Finding All Impossible Differentials When Considering the DDT

Kai Hu[1], Thomas Peyrin[1], and Meiqin Wang[2,3,4](✉)

[1] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. kai.hu@ntu.edu.sg, thomas.peyrin@ntu.edu.sg
[2] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China. mqwang@sdu.edu.cn
[3] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Qingdao, Shandong, China.
[4] Quan Cheng Shandong Laboratory, Jinan, China.

Abstract. Impossible differential (ID) cryptanalysis is one of the most important attacks on block ciphers. The Mixed Integer Linear Programming (MILP) model is a popular method to determine whether a specific difference pair is an ID. Unfortunately, due to the huge search space (approximately 2^{2n} for a cipher with a block size n bits), we cannot leverage this technique to exhaust all difference pairs, which is a well-known long-standing problem.

In this paper, we propose a systematic method to find all IDs for SPN block ciphers. The idea is to partition the whole difference pair space into lots of small disjoint sets, each of which has a representative difference pair. All difference pairs in one small set are possible if its representative pair is possible, and this can be conveniently checked by the MILP model. In this way, the overall search space is drastically reduced to a practical size by excluding the sets containing no IDs. We then examine the remaining difference pairs to identify all IDs (if some IDs exist). If our method cannot find any ID, the target cipher is proved free of ID distinguishers.

Our method works especially well for SPN ciphers with block size 64. We apply our method to SKINNY-64 and successfully find all 432 and 12 truncated IDs (we find all IDs but all of them can be assembled into certain truncated IDs) for 11 and 12 rounds, respectively. We also prove, for the first time, that 13-round SKINNY-64 is free of ID distinguishers even when considering the differential transitions through the Difference Distribution Table (DDT). Similarly, we find all 12 truncated IDs (all IDs are assembled into 12 truncated IDs) for 13-round CRAFT and prove there is no ID for 14 rounds. For SbPN cipher GIFT-64, we prove that there is no ID for 8 rounds.

For SPN ciphers with larger block sizes, we show that our idea is also useful to strengthen the current search methods. For example, if we consider the Sbox to be ideal and only consider the branch number information of the diffusion matrix, we can find all 6,750 truncated IDs for 6-round Rijndael-192 in 1 second and prove that there is no truncated ID for 7 rounds. Previously, we need to solve approximately 2^{48} MILP models to

achieve the same goal. For GIFT-128, we exhausted all difference patterns that have an active superbox in the plaintext and ciphertext and proved there is no ID of such patterns for 8 rounds.
Although we have searched for a larger or even full space for IDs, no longer ID distinguishers have been found. This implies the reasonableness of the intuition that a small number (usually one or two) of active bits/words at the beginning and end of an ID will be the longest.

Keywords: Impossible Differential, MILP, SKINNY, CRAFT, GIFT, Rijndael-192

1 Introduction

The impossible differential (ID) attack [16,5] is one of the most important attacks for block ciphers. This attack exploits a pair of input and output differences (Δ_i, Δ_o) of a (round-reduced) cipher E_K that cannot be connected for any K. Namely, two plaintexts p, p' satisfying $p \oplus p' = \Delta_i$ never satisfy $E_K(p) \oplus E_K(p') = \Delta_o$. Such difference pair (Δ_i, Δ_o) is then called an ID. The ID attack has been one of the most powerful cryptographic attacks nowadays. For example, it is the first attack that could break 7 rounds of AES-128 [17] and remains the best attack on reduced SKINNY-64 under the single-tweakey setting [13].

In the early days, an ID (Δ_i, Δ_o) was detected by the miss-in-the-middle approach manually [6] and the details of the Sboxes are usually ignored. The first automated search attempt appeared in [5] with so-called *shrink* technique. It shrinks the word size to 3 bits and find impossible differentials of the global structure of the cipher by exhaustively testing all possible differences and values. This method is only applicable to those ciphers consisting of a small number of words with a big word size, so it doesn't work for many modern-day ciphers such as SKINNY [3], CRAFT [4] or GIFT [1], *etc.* In [15], Kim et al. presented a new automated tool called $\mathcal{U}$-method. To detect if (Δ_i, Δ_o) is impossible, one first propagates Δ_i forwards by r_1 rounds and checks the status of the difference of each output word (known active, active, inactive, or unknown). Then he/she propagates Δ_o backward by r_2 rounds and checks the status again. Finally, if any contradiction occurs, (Δ_i, Δ_o) is an $(r_1 + r_2)$-round ID. Several extensions of the $\mathcal{U}$-method have been done such as the UID-method by Luo et al. [18] and the method proposed by Wu and Wang [26]. Recently, a constraint-programming-aided version of the $\mathcal{U}$-method called $\mathcal{U}^\star$-method has been developed by Sun *et al.* [22], which allows exhausting all possible plaintext and ciphertext difference patterns automatically. All these methods above focus on truncated IDs, *i.e.*, the contradictions inside the Differential Distribution Tables (DDT) of corresponding Sboxes are not considered. Consequently, we have no way of knowing if we would have gotten more if the information of the DDTs is taken into consideration.

Several attempts focus on the upper bound on the rounds of IDs. At EUROCRYPT 2016, Sun et al. [21] used the *primitive index* of the characteristic matrix of the linear layer to give upper bounds on the length of IDs for some special Substitution-Permutation-Networks (SPN) block ciphers with the detail of the Sbox omitted. They proved that under some special conditions, the existence of IDs relies on the existence of low-weight IDs [21, Theorem 1]. In [25], by using linear algebra the authors gave a practical method that could give upper bounds on the length of IDs for any SPN block cipher when omitting the differential property of the Sboxes. Currently, all systematical methods for bounding the length of IDs are all without considering the Sbox details.

Another line of detecting IDs starts independently from [20,10]. The MILP model for searching for differential characteristics is simply modified by adding additional specific constraints on the plaintext and ciphertext differences. If the model is infeasible, the corresponding plaintext and ciphertext difference pair is an ID. Compared to the previous methods, this method can detect all kinds of contradictions (with the assumption that the round keys are uniformly random, which is a default assumption of this paper). However, since the constraints on the plaintext/ciphertext differences are fixed, the number of models we need to solve is equivalent to the number of difference pairs we want to check. Exhaustively checking all plaintext and ciphertext difference pairs is clearly computationally infeasible. Actually, for a cipher with block size n, the search space is as large as 2^{2n}. Based on the intuition that the longest IDs are usually caused by difference pairs with a small number of active bits or words in both plaintext and ciphertext ends, users of this model prefer to test only a small proportion of the difference pairs with one or two active bits or words for plaintext and ciphertext differences. Nowadays, the model has been very popular in measuring the security strength of newly designed ciphers against ID attacks. For example, the designers of GIFT [1] took it to prove that there does not exist any ID with one-active nibble against 7 rounds of GIFT-64. The designers of CRAFT searched for IDs with plaintext and ciphertext differences having at most two active nibbles and they found twelve 13-round IDs [4].

Apart from these works, it is worth mentioning that in [24] Wang and Jin proved that there is no ID for 5-round AES even with the information of the DDT based on some careful mathematical analyses. But unfortunately, this method is specifically designed for AES only. Generally speaking, the MILP method is much more convenient than other methods, since it only needs some slight modifications from the MILP models for searching for differential characteristics. However, as we mentioned, all current MILP models can check a small number of the difference pairs. How to tackle the huge search space has been a long-standing problem.

Contributions. In this paper, we propose a systematic method based on the MILP model to find all IDs in the whole search space. As mentioned above, to exhaust all input and output difference pairs requires a complexity of 2^{2n} which is infeasible. Our method delicately partitions the whole search space and efficiently excludes those containing no IDs. The search space is then reduced to a reasonable size. Finally, the remaining IDs (if they exist) can be detected with the plain MILP models. If our method finds no IDs for the r-round cipher, we know that there exists no ID for this r-round cipher. The provable security is thus achieved.

Our method is efficient for SPN ciphers with a block size equal to 64. For SKINNY-64, we find all IDs for 11 and 12 rounds in 4 and 1.5 hours, respectively. Interestingly, all these IDs can be assembled into 432 and 12 truncated IDs for 11 and 12 rounds. We also prove, for the first time, that the 13-round SKINNY-64 is free of ID distinguishers *with considering the DDT information*. Similarly, we find all 2,700 IDs for 13-round CRAFT which is equivalent to 12 truncated IDs and prove there is no ID for 14 rounds. For Substitution Bit-Permutation Network (SbPN) cipher GIFT-64, we prove that there is no ID for 8 rounds.

