Throwing Boomerangs into Feistel Structures

Application to CLEFIA, WARP, LBlock, LBlock-s and TWINE

Hosein Hadipour, Marcel Nageler and Maria Eichlseder

Graz University of Technology, Graz, Austria
{hossein.hadipour,marcel.nageler,maria.eichlseder}@iaik.tugraz.at

Abstract. Automatic tools to search for boomerang distinguishers have seen significant advances over the past few years. However, most previous work has focused on ciphers based on a Substitution Permutation Network (SPN), while analyzing the Feistel structure is of great significance. Boukerrou et al. recently provided a theoretical framework to formulate the boomerang switch over multiple Feistel rounds, but they did not provide an automatic tool to find distinguishers. In this paper, by enhancing the recently proposed method by Hadipour et al., we provide an automatic tool to search for boomerang distinguishers and apply it to block ciphers following the Generalized Feistel Structure (GFS). Applying our tool to a wide range of GFS ciphers, we show that it significantly improves the best previous results on boomerang analysis. In particular, we improve the best previous boomerang distinguishers for 20 and 21 rounds of WARP by a factor of $2^{38.28}$ and $2^{36.56}$, respectively. Thanks to the effectiveness of our method, we can extend the boomerang distinguishers of WARP by two rounds and distinguish 23 rounds of this cipher from a random permutation. Applying our method to the internationally-standardized cipher CLEFIA, we achieve a 9-round boomerang distinguisher which improves the best previous boomerang distinguisher by one round. Based on this distinguisher, we build a key-recovery attack on 11 rounds of CLEFIA, which improves the best previous sandwich attack on this cipher by one round. We also apply our method to LBlock, LBlock-s, and TWINE and improve the best previous boomerang distinguisher of these ciphers.

Keywords: Lightweight cryptography · Boomerang cryptanalysis · MILP · Generalized Feistel structure · CLEFIA · WARP · TWINE · LBlock · LBlock-s

1 Introduction

Boomerang analysis, initially invented by Wagner [Wag99], has been under significant development over the last years. Recent progress includes a theoretical framework to evaluate boomerang switches in SPN ciphers as well as automatic tools to search for sandwich distinguishers. For instance, Hadipour et al. [HBS21] introduced a tool to search for sandwich distinguishers taking the switching effect into account for multiple rounds. They applied their tool to significantly improve the rectangle distinguishers for SKINNY and CRAFT. Almost at the same time, Delaune et al. [DDV20] introduced another tool to discover sandwich distinguishers that handles the probability computation of the middle part automatically and applied their tool to SKINNY. Other works [QDW+21, DQSW21] improved these methods further to identify sandwich distinguishers which take the key-recovery phase into account for linear key schedules. However, to the best of our knowledge, all previous works focus on SPN ciphers, particularly those with linear key schedules. In contrast, Feistel structures, an important category of block ciphers, have not been analyzed well by these new methods. Although Boukerrou et al. [BHL+20] proposed a theoretical framework to compute the probability of boomerang switches in Feistel structures very recently, they do not provide a tool to search for distinguishers.

Our Contributions. We improve the method by Hadipour et al. [HBS21] and provide an easy-to-use automatic tool to search for sandwich distinguishers while considering the switching effect over multiple rounds. With this tool, we significantly improve the sandwich distinguishers for several Feistel ciphers. Our tool is also applicable to SPN ciphers. The main improvement we propose is to differentiate between the encoding of truncated trails over the outer and the inner parts of sandwich distinguishers. More precisely, instead of using a standard truncated model for the entire upper and lower truncated trails, we encode transitions of probability one in the middle part of sandwich distinguishers. Therefore, we get an improved estimate for the number of common active S-boxes in the middle part.

Table 1: Summary of boomerang distinguishers for `WARP`, `CLEFIA`, `TWINE`, and `LBlock`.

Block cipher	#Rounds	Probability	Verified	Reference
`WARP`	9 / 40	$\mathbf{1}$	✓	Section 4
	14 / 40	$\mathbf{2^{-20.58}}$	✓	Section 4
	15 / 40	$\mathbf{2^{-28.58}}$	✓	Section 4
	16 / 40	$\mathbf{2^{-34.50}}$	✓	Section 4
	20 / 40	$2^{-114.24}$		[TB21]
	20 / 40	$\mathbf{2^{-75.96}}$		Section 4
	21 / 40	$2^{-121.11}$		[TB21]
	21 / 40	$\mathbf{2^{-84.55}}$		Section 4
	22 / 40	$\mathbf{2^{-96.55}}$		Section 4
	23 / 40	$\mathbf{2^{-115.59}}$		Section 4
`CLEFIA`	3 / 18	$\mathbf{1}$	✓	Section 5
	5 / 18	$\mathbf{2^{-12.26}}$	✓	Section 5
	6 / 18	$\mathbf{2^{-22.45}}$	✓	Section 5
	7 / 18	$\mathbf{2^{-32.67}}$	✓	Section 5
	8 / 18	2^{-92}		[MQ14]
	8 / 18	$\mathbf{2^{-76.03}}$		Section 5
	9 / 18	$\mathbf{2^{-99.12}}$		Section 5
`TWINE`	5 / 36	$\mathbf{1}$	✓	Section 6
	13 / 36	$\mathbf{2^{-34.32}}$	✓	Section 6
	14 / 36	$\mathbf{2^{-42.25}}$		Section 6
	15 / 36	$2^{-58.92}$		[TB21]
	15 / 36	$\mathbf{2^{-51.03}}$		Section 6
	16 / 36	$2^{-61.62}$		[TB21]
	16 / 36	$\mathbf{2^{-58.04}}$		Section 6
`LBlock`	5 / 36	$\mathbf{1}$	✓	Section 7
	13 / 36	$\mathbf{2^{-30.28}}$	✓	Section 7
	14 / 36	$\mathbf{2^{-38.86}}$		Section 7
	15 / 36	$\mathbf{2^{-46.90}}$		Section 7
	16 / 36	$2^{-60.53}$		[CM13]
	16 / 36	$\mathbf{2^{-57.16}}$		Section 7
`LBlock-s`	5 / 32	$\mathbf{1}$	✓	Section 7
	13 / 32	$\mathbf{2^{-30.23}}$	✓	Section 7
	14 / 32	$\mathbf{2^{-38.47}}$		Section 7
	15 / 32	$2^{-58.64}$		[TB21]
	15 / 32	$\mathbf{2^{-46.49}}$		Section 7
	16 / 32	$2^{-56.14}$		[BHL+20]
	16 / 32	$\mathbf{2^{-53.59}}$		Section 7

To demonstrate the practicality of our method, we apply it to several generalized Feistel structures, which yields a significant improvement compared to the best previous results. Our applications cover a wide range of generalized Feistel structures from the internationally-standardized block cipher `CLEFIA`, which has a complex round function but few rounds, to the recently proposed GFS block cipher `WARP` which has a light round function but much more rounds. For instance, by applying our search method to `WARP`, we dramatically improve the probability of the best previous sandwich distinguishers for 20 and 21 rounds of this cipher by a factor of $2^{38.28}$ and $2^{36.56}$, respectively. We improve the sandwich distinguishers for `WARP` by two rounds to distinguish 23 rounds of this cipher from a random permutation.

For `CLEFIA`, we not only improve the probability of the best previous sandwich distinguisher of this cipher remarkably but also extend it by one round by introducing a 9-round sandwich distinguisher with a probability much higher than 2^{-n}. Moreover, we provide the first practical distinguisher for 7 rounds of `CLEFIA` which can be experimentally verified. Due to the high importance of `CLEFIA`, building upon our 9-round sandwich distinguisher, we also provide a key-recovery attack on 11 rounds of `CLEFIA`, which improves the previous best sandwich attack by one round. We also apply our tool to `TWINE`, `LBlock`, and `LBlock-s`. In all cases, we improve the best previous sandwich distinguishers. Table 1 summarizes our results. For all applications, we have identified several practical reduced-round distinguishers that we have verified experimentally. The source code of our tool for finding distinguishers and the experimental verification are publicly available in the following Github repository: `https://github.com/hadipourh/comeback`

Outline We review the background on boomerang analysis as well as the previous works and recall the theoretical framework to formulate the probability of boomerang distinguishers in Section 2. Next, in Section 3, we introduce our search method for boomerang distinguishers, where we give an overall view of our method and clarify its main difference from the previous methods. Then, in Section 4, Section 5, Section 6, and Section 7 we demonstrate the utility of our method to improve boomerang analysis of the block ciphers `WARP`, `CLEFIA`, `TWINE`, `LBlock`, and `LBlock-s`. We conclude in Section 8.

2 Background

2.1 Boomerang and Rectangle Distinguishers

Wagner introduced the boomerang attack at FSE 1999 to exploit two short differentials with high probability [Wag99]. In this attack, we split the targeted cipher E into two parts $E = E_1 \circ E_0$ such that there exist differentials $\Delta_1 \xrightarrow{E_0} \Delta_2$ and $\nabla_2 \xrightarrow{E_1} \nabla_3$ with a high probability p and q, respectively. Then, the two differentials, referred to as upper and lower differentials, are combined as shown in Figure 1 in an adaptively-chosen-plaintext-and-ciphertext setting (ACPC) to distinguish E from a random permutation using algorithm 1. Assuming that the two differentials are independent, the entire probability of the boomerang distinguisher is estimated by p^2q^2. The plaintexts $((P_1, P_2), (P_3, P_4))$ satisfying the boomerang condition are called *right quartets*. As an n-bit random permutation satisfies the condition with probability 2^{-n}, we require $p^2q^2 \gg 2^{-n}$. In that case, the number of required adaptively chosen plain- and ciphertexts to obtain at least one right quartet is approximately $4 \cdot (pq)^{-2}$ and the data/time complexity of constructing a boomerang distinguisher is in $\mathcal{O}\left((pq)^{-2}\right)$.

To remove the requirement a decryption oracle, Kelsey et al. [KKS00] proposed the amplified boomerang attack. This attack was further refined by Biham et al. [BDK01] and called rectangle attack. In this attack, the targeted cipher can be distinguished from a random permutation by querying enough quartets $((P_1, P_2), (P_3, P_4))$ with $P_1 \oplus P_2 = P_3 \oplus P_4 = \Delta_1$ and verifying whether the corresponding ciphertexts $((C_1, C_2), (C_3, C_4))$

Algorithm 1: Boomerang Distinguisher

```
    Input: Encryption and decryption algorithms denoted by $E_k$, $D_k$ respectively
    Output: Distinguishing the targeted cipher from a random permutation
1   Generate $(pq)^{-2}$ different pairs of plaintexts $(P_1, P_2)$ such that $P_2 = P_1 \oplus \Delta_1$;
2   forall pairs $(P_1, P_2)$ do
3       $C_1 \leftarrow E_k(P_1)$,   $C_2 \leftarrow E_k(P_2)$;
4       $C_3 \leftarrow C_1 \oplus \nabla_3$,   $C_4 \leftarrow C_2 \oplus \nabla_3$;
5       $P_3 \leftarrow D_k(C_3)$,   $P_4 \leftarrow D_k(C_4)$;
6       if $P_3 \oplus P_4 = \Delta_1$ then
7           return The underlying oracle is the cipher $E$.
8   return The underlying oracle is a random permutation.
```

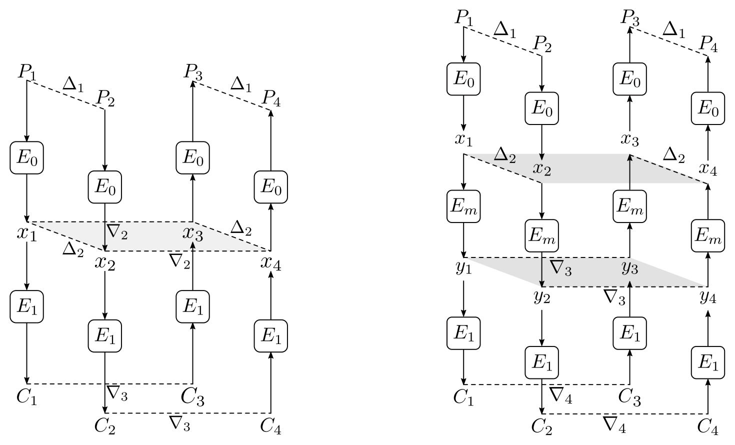

Figure 1: Boomerang distinguisher (left) and sandwich distinguisher (right).

satisfy $C_1 \oplus C_3 = C_2 \oplus C_4 = \nabla_3$ (see Figure 1). If $\Delta_1 \xrightarrow{E_0} \Delta_2$ holds with probability p, then $x_1 \oplus x_2 = x_3 \oplus x_4$ and thus $x_1 \oplus x_3 = x_2 \oplus x_4$ is satisfied with probability p^2. Assuming that these differences are equal to ∇_2, which happens with probability 2^{-n}, then $C_1 \oplus C_3 = C_2 \oplus C_4 = \nabla_3$ holds with probability q^2. As a result, the probability of getting a right quartet $((C_1, C_3), (C_2, C_4))$ is $2^{-n}p^2q^2$. In contrast, a random permutation generates a right quartet with probability 2^{-2n}. Hence, we can distinguish E from a random permutation if $p^2q^2 \gg 2^{-n}$. To produce $2^n(pq)^{-2}$ quartets of ciphertexts, we need $2^{\frac{n}{2}}(pq)^{-1}$ plaintext pairs, so we have to encrypt $4 \cdot 2^{\frac{n}{2}}(pq)^{-1}$ different chosen plaintexts. Although we have to check $\mathcal{O}(2^n(pq)^{-2})$ ciphertext quartets, by using a hash table the time complexity of a rectangle distinguisher can be reduced to $\mathcal{O}(2^{\frac{n}{2}}(pq)^{-1})$. Thus, the data and time complexity of a rectangle distinguisher is in $\mathcal{O}(2^{\frac{n}{2}}(pq)^{-1})$.

