S(x) = \bigoplus_{i=0}^{2^m-1} b_i S^i(x)$.

We call the c_0-th,..., the c_j-th columns in the LAT *independent columns* if $c_0,\dots,c_j$ are linearly independent over $\mathbb{F}_2^m$, $0 \le j < m$. If the adversary solves the sign determination problems over m independent columns, the S-box can be reconstructed by solving linear equations with negligible complexity $O(2^n m^3)$. It is obvious that the adversary can solve the sign determination problem by using a brute force attack but the complexity is very high, $O(2^{m\cdot 2^n})$ as there are m columns of 2^n elements each. We propose a significantly more efficient method for for solving this problem of the b-th column in Section 3.2 .

3.2 An Algorithm for Solving the Sign Determination Problem

3.2.1 Block representation of the Hadamard matrix

Before presenting our algorithm, we discuss a basic property of Hadamard matrices.

Property 3. The Hadamard matrix H_i can be represented as

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

With the block representation of the Hadamard matrix in Property 3, we can solve the system of linear equations $H_n\vec{s}_b = 2\vec{\lambda}_b$ recursively, given $\vec{\lambda}_b$. It is clear that by elementary transformation,

$$(H_n, 2\vec{\lambda}_b) = \begin{pmatrix} H_{n-1} & H_{n-1} & 2\vec{\lambda}_b^{[0,2^{n-1}-1]} \\ H_{n-1} & -H_{n-1} & 2\vec{\lambda}_b^{[2^{n-1},2^n-1]} \end{pmatrix}$$
$$\Rightarrow \begin{pmatrix} H_{n-1} & 0 & \vec{\lambda}_b^{[0,2^{n-1}-1]} + \vec{\lambda}_b^{[2^{n-1},2^n-1]} \\ 0 & H_{n-1} & \vec{\lambda}_b^{[0,2^{n-1}-1]} - \vec{\lambda}_b^{[2^{n-1},2^n-1]} \end{pmatrix}.$$

It is easy to see that the original problem is divided into two subproblems as follows:

$$H_{n-1}\vec{s}_b^{\,[0,2^{n-1}-1]} = \vec{\lambda}_b^{[0,2^{n-1}-1]} + \vec{\lambda}_b^{[2^{n-1},2^n-1]}$$
$$H_{n-1}\vec{s}_b^{\,[2^{n-1},2^n-1]} = \vec{\lambda}_b^{[0,2^{n-1}-1]} - \vec{\lambda}_b^{[2^{n-1},2^n-1]}.$$

The process above is the first step in solving $H_n\vec{s}_b = 2\vec{\lambda}_b$. We then recursively apply the above process to the problems in the ℓ-th step. At the beginning of the ℓ-th step, there are $2^{\ell-1}$ problems with $2^{n-\ell+1}$ constraints, denoted as:

$$\begin{aligned} H_{n-\ell+1}\vec{s}_b^{\,[0,2^{n-\ell+1}-1]} &= \vec{\beta}_0, \\ &\vdots \\ H_{n-\ell+1}\vec{s}_b^{\,[2^n-2^{n-\ell+1},2^n-1]} &= \vec{\beta}_{2^{\ell-1}-1}, \end{aligned} \tag{5}$$

where $\vec{\beta}_0,\ldots,\vec{\beta}_{2^{\ell-1}-1}$ are the vectors obtained from the last step and $1 \le \ell \le n$. Each problem in Equation 5 is divided into two subproblems as follows:

$$\begin{aligned} H_{n-\ell}\vec{s}_b^{\,[0,2^{n-\ell}-1]} &= \vec{\gamma}_0, \quad H_{n-\ell}\vec{s}_b^{\,[2^{n-\ell},2^{n-\ell+1}-1]} = \vec{\gamma}_1, \\ &\vdots \\ H_{n-\ell}\vec{s}_b^{\,[2^n-2^{n-\ell+1},2^n-2^{n-\ell}-1]} &= \vec{\gamma}_{2^\ell-2}, \quad H_{n-\ell}\vec{s}_b^{\,[2^n-2^{n-\ell},2^n-1]} = \vec{\gamma}_{2^\ell-1}. \end{aligned} \tag{6}$$

where,

$$\begin{aligned} \vec{\gamma}_0 &= \left(\vec{\beta}_0^{[0,2^{n-\ell}-1]} + \vec{\beta}_0^{[2^{n-\ell},2^{n-\ell+1}-1]}\right)/2 \quad , \quad \vec{\gamma}_1 = \left(\vec{\beta}_0^{[0,2^{n-\ell}-1]} - \vec{\beta}_0^{[2^{n-\ell},2^{n-\ell+1}-1]}\right)/2 \\ &\vdots \\ \vec{\gamma}_{2^\ell-2} &= \left(\vec{\beta}_{2^{\ell-1}-1}^{[0,2^{n-\ell}-1]} + \vec{\beta}_{2^{\ell-1}-1}^{[2^{n-\ell},2^{n-\ell+1}-1]}\right)/2 \quad , \quad \vec{\gamma}_{2^\ell-1} = \left(\vec{\beta}_{2^{\ell-1}-1}^{[0,2^{n-\ell}-1]} - \vec{\beta}_{2^{\ell-1}-1}^{[2^{n-\ell},2^{n-\ell+1}-1]}\right)/2 \end{aligned}$$

The total number of subproblems after the ℓ-th step is 2^ℓ and the number of constraints in each subproblem is $2^{n-\ell}$. At the n-th step, the coefficient matrix in the subproblems is $H_0 = 1$. Thus, the entries of $\vec{s}_b$ are obtained.

3.2.2 Main idea

We propose to solve the sign determination problem using a recursive procedure. In each layer, the algorithm works on the systems of linear equations with the size reduced by half compared to the ones in the previous layer. Finally, when it reaches the n-th layer, the algorithm returns the solutions to the sign determination problem.

More precisely, the result of the above recursive approach is an immediate recursive algorithm. This algorithm can be represented by a tree structure. For ease of explanation we denote the ℓ-th layer of the recursion tree by T_ℓ.

The algorithm is initialized by guessing the signs of $\lambda(i,b)$ if $\lambda(i,b) \neq 0$. Thus, the leaf node $T_0[i]$ is assigned with $\{2\lambda^\dagger(i,b), -2\lambda^\dagger(i,b)\}, 0 \le i < 2^n$. At the beginning of the ℓ-th layer, the subproblems in Equation 5 are recorded in $T_{\ell-1}$. The i-th constraint in Equation 5 is stored in a vector as $\vec{v}^T = (\vec{\beta}_0[i],\ldots,\vec{\beta}_{2^{\ell-1}-1}[i])$, $0 \le i < 2^{n-\ell+1}$. We call the set which contains all the possible i-th constraints a *full set*, denoted by $F_{\ell-1}[i]$, $0 \le i < 2^{n-\ell+1}, 1 \le \ell \le (n+1)$. The naive strategy is to record the full set $F_{\ell-1}[i]$ in the internal node $T_{\ell-1}[i]$.

In the ℓ-th layer, the i-th possible constraints of the new subproblems in Equation 6 are deduced from Equation 5 to construct $F_\ell[i]$, $0 \le i \le 2^{n-\ell}$. To do so, for each vector in $\vec{p} \in F_{\ell-1}[i]$ and $\vec{q} \in F_{\ell-1}[i+2^{n-\ell}]$, a new vector which is defined as $E_{\ell-1}(\vec{p},\vec{q})$ is computed as described below:

$$\begin{aligned} E_{\ell-1}(\vec{p},\vec{q})^T =& ((p_0+q_0)/2, (p_0-q_0)/2, \ldots, \\ & (p_{2^{\ell-1}-1}+q_{2^{\ell-1}-1})/2, (p_{2^{\ell-1}-1}-q_{2^{\ell-1}-1})/2), \end{aligned}$$

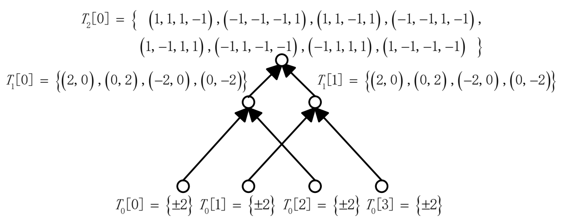

$T_2[0] = \{ (1,1,1,-1), (-1,-1,-1,1), (1,1,-1,1), (-1,-1,1,-1), (1,-1,1,1), (-1,1,-1,-1), (-1,1,1,1), (1,-1,-1,-1) \}$
$T_1[0] = \{(2,0),(0,2),(-2,0),(0,-2)\}$
$T_1[1] = \{(2,0),(0,2),(-2,0),(0,-2)\}$
$T_0[0] = \{\pm 2\}$ $T_0[1] = \{\pm 2\}$ $T_0[2] = \{\pm 2\}$ $T_0[3] = \{\pm 2\}$

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

where p_j and q_j are the j-th entries with respect to $\vec{p}$ and $\vec{q}$, $0 \le j < 2^{\ell-1}$.