Our method is also useful to improve the current search strategies for ciphers with large blocks. We show its usage in applications to Rijndael-192 and GIFT-128 as examples. For Rijndael-192, we search for IDs under the arbitrary Sbox/MC mode, *i.e.*, only the activeness of an Sbox and the branch number of the MixColumn operation would be considered (which is inspired by the arbitrary Sbox model [20]). In this scenario, we show that all 6,750 truncated IDs of 6-round Rijndael-192 can be identified in 1 second, and prove there is no truncated ID for 7 rounds. In previous methods, we need to solve approximately 2^{48} plain MILP models to achieve this. For GIFT-128, we search for IDs that have one active superbox in both plaintext and ciphertext differences. In previous methods, we need to solve 2^{38} plain MILP models, now with our new tool, we only need to handle 4,608 MILP models. We prove that there is no ID with one active superbox in both ends for 8-round GIFT-128. We list all our application results in Table 1 for readers' quick reference.

Implications of Finding All IDs. On the one hand, if our new model finds no ID for a (round-reduced) cipher, we achieve a more thorough security proof for the cipher against ID distinguishers compared to [20]. On the other hand, it is also useful to list all IDs for a (round-reduced) cipher. Firstly, different IDs would affect the concrete attacking phase as well as the data and time complexity. In terms of the ID distinguishers, more active bits in the output mean less data/time complexities. In terms of the key recovery attacks, IDs with good input and output difference patterns may have a better performance, *e.g.*, the ID attacks on AES were improved with alternative IDs [17]. Secondly, finding all IDs (with or without considering the DDT) is a long-standing challenge in cryptanalysis and cipher design. Finding out all IDs can help us understand better the structure of target ciphers and the ID attack itself.

We highlight that all IDs we discuss in this paper are under the assumption that the round keys are uniformly random. All source codes of this work are provided in the anonymous git repository `https://github.com/hukaisdu/SearchForID.git` to help readers understand our tool better.

Organization of the remaining paper. In Section 2, we introduce the notations and some global settings used in this paper. In Section 3, we show how to partition the whole search space and quickly exclude those containing no IDs and identify all IDs in the remaining candidates. Applications to SKINNY-64, CRYFT and GIFT-64 are presented in Section 4 and Section 5. In Section 6,

Table 1: The application results of this paper. N in the #ID column means no IDs. #Space is the size of the whole search space with plain MILP models.

Cipher	#Space	#Round	#ID	Time	Remarks
SKINNY-64	2^{128}	11	432	4h	All IDs can be assembled into 432 truncated IDs
		12	12	1.5h	All IDs can be assembled into 12 truncated IDs
		13	N	4h	No IDs with the DDT considered
CRAFT	2^{128}	13	12	7d	All IDs can be assembled into 12 truncated IDs
		14	N	7d	No IDs with the DDT considered
GIFT-64	2^{128}	8	N	17h	No IDs with the DDT considered
Rijndael-192	2^{48}	6	6,750	1s	Truncated IDs in arbitrary Sbox/MC model†
		7	N	1s	No Truncated IDs in arbitrary Sbox/MC model
GIFT-128	2^{38}	8	N	30h	No IDs with one-active superbox with the DDT considered

† Arbitrary Sbox/MC model: we only consider the activeness of the Sbox and branch number of the MixColumn

we discuss how to apply our idea to enhance some traditional search strategies based on MILP for large-size ciphers. In Section 7 we conclude our paper and highlight two future works.

2 Preliminaries

2.1 Notations and Definitions

In this paper, we are only interested in the differences, so the differences are directly represented by lowercase letters such as x rather than conventional Δx. Consider a (round-reduced) cipher E, if (x, y) is an ID over E, we write it as $x \overset{E}{\nrightarrow} y$. Conversely, $x \xrightarrow{E} y$ means x can propagate to y over E, *i.e.*, (x, y) is a possible pattern. We use uppercase letters to represent the sets of differences such as X and Y. $X \xrightarrow{E} Y$ means for all $(x, y) \in X \otimes Y$ we have $x \xrightarrow{E} y$. $x \xrightarrow{E} X$ is equivalent to $\{x\} \xrightarrow{E} X$. Similarly, $X \xrightarrow{E} y$ is equivalent to $X \xrightarrow{E} \{y\}$. $X \otimes Y$ (sometimes we use $X \bigotimes Y$ for a better looking of a complicated equation) is defined as $\{(x, y) : x \in X, y \in Y\}$. If $X \cap Y = \varnothing$, we would write $X \cup Y$ as

$$x_0/X_0 \longrightarrow \boxed{E_0} \longrightarrow x_1/X_1 \longrightarrow \boxed{\quad E_1 \quad} \longrightarrow x_2/X_2 \longrightarrow \boxed{E_2} \longrightarrow x_3/X_3$$

Fig. 1: The global settings of our theoretical model.

$X + Y$ to highlight $X \cap Y = \varnothing$. Generally, if $\bigcap_i X_i = \varnothing$, the union set of all X_i is written as $\sum_i X_i$. $X - Y$ is defined as $\{x \in X : x \notin Y\}$.

Global Settings. A modern block cipher usually iterates a simple round function many times with different round keys. So we can always decompose a (round-reduced) cipher into three parts, say $E = E_2 \circ E_1 \circ E_0$. We denote the input difference/set of E_0, E_1 and E_2 by $x_0/X_0, x_1/X_1$ and x_2/X_2 respectively, and output difference/set of E by x_3/X_3. See Figure 1 for details of the settings. In the remaining paper, if we do not specify x_0, x_1, x_2, x_3 and X_0, X_1, X_2, X_3, they denote the difference or sets as defined here.

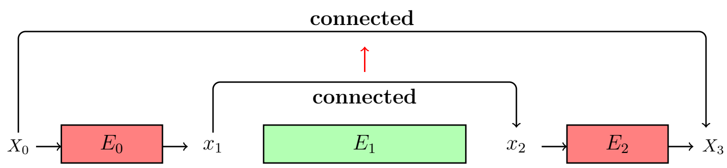
2.2 Current MILP Model for Detecting IDs

In [20,10,9], the MILP models for detecting IDs are independently proposed. This method is developed from the MILP models for searching for differential characteristics [19,23] by adding some additional constraints on the plaintext and ciphertext differences. To construct the MILP model for checking if a given difference pair (Δ_i, Δ_o) for a cipher E is impossible, we first declare a sequence of variables to represent input and output differences for all components of E such as Sboxes and linear layers. Next, we use inequalities to force these variables to be legal patterns that are compatible with the differential propagation rules of the corresponding components. Thus, any solutions satisfying these constraints are legal differential characteristics. Additionally, suppose the variables representing the differences of plaintext and ciphertext are u_0 and u_r, we add two more constraints as

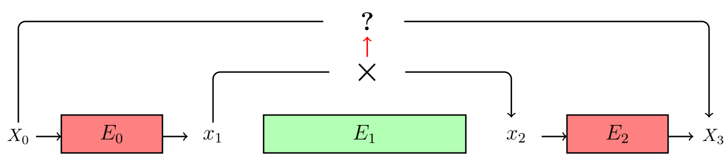
$$u_0 = \Delta_i,\ u_r = \Delta_o.$$

If the overall MILP model is feasible, there is one differential characteristic propagating from Δ_i to Δ_o, *i.e.*, (Δ_i, Δ_o) is a possible differential. Otherwise, (Δ_i, Δ_o) is an ID.

According to different methods in which we use inequalities to describe the differential propagations over an Sbox or linear layer, the capabilities to detect IDs of the corresponding MILP models are also different. For example, if details of the DDT and linear layers are all described, then all kinds of contradictions could be detected. This is the default mode we use in this paper. If only the information that an Sbox is active or not and the branch number of a linear layer is described in the MILP search model, truncated IDs could be detected. We refer to such a model as the *arbitrary Sbox/MC model*. In this paper, the application to Rijndael-192 is the only instance using this mode. We will assume that the readers of this paper have been familiar with the plain MILP models for detecting IDs. Or we refer the readers to [20,10] for more details of this topic.



(a) If $x_1 \xrightarrow{E_1} x_2$, then $X_0 \xrightarrow{E} X_3$



(b) If $x_1 \overset{E_1}{\nrightarrow} x_2$, $X_0 \otimes X_3$ has the potential to contain IDs

Fig. 2: The illustration of Proposition 1 and the implication.

3 Finding All Impossible Differentials

Taking the MILP model in [20,10], we need to solve 2^{2n} models for a cipher with block size n bits to check all input and output difference pairs. The search space is obviously too large. So we partition the whole search space into many smaller sets and then process each set one by one to exclude those containing no IDs. For the remaining smaller sets that have the potential to contain IDs, we apply several methods to identify all IDs contained by them.

