Probabilistic Extensions: A One-Step Framework for Finding Rectangle Attacks and Beyond

Ling Song[2,3], Qianqian Yang[1,5]✉, Yincen Chen[2,3], Lei Hu[1,5], and Jian Weng[2,3,4]

[1] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] College of Cyber Security, Jinan University, Guangzhou, China
[3] National Joint Engineering Research Center of Network Security Detection and Protection Technology, Jinan University, Guangzhou, China
[4] Guangdong Key Laboratory of Data Security and Privacy Preserving, Jinan University, Guangzhou, China
[5] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
songling.qs@gmail.com,yangqianqian@iie.ac.cn,icsnow98@gmail.com, hulei@iie.ac.cn, cryptjweng@gmail.com

Abstract. In differential-like attacks, the process typically involves extending a distinguisher forward and backward with probability 1 for some rounds and recovering the key involved in the extended part. Particularly in rectangle attacks, a holistic key recovery strategy can be employed to yield the most efficient attacks tailored to a given distinguisher. In this paper, we treat the distinguisher and the extended part as an integrated entity and give a one-step framework for finding rectangle attacks with the purpose of reducing the overall complexity or attacking more rounds. In this framework, we propose to allow probabilistic differential propagations in the extended part and incorporate the holistic recovery strategy. Additionally, we introduce the "split-and-bunch technique" to further reduce the time complexity. Beyond rectangle attacks, we extend these foundational concepts to encompass differential attacks as well. To demonstrate the efficiency of our framework, we apply it to `Deoxys-BC-384`, `SKINNY`, `ForkSkinny`, and `CRAFT`, achieving a series of refined and improved rectangle attacks and differential attacks. Notably, we obtain the first 15-round attack on `Deoxys-BC-384`, narrowing its security margin to only one round. Furthermore, our differential attack on `CRAFT` extends to 23 rounds, covering two more rounds than the previous best attacks.

Keywords: Rectangle attack · Differential attack · Key recovery attack · `Deoxys-BC` · `SKINNY` · `ForkSkinny` · `CRAFT`

1 Introduction

Differential cryptanalysis, proposed by Biham and Shamir [BS91] in 1991, is one of the most efficient and powerful cryptanalysis for symmetric ciphers. In

differential cryptanalysis, the adversary aims to discover the in-homogeneity in high-probability occurrences of plaintext and ciphertext differences, *i.e.*, high-probability differentials. For a certain block cipher, if an r-round high-probability differential is found, one could add some outer rounds and extract the information of the key in the outer rounds using the differential.

However, it is a challenging task to find a long differential trail with high probability for most ciphers. To this, Wanger proposed the boomerang attack [Wag99] in 1999 as an extension of differential cryptanalysis. In a boomerang attack, two short differential trails are combined to form a long one. The basic boomerang attack requires adaptive chosen plaintexts and ciphertexts. In [KKS00], Kelsey *et al.* converted it into a chosen-plaintext variant, which was named the amplified boomerang attack. Later, Biham *et al.* [BDK01] made further improvements on the amplified boomerang attack by proposing the rectangle attack, which takes into account as many differences as possible in the middle of the distinguisher to estimate the probability more accurately. Since then, the boomerang and rectangle attacks have been extensively studied and applied to many block ciphers. For example, boomerang attacks on full `AES-192` and `AES-256` in the related-key setting were proposed in [BK09, DEFN22]; the best cryptanalysis results so far on `Deoxys` [BL23], `SKINNY` [SZY+22], and `GIFT` [DQSW22] were all based on boomerang attacks or rectangle attacks.

In either differential attacks or rectangle attacks, it is common to take two steps to mount key recovery attacks. The first step is to find a high-probability distinguisher covering a large number of rounds. To this end, many approaches have been proposed in the literature [Mat94, MP13, SHW+14, HBS21, DDV21]. Once a high-probability distinguisher is obtained, then in the second step, the distinguisher is extended backward and forward over some rounds with probability 1 for the key recovery attack. In recent years, many studies focused on achieving key recovery attacks as efficiently as possible for a given distinguisher. For differential attacks, Boura *et al.* [BDD+23] introduced a novel method that recovers the key in a meet-in-the-middle manner, which brought out improved results on block ciphers `SKINNY-128-384` and `AES-256`. For rectangle attacks, various algorithms have been proposed for the key recovery [BDK01, BDK02, ZDJ19, DQSW21, SZY+22], each of which follows a different strategy for guessing the key. Among them, the unified key recovery algorithm proposed in [SZY+22] supports any strategy for guessing the key and covers all the previous algorithms. Most notably, this algorithm is able to produce the most efficient key recovery attack for a given distinguisher. As a result, it led to the state-of-the-art results of rectangle key recovery attack on `Serpent` [ABK98], `CRAFT` [BLMR19], `SKINNY` [BJK+16] and `Deoxys` [JNPS16].

However, prior research has demonstrated that the best distinguisher does not necessarily yield the most effective key recovery attack [LGS17, ZDC+21, QDW+21]. This phenomenon is even not limited to differential attacks and rectangle attacks [SSD+18, HSE23]. These studies have revealed that distinguishers are more conducive to key recovery attacks when their input and output differences exhibit sparsity and slow diffusion in the extended rounds. In [ZDC+21], the

authors specifically sought differentials that were more favorable for key recovery, while [QDW+21] incorporated both the rectangle distinguisher and the extended part in their modeling to find better attacks.

Motivation. Previous research offers valuable insights that inform our work.

- The effectiveness of a key recovery attack hinges not only on the probability of the distinguisher but also on the differential propagation in the outer rounds. Slower diffusion of differences in the outer rounds tends to yield better results.
- The key recovery is likely to be more efficient when the outer rounds involve fewer key bits.
- The choice of key guessing strategy significantly influences the complexity of the key recovery attack.

Despite the key recovery algorithm in [SZY+22] offering the best rectangle attacks for a given distinguisher, we are prompted to explore whether there is room for improving rectangle attacks further when we treat the distinguisher and the extended part as a whole and meanwhile incorporate the lessons mentioned above. Additionally, we are curious whether similar advancements can be applied to differential attacks.

Our contributions. In this work, we treat the distinguisher and the extended part as a whole and present a one-step framework for finding rectangle attacks. Instead of extending the distinguisher forward and backward with probability 1, we introduce probabilistic differential propagations in the extended part. Traditionally, the extended part only contained deterministic truncated differentials, but now it can include probabilistic (truncated) differential propagations as well. Since the probability of the extended part may be smaller than 1, we refer to the part relevant to the key recovery phase as the *outer part* and the remaining part as the *inner part*.

The effects of probabilistic extensions are multifaceted

- Probabilistic extensions help make differences in the outer part sparse, which potentially reduces the time complexity or covers more rounds.
- The boundaries separating the inner part and the outer part are no longer predefined and not necessarily well-aligned. Instead, they are dynamically determined in conjunction with the probabilistic extensions
- The data complexity is then determined by the overall probability.

To further optimize the time complexity, we propose the *split-and-bunch technique*. When probabilistic extensions are used, differences in the outer part become sparse and certain key bits may form intricate connections with the inner part. In cases where we can identify a compressed representation of these connections, we can isolate and replace these key bits with this compressed version. As a result, the actual number of involved key bits decreases, leading to reduced time complexity.

Table 1: Summary of the cryptanalytic results. Rect./ D/ ZC/ ID=rectangle, differential, zero-correlation, impossible differential. RTK/ SK/ ST/ WK/ WT=related-tweakey, single key, single tweak, weak-key, weak-tweak.

Cipher	Rounds	Data	Memory	Time	Approach	Setting	Ref.
Deoxys-BC-384	14	$2^{125.2}$	2^{140}	2^{260}	Rect.	RTK	[DQSW22]
	14	$2^{115.7}$	2^{160}	$2^{260.59}$	Rect.	RTK	Sect. 4.1
	14	$2^{115.7}$	2^{128}	$2^{242.7}$	Rect.	RTK	Sect. 4.1
	15	$2^{115.7}$	2^{128}	$2^{371.7}$	Rect.	RTK	Sect. 4.1
SKINNY-128-256	26	$2^{126.53}$	$2^{128.44}$	$2^{254.4}$	Rect.	RTK	[DQSW22]
	26	$2^{126.53}$	2^{136}	$2^{241.38}$	Rect.	RTK	[SZY+22]
	26	$2^{121.93}$	2^{136}	$2^{219.93}$	Rect.	RTK	Sect. 4.2
ForkSkinny -128-256	28	$2^{118.88}$	$2^{118.88}$	$2^{224.76}$	Rect.	RTK	[DQSW22]
	28	$2^{123.89}$	$2^{123.89}$	$2^{212.89}$	Rect.	RTK	Sect. D
CRAFT	23	2^{74}	2^{51}	2^{94}	D	WK&ST	[LR22]
	26	2^{73}	2^{60}	2^{105}	D	WK&WT	[LR22]
	20	$2^{62.89}$	2^{49}	$2^{120.43}$	ZC	SK&ST	[HSE23]
	21	$2^{60.99}$	2^{100}	$2^{106.53}$	ID	SK&ST	[HSE23]
	19	$2^{60.99}$	2^{68}	$2^{94.59}$	D	SK&WT	[GSS+20]
	21	$2^{60.99}$	2^{92}	$2^{87.60}$	D	SK&WT	Sect. 4.3
	23	$2^{60.99}$	2^{120}	$2^{111.46}$	D	SK&WT	Sect. 4.3

Moreover, our framework incorporates the holistic key recovery strategy, *i.e.*, the method of finding the best key recovery attack by choosing a proper key guessing strategy.

The main ideas of this framework are not limited to rectangle attacks but can also be applied to differential attacks. We apply the framework to several block ciphers and obtain the following results, which are summarized in Table 1.

- We provide improved 14-round rectangle attacks on Deoxys-BC-384 and introduce the first attack on 15 rounds, reducing the security margin to just one round.
- The data and time complexities of the 26-round rectangle attack on SKINNY-128-256 are reduced. Similar improvements can be made on the 28-round attack on ForkSkinny-128-256.
- We propose differential attacks on CRAFT with up to 23 rounds in the single-key setting, which is two rounds more than the previous best attacks.

Organization. The rest of the paper is organized as follows. In Section 2, we review differential attacks, rectangle attacks, and the key recovery algorithms. In Section 3, probabilistic extensions will be introduced via examples, and the one-step framework will be given. In Section 4, we apply our framework to several block ciphers. We conclude this paper in Section 5.

2 Preliminaries

2.1 Differential Attacks

Differential cryptanalysis [BS91] is a technique used to analyze the propagation of difference through a cipher $E : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Typically, an attacker aims to find a differential $(\alpha, \delta) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ such that the probability $Pr(\alpha \to \delta) = Pr\big[E(x) \oplus E(x \oplus \alpha) = \delta\big]$ is high. Since E is a permutation, it follows that $Pr(\alpha \to \delta) = Pr(\alpha \leftarrow \delta)$. A truncated differential [Knu95] is characterized by a set of input differences $\mathcal{D}_i$ and a set of output differences $\mathcal{D}_o$ and the probability $Pr(\mathcal{D}_i \to \mathcal{D}_o)$ is defined as $\text{Avg}_{\alpha \in \mathcal{D}_i} Pr\big[E(x) \oplus E(x \oplus \alpha) \in \mathcal{D}_o\big]$. Note that $Pr(\mathcal{D}_i \to \mathcal{D}_o)$ and $Pr(\mathcal{D}_i \leftarrow \mathcal{D}_o)$ are usually not equal. Typical sets for truncated differentials often involve patterns where some bits can take on all possible values, while others are constrained to be zero.

In differential-like attacks, a distinguisher is extended backward and forward with probability 1 over some rounds. The goal of these attacks is to recover the key used in the extended rounds. Essentially, the differential in the extended rounds is actually truncated differentials with probability 1. These types of attacks are designed to understand how differences propagate with a high degree of certainty, allowing for the identification of key bits used in those rounds.

Key Recovery Attacks. Suppose a differential $\alpha \to \delta$ over E_d is of high probability P. Suppose E_b and E_f are added around E_d, as shown in Fig. 1.

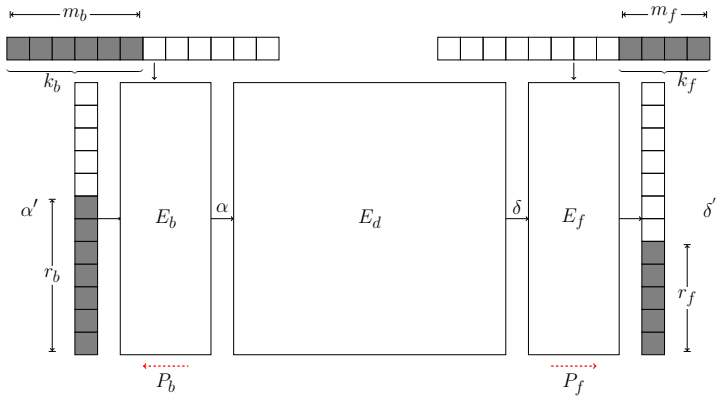

Fig. 1: Outline of the key recovery attack

The input difference of E_d α propagates back over E_b^{-1} with probability $P_b = 1$ to α'. Let V_b be the space spanned by all possible α' where $r_b = \log_2 |V_b|$. The output difference δ of E_d propagates forward over E_f with probability $P_f = 1$ to δ'. Let V_f be the space spanned by all possible δ' where $r_f = \log_2 |V_f|$. Let k_b be the subset of subkey bits that are employed in E_b and affect the propagation $\alpha' \to \alpha$. Let k_f be the subset of subkey bits which are employed in E_f and affect the propagation $\delta \leftarrow \delta'$. Let $m_b = |k_b|$ and $m_f = |k_f|$ be the number of the bits in k_b and k_f.

