Key Guessing Strategies for Linear Key-Schedule Algorithms in Rectangle Attacks [1]

Xiaoyang Dong[1], Lingyue Qin[1(✉)], Siwei Sun[2,3], and Xiaoyun Wang[1,4,5]

[1] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
{xiaoyangdong,qinly,xiaoyunwang}@tsinghua.edu.cn
[2] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
[3] State Key Laboratory of Cryptology, Beijing, China
sunsiwei@ucas.ac.cn
[4] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[5] School of Cyber Science and Technology, Shandong University, Qingdao, China

Abstract. When generating quartets for the rectangle attacks on ciphers with linear key-schedule, we find the right quartets which may suggest key candidates have to satisfy some nonlinear relations. However, some quartets generated always violate these relations, so that they will never suggest any key candidates. Inspired by previous rectangle frameworks, we find that guessing certain key cells before generating quartets may reduce the number of invalid quartets. However, guessing a lot of key cells at once may lose the benefit from the early abort technique, which may lead to a higher overall complexity. To get better tradeoff, we build a new rectangle framework on ciphers with linear key-schedule with the purpose of reducing overall complexity or attacking more rounds.

In the tradeoff model, there are many parameters affecting the overall complexity, especially for the choices of the number and positions of key guessing cells before generating quartets. To identify optimal parameters, we build a uniform automatic tool on SKINNY as an example, which includes the optimal rectangle distinguishers for key-recovery phase, the number and positions of key guessing cells before generating quartets, the size of key counters to build that affecting the exhaustive search step, etc. Based on the automatic tool, we identify a 32-round key-recovery attack on SKINNY-128-384 in the related-key setting, which extends the best previous attack by 2 rounds. For other versions with n-$2n$ or n-$3n$, we also achieve one more round than before. In addition, using the previous rectangle distinguishers, we achieve better attacks on round-reduced ForkSkinny, Deoxys-BC-384 and GIFT-64. At last, we discuss the conversion of our rectangle framework from related-key setting into single-key setting and give new single-key rectangle attack on 10-round Serpent.

Keywords: Rectangle · Automated Key-recovery · SKINNY · ForkSkinny · Deoxys-BC · GIFT

[1] The full version of the paper is available at https://ia.cr/2021/856.

1 Introduction

The boomerang attack [60] proposed by Wagner, is an adaptive chosen plaintext and ciphertext attack derived from differential cryptanalysis [15]. Wagner constructed the boomerang distinguisher on E_d by splitting the encryption function into two parts $E_d = E_1 \circ E_0$ as shown in Figure 1, where two differentials $\alpha \xrightarrow{E_0} \beta$ with probability p and $\gamma \xrightarrow{E_1} \delta$ with probability q are combined into a boomerang distinguisher. The probability of a boomerang distinguisher is estimated by:

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

The adaptive chosen plaintext and ciphertext of boomerang attack can be converted into a chosen-plaintext attack that is known as amplified boomerang attack [45] or rectangle attack [13]. In rectangle attack, only α and δ are fixed and the internal differences β and γ can be arbitrary values as long as $\beta \neq \gamma$. Hence, the probability would be increased to $2^{-n}\hat{p}^2\hat{q}^2$, where

$$\hat{p} = \sqrt{\sum\nolimits_{\beta_i} Pr^2(\alpha \rightarrow \beta_i)} \text{ and } \hat{q} = \sqrt{\sum\nolimits_{\gamma_j} Pr^2(\gamma_j \rightarrow \delta)}. \quad (2)$$

The boomerang attack and rectangle attack have been successfully applied to numerous block ciphers, including `Serpent` [13,12], `AES` [18,14], `IDEA` [11], `KASUMI` [38], `Deoxys-BC` [29], etc. Recently, a new variant of boomerang attack was developed and applied to `AES`, named as retracing boomerang attack [35]. There are two steps when applying the boomerang and rectangle attack, i.e., building distinguishers and performing key-recovery attacks. In building distinguishers, Murphy [51] pointed out that two independently chosen differentials for the boomerang can be incompatible. He also showed that the dependence between two differentials of the boomerang may lead to larger probability, which is also discovered by Biryukov *et al.* [16]. To further explore the dependence and increase the probability of boomerang, Biryukov and Khovratovich [18] introduced the *boomerang switch* technique including the *ladder switch* and *S-box switch*. Then, those techniques were generalized and formalized by Dunkelman *et al.* [37,38] as the *sandwich attack*. Recently, Cid *et al.* [28] introduced the boomerang connectivity table (BCT) to clarify the probability around the boundary of boomerang and compute more accurately. Later, various improvements or further studies [23,5,57,61,30,21,22] on BCT technique enriched boomerang attacks.

Given a distinguisher, we usually need more complicated key-recovery algorithms to identify the right quartets [45,13] when performing rectangle attack than boomerang attack. Till now, a series of generalized key-recovery algorithms [13,12,14] for the rectangle attacks are introduced. In this paper, we focus on further exploration on the generalized rectangle attacks. Undoubtedly, generalizing the attack algorithms is very important in the development of cryptanalytic tools, such as the generalizations of the impossible differential attacks [25,24], linear attacks [39], invariant attacks [9], meet-in-the-middle attacks [27,32], etc.