Main Idea. To determine if (x_0, x_3) is possible or not, we try to find a pair of (x_1, x_2) satisfying

$$x_0 \xrightarrow{E_0} x_1 \xrightarrow{E_1} x_2 \xrightarrow{E_2} x_3.$$

Obviously, if such (x_1, x_2) exists, (x_0, x_3) is possible. Further, if we have known $x_1 \xrightarrow{E_1} x_2$, then all (x_0, x_3) satisfying (1) $x_0 \xrightarrow{E_1} x_1$ (2) $x_2 \xrightarrow{E_1} x_3$ are possible. We have the following proposition (also illustrated by Figure 2a),

Proposition 1. *Let $E = E_2 \circ E_1 \circ E_0$, $X_0 \subseteq \mathbb{F}_2^n$ be a set of differences satisfying $X_0 \xrightarrow{E_0} x_1$ and $X_3 \subseteq \mathbb{F}_2^n$ satisfying $x_2 \xrightarrow{E_2} X_3$. If $x_1 \xrightarrow{E_1} x_2$, then $X_0 \xrightarrow{E} X_3$.*

The proof is obvious from above analyses, so we omit it here. If $x_1 \overset{E_1}{\nrightarrow} x_2$, we cannot predict anything so we say $X_0 \otimes X_3$ has the potential to contain IDs, see Figure 2b. We conclude it into a corollary of Proposition 1 for better understanding this fact.

Corollary 1. *With the same notations as Proposition 1, if $(x_0, x_3) \in X_0 \otimes X_3$ is an ID, then all x_1 and x_2 satisfying $X_0 \xrightarrow{E_0} x_1$ and $x_2 \xrightarrow{E_2} X_3$ cannot be connected, i.e., (x_1, x_2) must satisfy $x_1 \overset{E_1}{\nrightarrow} x_2$.*

To make our method more general, we assume we study a question how to find all IDs in the sets $X_0 \otimes X_3$ over E, where X_0 and X_3 can be subsets of $\mathbb{F}_2^n$. Thus, now we want to find all $(x_0, x_3) \in X_0 \otimes X_3$ satisfying $x_0 \overset{E}{\nrightarrow} x_3$.

Our method consists of three steps:

(1) Partition the whole set $X_0 \otimes X_3$ into many non-overlapping smaller sets, *i.e.*,

$$X_0 \otimes X_3 = \sum_i \sum_j X_0^i \otimes X_3^j, \text{ where } X_0 = \sum_i X_0^i,\ X_3 = \sum_j X_3^j$$

For each pair i, j, we require that there always exist x_1^i and x_2^j satisfying $X_0^i \xrightarrow{E_0} x_1^i$ and $x_2^j \xrightarrow{E_2} X_3^j$, respectively;

(2) Exhaustively check all possible (x_1^i, x_2^j) pairs to see if $x_1^i \xrightarrow{E_1} x_2^j$ by MILP models introduced in [20,10]. $X_0^i \otimes X_3^j$ contains no IDs if $x_1^i \xrightarrow{E_1} x_2^j$, and otherwise has a potential to contain some IDs;

(3) Process those $X_0^i \otimes X_3^j$ that have the potential to contain IDs one by one to identify all IDs with specific strategies that we will introduce later.

3.1 Partition: A Theoretical Viewpoint

In this paper, we always assume that E_0 and E_2 are non-linear functions, so there exists an expansion property for the difference propagation over E_0 and E_2. Consequently, it is possible for us to find two smaller sets X_1 and X_2 satisfying

(1) $\forall x_0 \in X_0$, $\exists x_1 \in X_1$ *s.t.* $x_0 \xrightarrow{E_0} x_1$,

(2) $\forall x_3 \in X_3$, $\exists x_2 \in X_2$ *s.t.* $x_2 \xrightarrow{E_2} x_3$.

We call X_1 a *representative set* of X_0 over E_0 while X_2 a representative set of X_3 over E_2^{-1}. Suppose we have obtained one such representative set X_1, we know the following

$$\bigcup_{x_1 \in X_1} \left\{ x_0 \in X_0 : x_0 \xrightarrow{E_0} x_1 \right\} = X_0.$$

By removing the overlapping elements among $\left\{ x_0 \in X_0 : x_0 \xrightarrow{E_0} x_1 \right\}$ for all $x_1 \in X_1$, we get a partition of X_0 which can be stored in a hash table with the elements in X_1 as keys and sets after partitioning as values (similar to X_3 and X_2). We call such a hash table a *partition (hash) table* of X_0 over E_0. An intuitive algorithm for determining one representative set as well as the corresponding partition table for a non-linear function and its input difference set is given in Algorithm 1.

The basic idea of Algorithm 1 is to *select* representative for X one by one and exclude corresponding elements from X until X is reduced to an empty set. The complexity of Algorithm 1 is roughly limited by $\mathcal{O}(|X|)$ times of loops (line 3-10). The operations in line 5 and 8 determine the real time of this algorithm, whose complexity is at most $2^{2\log|X|}$ (the complexity of computing the whole DDT of f). Thus the overall complexity of Algorithm 1 is bounded by $\mathcal{O}(2^{3\log|X|})$. Note

Algorithm 1: Determine a representative set and partition table of X over f

Data: $X \subseteq \mathbb{F}_2^n$ and $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$
Result: A representative set S, a partition table H

Allocate $S \leftarrow \varnothing$
Allocate a hash table $H' \leftarrow \varnothing$
while X *is not empty* **do**
  $x \xleftarrow{\mathcal{R}} X$ `/* randomly select` x `from` X `*/`
  compute Y *s.t.* $x \xrightarrow{f} Y$
  $y \xleftarrow{\mathcal{R}} Y$ `/* randomly select` y `from` Y `*/`
  $S \leftarrow S \cup \{y\}$ `/*` y `is chosen as a representative */`
  compute T *s.t.* $T \xrightarrow{f} y$ `/*` T `has been represented by` y `*/`
  $H'[y] \leftarrow T$
  $X \leftarrow X - T$ `/* Proceed with the remaining elements */`
`/* remove overlapping elements */`
Allocate a hash table H
for $s \in S$ **do**
  for $h \in H.keys$ **do**
    $H[s] \leftarrow H'[s] - H[h]$ `/* elements in` $H[h]$ `are recorded already */`
return S, H

that the actual complexity should be much less than this upper bound for the number of loops usually small. In applications of this paper, f will be at most a 16-bit-input function, so this algorithm is practical.

We first apply Algorithm 1 to X_0 to obtain its representative set X_1 and a partition table H_1 over E_0, *i.e.*,

$$X_0 = \sum_{x_1 \in X_1} H_1[x_1] \tag{1}$$

Similarly we get the representative set X_2 and partition table H_2 of X_3 over E_2^{-1}, *i.e.*,

$$X_3 = \sum_{x_2 \in X_2} H_1[x_2] \tag{2}$$

Then the whole search space $X_0 \otimes X_3$ has been partitioned into $|X_1| \times |X_2|$ smaller sets through combining Equation (1) and (2), *i.e.*,

$$X_0 \otimes X_3 = \sum_{x_1 \in X_1} H_1[x_1] \bigotimes \sum_{x_2 \in X_2} H_2[x_2] = \sum_{x_1 \in X_1} \sum_{x_2 \in X_2} H_1[x_1] \otimes H_2[x_2] \tag{3}$$

Figure 3 demonstrates the partition of $X_0 \otimes X_3$. A partition of $X_0 \otimes X_3$ can be uniquely determined by a quartet (X_1, H_1, X_2, H_2). For simplicity of description, we define the partition of a (round-reduced) cipher.

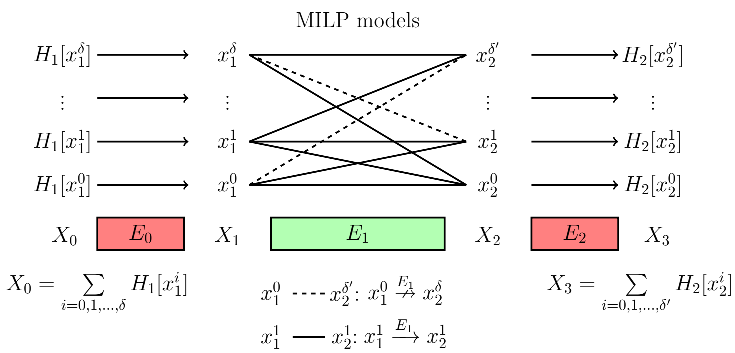


Fig. 3: The partitions of the input difference set X_0 and X_1. Since $x_1^0 \overset{E_1}{\nrightarrow} x_2^{\delta'}$, $H_1[x_1^0] \otimes H_2[x_2^\delta]$ has potential to contain IDs. Since $x_1^1 \xrightarrow{E_1} x_2^1$, $H_1[x_1^1] \otimes H_2[x_2^1]$ contains no IDs.