It can be seen from Equation 6 that each entry of the vector in $F_\ell[i]$ is an even number in the range from $-2^{n-\ell}$ to $2^{n-\ell}$ when $1 \le \ell < n$. As the components of $\vec{s}_b$ are 1 or -1, then in the ℓ-th layer the entries of the vectors in $F_\ell[0]$ take the values from the set $\{1, -1\}$. If the constraints over the elements of the vectors in $F_\ell[i]$ are satisfied for the vector $E_{\ell-1}(\vec{p}, \vec{q})$, the new vector is a possible i-th constraint; otherwise, it should be discarded. When it reaches the n-th layer, the solutions of the sign determination problem are the vectors in the root node $T_n[0]$.

To illustrate our idea more intuitively, we refer to the recursive tree for $n = 2$ in Figure 1 and show an example when $\vec{\lambda}_b^\dagger = (1,1,1,1)$ and the corresponding LAT column $\vec{\lambda}_b$ is $(1,1,1,-1)$ and $\vec{s}_b = (1,1,1,-1)$. The nodes in T_0 are initialized by $T_0[0] = T_0[1] = T_0[2] = T_0[3] = \{\pm 2\}$. As shown in Figure 1, $T_1[0]$ is constructed from $T_0[0]$ and $T_0[2]$ and $T_1[1]$ is from $T_0[1]$ and $T_0[3]$. $T_1[0] = T_1[1] = \{(2,0),(0,2),(-2,0),(0,-2)\}$. Similarly, $T_2[0]$ is built from $T_1[0]$ and $T_1[1]$. For each $\vec{p} \in T_1[0]$ and $\vec{q} \in T_1[1]$, we compute $E_1(\vec{p}, \vec{q})$. For example, when $\vec{p} = \vec{q} = (2,0)$, $E_1(\vec{p}, \vec{q}) = (2,0,0,0) \notin T_2[0]$. Therefore, there are eight vectors in $T_2[0]$, which are $(1,1,1,-1)$, $(-1,-1,-1,1)$, $(1,1,-1,1)$, $(-1,-1,1,-1)$, $(1,-1,1,1)$, $(-1,1,-1,-1)$, $(-1,1,1,1)$ and $(1,-1,-1,-1)$. It can be seen that $\vec{s}_b \in T_2[0]$.

3.2.3 Core set and full set

When we examine the full set $F_\ell[i]$, we notice that its vectors are related, $1 \le \ell \le n$ and $0 \le i < 2^{n-\ell}$. Using this relation, the intermediate results in the technique above can be stored in a more efficient manner without losing any solutions, which reduces both time and memory complexities of the search algorithm.

In the following, we define a *core set* $C_\ell[i]$ to be a compact representation of the full set $F_\ell[i]$. The compact set $C_\ell[i]$ allows rebuilding $F_\ell[i]$ efficiently and thus it is stored in the internal node $T_\ell[i]$ of the tree structure to optimize the recursive procedure above. After that we discuss how to construct the core set $C_{\ell+1}[i]$ from $C_\ell[i]$ and $C_\ell[i + 2^{n-\ell-1}]$, $0 \le i < 2^{n-\ell-1}$.

Before presenting the structure of $F_\ell[i]$, we define a set of symmetric permutations. Let $\vec{v}^T$ be $(v_0, \ldots, v_{2^\ell-1})$ and Π_ℓ be a set of symmetric permutations $\pi_0^\ell, \ldots, \pi_{\ell-1}^\ell$. We define π_j^ℓ as follows:

$$\pi_j^\ell(\vec{v}) = (v_{2^j}, \ldots, v_{2^{j+1}-1}, v_0, \ldots, v_{2^j-1}, \ldots,$$
$$v_{2^\ell-2^j}, \ldots, v_{2^\ell-1}, v_{2^\ell-2^{j+1}}, \ldots, v_{2^\ell-2^j-1})^T,$$

where $0 \le j < \ell$. $\vec{v}$ is divided into $2^{\ell-j}$ blocks each of size 2^j. Note that the permutation π_j^ℓ swaps every two consecutive blocks of size 2^j in $\vec{v}$ pairwisely. It can be easily verified that each permutation in Π_ℓ is of order two and $\pi_0^\ell, \ldots, \pi_{\ell-1}^\ell$ are commutative. Now, we define a j-symmetric relation between two vectors with respect to the permutations in Π_ℓ that helps in capturing the structure of the full set $F_\ell[i]$.

Definition 2. *The vector $\vec{u}$ is j-**symmetric** to the vector $\vec{v}$ if there exist p permutations in Π_ℓ such that*

$$\vec{u} = \begin{cases} \pi^\ell_{j_{p-1}} \circ \ldots \circ \pi^\ell_{j_0}(\vec{v}) & , 0 \le j_0 < \cdots < j_{p-1} = j,\ p \ge 1, \\ & \textit{or} \\ -\pi^\ell_{j_{p-1}} \circ \ldots \circ \pi^\ell_{j_0}(\vec{v}) & , 0 \le j_0 < \cdots < j_{p-1} = j,\ p \ge 1, \end{cases} \tag{7}$$

*where $0 \le j < \ell$. For the first scenario in Equation 7, $\vec{u}$ is positive j-symmetric to $\vec{v}$; otherwise, $\vec{u}$ is negative j-symmetric to $\vec{v}$. For the special case when $j = \ell$, the ℓ-**symmetric** vectors to $\vec{u}$ are defined as $\vec{u}$ and $-\vec{u}$.*

We say that $\vec{u}$ is symmetric-equivalent to $\vec{v}$ if there exists j such that $\vec{u}$ is j-symmetric to $\vec{v}$. It can be easily verified that the symmetric-equivalent relation is an equivalence relation. Thus, for each vector $\vec{u} \in F_\ell[i]$, the symmetric-equivalence class of $\vec{u}$ is $[\vec{u}] = \bigcup_{j=0}^{\ell} \{\vec{v} | \vec{v}$ is j-symmetric to $\vec{u}\}$. In Theorem 2, we present the symmetric structure of $F_\ell[i]$ that for each $\vec{u} \in F_\ell[i]$, $[\vec{u}] \subset F_\ell[i]$.

Theorem 2. *For any vector $\vec{u} \in F_\ell[i]$, if a vector $\vec{v}$ is j-symmetric to $\vec{u}$ for $0 \le j \le \ell$, then $\vec{v} \in F_\ell[i]$, $0 \le i < 2^{n-\ell}$.*

We define the core set $C_\ell[i]$ with respect to $F_\ell[i]$, which is a set of representatives of the symmetric-equivalence classes. More precisely, the relation between the full set $F_\ell[i]$ and its core set $C_\ell[i]$ is described as follows:

$$F_\ell[i] = \bigcup_{\vec{u} \in C_\ell[i]} [\vec{u}]. \tag{8}$$

For each vector $\vec{u} \in C_\ell[i]$, $\vec{v} \in [\vec{u}]$ can be constructed using Definition 2. The full set $F_\ell[i]$ can thus be obtained by applying Equation 8 to all $\vec{u} \in C_\ell[i]$. Thus, the original process is optimized by storing the core set $C_\ell[i]$ instead of the full set $F_\ell[i]$ in the intermediate node $T_\ell[i]$, which avoids the repeated computation and reduces the memory consumption.

Before we discuss our technique for constructing the core sets in the $(\ell+1)$-th layer, we define the symmetric characteristics of a vector and a set, respectively.