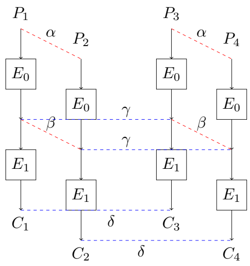

Fig. 1: Boomerang attack.

Our Contributions. When performing the rectangle attacks, we usually add several rounds before and after the rectangle distinguisher. Then, the input and output differences (α, δ) of the rectangle distinguisher propagate to certain truncated form (α', δ') in the plaintext and ciphertext. Similar with the differential attack, in rectangle attack we first collect data and generate quartets whose plaintext difference and ciphertext difference meet (α', δ'). Then the early-abort technique [49] is applied to determine key candidates for each quartet. However, for ciphers with linear key schedule, we find that many quartets meet (α', δ') never suggest any key candidates. In further study, we find the right quartets that suggest key candidates have to meet certain nonlinear relations. However, many quartets meeting (α', δ') always violate those nonlinear relations for all the key guessing, and thereby never suggest any key candidates. This feature is peculiar for rectangle attack on ciphers with linear key schedule, and it rarely appears in other differential-like attack.

Inspired from the previous rectangle attacks [13,12,62], we find that guessing certain key cells before generating quartets may avoid many invalid quartets in advance. However, guessing a lot of key cells as a whole may lose the advantage of early-abort technique [49], which may lead to higher complexity. In addition, we have to take the exhaustive search step into consideration. Hence, to get a tradeoff between so many factors affecting the complexity, we introduce a new generalized rectangle attack framework on ciphers with linear key schedule.

When evaluating dedicated cipher with the tradeoff framework, we have to identify many attack parameters, such as finding an optimal rectangle distinguisher for our new key-recovery attack framework, determining the number and positions of guessed key cells before generating quartets, as well as the size of key counters, etc. Hence, in order to launch the optimal key-recovery attacks with our tradeoff model, we build a uniform automatic tool for `SKINNY` as an example, which is based on a series of automatic tools [30,41,52] on `SKINNY` proposed recently, to determine a set of optimal parameters affecting the attack complexity or the number of attacked rounds. Note that in the field of automatic cryptanalysis, there are many works focusing on searching for distinguishers [19,20,50,59,54,46,17], but only a few works [31,56,52] deal with the uniform automatic models that take the distinguisher and key-recovery as a whole optimization model. Thanks to our uniform automatic model, we identify a 32-round key-recovery attack on SKINNY-128-384, which attacks two more rounds than the best previous attacks [52,41]. In addition, for other versions of SKINNY with n-$2n$ or n-$3n$, one more round is achieved.

Table 1: Summary of the cryptanalytic results.

Version	Rounds	Data	Time	Memory	Approach	Setting	Ref.
SKINNY							
64-128	22	$2^{63.5}$	$2^{110.9}$	$2^{63.5}$	Rectangle	RK	[47]
	23	$2^{62.47}$	$2^{125.91}$	2^{124}	ID	RK	[47]
	23	$2^{62.47}$	2^{124}	$2^{77.47}$	ID	RK	[53]
	23	$2^{71.4}$	2^{79}	$2^{64.0}$	ID	RK	[3]
	23	$2^{60.54}$	$2^{120.7}$	$2^{60.9}$	Rectangle	RK	[41]
	24	$2^{61.67}$	$2^{96.83}$	2^{84}	Rectangle	RK	[52]
	25	$2^{61.67}$	$2^{118.43}$	$2^{64.26}$	Rectangle	RK	Full Ver. [33]
64-192	27	$2^{63.5}$	$2^{165.5}$	2^{80}	Rectangle	RK	[47]
	29	$2^{62.92}$	$2^{181.7}$	2^{80}	Rectangle	RK	[41]
	30	$2^{62.87}$	$2^{163.11}$	$2^{68.05}$	Rectangle	RK	[52]
	31	$2^{62.78}$	$2^{182.07}$	$2^{62.79}$	Rectangle	RK	Full Ver. [33]
128-256	22	2^{127}	$2^{235.6}$	2^{127}	Rectangle	RK	[47]
	23	$2^{124.47}$	$2^{251.47}$	2^{248}	ID	RK	[47]
	23	$2^{124.41}$	$2^{243.41}$	$2^{155.41}$	ID	RK	[53]
	24	$2^{125.21}$	$2^{209.85}$	$2^{125.54}$	Rectangle	RK	[41]
	25	$2^{124.48}$	$2^{226.38}$	2^{168}	Rectangle	RK	[52]
	25	$2^{120.25}$	$2^{193.91}$	2^{136}	Rectangle	RK	Full Ver. [33]
	26	$2^{126.53}$	$2^{254.4}$	$2^{128.44}$	Rectangle	RK	Full Ver. [33]
128-384	27	2^{123}	2^{331}	2^{155}	Rectangle	RK	[47]
	28	2^{122}	$2^{315.25}$	$2^{122.32}$	Rectangle	RK	[64]
	30	$2^{125.29}$	$2^{361.68}$	$2^{125.8}$	Rectangle	RK	[41]
	30	2^{122}	$2^{341.11}$	$2^{128.02}$	Rectangle	RK	[52]
	32	$2^{123.54}$	$2^{354.99}$	$2^{123.54}$	Rectangle	RK	Sect. 5.1
ForkSkinny							
128-256 (256-bit key)	26	2^{125}	$2^{254.6}$	2^{160}	ID	RK	[6]
	26	2^{127}	$2^{250.3}$	2^{160}	ID	RK	[6]
	28	$2^{118.88}$	$2^{246.98}$	2^{136}	Rectangle	RK	[52]
	28	$2^{118.88}$	$2^{224.76}$	$2^{118.88}$	Rectangle	RK	Full Ver. [33]
Deoxys-BC							
128-384	13	2^{127}	2^{270}	2^{144}	Rectangle	RK	[29]
	14	2^{127}	$2^{286.2}$	2^{136}	Rectangle	RK	[62]
	14	$2^{125.2}$	$2^{282.7}$	2^{136}	Rectangle	RK	[63]
	14	$2^{125.2}$	2^{260}	2^{140}	Rectangle	RK	Full Ver. [33]
GIFT							
64-128	25	$2^{63.78}$	$2^{120.92}$	$2^{64.1}$	Rectangle	RK	[44]
	26	$2^{60.96}$	$2^{123.23}$	$2^{102.86}$	Differential	RK	[58]
	26	$2^{63.78}$	$2^{122.78}$	$2^{63.78}$	Rectangle	RK	Full Ver. [33]