In a key recovery attack, some key bits may be guessed in advance to sieve the data faster. Suppose a part of k_b and k_f, denoted by k'_b, k'_f, is guessed at first. Let $m'_b = |k'_b|$ and $m'_f = |k'_f|$ and $0 \le m'_b \le m_b, 0 \le m'_f \le m_f$. Suppose under the guessed subkey bits a r'_b-bit condition on the top and a r'_f-bit condition on the bottom can be verified. Finally, let $r^*_b = r_b - r'_b$ and $r^*_f = r_f - r'_f$. Note that the other parameters are determined when k'_b, k'_f are chosen.

In [SYL23], a unified key recovery algorithm for differential attack was proposed and we recall it in Appendix A.1. With k'_b, k'_f being guessed at first, the complexities of the differential attack are as follows.

Complexities. A plaintext structure takes all possible values for the r_b bits and chooses a constant for the remaining $n-r_b$ bits. For one structure, there are 2^{2r_b-1} pairs of plaintext and 2^{r_b-1} of them satisfy α difference by meeting the r_b-bit condition. Suppose the number of the structures needed is y which y structures can constitute $y \cdot 2^{r_b-1}$ pairs that satisfy α difference. Set s to be the number of right pairs, then $y \cdot 2^{r_b-1} = s \cdot P^{-1}$ and the data complexity $D = y \cdot 2^{r_b} = 2s \cdot P^{-1}$. The memory complexity is $M = \max\{D, 2^{t+m_b+m_f-m'_b-m'_f}\}$ for storing the data and key counters where $0 \le t \le m'_b + m'_f$.

The time complexity of the differential attack contains four parts:

- $T_0 = 2^{m'_b+m'_f} \times D$ for partial encryption and decryption under the guessed key bits;
- $T_1 = 2^{m'_b+m'_f} \times D \times 2^{r_b-1+r_f-n-r'_b-r'_f}$ for getting the pairs that satisfy some filtering conditions;
- $T_2 = D \cdot 2^{m_b+m_f-n-1} \cdot \epsilon, \epsilon \ge 1$ for extracting all the $D \cdot 2^{m_b+m_f-n-1}$ key candidates where ϵ depends on the concrete situation;
- $T_3 = 2^{k-h}$, where $h \le t + m_b + m_f - m'_b - m'_f$ for the exhaustive search.

2.2 Rectangle Attacks

In a boomerang attack, the target cipher is treated as a composition of two sub-ciphers E_0 and E_1, *i.e.*, $E = E_1 \circ E_0$. As illustrated in Fig. 2, the differential trail $\alpha \to \beta$ travels in E_0 with probability p, and the differential trail $\gamma \to \delta$ travels in E_1 with probability q, respectively. Then the probability of the boomerang distinguisher is

$$\Pr\left[E^{-1}\big(E(x) \oplus \delta\big) \oplus E^{-1}\big(E(x \oplus \alpha) \oplus \delta\big) = \alpha\right] = p^2q^2.$$

The basic boomerang attack requires adaptive chosen plaintexts and ciphertexts. As a refinement of the boomerang attack, the rectangle attack requires only chosen plaintexts and considers as many differences as possible in the middle.

Since the boomerang attack was proposed, a series of studies have emerged focusing on the analysis of the connection probability. The probability p^2q^2 of a boomerang distinguisher is obtained under the assumption that the two differentials are independent. However, the probability may deviate from p^2q^2 on some concrete ciphers [BK09, Mur11], which demonstrates the dependency

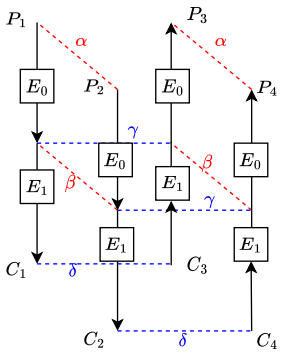

Fig. 2: Boomerang distinguisher

between the two differentials for boomerang attack. Instead of splitting the target cipher into two sub-ciphers, Dunkelman *et al.* [DKS10,DKS14] proposed to split it into three sub-ciphers and estimate the probability as p^2q^2r where r is the exact probability of the middle part, *i.e.*, the connection probability. That is also known as the sandwich attack. Later in [CHP+18], the dependency issue in boomerang distinguishers was revisited, and a tool named Boomerang Connectivity Table (BCT) was proposed. Immediately after, a generalized framework of BCT is given in [SQH19], which allows the probability r to be calculated systematically.

2.3 Key Recovery of Rectangle Attacks

The key recovery of rectangle attacks has been extensively studied in [BDK01, BDK02, ZDM+20]. Each of these works proposed its own key recovery algorithm that uses a fixed strategy for guessing key bits and the performance varies from cipher to cipher.

Recently in [SZY+22], a unified framework was put forward for finding the best rectangle attack for a given distinguisher. It contains a generic key recovery algorithm and a strategy for finding the best attack. The algorithm covers all the previous key recovery algorithms, as it supports any strategies for guessing key bits or utilizing the filters. Among all the strategies, there will be a certain one leading to an optimal attack. We call this method of finding the best key recovery attack by choosing a proper strategy the *holistic key recovery strategy*.

Suppose we treat a target cipher as $E = E_f \circ E_d \circ E_b$, where there is a boomerang distinguisher over E_d of probability P^2. As in differential attacks, a part of k_b and k_f, denoted by k'_b, k'_f, is guessed at first, and $m'_b = |k'_b|$ and $m'_f = |k'_f|$. Similarly, with the guessed subkey bits, a r'_b-bit condition on the top and a r'_f-bit condition on the bottom can be verified. Following the generic key recovery algorithm, which is recalled in Appendix A.2, the complexities of the rectangle attack are as follows.

Complexities. The data complexity is $D = \sqrt{s}2^{n/2+1}/P$ where s is the expected number of right quartets. The memory complexity is $M = f_M(D, k'_b, k'_f) =$

$D + min\{D \cdot 2^{r_b^*-1}, D^2 \cdot 2^{r_f^*-n-1}\} + 2^{t+m_b+m_f-m_b'-m_f'}$ for storing the data, the pairs and the key counters, where $0 \leq t \leq m_b' + m_f'$. The time complexity $T = f_T(D, k_b', k_f')$ is composed of four parts. The time complexity of collecting data is $T_0 = D$, the time complexity of doing partial encryption and decryption under guessed key bits is

$$T_1 = 2^{m_b'+m_f'} \cdot D,$$

the time complexity of generating pairs is

$$T_2 = 2^{m_b'+m_f'} \cdot D \cdot min\{2^{r_b^*-1}, D \cdot 2^{r_f^*-n-1}\},$$

the time complexity of generating and processing quartet candidates is

$$T_3 = 2^{m_b'+m_f'} \cdot D^2 \cdot 2^{2r_b^*} \cdot 2^{2r_f^*} \cdot 2^{-2n-2} \cdot \epsilon,$$

where ϵ is a factor that depends on the cipher, and the time complexity of the exhaustive search is $T_4 = 2^{k-h}$, where $h \leq t + m_b + m_f - m_b' - m_f'$.

Remark. In essence, the holistic key recovery strategy is to find a proper choice for k_b', k_f' such that the four parts of the time complexity are balanced and that the overall time complexity is minimized under constraints for data and memory.

3 A One-Step Framework for Finding Rectangle Attacks

In this section, we present our one-step framework for finding rectangle attacks. To begin, we use some examples from a toy cipher to convey the essential notions of the probabilistic extensions and the split-and-bunch technique. Subsequently, we provide a concise overview of the holistic key recovery strategy. By leveraging these concepts, we present a one-step framework for rectangle attacks, which is likewise applicable to differential attacks.

3.1 Probabilistic Extensions

In a classical differential key recovery attack, some rounds are added around the differential distinguisher $\alpha \to \delta$, as shown in Fig. 1. The α difference propagates to α' via E_b^{-1} with probability 1, and the δ difference propagates to δ' via E_f with probability 1. It is natural to consider what occurs when difference propagates probabilistically in the outer parts. The following are examples to illustrate the impact of probabilistic extensions on the data and time complexities. For simplicity, none of the key bits are guessed in advance and the analysis of the memory complexity is omitted in the examples.

We suppose that the round function of the 128-bit toy cipher is the same as `AES` and the key schedule is rather simple. Each round uses the 128-bit master key K as the round key RK_i. The ordering of the bytes in the state matrix is as follows.

$$\begin{pmatrix} 0 & 4 & 8 & 12 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \end{pmatrix}$$

Example 1. Suppose the probability of the inner part is P_d, and append 3 rounds. These three rounds are depicted in Fig. 3. The parameters of the attack are $n = 128, k = 128, m_b = r_b = 0, m_f = r_f = 128$.

According to Section 2.1, the data complexity is $D = 2s \cdot P_d^{-1}$. From D plaintexts, there are $D \cdot 2^{r_b + r_f - n - 1}$ pairs satisfying the plaintext difference and ciphertext difference. Thus the time complexity for constructing pairs is $T_1 = 2^{-1} \cdot D$. As there will be $2^{-1} \cdot D \cdot 2^{m_f - r_f} = 2^{-1} \cdot D$ suggestions for k_f in total, the time complexity for extracting key candidates is $T_2 = D \cdot 2^{-1} \cdot \epsilon$. We pre-compute several tables as illustrated in Table 2, so that ϵ is equivalent to about 2^{24} memory accesses. As the time complexity for the exhaustive search is flexible, we assume it is not dominant. Therefore, the overall time complexity is $T = T_2 = 2^{-1} \cdot D \cdot 2^{24} = 2^{24} \cdot s \cdot P_d^{-1}$ memory accesses.

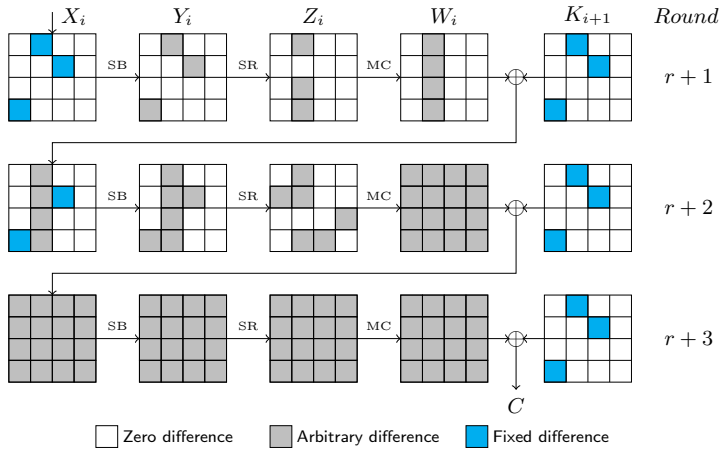

Fig. 3: The toy example of differential attack in the related-key model

Table 2: Precomputation tables for Example 1 where $eqk = \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(K)$

Tables	Involved key	Filters	Remaining pairs
1	$eqk[4,5,6,7]$	$\Delta Z_{r+2}[6] = 0$	$2^{24} \cdot 2^{-1} \cdot D$
2	$eqk[3,9]$	$\Delta X_{r+2}[3,9] = \Delta K_{r+1}[3,9]$	$2^{24} \cdot 2^{-1} \cdot D$
3	$eqk[0,1,2]$	$\Delta Z_{r+2}[0,2,3] = 0$	$2^{24} \cdot 2^{-1} \cdot D$
4	$eqk[8,10,11]$	$\Delta Z_{r+2}[8,9,10] = 0$	$2^{24} \cdot 2^{-1} \cdot D$
5	$eqk[12,13,14,15]$	$\Delta Z_{r+2}[12,13,15] = \Delta Z_{r+1}[5] = 0$ $\Delta X_{r+1}[3,4,9]$	$2^{-1} \cdot D$

Example 2. Different from Example 1, in this example the δ difference propagates to δ' via E_f with probability $P_f = 2^{-16}$. As shown in Fig. 4, the probability comes from two inactive bytes of W_{r+1}, *i.e.*, $W_{r+1}[6,7]$. The effect is that fewer bytes are activated afterward. The parameters of the attack are $n = 128, k = 128, m_b = r_b = 0, m_f = 128, r_f = 72$.

The data complexity is $D = 2s \cdot (P_d P_f)^{-1}$. The time complexities are $T_1 = D \cdot 2^{0+72-128-1} = 2^{-57} \cdot D$, $T_2 = D \cdot 2^{-57} \cdot \epsilon$. We pre-compute several tables as illustrated in Table 3, so that ϵ is equivalent to about 2^{40} memory accesses. The overall time complexity is $T_2 = 2^{-57} \cdot D \cdot 2^{40} = s \cdot P_d^{-1}$ memory accesses.

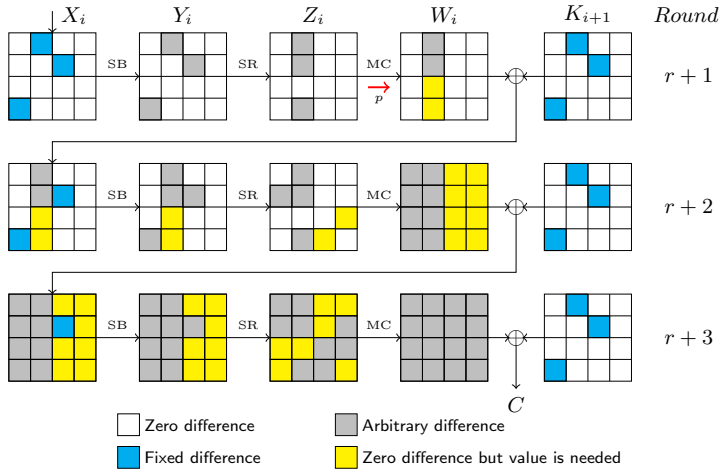

Fig. 4: The toy example of differential attack in the related-key model with probabilistic extension

Table 3: Precomputation tables for Example 2 where $eqk = \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(K)$

Tables	Involved key	Filters	Remaining pairs
1	$eqk[9]$	$\Delta X_{r+3}[9] = \Delta K_{r+2}[9]$	$2^{-57} \cdot D$
2	$eqk[0,1,2,3]$	$\Delta Z_{r+2}[0,2,3] = 0$	$2^{-49} \cdot D$
3	$eqk[4,5,6,7]$	$\Delta Z_{r+2}[6] = \Delta Z_{r+1}[6] = 0$ $\Delta X_{r+2}[3,9] = \Delta K_{r+1}[3,9]$	$2^{-49} \cdot D$
4	$eqk[8,10 \sim 15]$	$\Delta X_{r+1}[3,4,9]$	$2^{-17} \cdot D$

Comparison between Example 1 and Example 2. Example 1 uses traditional deterministic extensions in which the δ difference propagates to δ' via E_f with probability 1. On the contrary, Example 2 uses probabilistic extensions in which the δ difference propagates to δ' via E_f with probability $P_f = 2^{-16}$.