Definition 3. *$\vec{u} \in F_\ell[i]$ is a j-symmetric vector for $0 \le i < 2^{n-\ell}, 0 \le j < \ell$ if $\vec{u}$ is j-symmetric to itself. $F_\ell[i]$ is a j-symmetric set for $0 \le i < 2^{n-\ell}, 0 \le j < \ell$ if $\vec{u}$ if all the vectors in $F_\ell[i]$ are j-symmetric.*

It is not necessary to check all the vectors in the full set $F_\ell[i]$ to detect whether $F_\ell[i]$ is j-symmetric. We show in Theorem 3 that all the vectors in the full set $F_\ell[i]$ have the same symmetric characteristic. As $C_\ell[i]$ is a subset of $F_\ell[i]$, all the vectors in $C_\ell[i]$ are with the same symmetric characteristic as well. The proof of Theorem 3 is given in Appendix B.

Theorem 3. *For every $0 \le j < \ell$, if a vector $\vec{u} \in F_\ell[i]$ is a j-symmetric vector, $F_\ell[i]$ is a j-symmetric set, $0 \le i < 2^{n-\ell}$ and $1 \le \ell \le n$.*

In order to build the core set $C_{\ell+1}[i]$, we construct the smallest set $M_\ell[i + 2^{n-\ell-1}]$ such that $C_{\ell+1}[i]$ is obtained by computing $E_\ell(\vec{p}, \vec{q})$ for each $\vec{p} \in C_\ell[i]$ and $\vec{q} \in M_\ell[i + 2^{n-\ell-1}]$, where $C_\ell[i + 2^{n-\ell-1}] \subseteq M_\ell[i + 2^{n-\ell-1}] \subseteq F_\ell[i + 2^{n-\ell-1}]$.[3] The structure of $M_\ell[i + 2^{n-\ell-1}]$ is related to the symmetric property of the vectors in the core set $C_\ell[i]$. Now, we show the structure of $M_\ell[i + 2^{n-\ell-1}]$ in Theorem 4, proved in Appendix C.

[3]It follows from Theorem 4 that only when $C_\ell[i]$ is not j-symmetric for $0 \le j < \ell$, $M_\ell[i + 2^{n-\ell}] = F_\ell[i + 2^{n-\ell}]$

Theorem 4. *For $0 \leq i \leq 2^{n-\ell-1}$ and $0 \leq \ell < n$,*

$$M_\ell[i + 2^{n-\ell-1}] = \bigcup_{\vec{v} \in C_\ell[i]} \bigcup_{j \in J} \{\vec{u} | \vec{u} \text{ is } j\text{-symmetric to } \vec{v}\},$$

where $J = \{j | C_\ell[i]$ is not a j-symmetric set for $0 \leq j < \ell\}$.

For example, we assume that $C_1[0] = \{(a, a)\}$ and $C_1[2^{n-1}] = \{(c, d)\}$ when $c \neq d$. It follows from Theorem 4 that $M_1[2^{n-1}] = \{(c, d)\} \subset F_1[2^{n-1}] = \{(c, d), (-c, -d), (d, c), (-d, -c)\}$. From $C_1[0]$ and $M_1[2^{n-1}]$, we construct the core set $C_2[0] = \{(\frac{a+c}{2}, \frac{a-c}{2}, \frac{a+d}{2}, \frac{a-d}{2})\}$. Thus, there exist no two vectors in $C_2[0]$ which are symmetric-equivalent to each other.

Based on Theorem 3 and Theorem 4, $M_\ell[i+2^{n-\ell-1}]$ can be constructed by Algorithm 1, $0 \leq i < 2^{n-\ell-1}$ and $1 \leq \ell < n$. Note that we use the notation $\pi_j^\ell(S)$ in Algorithm 1 to denote the set $\{\pi_j^\ell(\vec{v}) | \vec{v} \in S\}$. It can be easily verified that the set constructed from $C_\ell[i]$ and $M_\ell[i + 2^{n-\ell-1}]$ is indeed $C_{\ell+1}[i]$.

Algorithm 1 Construct $M_\ell[i + 2^{n-\ell-1}]$ from $C_\ell[i]$ and $C_\ell[i + 2^{n-\ell-1}]$

```
1: procedure ConstructSet($C_\ell[i]$,$C_\ell[i + 2^{n-\ell-1}]$)
2:     $M_\ell[i + 2^{n-\ell-1}] = \emptyset$
3:     Randomly pick a vector $\vec{p}$ from $C_\ell[i]$
4:     for each vector $\vec{q} \in C_\ell[i + 2^{n-\ell-1}]$ do
5:         $S = \{\vec{q}\}$
6:         for all integers $j \in [0, \ell - 1]$ do
7:             if $\vec{p}$ is not $j$-symmetric then
8:                 $S = S \cup \pi_j^\ell(S)$
9:             end if
10:        end for
11:        $M_\ell[i + 2^{n-\ell-1}] = M_\ell[i + 2^{n-\ell-1}] \cup S$
12:    end for
13:    return $M_\ell[i + 2^{n-\ell-1}]$
14: end procedure
```

In the initial phase, the leaf nodes are assigned as $C_0[i] = \{2\lambda^\dagger(i, b)\}$, $0 \leq i < 2^n$. In the first layer, $C_1[i] = \{(\lambda^\dagger(i, b) + \lambda^\dagger(i + 2^{n-1}, b), \lambda^\dagger(i, b) - \lambda^\dagger(i + 2^{n-1}, b))\}$, $0 \leq i < 2^n$. In the ℓ-th layer, $C_\ell[i]$, $2 \leq \ell \leq n$ and $0 \leq i < 2^{n-\ell}$, is constructed from combining the elements in $C_{\ell-1}[i]$ and $M_{\ell-1}[i + 2^{n-\ell}]$, where $M_{\ell-1}[i + 2^{n-\ell}]$ is built from $C_{\ell-1}[i + 2^{n-\ell}]$ with Algorithm 1. After n iterations, the solutions to the sign determination problem are the vectors in the full set $F_n[0]$, which can be easily reconstructed from the core set $C_n[0]$ by Equation 8. The search process of the sign determination problem is stated in Algorithm 2.

We fix $S(0)$ to 0 (or any other constant) to find one representative of the DDT-equivalence class in Property 1. Therefore, Algorithm 2 only returns the vectors with the first element as $(-1)^{b \cdot S(0)} = 1$.

The Boolean functions corresponding to the solutions of its sign determination problem share the same squared LAT with $b \cdot S(x)$. These Boolean functions are DDT-equivalent with $b \cdot S(x)$. When $b \cdot S(x)$ is contained in a nontrivial DDT-equivalence classes, $T_n[0]$ contains multiple vectors.

The number of the solutions of its sign determination problem is equal to the size of the Boolean functions which are DDT-equivalent to $b \cdot S(x)$, $1 \leq b < 2^m$. We note that determining the size of the DDT-equivalence classes of a Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$ is an open problem.

Given enough memory, Algorithm 2 can solve all the instances of the sign determination problem. However, for some columns, the amount of vectors in the internal layer grows

Algorithm 2 An Algorithm for Solving the Sign Determination Problem

```
1: Input: $\vec{\lambda}_b^\dagger$;
2: Output: $F = \{\vec{u} | H_n \vec{u} = 2\vec{\lambda}_b, \vec{u}[0] = 1\}$
3: for each integer $i \in [0, 2^n - 1]$ do
4:     $C_0[i] = \{2\lambda^\dagger(i, b)\}$        ▷Initial phase
5: end for
6: $C_n[0] = $ LAYER$(0, T_0)$
7: Construct the full set $F_n[0]$ from $C_n[0]$.
8: $F = \{\vec{u} | \vec{u} \in F_n[0], \vec{u}[0] = 1\}$.
9: return $F$.
10:
11: procedure LAYER$(C_\ell, \ell)$;
12:     for each integer $i \in [0, 2^{n-\ell-1} - 1]$ do
13:         if there are no vectors in $C_\ell[i]$ or $C_\ell[i + 2^{n-\ell-1}]$ then
14:             return There exist no S-boxes corresponding to the given DDT!
15:         end if
16:         $C_{\ell+1}[i] = \emptyset$
17:         $M = $ CONSTRUCTSET$(C_\ell[i], C_\ell[i + 2^{n-\ell-1}])$
18:         for each vector $\vec{t_0} \in C_\ell[i]$ and each $\vec{t_1}$ in $M$ do
19:             compute $\vec{u} = E_\ell(\vec{t_0}, \vec{t_1})$
20:             if $\ell < n$ then
21:                 if every entry in $\vec{u}$ is even and ranges from $-2^{n-\ell-1}$ to $2^{n-\ell-1}$ then
22:                     $C_{\ell+1}[i] = C_{\ell+1}[i] \cup \{\vec{u}\}$
23:                 end if
24:             else
25:                 if every entry in $\vec{u}$ is 1 or $-1$ then        ▷ when $\ell = n$
26:                     $C_n[i] = C_n[i] \cup \{\vec{u}\}$
27:                 end if
28:             end if
29:         end for
30:     end for
31:     if $\ell < n$ then
32:         LAYER$(C_{\ell+1}, \ell + 1)$
33:     else
34:         return $C_n[0]$
35:     end if
36: end procedure
```

sharply, which demands too much memory. In this situation, a threshold H on the number of internal vectors can be preset heuristically. In the ℓ-th layer, if the size of $C_\ell[i]$ rises above the threshold H, the search process is interrupted, where $0 \le \ell < n$ and $0 \le i < 2^{n-\ell}$.