As the second application, we perform our new key-recovery framework on round-reduced ForkSkinny [1,2], Deoxys-BC-384 [43] and GIFT-64 [4] with some previous proposed distinguishers. All the attacks achieve better complexities than before, which also proves the efficiency of our tradeoff model. At last, we discuss the conversion of our attack framework from related-key setting to single-key setting. Since our related-key attack framework is on ciphers with linear key-schedule, it is trivial to convert it into a single-key attack by assigning the key difference as zero. We then apply the new single-key framework to the 10-round `Serpent`[6] and achieve better complexity than the previous rectangle attack [12]. We summarize our main results in Table 1.

2 Generalized Key-Recovery Algorithms for the Rectangle Attacks

There have been several key-recovery frameworks of rectangle attacks [13,12,14] introduced before. We briefly recall them with the symbols from [12]. Let E be a cipher which is described as a cascade $E = E_f \circ E_d \circ E_b$ as shown in Figure 2. The probability of the N_d-round rectangle distinguisher on E_d is given by Eq. (2). E_d is surrounded by the N_b-round E_b and N_f-round E_f. Then the difference α of the distinguisher propagates to a truncated differential form denoted as α' by E_b^{-1}, and δ propagates to δ' by E_f. Denote the number of active bits of the plaintext and ciphertext as r_b and r_f. Denote the subset of subkey bits which is involved in E_b as k_b, which affects the difference of the plaintexts by decrypting the pairs of internal states with difference α. Then denote $m_b = |k_b|$. Let k_f be the subset of subkey bits involved in E_f and $m_f = |k_f|$.

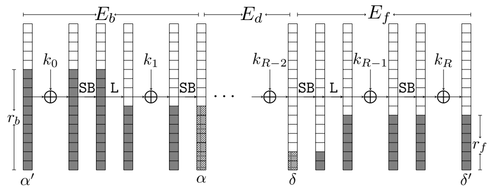

Fig. 2: Framework of rectangle attack on E.

Related-key boomerang and rectangle attacks were proposed by Biham *et al.* in [14]. Assuming one has a related-key differential $\alpha \to \beta$ over E_0 under a key difference ΔK with probability $\hat{p}$ and another related-key differential $\gamma \to \delta$ over E_1 under a key difference ∇K with probability $\hat{q}$. If the master key K_1 is known, the other three keys are all determined, where $K_2 = K_1 \oplus \Delta K, K_3 = K_1 \oplus \nabla K$ and $K_4 = K_1 \oplus \Delta K \oplus \nabla K$. A typical example of the successful application of the boomerang attack is the best known related-key attack on the full versions of `AES-192` and `AES-256`, presented by Biryukov and Khovratovich [18].

[6]The example attack only wants to prove the efficiency of our model in single-key setting. There are better attacks on `Serpent` achieved by differential-linear cryptanalysis [34,48].

As shown by Biham, Dunkelman and Keller [11], when the key schedule is linear, the related-key rectangle attack is similar to the single-key rectangle framework. Different from non-linear key schedule, the differences between the subkeys of K_1, K_2, K_3 and K_4 are all determined in each round for linear key schedule. Hence, if we guess parts of the subkeys of K_1, all the corresponding parts of subkeys of K_2, K_3 and K_4 are determined by xoring the differences between the subkeys. In this paper, we focus on the rectangle attacks on ciphers with linear key schedule and list the previous frameworks below. In addition, we give a comprasion of different frameworks in Section 3.2 and Section 6.