In practice, the dependency between the upper differential trail of E_0 and the lower differential trail of E_1 has a significant (positive or negative) impact on the actual probability of the resulting boomerang distinguisher. The importance of this effect was shown in follow-up studies [BK09, Mur11]. To formalize this dependency between the upper and lower differentials, Dunkelman et al. [DKS10, DKS14] introduced the sandwich attack. In this attack, the cipher E is divided into three parts as depicted in Figure 1: $E = E_1 \circ E_m \circ E_0$, where E_m is the middle (inner) part that includes the dependency between the upper and lower differential trails. E_0 and E_1 are also referred to as the outer parts of sandwich distinguishers. The entire probability of a sandwich distinguisher is estimated by p^2q^2r, where the probability $r = r(\Delta_2, \nabla_3)$ of the middle part can be calculated as

$$r(\Delta_2, \nabla_3) = \Pr\left(E_m^{-1}(E_m(x_1) \oplus \nabla_3) \oplus E_m^{-1}(E_m(x_2) \oplus \nabla_3) = \Delta_2 \mid x_1 \oplus x_2 = \Delta_2\right) \quad (1)$$

Because the intermediate differences Δ_2 and ∇_3 can take arbitrary values in the sandwich distinguisher, we can consider the clustering effect. Therefore, a more accurate formula to compute the probability of sandwich distinguisher is:

$$\sum_{\Delta_2, \Delta_2', \nabla_3, \nabla_3' \in \mathbb{F}_2^n} p_{\Delta_2} \cdot p_{\Delta_2'} \cdot r(\Delta_2, \Delta_2', \nabla_3, \nabla_3') \cdot q_{\nabla_3} \cdot q_{\nabla_3'},$$

$$\text{where} \quad p_\Delta = \Pr(\Delta_1 \xrightarrow{E_0} \Delta), \quad q_\nabla = \Pr(\nabla \xrightarrow{E_1} \nabla_4),$$

$$r(\Delta_2, \Delta_2', \nabla_3, \nabla_3') = \Pr\left(E_m^{-1}(E_m(x_1) \oplus \nabla_3') \oplus E_m^{-1}(E_m(x_2) \oplus \nabla_3) = \Delta_2' \mid x_1 \oplus x_2 = \Delta_2\right).$$

2.2 Boomerang Switch

Since the introduction of the sandwich attack, there have been attempts to formulate the probability of the middle part E_m, which is also called the boomerang switch. Starting from the simplest case where the boomerang switch E_m includes only one S-box layer, Cid et al. [CHP+18] proposed the boomerang connectivity table (BCT). This idea was further developed in follow-up works [SQH19, WP19, HBS21] to provide a theoretical framework for evaluating the probability of the middle part when it is composed of multiple rounds. However, the BCT framework only works for block ciphers following the SPN design strategy. To formulate the probability of the boomerang switch over multiple rounds of Feistel ciphers, Boukerrou et al. [BHL+20] proposed the Feistel boomerang connectivity table (FBCT) as the Feistel counterpart of the BCT framework. The setup is depicted in Figure 2.

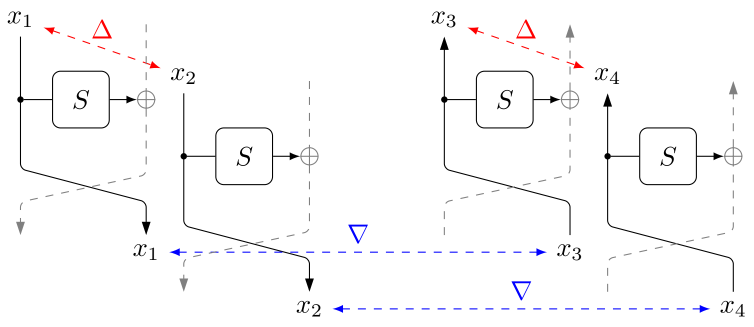

Figure 2: Differences of S-box at four sides of boomerang switch in Feistel structure.

Definition 1 (DDT)**.** Let S be a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ and $\Delta_1 \in \mathbb{F}_2^n$, $\Delta_2 \in \mathbb{F}_2^m$. The differential distribution table (DDT) of S is a $2^n \times 2^m$ table which is defined as follows:

$$\texttt{DDT}(\Delta_1, \Delta_2) := \#\{x \in \mathbb{F}_2^n : S(x) \oplus S(x \oplus \Delta_1) = \Delta_2\}.$$

Definition 2 (FBCT [BHL+20])**.** Let S be a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$, and $\Delta, \nabla \in \mathbb{F}_2^n$. The Feistel boomerang connectivity table (FBCT) of S is a $2^n \times 2^n$ table defined as follows:

$$\texttt{FBCT}(\Delta, \nabla) := \#\{x \in \mathbb{F}_2^n : S(x) \oplus S(x \oplus \Delta) \oplus S(x \oplus \nabla) \oplus S(x \oplus \Delta \oplus \nabla) = 0\}.$$

The FBCT can be used to compute the probability over one round of boomerang switch in a Feistel structure. For fixed Δ and ∇ as depicted in Figure 2, the probability of a returning boomerang over 1 round of a Feistel structure is equal to $2^{-n} \cdot \texttt{FBCT}(\Delta, \nabla)$.

The entry located in row Δ and column ∇ of the FBCT is the number of times the second-order derivative of S becomes zero at point (Δ, ∇). Moreover, $\texttt{FBCT}(\Delta, 0) = \texttt{FBCT}(0, \nabla) = 2^n$ for all $\Delta \in \mathbb{F}_2^n$ and $\nabla \in \mathbb{F}_2^m$, corresponding to the ladder switch [BK09] and $\texttt{FBCT}(\Delta, \Delta) = 2^n$ for all $\Delta \in \mathbb{F}_2^n$, corresponding to the Feistel switch [Wag99]. Let

$\mathcal{X}_{\mathtt{DDT}}(\Delta_1, \Delta_2) := \{x \in \mathbb{F}_2^n : S(x) \oplus S(x \oplus \Delta_1) = \Delta_2\}$ denote the set of valid inputs satisfying the differential transition $\Delta_1 \xrightarrow{S} \Delta_2$. Then, the FBCT can be reformulated [BHL+20]:

$$\mathtt{FBCT}(\Delta_1, \nabla_2) = \sum_{\Delta_2 \in \mathbb{F}_2^m} \#\left(\mathcal{X}_{\mathtt{DDT}}(\Delta_1, \Delta_2) \cap (\mathcal{X}_{\mathtt{DDT}}(\Delta_1, \Delta_2) \oplus \nabla_2)\right).$$

Assuming that Δ in Figure 2 is fixed and ∇ is distributed uniformly and has not been affected by the upper differential trails, the probability of a returning boomerang is:

$$r = \sum_{\nabla} \frac{\mathtt{FBCT}(\Delta, \nabla)}{2^n} \cdot \Pr(x_1 \oplus x_3 = \nabla) = \sum_{\nabla, \delta} \frac{\mathtt{DDT}(\Delta, \delta)}{2^{2n}} \cdot \frac{\#\left(\mathcal{X}_{\mathtt{DDT}}(\Delta, \delta) \cap (\mathcal{X}_{\mathtt{DDT}}(\Delta, \delta) \oplus \nabla)\right)}{\#\mathcal{X}_{\mathtt{DDT}}(\Delta, \delta)}$$
$$= \sum_{\delta} \left(\frac{\mathtt{DDT}(\Delta, \delta)}{2^n}\right)^2,$$

which is the same as the probability calculation according to the traditional boomerang framework, p^2q^2. It can be shown in a similar way that when ∇ is fixed and Δ is independent and uniformly distributed, the probability of a returning boomerang can be calculated based on the p^2q^2 formula. The differences propagated from the upper and lower trails through the middle part are referred to as the upper and lower *crossing differences*. Accordingly, the boundaries of E_m are where the lower and upper crossing differences become uniformly distributed, which mainly depends on the diffusion layer of the targeted cipher as well as the number of active positions in the input/output differences of the middle part. Analogous to the differential uniformity, the boomerang uniformity of an S-box in Feistel structures is defined as follows [BHL+20]:

$$\beta := \max_{\Delta \neq 0, \nabla \neq 0, \Delta \neq \nabla} \mathtt{FBCT}(\Delta, \nabla),$$

which should be small to harden a design against boomerang-like attacks.

To calculate the boomerang switch over multiple rounds, the Feistel boomerang difference table (FBDT) is needed. This table is analogous to the UBCT (Upper BCT or boomerang difference table (BDT)) [WP19] and the LBCT (Lower BCT) [SQH19] in the BCT framework.

Definition 3 (FBDT [BHL+20])**.** Let S be a function from $\mathbb{F}_2^n$ to itself and $(\Delta, \delta, \nabla) \in (\mathbb{F}_2)^3$. The three-dimensional Feistel boomerang difference table (FBDT) is defined as follows:

$$\mathtt{FBDT}(\Delta, \delta, \nabla) := \#\{x \in \mathbb{F}_2^n : S(x) \oplus S(x \oplus \Delta) \oplus S(x \oplus \nabla) \oplus S(x \oplus \Delta \oplus \nabla) = 0, S(x) \oplus S(x \oplus \Delta) = \delta\}.$$

Now, we recall the *ladder switch*, one of the most important switching effects that plays a vital role in our automatic search for sandwich distinguishers. According to Equation 1, if Δ_2, which is propagated from the upper differential transition through the middle part, is zero, then r, the probability of boomerang switch, is one. This also happens if ∇_3, which is propagated from the lower differential transition through the middle part, is zero. Now, assume that the upper and lower crossing differences are propagated with probability one over the middle part. By generalizing the previous argument, it can be seen that if a certain S-box in the middle is activated by at most one of the upper and lower crossing differences, it is "free", i.e., it does not affect the probability of the middle part. In other words, the probability of the boomerang switch only depends on the common active S-boxes between the upper and lower crossing differences over the middle part. Thus, the probability p^2q^2r of the sandwich distinguisher is determined as follows: p and q depend on the number of active S-boxes in E_0 and E_1, while r is depends on the number of common active S-boxes between the upper and lower crossing differences in E_m. As the cost of active S-boxes in the outer parts, E_0 and E_1, is higher than the cost of common active S-boxes in the middle, we can minimize an adequately weighted sum over these active S-boxes to find a sandwich distinguisher with a high probability.

3 Our Method to Find Distinguishers

Our strategy to find sandwich distinguishers is based on Mixed-Integer Linear Programming (MILP) modeling and divided into three phases. First, we identify appropriate upper and lower truncated differential trails. To do this, we optimize the number of active S-boxes in E_0 and E_1 as well as the number of common active S-boxes in E_m. Next, these truncated characteristics are instantiated by concrete differential trails. Finally, by fixing the differences in 4 positions, the input of E_0, the input and output of E_m, and the output of E_1, we compute p, q, r separately to derive the probability p^2q^2r of our distinguisher.

The main difference between our search method and the previous models [HBS21, DDV20] lies in the first step: while these methods utilize a standard truncated model to encode the propagation of truncated differential characteristics in the outer and inner parts of the sandwich distinguisher, following the nature of `BCT` or `FBCT` frameworks, we differentiate between the encoding of truncated trails over the inner and the outer parts of the sandwich distinguisher. Concretely, for the inner part, we model the propagation of truncated differential trails with probability one.

In a standard model of truncated trails, differential cancellation is very likely to happen in the diffusion layer, especially when minimizing the number of active S-boxes. For example, assuming that $z = x \oplus y$ and $x, y, z \in \mathbb{F}_2^n$ for some $n \in \mathbb{N}$, the propagation of truncated differential trails over the XOR operation is normally encoded as follows:

$$X + Y - Z \geq 0, \quad X - Y + Z \geq 0, \quad -X + Y + Z \geq 0,$$

where X, Y, Z are binary variables indicating the activity of x, y, and z respectively. In this model, $(X, Y, Z) = (1, 1, 0)$ is a valid transition. However, according to the `BCT` or `FBCT` frameworks, any common active S-boxes in the middle part of the sandwich distinguisher affect the entire probability of the boomerang switch and should not be neglected. As a result, the standard encoding, where differential cancellation through the diffusion layer is allowed, might indicate too few common active S-boxes.

To build such a model for differential trails of probability one, we need to consider the direction of propagation. This is in contrast to the standard model, where no directionality is encoded. For example, the inequalities describing the truncated model of the XOR operation only describe its differential branch number and are thus symmetric with respect to the input/output variables. However, in the `BCT` or `FBCT` framework, the upper and lower crossing differences must be propagated in forward and backward directions to explore the interaction between active S-boxes of upper and lower trails in the middle part. Therefore, to improve the encoding of truncated boomerang trails and to avoid spurious solutions, we differentiate between the encoding of truncated trails over the outer and the inner parts of the sandwich distinguisher. More precisely, instead of using the same truncated model for the entire upper truncated trail, we encode the propagation of the upper truncated trail through the outer part based on a standard approach while it is propagated forward with probability one through the middle part in our encoding. Similarly, the lower truncated trail is encoded using a standard model for the outer part, while it is propagated backward with probability one over the inner part in our tool. For example, to encode the XOR operation in the middle part of our word-based models, we use the following inequalities:

$$Z - X \geq 0, \quad Z - Y \geq 0, \quad X + Y - Z \geq 0.$$

This excludes the $(X, Y, Z) = (1, 1, 0)$ point from the solution space with the aim of preventing the difference cancellation over the diffusion layer.

Modifying the previous approach [HBS21] accordingly, our method works as follows:

1. We partition the targeted cipher E into $r_0 + r_m + r_1$ rounds for a sandwich distinguisher, as Figure 3 illustrates. Our tool first generates two MILP models with

independent variables to encode the propagation of truncated upper and lower differential trails through $r_0 + r_m$ and $r_m + r_1$ rounds, respectively. We encode the propagation of the upper truncated trail in a standard way over the first r_0 rounds, but switch to encoding the propagation forward with probability one for the last r_m rounds of the upper trail. Similarly, the truncated lower differential trail is propagated backward with probability one over the first r_m rounds, whereas its propagation in the last r_1 rounds is encoded in a standard way.

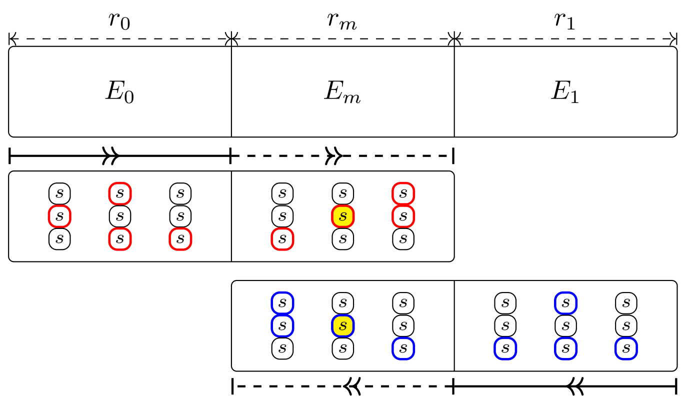

Figure 3: High-level view of common active S-boxes in the boomerang switch.