We call $\vec{\lambda}_b^\dagger (1 \le b < 2^m)$ a *good column* if it can be recovered under the threshold T; otherwise, it is called a *bad column*. For example, the S-boxes in CAST-256 [3], like $S0$, are 8×32 S-boxes, which were constructed by choosing 32 distinct bent functions as the components (see [2]). It indicates that each entry of columns $\vec{\lambda}_1^\dagger, \vec{\lambda}_2^\dagger, \ldots, \vec{\lambda}_{2^{31}}^\dagger$ is $2^{8/2-1} = 8$. For these instances, each core set in the fourth layer contains 2048 vectors. It implies that if the threshold is set as 2000, these columns are bad columns which cannot be solved by Algorithm 2 with respect to the memory. However, there are still some good columns in the absolute LAT of CAST-256's $S0$, for example $\vec{\lambda}_6^\dagger$ and $\vec{\lambda}_7^\dagger$ corresponding to the 6th and 7th columns of its LAT.

4 Applying Algorithm 2 for Reconstructing the S-box

The procedure of reconstructing an $n \times m$ S-box is related to the number of good columns. We suppose that the adversary has solved the sign determination problem for k independent good columns, i.e., the c_0-column,$\cdots$, and the c_{k-1}-column, $1 \le k \le m$. In the sign determination problem for the c_i-th column, the possible candidates for the Boolean function $c_iS(x)$ are recovered by Algorithm 2. We call it the matching phase for the k good columns for $1 < k \le m$ when the combination of these candidates is searched with respect to the squared LAT.

After the matching phase for the k good columns, the Boolean functions $c_0S(x)$, $\cdots$, $c_{k-1}S(x)$ are obtained. As mentioned before, when $k = m$, the adversary can reconstruct the S-box using linear algebra. When $k < m$, applying the knowledge of $c_0S(x)$, $\cdots$, $c_{k-1}S(x)$, we propose a new technique that improves the guess-and-determine algorithm in [11].

4.1 The Matching Phase for the k Good Columns

Let V_i be the set which contains the output vectors from Algorithm 2 with respect to the c_i-th squared LAT column, where $0 \le i < k$. In the matching phase, the Boolean functions $c_0S(x)$, $\cdots$, $c_{k-1}S(x)$ are obtained by searching the vectors in V_i to match the squared LAT applying a basic property of the Hadamard product.

Property 4. 1. $\vec{s}_{b\oplus c} = \vec{s}_b \odot \vec{s}_c$.

2. For $0 \le j < n$, $\pi_j^n(\vec{s}_{b\oplus c}) = \pi_j^n(\vec{s}_b) \odot \pi_j^n(\vec{s}_c)$.

Property 4 is obvious from the definition of $\vec{s}_b$ and the Hadamard product. Combining the first formula in Property 4 with Property 2, we obtain that the $(b \oplus c)$-th column $\vec{\lambda}_{b\oplus c}$ in the LAT can be deduced by $1/2H_n \cdot \vec{s}_{b\oplus c} = 1/2H_n \cdot (\vec{s}_b \odot \vec{s}_c)$. For each two vectors $\vec{u} \in V_i$ and $\vec{v} \in V_j$, the adversary computes a new vector $\vec{w} = 1/2H_n \cdot (\vec{u} \odot \vec{v})$. Then, the adversary can easily detect whether $\vec{u}$ and $\vec{v}$ are consistent with the squared LAT by verifying whether $\vec{w}^\dagger = \vec{\lambda}_{b\oplus c}^\dagger$. We call $\vec{u}$ and $\vec{v}$ a match vector pair if they are consistent with the squared LAT.

Now we discuss the matching phase of the c_i-th column and the c_j-th column, $0 \le i < j < k$. It should be noted that it is not necessary to verify the match for each pair of vectors from V_i and V_j. In the reconstruction problem, our purpose is to find a representative $S(x)$ in the equivalence class $\{S(x \oplus c) \oplus d | c \in \mathbb{F}_2^n, d \in \mathbb{F}_2^m\}$. For example, when the matching phase begins with the c_0-th and c_1-th columns, let us assume that there are j distinct symmetric-equivalence classes in the solution of the sign determination problem of the c_0-th column, i.e., $V_0 = \{\vec{v} | \vec{v} \in [\vec{u}_k], 0 \le k < j\}$. The set of vector pairs which needs to be tested is $\{(\vec{u}_k, \vec{v}) | 0 \leq k < j, \vec{v} \in V_1\}$. When the adversary finds that $\vec{u}_{k_0} \in V_0$ and $\vec{v}_0 \in V_1$ are consistent with the squared LAT, the other matching pairs can be constructed by the second formula in Property 4. Similarly, once the adversary obtains the $c_0 S(x)$ and $c_1 S(x)$ corresponding to the matching vectors, all other Boolean functions can be recovered by the translation $c_0 S(x \oplus c) \oplus d$ and $c_1 S(x \oplus c) \oplus d$ following Property 1.

The number of the match vector pairs between V_i and V_j is related to the number of the Boolean functions which are DDT-equivalent with $(c_i S(x), c_j S(x))$. More precisely, the matching phase over V_i and V_j finds the vectorial Boolean function $G(x)$ from $\mathbb{F}_2^n$ to $\mathbb{F}_2^2$, whose absolute LAT is $(\vec{\lambda}_0^\dagger, \vec{\lambda}_{c_i}^\dagger, \vec{\lambda}_{c_j}^\dagger, \vec{\lambda}_{c_i \oplus c_j}^\dagger)$. Thus, $G(x)$ shares the same DDT with $(c_i S(x), c_j S(x))$. Note that the problem to determine the size of DDT-equivalent class of a Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^2$ is also an open issue.

As the size of DDT-equivalence class is unknown, we restrict the prescribed DDT to be a family of S-boxes for which the DDT-equivalence class is trivial according to the following conjecture proposed in [11].

Conjecture 1. *Suppose that S is a permutation over $\mathbb{F}_2^n$ and the rows of the DDT of S are pairwise distinct. Then, the DDT-equivalence class of S is trivial, i.e., only contains the permutations of the form $S(x \oplus c) \oplus d$, where $c, d \in \mathbb{F}_2^n$.*

The matching phase for k good columns is shown in Algorithm 3 repeating the matching phase of the i-th good column and the $(i+1)$-th good column, $0 \leq i \leq k-2$. For the S-boxes with trivial DDT-equivalence class, one combination is expected to be returned from Algorithm 3. If Conjecture 1 does not hold when the DDT-equivalence class of S is nontrivial, lines 9 and 17 in Algorithm 3 should be removed and the search continues with a set of the match vector pairs.

4.2 The Improved Guess-and-Determine Algorithm

Now we suppose that the adversary has obtained k Boolean functions, i.e., $c_0 S(x)$, ..., $c_{k-1} S(x)$ implementing Algorithm 3, $1 \leq k < m$. We present an improved GD algorithm that takes the DDT table and the k Boolean functions as its inputs and returns a representative of the DDT-equivalence class.