Increasing the data complexity. The data complexity for the case with probabilistic extensions is $P_f^{-1} = 2^{16}$ times the data complexity with deterministic extensions. In other words, the data complexity of the attack is determined by the overall probability $P_d P_f$ rather than by the probability P_d only.

Decreasing the time complexity. The time complexity of Example 1 is $2^{24} \cdot s \cdot P_d^{-1}$ and it is $s \cdot P_d^{-1}$ for Example 2. By using the probabilistic extensions, the time complexity is reduced by a factor of 2^{24}. Thus, probabilistic extensions are a technique that not only trades off data against time but may also have a gain as $2^{24} > 2^{16}$.

Flexible boundaries. Different from the classical attack in which the boundaries between the inner and outer parts are fixed and well-aligned, the boundaries are flexible when probabilistic extensions are allowed. For a specified round, it is possible that some S-boxes belong to the inner part while others belong to the outer part. More interestingly, probabilistic differential transitions and deterministic differential transitions may occur simultaneously in the outer part and they may be even interleaved, see the attack on `SKINNY-128-256` in Section 4.2. Thus, there are no predefined boundaries. The boundaries will depend on differential transitions.

Increasing the number of filters and earlier usage. On one hand, the utilization of probabilistic extensions leads to an expansion of the total number of filter bits, growing from n to $n + \log_2 P_f^{-1}$. On the other hand, the filters appear earlier than before in the key recovery attack. The ability to employ filters at an earlier stage significantly enhances the efficiency of key candidate extraction, providing a reduction in the time complexity.

Remarks. Allowing probabilistic extensions essentially enables probabilistic differential propagations in the outer parts, which is the focus of key recovery.

- It is clear that the data complexity is related to the overall probability. Probabilistic extensions can strike a harmonious balance between data complexity and time complexity. However, it is important to note that probabilistic extensions do not necessarily increase data complexity in the search for a globally optimal attack, as exemplified by the attack on `Deoxys-BC-384` in Section 4.1.
- Employing probabilistic extensions has the potential to reduce the density of active cells in the outer parts while simultaneously increasing the number of available filters. Notably, the probabilistic differential transitions are flexibly distributed in the outer parts and they do not have to connect to the differential trail over the inner part directly. This flexibility facilitates the adaptable selection of filters, ultimately resulting in improved time complexity.
- These extensions can be applied to E_b as well. Setting $P_b \geq 1$, $P_f \geq 1$ makes the probabilistic extensions cover the traditional deterministic extensions. Different extensions lead to different placements of filters. Among all possible extensions, there may be some that lead to more efficient key recovery attacks, in terms of time complexity or the number of rounds that can be attacked, compared to what deterministic extensions can achieve. Our applications to concrete ciphers will confirm this. Importantly, although our examples in this section primarily pertain to differential attacks, these same principles are applicable to rectangle attacks as well.

3.2 The Split-and-Bunch Technique

From Example 2, it is known that using probabilistic extensions brings the filters closer to the ciphertext, but the involved keys are the same. In this example, the last step is to verify $\Delta X_{r+1}[3,4,9]$. To compute this 3-byte difference, one has to know the values of $W_{r+1}[6,7]$. However, another 56 key bits, *i.e.*, $eqtk[8,10 \sim 15]$ are required to compute $W_{r+1}[6,7]$ from the ciphertext. In other words, the 56-bit key brings the final 24-bit filter. Therefore, the remaining pairs expand by a factor of 2^{32} as shown in Table 3. A question arises: Does the 56-bit key have to be traversed? To answer this question, we start with an observation on Example 2.

Observation 1 *Let $k_f = k_f^1 || k_f^2$ with $m_f^1 = 72$ and $m_f^2 = 56$. For one pair of ciphertexts, if we traverse all possible values of the 16-bit $W_{r+1}[6,7]$ without trying all possible values of the 56-bit k_f^2, we can find that:*

- *For the right k_f^1, when a right pair takes the right values in the two bytes, it makes the counter plus one. Traversal must cause the right value to be taken, so the number of suggestions for the right subkey is the same.*
- *For a wrong pair, the number of suggestions for the wrong key is equal to expanding the number of pairs by a factor of 2^{16}.*

We call this technique a split-and-bunch technique. Let e be the average number of suggestions for a wrong key k_f when the split-and-bunch technique is not used. When it is used, the number of suggestions for a wrong k_f^1 is expanded by a factor of $2^{m_e} = 2^{16}$ from e, i.e., it is $e \cdot 2^{m_e}$, while the number of suggestions for the right k_f^1 is $s + e \cdot 2^{m_e} = s + e \cdot 2^{16}$.

In Example 2, the whole key is involved in E_f, *i.e.*, $m_f = 128$. In total we will get $D \cdot 2^{m_f - n - 1} = D \cdot 2^{-1}$ suggestions for 128-bit k_f. On average, the number of suggestions for a wrong subkey is $e = D \cdot 2^{-129} = P_d^{-1} \cdot s \cdot 2^{-128} \leq 1$, while it is $s + e$ for the right subkey.

When we use the split-and-bunch technique and set counters only for the $m_f^1 = 72$ key bits, we will get $D \cdot 2^{-57} \cdot 2^{16} = D \cdot 2^{-41}$ suggestions for 72-bit k_f. Thus the number of suggestions for a wrong k_f^1 is $e' = 2^{-41-72} \cdot D = P_d^{-1} \cdot s \cdot 2^{-112}$, while it is $s + e'$ for the right k_f^1, where $e' = 2^{m_e} \cdot e$. Thus, $m_e = 16$ is called the number of expansion bits. When e and e' are enough small, one could use the split-and-bunch technique to decrease the time complexity. The following example gives a better illustration.

Example 3. we use the split-and-bunch technique for this attack, as shown in Fig. 4. The number m_f of the involved key k_f in E_f is 72. The other parameters are $n = 128, k = 128, m_b = r_b = 0, r_f = 72, m_e = 16$.

The data complexity is $D = 2s \cdot (P_d P_f)^{-1}$. The time complexities are $T_1 = D \cdot 2^{72-128-1} = 2^{-57} \cdot D$, $T_2 = D \cdot 2^{-57} \cdot \epsilon$. We pre-compute several tables as illustrated in Table 4, so that ϵ is equivalent to about 2^8 memory accesses. The overall time complexity is $T = T_2 = 2^{-49} \cdot D = 2^{-32} \cdot s \cdot P_d^{-1}$ memory accesses.

Table 4: Precomputation tables for Example 3 where $eqk = \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(\mathtt{K})$

Tables	Involved key	Filters	Remaining pairs
1	$eqk[9]$	$\Delta X_{r+3}[9] = \Delta K_{r+2}[9]$	$2^{-57} \cdot D$
2	$eqk[0,1,2,3]$	$\Delta Z_{r+2}[0,2,3] = 0$	$2^{-49} \cdot D$
3	$eqk[4,5,6,7]$	$\Delta Z_{r+2}[6] = \Delta Z_{r+1}[6] = 0$ $\Delta X_{r+2}[3,9] = \Delta K_{r+1}[3,9]$	$2^{-49} \cdot D$
4	$W_{r+1}[6,7]$	$\Delta X_{r+1}[3,4,9]$	$2^{-57} \cdot D$

Comparisons between Example 2 and Example 3. Example 3 uses the split-and-bunch technique and the time complexity is $T = 2^{-49} \cdot D = 2^{-32} \cdot s \cdot P_d^{-1}$, which reduces the time complexity of Example 2 by a factor 2^{-32}.

Decreasing the time complexity. The split-and-bunch technique reduces the time complexity further. This technique splits the key bits into two distinct categories: friendly key bits and unfriendly key bits. The former facilitates the efficient application of filters, while the latter, albeit intricately connected to some filters, can map to values comprising only a limited number of bits (i.e., m_e bits). Rather than directly trying all possibilities of the unfriendly key bits themselves, we streamline the process by traversing the m_e-bit value, ensuring that the correct key is not overlooked. Consequently, the split-and-bunch technique results in an expansion of the counter value for incorrect keys but simultaneously reduces the overall number of key suggestions. This deliberate trade-off effectively minimizes the overall time complexity.

When counters are not used. Differential attacks are possible without using counters in some cases, for example, using enumeration [Din14]. The split-and-bunch can also be used and may provide an improvement. Suppose we split at an m_e-bit state, after which key extraction becomes two parts. If the complexity of key extraction for each part is reduced by more than 2^{m_e}, then split-and-bunch can provide an improvement.

3.3 Holistic Key Recovery Strategy

Probabilistic extensions fit well with the unified key recovery algorithm from [SZY+22]. However, the situation for the holistic key recovery strategy is different. Previously, the holistic key recovery strategy was considered in cases where a boomerang distinguisher was given. Hence, the boundaries between the inner and outer parts are clear and well-aligned. The only task is to find a proper set of key bits to be guessed in advance so that the overall time complexity is optimized.

On the contrary, the boundaries are not predetermined when probabilistic extensions are allowed. Instead, the boundaries and the set of guessed key bits should be determined together. That is, the situation for the holistic key recovery strategy is more generic in the presence of probabilistic extensions. To apply the

holistic key recovery strategy, one needs to determine the involved key bits k_b, k_f from the boundaries, the guessed key bits k'_b, k'_f, and other parameters affected by k'_b, k'_f, including the obtained filters.

3.4 The Framework for Rectangle Attacks

Previously, one looked for the best rectangle attack for a given distinguisher. In this subsection, we present the one-step framework for rectangle attacks where the distinguisher and the extended part are considered together as a whole. The framework has four components: the unified key recovery algorithm, the holistic key recovery strategy, the core part, and the probabilistic extensions. In practice, we search for a cell-wise active pattern with the overall time complexity of the attack as the objective function and then instantiate it. As the treatment of the core part remains as in previous rectangle attacks, we focus on the upper and lower parts where probabilistic extensions occur and the determination of the boundaries is needed. Next, we present a constraint programming model for these parts.

Suppose the boomerang distinguisher $\alpha \to \delta$ over the middle part E_d has probability P_d^2. Suppose α propagates backward to α' with probability $P_b \leq 1$ and δ propagates backward to δ' with probability $P_f \leq 1$. The data complexity depends on these probabilities. Besides the probabilities, another important factor is the boundaries between the inner and outer parts since they mark the positions the key recovery phase has to reach and also determine the involved keys k_b, k_f.

Data Complexity. The probability for the whole attack is $P^2 = P_b^2 P_d^2 P_f^2$. We will show the formula for the data complexity remains as $D = \sqrt{s} 2^{n/2+1}/P$. Each plaintext structure takes all possible values for the r_b bits and chooses a constant for the remaining $n - r_b$ bits. For each structure, there are 2^{2r_b-1} pairs of plaintext with difference in V_b and $P_b \cdot 2^{r_b-1}$ of them satisfy α difference. That is equal to a $(\log_2 P_b^{-1} + r_b)$-bit condition. The distinguisher over the inner part has probability P_d^2, the number of quartets satisfying the input difference α should be at least $sP_d^{-2}2^n$ for a rectangle attack, where s is the expected number of right quartets. Suppose the number of structures needed is y. These structures can constitute $2 \cdot \binom{y \cdot P_b \cdot 2^{r_b-1}}{2}$ quartets that satisfy α difference. Due to the output difference of E_d δ propagating forward over E_f with probability P_f to δ', a right quartet propagates forward over E_f with probability P_f^2. Thus we get $2 \cdot \binom{y \cdot P_b \cdot 2^{r_b-1}}{2} \cdot P_f^2 = sP_d^{-2}2^n$. Then $y = \sqrt{s} 2^{n/2-r_b+1}/P$ and the data complexity is $D = y \cdot 2^{r_b} = \sqrt{s} 2^{n/2+1}/P$.

Labels. Due to the probabilistic extensions, there is a mix of concrete differences and truncated differences in the extended parts. For each cell of the internal state, we use two labels to describe its difference: $(x, y) \in \{(0,0), (1,0), (1,1)\}$, where

- inactive, denoted by $(x, y) = (0, 0)$ or $\square$;

– active with a fixed non-zero difference, denoted by $(x, y) = (1, 0)$, or ■;
– active with an arbitrary (truncated) difference, denoted by $(x, y) = (1, 1)$ or ■.

Additionally, we use a label v to denote if the value of the cell is needed to verify the distinguisher. If it is, we denote by $v = 1$ and $v = 0$ otherwise. Then the labels will help to identify the boundaries as well as the probabilities P_b, P_f of the extension.

Boundaries and P_b, P_f. We take the forward extension as an example. The same techniques apply to the backward extension but in a reverse direction. There are several cases that happen with a probability.

– For the S-box layer, the probabilistic extension involves two cases: ■ → ■ and ■ → ■. The number of such cases can be computed by

$$\sum_i (O_i.x - O_i.y), \tag{1}$$

where $O_i.x, O_i.y$ are labels for the output cells of the S-box layer.
If the transition is ■ → ■, then the output of the S-box is needed for the verification, *i.e.*, $O.v = 1$ for the output. Besides, $O.v = I.v$. Meanwhile, the fixed input difference acts as a one-cell filter.
– For the linear layer, where each output cell is a linear combination of some input cells, the probabilistic extension happens when truncated differences exist in the input but the output is a fixed difference. This fixed difference happens with a probability. Let $T = 1$ if some $I_i.y = 1$ and $T = 0$ if all $I_i.y = 0$. The number of such cases can be obtained by

$$\sum (T - O.y). \tag{2}$$

Conversely, each input cell is a linear combination of some output cells. If $v = 1$ for the input cell, then the values for the involved output cells are also needed.