Next, we encode the common active S-boxes between the upper and lower differential trails in the middle part. We define additional variables to indicate whether a certain S-box is active in both upper and lower truncated trails and use them to link the two MILP models. Let $u_0, \ldots, u_{t-1}$ denote the activity of S-boxes in the last r_m rounds of $E_m \circ E_0$, and $l_0, \ldots, l_{t-1}$ those in the first r_m rounds of $E_1 \circ E_m$, as depicted in Figure 4. Consequently, u_i and l_i correspond to the same S-box positions in the middle part for all $0 \le i \le t-1$. We denote the corresponding t new binary variables by $s_0, \ldots, s_{t-1}$. We use them to link u_i with l_i for all $0 \le i \le t-1$ as follows:

$$u_i - s_i \ge 0, \quad l_i - s_i \ge 0, \quad -u_i - l_i + s_i \ge -1.$$

As a result, $s_i = 1$ if and only if $u_i = l_i = 1$. As Figure 4 illustrates, the binary variables $\tilde{u}_0, \ldots, \tilde{u}_{m-1}$ and $\tilde{l}_0, \ldots, \tilde{l}_{n-1}$ denote the activity of S-boxes in the first r_0 and last r_1 rounds, respectively. We use the constants w_0, w_1, and w_m corresponding to the cost of active S-boxes in E_0, E_m, and E_1 to define our objective function:

$$\min \sum_{i=0}^{m-1} w_0 \cdot \tilde{u}_i + \sum_{j=0}^{t-1} w_m \cdot s_j + \sum_{k=0}^{n-1} w_1 \cdot \tilde{l}_k.$$

2. Next, based on the discovered truncated differential characteristics for E_0 and E_1, our tool looks for the best concrete differential trails instantiating the derived truncated trails over E_0, and E_1. To do so, it generates a bit-wise MILP model encoding the propagation of differential trails. If there is no differential satisfying the derived truncated trails, we go back to step 1 and try again. After deriving the concrete differential trails satisfying the desired activity pattern, we consider the clustering effect of differential characteristics. Therefore, our tool fixes the input/output differences of E_0, and E_1, and computes the differential effect of upper and lower differentials, i.e., $p = \Pr(\Delta_1 \xrightarrow{E_0} \Delta_2)$ and $q = \Pr(\nabla_3 \xrightarrow{E_1} \nabla_4)$. To achieve this, we create a new MILP model where we only fix the input/output difference and search for all compatible differential characteristics. This is computationally feasible in our case as we are working with rather few rounds for E_0 and E_1.

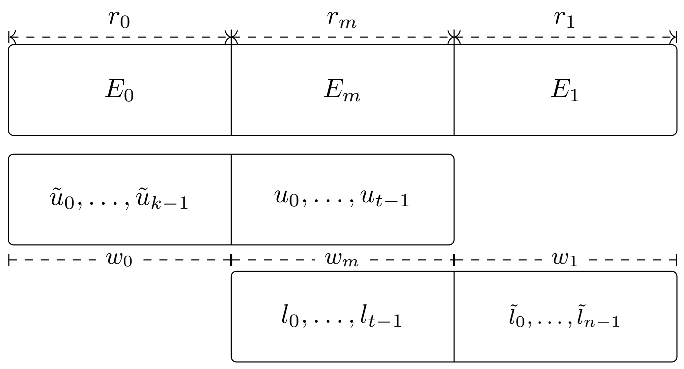

Figure 4: The variables of our MILP model to find truncated upper/lower trails.

3. By using the fixed input/output differences of the middle part, we experimentally evaluate the probability of the boomerang switch according to the following formula:

$$r = \Pr\left(E_m^{-1}(E_m(x_1) \oplus \nabla_3) \oplus E_m^{-1}(E_m(x_2) \oplus \nabla_3) = \Delta_2 \mid x_1 \oplus x_2 = \Delta_2\right).$$

   The number of common active S-boxes provides an initial estimate for r which allows us to deduce the number of trials needed to experimentally estimate r. We always make sure that the number of trials is significantly larger than r^{-1}. If $r = 0$, the discovered upper and lower differential trails are either incompatible, or combining them yields a weak distinguisher. If so, we go back to step 1 and repeat the process.

4. In the final step, we compute the entire probability p^2q^2r of the discovered sandwich distinguisher. To make sure that the computed probability is accurate enough, we perform an additional check. The accuracy of the estimated probability is highly related to correctly allocating the boundaries of the middle part. If the middle part is too short, the upper and lower crossing differences are not uniformly distributed at the boundaries of E_m and the p^2q^2r formula underestimates the actual probability. On the other hand, if we choose E_m too large, the probability of the middle part decreases, and more computational power is required to evaluate the probability based on theoretical and experimental frameworks. A good indicator to see whether the middle part was appropriately allocated is extending E_m by a few rounds and comparing the value p^2q^2r with the experimental probability of the extended E_m. If the boundaries of E_m are chosen appropriately, p^2q^2r gives a good estimate of the actual probability. Otherwise, we need to extend E_m.

Our tool is very easy to use and extend for other ciphers. It receives the lengths r_0, r_m, r_1 of the partitions in sandwich distinguishers as well as the cost w_0, w_m, w_1 of active S-boxes in each partitions. It outputs the discovered truncated trails, the total number of active S-boxes, the number of common active S-boxes in the middle part, and the concrete differential trails covering E_0 and E_1. Note that our sandwich distinguishers do not rely on individual differential characteristics and instead use more accurate estimates for the probability of the differential and the boomerang switch. More precisely, when computing p, q, and r, our tool fixes the differences at four positions only: the input of E_0, the input and output of E_m (on two different sides of boomerang switch), and the output of E_1. All other differences are unrestricted. Besides the main outputs, our tool generates a figure representing the propagation of upper and lower differential trails through different parts of the sandwich distinguisher (e.g., Figure 6 and Figure 12) which not only gives us intuition about correctly allocating E_m but also makes the manual verification of our discovered distinguishers much easier.

To perform MILP optimization, we use the Gurobi solver [Gur22]. We also use this solver to compute the differential effect by setting the `PoolSearchMode` parameter to 2, which instructs Gurobi to find the n best solution, where n is a very large parameter. To avoid deriving multiple solutions corresponding to the same differential characteristic and thus counting it twice, we only define dummy variables in our bit-oriented MILP models when strictly necessary. Even if we have to define dummy variables (e.g., to encode the large binary matrices of CLEFIA), we ensure no extra solutions are created. In other words, we make sure that there is a one-to-one correspondence between the solution space of our models and the possible differential trails of the targeted cipher.

By adjusting the weights w_0, w_m, and w_1 in step 1, we can find sandwich distinguishers with different probabilities. In addition, if the targeted cipher employs different S-boxes with different differential uniformity or different (Feistel) boomerang uniformity, we can use a different weight for each S-box in our objective function to appropriately adjust the cost of its activity in the resulting truncated boomerang trail. To show the utility of our method, we demonstrate its application for several Feistel ciphers in the next sections.

To encode the differential behavior of nonlinear operations, particularly S-boxes, we have implemented the methods introduced in previous works [AST+17, SWW18, AK18]. We use the off-the-shelf logic minimization tool ESPRESSO, from the University of California, Berkeley, to simplify the extracted MILP constraints. ESPRESSO includes the efficient implementation of the Espresso [BHMSV84] algorithm and supports both fast and exact logic minimization. Unlike Logic Friday, a closed-source Windows program supporting Boolean functions with at most 16 input variables, ESPRESSO is an open-source tool that supports Boolean functions with any input sizes. It also outperforms the results derived by Logic Friday for large S-boxes [YK21]. Given that extracting and simplifying the MILP (or SMT/SAT) constraints encoding the differential and linear behaviors of S-boxes is important in automatic differential and linear analysis, we have implemented this part of our tool as a subclass of the `Sbox` class in SageMath [Sag22]. Thus, other researchers can use our S-box encoding tool with ease. Appendix H briefly describes the usage of our SageMath module to derive and simplify the constraints encoding the DDT of S-boxes.

4 Application to WARP

In this section we briefly describe the specification of WARP and then illustrate the efficiency of our tool to significantly improve the sandwich distinguishers of this cipher.

4.1 WARP

WARP is a lightweight block cipher that was proposed by Banik et al. at SAC 2020 [BBI+20]. It receives a 128-bit plaintext and a 128-bit master key and then performs 40 full rounds as represented in Figure 5 plus one partial round (without nibble permutation) to produce a 128-bit ciphertext. Employing a 32-branch generalized Feistel structure (GFS), WARP aims at providing 128-bit security in the single-key setting while achieving a small footprint.

The internal state of WARP can be represented as $X = X_0 || \cdots || X_{31}$, where $X_i \in \{0,1\}^4$. WARP splits the 128-bit master key K into two 64-bit halves, $K = K^0 || K^1$. $K^{(r-1) \bmod 2}$ is used as the round-key in the rth round. As shown in Figure 5, the round function of WARP applies the same 4-bit S-box and round-key addition to one of each two consecutive nibbles of the internal state. Afterwards, a permutation π is applied to the nibbles of the state. We refer to design paper [BBI+20] for a full specification. We use $X^{(r)}$ to denote the input state of round $r+1$. To denote the input difference of round $r+1$ in upper and lower trails of sandwich distinguishers, we use $\Delta X^{(r)}$ and $\nabla X^{(r)}$. In addition, we use $\Delta X_i^{(r)}$ (or $\nabla X_i^{(r)}$) to denote the difference of the ith nibble in the input of round $r+1$.

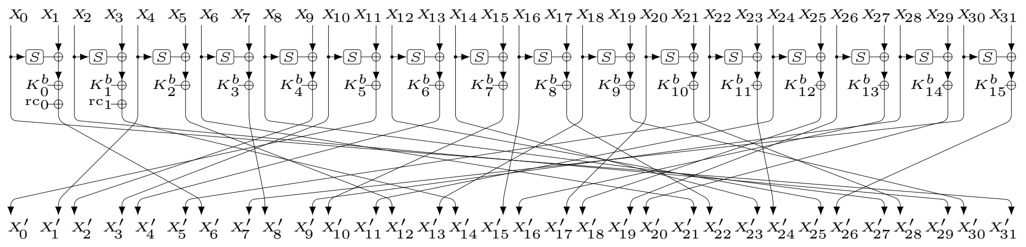

Figure 5: The round function of WARP.

4.2 Sandwich Distinguishers for WARP

WARP's designers investigated the security of their cipher against well-known attacks on block ciphers. For instance, by applying the automatic methods for differential and linear cryptanalysis, they found a 21-round impossible differential distinguisher for WARP and computed the minimum number of differentially and linearly active S-boxes for up to 19 rounds of this cipher. They also applied the division property to find a 20-round integral distinguisher. WARP has also received third-party analysis which mostly focused on differential cryptanalysis [TB21, KY21]. For example, by employing the FBCT framework, Teh and Biryukov [TB21] investigated the security of WARP against boomerang attacks and introduced a 21-round sandwich distinguisher with probability $2^{-121.11}$ building upon which they mounted a key-recovery attack on 24 rounds. This 21-round distinguisher has been discovered with an automatic tool that takes the FBCT into account. However, as the FBCT only handles the boomerang switch over one round, this tool also only considers the boomerang switch for one round in the middle.

Here, using our method, we take advantage of boomerang switches up to 10 rounds of WARP. This not only enables us to dramatically improve the probability of the sandwich distinguishers but also allows us to improve the sandwich distinguishers of WARP by two rounds and distinguish up to 23 rounds of WARP from a random permutation. WARP achieves nibble-wise full diffusion after 10 rounds for both encryption and decryption. Hence, if only one nibble is active at the input (output) of the middle part, the upper (lower) crossing differences become almost uniformly distributed after 10 rounds. Consequently, 10 rounds are a good choice for the length of the middle part. The FBCT of WARP's S-box is shown in Table 6 and its differential and (Feistel) boomerang properties are listed in Table 9. As the FBCT shows, two difference values $\{2, \mathtt{a}\}$ result in a better boomerang switch compared to the other difference values. Thus, we limit the input/output differences of the middle part to 2 and a. This guides our tool to find better bit-wise differences for the boundary between middle and outer parts when instantiating the truncated trails.

As the lower and upper crossing differences are propagated with probability one in our tool, we are able to find the longest deterministic nibble-level sandwich distinguisher. To do so, we set the length r_0, r_1 of the outer parts to zero and increase r_m as long as there is a deterministic sandwich distinguisher. Accordingly, we discover that there are 9-round deterministic sandwich distinguishers for WARP. One of these is listed in Table 2 and illustrated in Figure 6. This automatically generated figure shows that there is no interaction between the propagation of upper crossing differences (red) and lower crossing differences (blue). As a result, the probability of our 9-round distinguisher is one due to the ladder switch. Compared to our 9-round deterministic distinguisher, the best differential covering 9 rounds has a probability of 2^{-28}.

Now, we set $(r_0, r_m, r_1) = (2, 10, 2)$ and apply our tool to find a 14-round sandwich distinguisher. If $r_0 + r_1 > 0$, choosing appropriate weights for the active S-boxes in the middle and outer parts affects the identified distinguisher. Given that WARP employs the

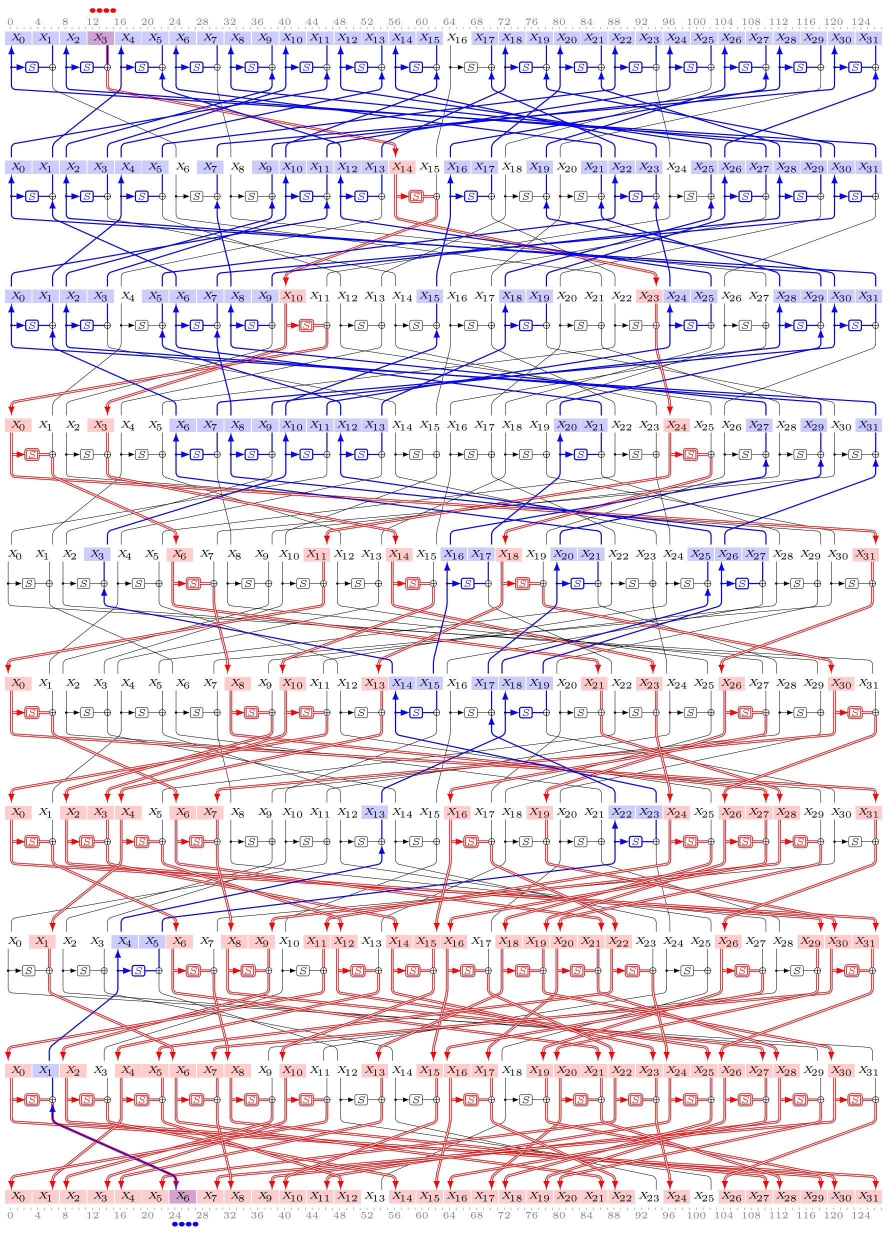

Figure 6: Sandwich distinguisher for 9 rounds of `WARP`.