Definition 1 (Partition). *For a (round-reduced) cipher $E = E_2 \circ E_1 \circ E_0$, a partition of its whole input and output difference spaces is a set of smaller sets as follows,*

$$\mathbb{P}(X_1, H_1, X_2, H_2) = \{H[x_1] \otimes H[x_2] : x_1 \in X_1, x_2 \in X_2\}$$

When there is no ambiguity, we just say $\mathbb{P}$ is a partition of E.

3.2 Partition: A Practical Viewpoint

If we apply directly Algorithm 1 to $\mathbb{F}_2^n$, the complexity is not affordable even for a 64-bit block cipher. However, an important observation is that SPN ciphers usually comprise several smaller parallel parts. The well-known examples include the superboxes used in AES-like ciphers such as SKINNY [3] and CRAFT [4]. Two continuous rounds can be represented by 4 parallel superboxes. Another example is GIFT [1] which follows a so-called Substitution bit-Permutation Network (SbPN) paradigm. All Sboxes of the i-th round of GIFT, denoted by $Sb_0^i, Sb_1^i, \ldots, Sb_s^i$ where $s = n/4$ and n is the block size, can be grouped in two different ways – the Quotient and Remainder groups, Qx and Rx, defined as

– $Qx = \{Sb_{4x}, Sb_{4x+1}, Sb_{4x+2}, Sb_{4x+3}\}$,
– $Rx = \{Sb_x, Sb_{x+q}, Sb_{x+2q}, Sb_{x+3q}\}$, where $q = \frac{s}{4}, 0 \leq x \leq q-1$.

Taking GIFT-64 as an instance, the 16-bit output of $Qx^i = \{Sb_{4x}^i, Sb_{4x+1}^i, Sb_{4x+2}^i, Sb_{4x+3}^i\}$ map to input bits of $Rx^{i+1} = \{Sb_x^{i+1}, Sb_{x+4}^{i+1}, Sb_{x+8}^{i+1}, Sb_{x+12}^{i+1}\}$. Then the interfacing two rounds of GIFT-64 can be also represented by 4 parallel superboxes. An illustration for GIFT-64 is shown in Figure 4.

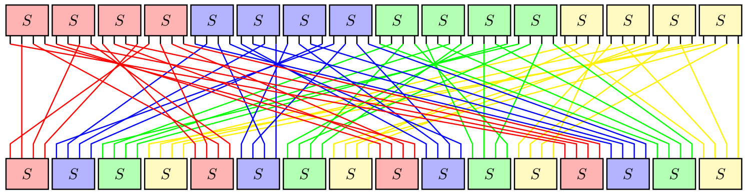


Fig. 4: The superbox representation of GIFT-64 based on the two groups (Quotient and Remainder) of Sboxes.

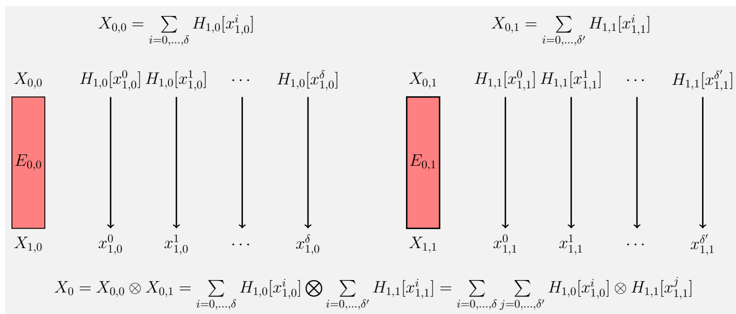


Fig. 5: The partition of the input difference set X_0 (the case $i = 0$ in Equation (4)) based on 2 superboxes ($m = 2$).

In the remaining part of this paper, we focus on these SPN or SbPN ciphers with superboxes. Suppose E_0 and E_2 comprise respectively of m parallel superboxes, denoted by $E_0 = E_{0,0}||E_{0,1}||\cdots||E_{0,m-1}$ and $E_2 = E_{2,0}||E_{2,1}||\cdots||E_{2,m-1}$, where the size of input and output of $E_{i,j}, i \in \{0,2\}, j \in \{0,1,\ldots,m-1\}$ is $n/4$ bits. Then we apply Algorithm 1 to each $E_{i,j}$, which is a function with 16-bit block size.

For $i \in \{0,1,\ldots,m-1\}$, let $X_{1,i}$ and $H_{1,i}$ be representative sets and partition tables for $E_{0,i}$ of its input difference set $X_{0,i}$ while $X_{2,i}$ and $H_{2,i}$ the representative sets and partition tables for $E^{-1}_{2,i}$ of $X_{3,i}$. The Equation (1) and (2) can be re-written as

$$\begin{aligned} X_i = \bigotimes_{0\le j<m} X_{i,j} &= \bigotimes_{0\le j<m} \left(\sum_{x_{i,j}\in X_{i,j}} H_{i,j}[x_{i,j}] \right) \\ &= \sum_{x_{i,0}\in X_{i,0}} \cdots \sum_{x_{i,m-1}\in X_{i,m-1}} H_{i,0}[x_{i,0}] \otimes \cdots \otimes H_{i,m-1}[x_{i,m-1}] \end{aligned} \tag{4}$$

where $i \in \{1,2\}$. See Figure 5 for a better understanding to Equation (4).

Then we can accordingly rewrite Equation (3) as

$$\begin{aligned} X_0 \otimes X_3 &= \bigotimes_{i=1,2} \left(\sum_{x_{i,0} \in X_{i,0}} \cdots \sum_{x_{i,m-1} \in X_{i,m-1}} H_{i,0}[x_{i,0}] \otimes \cdots \otimes H_{i,m-1}[x_{i,m-1}] \right) \\ &= \sum_{x_{1,0} \in X_{1,0}} \cdots \sum_{x_{2,m-1} \in X_{2,m-1}} H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}] \end{aligned} \tag{5}$$

That is to say, considering the superbox effects, we can partition the whole difference space into $\prod_{i=1,2;j=0,1,\ldots,m-1} |X_{i,j}|$ smaller sets. A partition of E is consequently updated to

$$\begin{aligned} &\mathbb{P}(X_{1,0}, H_{1,0}, \ldots, X_{2,m-1}, H_{2,m-1}) \\ &\quad = \{H[x_{1,0}] \otimes \cdots \otimes H[x_{2,m-1}] : x_{i,j} \in X_{i,j} \text{ for } i = 1,2; 0 \leq j < m\} \end{aligned}$$

which is related to a tuple with $4 \cdot m$ elements.

3.3 Solving MILP Models for E_1

Once we get a partition $\mathbb{P}(X_{1,0}, H_{1,0}, \ldots, X_{2,m-1}, H_{2,m-1})$ of E, we construct $\prod_{i=0,1;j=0,\ldots,m-1} |X_{i,j}|$ MILPs for each elements $(x_{1,0}, \ldots, x_{2,m-1}) \in X_{1,0} \otimes \cdots \otimes X_{2,m-1}$ to see whether $(x_{1,0}, \ldots, x_{1,m-1}) \xrightarrow{E_1} (x_{2,0}, \ldots, x_{2,m-1})$. If the MILP model for $(x_{1,0}, \ldots, x_{2,m-1})$ is feasible, we do not need to consider $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,0}[x_{2,0}]$ any more. Otherwise, $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,0}[x_{2,0}]$ has the potential to contain some IDs, we have to proceed with it in the next step.

3.4 Identify All IDs in Remaining $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}]$

The final step is to handle the remaining $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}] \in \mathbb{P}$ that survive the second step one by one. We mainly introduce two methods to find all IDs in each $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}]$ in this subsection.

Direct Search. Considering the case when the size of $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}]$ is small. Let $\sigma = \prod_{i=1,2;0 \leq j < m} |H_{i,j}|$, for example, we say the size is small when $\sigma \leq 2^{28}$, we can just directly test every pattern with a MILP model as [20,10]. All IDs contained in $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}]$ can be found naturally.