The improved GD algorithm implements the tree-traversal structure of [11]. The improved GD algorithm begins by fixing $S(0)$ to be zero in the initial layer. In the i-th layer, the algorithm determines the possible assignments for $S(i)$, $i = 1, \ldots, 2^n - 1$, by checking the constraints imposed by the DDT. We follow the notations from [11] by denoting the set of possible values for $S(i)$ by $\mathcal{R}_i = \{y \big| \delta(i, y) \neq 0\}$ imposed by the given DDT. It implies that $S(i)$ is in the set $\mathcal{L} = \{x \oplus S(0) | x \in \mathcal{R}_i\} \cap \cdots \cap \{x \oplus S(i-1) | x \in \mathcal{R}_{i \oplus (i-1)}\}$.

In our approach, the knowledge of $c_0 S(x)$, ..., and $c_{k-1} S(x)$ can help us to reduce the size of the set $\mathcal{L}$. For every element $x \in \mathcal{L}$, if any equalities $c_0 x = c_0 S(i)$, $\cdots$, $c_{k-1} x = c_{k-1} S(i)$ does not hold, x is removed from $\mathcal{L}$. The reconstruction process is illustrated in a recursive way in Algorithm 4.

Next, we analyse the time complexity of Algorithm 4 for $1 \leq k < m$. The analysis of the original GD algorithm when $k = 0$ is presented in Appendix D. In the first layer, after discarding the non-consistent values of $S(1)$ based on the DDT, there are at most 2^{n-1} possible values. Similarly, after checking the constraints imposed from $c_0 S(x)$, $\cdots$, and $c_{k-1} S(x)$, there are $2^{n-1} \cdot \frac{1}{2^k} = 2^{n-k-1}$ possible values for $S(0)$. By the i-th layer, the above process is repeated and the number of the possible assignments $S(1), \cdots, S(i)$ on average is

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

where $K = \lceil \frac{n+m-2k-1}{m-n+1} \rceil$. In the $(i+1)$-th layer, there are at most 2^{n-1} possible assignments for $S(i+1)$. For each possible assignment, the adversary checks whether $S(i+1) \oplus$

Algorithm 3 The Matching Phase Given k Good Columns

```
1: Input: the index set of the good columns $C = \{c_0, \dots, c_{k-1}\}$, the corresponding
   solution sets $V_0, \dots, V_{k-1}$ and the squared LAT;
2: Output: $c_0 S(x), \dots, c_{k-1} S(x)$;
3: for each $i \in [0, k-2]$ do
4:     if $i = 0$ then
5:         for each $\vec{u} \in \{\vec{u}_0, \dots, \vec{u}_j\}$ and $\vec{v} \in V_1$ do
6:             $\vec{w} = 1/2 H_n \cdot (\vec{u} \odot \vec{v})$
7:             if $\vec{w}^\dagger = \vec{\lambda}^\dagger_{c_i \oplus c_{i+1}}$ then
8:                 $\vec{p}_0 = \vec{u}, \vec{p}_1 = \vec{v}$
9:                 break        ▷ this line is to be removed if the DDT-equivalence class is
   nontrivial.
10:            end if
11:        end for
12:    else
13:        for each $\vec{v} \in V_{i+1}$ do
14:            $\vec{w} = 1/2 H_n \cdot (\vec{p}_i \odot \vec{v})$
15:            if $\vec{w}^\dagger = \vec{\lambda}^\dagger_{c_i \oplus c_{i+1}}$ then
16:                $\vec{p}_{i+1} = \vec{v}$
17:                break        ▷ this line is to be removed if the DDT-equivalence class is
   nontrivial.
18:            end if
19:        end for
20:    end if
21: end for
22: Deduce $c_0 S(x), \dots, c_{k-1} S(x)$ from $\vec{p}_0, \dots, \vec{p}_{k-1}$
23: return $c_0 S(x), \dots, c_{k-1} S(x)$.
```

Algorithm 4 The Improved Guess-and-Determine Algorithm

```
1: Input: the indices of good columns $c_0, \ldots, c_{k-1}$, the Boolean functions
   $c_0S(x), \cdots, c_{k-1}S(x)$ and the DDT
2: Output: one representative in the DDT-equivalence class
3: $\vec{s}$ is initialized as a vector of $2^m$ zeros.
4: ImprovedGD($\vec{s}, 1$)
5: return $\vec{s}$
6:
7: procedure ImprovedGD($\vec{s}$,$i$)
8:     if $i < 2^m$ then
9:         $\mathcal{L} = \bigcap_{0 \le j < i} \{x \oplus \vec{s}\,[j] | x \in \mathcal{R}_{i \oplus j}, c_0S(i) = c_0 \cdot x, \cdots, c_{k-1}S(i) = c_{k-1} \cdot x\}$
10:    else
11:        if the DDT of $\vec{s}$ matches the given DDT then
12:            return $\vec{s}$
13:        end if
14:    end if
15:    if $L \neq \emptyset$ then
16:        for each $x \in \mathcal{L}$ do
17:            $\vec{s}\,[i] = x$
18:            ImprovedGD($\vec{s}$,$i + 1$)
19:        end for
20:    else
21:        return
22:    end if
23: end procedure
```

Table 1: $\log_2 T_{n,m}(k)$ for $n = 8$ with Different m and k

m \ k	0	1	2	3	4	5	6	7	8	9	10
8	30.71	30.73	24.73	19.83	16.84	16.10	16.00	15.99	-	-	-
9	18.39	21.39	18.40	16.69	16.15	16.03	16.00	15.99	15.99	-	-
10	16.27	18.46	16.91	16.27	16.07	16.02	16.00	15.99	15.99	15.99	-
11	16.05	17.32	16.48	16.16	16.05	16.01	16.00	15.99	15.99	15.99	15.99

Table 2: t_{sd} for the 8-bit S-boxes in Different Block Ciphers

Block Cipher	AES	ARIA	Camellia	SEED-$S0$	SEED-$S1$
t_{sd} (ms)	131.44	131.64	142.11	108.02	106.92
Block Cipher	CAST-256-$S0$	CLEFIA-$S1$	SKIPJACK	Streebog	-
t_{sd} (ms)	340.00	177.56	60.28	56.08	-

$S(1), \ldots, S(i+1) \oplus S(i)$ are consistent with the DDT. The complexity of this process is $1 + 2^{n-m-1} + \cdots + 2^{(n-m-1)(i-1)} < 2$ tests. There are $2^{(n-m-1)i} \cdot W_i(k)$ possible assignments for $S(1), \cdots, S(i+1)$ at this stage. Each assignment should be tested with respect to the constraints $c_0 S(i+1)$, $\cdots$, and $c_{k-1} S(i+1)$. The number of checks on each assignment is also no more than 2. Thus, the time complexity of this layer is $2^n \cdot W_i(k) + 2^{(n-m-1)i+1} W_i(k)$, which is no more than $(2^n + 2) W_i(k)$.

From the K-th layer, $W_i(k) = 1$ and the time complexity of each layer is no more than 2^n. Thus, the time complexity is

$$T_{n,m}(k) = (2^n + 2) \sum_{i=1}^{K} W_i(k) + (2^n - K) \cdot 2^n.$$

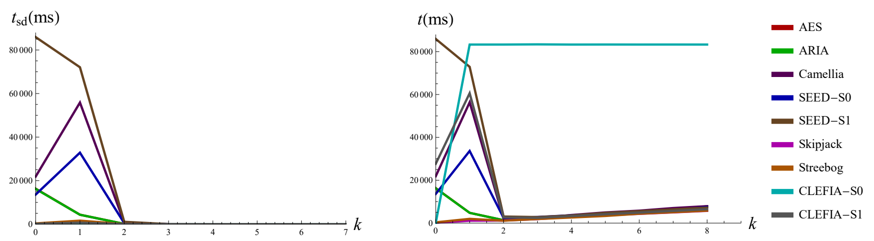
We evaluate the time complexity for the GD phase with $n = 8$ and $8 \leq m \leq 11$. The time complexity for $n = 8$ with different m and k are shown in Table 1. It could be seen that for the 8-bit S-boxes, more than two independent good columns are needed to optimize the original GD algorithm. When m increases, the improvement by the solutions from the good columns is not significant. It should be noted that increasing the size of output of the S-box makes the reconstruct process easier to be archived. Thus, an $n \times m$ S-box with $m \gg n$ is not a significantly secure option when designing a secret non-linear layer for a cryptographic primitive.