Table 2: 9-round deterministic sandwich distinguisher for `WARP`

$r_0 = 0, r_m = 9, r_1 = 0,\ p^2q^2r = 1$ for all $\delta, \gamma \in \mathbb{F}_2^4 \setminus \{0\}$			
$\Delta X^{(0)}$	000δ0000000000000000000000000000	$\nabla X^{(9)}$	000000γ0000000000000000000000000

same S-box in each round and taking the p^2q^2r formula into account, we set the weight of active S-boxes as $(w_0, w_m, w_1) = (2, 1, 2)$. This guides our tool to find a near-optimal sandwich distinguisher for 14 rounds. The resulting distinguisher is listed in Table 3. Figure 12 (Appendix) shows how the differences are propagated through each part of this distinguisher with common active S-boxes marked in yellow.

Table 3: 14-round sandwich distinguisher for `WARP`

$r_0 = 2, r_m = 10, r_1 = 2,\ p = 2^{-4},\ q = 2^{-4},\ r = 2^{-4.58},\ p^2q^2r = 2^{-20.58}$			
$\Delta X^{(0)}$	`0000000000000a00000000a500000000`	$\Delta X^{(2)}$	`0a000000000000000000000000000000`
$\nabla X^{(12)}$	`0000a000000000000000000000000000`	$\nabla X^{(14)}$	`0000f0a000000000000000000f000000`

Figure 12 shows that the middle part has been allocated properly: After 9 rounds, there are still some zero differences, e.g., $\Delta X_{15}^{(11)}$ among the upper crossing differences and $\nabla X_{28}^{(3)}$ from the lower crossing differences, which means the propagated upper and lower differences are not uniformly distributed after 9 rounds. However, after one additional round, the upper and lower crossing differences become almost uniformly distributed. Hence, choosing 10 rounds for the middle part is sufficient.

To motivate our approach of evaluating the probability of the Boomerang switch experimentally, we compare it to the theoretical estimates of the `FBCT` framework [BHL+20]. For the theoretical estimate, we use lower case Greek letters such that $\alpha_j^{(r)}$ and $\beta_j^{(r)}$ denote the upper and lower crossing differences in the jth nibble of $X^{(r)}$. As evident from Figure 12, there are 3 common active S-boxes between the upper and lower trails. Additionally, for the common active S-box in round 4, the input upper and lower crossing differences are $\alpha_6^{(3)}$ and $\beta_6^{(3)}$, where $\alpha_6^{(3)}$ is fixed and $\beta_6^{(3)}$ originates from $\beta_4^{(12)}$ in the lower differential trail. Therefore, its switching effect can be formulated by $\texttt{FBDT}(\alpha_6^{(3)}, \alpha_8^{(4)}, \beta_6^{(3)})$. However, given that the upper crossing difference $\alpha_8^{(4)}$ does not affect the other two common active S-boxes, we can simply use $\texttt{FBCT}(\alpha_6^{(3)}, \beta_3^{(3)})$ to formulate the switching effect of the common active S-box in round 4. Concerning the common active S-box in the 8th round, the upper and lower crossing differences at the input of this S-box are $\alpha_{28}^{(7)}$ and $\beta_{28}^{(7)}$. Since the output differences of the common active S-box in round 8 do not affect the other two common active S-boxes, its boomerang switch can be formulated by $\texttt{FBCT}(\alpha_{28}^{(7)}, \beta_{28}^{(7)})$. Tracking the propagation of $\alpha_{28}^{(7)}$ backward in Figure 12 shows that this upper crossing difference originates from $\alpha_6^{(3)}$ such that $\alpha_6^{(3)} \xrightarrow{\texttt{DDT}} \alpha_{20}^{(6)} \xrightarrow{\texttt{DDT}} \alpha_{28}^{(7)}$. It can also be seen that $\beta_{19}^{(10)} \xrightarrow{\texttt{DDT}} \beta_8^{(8)} \xrightarrow{\texttt{DDT}} \beta_{28}^{(7)}$. For the common active S-box in round 11, the lower crossing difference at the input of this S-box is $\beta_4^{(12)}$, which is fixed by the lower trail, and the input upper crossing difference at the input of this S-box is $\alpha_{18}^{(10)}$. Consequently, the boomerang switch of the common active S-box in round 11 can be formulated by $\texttt{FBDT}(\beta_4^{(12)}, \beta_{19}^{(10)}, \alpha_{18}^{(10)})$. Moreover, $\alpha_{18}^{(10)}$ originates from $\alpha_{20}^{(6)}$ such that $\alpha_{20}^{(6)} \xrightarrow{\texttt{DDT}} \alpha_{24}^{(9)} \xrightarrow{\texttt{DDT}} \alpha_{18}^{(10)}$. As a result, the total probability of the boomerang switch over the 10 middle rounds of our distinguisher is

$$\begin{aligned} r(\alpha_6^{(3)}, \beta_4^{(12)}) = 2^{-10\times 4} \cdot \sum & \texttt{FBCT}(\alpha_6^{(3)}, \beta_6^{(3)}) \cdot \texttt{DDT}(\beta_4^{(12)}, \beta_6^{(3)}) \cdot \texttt{FBCT}(\alpha_{28}^{(7)}, \beta_{28}^{(7)}) \\ & \cdot \texttt{DDT}(\alpha_6^{(3)}, \alpha_{20}^{(6)}) \cdot \texttt{DDT}(\alpha_{20}^{(6)}, \alpha_{28}^{(7)}) \cdot \texttt{DDT}(\beta_{19}^{(10)}, \beta_8^{(8)}) \\ & \cdot \texttt{DDT}(\beta_8^{(8)}, \beta_{28}^{(7)}) \cdot \texttt{FBDT}(\beta_4^{(12)}, \beta_{19}^{(10)}, \alpha_{18}^{(10)}) \\ & \cdot \texttt{DDT}(\alpha_{20}^{(6)}, \alpha_{24}^{(9)}) \cdot \texttt{DDT}(\alpha_{24}^{(9)}, \alpha_{18}^{(10)}), \end{aligned} \tag{2}$$

where the summation is over all the possible values of $\beta_6^{(3)}, \beta_4^{(12)}, \alpha_{28}^{(7)}$, $\beta_{28}^{(7)}, \alpha_{20}^{(6)}, \beta_{19}^{(10)}$, $\beta_8^{(8)}, \beta_{28}^{(7)}, \alpha_{24}^{(9)}$, and $\alpha_{18}^{(10)}$. To speed up the computation, it is possible to split Equation 2 into some precomputed tables according to Equation 3. We computed the above formula for several possible values of $(\alpha_6^{(3)}, \beta_4^{(12)})$. For example, if $(\alpha_6^{(3)}, \beta_4^{(12)}) = (\texttt{a}, \texttt{a})$ we obtain $r = \frac{45801799680}{2^{40}} = 2^{-4.58}$, which matches the experimental probability. Table 4 compares the value of this formula with the experimental probability for some further input/output differences $(\alpha_6^{(3)}, \beta_4^{(12)})$. Evidently, choosing $(\alpha_6^{(3)}, \beta_4^{(12)})$ from $\{(2, 2), (\texttt{a}, \texttt{a})\}$ results in a greater probability for the middle part, which is expected according to the FBCT (Table 6). To evaluate the experimental values in Table 4, we set up several random tests including 2^{26} boomerang queries corresponding to 2^{13} random keys with 2^{13} random messages each and compute the average number of returned boomerangs.

Table 4: Comparison between the theoretical and experimental probabilities of the boomerang switch over 10 rounds of WARP

Input/Output Differences	Equation 2	Exp. Probability
$(\Delta X_6^{(3)}, \nabla_4^{(12)}) = (\mathtt{a}, \mathtt{a})$	$2^{-4.5853}$	$2^{-4.5848}$
$(\Delta X_6^{(3)}, \nabla_4^{(12)}) = (2, 2)$	$2^{-6.9714}$	$2^{-6.9711}$
$(\Delta X_6^{(3)}, \nabla_4^{(12)}) = (5, 7)$	$2^{-7.0428}$	$2^{-7.0432}$
$(\Delta X_6^{(3)}, \nabla_4^{(12)}) = (9, \mathtt{b})$	$2^{-8.6165}$	$2^{-8.6128}$
$(\Delta X_6^{(3)}, \nabla_4^{(12)}) = (\mathtt{c}, 5)$	$2^{-9.0670}$	$2^{-9.0644}$
$(\Delta X_6^{(3)}, \nabla_4^{(12)}) = (\mathtt{a}, 4)$	$2^{-11.5070}$	$2^{-11.5050}$
$(\Delta X_6^{(3)}, \nabla_4^{(12)}) = (3, 3)$	0	0

Now, we examine the overall probability of our 14-round sandwich distinguisher. Our tool calculates the probability of the differentials over E_0 and E_1 as $p = q = 2^{-4}$ which takes the clustering effect into account. As $r(\mathtt{a}, \mathtt{a}) = 2^{-4.58}$ (see Table 4), the total probability of our 14-round distinguisher is $2^{-20.58}$. This is large enough to be experimentally verified on an ordinary laptop.

To perform this experimental verification, we evaluate 2^{28} boomerang queries corresponding to 2^{10} random keys with 2^{18} random messages each, and compute the average number of returned boomerangs. Accordingly, the experimental value for the whole 14-round distinguisher is $2^{-20.39}$, which is very close to our estimate based on the p^2q^2r formula. The best differential effect reported for 14 rounds of WARP so far is $2^{-72.14}$ [TB21]. This shows the advantage of sandwich distinguishers compared to differential distinguishers for reduced-round WARP.

To discover a 15-round sandwich distinguisher, we partition the cipher using $(r_0, r_m, r_1) = (2, 10, 3)$. Interestingly, the sandwich distinguishers discovered in this setting are the extension of our 14-round sandwich distinguisher by one round forward. As a result, we achieve a 15-round distinguisher with $p = 2^{-4}, q = 2^{-8}$, and $r = 2^{-4.58}$ as it is described in Table 10 with a total probability of $2^{-28.58}$. To experimentally verify our 15-round distinguisher for WARP, we carried out several random tests, including 2^{34} random boomerang queries corresponding to 2^{10} random keys with 2^{24} messages each. The experimental probability for the whole 15-round distinguisher is $2^{-28.33}$, which is very close to p^2q^2r and thus verifies the validity of our estimate.

Similar to before, we apply our tool on up to 23 rounds of WARP. The full specification of our sandwich distinguishers for 15, 16, and 20 to 23 rounds of WARP is listed in Table 10. These results improve the best sandwich distinguishers for WARP on 20 and 21 rounds by a factor of $2^{38.28}$ and $2^{36.56}$, respectively. We even distinguish 22 and 23 rounds of WARP with great advantage from a random permutation for the first time.

When comparing the number of active nibbles at input/output differences in our 20- and 21-round sandwich distinguishers with the best previous ones [TB21], we find that our distinguishers not only have much higher probabilities but also have fewer active nibbles in the input/output differences. This is advantageous when building key-recovery attacks. For instance, the number of active nibbles at the input/output of our 21-round distinguisher is 14, compared to previously 17 [TB21].

5 Application to CLEFIA

CLEFIA[SSA+07] is a 128-bit block cipher supporting key lengths of 128, 192, and 256 bits which are compatible with AES. Designed by Sony Corporation, CLEFIA was introduced in FSE 2007 and is internationally standardized in ISO/IEC 29192-2. Depending on the key size, the number of rounds in CLEFIA is 18 (128-bit key), 22 (192-bit key), and 26 (256-bit key). As shown in Figure 7, the round function of CLEFIA uses the generalized Feistel structure with four 32-bit branches in which two 32-bit functions F_0 and F_1 are applied in parallel. F_0 and F_1 follow the SP structure and perform three basic operations, including sub-key addition, application of four 8-bit S-boxes in parallel, and diffusing the output bytes of the S-box layer by applying a 4×4 MDS matrix over $\mathbb{F}_{2^8}$. As Figure 7 shows, CLEFIA employs two different S-boxes which are used in different order in F_0 and F_1. Moreover, the diffusion mechanism of CLEFIA was designed based on a novel design technique called Diffusion Switching Mechanism (DSM) [SS04, SSA+07] according to which two different MDS matrices with a certain property are used in the two branches. This guarantees a larger minimum number of active S-boxes in comparison to an ordinary GFS cipher without DSM. For a full specification, we refer the reader to [SSA+07]. Consistent with the previous sections, we use $X^{(r)}$ to denote the input state of the $r+1$th round and denote the differences in forward and backward directions by $\Delta X^{(r)}$ and $\nabla X^{(r)}$, respectively.

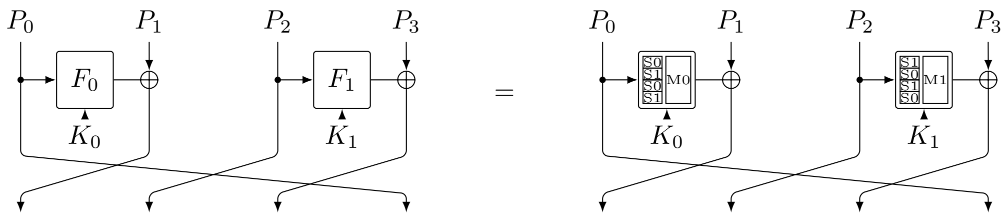

Figure 7: CLEFIA round function.

CLEFIA's security has been investigated by its designers as well as many other researchers [TTS+08, MDS11, Tez10, LWZ11, BGW+13, BN19]. The longest distinguishers for CLEFIA so far are 9-round impossible differential distinguishers [SSA+07, TTS+08], a 9-round integral distinguisher [LWZ11], and a 9-round zero-correlation distinguisher [BGW+13]. However, regarding the differential and boomerang analysis, the designers only estimated some upper bounds for the probability of differential and sandwich distinguishers based on the minimum number of differentially active S-boxes, and the best sandwich distinguisher for CLEFIA so far covers 8-round with probability 2^{-92} [MQ14]. Here, we not only improve the probability of the best previous 8-round sandwich distinguisher of CLEFIA by a factor of $2^{15.97}$, but also introduce a 9-round sandwich distinguisher for the first time. Thus, we contradict the claim by Biryukov and Nikolic that 9-round boomerang distinguishers for CLEFIA do not exist [BN19]. Still, their conclusion that 12 rounds of CLEFIA resist boomerang attacks holds up.