2.1 `Attack I`: Biham-Dunkelman-Keller's Attack

At EUROCRYPT 2001, Biham, Dunkelman and Keller introduced the rectangle attack [13] and applied it to the single-key attack on `Serpent` [10]. We trivially convert it to a related-key model with linear key schedule:

1. Create and store y structures of 2^{r_b} plaintexts each, and query the 2^{r_b} plaintexts under K_1, K_2, K_3 and K_4 for each structure.
2. Initialize the key counters for the $(m_b + m_f)$-bit subkey involved in E_b and E_f. For each $(m_b + m_f)$-bit subkey, do:
   (a) Partially encrypt plaintext P_1 under K_1 to the position of α by the guessed m_b-bit subkey, and partially decrypt it with K_2 to get the plaintext P_2 within the same structure after xoring the known difference α.
   (b) With m_f-bit subkey, decrypt C_1 to the position of δ of the rectangle distinguisher and encrypt it to the ciphertext C_3 after xoring δ. Similarly, we find C_4 from C_2 and generate the quartet (C_1, C_2, C_3, C_4).
   (c) Check whether ciphertexts (C_3, C_4) exist in our data. If these ciphertexts exist, we partially encrypt corresponding plaintexts (P_3, P_4) under E_b with m_b-bit subkey, and check whether the difference is α. If so, increase the corresponding counter by 1.

Complexity. Choosing
$$y = \sqrt{s} \cdot 2^{n/2-r_b}/\hat{p}\hat{q}, \tag{3}$$
we get about $(y \cdot 2^{2r_b})^2 \cdot 2^{-2r_b} \cdot 2^{-n}\hat{p}^2\hat{q}^2 = s$, where s is the expected number of right quartets. Therefore, the total data complexity for the 4 oracles with K_1, K_2, K_3 and K_4 is
$$4y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/\hat{p}\hat{q}. \tag{4}$$
In Step 2, the time complexity is about $2^{m_b+m_f} \cdot 4y \cdot 2^{r_b} = 2^{m_b+m_f} \cdot \sqrt{s} \cdot 2^{n/2+2}/\hat{p}\hat{q}$. The memory complexity is $4y \cdot 2^{r_b} + 2^{m_b+m_f}$ to store the data and key counters.

2.2 `Attack II`: Biham-Dunkelman-Keller's Attack

At FSE 2002, Biham, Dunkelman and Keller introduced a more generic algorithm to perform the rectangle attack [12] in the single-key setting. Later, Liu *et al.* [47] converted the model into related-key setting for ciphers with linear key schedule. The high-level strategy of this model is to generate quartets by birthday paradox without key guessing, whose plaintexts and ciphertexts meet the truncated difference α' and δ', respectively. Then, recover the key candidates for each quartet. The steps are:

1. Create and store y structures of 2^{r_b} plaintexts each, and query the 2^{r_b} plaintexts under K_1, K_2, K_3 and K_4 for each structure.
2. Initialize an array of $2^{m_b+m_f}$ counters, where each corresponds to an $(m_b + m_f)$-bit subkey guess.
3. Insert the 2^{r_b} ciphertexts into a hash table H indexed by the $n - r_f$ inactive ciphertext bits. For each index, there are $2^{r_b} \cdot 2^{r_f-n}$ plaintexts and corresponding ciphertexts for each structure, which collide in the $n-r_f$ bits.
4. In each structure S, we search for a ciphertext pair (C_1, C_2), and choose a ciphertext C_3 by the $n-r_f$ inactive ciphertext bits of C_1 from hash table H. Choose a ciphertext C_4 indexed by the $n - r_f$ inactive ciphertext bits of C_2 from hash table H, where the corresponding plaintexts P_4 and P_3 are in the same structure. Then we obtain a quartet (P_1, P_2, P_3, P_4) and corresponding ciphertexts (C_1, C_2, C_3, C_4).
5. For the quartets obtained above, determine the key candidates involved in E_b and E_f using hash tables and increase the corresponding counters.

Complexity. The data complexity is the same as Eq. (4) given at `Attack I`, with the same y given by Eq. (3).

- **Time I**: The time complexity to generate quartets in Step 3 and 4 is about $y^2 \cdot 2^{2r_b} \cdot 2^{r_f-n} + (y \cdot 2^{2r_b+r_f-n})^2 = s \cdot 2^{r_f}/\hat{p}^2\hat{q}^2 + s \cdot 2^{2r_b+2r_f-n}/\hat{p}^2\hat{q}^2$and $y^2 \cdot 2^{4r_b+2r_f-2n} = s \cdot 2^{2r_b+2r_f-n}/\hat{p}^2\hat{q}^2$ quartets remain.
- **Time II**: The time complexity to deduce the right subkey and generate the counters in Step 5 is $y^2 \cdot 2^{4r_b+2r_f-2n} \cdot (2^{m_b-r_b} + 2^{m_f-r_f}) = s \cdot 2^{r_b+r_f-n} \cdot (2^{m_b+r_f} + 2^{m_f+r_b})/\hat{p}^2\hat{q}^2$.