5 Experiments

We verified our results by implementing our reconstruction technique on the S-boxes of some existing block ciphers. Our algorithm is implemented in C++ using a g++ compiler with O2 optimization on a single core QEMU Virtual CPU @ 1.995GHZ. For the 8-bit S-boxes, we run the experiments on the DDTs of the S-boxes of several block ciphers, including AES [15], Camellia [5], SEED [17], ARIA [18], SKIPJACK [4], CLEFIA [23], and Streebog [24].

In our experiments for 8-bit S-boxes, Algorithm 2 is applied to solve the sign determination problems with the threshold preset to be 1000. While for many of the tested S-boxes, we found good columns, for the S-box $S0$ of CLEFIA, there exists no good column in its absolute LAT. The average time of solving the sign determination problem of one good column, denoted as t_{sd}, is listed in 2 for different S-boxes. It can be seen that for a good column $\vec{\lambda}_b^{\dagger}$, $\vec{\lambda}_b$ can be recovered in no more than 350ms.

We denote t_{sd} and t as time of the guess-and-determine phase and time of the reconstruction procedure, respectively. The experiment results of t_{sd} and t for the S-boxes are shown in Figure 2 for different k. It can be observed from Figure 2a that the curves of t_{sd} for different S-boxes and the time complexity of the guess-and-determine phase when $m = n = 8$ in Table 1 follow the same trend.



(a) t_{sd} for 8-bit S-boxes (b) t for 8-bit S-boxes

Figure 2: t_{sd} and t for the 8-bit S-boxes

It can be seen from Figure 2b that the most effective way to reconstruct the S-boxes of AES, ARIA, SEED,Camellia, and $S0$ of CLEFIA from their DDT is to solve the sign determination problem of two independent columns and apply Algorithm 4 with the knowledge of two Boolean functions related to the S-box. For example, using the original guess-and-determine algorithm, the reconstruction procedure takes $85,923$ms to recover the $S1$ in SEED from its DDT. However, when the adversary solves the sign determination problem of two independent columns, the reconstruction costs only $2,801$ms. It should be noted that the S-boxes of AES, ARIA, SEED, Camellia and $S0$ of CLEFIA are of 4-differential uniformity. The nonzero entries in the DDTs of these S-boxes are relatively dense.

For other 8-bit S-boxes in our experiments, i.e., the Sboxes of Streebog, Skipjack and $S0$ of CLEFIA, the most effective method to reconstruct the S-box from its DDT is the original GD algorithm. It should be noted that the Sboxes of Streebog, Skipjack and $S0$ in CLEFIA are 8-, 12-, and 10-differential uniform respectively. It indicates that the nonzero entries in the DDTs of these S-boxes are relatively sparse, which reduces the number of the possible assignments for $S(i)$ and accelerate the original guess-and-determine algorithm.

Thus, when the DDT has a low differential uniformity, it is more efficient to reconstruct the S-box with our approach; otherwise one should apply the original guess-and-determine algorithm to recover the S-box.

APN functions is harder to reconstruct in our experiments. We tried our technique on the $S7$ and $S9$ in the block ciphers KASUMI [1], MISTY1 [20], which are designed to be the APN permutations. We found no good columns in the absolute LATs of KASUMI's $S7$ and $S9$ and MISTY1's $S7$ even when we set the threshold $H = 5000$. There are two good columns in the absolute LAT of MISTY1's $S9$. However, the number of good columns is not enough to improve the original guess-and-determine phase. Then, we apply our technique to thirty 7-bit APN functions found by Yu et al. in [26]. It is interesting to note that there are no good columns in the LATs of these functions.

6 Conclusions

In this paper we presented a new algorithm for solving the problem of reconstructing an S-box from its DDT. The new algorithm is more efficient than the standard guess-and-determine algorithm presented before, and can be used in the different scenarios where the problem arises.

Most notably, the new algorithm will allow exploring problems related to DDTs, such as the ability to construct an S-box from a "made up" DDT (i.e., picking the DDT and then constructing an S-box out of it). Such a capability can be used for better (e.g., designing stronger S-boxes with 0 differences in the right place to improve resistence against various cryptanalytic attacks) or for worse (e.g., to design S-boxes with differential backdoor known to the designer only).

Another related open problem is the problem of reconstructing an S-box from its *Boomerang Connection Table*, introduced in [14]. These tables are useful for evaluating the boomerang attack [25], and they depend on the DDT.

References

[1] 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification, V3.1.1 (2001)

[2] Adams, C.M.: Constructing Symmetric Ciphers Using the CAST Design Procedure. Designs, Codes and Cryptography 12(3), 283–316 (Nov 1997)

[3] Adams, C.M.: The CAST-256 Encryption Algorithm (1999), https://tools.ietf.org/html/rfc2612, AES Candidate

[4] Agency(NSA), N.S.: SKIPJACK and KEA Algorithm Specifications

[5] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms — Design and Analysis. In: Stinson, D.R., Tavares, S. (eds.) Selected Areas in Cryptography. pp. 39–56. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[6] Bar-On, A., Biham, E., Dunkelman, O., Keller, N.: Efficient Slide Attacks. Journal of Cryptology 31(3), 641–670 (Jul 2018)

[7] Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology 4(1), 3–72 (Jan 1991)

[8] Biryukov, A., Perrin, L.: On Reverse-Engineering S-Boxes with Hidden Design Criteria or Structure. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 116–140. Springer (2015), http://dx.doi.org/10.1007/978-3-662-47989-6

[9] Blondeau, C., Leander, G., Nyberg, K.: Differential-Linear Cryptanalysis Revisited. Journal of Cryptology 30(3), 859–888 (Jul 2017)

[10] Blondeau, C., Nyberg, K.: New Links between Differential and Linear Cryptanalysis. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. pp. 388–404. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[11] Boura, C., Canteaut, A., Jean, J., Suder, V.: Two Notions of Differential Equivalence on Sboxes. Designs, Codes and Cryptography (Jun 2018)

[12] Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes. http://www.math.univ-paris13.fr/~carlet/chap-fcts-Bool-corr.pdf

[13] Chabaud, F., Vaudenay, S.: Links between Differential and Linear Cryptanalysis. In: De Santis, A. (ed.) Advances in Cryptology — EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings. pp. 356–365. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)

[14] Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang Connectivity Table: A New Cryptanalysis Tool. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 683–714. Springer International Publishing, Cham (2018)

[15] Daemen, J., Rijmen, V.: The Design of Rijndael: AES-the Advanced Encryption Standard. Springer Science & Business Media (2013)

[16] GOST 28147-89: Cryptographic Protection for Data Processing Systems, Cryptographic Transformation Algorithm. Government Standard of the U.S.S.R., Inv. No. 3583, UDC 681.325.6:006.354. (1998), (in Russian)

[17] Internet, K., Agency, S.: SEED 128 Algorithm Specification https://seed.kisa.or.kr/html/egovframework/iwt/ds/ko/ref/[2]_SEED+128_Specification_english_M.pdf

[18] Kwon, D., Kim, J., Park, S., Sung, S.H., Sohn, Y., Song, J.H., Yeom, Y., Yoon, E.J., Lee, S., Lee, J., Chee, S., Han, D., Hong, J.: New Block Cipher: ARIA. In: Lim, J.I., Lee, D.H. (eds.) Information Security and Cryptology - ICISC 2003. pp. 432–445. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)

[19] Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) Advances in Cryptology — EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, May 23–27, 1993 Proceedings. pp. 386–397. Springer Berlin Heidelberg, Berlin, Heidelberg (1994)

[20] Matsui, M.: New Block Encryption Algorithm MISTY. In: Biham, E. (ed.) Fast Software Encryption. pp. 54–68. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)

[21] Nyberg, K.: Differentially Uniform Mappings for Cryptography. In: Helleseth, T. (ed.) Advances in Cryptology — EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, May 23–27, 1993 Proceedings. pp. 55–64. Springer Berlin Heidelberg, Berlin, Heidelberg (1994)

[22] Schneier, B.: Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In: Anderson, R. (ed.) Fast Software Encryption. pp. 191–204. Springer Berlin Heidelberg, Berlin, Heidelberg (1994)

[23] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) Fast Software Encryption. pp. 181–195. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[24] on Technical Regulation, F.A., Metrology: GOST R 34.11-2012: Streebog hash function https://www.streebog.net/

[25] Wagner, D.: The Boomerang Attack. In: Knudsen, L. (ed.) Fast Software Encryption. pp. 156–170. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)

[26] Yu, Y., Wang, M., Li, Y.: A Matrix Approach for Constructing Quadratic APN Functions. Designs, Codes and Cryptography 73(2), 587–600 (Nov 2014), https://doi.org/10.1007/s10623-014-9955-3

Appendix A Proof of Theorem 2

Let $\vec{v}$ be a vector of length 2^ℓ and $\vec{v} = E_{\ell-1}(\vec{p}, \vec{q})$. Before we prove Theorem 2, the vectors which generate $\pi_j^\ell(\vec{v})$ in the $(\ell-1)$-th layer are shown in Lemma 1, $0 \leq j < \ell$.