Now P_b, P_f can be computed (or estimated) by considering Eqs. (1) and (2) together. The outer parts are composed of those internal state cells with label $v = 1$ as well as the operations and functions on them. This implies the boundaries. Then k_b, k_f are those key bits that are needed for determining the internal state cell with label $v = 1$.

Guessing the key and filters. The holistic key recovery strategy uses the guess-and-determine logic, which can be modeled as follows. For each key addition, if the key is guessed (a key cell in k_f) and the output is known, then the input can be determined. Therefore, we introduce another label d for each state cell to denote if it is determined. For each key cell, we introduce a label g to denote if it is guessed. We then have $I.d = 1$ when $(K.g = 1) \wedge (O.d = 1)$ for $I \oplus K = O$. Then the determination can proceed backward naturally. For a linear operation, a

filter is reached under the following conditions: (1) its output contains truncated cells, (2) the difference of an input cell can be determined, and (3) this input difference has a fixed difference. For the S-box, if the input can be determined and the input difference is fixed, then we have a filter. Finally, the number of filters r'_f can be computed by recording such cases and the number of guessed keys k'_f can be computed by counting distinct key cells with $g = 1$.

Constraints for the complexities. The constraints for the data and memory complexities are also added, such as $D < 2^n$. The objective is to minimize the overall time complexity as described in Section 2.3. The source codes for the constraint programmind model are available here.

3.5 The Framework for Differential Attacks

Similar to the framework for rectangle attacks, a framework for differential attacks can be built, because the basic ideas and all the involved parameters are the same. The only difference lies in the computation of complexities.

4 Applications

In this section, we apply our new framework to `Deoxys-BC-384`, `SKINNY-128-256`, and `CRAFT` block ciphers. For `Deoxys-BC-384`, using the one-step framework for rectangle attacks obtains a 15-round rectangle attack for the first time, containing a 10-round inner part rather than the longest 11-round one. For `SKINNY-128-256`, we get a new 26-round rectangle attack with reduced data and time complexities. For `CRAFT`, we obtain a 21-round differential attack and a 23-round differential attack by the new framework.

4.1 Application to `Deoxys-BC-384`

Specification. `Deoxys-BC` is an `AES`-based tweakable block cipher [JNPS16], based on the tweakey framework [JNP14]. The `Deoxys` authenticated encryption scheme makes use of two versions of the cipher as its internal primitive: `Deoxys-BC-256` and `Deoxys-BC-384`. Both versions are ad-hoc 128-bit tweakable block ciphers which besides the two standard inputs, a plaintext P (or a ciphertext C) and a key K, also take an additional input called a *tweak* T. The concatenation of the key and tweak states is called the *tweakey* state. For `Deoxys-BC-384`, the tweakey size is 384 bits.

`Deoxys-BC` is an `AES`-like design, *i.e.*, it is an iterative substitution-permutation network (SPN) that transforms the initial plaintext (viewed as a 4×4 matrix of bytes) using the `AES` round function, with the main differences with `AES` being the number of rounds and the round subkeys that are used every round. `Deoxys-BC-384` has 16 rounds.

Similarly to the `AES`, one round of `Deoxys-BC` has the following four transformations applied to the internal state in the order specified below:

– AddRoundTweakey – XOR the 128-bit round subtweakey to the internal state.
– SubBytes – Apply the 8-bit AES S-box to each of the 16 bytes of the internal state.
– ShiftRows – Rotate the 4-byte i-th row left by $\rho[i]$ positions, where $\rho = (0, 1, 2, 3)$.
– MixColumns – Multiply the internal state by the 4×4 constant MDS matrix of AES.

After the last round, a final AddRoundTweakey operation is performed to produce the ciphertext.

We denote the concatenation of the key K and the tweak T as KT, *i.e.*, $KT = K||T$. For Deoxys-BC-384, the size of KT is 384 bits, and we denote the first, second and third 128-bit words of KT by W_3, W_2 and W_1, respectively. Finally, we denote by STK_i the 128-bit *subtweakey* that is added to the state at round i during the AddRoundTweakey operation. For Deoxys-BC-384, a subtweakey is defined as $STK_i = TK_i^1 \oplus TK_i^2 \oplus TK_i^3 \oplus RC_i$. The tweakey schedule algorithm is defined as $TK_{i+1}^1 = h(TK_i^1)$, $TK_{i+1}^2 = h(LFSR_2(TK_i^2))$ and $TK_{i+1}^3 = h(LFSR_3(TK_i^3))$, where the byte permutation h is defined as

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 1 & 6 & 11 & 12 & 5 & 10 & 15 & 0 & 9 & 14 & 3 & 4 & 13 & 2 & 7 & 8 \end{pmatrix},$$

with the 16 bytes of a tweakey word numbered by the usual AES byte ordering.

14-Round rectangle attack on Deoxys-BC-384. By our new one-step framework, we get a 14-round rectangle attack on Deoxys-BC-384 with a 10-round inner part in E_d, 1-round in E_b and 3-round in E_f, as shown in Fig. 5. Detailed of the 10-round inner part refers to Fig. 11. The probability of E_b is 1, the probability of E_d is $P_d^2 = 2^{-14\times2-11.4-7\times2-14\times2} = 2^{-81.4}$ and the probability of E_f is $P_f^2 = 2^{-8\times2}$. Thus the probability of the whole rectangle attack is $P^2 = P_b^2 P_d^2 P_f^2 = 2^{-97.4}$, and other parameters of the attack are: $n = 128, k = 384, m_b = r_b = 80, m_f = 8\times(16+7+3) = 208$ and $r_f = 8\times13 = 104$.

The best guessing parameters are $m_b' = 8 \times 10 = 80, r_b' = 8 \times 10 = 80$, $m_f' = 8 \times (5+1) = 48, r_f' = 32$, $r_b^* = 0$ and $r_f^* = 72$, which means guessing 10 bytes of k_b and 6 bytes of k_f. The 6 bytes of k_f are $eqtk_{14}[8, 9, 10, 11, 12]$ and $eqtk_{13}[13]$ respectively. The complexities of our new attack are as follows[6].

- The data complexity is $D_R = 4 \cdot D = 4 \cdot y \cdot 2^{r_b} = 4 \cdot \sqrt{s} \cdot 2^{n/2}/P = 4 \cdot \sqrt{s} \cdot 2^{48.7+64} = \sqrt{s} \cdot 2^{114.7}$.
- The memory complexity is $M_R = D_R + D \cdot 2^{r_b*} + 2^{t+m_b+m_f-m_b'-m_f'} = \sqrt{s} \cdot 2^{114.7} + \sqrt{s} \cdot 2^{112.7} + 2^{t+160}$.
- The time complexity $T_1 = 2^{m_b'+m_f'} \cdot D_R = \sqrt{s} \cdot 2^{128+114.7} = \sqrt{s} \cdot 2^{242.7}$;
- $T_2 = 2^{m_b'+m_f'} \cdot D = \sqrt{s} \cdot 2^{128+112.7} = \sqrt{s} \cdot 2^{240.7}$;

[6] We use a variant of formulas from Section 2.3 in the related-key setting for ciphers with a linear key schedule.

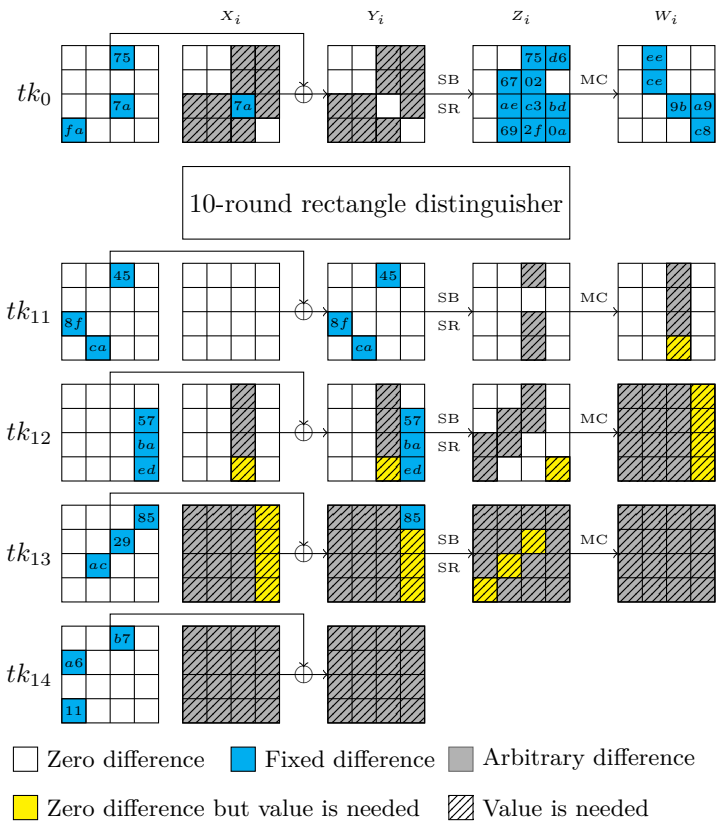

Fig. 5: 14-round Rectangle Attack on `Deoxys-BC-384`

- $T_3 = 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r^*_b} \cdot 2^{2r^*_f} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{128+112.7\times2+72\times2-2\times128} \cdot \epsilon = s \cdot 2^{241.4} \cdot \epsilon$;
- $T_4 = 2^{m'_b+m'_f-t} \cdot 2^{k+t-m'_b-m'_f-h} = 2^{k-h}$.

Processing a candidate quartet to retrieve the rest k_f can be realized by looking up tables. We pre-compute several tables as illustrated in Table 5, detailed refers to [SZY+22], so that ϵ is equivalent to about 2^{25} memory accesses which is around $2^{25} \times \frac{1}{14} \times \frac{1}{16} = 2^{17.19}$ encryption. The time complexity and memory complexity of the tables are related to the involved state bits, the involved subkeys and the filters. The time complexity and memory complexity of the tables are all 2^{128}. If we set $s = 4, t = 0$, then the data, memory, and time complexities of our attack are $2^{115.7}$, 2^{160} and $2^{260.59}$ respectively.

Table 5: Precomputation tables for the 14-round attack on `Deoxys-BC-384` where $eqtk_i = \mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(\mathtt{tk_i})$ (let $Q = s \cdot 2^{241.4}$)

Tables	Involved key	Filters	Remaining quartets
1	$eqtk_{14}[0,1,2,3]$	$\Delta Z_{12}[0,1] = 0$	Q
2	$eqtk_{13}[15]$	$\Delta Y_{12}[15] = 0xed$	$Q \cdot 2^{-8}$
3	$eqtk_{14}[4,5,6,7]$	$\Delta Z_{12}[4,7] = 0$	$Q \cdot 2^{-8}$
4	$eqtk_{13}[14]$	$\Delta Y_{12}[14] = 0xba$	$Q \cdot 2^{-16}$
5	$eqtk_{13}[8,9,10]$	$\Delta Z_{11}[10] = 0$	$Q \cdot 2^{-8}$
6	$eqtk_{14}[13,14,15], eqtk_{13}[11]$	-	$Q \cdot 2^{24}$
7	$eqtk_{12}[2,7,8]$	$\Delta Y_{12}[2,7,8] = 0x8f, 0xca, 0x45$	Q

Improved 14-round rectangle attack by the split-and-bunch technique. From the time complexity of the 14-round rectangle attack above, we can find that in subtable 6 there are 4 bytes involved subkeys and 0 filter. The number of quartets in this subtable reaches the maximum, making the time complexity of this step dominant.

With the same guessing strategy, by using the split-and-bunch technique, the number of the involved subkeys bits in E_f is $m_f = 8 \times (13+6+3) = 176$. And the number of the counts for a wrong key is $e = D^2 \cdot 2^{m_e-2n-16} = s \cdot 2^{225.4+8-272} = s \cdot 2^{-38.6} \ll s$. We get the new ϵ as illustrated in Table 6. The time complexity and memory complexity of the tables are 2^{128}. It is shown ϵ is equivalent to about 2 memory accesses which is around $2 \times \frac{1}{14} \times \frac{1}{16} = 2^{-6.81}$ encryption. The memory complexity is $\sqrt{s} \cdot 2^{114.7} + \sqrt{s} \cdot 2^{112.7} + 2^{t+(176+80)-80-48}$. If we set $s = 4, t = 0$, then the data, memory and time complexities of our attack are $2^{115.7}$, 2^{128} and $2^{243.7}$ respectively.

15-Round rectangle attack on `Deoxys-BC-384`. Using our new framework, we get a 15-round rectangle attack which is equal to add one round after the 14-round rectangle attack, as shown in Fig. 6.