5.1 Sandwich Distinguishers for CLEFIA

The differential and (Feistel) boomerang properties of the employed S-boxes in CLEFIA are briefly described in Table 9. As can be seen, the S-box S_0 is weaker against differential and boomerang attacks. More precisely, the maximum differential probabilities of S_0 and S_1 are $2^{-4.68}$ and 2^{-6} respectively. In addition, the Feistel boomerang uniformity of S_0 is 20, whereas the Feistel uniformity of S_1 is 4. Therefore, in contrast to our truncated MILP model for WARP where the cost of active S-boxes is only determined by the parameters w_0, w_m, w_1, we treat S_0 and S_1 differently in our truncated MILP model for CLEFIA.

Concretely, depending on which part of the sandwich distinguisher the active S-box is located in and which S-boxes it is, we use $4.68 \cdot w$ and $6 \cdot w$ for $w \in \{w_0, w_m, w_1\}$ as the actual weight of S_0 and S_1, respectively. Additionally, we have to take the DSM property of CLEFIA into account. According to DSM, the difference cancellation, which may occur in multiple rounds of a normal GFS cipher, is prevented in CLEFIA thanks to a clever way of choosing two different MDS matrices for the diffusion layer. To model the DSM, we adopt the method introduced by Sajadieh and Vaziri [SV18] in our truncated MILP model to avoid activity patterns excluded by the DSM property of CLEFIA.

Our bit-wise MILP model for CLEFIA is much heavier compared to WARP's bit-wise MILP model, which is mainly due to the large (8-bit) S-boxes and large MDS matrices employed by CLEFIA. Table 5 briefly describes the number of constraints derived by our tool to encode the differential behavior of S_0 and S_1.

Table 5: Number of MILP constraints to encode the *pb*-DDTs of CLEFIA

S-box	Probability	# Zero entries in sub-DDT	# Constraints
S_0	2^{-7}	46035	6556
	2^{-6}	60499	2854
	$2^{-5.4}$	64688	779
	2^{-5}	65417	173
	$2^{-4.7}$	65527	30
S_1	2^{-7}	33406	7917
	2^{-6}	65281	320

As before, we apply our tool to find the longest deterministic sandwich distinguisher for CLEFIA which results in a 3-round distinguisher. To search for longer distinguishers, we set the length of the middle part in our sandwich distinguishers to 4 or 5 rounds. As CLEFIA reaches full diffusion on byte-level after 5 rounds, this is sufficient to model the dependency between the upper and lower trails. The specification of our sandwich distinguishers for 4 to 8 rounds is listed in Table 11. As evident from the table, by partitioning 7 rounds into $1 + 5 + 1$ rounds, we discover a practical sandwich distinguisher with probability $2^{-32.67}$. To the best of our knowledge, this is the first 7-round distinguisher for CLEFIA which can be experimentally verified with a very limited computational power. The minimum number of differentially active S-boxes over the 7 rounds is 14, and hence the probability of a 7-round differential characteristic is at most $2^{-14\times4.68} = 2^{-65.52}$. In reality, the probability will be much lower as the stronger S-box S_1 will also be active. Consequently, even if we take the clustering effect into account, there is still a huge gap between the probability of our 7-round sandwich distinguisher and the best possible 7-round differential for CLEFIA. For 8 rounds, we split the cipher into $2 + 5 + 1$ rounds and discover a distinguisher with a probability of $2^{-76.03}$. Similarly, for 9 rounds, we split the cipher into $2 + 4 + 3$ rounds and find a distinguisher with a probability of $2^{-99.12}$, which is illustrated in Figure 8.

The huge gap between the probabilities of our sandwich distinguishers for 7 and 8 rounds is due to the strong diffusion property of CLEFIA as well as a limitation of our method. The diffusion switching mechanism (DSM) [SS04, SSA+07] of CLEFIA comes into effect for more than 7 rounds and increases the minimum number of active S-boxes by up to 40% in comparison to a normal GFS without DSM. This also increases the number of active S-boxes in the middle part of our sandwich distinguisher. Unfortunately, when the number of common active S-boxes in the middle increases, computing the probability based on either theoretical frameworks or by experimental approach becomes infeasible. Therefore, when applying our tool for more than 7 rounds of CLEFIA, we constrain it to find sandwich distinguishers with a limited number of common active S-boxes so that we can compute the probability of the middle part in a reasonable time. Due to the

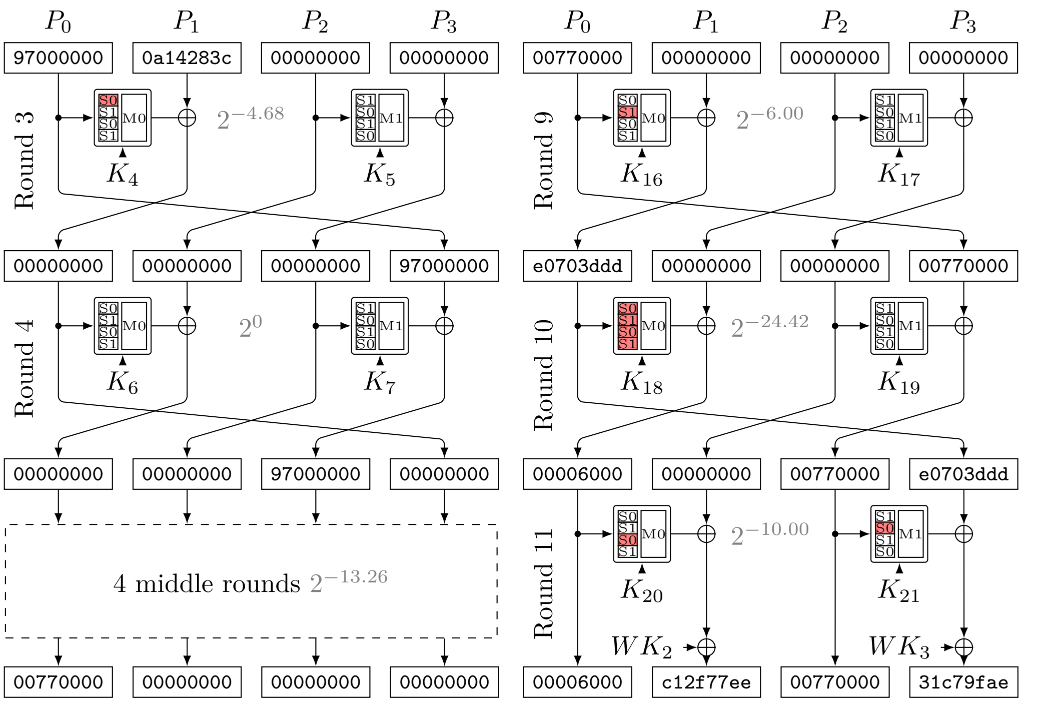

Figure 8: CLEFIA: 9-round boomerang distinguisher with probability $2^{-99.12}$ based on 2-round upper trail (left, probability $2^{-4.68}$), 4 middle rounds (left, $2^{-13.26}$), and 3-round lower trail (right, probability $2^{-38.25}$ including differential effect).

importance of CLEFIA as a standardized cipher, we propose a key-recovery attack on 11 rounds of CLEFIA based on our 9-round sandwich distinguisher.

5.2 Key-recovery Attack on 11 rounds of CLEFIA

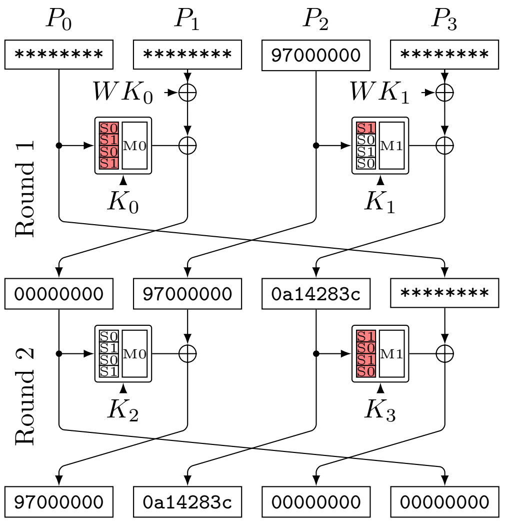

Figure 9: CLEFIA: Key-recovery for 2 initial rounds.

We propose a key-recovery attack on 11 rounds of CLEFIA by prepending two rounds before our 9-round sandwich distinguisher. Our attack has a time complexity of $2^{116.1}$, a data complexity of $2^{103.13}$, and a memory complexity of $2^{113.6}$.

To acquire the pairs, we use the initial structure depicted in Figure 9. This structure is built such that for each of the 2^{97} elements, there is a second element that leads to the required difference at the beginning of round 3. Therefore, we get 2^{96} pairs for 2^{97} encryption and decryption queries (2^{98} data). To arrive at $4 \cdot 2^{99.12}$ pairs for the distinguisher, we need $2^{5.13} \approx 35$ of these structures. Therefore, we expect at least one right pair for the distinguisher with a probability of 98 %.

We carry out our attack in the following steps, which we repeat for each of the $2^{5.13}$ initial structures.

1. Generate 2^{97} plaintexts according to the initial structure.

2. For each plaintext P, obtain $C = E(P)$ and $P' = E^{-1}(C \oplus \nabla)$. Store (P, P', C) in the list $\mathcal{P}$. This leads to our data complexity of $2^{5.13} \times 2 \times 2^{97} = 2^{103.13}$. Note that while we iterate over all 2^{32} values in the rightmost 32 bits in P, we only care about pairs with a difference that can generate a difference of `0a14283c` after being XORed with the output of `MixColumns`. Concretely, we want pairs with a difference in the set $\mathcal{H} = \{M_1(S_1(x) \oplus S_1(x \oplus \mathtt{97}), 0, 0, 0) \oplus \mathtt{0a14283c}\}$ with $|\mathcal{H}| = 127$.

3. For each of the 2^{97} $(P, P', C) \in \mathcal{P}$ and the each of the $(\Delta P, \Delta P') \in \mathcal{H} \times \mathcal{H}$ possible differences in the rightmost 64 bits of P and P', find a matching second element $(Q, Q', D) \in \mathcal{P}$. We require $\mathrm{LSB}_{64}(P \oplus Q) = \mathtt{97000000} \parallel \Delta P$ and $\mathrm{LSB}_{64}(P' \oplus Q') = \mathtt{97000000} \parallel \Delta P'$. Therefore, we expect to find $2^{97+97+14-64-32-1} = 2^{111}$ matching quartets. Accounting for the $2^{5.13}$ initial structures, this step requires about $2^{5.13+97+14} = 2^{116.1}$ time complexity.

4. For each of the 2^{111} quartets (P, P', C, Q, Q', D), find the possible values for K_0 based on P and P' as well as based on Q and Q'. This can be done efficiently by using the $\mathcal{X}_{\mathtt{DDT}}$ of the S-boxes. For each transition, we expect one valid key candidate on average. Note that the key candidates for both pairs need to match. As this happens with a probability of 2^{-32}, we are left with about 2^{79} quartets.

5. Next, we target 8 bits of K_1 based on the transition of the relevant S-box in the first round. We expect two candidates on average, as each quartet is already filtered to only contain the 127 valid differences after the S-box. Due to the structure of the $\mathcal{X}_{\mathtt{DDT}}$, either both candidates or no candidates match, and we expect a match with probability 2^{-7}. Thus, we are left with 2^{72} quartets. Note that when the key candidates match, we always get two candidates: k and $k \oplus \mathtt{97}$.

6. Now, we recover $K_3 \oplus WK_1$. The transition in the relevant S-boxes of F_1 in the second round also depends on K_1. For each of the 2^{25} candidates of K_1, we find the relevant candidates for $K_3 \oplus WK_1$. These candidates match with a probability of 2^{-32}. Thus, we expect to be left with 2^{65} quartets.

7. Now, we consider the final round by targeting 8 bits of K_{20} and K_{21} each. This further reduces the number of quartets by 2^{-14} to 2^{51}.

8. Next, we target $K_{18} \oplus WK_3$. We consider all 2^{25} values for K_{21} and expect a match with a probability of 2^{-32} and thus reduce the number of quartets to 2^{44}.

9. With the number of quartets reduced to 2^{44} and each quartet only compatible with a few candidates for K_0 and K_1, we can brute-force the remaining 64 bits of K_2 and K_3. As $K_0, \ldots, K_3$ are calculated by applying 12 Feistel rounds to $WK_0, \ldots, WK_3$, we unfortunately cannot use the additional key material to speed up this process. Accounting for the fact that we have $2^{5.13}$ initial structures leading 2^{44} quartets each, this step needs about $2^{5.13+44+64} = 2^{113}$ time complexity.

The overall time complexity is dominated by step 3, where we identified the set of quartets with valid input and output differences. Thus, we get a total time complexity of $2^{116.1}$. The memory complexity of $2^{113.6} = 6 \times 2^{111}$ is dominated by the need to store 2^{111} quartets.

6 Application to TWINE

Now, we apply our tool to TWINE, a 64-bit block cipher which supports key sizes of 80 and 128 bits [SMMK12]. This cipher uses a Type-2 generalized Feistel structure with 16 4-bit branches. Both variants perform 36 applications of the round function illustrated in Figure 10. The round function includes a nonlinear layer consisting of 8 parallel applications of the same 4-bit S-box and a diffusion layer permuting the 16 nibbles.

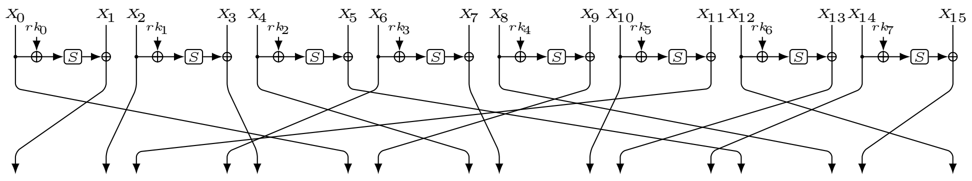

Figure 10: The round function of TWINE.

Table 8 shows the FBCT of TWINE's S-box. Additionally, the differential and boomerang properties of this S-box are briefly described in Table 9. The table shows that the differential and F-boomerang uniformity of TWINE's S-box are 4. As the minimum F-boomerang uniformity of a non-APN function is 4 [BHL+20], TWINE's S-box achieves optimal values for differential and F-boomerang uniformity for 4-bit S-boxes. Hence, the probability of the boomerang switch mainly depends on the position of active nibbles at the input/output differences rather than the concrete difference in these nibbles. Consequently, we expect that our method finds nearly optimal sandwich distinguishers for TWINE.