2.3 `Attack III`: Zhao *et al.*'s Related-Key Attack

For block ciphers with linear key-schedule, Zhao *et al.* [62,64] proposed a new generalized related-key rectangle attack as shown below:

1. Construct y structures of 2^{r_b} plaintexts each. For each structure, query the 2^{r_b} plaintexts under K_1, K_2, K_3 and K_4.
2. Guess the m_b-bit subkey involved in E_b:
   (a) Initialize a list of 2^{m_f} counters.
   (b) Partially encrypt plaintext P_1 with K_1 to obtain the intermediate values at the position of α, and xor the known difference α, and then partially decrypt it to the plaintext P_2 under K_2 within the same structure. Construct the set S_1 and also S_2 in similar way:

$$S_1 = \{(P_1, C_1, P_2, C_2) : E_{b_{K_1}}(P_1) \oplus E_{b_{K_2}}(P_2) = \alpha\},$$
$$S_2 = \{(P_3, C_3, P_4, C_4) : E_{b_{K_3}}(P_3) \oplus E_{b_{K_4}}(P_4) = \alpha\}.$$

(c) The size of S_1 and S_2 is $y \cdot 2^{r_b}$. Insert S_1 into a hash table H_1 indexed by the $n-r_f$ inactive bits of C_1 and $n-r_f$ inactive bits of C_2. Similarly build H_2. Under the same $2(n-r_f)$-bit index, randomly choose (C_1, C_2) from H_1 and (C_3, C_4) from H_2 to construct the quartet (C_1, C_2, C_3, C_4).

(d) We use all the quartets obtained above to determine the key candidates involved in E_f and increase the corresponding counters. This phase is a guess and filter procedure, whose time complexity is denoted as ε.

Complexity. The data complexity is the same as Eq. (4) given by `Attack I`, with the same y given by Eq. (3).

- **Time I**: The time complexity to generate S_1 and S_2 is about $2^{m_b} \cdot y \cdot 2^{r_b}$.
- **Time II**: We generate $2^{m_b} \cdot (y2^{r_b})^2 \cdot 2^{-2(n-r_f)} = 2^{m_b} \cdot y^2 \cdot 2^{2r_b-2(n-r_f)} = s \cdot 2^{m_b-n+2r_f}/\hat{p}^2\hat{q}^2$quartets from Step 2(c). The time to generate the key counters is $(s \cdot 2^{m_b-n+2r_f}/\hat{p}^2\hat{q}^2) \cdot \varepsilon$.

3 Key-Guessing Strategies in the Rectangle Attack

Suppose Figure 2 shows a framework for differential attack, then E_d is a differential trail $\alpha \mapsto \delta$. In the differential attack, we collect plaintext-ciphertext pairs by traversing the r_b active bits of plaintext to construct a structure. Store the structure indexed by the $n - r_f$ inactive bits of ciphertext in a hash table H. Thereafter, we generate (P_1, C_1, P_2, C_2) by randomly picking (P_1, C_1) and (P_2, C_2) from H within the same index. For each structure, with the birthday paradox, we expect to get $2^{2r_b-1-(n-r_f)}$ plaintext pairs, and the differences of plaintexts and ciphertexts in each pair conform to the truncated form α' and δ', respectively. Using the property of truncated differential of the ciphertext to filter wrong pairs in advance due to the birthday paradox is an efficient and generic way in differential attack and its variants, such as impossible differential attack, truncated differential attack, boomerang attack, rectangle attack, etc.

In `Attack II` of the rectangle attack, Biham, Dunkelman and Keller [12] also generated the quartets using birthday paradox. For each quartet (P_1, P_2, P_3, P_4), the plaintexts and ciphertexts also conform to the truncated forms (α', δ' in Figure 2), i.e., $P_1 \oplus P_2$ and $P_3 \oplus P_4$ are of truncated form α', $C_1 \oplus C_3$ and $C_2 \oplus C_4$ are of truncated form δ'. However, when deducing key candidates for each of the generated quartets, we find that the rectangle attack enjoys a very big filter ratio. In other words, the ratio of right quartets which satisfy the input and output differences of the rectangle distinguisher (α, δ in Figure 2) and suggest key candidates is very small, when compared to the number of the quartets that satisfy the truncated differential (α', δ') in the plaintext and ciphertext.

For the differential attack, given a pair conforming to (α', δ'), it will suggest $2^{m_b+m_f-(r_b+r_f)}$ key candidates. However, for the rectangle attack, given a quartet conforming to (α', δ'), it will suggest $2^{m_b+m_f-2(r_b+r_f)}$ key candidates due to the filter in both sides of the boomerang. Hence, if $2(r_b + r_f)$ is bigger than m_b+m_f, some quartets conforming to (α', δ') may never suggest key candidates.