In order to get an effective attack, we also use the split-and-bunch technique in the 15-round rectangle attack. The parameters of the attack are: $m_b = r_b =$

Table 6: Precomputation tables for the improved 14-round attack on `Deoxys-BC-384` where $eqtk_i = \texttt{SR}^{-1} \circ \texttt{MC}^{-1}(\texttt{tk}_\texttt{i})$ (let $Q = s \cdot 2^{241.4}$)

Tables	Involved key	Filters	Remaining quartets
1	$eqtk_{14}[0,1,2,3]$	$\Delta Z_{12}[0,1] = 0$	Q
2	$eqtk_{13}[15]$	$\Delta Y_{12}[15] = 0xed$	$Q \cdot 2^{-8}$
3	$eqtk_{14}[4,5,6,7]$	$\Delta Z_{12}[4,7] = 0$	$Q \cdot 2^{-8}$
4	$eqtk_{13}[14]$	$\Delta Y_{12}[14] = 0xba$	$Q \cdot 2^{-16}$
5	$eqtk_{13}[8,9,10]$	$\Delta Z_{11}[10] = 0$	$Q \cdot 2^{-8}$
6	$eqtk_{12}[2,7,8], W_{11}[11]$	$\Delta Y_{12}[2,7,8] = 0x8f, 0xca, 0x45$	$Q \cdot 2^{-24}$

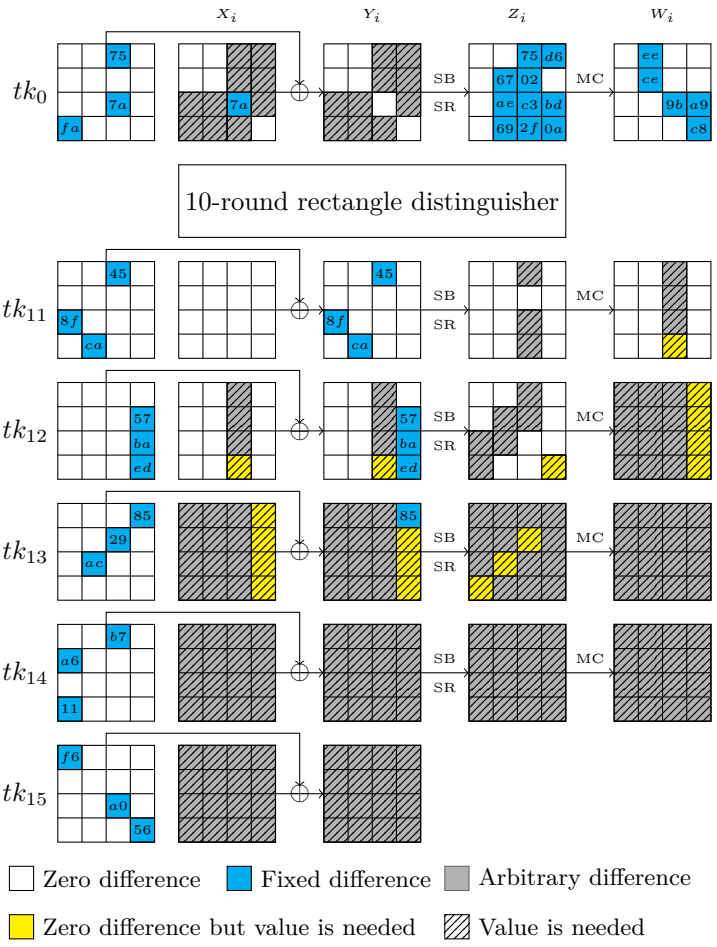

Fig. 6: 15-round Rectangle Attack on `Deoxys-BC-384`

$80, m_f = 8\times(16+13+6+3) = 304$ and $r_f = 128$. Due to the key schedule, $k_b \cup k_f$ contains 376 information bits. The best guessing parameters are $m'_b = 80, r'_b = 80$ and $m'_f = 8 \times (16 + 5 + 1) = 176, r'_f = 8 \times (4 + 3) = 56$, which means guessing the whole subkey of k_b, 16 bytes of tk_{15}, 5 bytes of the equivalent subkey $eqtk_{14}$ and 1 bytes of the equivalent subkey $eqtk_{13}$. The complexities of our 15-round attack are as follows:

- The data complexity is $D_R = 4 \cdot D = 4 \cdot y \cdot 2^{r_b} = 4 \cdot \sqrt{s} \cdot 2^{n/2}/\tilde{P} = 4 \cdot \sqrt{s} \cdot 2^{48.7+64} = \sqrt{s} \cdot 2^{114.7}$.
- The memory complexity is $M_R = D_R + D \cdot 2^{r_b*} + 2^{t+m_b+m_f-m'_b-m'_f} = \sqrt{s} \cdot 2^{114.7} + \sqrt{s} \cdot 2^{112.7} + 2^{t+120}$.
- The time complexity $T_1 = 2^{m'_b+m'_f} \cdot D_R = \sqrt{s} \cdot 2^{256+114.7} = \sqrt{s} \cdot 2^{370.7}$;
- $T_2 = 2^{m'_b+m'_f} \cdot D = \sqrt{s} \cdot 2^{256+112.7} = \sqrt{s} \cdot 2^{368.7}$;
- $T_3 = 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r^*_b} \cdot 2^{2r^*_f} \cdot 2^{-2n} \cdot \epsilon = s \cdot 2^{256+112.7\times2+72\times2-2\times128} \cdot \epsilon = s \cdot 2^{369.4} \cdot \epsilon$;
- $T_4 = 2^{m'_b+m'_f-t} \cdot 2^{k+t-m'_b-m'_f-h} = 2^{k-h}$.

Similar to the improved rectangle attack on 14-round `Deoxys-BC-384`, for $s \cdot 2^{369.4}$ quartets, ϵ is equivalent to about 2 memory accesses which is around $2 \times \frac{1}{14} \times \frac{1}{16} = 2^{-6.81}$ encryption. The time complexity and memory complexity of the tables both are 2^{128}. If we set $s = 4, t = 0$ then the data, memory, and time complexities are $2^{115.7}$, 2^{128} and $2^{371.7}$, respectively.

Table 7: Comparisons of rectangle attacks on `Deoxys-BC-384`

P^2	Rounds	m_b, m_f	m'_b, m'_f	Data	Memory	Time	Reference
$2^{-118.4}$	14	88, 168	88, 24	$2^{125.2}$	2^{140}	2^{260}	[DQSW22]
$2^{-97.4}$	14	80, 208	80, 48	$2^{115.7}$	2^{160}	$2^{260.59}$	This
$2^{-97.4}$	14	80, 176	80, 48	$2^{115.7}$	2^{128}	$2^{243.7}$	This
$2^{-97.4}$	15	80, 304	80, 176	$2^{115.7}$	2^{128}	$2^{371.7}$	This

The comparison with the previous rectangle attacks is presented in Table 7.

Note: For `Deoxys-BC`, we search for the active patterns of the whole attack with the total time complexity of the attack as the objective function in one step and then instantiate these patterns. In order to balance the time complexity and data complexity, we impose some restrictions on the data complexity in the model. These two attacks are the instances with the optimal time complexity found under the restriction of data complexity.

4.2 Application to `SKINNY`

Specification. `SKINNY` [BJK+16] is a family of lightweight block ciphers which adopt the substitution-permutation network and elements of the TWEAKEY framework [JNP14]. Members of `SKINNY` are denoted by `SKINNY`-n-tk, where

$n \in \{64, 128\}$ is the block size and $tk \in \{n, 2n, 3n\}$ is the tweakey size. The internal states of SKINNY are represented as 4×4 arrays of cells with each cell being a nibble in case of $n = 64$ bits and a byte in case of $n = 128$ bits. The tweakey state is seen as a group of z 4×4 arrays, where, $z = tk/n$. The arrays are marked as $TK1$, $(TK1, TK2)$ and $(TK1, TK2, TK3)$ for $z = 1, 2, 3$ respectively.

SKINNY iterates a round function for N_r rounds and each round consists of the following five steps.

1. SubCells (SC) - A 4-bit (resp. 8-bit) S-box is applied to all cells when n is 64 (resp. n is 128).
2. AddConstants (AC) - This step adds constants to the internal state.
3. AddRoundTweakey (ART) - The first two rows of the internal state absorb the first two rows of TK, where $TK = \bigoplus_{i=1}^{z} TK_i$.
4. ShiftRows (SR) - Each cell in row j is rotated to the right by j cells.
5. MixColumns (MC) - Each column of the internal state is multiplied by matrix M whose branch number is only 2.

The tweakey schedule of SKINNY is a linear algorithm. The tk-bit tweakey is first loaded into z 4×4 tweakey states. After each ART step, a cell-wised permutation P is applied to each tweakey state, where P is defined as: $P = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$. Then cells in the first two rows of all tweakey states but TK_1 are individually updated using LFSRs. For complete details of the tweakeys scheduling algorithm, one can refer to [BJK+16].

26-Round rectangle attack on SKINNY-128-256. We obtain a new 26-round rectangle attack with the inner part is modified by the original 18-round distinguisher in [DQSW22], as depicted in Fig. 7. The probability of the whole rectangle attack is $P_d^2 = 2^{-115.86}$. The other parameters of the attack are $n = 128, k = 256, m_b = 64, r_b = 56, m_f = 168$ and $r_f = 96$. The best-guessing parameters are $m'_b = 64, r'_b = 56$ and $m'_f = 32, r'_f = 32$. The complexities of our 26-round rectangle attack are as follows:

- The data complexity is $D_R = 4 \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/P = \sqrt{s} \cdot 2^{123.93}$.
- The memory complexity is $M_R = D_R + D \cdot 2^{r_b^*} + 2^{t+m_b+m_f-m'_b-m'_f} = \sqrt{s} \cdot 2^{123.93} + \sqrt{s} \cdot 2^{121.93} + 2^{136+t}$.
- The time complexity $T_1 = 2^{m'_b+m'_f} \cdot D_R = \sqrt{s} \cdot 2^{12\times 8+123.93} = \sqrt{s} \cdot 2^{219.93}$;
- $T_2 = 2^{m'_b+m'_f} \cdot D \cdot 2^{r_b-r'_b} = \sqrt{s} \cdot 2^{12\times 8+121.93} = \sqrt{s} \cdot 2^{217.93}$;
- $T_3 = 2^{m'_b+m'_f} \cdot D^2 \cdot 2^{2r_b^*+2r_f^*-2n} \cdot \epsilon = s \cdot 2^{12\times 8+121.93\times 2+2\times 64-2\times 128} \cdot \epsilon = s \cdot 2^{211.86} \cdot \epsilon$;
- $T_4 = 2^{256-h}, h < 136 + t$.

The remaining step is similar to [SZY+22], the ϵ is equivalent to about 35 memory access which is around $35 \times \frac{1}{26} \times \frac{1}{16} \approx 2^{-3.57}$ encryption. If we set $s = 1, t = 0$, then the data, memory, and time complexities of our attack are $2^{123.93}$, 2^{136}, and $2^{219.93}$.

The comparison with the previous rectangle attacks is presented in Table 8.

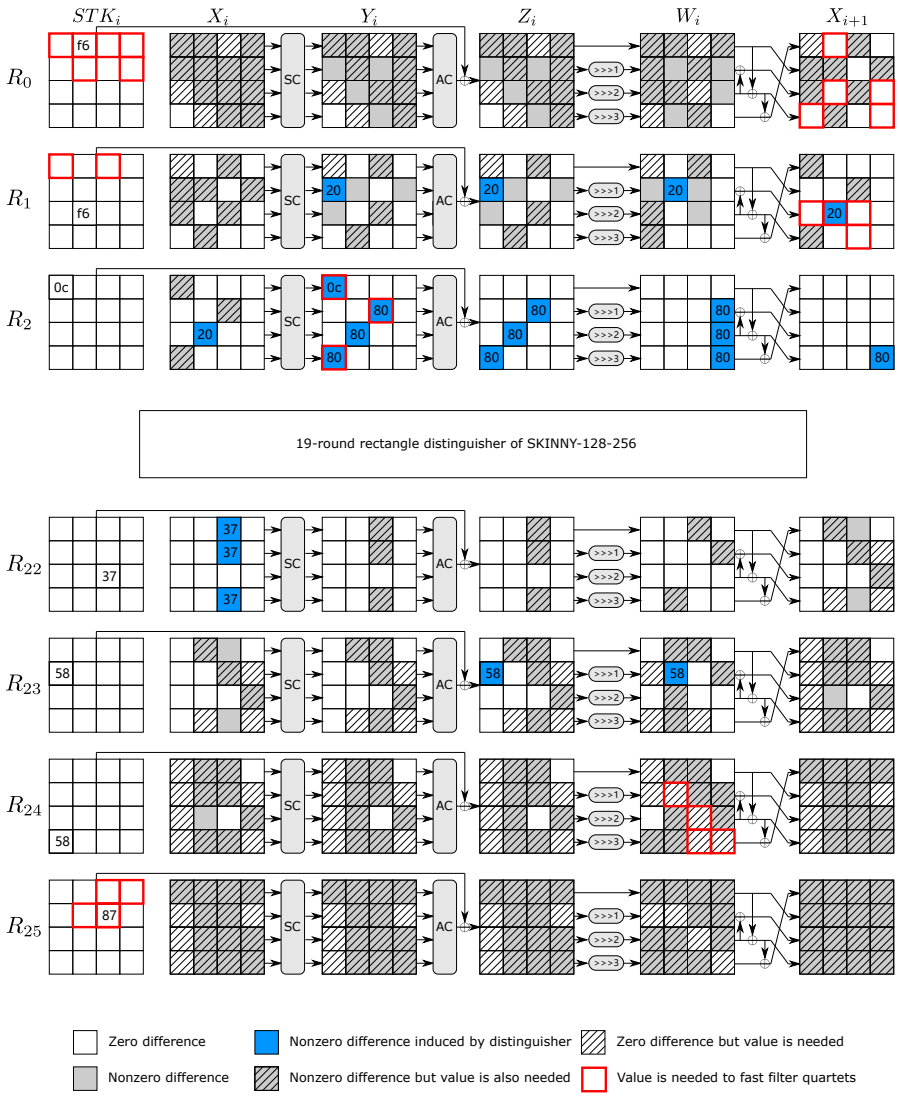

Fig. 7: 26-round Rectangle Attack on SKINNY-128-256

Table 8: Comparisons of key recovery attacks on SKINNY-128-256

P^2	m_b, m_f	m'_b, m'_f	Data	Memory	Time	Reference
$2^{-121.07}$	88, 168	88, 24	$2^{126.53}$	$2^{128.44}$	$2^{254.4}$	[DQSW22]
$2^{-121.07}$	88, 168	72, 32	$2^{126.53}$	2^{136}	$2^{241.38}$	[SZY+22]
$2^{-115.86}$	64,168	64,32	$2^{123.93}$	2^{136}	$2^{219.93}$	This

4.3 Application to CRAFT

Specification. CRAFT is a lightweight tweakable block cipher that was introduced by Beierle *et al.* [BLMR19]. It supports 64-bit plaintexts, 128-bit keys, and 64-bit tweaks. Its round function is composed of involutory building blocks. The 64-bit input is arranged as a state of 4×4 nibbles. The state is then going through 32 rounds $\mathcal{R}_i, i \in 0, \cdots, 31$, to generate a 64-bit ciphertext. As depicted in Fig. 8, each round, excluding the last round, has five functions, *i.e.*, MixColumn (MC), AddRoundConstants (ARC), AddTweakey (ATK), PermuteNibbles (PN), and S-box (SB). The last round only includes MC, ARC and ATK, *i.e.*, $\mathcal{R}_{31} = \texttt{ATK}_{31} \circ \texttt{ARC}_{31} \circ \texttt{MC}$, while for any $0 \leq i \leq 30$, $\mathcal{R}_i = \texttt{SB} \circ \texttt{PN} \circ \texttt{ATK}_i \circ \texttt{ARC}_i \circ \texttt{MC}$.