To enhance the efficiency of this step, we introduce the *fast reduce* technique. For a randomly chosen plaintext-ciphertext difference pair from $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}]$, we construct the MILP model to see if this pair is an ID. If this pair is truly an ID, we continue checking whether it belongs to a truncated ID. If so, we will record this truncated ID and remove all related IDs that belong to this truncated ID from the search pool. If this ID doesn't belong to any truncated ID, we record it and proceed with another pair. If it is not an ID, then a difference characteristic will be returned by the MILP solver. We extract the values of $x_1^\star$ and $x_2^\star$ of this characteristic (this is easy with the interface of

MILP solvers), and calculate two sets $X_0^\star$ and $X_3^\star$ satisfying $X_0^\star \xrightarrow{E_0} x_1^\star$ and $x_2^\star \xrightarrow{E_1} X_3^\star$. Thus all patterns in $X_0^\star \otimes X_3^\star$ are all possible, we only need to proceed with $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}] \backslash X_0^\star \otimes X_3^\star$ until all patterns are determined as possible or impossible.

Partition Further. If σ is larger (e.g., $\sigma > 2^{28}$), exhausting all patterns is not a good idea. We can apply Algorithm 1 to sets in every $H_{i,j}, i \in \{1,2\}, j \in \{0,1,\ldots,m\}$ and repeatedly partition $H_{1,0}[x_{1,0}] \otimes \cdots \otimes H_{2,m-1}[x_{2,m-1}]$ into several smaller sets. We handle each smaller set according to its size recursively until the size is below the threshold and can be handled by a direct search.

The whole procedure for identifying all IDs among $X_0 \otimes X_3$ over E is demonstrated in Algorithm 2.

Algorithm 2: Find all IDs over a cipher E

Data: $E_0 = E_{0,1}|| \cdots ||E_{0,m-1}$, $E_2 = E_{2,1}|| \cdots ||E_{2,m-1}$, E_1

Result: a set I containing all IDs

`/* step 1: partition the whole set */`

for *each* $i \in \{1,2\}$ **do**
- **for** *each* $j \in \{0,1,\ldots,m-1\}$ **do**
  - apply Algorithm 1 to $H_{i,j}$ getting its representative set $X_{i,j}$ and partition table $H_{i,j}$

`/* step 2: solve MILP Models for` E_1 `*/`

Allocate $J \leftarrow \varnothing$

for *each* $(x_{1,0},\ldots,x_{2,m-1}) \in X_{1,0} \otimes \cdots \otimes X_{2,m-1}$ **do**
- construct a MILP model for E_1 with the input/output difference with $(x_{1,0},\ldots,x_{1,m-1})$ and $(x_{2,0},\ldots,x_{2,m-1})$, respectively
- **if** *model is infeasible* **then**
  - $J \leftarrow J \cup \{(x_{1,0},\ldots,x_{2,m-1})\}$

`/* step 3: identify all IDs */`

Allocate $I \leftarrow \varnothing$

for *each* $(x_{1,0},\ldots,x_{2,m-1}) \in J$ **do**
- **if** $\prod_{i=1,2;j=0,1,\ldots,m-1} |H_{i,j}| > 2^{28}$ **then**
  - recursively recall Alg. 2 to $H_{1,0} \otimes \cdots \otimes H_{2,m-1}$ and push all the IDs into I
- **else**
  - $\prod_{i=1,2;j=0,1,\ldots,m-1} |H_{i,j}| \leq 2^{28}$
- construct MILP models to test every patterns in $H_{1,0} \otimes \cdots \otimes H_{2,m-1}$, and push those impossible ones into I

return I

---

4 Applications to AES-Like SPN Ciphers

One of the standard ways for designing a good round function from an Sbox and an MDS mapping is the one followed by the AES [12] and is known as the wide trail strategy [11]. Some newly proposed lightweight ciphers also follow the AES structure by replacing the MDS matrix with simple ones such as SKINNY [3] and CRAFT [4]. We say these ciphers are AES-like. In this section, we show how to apply our methods to SKINNY-64, the application to CRAFT is provided in Appendix A.

The experiments are conducted by Gurobi Solver (version 9.1.1) on a workstation with 2×AMD EPYC 7302 16-core (32 siblings) Processor 3.3 GHz, (a total 64 threads), 256G RAM, and Ubuntu 20.10.

Application to SKINNY-64. Our first application is to SKINNY-64. The block cipher family SKINNY was presented at CRYPTO 2016 [3] designed under the TWEAKEY framework [14], whose goal is to compete with the NSA design SIMON [2] in terms of hardware/software performance. According to the length of block and tweakey, the SKINNY family consists of 6 different members represented as SKINNY-n-t, where $n = 64, 128$ and $t = n, 2n, 3n$, which respectively represent the sizes of the block and tweakey. We are only interested in the security of the 64-bit version of SKINNY in this paper, *i.e.*, SKINNY-64, under the single tweakey model. The round function of SKINNY-64 comprises five operations as $\mathsf{SubCells}$ (SC), $\mathsf{AddConstants}$ (AC), $\mathsf{AddRoundTweakey}$ (ART), $\mathsf{ShiftRows}$ (SR) and $\mathsf{MixColumns}$ (MC). Since we only consider the single-tweakey scenario, we can ignore the ART and AC operations and pay attention to the remaining three ones. Therefore, a round of SKINNY can be written as

$$R = \mathsf{MC} \circ \mathsf{SR} \circ \mathsf{SC}$$

When applying Algorithm 2 to r-round SKINNY-64, we rearrange the functions in the r rounds as

$$R^r = \underbrace{\mathsf{SC} \circ \mathsf{MC} \circ \mathsf{SC}}_{E_2} \circ \underbrace{\mathsf{SR} \circ R^{r-4} \circ \mathsf{MC} \circ \mathsf{SR}}_{E_1} \circ \underbrace{\mathsf{SC} \circ \mathsf{MC} \circ \mathsf{SC}}_{E_0}$$

As can be seen, the SR in the first round and $\mathsf{MC} \circ \mathsf{SR}$ in the last round are omitted for they do not affect our analysis. E_0 and E_2 consist of four parallel superboxes $E_{0,i}$ and $E_{2,i}$ for $i = 0, 1, 2, 3$, respectively.

We apply Algorithm 1 to the four superboxes of E_0 and the four inverse superboxes of E_2^{-1}. The representative sets we calculated in the experiment are listed in Table 2. Note that the four superboxes are identical as well as their representative sets and partition tables.

As is seen, each representative set of the superbox of $E_{0,i}$ and $E_{2,i}^{-1}$ contains only 7 values, so the sizes of X_1 and X_2 are both $7^4 - 1 = 2,400$ non-zero values. Considering the rotational symmetry of SKINNY-64, we can remove the rotationally-symmetric elements in X_1. After this treatment, only 615 elements remain in X_1. Therefore, the total number of MILP models we need to solve

Table 2: Representative sets of the superboxes $E_{0,i}$ and $E_{2,i}^{-2}$ for $i = 0, 1, 2, 3$, of SKINNY-64

Representative set	Values (hexadecimal)
$X_{1,i}, i = 0, 1, 2, 3$	`0, b0, b000, b080, de9d, e0e4, ee0e`
$X_{2,i}, i = 0, 1, 2, 3$	`0, a, 606, eee, f00, 3330, eeef`

is $615 \times 2,400 = 147,600 \approx 2^{20.5}$. Since these MILP models are treated independently, we can solve them by a parallel strategy based on multi-threading programming.

11-Round. For 11-round SKINNY-64, We need to solve $2^{20.5}$ MILP models for $11-4 = 7$ rounds. These MILP models were solved in roughly 4 hours. There are 618 infeasible patterns $(x_{1,0}, \ldots, x_{1,3}, x_{2,0} \ldots, x_{2,3}) \in X_{1,0} \otimes X_{1,1} \otimes X_{2,0} \cdots \otimes X_{2,3}$. We then try to identify all the IDs from the corresponding sets related to these 618 patterns. Finally, since every ID belongs to one certain truncated ID, we identified all 432 truncated IDs. All these IDs are provided in our anonymous git repository. For a better impression to them, we visualize one ID in Figure 8.

12-Round. For 12-round SKINNY-64, we need to solve $2^{20.5}$ MILP models for 8 rounds. Solving these MILP models cost about 1.5 hours in our cluster. Only 15 patterns out of them $(x_{1,0}, x_{1,1}, x_{2,0} \ldots, x_{2,3}) \in X_{1,0} \otimes X_{1,1} \otimes X_{2,0} \cdots \otimes X_{2,3}$ are infeasible. Among them, we extracted 2,700 IDs, which are assembled into 12 truncated IDs. These IDs are identical to those reported in previous works [3,22].

13-Round. We prove that there does not exist any ID for 13-round SKINNY-64 even with consideration of the details of Sboxes and linear layers.