Here is an example of E_b part in Figure 3. Since we are considering linear key schedule, we have $k_{2b} = k_{1b} \oplus \Delta$, $k_{3b} = k_{1b} \oplus \nabla$ and $k_{4b} = k_{1b} \oplus \Delta \oplus \nabla$ with fixed (Δ, ∇). Hence, when k_{1b} is known, all other k_{2b}, k_{3b} and k_{4b} are determined. Let S be an Sbox. Then we have

$$S(k_{1b} \oplus P_1) \oplus S(k_{2b} \oplus P_2) = \alpha, \tag{5}$$
$$S(k_{3b} \oplus P_3) \oplus S(k_{4b} \oplus P_4) = \alpha. \tag{6}$$

For a quartet (P_1, P_2, P_3, P_4), when (P_1, P_2) is known, together with $k_{1b} \oplus k_{2b} = \Delta$, we can determine a value for k_{1b} and k_{2b} by Eq. (5). Then k_{3b}, k_{4b} are determined. Hence, by Eq. (6), P_4 is determined by P_3. Hence, P_4 is fully determined by (P_1, P_2, P_3) within a *good* quartet, which may suggest a key. For certain quartets, (P_1, P_2, P_3, P_4) may violate the nonlinear relations (e.g., Eq. (5) and (6)), so that it will never suggest a key.

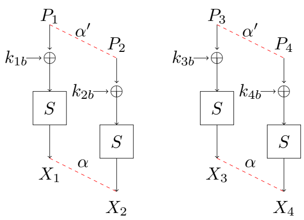

Fig. 3: Nonlinear relations in key-recovery phase.

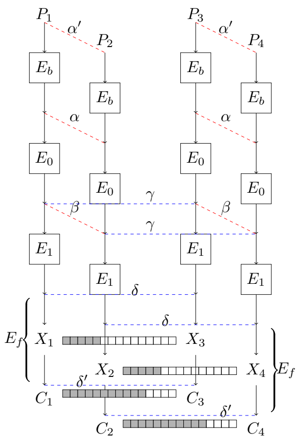

Fig. 4: Filter with internal state.

According to the analysis of `Attack I` and `Attack III` in Section 2, both of them guess (part of) key bits before generating the quartets, i.e., $(m_b + m_f)$-bit and m_b-bit key are guessed in `Attack I` and `Attack III`, respectively. For example in Figure 3, if we guess k_{1b} in E_b, we can deduce k_{2b}, k_{3b}, k_{4b}. Then, for given P_1 and P_3, we compute P_2 and P_4 with Eq. (5) and (6), respectively. Thereafter, a quartet (P_1, P_2, P_3, P_4) is generated under guessed key k_{1b}, which meets the input difference α. In this way, we can avoid some invalid quartets that never suggest a key in advance.

However, if we guess all the key bits (k_b in E_b and k_f in E_f) at once and then construct quartets as `Attack I`, we may lose the benefit from the early abort technique [49], which tests the key candidates step by step, by reducing the size of the remaining possible quartets at each time, without (significantly) increasing the time complexity. Guessing a lot of key bits at once may reduce the number of invalid quartets, but may also lead to higher overall complexity. To get a better tradeoff, we try to guess all k_b and part of k_f, denoted as k'_f whose size is m'_f. With partial decryption, we may gain more inactive bits (or bits with fixed differences) from the internal state as shown in Figure 4.

3.1 New Related-key Rectangle Attack with Linear Key Schedule

With the above analysis, we derive a new tradeoff of the rectangle attack framework with linear key schedule, which tries to obtain better attacks by the overall consideration on various factors affecting the complexity and the number of attacked rounds. We list our tradeoff model in Algorithm 1. Before diving into it, we give Figure 5 to illustrate which key to guess.

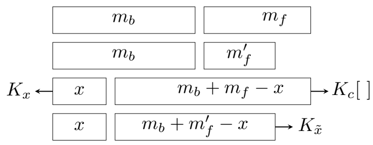

Fig. 5: The guessed key in Algorithm 1

Totally, m_b-bit k_b and m_f-bit k_f are involved in E_b and E_f. Among them, we first guess m_b-bit k_b and m'_f-bit k'_f before generating quartets. Then we use both the inactive bits of the ciphertexts and the difference of internal states computed by k'_f to act as early filters. In order to possibly reduce the memory cost of key counters, we introduce an auxiliary variable x and guess x-bit K_x in Line 3 before initializing the (m_b+m_f-x)-bit key counter $K_c[\,]$. The remaining $(m_b+m'_f-x)$-bit $K_{\tilde{x}}$ is guessed in Line 5 of Algorithm 1.