We apply our tool to find good sandwich distinguishers for TWINE. When searching for deterministic sandwich distinguishers, we find that there is a 5-round sandwich distinguisher with probability 1. To find longer distinguishers, we set the length of the middle part to 8 rounds as TWINE achieves full nibble-wise diffusion after 8 rounds. Our distinguishers for 13 to 16 rounds of TWINE are listed in Table 12. Figure 13 shows our 16-round sandwich distinguisher. Our distinguishers have a higher probability than the best previous sandwich distinguishers [TB21], while the number of active nibbles of input/output differences in our distinguishers remains the same as before [TB21].

7 Application to LBlock and LBlock-s

In this section, we apply our tool to LBlock, a 64-bit block cipher with 80-bit keys proposed at ACNS 2011 [WZ11]. As shown in Figure 11, the round function of LBlock follows a 2-branch balanced Feistel structure, where the right branch is modified by an 8-bit left rotation. The keyed F-function applies eight 4-bit S-boxes in parallel, after which a permutation is applied to the nibbles. Similar to TWINE, for both encryption and decryption, LBlock can provide the full nibble-wise diffusion after 8 rounds. Hence, we set the length of the middle part of our sandwich distinguishers to 8. LBlock employs 8 different S-boxes for the F-function. We do not need to differentiate between these S-boxes in our truncated MILP model, as their differential and boomerang uniformity is identical. However, as evident from the FBCT of LBlock's Sboxes (Table 7), two difference values $\{3, b\}$ yield a better probability for the boomerang switch. Thus, these differences are a good choice for the active nibbles at the input/output of the boomerang switch.

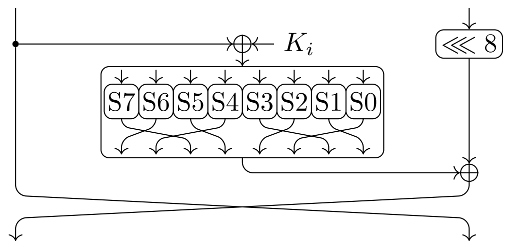

Figure 11: The round function of `LBlock`.

We obtain the following results for `LBlock`. First, we discover a 5-round deterministic sandwich distinguisher. To find longer distinguishers, we set the length of the middle part to 8 rounds, similar to the application on `TWINE`. The resulting distinguishers for 13 to 16 rounds are listed in Table 14. In comparison to the best previous boomerang distinguisher for `LBlock` [CM13], our distinguisher has a higher probability.

We also apply our tool to `LBlock-s`, a simplified version of `LBlock` that uses the S-box S_0 for all nibbles in the F-function. Our distinguishers for 13 to 16 rounds of `LBlock-s` are listed in Table 13. The best previous sandwich distinguisher for `LBlock-s` covers 16 rounds of this cipher with probability $2^{-56.14}$ [BHL+20]. As Table 13 illustrates, the probability of the 16-round distinguisher discovered by our tool is $2^{-53.59}$ (improvement of $2^{2.55}$), whereas the number of active nibbles at its input/output differences is the same as before [BHL+20]. Although Boukerrou et al. [BHL+20] considered the boomerang switch over the 8 rounds, they generated the upper and lower differential trails independently. In contrast, our tool takes the switching effect into account while searching for a sandwich distinguisher and thus yields a better distinguisher.

The diffusion strength of `TWINE` and `LBlock` are almost the same. Furthermore, as Table 9 shows, `TWINE`'s and `LBlock`'s S-boxes have the same differential uniformity. However, according to Table 9, the smaller F-boomerang uniformity of `TWINE`'s S-box results in weaker sandwich distinguishers in comparison to `LBlock`.

8 Conclusion

In this paper, we introduced an improved automatic method to search for boomerang distinguishers by enhancing the method proposed by Hadipour et al. and applied it to several ciphers following the generalized Feistel structure. Thanks to the effectiveness of our method, we managed to improve the best previous results concerning the boomerang analysis on a wide range of GFS ciphers. Notably, we improved the probability of the best previous boomerang distinguishers for 20 and 21 rounds of `WARP` by a factor of $2^{38.28}$ and $2^{36.56}$. In terms of the number of rounds, we also improved the boomerang distinguishers of `WARP` by 2 rounds and managed to distinguish up to 23 rounds of `WARP` from a random permutation. Applying our method to the internationally-standardized cipher `CLEFIA`, we proposed a 9-round boomerang distinguisher for this cipher which improves the best previous boomerang distinguisher by one round. We also built an 11-round key-recovery attack based on this distinguisher. Moreover, we introduced a practical boomerang distinguisher with probability $2^{-32.67}$ for 7 rounds of `CLEFIA` which is, to the best of our knowledge, the first practical distinguisher for 7 rounds of this cipher. We also applied our method to `TWINE`, `LBlock`, and `LBlock-s`. In all cases, we succeeded in improving the best previous boomerang distinguishers of these GFS ciphers.

References

[AK18] Ralph Ankele and Stefan Kölbl. Mind the gap – A closer look at the security of block ciphers against differential cryptanalysis. In *SAC 2018*, volume 11349 of *LNCS*, pages 163–190. Springer, 2018. doi:10.1007/978-3-030-10970-7_8.

[AST+17] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP modeling for (large) S-boxes to optimize probability of differential characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017. doi:10.13154/tosc.v2017.i4.99-129.

[BBI+20] Subhadeep Banik, Zhenzhen Bao, Takanori Isobe, Hiroyasu Kubo, Fukang Liu, Kazuhiko Minematsu, Kosei Sakamoto, Nao Shibata, and Maki Shigeri. WARP: Revisiting GFN for lightweight 128-bit block cipher. In *SAC 2020*, volume 12804 of *LNCS*, pages 535–564. Springer, 2020. doi:10.1007/978-3-030-81652-0_21.

[BDK01] Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack – rectangling the Serpent. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 340–357. Springer, 2001. doi:10.1007/3-540-44987-6_21.

[BGW+13] Andrey Bogdanov, Huizheng Geng, Meiqin Wang, Long Wen, and Baudoin Collard. Zero-correlation linear cryptanalysis with FFT and improved attacks on ISO standards camellia and CLEFIA. In *SAC 2013*, volume 8282 of *LNCS*, pages 306–323. Springer, 2013. doi:10.1007/978-3-662-43414-7_16.

[BHL+20] Hamid Boukerrou, Paul Huynh, Virginie Lallemand, Bimal Mandal, and Marine Minier. On the Feistel counterpart of the boomerang connectivity table introduction and analysis of the FBCT. *IACR Trans. Symmetric Cryptol.*, 2020(1):331–362, 2020. doi:10.13154/tosc.v2020.i1.331-362.

[BHMSV84] Robert K. Brayton, Gary D. Hachtel, Curtis T. McMullen, and Alberto L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*, volume 2 of *The Kluwer International Series in Engineering and Computer Science*. Springer, 1984. doi:10.1007/978-1-4613-2821-6.

[BK09] Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 1–18. Springer, 2009. doi:10.1007/978-3-642-10366-7_1.

[BN19] Alex Biryukov and Ivica Nikolic. Security analysis of the block cipher CLEFIA. Final Report for CRYPTREC, 2019. URL: https://www.cryptrec.go.jp/exreport/cryptrec-ex-2202-2012p2.pdf.

[CHP+18] Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. In *EUROCRYPT 2018*, volume 10821 of *LNCS*, pages 683–714. Springer, 2018. doi:10.1007/978-3-319-78375-8_22.

[CM13] Jiageng Chen and Atsuko Miyaji. Differential cryptanalysis and boomerang cryptanalysis of LBlock. In *CD-ARES Workshops 2013*, volume 8128 of *LNCS*, pages 1–15. Springer, 2013. doi:10.1007/978-3-642-40588-4_1.

[DDV20] Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs – application to SKINNY. *IACR Trans. Symmetric Cryptol.*, 2020(4):104–129, 2020. doi:10.46586/tosc.v2020.i4.104-129.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 393–410. Springer, 2010. doi:10.1007/978-3-642-14623-7_21.

[DKS14] Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. *J. Cryptol.*, 27(4):824–849, 2014. doi:10.1007/s00145-013-9154-9.

[DQSW21] Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. 2021. URL: https://ia.cr/2021/856, doi:10.1007/978-3-031-07082-2_1.

[Gur22] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL: https://www.gurobi.com.

[HBS21] Hosein Hadipour, Nasour Bagheri, and Ling Song. Improved rectangle attacks on SKINNY and CRAFT. *IACR Trans. Symmetric Cryptol.*, 2021(2):140–198, 2021. doi:10.46586/tosc.v2021.i2.140-198.

[KKS00] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round MARS and Serpent. In *FSE 2000*, volume 1978 of *LNCS*, pages 75–93. Springer, 2000. doi:10.1007/3-540-44706-7_6.

[KY21] Manoj Kumar and Tarun Yadav. MILP based differential attack on round reduced WARP. In *SPACE 2021*, volume 13162 of *LNCS*, pages 42–59. Springer, 2021. doi:10.1007/978-3-030-95085-9_3.

[LMR22] Virginie Lallemand, Marine Minier, and Loïc Rouquette. Automatic search of rectangle attacks on feistel ciphers: Application to WARP. *IACR Trans. Symmetric Cryptol.*, 2022(2):113–140, 2022. doi:10.46586/tosc.v2022.i2.113-140.

[LWZ11] Yanjun Li, Wenling Wu, and Lei Zhang. Improved integral attacks on reduced-round CLEFIA block cipher. In *WISA 2011*, volume 7115 of *LNCS*, pages 28–39. Springer, 2011. doi:10.1007/978-3-642-27890-7_3.

[MDS11] Hamid Mala, Mohammad Dakhilalian, and Mohsen Shakiba. Impossible differential attacks on 13-round CLEFIA-128. *J. Comput. Sci. Technol.*, 26(4):744–750, 2011. doi:10.1007/s11390-011-1173-0.

[MQ14] Ming Mao and Zhiguang Qin. Sandwich-boomerang attack on reduced round CLEFIA. *High Technology 1 (2014)*, 20:48–53, 03 2014. doi:10.3772/j.issn.1006-6748.2014.01.008.

[Mur11] Sean Murphy. The return of the cryptographic boomerang. *IEEE Trans. Inf. Theory*, 57(4):2517–2521, 2011. doi:10.1109/TIT.2011.2111091.

[QDW$^+$21] Lingyue Qin, Xiaoyang Dong, Xiaoyun Wang, Keting Jia, and Yunwen Liu. Automated search oriented to key recovery on ciphers with linear key schedule applications to boomerangs in SKINNY and ForkSkinny. *IACR Trans. Symmetric Cryptol.*, 2021(2):249–291, 2021. doi:10.46586/tosc.v2021.i2.249-291.

[Sag22] Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.5.0)*, 2022. URL: https://www.sagemath.org.

[SMMK12] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE: A lightweight block cipher for multiple platforms. In *SAC 2012*, volume 7707 of *LNCS*, pages 339–354. Springer, 2012. doi:10.1007/978-3-642-35999-6_22.

[SQH19] Ling Song, Xianrui Qin, and Lei Hu. Boomerang connectivity table revisited. application to SKINNY and AES. *IACR Trans. Symmetric Cryptol.*, 2019(1):118–141, 2019. doi:10.13154/tosc.v2019.i1.118-141.

[SS04] Taizo Shirai and Kyoji Shibutani. Improving immunity of feistel ciphers against differential cryptanalysis by using multiple MDS matrices. In *FSE 2004*, volume 3017 of *LNCS*, pages 260–278. Springer, 2004. doi:10.1007/978-3-540-25937-4_17.

[SSA$^+$07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In *FSE 2007*, volume 4593 of *LNCS*, pages 181–195. Springer, 2007. doi:10.1007/978-3-540-74619-5_12.

[SV18] Mahdi Sajadieh and Mohammad Vaziri. Using MILP in analysis of feistel structures and improving type II GFS by switching mechanism. In *INDOCRYPT 2018*, volume 11356 of *LNCS*, pages 265–281. Springer, 2018. doi:10.1007/978-3-030-05378-9_15.

[SWW18] Ling Sun, Wei Wang, and Meiqin Wang. More accurate differential properties of LED64 and Midori64. *IACR Trans. Symmetric Cryptol.*, 2018(3):93–123, 2018. doi:10.13154/tosc.v2018.i3.93-123.

[TB21] Je Sen Teh and Alex Biryukov. Differential cryptanalysis of WARP. Cryptology ePrint Archive, Report 2021/1641, 2021. URL: https://ia.cr/2021/1641.

[Tez10] Cihangir Tezcan. The improbable differential attack: Cryptanalysis of reduced round CLEFIA. In *INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 197–209. Springer, 2010. doi:10.1007/978-3-642-17401-8_15.

[TTS$^+$08] Yukiyasu Tsunoo, Etsuko Tsujihara, Maki Shigeri, Teruo Saito, Tomoyasu Suzaki, and Hiroyasu Kubo. Impossible differential cryptanalysis of CLEFIA. In *FSE 2008*, volume 5086 of *LNCS*, pages 398–411. Springer, 2008. doi:10.1007/978-3-540-71039-4_25.

[Wag99] David A. Wagner. The boomerang attack. In *FSE 1999*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999. doi:10.1007/3-540-48519-8_12.

[WP19] Haoyang Wang and Thomas Peyrin. Boomerang switch in multiple rounds. application to AES variants and Deoxys. *IACR Trans. Symmetric Cryptol.*, 2019(1):142–169, 2019. doi:10.13154/tosc.v2019.i1.142-169.

[WZ11] Wenling Wu and Lei Zhang. LBlock: A lightweight block cipher. In *ACNS 2011*, volume 6715 of *LNCS*, pages 327–344, 2011. doi:10.1007/978-3-642-21554-4_19.

[YK21] Tarun Yadav and Manoj Kumar. MILES: Modeling large s-box in MILP based differential characteristic search. Cryptology ePrint Archive, Report 2021/1388, 2021. URL: https://ia.cr/2021/1388.

A Comparison with Lallemand et al.'s Approach [LMR22]

In parallel to this work, Lallemand et al. [LMR22] introduced another method to search for rectangle attacks on Feistel ciphers. Here, we provide a brief comparison between our method and theirs, which were developed independently. Lallemand et al. adapted the method proposed by Delaune et al. [DDV20] for finding boomerang distinguishers to the case of Feistel ciphers. However, we enhanced the method proposed by Hadiour et al. [HBS21]. Thus, most of the differences stem from the differences between these original methods [HBS21, DDV20]. Hadipour et al. [HBS21, Section 8] provide a precise comparison between these methods [DDV20, HBS21]. We first recall the main similarities.

Both approaches have three main steps. They first find suitable upper and lower truncated differential trails. Then, they instantiate the discovered truncated trails with concrete differential trails, and lastly, compute the probability of the three main parts of the discovered sandwich distinguisher. Both approaches can also be extended to construct a unified model for the key recovery of rectangle attacks. For instance, Qin et al. [QDW+21] and Dong et al. [DQSW21] extended the methods proposed in [HBS21] and [DDV20], respectively, to make a unified model for key recovery of rectangle attacks.