5 Applications to SbPN Cipher GIFT-64

Substitution and bit-Permutation Network (SbPN) is a special SP network where the permutation layer takes a bit shuffle rather than a word-oriented diffusion. It was introduced by the first ultra-lightweight block cipher PRESENT at CHES 2007 [7], and recently refined by GIFT at CHES 2017 [1]. SbPN consists of a layer of Sboxes (denoted by S), a bit shuffle (denoted by P), and a layer of key/constant addition only, which can be very efficient, especially in hardware implementation. In this section, we show how to apply our methods to GIFT-64. This paper is only interested in the single-key scenario, we do not need to consider the key/constant addition. So a round function of GIFT-64 is viewed as $R = \mathsf{P} \circ \mathsf{S}$. As is mentioned in Section 3.2 and also shown in Figure 6, two interfacing rounds of GIFT-64 can be viewed as four parallel superboxes. Then we have

$$\mathsf{S} \circ \mathsf{P} \circ \mathsf{S} = \mathsf{P}_2 \circ \mathsf{S} \circ \mathsf{P}_1 \circ \mathsf{S},$$

where P_1 consists of four identical small bit shuffles and P_2 is a word-oriented permutation. P_1 and P_2 in GIFT-64 are illustrated in Figure 6. Let $\mathsf{P}_1 =$

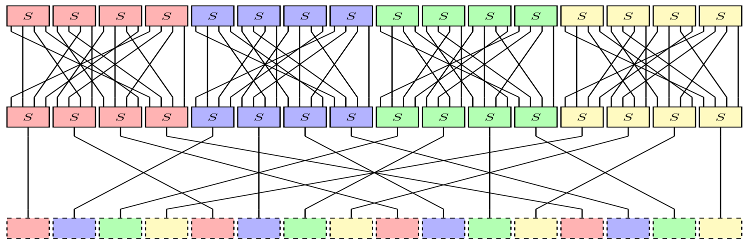


Fig. 6: An equivalent representation of the GIFT-64 round functions.

$\mathsf{P}'_1||\mathsf{P}'_1||\mathsf{P}'_1||\mathsf{P}'_1$, then P'_1 and P_2 are as follows (inside each permutation, the 0-th is the leftmost unit),

$$\mathsf{P}'_1 = [12, 1, 6, 11, 8, 13, 2, 7, 4, 9, 14, 3, 0, 5, 10, 15],$$
$$\mathsf{P}_2 = [0, 4, 8, 12, 1, 5, 9, 13, 2, 5, 10, 14, 3, 6, 11, 15].$$

We thus rearrange the round functions of an r-round GIFT-64 cipher as

$$R^r = \underbrace{\mathsf{S} \circ P_1 \circ \mathsf{S}}_{E_2} \circ \underbrace{R^{r-4} \circ \mathsf{P} \circ \mathsf{P}_2}_{E_1} \circ \underbrace{\mathsf{S} \circ \mathsf{P}_1 \circ \mathsf{S}}_{E_0}$$

Again, we apply Algorithm 1 to superboxes of $E_{0,i}$ and $E_{2,i}^{-2}$ and get the representative sets as shown in Table 3. The number of elements in the representative sets of superboxes for $E_{0,i}$ are 10, and for $E_{2,i}^{-1}$ is 9, so the number of non-zero elements in $X_1 \otimes X_2$ is $(10^4 - 1) \times (9^4 - 1) = 65,593,440 \approx 2^{26}$. We then need to construct 2^{26} MILP models with these patterns for E_1.

In the specification, the designers showed that there do not exist any impossible differentials with 1-active nibble against 7 rounds of GIFT-64. We then check the 7-round GIFT-64 first, unfortunately, after the first step of Algorithm 2 costing about 12 hours, there are too many (363,510) impossible patterns out of the 2^{26} patterns. We find that processing these bad inner patterns costs a significant amount of time, this may imply that 7 rounds is the borderline of whether IDs exist or not. Considering that GIFT-64 is a 28-round cipher, 7 round IDs (even if an ID exists) should not threaten its security, we do not pursue the exact security proof for 7 rounds.

For 8-round GIFT-64, these 2^{26} MILP models can be processed within 17 hours, where only 236 are impossible. With another 4 minutes, all the 236 patterns can be processed and none of them imply IDs. In other words, 8-round GIFT-64 is free of any IDs.

6 Towards Large-Size Ciphers

Our tool works very well on 64-bit SPN and SbPN ciphers. However, its efficiency for ciphers with larger blocks is not high. Note that in the second step

Table 3: The representative sets of superboxes $E_{0,i}$ and $E_{2,i}^{-1}$ for $i = 0, 1, 2, 3$, of GIFT-64

Representative set	Values (hexadecimal)
$X_{1,i}, i = 0, 1, 2, 3$	`0, 50b, f39, 5a97, a9d9, b35f, b3d0, b706, d0b3, d5f0`
$X_{2,i}, i = 0, 1, 2, 3$	`0, ec, d90, e0f, 9b7b, cd7e, e00f, e7cf, fdd7`

of Algorithm 2 (also introduced in Section 3.3), we still need to solve a set of MILP models whose size can be larger than 2^{20}. For large-size ciphers, this step would take a lot of time. However, with some compromises between accuracy and efficiency, our method can be very useful to strengthen some existing search strategies for large-size ciphers. In this section, we show how to use our idea to enhance the search for the large-size cipher Rijndael-192. The application to GIFT-128 is provided in Appendix B.

Application to Rijndael-192. In [20], Sasaki and Todo also encountered similar problems with the efficiency when processing 8-bit Sbox ciphers, so they took a degenerated version of the MILP model called the *arbitrary Sbox mode* to boost the search efficiency. The arbitrary Sbox mode allows the model to ignore the details of the Sbox while mainly reflecting the property of the linear layers, thus the search time can be saved a lot. Despite this compromise, their MILP model still cannot work to search for all plaintext and ciphertext difference pairs. Let s be the number of Sboxes of a cipher, then the whole search space is 2^{2s}, *e.g.*, for Rijndael-192 that has 24 Sboxes, the search space is approximately 2^{48} which is also very costly.

Inspired by their work, we can also boost our search efficiency by ignoring the details of Sboxes and even the linear layers. Different from Sasaki and Todo's tool, ours can exhaust all truncated difference pairs very fast. As is well known, Rijndael was designed by Daemen and Rijmen in 1998 and the 128 block size version was selected as the AES [12]. In this section, we take Rijndael-192 as an example and show how to identify all truncated IDs of 6-round Rijndael-192 within seconds. The state of Rijndael-192 is arranged as 4×6 matrix of bytes. Its round function comprises four operations, $\mathsf{AddRoundKey}$ (AK), $\mathsf{SubBytes}$ (SB), $\mathsf{ShiftRows}$(SR) and $\mathsf{MixColumns}$(MC). Without considering AK, r-round Rijndael-192 can be written similar to SKINNY-64,

$$R^r = \underbrace{\mathsf{SB} \circ \mathsf{MC} \circ \mathsf{SB}}_{E_2} \circ \underbrace{\mathsf{SR} \circ R^{r-4} \circ \mathsf{MC} \circ \mathsf{SR}}_{E_1} \circ \underbrace{\mathsf{SB} \circ \mathsf{MC} \circ \mathsf{SB}}_{E_0},$$

Note that we also omit the SR of the first round and $\mathsf{MC} \circ \mathsf{SR}$ of the last round. Thus E_0 and E_2 of Rijndael-192 can be seen as 6 parallel superboxes, respectively.

In the arbitrary Sbox/MC mode, only being active or inactive for an Sbox would be considered instead of its detailed input and output differences. Thus, the difference of an Sbox can be labeled by one bit 0 or 1. Consequently, the differences of plaintexts, ciphertexts, and intermediate states are labeled by a

Table 4: The representative sets of superboxes $E_{0,i}$ and $E_{2,i}^{-1}$ for $i = 0, 1, 2, 3, 4, 5$, of Rijndael-192

Representative set	Values (hexadecimal)
$X_{1,i}, i = 0, 1, 2, 3, 4, 5$	`0, f`
$X_{2,i}, i = 0, 1, 2, 3, 4, 5$	`0, f`

binary vector $x \in \mathbb{F}_2^{24}$. Consequently, to apply our new tool with the arbitrary Sbox/MC mode, we adapt Algorithm 1 to compute the representative sets and partition tables for all truncated differences into a superbox. In terms of Rijndael-192, there are only 16 kinds of truncated IDs as input of a superbox (from `0b0000` to `0b1111`). Since we only consider the branch number of MC, its representative set contains only 2 elements as shown in Table 4.