Complexity. Choosing

$$y = \sqrt{s} \cdot 2^{n/2-r_b}/\hat{p}\hat{q}, \tag{7}$$

we get about $(y \cdot 2^{2r_b})^2 \cdot 2^{-2r_b} \cdot 2^{-n}\hat{p}^2\hat{q}^2 = s$, where s is the expected number of right quartets. Therefore, the total data complexity for the 4 oracles with K_1, K_2, K_3 and K_4 is

$$4y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/\hat{p}\hat{q}. \tag{8}$$

- ▶ **Time I** (T_1): In Line 7 to 26 of Algorithm 1, the time complexity is about

$$T_1 = 2^{x+m_b+m'_f-x} \cdot y \cdot 2^{r_b} \cdot 2 = \sqrt{s} \cdot 2^{m_b+m'_f+n/2+1}/\hat{p}\hat{q}. \tag{9}$$

- ▶ **Time II** (T_2): In Line 29, we generate about

$$2^{x+m_b+m'_f-x} \cdot y^2 \cdot 2^{2r_b-2(n-r_f)-2h_f} = s \cdot 2^{m_b+m'_f-n+2r_f-2h_f}/\hat{p}^2\hat{q}^2 \tag{10}$$

quartets. The time complexity of Line 32 to generate the key counters is

$$T_2 = (s \cdot 2^{m_b+m'_f-n+2r_f-2h_f}/\hat{p}^2\hat{q}^2) \cdot \varepsilon. \tag{11}$$

Algorithm 1: Related-key rectangle attack with linear key schedule (`Attack IV`)

Construct y structures of 2^{r_b} plaintexts each
For structure i ($1 \leq i \leq y$), query the 2^{r_b} plaintexts by encryption under K_1, K_2, K_3 and K_4 and store them in $L_1[i]$, $L_2[i]$, $L_3[i]$ and $L_4[i]$
for *each of the x-bit key K_x, which is a part of $(m_b + m'_f)$-bit K_1* **do**
  $K_c \leftarrow [\,]$ `/* Key counters of size` $2^{m_b+m_f-x}$ `*/`
  for *each of $(m_b + m'_f - x)$-bit $K_{\tilde{x}}$ of K_1 involved in E_b and E_f* **do**
    $S_1 \leftarrow [\,]$, $S_2 \leftarrow [\,]$
    for *i from 1 to y* **do**
      for $(P_1, C_1) \in L_1[i]$ **do**
        `/* Partially encrypt` P_1 `to` α `under guessed` K_1 `and partially decrypt to get the plaintext` $P_2 \in L_2[i]$ `*/`
        $P_2 = E^{-1}_{b_{K_1 \oplus \Delta K}}(E_{b_{K_1}}(P_1) \oplus \alpha)$
        $S_1 \leftarrow (P_1, C_1, P_2, C_2)$
      end
      for $(P_3, C_3) \in L_3[i]$ **do**
        $P_4 = E^{-1}_{b_{K_1 \oplus \Delta K \oplus \nabla K}}(E_{b_{K_1 \oplus \nabla K}}(P_3) \oplus \alpha)$
        $S_2 \leftarrow (P_3, C_3, P_4, C_4)$
      end
    end
    `/*` $S_1 = \{(P_1,C_1,P_2,C_2) : (P_1,C_1) \in L_1, (P_2,C_2) \in L_2, E_{b_{K_1}}(P_1) \oplus E_{b_{K_2}}(P_2) = \alpha\}$
    $S_2 = \{(P_3,C_3,P_4,C_4) : (P_3,C_3) \in L_3, (P_4,C_4) \in L_4, E_{b_{K_3}}(P_3) \oplus E_{b_{K_4}}(P_4) = \alpha\}$ `*/`
    $H \leftarrow [\,]$
    for $(P_1, C_1, P_2, C_2) \in S_1$ **do**
      `/* Assuming the first` h_f`-bit internal states of` X_1 `and` X_2 `are derived by decrypting` (C_1, C_2) `with` k'_f `*/`
      $X_1[1, \cdots, h_f] = E^{-1}_{f_{K_1}}(C_1)$, $X_2[1, \cdots, h_f] = E^{-1}_{f_{K_1 \oplus \Delta K}}(C_2)$
      `/* Assume the inactive bits of` δ' `are first` $n - r_f$ `bits */`
      $\tau = (X_1[1,\cdots,h_f], X_2[1,\cdots,h_f], C_1[1,\cdots,n-r_f], C_2[1,\cdots,n-r_f])$
      $H[\tau] \leftarrow (P_1, C_1, P_2, C_2)$
    end
    for $(P_3, C_3, P_4, C_4) \in S_2$ **do**
      $X_3[1, \cdots, h_f] = E^{-1}_{f_{K_1 \oplus \nabla K}}(C_3)$, $X_4[1, \cdots, h_f] = E^{-1}_{f_{K_1 \oplus \Delta K \oplus \nabla K}}(C_4)$
      $\tau' = (X_3[1,\cdots,h_f], X_4[1,\cdots,h_f], C_3[1,\cdots,n-r_f], C_4[1,\cdots,n-r_f])$
      Access $H[\tau']$ to find (P_1, C_1, P_2, C_2) to generate quartet (C_1, C_2, C_3, C_4).
      for *each generated quartet* **do**
        Determine the other $(m_f - m'_f)$-bit key k''_f involved in E_f
        $K_c[K_{\tilde{x}} \| k''_f] \leftarrow K_c[K_{\tilde{x}} \| k''_f] + 1$ `/* Denote the time as` ε `*/`
      end
    end
  end
  `/* Exhaustive search step */`
  Select the top $2^{m_b+m_f-x-h}$ hits in the counter to be the candidates, which delivers an h-bit or higher advantage. Guess the remaining $k - (m_b + m_f)$ bit keys combined with the guessed x subkey bits to check the full key.
end