Lemma 1. *When* $0 < j < \ell$, $\pi_j^\ell(\vec{v}) = E_{\ell-1}(\pi_{j-1}^{\ell-1}(\vec{p}), \pi_{j-1}^{\ell-1}(\vec{q}))$*; when* $j = 0$, $\pi_0^\ell(\vec{v}) = E_{\ell-1}(\vec{p}, -\vec{q})$.

Proof. By the definition of operation $E_{\ell-1}$,

$$\begin{aligned}\vec{p}^{\,T} &= (v_0 + v_1, v_2 + v_3, \ldots, v_{2^{\ell+1}-2} + v_{2^{\ell+1}-1})\\ \vec{q}^{\,T} &= (v_0 - v_1, v_2 - v_3, \ldots, v_{2^{\ell+1}-2} - v_{2^{\ell+1}-1}).\end{aligned}$$

When $1 \le j < \ell$, the vectors that generate $\pi_j^\ell(\vec{v})$ are

$$\begin{aligned}\vec{p'} = (&v_{2^j} + v_{2^j+1}, \ldots, v_{2^{j+1}-2} + v_{2^{j+1}-1}, v_0 + v_1, \ldots, v_{2^j-2} + v_{2^j-1}, \ldots,\\ &v_{2^\ell-2^j} + v_{2^\ell-2^j+1}, \ldots, v_{2^\ell-2} + v_{2^\ell-1},\\ &v_{2^\ell-2^{j+1}} + v_{2^\ell-2^{j+1}+1}, \ldots, v_{2^\ell-2^j-2} + v_{2^\ell-2^j-1})^T,\\ \vec{q'} = (&v_{2^j} - v_{2^j+1}, \ldots, v_{2^{j+1}-2} - v_{2^{j+1}-1}, v_0 - v_1, \ldots, v_{2^j-2} - v_{2^j-1}, \ldots,\\ &v_{2^\ell-2^j} - v_{2^\ell-2^j+1}, \ldots, v_{2^\ell-2} - v_{2^\ell-1},\\ &v_{2^\ell-2^{j+1}} - v_{2^\ell-2^{j+1}+1}, \ldots, v_{2^\ell-2^j-2} - v_{2^\ell-2^j-1})^T.\end{aligned}$$

It follows from the definition of π_j^ℓ that $\vec{p'} = \pi_{j-1}^{\ell-1}(\vec{p})$ and $\vec{q'} = \pi_{j-1}^{\ell-1}(\vec{q})$. Thus, $\pi_j^\ell(\vec{v}) = E_{\ell-1}(\pi_{j-1}^{\ell-1}(\vec{p}), \pi_{j-1}^{\ell-1}(\vec{q}))$. For the case when $j = 0$, it can be easily verified that $\pi_0^\ell(\vec{v}) = E_{\ell-1}(\vec{p}, -\vec{q})$. □

Proof. Let us consider the case when vector $\vec{v}$ is j-symmetric to $\vec{u}$, $0 \le j < l$. For the positive case in Definition 2, we first prove inductively that for each $j' \le j$, $\pi_{j'}^\ell(\vec{v}) \in F_\ell[i]$. The negative case in Definition 2 can be proved with the similar method.

The statement is true when $j' = 0$. It follows from Lemma 1 that if $\vec{v} = E_{\ell-1}(\vec{p}, \vec{q})$, then $\pi_0^\ell(\vec{v}) = E_{\ell-1}(\vec{p}, -\vec{q})$. As $-\vec{q} \in F_{\ell-1}[i + 2^{n-\ell+1}]$, $\pi_0^\ell(\vec{v}) \in F_\ell[i]$. Assume that when $j' = k$, $\pi_k^\ell(\vec{v}) \in F_\ell[i]$. When $j' = k+1$, it can be seen that $\pi_{k+1}^\ell(\vec{v}) \in F_\ell[i]$. From Lemma 1, if $\vec{v} = E_{\ell-1}(\vec{p}, \vec{q})$, $\pi_{k+1}^\ell(\vec{v}) = E_{\ell-1}(\pi_k^{\ell-1}(\vec{p}), \pi_k^{\ell-1}(\vec{q}))$. As $\vec{p} \in F_{\ell-1}[i]$ and $\vec{q} \in F_{\ell-1}[i + 2^{n-\ell+1}]$, it follows from the assumption directly that $\pi_k^{\ell-1}(\vec{p}) \in F_{\ell-1}[i]$ and $\pi_k^{\ell-1}(\vec{q}) \in F_{\ell-1}[i+2^{n-\ell+1}]$. It can be concluded that $\pi_{k+1}^\ell(\vec{v})$ is in $F_\ell[i]$. Therefore, it can be observed that if $\vec{v} \in F_\ell[i]$, $\pi_{j_{p-1}}^\ell \circ \ldots \circ \pi_{j_0}^\ell(\vec{v}) \in F_\ell[i]$, for any $0 \le j_0 < \cdots < j_{p-1} = j,\ p \ge 1$.

For the case when $j = \ell$, the positive case is trivial as $\vec{u} = \vec{v}$. The negative case is proved inductively. When $\ell = 0$, the statement is true: if $\lambda^\dagger(i, b) \ne 0$, $F_0[i] = \{\lambda^\dagger(i, b), -\lambda^\dagger(i, b)\}$; otherwise, $F_0[i] = \{0\}$. Assume that the proposition holds when $\ell = k$. When ℓ takes $k+1$, if $\vec{v} = E_k(\vec{p}, \vec{q})$, then $-\vec{v} = E_k(-\vec{p}, -\vec{q})$, where $\vec{p} \in F_k[i]$ and $\vec{q} \in F_k[i + 2^{n-k}]$. From the assumption, $-\vec{p}$ is in $F_k[i]$ and $-\vec{q}$ is in $F_k[i + 2^{n-k}]$. Thus, $-\vec{v}$ is in $F_{k+1}[i]$. □

Appendix B Proof of Theorem 3

Proof. We define the statement of Theorem 3 as $D(j, \ell)$. We only prove the positive case in Definition 2 inductively for j and ℓ. The negative case can also be proved using the similar method. Let $\vec{p} \in F_{\ell-1}[i]$ and $\vec{q} \in F_{\ell-1}[i + 2^{n-\ell}]$ be the vectors such that $\vec{u} = E_\ell(\vec{p}, \vec{q})$.

We claim that $D(0, \ell)$ is true for $j = 0$ and $1 \le \ell \le n$. If $\vec{u} \in F_\ell[i]$ is a 0-symmetric vector, it indicates that $\vec{u} = (u_0, u_0, \ldots, u_{2^{\ell-1}-1}, u_{2^{\ell-1}-1})$. The vector in $F_{\ell-1}[i + 2^{n-\ell+1}]$ that generate $\vec{u}$ is the zero vector. Let us trace back to the initial values in the 0-th layer which generate the zero vector in $F_{\ell-1}[i + 2^{n-\ell+1}]$. These initial values are zero and the zero vector is the only vector in $F_{\ell-1}[i + 2^{n-\ell+1}]$. It can be seen that for each vector $\vec{v} \in F_\ell[i]$, $\vec{v}$ can be presented as $E_{\ell-1}(\vec{p}, \vec{0})$, where $\vec{p}$ is in $F_{\ell-1}[i]$. It can be verified that $\vec{v}$ is a 0-symmetric vector.