We first call Algorithm 2 to test 6-round Rijndael-192 (note that the MILP models here are in the arbitrary Sbox mode and only the branch number of MC are considered.). Within 1 second, we can find out all 6,750 IDs, which are provided with our codes in our git repository. An example of these IDs is shown in Figure 9. Next, we test 7 rounds, no ID can be found. Therefore, we prove that there is no truncated ID for 7-round Rijndael-192.

7 Conclusion and Future Work

In this paper, we proposed a new method to detect all IDs based on MILP models with the DDT considered. The whole search space is partitioned into smaller ones and some of them can be quickly determined to contain no IDs. Thus the search space is significantly reduced, sometimes to a practical size. Then we could handle the remaining candidates to check if there are any IDs. With this novel strategy, we identified all IDs for 11-, and 12-round SKINNY-64 and prove there exists no ID for 13-round SKINNY-64. Similarly, we identified all IDs for 13-round CRAFT and prove there is no ID for 14 rounds. We also proved there is no ID for 8-round GIFT-64. The idea of our new method is also very useful to enhance the current MILP models for ciphers with large blocks. For example, we can partition the whole space of truncated differences for Rijndael-192 into smaller ones under the arbitrary Sbox mode. We quickly identified all truncated IDs for 6-round Rijndael-192 in 1 second and proved there is no truncated ID for 7 rounds. For GIFT-128, we searched in a smaller space where all differences have an active superbox in plaintext and ciphertext differences.

It is interesting to study if our method is also applicable to searching for all zero-correlation linear hull distinguishers [8] due to the dual property of the ID and zero-correlation linear hull. However, since the correlation of a linear hull is equivalent to the summation of the correlation of all its trails, in theory there might be a linear hull with the zero-correlation consisting some non-zero-correlation linear trails. Thus, our method cannot be directly used for zero-correlation linear hulls, or more assumptions might be necessary. This would be one of our future works. The representative sets and partition tables in our work are generated based on an intuitive algorithm (Algorithm 1) that is not very efficient. We guess there may be other methods to choose the representative sets and partition tables that could consider the property of E_1 simultaneously, such that we could reduce the number of MILP models we need to solve in the second step of Algorithm 2 and fewer impossible patterns for E_1 remains after it. This would be another interesting future work. Finally, our method currently works only for ciphers whose round functions are based on superboxes, it is also interesting to see how to generalize it to more types of ciphers.

Acknowledgment. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. Meiqin Wang is supported by the National Natural Science Foundation of China (Grant No. 62002201, Grant No. 62032014), the National Key Research and Development Program of China (Grant No. 2018YFA0704702, 2018YFA0704704), the Major Scientific and Technological Innovation Project of Shandong Province, China (Grant No. 2019JZZY010133), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025).

References

1. S. Banik, S. Pandey, T. Peyrin, Y. Sasaki, S. Sim, and Y. Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In *CHES 2017*, volume 10529 of *LNCS*, pages 321–345. Springer, 2017.
2. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK lightweight block ciphers. In *DAC 2015*, pages 175:1–175:6. ACM, 2015.
3. C. Beierle, J. Jean, S. Kölbl, G. Leander, A. Moradi, T. Peyrin, Y. Sasaki, P. Sasdrich, and S. Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO 2016, Part II*, pages 123–153, 2016.
4. C. Beierle, G. Leander, A. Moradi, and S. Rasoolzadeh. CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. *IACR Trans. Symmetric Cryptol.*, 2019(1):5–45, 2019.
5. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In *EUROCRYPT 1999*, pages 12–23. Springer, 1999.
6. A. Biryukov. Miss-in-the-middle attack. In *Encyclopedia of Cryptography and Security, 2nd Ed*, page 786. Springer, 2011.
7. A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *CHES 2007*, pages 450–466, 2007.
8. A. Bogdanov and V. Rijmen. Linear hulls with correlation zero and linear cryptanalysis of block ciphers. *Designs Codes and Cryptography*, 70(3):369–383, 2014.
9. T. Cui, S. Chen, K. Fu, M. Wang, and K. Jia. New automatic tool for finding impossible differentials and zero-correlation linear approximations. *Sci. China Inf. Sci.*, 64(2), 2021.

10. T. Cui, K. Jia, K. Fu, S. Chen, and M. Wang. New automatic search tool for impossible differentials and zero-correlation linear approximations. *IACR Cryptol. ePrint Arch.*, page 689, 2016.
11. J. Daemen and V. Rijmen. AES and the wide trail design strategy. In *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 108–109. Springer, 2002.
12. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. ISC. Springer, 2002.
13. O. Dunkelman, S. Huang, E. Lambooij, and S. Perle. Single tweakey cryptanalysis of reduced-round SKINNY-64. In *CSCML 2020*, volume 12161 of *LNCS*, pages 1–17. Springer, 2020.
14. J. Jean, I. Nikolic, and T. Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In *ASIACRYPT 2014 Part II*, volume 8874 of *LNCS*, pages 274–288. Springer, 2014.
15. J. Kim, S. Hong, J. Sung, C. Lee, and S. Lee. Impossible differential cryptanalysis for block cipher structures. In *INDOCRYPT 2003*, volume 2904 of *LNCS*, pages 82–96. Springer, 2003.
16. L. Knudsen. Deal-a 128-bit block cipher. *complexity*, 258(2):216, 1998.
17. J. Lu, O. Dunkelman, N. Keller, and J. Kim. New impossible differential attacks on AES. In *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 279–293. Springer, 2008.
18. Y. Luo, X. Lai, Z. Wu, and G. Gong. A unified method for finding impossible differentials of block cipher structures. *Inf. Sci.*, 263:211–220, 2014.
19. N. Mouha, Q. Wang, D. Gu, and B. Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Inscrypt 2011*, volume 7537 of *LNCS*, pages 57–76. Springer, 2011.
20. Y. Sasaki and Y. Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In *EUROCRYPT 2017 Part III*, volume 10212 of *LNCS*, pages 185–215, 2017.
21. B. Sun, M. Liu, J. Guo, V. Rijmen, and R. Li. Provable security evaluation of structures against impossible differential and zero correlation linear cryptanalysis. In *EUROCRYPT 2016 Part I*, volume 9665 of *LNCS*, pages 196–213. Springer, 2016.
22. L. Sun, D. Gérault, W. Wang, and M. Wang. On the usage of deterministic (related-key) truncated differentials and multidimensional linear approximations for SPN ciphers. *IACR Trans. Symmetric Cryptol.*, 2020(3):262–287, 2020.
23. S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 158–178. Springer, 2014.
24. Q. Wang and C. Jin. More accurate results on the provable security of AES against impossible differential cryptanalysis. *Design, Codes and Cryptography*, 87(12):3001–3018, 2019.
25. Q. Wang and C. Jin. Bounding the length of impossible differentials for SPN block ciphers. *Design, Codes and Cryptography*, 89(11):2477–2493, 2021.
26. S. Wu and M. Wang. Automatic search of truncated impossible differentials for word-oriented block ciphers. In *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 283–302. Springer, 2012.

A Applications to CRAFT

CRAFT is a lightweight tweakable block cipher proposed at FSE 2019 [4], which allows countermeasures against *differential fault attacks* to be integrated into the cipher at the algorithmic level with ease. CRAFT employs a lightweight and involutory Sbox and linear layer, such that the encryption function can be turned into decryption at a low cost. CRAFT is a 32-round iterative tweakable block cipher operating on 64-bit blocks of data with a 128-bit key, and 64-bit tweak, whose round function consists of five operations including

1. MixColumn (MC): MC multiplies four nibbles of each state column with the involutory binary matrix M. The details of M is listed below,

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. AddConstants (AC) and AddTweakey (ATK): since our work focuses on the ID attacks under the single tweakey model, we can ignore the two operations.
3. PermuteNibbles (PN): PN permutes the cells of the state by an involutory permutation P such that the i-th cell of the new state is replaced by the $P[i]$-th cell of the original state, where

$$P = [15, 12, 13, 14, 10, 9, 8, 11, 6, 5, 4, 7, 1, 2, 3, 0].$$

4. SubBox (SB): SB is the only non-linear layer of CRAFT, using a 4-bit Sbox S as follows, note that this Sbox is involutory.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S[x]	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

One round function then comprises of (without AC and ATK)

$$R = \mathsf{SB} \circ \mathsf{PN} \circ \mathsf{MC}.$$

And functions of r-round CRAFT can be rearranged as

$$R^r = \underbrace{\mathsf{SB} \circ \mathsf{MC} \circ \mathsf{SB}}_{E_2} \circ \underbrace{\mathsf{PN} \circ \mathsf{MC} \circ R^{r-4} \circ \mathsf{PN}}_{E_1} \circ \underbrace{\mathsf{SB} \circ \mathsf{MC} \circ \mathsf{SB}}_{E_0}$$

Note that we omit the PN∘MC operation of the first round and the PN operation of the last round because they would not affect the impossible differential results. What's more, the order between SB and PN near the border of E_0 and E_1 are swapped.