The tweakey schedule of CRAFT is rather simple. Given the secret key $K = K_0 \| K_1$ and the tweak $T \in \{0,1\}^{64}$, where $K_i \in \{0,1\}^{64}$, four round tweakeys $TK_0 = K_0 \oplus T$, $TK_1 = K_1 \oplus T$, $TK_2 = K_0 \oplus Q(T)$ and $TK_3 = K_1 \oplus Q(T)$ are generated, where Q is a nibble-wise permutation. Then at the round $\mathcal{R}_i$, $TK_{i\%4}$ is used as the subtweakey.

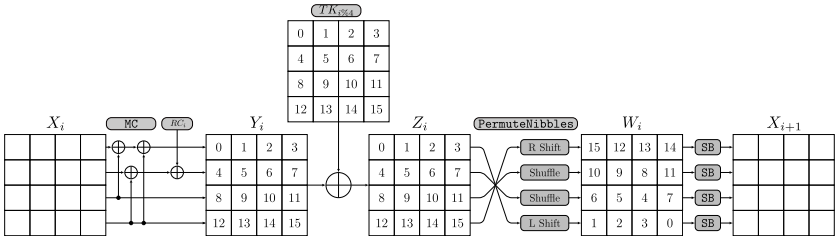

Fig. 8: A round of CRAFT

21-Round differential attack on CRAFT. By using our new framework, we obtain a 21-round differential attack on CRAFT, as depicted in Fig. 9, which contains a 14-round inner part, 3-round in E_b and 4-round in E_f. The input difference is $0a00000000a000a0$ and the output difference is $0a0000?000x000x0$. The probability of the inner part is $P = 2^{-54}$ and for a random permutation, the probability is 2^{-56}. This inner part uses a 2-round invariant property of CRAFT proposed in [GSS+20] when the tweak T satisfies the conditions $T[6] \oplus K[12] \in \{0x0, 0xa\}$ and $T[12] \oplus K[12] \in \{0x0, 0xa\}$ the difference $0xa$ propagates 2-round to $0xa$ with probability 1, detailed shows in Appendix C. This is a single-key model with some conditions in tweak T.

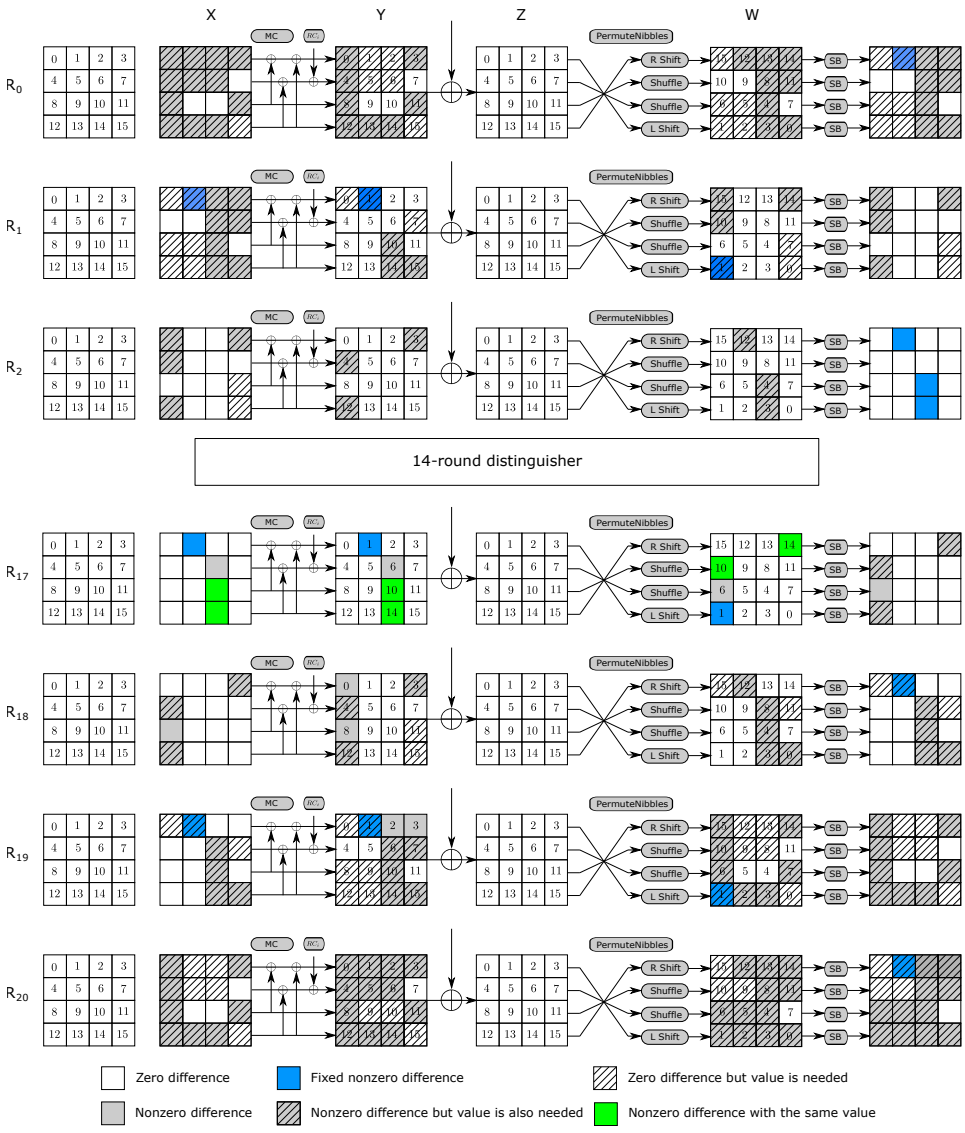

Fig. 9: 21-round Differential Attack on `CRAFT`

In this attack, we build 2^x structures for each value of $T[6]$ with the tweak T satisfying $T[6] = T[12]$. For each $T[6]$, we perform the following steps.

1. We build 2^x structures that $Y_0[0, 3, 4, 8, 11, 12, 13, 14]$ traverse all possible values while the other cells are fixed to some random constants. The data complexity is $D_0 = 2^{x+32}$. By using 2^{x+32} plaintexts, there are $N_0 = 2^x \cdot 2^{32\times 2-1} = 2^{x+63}$ pairs of plaintexts. The expected number of the remaining pairs of ciphertexts are $N_1 = N_0 \cdot 2^{-4\times 8} = 2^{x+31}$. The time complexity of this step is $T_0 = D = 2^{x+32}$.
2. For each of $N_1 = 2^{x+31}$ pairs, determine the key candidates and increase the corresponding counters. We pre-compute several tables as shown in Table 9, so that the time complexity of this step is $T_1 = 2^{36} \cdot N_1 = 2^{x+67}$ which is about $2^{x+67} \times \frac{1}{21} \times \frac{1}{16} \approx 2^{x+58.61}$ encryption.
3. Exhaustively the remaining keys.

Table 9: Precomputation tables for the 21-round attack on `CRAFT`.

Tables	Involved key	Filters	Remaining pairs
1	$K_0[12] = T[12]$	$\Delta X_1[1] = \Delta W_{19}[12] = a$	$2^{-8} \cdot N_1$
2	$K_0[0, 8, 11, 14]$	$\Delta Y_1[3, 7] = \Delta X_{19}[3, 7] = 0$	$2^{-8} \cdot N_1$
3	$K_0[13, 4, 3, 15]$	$\Delta Y_1[2, 6] = \Delta X_{19}[2] = 0$	$2^{-4} \cdot N_1$
4	$K_0[1, 2, 5, 6, 9, 10]$	-	$2^{20} \cdot N_1$
5	$K_1[1], K_1[9] \oplus K_1[13]$	-	$2^{28} \cdot N_1$
6	$K_1[6, 14, 10, 15]$	$\Delta X_2[0] = \Delta X_{18}[0] = 0, \Delta X_3[10] = a$	$2^{32} \cdot N_1$
7	$K_1[0, 7], K_1[8] \oplus K_1[12]$	$\Delta X_3[14] = a, \Delta W_{17}[3] = \Delta W_{17}[4]$	$2^{36} \cdot N_1$

Set $x = 24.99$, then the data, memory, and time complexities are $2^{4+24.99+32} = 2^{60.99}$, 2^{92} and $2^{87.60}$, respectively.

23-Round differential attack on `CRAFT`. Expending one round before and one round after the 21-round differential attack, we obtain a 23-round differential attack on `CRAFT`. As for the 21-round attack, we perform the following steps for each $T[6]$. For 23-round attack, $K_1[12] = T[6] = T[12]$.

1. We build 2^x structures that $Y_0[2, 3, 6, 7, 8, 9, 10, 12, 13, 14, 15]$ traverse all possible values while the other cells are fixed to some random constants. The data complexity is $D_0 = 2^{x+44}$. By using 2^{x+44} plaintexts, there are $N_0 = 2^x \cdot 2^{44\times 2-1} = 2^{x+87}$ pairs of plaintexts. The expected number of the remaining pairs of ciphertexts are $N_1 = N_0 \cdot 2^{-4\times 5} = 2^{x+67}$ pairs. The time complexity of this step is $T_0 = D = 2^{x+44}$.
2. For each of $N_1 = 2^{x+87}$ pairs, determine the key candidates and increase the corresponding counters. We pre-compute several tables as shown in Table 10, so that the time complexity of this step is $T_1 = 2^{40} \cdot N_1 = 2^{x+107}$ which is about $2^{x+107} \times \frac{1}{23} \times \frac{1}{16} \approx 2^{x+98.47}$ encryption.
3. Exhaustively the remaining keys.

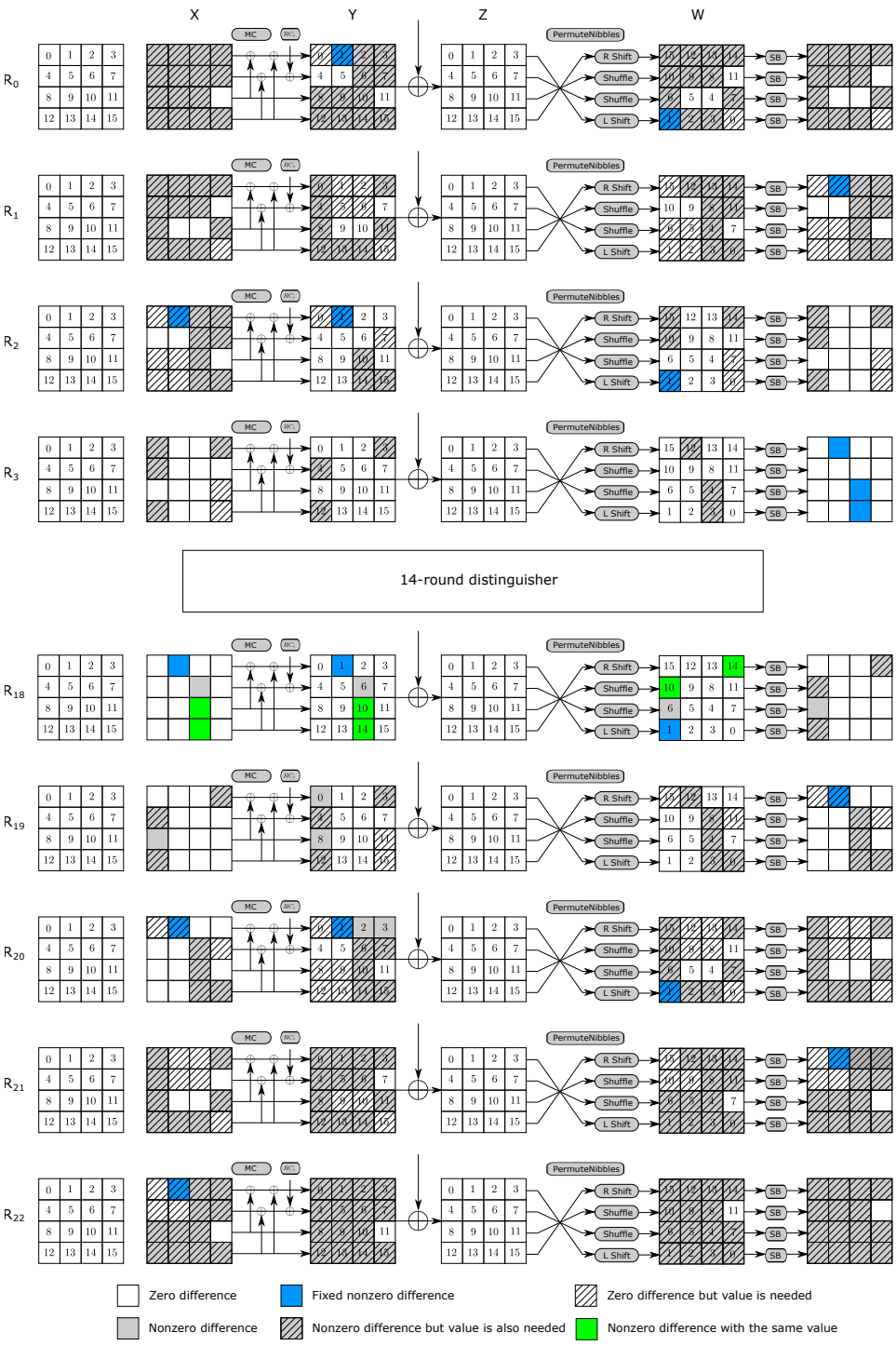

Fig. 10: 23-round Differential Attack on CRAFT

Set $x = 12.99$, then the data, memory, and time complexities are $2^{4+12.99+44} = 2^{60.99}$, 2^{120} and $2^{111.46}$, respectively.