Suppose that the statement $D(r, k)$ holds for each $r < k$. Then, it can be proved that for $D(r+1, k+1)$ is also true. In this case $\vec{u} \in F_{k+1}[i]$ is an $(r+1)$-symmetric vector. There are two scenarios in this situation.

The first scenario is that there exist p permutations in Π_{k+1} such that $\pi^{k+1}_{j_{p-1}} \circ \ldots \circ \pi^{k+1}_{j_1} \circ \pi^{k+1}_{j_0}(\vec{u}) = \vec{u}$, where $p \geq 1$ and $0 = j_0 < j_1 < \cdots < j_{p-1} = r+1$. Then, $\pi^k_r \circ \ldots \circ \pi^k_{j_1-1}(\vec{p}) = \vec{p}$ and $\pi^k_r \circ \ldots \circ \pi^k_{j_1-1}(\vec{q}) = -\vec{q}$, which indicates that $\vec{p}$ and $\vec{q}$ are r-symmetric vectors. It can be concluded that each vector in $F_k[i]$ and $F_k[i+2^{n-k+1}]$ are r-symmetric. Thus, each vector in $F_{k+1}[i]$ is $(r+1)$-symmetric vector.

The second scenario is that there exist p permutations in Π_{k+1} such that $\pi^{k+1}_{j_{p-1}} \circ \ldots \circ \pi^{k+1}_{j_1} \circ \pi^{k+1}_{j_0}(\vec{u}) = \vec{u}$, where $p \geq 1$ and $0 < j_0 < j_1 < \cdots < j_{p-1} = r+1$. It can be seen that $\pi^k_r \circ \ldots \circ \pi^k_{j_0-1}(\vec{p}) = \vec{p}$ and $\pi^k_r \circ \ldots \circ \pi^k_{j_0-1}(\vec{q}) = \vec{q}$. It can also be concluded that the vectors in $F_{k+1}[i]$ are $(r+1)$-symmetric vectors.

As mentioned above, the statement is true for $j = 0$ and each $1 \leq \ell \leq n$. For each j and ℓ such that $0 \leq j < \ell \leq n$, starting from the statement $D(0, \ell - j)$, $D(j, \ell)$ can be proved by applying the inductive process above. Thus, Theorem 3 is proved. □

Appendix C Proof of Theorem 4

Proof. For the first statement, we assume that $-\vec{v} \in M_\ell[i + 2^{n-\ell-1}]$. $\vec{u}$ is an arbitrary vector in $C_\ell[i]$ and we denote $\vec{w}$ as $E_\ell(\vec{u}, \vec{v})$. It implies from Lemma 1 that $\pi^{\ell+1}_0(\vec{w}) = E_\ell(\vec{u}, -\vec{v}) \in C_{\ell+1}[i]$. This contradicts the fact that every two vectors in $C_{\ell+1}[i]$ cannot be 0-symmetric with each other. Thus, $-\vec{v} \notin M_\ell[i + 2^{n-\ell-1}]$.

Now we prove the second statement inductively. The statement is true when $j = 0$. We assume that there are two distinct vectors $\vec{v}_1$ and $\vec{v}_2$ in $M_\ell[i + 2^{n-\ell-1}]$ such that $\vec{v}_1$ is positive 0-symmetric to $\vec{v}_2$, i.e., $\pi^\ell_0(\vec{v}_1) = \vec{v}_2$. We only prove the positive case and the proof is the same when $\vec{v}_1$ is negative 0-symmetric to $\vec{v}_2$ It implies that for an arbitrary vector $\vec{u} \in C_\ell[i]$, $\vec{w}_1 = E_\ell(\vec{u}, \vec{v}_1) \in C_{\ell+1}[i]$ and $\vec{w}_2 = E_\ell(\vec{u}, \vec{v}_2) = E_\ell(\pi^\ell_0(\vec{u}), \pi^\ell_0(\vec{v}_1)) = \pi^{\ell+1}_1(E_\ell(\vec{u}, \vec{v}_1)) \in C_{\ell+1}[i]$. It is a contradiction because $\vec{w}_1, \vec{w}_2 \in C_{\ell+1}[i]$ and $\vec{w}_1$ is 1-symmetric to $\vec{w}_2$.

Suppose that the second statement in Theorem 4 is true when $j \leq k$. Now we prove the claim when $j = k+1$. Similarly, we assume for the sake of contradiction that there are two distinct vectors $\vec{v}_1$ and $\vec{v}_2$ in $M_\ell[i + 2^{n-\ell-1}]$ such that $\vec{v}_1$ is positive $(k+1)$-symmetric to $\vec{v}_2$. There exist p permutations in Π_ℓ such that $\pi^\ell_{j_{p-1}} \circ \ldots \circ \pi^\ell_{j_1} \circ \pi^\ell_{j_0}(\vec{v}_1) = \vec{v}_2$, where $0 \leq j_0 < j_1 < \cdots < j_{p-1} = k+1$ and $p \geq 1$. For an arbitrary vector $\vec{u}$ in $C_\ell[i]$, as $\vec{u}$ is $(k+1)$-symmetric, there are q permutations in Π_ℓ such that $\pi^\ell_{k_{q-1}} \circ \ldots \circ \pi^\ell_{k_1} \circ \pi^\ell_{k_0}(\vec{u}) = \vec{u}$, where $0 \leq k_0 < k_1 < \cdots < k_{p-1} = k+1$ and $q \geq 1$. We denote $\vec{w}_1 \in C_{\ell+1}[i]$ as $E_\ell(\vec{u}, \vec{v}_2)$. It follows from Lemma 1 that

$$\begin{aligned}
\vec{w}_1 &= E_\ell(\pi^\ell_{k_{q-1}} \circ \ldots \circ \pi^\ell_{k_1} \circ \pi^\ell_{k_0}(\vec{u}), \vec{v}_2) \\
&= \pi^{\ell+1}_{k_{q-1}+1} \circ \ldots \circ \pi^{\ell+1}_{k_1+1} \circ \pi^{\ell+1}_{k_0+1}(E_\ell(\vec{u}, \vec{v}_3)),
\end{aligned}$$

where $\vec{v}_3 = \pi^\ell_{k_{q-1}} \circ \ldots \circ \pi^\ell_{k_1} \circ \pi^\ell_{k_0} \circ \pi^\ell_{j_{p-1}} \circ \ldots \circ \pi^\ell_{j_1} \circ \pi^\ell_{j_0}(\vec{v}_1)$. It is fact that $\vec{v}_3$ is k'-symmetric to $\vec{v}_1$, $k' \leq k$. $\vec{v}_3 \in M_\ell[i + 2^{n-\ell-1}]$ because the vectors in $C_\ell[i]$ are not k'-symmetric. It reaches a contradiction as both $\vec{w}_1$ and $E_\ell(\vec{u}, \vec{v}_3)$ are in $C_{\ell+1}[i]$ and $\vec{w}_1$ is $(k+2)$-symmetric to $E_\ell(\vec{u}, \vec{v}_3)$. Thus, the second claim is true when $j = k+1$. The vectors which are j-symmetric to $\vec{v}$ are not in $M_\ell[i + 2^{n-\ell-1}]$, $0 \leq j < \ell$. □

Appendix D the Time Complexity of the Original Guess-and-Determine Algorithm

The original guess-and-determine algorithm in [11] also returns a representative S in the set $\{S(x \oplus c) \oplus d \big| c \in \mathbb{F}_2^n, d \in \mathbb{F}_2^m\}$. To archive so, the adversary can fix $S(0)$ to be zero and fix $S(1)$ to be any value in $\mathcal{R}_i$. Thus, there is one possible case after the first layer.

Similar to the analysis in Section 4.2, the number of the possible cases at the end of the i-th layer is

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

where K is the positive root of the formula $(m-n+1)x^2+(n-m-1)x-2(n-1)=0$. In the $(i+1)$-th layer, the adversary need to check the consistency of $2^{n-1}W_i$ cases with respect to the DDT. The complexity of the $(i+1)$-th layer is no more than $2^n W_i$. As the algorithm starts from searching the assignment of $S(2)$, the time complexity of the original guess-and-determine algorithm is

$$T_{n,m}(0) = 2^n \sum_{i=1}^{2^n-2} W_i.$$