13-Round. Similar to SKINNY-64, we apply Algorithm 1 to superboxes of $E_{0,i}$ and $E_{2,i}^{-1}$ and get their representative sets as shown in Table 5. As is seen, the number of elements in the representative sets are 10, then the sizes of non-zero

Table 5: The representative sets of superboxes $E_{0,i}$ and $E_{2,i}^{-2}$ for $i = 0, 1, 2, 3$, of CRAFT

Representative set	Values (hexadecimal)
$X_{1,i}, i = 0, 1, 2, 3$	`0, 600, 3303, 5fe3, 6000, 9c00, b0b0, b430, bd0c, ce3e`
$X_{2,i}, i = 0, 1, 2, 3$	`0, c00, 30b0, 3b0b, 594f, 6000, 9100, 9363, b404, b9b0`

elements in X_1 and X_2 are all $10^4 - 1 = 9,999$. Finally, we need to solve $9,999 \times 9,999 \approx 2^{24.6}$ MILP models. Such MILP models can be processed within 7 days, where only 48 is impossible out of the $2^{26.6}$. By checking the corresponding partition table entry related to the 48 impossible patterns, 12 IDs are identified with seconds. These IDs are identical to those found by designers in [4].
14-Round. We prove that there exists no ID for 14-round CRAFT with consideration of the details of Sboxes and linear layers.

B Application to GIFT-128

GIFT-128 is the 128-bit member in the GIFT family [1]. It utilizes the same Sbox with GIFT-64 but a large-size bit permutation. Similar to GIFT-64, the 32 Sboxes of GIFT-128 can also be arranged into the Quotient and Remainder groups. So the superbox representation is applicable to GIFT-128 too which is illustrated in Figure 7. The r-round GIFT-128 can be written as

$$R^r = \underbrace{\mathsf{S} \circ P_1 \circ \mathsf{S}}_{E_2} \circ \underbrace{R^{r-4} \circ \mathsf{P} \circ \mathsf{P}_2}_{E_1} \circ \underbrace{\mathsf{S} \circ \mathsf{P}_1 \circ \mathsf{S}}_{E_0}.$$

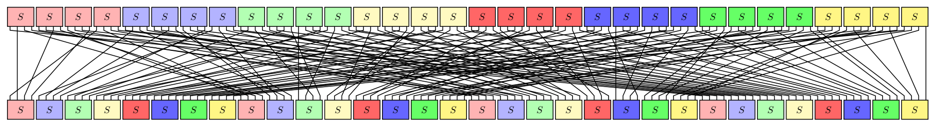
where each component of P_1 is as the same as that used by GIFT-64, P_2 is a word shuffle as

$$\mathsf{P}_2 = [0, 8, 16, 24, 1, 9, 17, 25, 2, 10, 18, 26, 3, 11, 19, 27, 4, \\ 12, 20, 28, 5, 13, 21, 29, 6, 14, 22, 30, 7, 15, 23, 31].$$

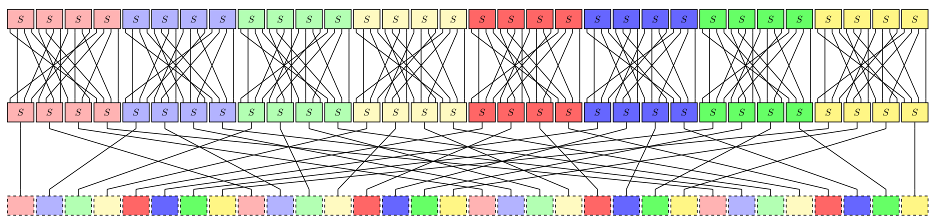
E_0 and E_2 consist of 8 parallel superboxes whose representative sets and partition tables are identical to those of GIFT-64. Thus if we apply Algorithm 2 to r-round GIFT-128, we need to process approximately $10^8 \times 9^8 \approx 2^{52}$ MILP models which are not affordable.

Our results for SKINNY-64, CRAFT, and GIFT justify the intuition used in previous works that the longest IDs for a cipher usually have few active bits or words at both ends. Thus it is meaningful if we can improve the efficiency of this search strategy. Leveraging our new method, instead of searching for IDs in the whole space, we can target a smaller space. For example, for r-round GIFT-128, we will only search the space that there is only one active superbox in both plaintext and ciphertext ends.

With the traditional strategy, in order to exhaust all difference pairs in this space, we need to construct approximately $2^{16} \times 2^{16} \times 8 \times 8 \approx 2^{38}$ MILP models



(a) The superbox representation of GIFT-128 based on the two groups of Sboxes.



(b) Rearrangement of the round function of GIFT-128 based on the superbox representation.

Fig. 7: The illustration of Proposition 1 and some implications.

for r-round GIFT-128. While taking our strategy, we only need to handle $9 \times 8 \times 8 \times 8 \approx 4,608$ MILP models for $r - 4$ rounds of GIFT-128 to reduce the search space, finally we use some additional time to process the remaining difference pairs. We apply this improved search strategy to 8-round GIFT-128, and prove there is no ID of the one-active-superbox pattern.

C ID Examples for 11-Round SKINNY and 6-Round Rijndael-192

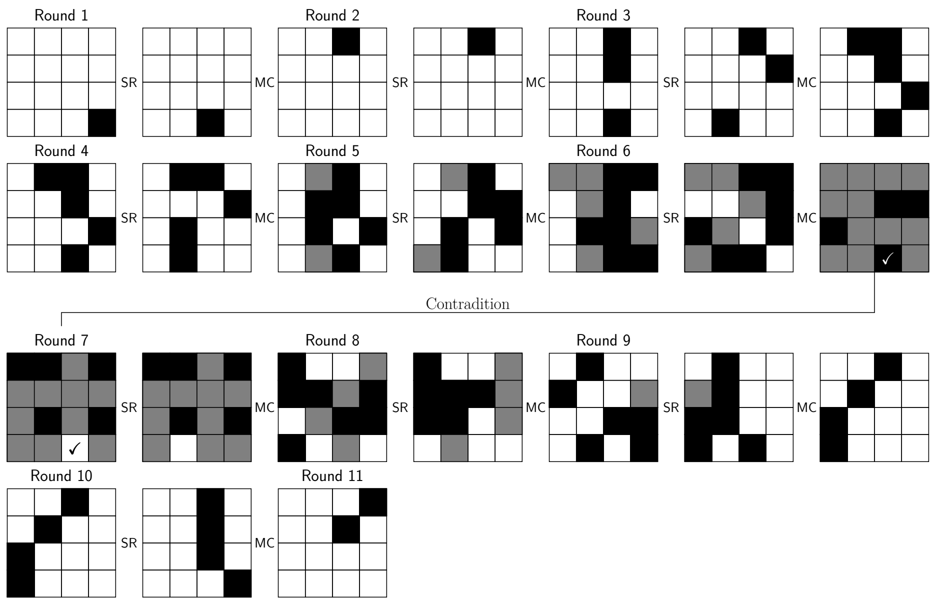


Fig. 8: An example ID visualization of 11-round SKINNY.

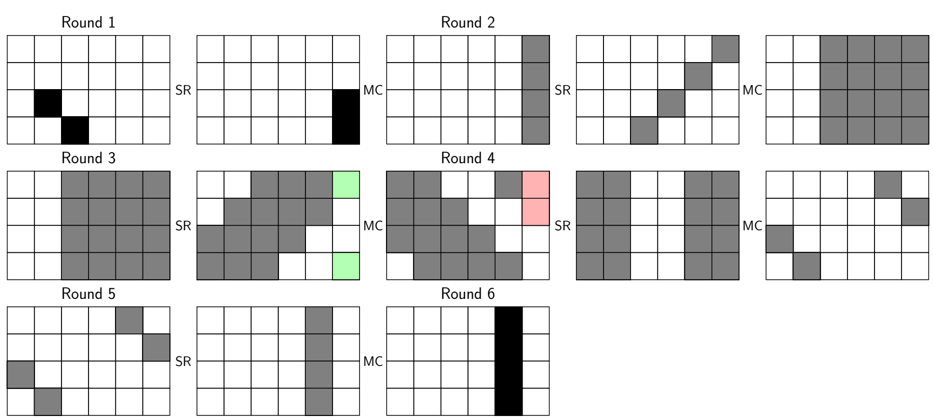


Fig. 9: An example ID visualization of 6-round Rijndael-192.