However, the two methods follow a different approach to computing the probability of boomerang switch. Our method uses the experimental approach to compute the probability of a boomerang switch, whereas the methods employed in [DDV20] and [LMR22] automatically handle the probability computation of the boomerang switch. To achieve a fully automatic tool, one has to encode different types of (F)BCT tables in boomerang switch at the bit level. However, this makes the resulting models harder to solve, and thus the execution time is longer when the boomerang switch includes many rounds or the S-box size is large enough ($\geq$ 8-bit). Therefore, there is no choice but to sacrifice the accuracy of the probability computation. For example, the difference values over the boomerang switch should be equal at both sides of boomerang distinguishers in [DDV20] and [LMR22]. Lallemand et al. proposed more techniques for speeding up the tool to keep the execution time reasonable. Although these techniques can decrease the execution time, they can also reduce the accuracy of the probability computation. However, we do not consider additional constraints on the difference values over the boomerang switch. After finding suitable truncated upper and lower trails and instantiating them with concrete differential paths, we only fix the difference values at four positions as discussed in Section 3. That is why we achieved better distinguishers for WARP compared to [LMR22]. For instance, we obtained a 23-round sandwich distinguisher for WARP, which has a higher (by a factor of $2^{8.41}$) success probability compared to the one proposed in [LMR22]. Notably, our tool finds this distinguisher in 36 seconds running on a regular laptop (Core(TM) i7-1165G7 @ 2.80GHz).

Comparing the results derived from our approach and the one used in [DDV20, LMR22] reveals that the most important switching effect is the ladder switch which is taken into account in our new truncated models accurately. Therefore, as long as the probability of a boomerang switch is large enough (e.g., larger than 2^{-40}), one can use our method to efficiently find nearly optimal sandwich distinguishers for both SPN and Feistel ciphers.

B Boomerang Properties of S-boxes

Table 6: FBCT of WARP's S-box

$\Delta \setminus \nabla$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
1	16	16	4	4	0	0	0	0	0	0	0	0	0	0	0	0
2	16	4	16	4	4	0	4	0	0	4	0	4	4	0	4	0
3	16	4	4	16	0	0	0	0	0	0	0	0	0	0	0	0
4	16	0	4	0	16	0	4	0	0	0	0	0	0	0	0	0
5	16	0	0	0	0	16	0	0	0	0	8	0	0	0	0	8
6	16	0	4	0	4	0	16	0	0	0	0	0	0	0	0	0
7	16	0	0	0	0	0	0	16	0	0	8	0	0	8	0	0
8	16	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0
9	16	0	4	0	0	0	0	0	0	16	0	4	0	0	0	0
a	16	0	0	0	0	8	0	8	0	0	16	0	0	8	0	8
b	16	0	4	0	0	0	0	0	0	4	0	16	0	0	0	0
c	16	0	4	0	0	0	0	0	0	0	0	0	16	0	4	0
d	16	0	0	0	0	0	0	8	0	0	8	0	0	16	0	0
e	16	0	4	0	0	0	0	0	0	0	0	0	4	0	16	0
f	16	0	0	0	0	8	0	0	0	0	8	0	0	0	0	16

Table 7: FBCT of S-box S_i in LBlock for $0 \leq i \leq 9$

$\Delta \setminus \nabla$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
1	16	16	0	0	0	0	0	0	0	0	8	8	0	0	0	0
2	16	0	16	0	0	0	0	0	0	8	0	8	0	0	0	0
3	16	0	0	16	8	8	8	8	0	0	0	0	0	0	0	0
4	16	0	0	8	16	0	0	8	0	0	0	0	0	0	0	0
5	16	0	0	8	0	16	8	0	0	0	0	0	0	0	0	0
6	16	0	0	8	0	8	16	0	0	0	0	0	0	0	0	0
7	16	0	0	8	8	0	0	16	0	0	0	0	0	0	0	0
8	16	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0
9	16	0	8	0	0	0	0	0	0	16	0	8	0	0	0	0
a	16	8	0	0	0	0	0	0	0	0	16	8	0	0	0	0
b	16	8	8	0	0	0	0	0	0	8	8	16	0	0	0	0
c	16	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0
d	16	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
e	16	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0
f	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16

Table 8: `FBCT` of `TWINE`'s S-box

$\Delta \setminus \nabla$	`0`	`1`	`2`	`3`	`4`	`5`	`6`	`7`	`8`	`9`	`a`	`b`	`c`	`d`	`e`	`f`
`0`	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
`1`	16	16	0	0	0	0	4	4	0	0	0	0	0	0	0	0
`2`	16	0	16	0	0	0	0	0	0	0	0	0	4	0	4	0
`3`	16	0	0	16	0	0	0	0	0	4	4	0	0	0	0	0
`4`	16	0	0	0	16	0	0	0	0	0	0	4	0	0	0	4
`5`	16	0	0	0	0	16	0	0	4	0	0	0	0	4	0	0
`6`	16	4	0	0	0	0	16	4	0	0	0	0	0	0	0	0
`7`	16	4	0	0	0	0	4	16	0	0	0	0	0	0	0	0
`8`	16	0	0	0	0	4	0	0	16	0	0	0	0	4	0	0
`9`	16	0	0	4	0	0	0	0	0	16	4	0	0	0	0	0
`a`	16	0	0	4	0	0	0	0	0	4	16	0	0	0	0	0
`b`	16	0	0	0	4	0	0	0	0	0	0	16	0	0	0	4
`c`	16	0	4	0	0	0	0	0	0	0	0	0	16	0	4	0
`d`	16	0	0	0	0	4	0	0	4	0	0	0	0	16	0	0
`e`	16	0	4	0	0	0	0	0	0	0	0	4	0	0	16	0
`f`	16	0	0	0	4	0	0	0	0	0	0	4	0	0	0	16

Table 9: Differential and boomerang properties of S-boxes in our applications

Cipher	Size (bits)	Diff. uniformity	Boom. uniformity	F-Boom. uniformity
`WARP`	4	4	16	**8**
`LBlock`	4	4	16	**8**
`TWINE`	4	4	6	**4**
`CLEFIA` S_0	8	10	32	**20**
`CLEFIA` S_1	8	4	6	**4**

C Reformulation of Equation 2

Equation 3 reformulates Equation 2 by dividing it into precomputed tables T_1, T_2, T_3, T_4:

$$T_1(\alpha_6^{(3)}, \beta_4^{(12)}) = \sum_{\beta_6^{(3)}} \texttt{FBCT}(\alpha_6^{(3)}, \beta_6^{(3)}) \cdot \texttt{DDT}(\beta_4^{(12)}, \beta_6^{(3)}),$$

$$T_2(\alpha_6^{(3)}, \alpha_{20}^{(6)}, \beta_{28}^{(7)}) = \sum_{\alpha_{28}^{(7)}} \texttt{FBCT}(\alpha_{28}^{(7)}, \beta_{28}^{(7)}) \cdot \texttt{DDT}(\alpha_6^{(3)}, \alpha_{20}^{(6)}) \cdot \texttt{DDT}(\alpha_{20}^{(6)}, \alpha_{28}^{(7)}),$$

$$T_3(\beta_{19}^{(10)}, \beta_{28}^{(7)}) = \sum_{\beta_8^{(8)}} \texttt{DDT}(\beta_{10}^{(19)}, \beta_8^{(8)}) \cdot \texttt{DDT}(\beta_8^{(8)}, \beta_{28}^{(7)}),$$

$$T_4(\alpha_{20}^{(6)}, \beta_{19}^{(10)}, \beta_4^{(12)}) = \sum_{\alpha_{24}^{(9)}} \sum_{\alpha_{18}^{(10)}} \texttt{DDT}(\alpha_{20}^{(6)}, \alpha_{24}^{(9)}) \cdot \texttt{DDT}(\alpha_{24}^{(9)}, \alpha_{18}^{(10)}) \cdot \texttt{FBDT}(\beta_4^{(12)}, \beta_{19}^{(10)}, \alpha_{18}^{(10)}).$$

$$\begin{aligned} r(\alpha_6^{(3)}, \beta_4^{(12)}) = 2^{-10\times 4} \cdot \sum & \texttt{FBCT}(\alpha_6^{(3)}, \beta_6^{(3)}) \cdot \texttt{DDT}(\beta_4^{(12)}, \beta_6^{(3)}) \cdot \texttt{FBCT}(\alpha_{28}^{(7)}, \beta_{28}^{(7)}) \\ & \cdot \texttt{DDT}(\alpha_6^{(3)}, \alpha_{20}^{(6)}) \cdot \texttt{DDT}(\alpha_{20}^{(6)}, \alpha_{28}^{(7)}) \cdot \texttt{DDT}(\beta_{19}^{(10)}, \beta_8^{(8)}) \\ & \cdot \texttt{DDT}(\beta_8^{(8)}, \beta_{28}^{(7)}) \cdot \texttt{FBDT}(\beta_4^{(12)}, \beta_{19}^{(10)}, \alpha_{18}^{(10)}) \\ & \cdot \texttt{DDT}(\alpha_{20}^{(6)}, \alpha_{24}^{(9)}) \cdot \texttt{DDT}(\alpha_{24}^{(9)}, \alpha_{18}^{(10)}) \\ = 2^{-10\times 4} \cdot \sum_{\alpha_{20}^{(6)}, \beta_{28}^{(7)}, \beta_{19}^{(10)} \in \mathbb{F}_2^4} & T_1(\alpha_6^{(3)}, \beta_4^{(12)}) \cdot T_2(\alpha_6^{(3)}, \alpha_{20}^{(6)}, \beta_{28}^{(7)}) \cdot T_3(\beta_{19}^{(10)}, \beta_{28}^{(7)}) \cdot T_4(\alpha_{20}^{(6)}, \beta_{19}^{(10)}, \beta_4^{(12)}). \end{aligned} \tag{3}$$

D Distinguisher for WARP

Table 10: Specification of Sandwich Distinguishers for WARP

15 Rounds	
$r_0 = 2, r_m = 10, r_1 = 3,\ p = 2^{-4},\ q = 2^{-8},\ r = 2^{-4.58},\ p^2q^2r = 2^{-28.58}$	
$\Delta X^{(0)}$	$\Delta X^{(2)}$
0000000000000a00000000a500000000	0a000000000000000000000000000000
$\nabla X^{(12)}$	$\nabla X^{(15)}$
0000a000000000000000000000000000	0a000000f000a00000500a0000000000
16 Rounds	
$r_0 = 3, r_m = 10, r_1 = 3,\ p = 2^{-8},\ q = 2^{-4},\ r = 2^{-10.50},\ p^2q^2r = 2^{-34.50}$	
$\Delta X^{(0)}$	$\Delta X^{(3)}$
0000000000000a00000000aaaa000000	0000000000000000000000000000000a
$\nabla X^{(13)}$	$\nabla X^{(16)}$
00000000000000000a00000000000000	00000000000aa00000a0000000000000
20 Rounds	
$r_0 = 5, r_m = 10, r_1 = 5,\ p = 2^{-14},\ q = 2^{-14},\ r = 2^{-19.96},\ p^2q^2r = 2^{-75.96}$	
$\Delta X^{(0)}$	$\Delta X^{(5)}$
00aa5a00000a000000000a00000000da	000000000000000000000000000000a0
$\nabla X^{(15)}$	$\nabla X^{(20)}$
000000000000000000000000000a0000	70007050000005a0000000000500007a
21 Rounds	
$r_0 = 5, r_m = 10, r_1 = 6,\ p = 2^{-10},\ q = 2^{-19},\ r = 2^{-26.55},\ p^2q^2r = 2^{-84.55}$	
$\Delta X^{(0)}$	$\Delta X^{(5)}$
00aa00000000000000000a00000000aa	000000000000000a000000a000000000
$\nabla X^{(15)}$	$\nabla X^{(21)}$
0000000a000000000000000000000000	f00a00a00a00a070000050a000000005
22 Rounds	
$r_0 = 6, r_m = 10, r_1 = 6,\ p = 2^{-16},\ q = 2^{-19},\ r = 2^{-26.55},\ p^2q^2r = 2^{-96.55}$	
$\Delta X^{(0)}$	$\Delta X^{(6)}$
5a0000aa05a50000000a000000000000	000000000000000a000000a000000000
$\nabla X^{(16)}$	$\nabla X^{(22)}$
0000000a000000000000000000000000	f00a00a00a00a070000050a000000005
23 Rounds	
$r_0 = 6, r_m = 10, r_1 = 7,\ p = 2^{-24},\ q = 2^{-20},\ r = 2^{-27.59},\ p^2q^2r = 2^{-115.59}$	
$\Delta X^{(0)}$	$\Delta X^{(6)}$
00000000aaaaaaaa0a00aa00000a000a	a0000000000000000000000000000000
$\nabla X^{(16)}$	$\nabla X^{(23)}$
00000000000000aa0000000000000000	0000a0a00000000aa00a00a00a00a0a0

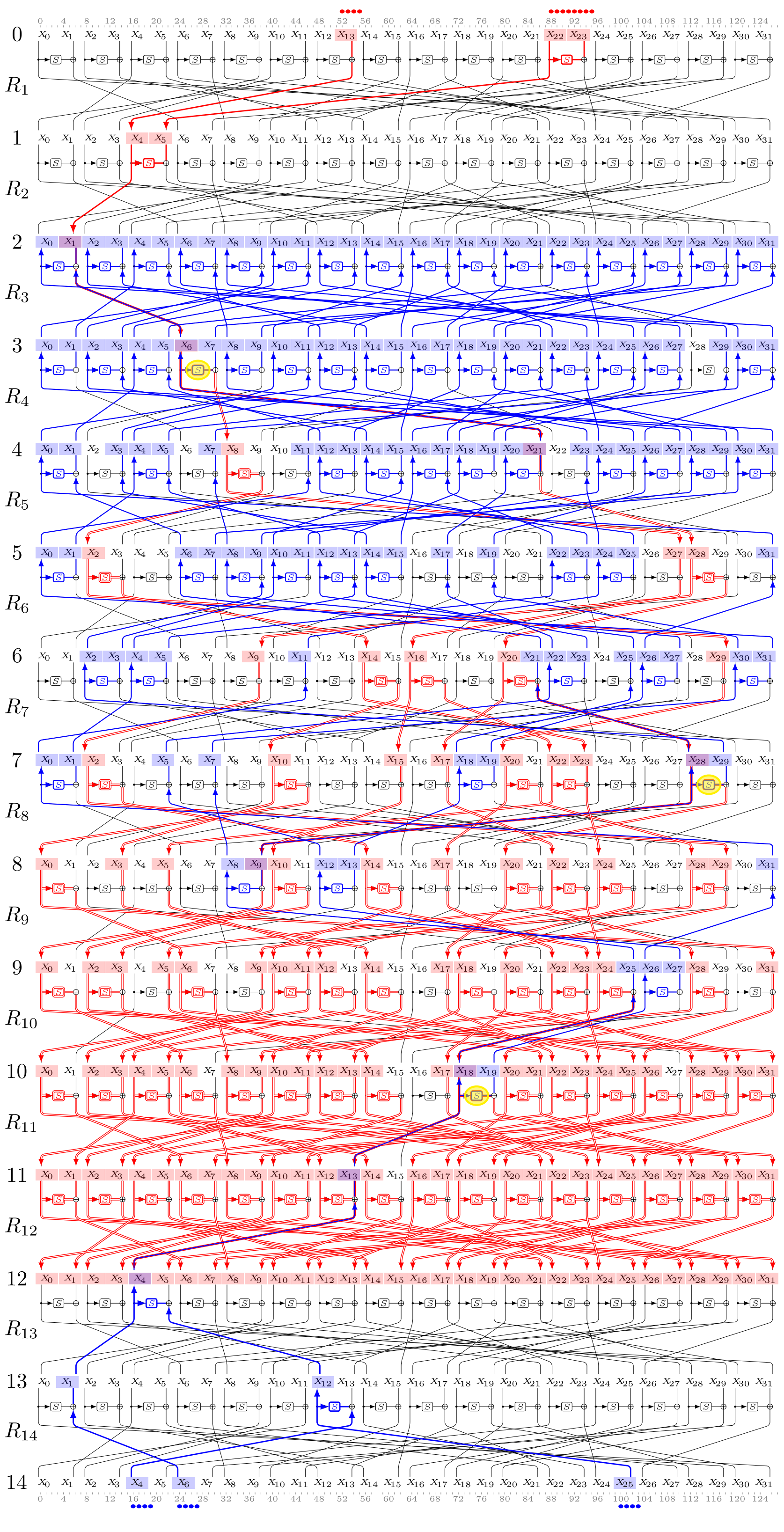

Figure 12: Sandwich distinguisher for 14 rounds of `WARP`.