Table 10: Precomputation tables for the 23-round attack on `CRAFT`.

Tables	Involved keys	Filters	Remaining pairs
1	$K_0[9, 2, 10, 14, 12]$	$\Delta X_{21}[1, 5] = \Delta Y_1[1, 5] = 0$	$2^4 \cdot N_1$
2	$K_0[8, 13, 3, 11, 15]$	$\Delta X_{21}[2, 6] = \Delta Y_1[2, 6] = 0$	$2^8 \cdot N_1$
3	$K_0[0, 1, 4, 5, 6, 7]$	-	$2^{32} \cdot N_1$
4	$K_1[0, 8, 11, 14]$	$\Delta Y_2[3, 7] = \Delta X_{20}[3, 7] = 0$	$2^{32} \cdot N_1$
5	$K_1[13, 4, 3, 15]$	$\Delta Y_2[2, 6] = \Delta X_{20}[2] = 0$	$2^{36} \cdot N_1$
6	$K_1[2, 5, 9, 10]$	$\Delta Y_3[1] = \Delta X_{19}[0] = 0, \Delta X_4[10] = a$	$2^{40} \cdot N_1$
7	$K_1[1, 6]$	$\Delta X_4[14] = a$, $\Delta W_{18}[3] = \Delta W_{18}[4]$	$2^{40} \cdot N_1$

The comparison with the previous attacks is presented in Table 11.

Table 11: Comparison of attacks on `CRAFT`

Rounds	Data	Memory	Time	Attack	Setting	Reference
23	2^{74}	2^{51}	2^{94}	D	WK(16-bit) & ST	[LR22]
26	2^{73}	2^{60}	2^{105}	D	WK(20-bit) & WT(12-bit)	[LR22]
20	$2^{62.89}$	2^{49}	$2^{120.43}$	ZC	SK & ST	[HSE23]
21	$2^{60.99}$	2^{100}	$2^{106.53}$	ID	SK & ST	[HSE23]
19	$2^{60.99}$	2^{68}	$2^{94.59}$	D	SK & WT(4-bit)	[GSS+20]
21	$2^{60.99}$	2^{92}	$2^{87.60}$	D	SK & WT(4-bit)	This
23	$2^{60.99}$	2^{120}	$2^{111.46}$	D	SK & WT(4-bit)	This

5 Conclusion

In this paper, we propose a one-step framework for finding the rectangle attacks with the purpose of reducing the overall complexities or attacking more rounds. Instead of extending the distinguisher forward and backward with probability 1, we propose to allow probabilistic propagations in the extended part. We treat the distinguisher and the extended part as a whole with a more flexible selection of propagation and involved keys to get better attacks. Moreover, we incorporate the holistic key recovery strategy into our one-step framework. Further, we introduce a technique, which is called the split-and-bunch technique, to reduce the time complexity. Applying our framework to `Deoxys-BC-384`, we obtain the first 15-round rectangle attack for `Deoxys-BC-384`, narrowing its security margin to only one round. Applying to `SKINNY` and `ForkSkinny`, we obtain a new rectangle attack with reduced data and time complexities. We also apply the main ideas

of the framework to differential attacks on CRAFT block cipher, which achieves 2 more rounds than the previous best attacks.

Further works. In this paper, we only apply the new framework to rectangle attacks and differential attacks. It would be a potential future work to explore the application of these ideas to other attacks.

Acknowledgement. The authors would like to thank anonymous reviewers for their helpful comments and suggestions. The work of this paper was supported by the National Key Research and Development Program (No. 2018YFA0704704) and the National Natural Science Foundation of China (Grants 62202460, 62372213, 62132008, 62022036). Jian Weng is supported by the National Natural Science Foundation of China under Grant Nos. 61825203, 62332007, and U22B2028, Science and Technology Major Project of Tibetan Autonomous Region of China under Grant No. XZ202201ZD0006G, National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education.

References

ABK98. Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A proposal for the advanced encryption standard. *NIST AES Proposal*, 174:1–23, 1998.

ALP+19. Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. Forkcipher: A new primitive for authenticated encryption of very short messages. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II*, volume 11922 of *Lecture Notes in Computer Science*, pages 153–182. Springer, 2019.

BDD+23. Christina Boura, Nicolas David, Patrick Derbez, Gregor Leander, and María Naya-Plasencia. Differential meet-in-the-middle cryptanalysis. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 240–272. Springer, 2023.

BDK01. Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack—rectangling the Serpent. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 340–357. Springer, 2001.

BDK02. Eli Biham, Orr Dunkelman, and Nathan Keller. New results on boomerang and rectangle attacks. In *International Workshop on Fast Software Encryption*, pages 1–16. Springer, 2002.

BJK+16. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In

Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

BK09. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *International conference on the theory and application of cryptology and information security*, pages 1–18. Springer, 2009.

BL23. Augustin Bariant and Gaëtan Leurent. Truncated boomerang attacks and application to AES-based ciphers. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 3–35. Springer, 2023.

BLMR19. Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. *IACR Trans. Symmetric Cryptol.*, 2019(1):5–45, 2019.

BS91. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.

CHP+18. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: a new cryptanalysis tool. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 683–714. Springer, 2018.

DDV21. Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs - application to SKINNY. *IACR Cryptol. ePrint Arch.*, page 20, 2021.

DEFN22. Patrick Derbez, Marie Euler, Pierre-Alain Fouque, and Phuong Hoa Nguyen. Revisiting related-key boomerang attacks on AES using computer-aided tool. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III*, volume 13793 of *Lecture Notes in Computer Science*, pages 68–88. Springer, 2022.

Din14. Itai Dinur. Improved differential cryptanalysis of round-reduced SPECK. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2014.

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 *Annual cryptology conference*, pages 393–410. Springer, 2010.

DKS14. Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. *Journal of cryptology*, 27(4):824–849, 2014.

DQSW21. Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. *IACR Cryptol. ePrint Arch.*, page 856, 2021.

DQSW22. Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology -*

EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III, volume 13277 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2022.

GSS+20. Hao Guo, Siwei Sun, Danping Shi, Ling Sun, Yao Sun, Lei Hu, and Meiqin Wang. Differential attacks on CRAFT exploiting the involutory s-boxes and tweak additions. *IACR Trans. Symmetric Cryptol.*, 2020(3):119–151, 2020.

HBS21. Hosein Hadipour, Nasour Bagheri, and Ling Song. Improved rectangle attacks on SKINNY and CRAFT. *IACR Trans. Symmetric Cryptol.*, 2021(2):140–198, 2021.

HSE23. Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 128–157. Springer, 2023.

JNP14. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2014.

JNPS16. Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. Deoxys v1. 41. *Submitted to CAESAR*, 124, 2016.

KKS00. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round MARS and Serpent. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2000.

Knu95. Lars R Knudsen. Truncated and higher order differentials. In *Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings 2*, pages 196–211. Springer, 1995.

LGS17. Guozhen Liu, Mohona Ghosh, and Ling Song. Security analysis of SKINNY under related-tweakey settings. *IACR Trans. Symmetric Cryptol.*, 2017(3):37–72, 2017.

LR22. Gregor Leander and Shahram Rasoolzadeh. Weak tweak-keys for the CRAFT block cipher. *IACR Trans. Symmetric Cryptol.*, 2022(1):38–63, 2022.

Mat94. Mitsuru Matsui. On correlation between the order of S-boxes and the strength of DES. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1994.

MP13. Nicky Mouha and Bart Preneel. A proof that the ARX cipher Salsa20 is secure against differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 328, 2013.

Mur11. Sean Murphy. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory*, 57(4):2517–2521, 2011.

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.

SHW+14. Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

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.

SSD+18. Danping Shi, Siwei Sun, Patrick Derbez, Yosuke Todo, Bing Sun, and Lei Hu. Programming the demirci-selçuk meet-in-the-middle attack with constraints. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2018.

SYL23. Ling Song, Qianqian Yang, and Huimin Liu. Revisiting the differential meet-in-the-middle cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 1302, 2023.

SZY+22. Ling Song, Nana Zhang, Qianqian Yang, Danping Shi, Jiahao Zhao, Lei Hu, and Jian Weng. Optimizing rectangle attacks: A unified and generic framework for key recovery. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 410–440. Springer, 2022.

Wag99. David A. Wagner. The boomerang attack. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999.

ZDC+21. Rui Zong, Xiaoyang Dong, Huaifeng Chen, Yiyuan Luo, Si Wang, and Zheng Li. Towards key-recovery-attack friendly distinguishers: Application to GIFT-128. *IACR Trans. Symmetric Cryptol.*, 2021(1):156–184, 2021.

ZDJ19. Boxin Zhao, Xiaoyang Dong, and Keting Jia. New related-tweakey boomerang and rectangle attacks on Deoxys-BC including BDT effect. *IACR Transactions on Symmetric Cryptology*, pages 121–151, 2019.

ZDM+20. Boxin Zhao, Xiaoyang Dong, Willi Meier, Keting Jia, and Gaoli Wang. Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT. *Designs, Codes and Cryptography*, 88(6):1103–1126, 2020.

A Unified Key Recovery Algorithms

A.1 Unified Key Recovery Algorithm for Differential Attacks

This subsection recalls the unified key recovery algorithm from [SYL23]. Suppose E_b and E_f are added around the distinguisher where E_b and E_f can be identity

functions. Suppose the probability of the distinguisher is $P = 2^{-p}$. Then the unified key recovery algorithm proceeds as follows with k'_b, k'_f guessed at first.

Unified key recovery algorithm for differential attacks

1. $S \leftarrow 2^{p-r_b+1}$ structures, each of 2^{r_b} plaintexts
2. Guess k'_b and k'_f, $0 \le m'_b \le m_b$, $0 \le m'_f \le m_f$:
3. For $S_* \in S$:
4. Do partial encryption and decryption for elements in S_* if $m'_b + m'_f \neq 0$.
5. // Additional r'_b, r'_f filtering bits are obtained, respectively.
6. Store the data into a hash table indexed by the filtering bits.
7. Get $2^{2r_b-1+r_f-n-r'_b-r'_f}$ pairs having fixed difference on the filtering bits.
8. For each of such pairs:
9. Extract $2^{m^*_b-r^*_b}$ candidates for k^*_b, under which α can be reached.
10. Extract $2^{m^*_f-r^*_f}$ candidates for k^*_f, under which δ can be reached.
11. Update the key counters.
12. Repeat Step 1∼11 for s times where s is the expected number of right pairs.

A.2 Unified Key Recovery Algorithm for Rectangle Attacks

In the following, we recall the unified algorithm for the rectangle key recovery attack from [SZY+22]. This algorithm works for any number of guessed key bits and uses the traditional deterministic extensions.

Like most key recovery algorithms, this algorithm also employs the counting method. Namely, one sets counters for the involved subkey bits and searches for the correct one among the subkey candidates with a large number of suggestions. Suppose m'_b-bit k'_b and m'_f-bit k'_f are to be guessed. For these guessed subkey bits, one may or may not set counters for them. To enjoy such flexibility, set counters for t bits of the guessed subkey bits, $0 \le t \le m'_b + m'_f$. Then the specific steps of the algorithm are as follows.

1. Collect and store y structures of 2^{r_b} plaintexts. Hence, the data complexity is $D = y \cdot 2^{r_b}$. The time and memory complexities of this step are also D.
2. Split $(m'_b + m'_f)$-bit $k'_b \| k'_f$ into two parts: $G_L \| G_R$ where G_L has t bits.
3. Guess G_R:
   (a) Initialized a list of key counters for G_L and the unguessed key bits of k_b, k_f. The memory complexity in this step is $2^{t+m_b+m_f-m'_b-m'_f}$.
   (b) Guess the t-bit G_L:
      i. For each data (P_1, C_1), partially encrypt P_1 and partially decrypt C_1 under the guessed subkey bits. Let $P_1^* = Enc_{k'_b}(P_1)$ and $C_1^* = Dec_{k'_f}(C_1)$. For each structure, we will get $2^{r'_b}$ sub-structures, each of which includes $2^{r_b-r'_b} = 2^{r^*_b}$ plaintexts which take all possible values for the active bits. In other words, there are $y^* = y \cdot 2^{r'_b}$ structures of $2^{r^*_b}$ plaintexts. The time complexity of this step is D.

ii. Let $2^{-\mu} = D \cdot 2^{-n}$. If $r_b^* \leq r_f^* - \mu$[7], it turns to step (A); else if $r_b^* > r_f^* - \mu$, it turns to step (D).

A. Insert all the obtained (P_1^*, C_1^*) into a hash table according to $n - r_b^*$ bits of P_1^*. Then construct a set as $S = \{(P_1^*, C_1^*, P_2^*, C_2^*) : P_1^*$ and P_2^* have difference only in r_b^* bits$\}$. The size of S is $y \cdot 2^{r_b'} \cdot 2^{2(r_b - r_b')-1} = D \cdot 2^{r_b^*-1}$. Hence, the time and memory complexities of this step are both $D \cdot 2^{r_b^*-1}$.

B. Insert S into a hash table by $n - (r_f - r_f') = n - r_f^*$ inactive bits of C_1^* and $n - (r_f - r_f') = n - r_f^*$ inactive bits of C_2^*.

C. For each $2(n - r_f^*)$-bit index, we pick two distinct $(P_1^*, C_1^*, P_2^*, C_2^*)$, $(P_3^*, C_3^*, P_4^*, C_4^*)$ to generate the quartet. We will get

$$2 \cdot \binom{\frac{|S|}{2^{2(n-r_f^*)}}}{2} \cdot 2^{2(n-r_f^*)} = D^2 \cdot 2^{2r_b^*} \cdot 2^{2r_f^*} \cdot 2^{-2n-2}$$

quartets. Then go to step (iii).

D. Insert all the obtained (P_1^*, C_1^*) into a hash table according to $n - r_f^*$ bits of C_1^*. Then construct a set as $S = \{(P_1^*, C_1^*, P_3^*, C_3^*) : C_1^*$ and C_3^* are colliding in $n - r_f^*$ bits$\}$. The size of S is $D^2 \cdot 2^{r_f - r_f' - n - 1} = D \cdot 2^{r_f^* - 1 - \mu}$. Hence, the time and memory complexities of this step are both $D \cdot 2^{r_f^* - 1 - \mu}$.

E. Insert S into a hash table by $n - r_b^*$ inactive bits of P_1^* and $n - r_b^*$ inactive bits of P_3^*.

F. There are at most $2^{2(n - r_b^* - \mu)}$ possible values for the $2(n - r_b^*)$-bit index. For each index, we pick two distinct entries $(P_1^*, C_1^*, P_3^*, C_3^*)$, $(P_2^*, C_2^*, P_4^*, C_4^*)$ to generate the quartet. We will get

$$2 \cdot \binom{\frac{|S|}{2^{2(n-r_b^*-\mu)}}}{2} \cdot 2^{2(n-r_b^*-\mu)} = D^2 \cdot 2^{2r_b^*} \cdot 2^{2r_f^*} \cdot 2^{-2n-2}$$

quartets.

iii. Determine the key candidates involved in E_b and E_f and increase the corresponding counters. Denote the time complexity for processing one quartet as ϵ. Then the time complexity in this step is $D^2 \cdot 2^{2r_b^*} \cdot 2^{2r_f^*} \cdot 2^{-2n-2} \cdot \epsilon$.

(c) Select the top $2^{t + m_b + m_f - m_b' - m_f' - h}$ hits in the counters to be the candidates, which delivers a h-bit or higher advantage, where $0 < h \leq t + m_b + m_f - m_b' - m_f'$.

(d) Guess the remaining $k - m_b - m_f$ unknown key bits according to the key schedule algorithm and exhaustively search over them to recover the correct key. The time complexity of this step is $2^{k + t - m_b' - m_f' - h}$.

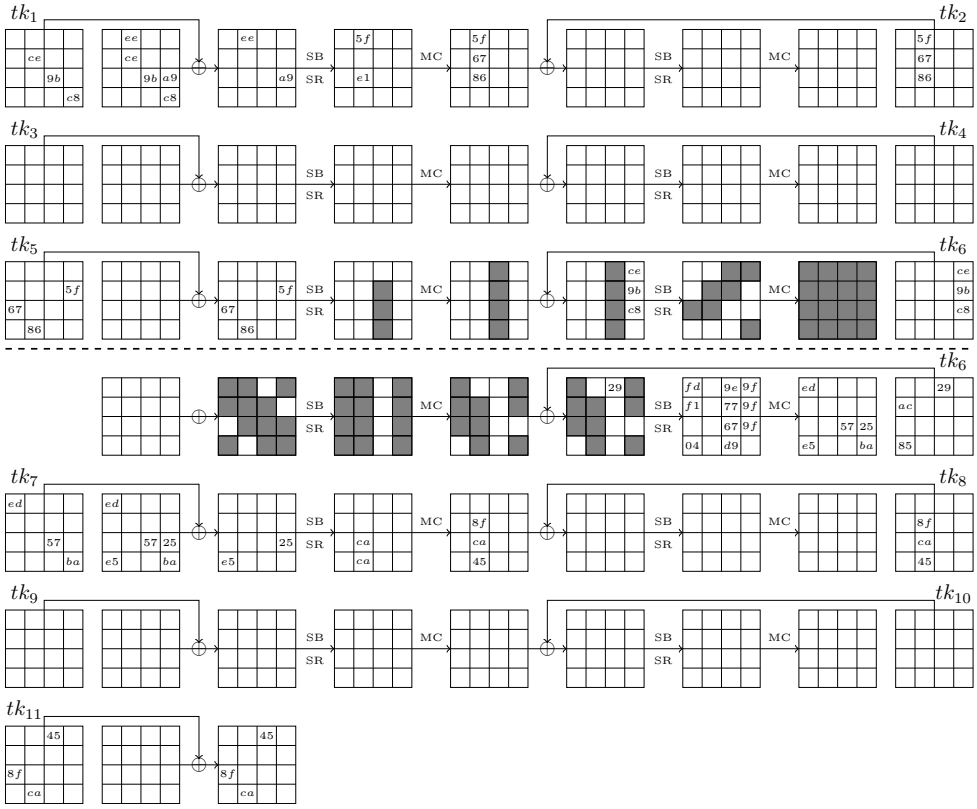

Fig. 11: 10-round Rectangle distinguisher of `Deoxys-BC-384`

B The Rectangle Distinguisher of Deoxys-BC-384

C The property proposed in [GSS+20]

Property 1. Let S be the involutory S-box of CRAFT, and $\tau_k : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ be a function mapping x to $S(S(x)\oplus k)$, where x and $k \in \mathbb{F}_2^4$. Then $\tau_0(x\oplus\delta)\oplus\tau_0(x) = \delta$, and $\tau_{0xa}(x\oplus 0xa)\oplus\tau_{0xa}(x) = 0xa$ that is, τ_k preserves the input difference δ with probability 1 when $k = 0$ and preserves the input difference $0xa$ with probability 1 when $k = 0xa$.

D Application to ForkSkinny

Specification of ForkSkinny. ForkSkinny is designed by Andreeva *et al.* [ALP+19], which is the internal primitive of ForkAE, a 2nd round candidate in the NIST lightweight authenticated encryption standardization project. It is a forked variant of SKINNY. The encryption of ForkSkinny is split into two steps. The first step applies the round function of SKINNY $\mathtt{R_{init}}$ times. Then it forks the state into $\mathtt{SKINNY_0}$ with $\mathtt{R_I}$ rounds and $\mathtt{SKINNY_1}$ with $\mathtt{R_{II}}$ rounds. The subkeys are generated by extending the tweakey schedule to produce $\mathtt{R_{init}} + \mathtt{R_I} + \mathtt{R_{II}}$ subkeys, detailed refers to [ALP+19].

28-Round rectangle attack on ForkSkinny-128-256 with 256-bit key. With our framework to ForkSkinny, we get a new 28-round rectangle attack ($\mathtt{R_{init}} = 20, \mathtt{R_I} = 27, \mathtt{R_{II}} = 8$) to reduce the time complexity and decrease the data complexity of the previous attack proposed in [DQSW22] with 256-bit key. The probability of the rectangle attack is $P^2 = 2^{-113.77}$. The other parameters are $n = 128, k = 256, r_b = 8\times 8 = 64, m_b = 8\times 8 = 64, r_f = 12\times 8 = 96$ and $m_f = 17\times 8 = 136$. The best-guessing parameters are $m'_b = 64, r'_b = 64, m'_f = 24$ and $r'_f = 24$. The complexities of the 28-round rectangle attack are as follows:

- The data complexity is $D_R = 4\cdot y\cdot 2^{r_b} = \sqrt{s}\cdot 2^{n/2+2}/P = \sqrt{s}\cdot 2^{122.885}$.
- The memory complexity is $M_R = D_R + D\cdot 2^{r^*_b} + 2^{t+m_b+m_f-m'_b-m'_f} = \sqrt{s}\cdot 2^{122.885} + \sqrt{s}\cdot 2^{120.885} + 2^{112+t}$.
- The time complexity $T_1 = 2^{m'_b+m'_f}\cdot D_R = \sqrt{s}\cdot 2^{11\times 8+122.885} = \sqrt{s}\cdot 2^{210.885}$;
- $T_2 = 2^{m'_b+m'_f}\cdot D\cdot 2^{r_b-r'_b} = \sqrt{s}\cdot 2^{11\times 8+120.885} = \sqrt{s}\cdot 2^{208.885}$;
- $T_3 = 2^{m'_b+m'_f}\cdot D^2\cdot 2^{2r^*_b+2r^*_f-2n}\cdot\epsilon = s\cdot 2^{11\times 8+120.885\times 2+2\times 0+2\times 72-2\times 128}\cdot\epsilon = s\cdot 2^{217.77}\cdot\epsilon$;
- $T_4 = 2^{256-h}, h < 136 + t$.

We pre-compute several tables as illustrated in Table 12, so that the ϵ is equivalent to about 1 memory access which is around $1\times\frac{1}{28}\times\frac{1}{16}\approx 2^{-8.80}$. Set $s = 4, t = 0$, the data, memory and time complexities are $2^{123.89}$, $2^{123.89}$ and $2^{212.89}$.

The comparison with the previous rectangle attacks is presented in Table 13.

[7] The number of filters for plaintext pairs is $n - r^*_b - \mu$ while it is $n - r^*_f$ for ciphertext pairs.

Table 12: Precomputation tables for the 28-round attack on `ForkSkinny` ($STK_{25}[2,6]$ can be deduced by $STK_0[1,2]$ and $STK_{27}[4,5]$, $Q = s \cdot 2^{217.77}$)

Tables	Involved key	Filters	Remaining quartets
1	$STK_{27}[2]$	$\Delta W_{26}[14] = 0$	$Q \cdot 2^{-8}$
2	$STK_{27}[6], STK_{26}[5]$	$\Delta X_{26}[5] = 0x5e$	$Q \cdot 2^{-8}$
3	$STK_{27}[3]$	$\Delta W_{25}[4] = 0$	$Q \cdot 2^{-16}$
4	$STK_{26}[0]$	$\Delta W_{25}[12] = 0$	$Q \cdot 2^{-24}$
5	$STK_{27}[1,7], STK_{26}[2,6]$	$\Delta W_{25}[10,14] = 0$	$Q \cdot 2^{-24}$
6	$STK_{26}[1,4], (STK_{25}[2])$	$\Delta W_{24}[6,14] = 0$	$Q \cdot 2^{-40}$
7	$STK_{26}[7], (STK_{25}[6])$	$\Delta X_{24}[8] = 0x35$	$Q \cdot 2^{-48}$

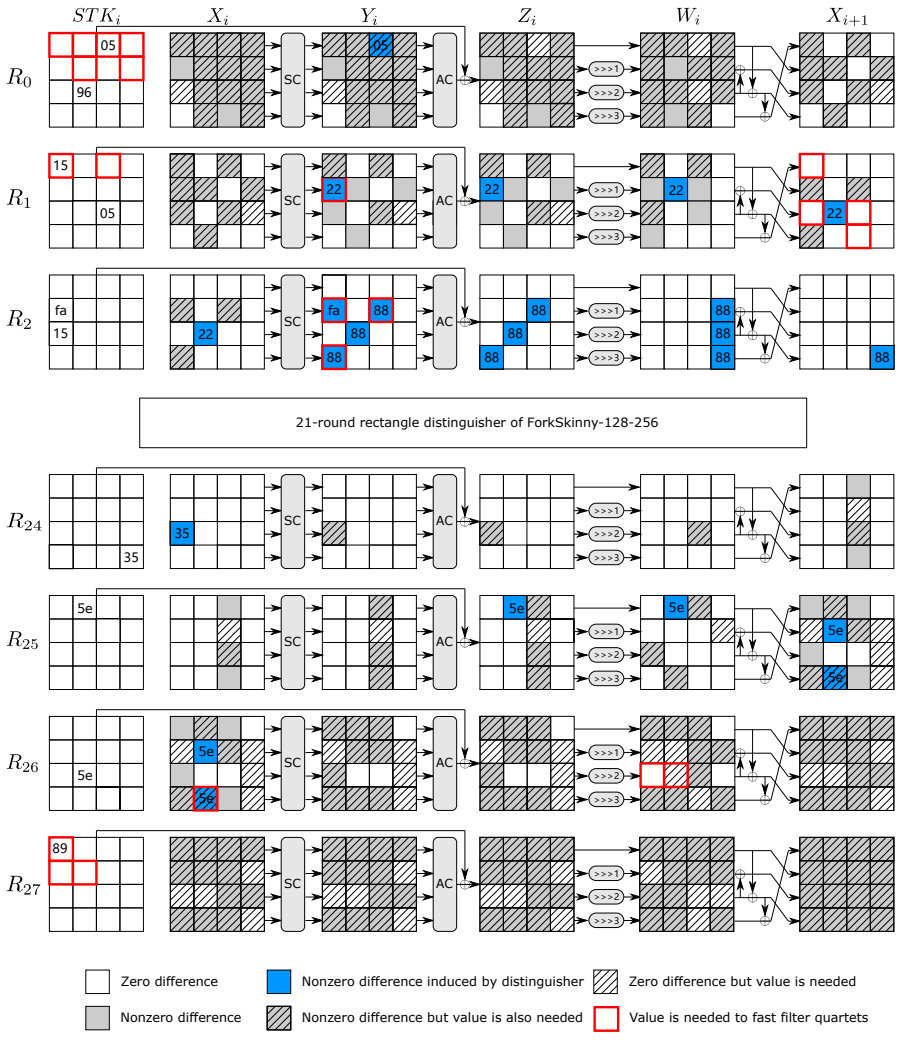

Fig. 12: 28-round Rectangle Attack on `ForkSkinny`-128-256

Table 13: Comparisons of key recovery attacks on `ForkSkinny`-128-256

P^2	m_b, m_f	m'_b, m'_f	Data	Memory	Time	Reference
$2^{-105.77}$	$80, 176$	$80, 24$	$2^{118.88}$	2^{136}	$2^{246.98}$	[QDW+21]
$2^{-105.77}$	$80, 176$	$64, 24$	$2^{118.88}$	$2^{118,88}$	$2^{224.76}$	[DQSW22]
$2^{-113.77}$	64,176	64,24	$2^{124.89}$	$2^{124.89}$	$2^{212.89}$	This