E Distinguisher for `CLEFIA`

Table 11: Sandwich distinguishers for 4 to 8 rounds of `CLEFIA` (with final permutation)

4 Rounds			
$r_0 = 0, r_m = 4, r_1 = 0,\ p = 1,\ q = 1,\ r = 2^{-6.46},\ p^2q^2r = 2^{-6.46}$			
$\Delta X^{(0)}$	`00000000000000000000000000000100`	$\nabla X^{(4)}$	`00000001000000000000000000000000`
5 Rounds			
$r_0 = 0, r_m = 5, r_1 = 0,\ p = 1,\ q = 1,\ r = 2^{-12.26},\ p^2q^2r = 2^{-12.26}$			
$\Delta X^{(0)}$	`000000000000000000000000000000e2`	$\nabla X^{(5)}$	`00e20000000000000000000000000000`
6 Rounds			
$r_0 = 0, r_m = 5, r_1 = 1,\ p = 1,\ q = 2^{-4.68},\ r = 2^{-13.09},\ p^2q^2r = 2^{-22.45}$			
$\Delta X^{(0)}$	`00000000e20000000000000000000000`		-
$\nabla X^{(5)}$	`000000000000000000e2000000000000`	$\nabla X^{(6)}$	`0000000000e20000680d721a00000000`
7 Rounds			
$r_0 = 1, r_m = 5, r_1 = 1,\ p = 2^{-4.68},\ q = 2^{-4.68},\ r = 2^{-13.95},\ p^2q^2r = 2^{-32.67}$			
$\Delta X^{(0)}$	`000000000000000003200001d205d40`	$\Delta X^{(1)}$	`00000000003200000000000000000000`
$\nabla X^{(6)}$	`00000000000000000320000000000000`	$\nabla X^{(7)}$	`00000000003200001d205d4000000000`
8 Rounds			
$r_0 = 2, r_m = 5, r_1 = 1,\ p = 2^{-26.36},\ q = 2^{-4.68},\ r = 2^{-13.95},\ p^2q^2r = 2^{-76.03}$			
$\Delta X^{(0)}$	`2bfcd77e9d96be910000000000000008`	$\Delta X^{(2)}$	`00000000000000080000000000000000`
$\nabla X^{(7)}$	`00000000000000000000003200000000`	$\nabla X^{(8)}$	`00000000000000325d401d2000000000`

F Distinguisher for `TWINE`

Table 12: Specification of Sandwich Distinguishers for `TWINE`

13 Rounds			
$r_0 = 3, r_m = 8, r_1 = 2,\ p = 2^{-4},\ q = 2^{-4},\ r = 2^{-18.32},\ p^2q^2r = 2^{-34.32}$			
ΔX_0	`0000060052000000`	ΔX_3	`0000000000000060`
∇X_{11}	`6000000000000000`	∇X_{13}	`2000050000006000`
14 Rounds			
$r_0 = 3, r_m = 8, r_1 = 3,\ p = 2^{-4},\ q = 2^{-8},\ r = 2^{-18.25},\ p^2q^2r = 2^{-42.25}$			
ΔX_0	`0652000000000000`	ΔX_3	`0000000000006000`
∇X_{11}	`0060000000000000`	∇X_{14}	`5000060050a00002`
15 Rounds			
$r_0 = 4, r_m = 8, r_1 = 3,\ p = 2^{-8},\ q = 2^{-8},\ r = 2^{-19.03},\ p^2q^2r = 2^{-51.03}$			
ΔX_0	`052a006500000000`	ΔX_4	`0000000060000000`
∇X_{12}	`6000000000000000`	∇X_{15}	`a000020000505006`
16 Rounds			
$r_0 = 4, r_m = 8, r_1 = 4,\ p = 2^{-8},\ q = 2^{-8},\ r = 2^{-26.04},\ p^2q^2r = 2^{-58.04}$			
ΔX_0	`052a006500000000`	ΔX_4	`0000000060000000`
∇X_{12}	`0500000000000000`	∇X_{16}	`70000a0000202005`

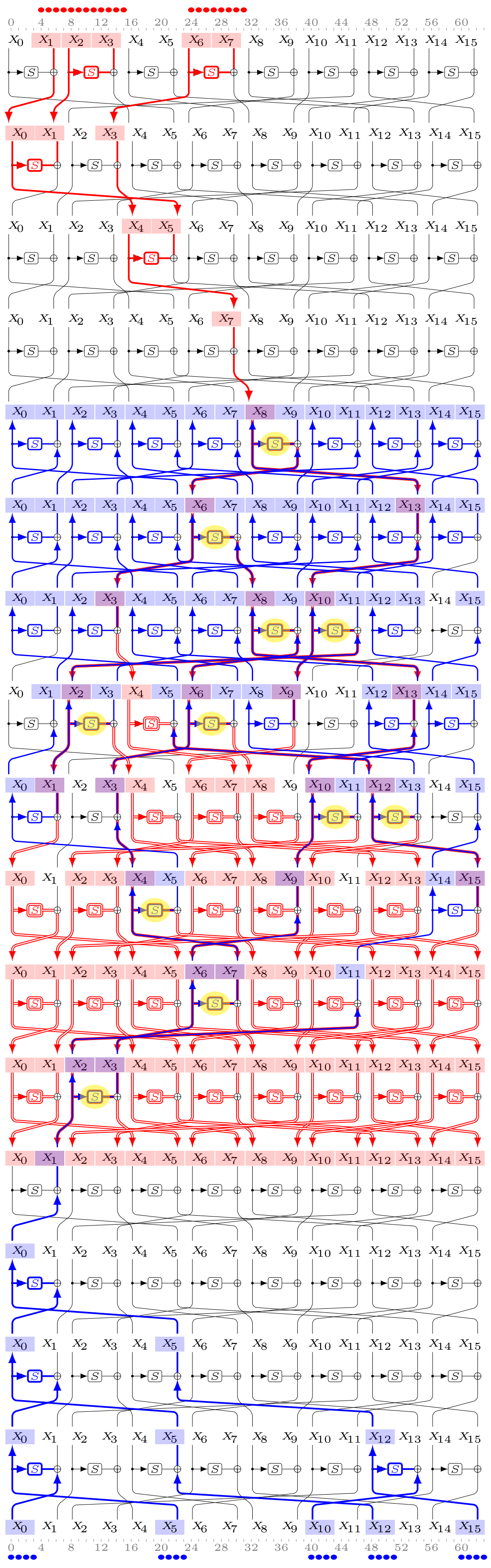

Figure 13: Sandwich distinguisher for 16 rounds of `TWINE`

G Distinguishers for `LBlock` and `LBlock-s`

Table 13: Specification of Sandwich Distinguishers for `LBlock-s`

13 Rounds			
$r_0 = 3, r_m = 8, r_1 = 2,\ p = 2^{-4},\ q = 2^{-4},\ r = 2^{-14.23},\ p^2q^2r = 2^{-30.23}$			
ΔX_0	`0001000000010b00`	ΔX_3	`0b00000000000000`
∇X_{11}	`00000b0000000000`	∇X_{13}	`000b009000001000`
14 Rounds			
$r_0 = 3, r_m = 8, r_1 = 3,\ p = 2^{-8},\ q = 2^{-4},\ r = 2^{-14.47},\ p^2q^2r = 2^{-38.47}$			
ΔX_0	`3020000000062100`	ΔX_3	`0000000000000030`
∇X_{11}	`0000000000300000`	∇X_{14}	`0001003000200000`
15 Rounds			
$r_0 = 4, r_m = 8, r_1 = 3,\ p = 2^{-8},\ q = 2^{-8},\ r = 2^{-14.49},\ p^2q^2r = 2^{-46.49}$			
ΔX_0	`0001030000010220`	ΔX_4	`0300000000000000`
∇X_{12}	`0000030000000000`	∇X_{15}	`0260000100030020`
16 Rounds			
$r_0 = 4, r_m = 8, r_1 = 4,\ p = 2^{-8},\ q = 2^{-8},\ r = 2^{-21.59},\ p^2q^2r = 2^{-53.59}$			
ΔX_0	`0400000404400006`	ΔX_4	`0000040000000000`
∇X_{12}	`0000000040000000`	∇X_{16}	`0000444000004400`

Table 14: Specification of Sandwich Distinguishers for `LBlock`

13 Rounds			
$r_0 = 3, r_m = 8, r_1 = 2,\ p = 2^{-4},\ q = 2^{-4},\ r = 2^{-14.28},\ p^2q^2r = 2^{-30.28}$			
ΔX_0	`00070000000e0b00`	ΔX_3	`0b00000000000000`
∇X_{11}	`00000b0000000000`	∇X_{13}	`000b001000002000`
14 Rounds			
$r_0 = 3, r_m = 8, r_1 = 3,\ p = 2^{-4},\ q = 2^{-8},\ r = 2^{-14.86},\ p^2q^2r = 2^{-38.86}$			
ΔX_0	`0006000000080900`	ΔX_3	`0900000000000000`
∇X_{11}	`0000010000000000`	∇X_{14}	`0160000500010040`
15 Rounds			
$r_0 = 4, r_m = 8, r_1 = 3,\ p = 2^{-8},\ q = 2^{-8},\ r = 2^{-14.90},\ p^2q^2r = 2^{-46.90}$			
ΔX_0	`0001030000010a80`	ΔX_4	`0300000000000000`
∇X_{12}	`0000030000000000`	∇X_{15}	`0890000a00030020`
16 Rounds			
$r_0 = 4, r_m = 8, r_1 = 4,\ p = 2^{-8},\ q = 2^{-8},\ r = 2^{-25.18},\ p^2q^2r = 2^{-57.16}$			
ΔX_0	`0000b0402500000b`	ΔX_4	`b000000000000000`
∇X_{12}	`0000000000000001`	∇X_{16}	`0120000100010030`

H Our Tool for Encoding the DDT and LAT of S-boxes

Our tool to extract and simplify the MILP and SMT/SAT constraints encoding the differential/linear behavior of S-boxes is publicly available in the following Github repository:

https://github.com/hadipourh/sboxanalyzer

The following example represents how to use our tool within the SageMath to extract and simplify the MILP and SAT constraints encoding the DDT of `LBlock`'s S-box.

```
from sboxanalyzer import *
from sage.crypto.sboxes import LBlock_0 as sb
sa = SboxAnalyzer(sb)
cnf, milp = sa.minimized_diff_constraints(mode=2)

Number of constraints: 35
Input:  a0||a1||a2||a3; a0: msb
Output: b0||b1||b2||b3; b0: msb
Weight: 3.0000 p0 + 2.0000 p1
pretty_print(milp)
['- p0 - p1 >= -1',
 '- a0 - b2 + p0 >= -1',
 '- a1 - b3 + p0 >= -1',
 '- b2 - b3 + p0 >= -1',
 'a0 + a1 - b1 + b2 >= 0',
 'a1 + b0 - b1 + b2 >= 0',
 'a2 + a3 + b1 - p1 >= 0',
 'a0 - a1 + b3 + p1 >= 0',
 'a1 - a3 + b3 + p1 >= 0',
 '- a0 + b0 + b2 + b3 >= 0',
 '- a1 - b0 + b2 + p0 >= -1',
 'a0 + a1 - a2 + a3 + b3 >= 0',
 'a0 + a1 + a2 - b2 + b3 >= 0',
 'a1 + a3 - b0 + b2 + b3 >= 0',
 'a0 + a1 + a2 + a3 - p0 >= 0',
 'a0 + b1 + b2 + b3 - p1 >= 0',
 'a0 + b1 + b2 - b3 + p1 >= 0',
 'a0 + a1 - a2 - a3 - b3 >= -2',
 '- a0 + a2 + b0 + b1 + b2 >= 0',
 '- a1 + a3 - b0 - b2 + b3 >= -2',
 '- a1 - a3 + b0 - b2 + b3 >= -2',
 '- a0 + a1 - a2 - b3 + p1 >= -2',
 'a0 - a1 - a2 - a3 + b1 + b3 >= -2',
 'a0 + a2 - a3 - b1 - b2 + b3 >= -2',
 'a0 - a1 + a2 - a3 + b1 + p1 >= -1',
 'a0 - a1 - a2 + a3 + b1 + p1 >= -1',
 'a0 - a2 - a3 - b1 - b2 + p1 >= -3',
 'a0 + a2 + a3 - b1 - b2 + p1 >= -1',
 '- a0 + a2 - b0 - b1 + b2 - b3 >= -3',
 '- a0 + a1 - b0 + b1 + b2 - b3 >= -2',
 '- a0 - a2 - b0 - b1 - b3 + p1 >= -4',
 '- a0 - a2 + b0 + b1 - b3 + p1 >= -2',
 'a0 - a1 - a2 + b0 - b1 - b2 + b3 >= -3',
 '- a0 - a1 + a2 + b0 - b1 - b2 - b3 >= -4',
 '- a0 - a1 + a2 - b0 + b1 - b2 - b3 >= -4']
```

```
cnf, milp = sa.minimized_diff_constraints(mode=5, cryptosmt_compatible=True)

Number of constraints: 37
Input:  a0||a1||a2||a3; a0: msb
Output: b0||b1||b2||b3; b0: msb
Weight: p0 + p1 + p2
```