▶ **Time III (T_3)**: The time complexity of the exhaustive search is

$$T_3 = 2^x \cdot 2^{m_b+m_f-x-h} \cdot 2^{k-(m_b+m_f)} = 2^{k-h}. \tag{12}$$

For choosing h (according to the success probability Eq. (14)), the conditions $m_b + m_f - x - h \geq 0$ and $x \leq m_b + m'_f$ have to be satisfied.

The memory to store the key counters and the data structures is

$$2^{m_b+m_f-x} + 4y \cdot 2^{r_b} = 2^{m_b+m_f-x} + \sqrt{s} \cdot 2^{n/2+2}/\hat{p}\hat{q}. \tag{13}$$

3.2 On the Success Probability and Exhaustive Search Phase

The success probability given by Selçuk [55] is evaluated by

$$P_s = \Phi(\frac{\sqrt{sS_N} - \Phi^{-1}(1-2^{-h})}{\sqrt{S_N+1}}), \tag{14}$$

where $S_N = \hat{p}^2\hat{q}^2/2^{-n}$ is the signal-to-noise ratio, with an h-bit or higher advantage. s is the expected number of right quartets, which will be adjusted to achieve a relatively higher P_s, usually $s = 1, 2, 3$. In previous `Attack I, II, III` and our `Attack IV`, after generating the k_c-bit key counter, we select the top 2^{k_c-h} hits in the counters to be the candidates, which delivers an h-bit or higher advantage, and determine the right key by exhaustive search.

In `Attack I/II`, the size of key counters is $2^{m_b+m_f}$. Hence, we have to prepare a memory with size of $2^{m_b+m_f}$ to store the counters. Then the complexity of exhaustive search is $2^{(m_b+m_f-h)} \times 2^{k-(m_b+m_f)} = 2^{k-h}$, where $h \leq m_b + m_f$. Hence, the time of exhaustive search is larger than $2^{k-(m_b+m_f)}$.

In `Attack III`, the size of key counter is 2^{m_f}, which is smaller than `Attack I/II`. Then the complexity of the exhaustive search is $2^{m_b} \times 2^{m_f-h} \times 2^{k-(m_b+m_f)} = 2^{k-h}$ for `Attack III`, where $h < m_f$ because the size of key counters is 2^{m_f}. Hence, the time complexity is larger than 2^{k-m_f}. Compared to `Attack I/II`, the memory is reduced but the time may be increased.

In `Attack IV`, the size of key counter is bigger than $2^{m_b+m_f-x}$, which is smaller than `Attack I/II`, but may be larger than `Attack III` by choosing x. The time complexity of exhaustive search (T_3 in `Attack IV`) is $2^x \times 2^{m_b+m_f-x-h} \times 2^{k-(m_b+m_f)} = 2^{k-h}$ with $h < m_b + m_f - x$ and $x \leq m_b + m'_f$. Hence, the time is larger than $2^{k-(m_b+m_f-x)}$ with a key counter of size $2^{m_b+m_f-x}$. Namely, we can further tradeoff the time and memory by tweaking x between the two points achieved by `Attack I/II` ($x = 0$) and `Attack III` ($x = m_b$).

As shown in Algorithm 1 and its complexity analysis, we have to determine various parameters to derive a better attack. Many parameters are determined by the boomerang distinguishers, such as m_b, m_f, r_b, r_f and $\hat{p}\hat{q}$. Parameters like x affect the exhaustive search. Moreover, we have to determine the m'_f-bit k'_f including the number of cells and their positions. All these parameters affect the overall complexity of our tradeoff attacks.

To determine a series of optimal parameters, we take `SKINNY` as an example to build a fully automatic model to identify the boomerang distinguishers with optimal key-recovery parameters in the following section.

4 Automatic Model For SKINNY

SKINNY [7] is a family of lightweight block cipher proposed by Beierle *et al.* at CRYPTO 2016, which follows an SPN structure and a TWEAKEY framework [42]. Denote n as the block size and $\tilde{n}$ as the tweakey size. There are six main versions SKINNY-n-$\tilde{n}$: $n = 64, 128$, $\tilde{n} = n, 2n, 3n$. The internal state is viewed as a 4×4 square array of cells, where c is the cell size. For more details of the cipher's structure, please refer to Section A of the full version of the paper and [7]. The MC operation adopts non-MDS binary matrix:

$$\texttt{MC}\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \oplus c \oplus d \\ a \\ b \oplus c \\ a \oplus c \end{pmatrix} \text{ and } \texttt{MC}^{-1}\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \beta \\ \beta \oplus \gamma \oplus \delta \\ \beta \oplus \delta \\ \alpha \oplus \delta \end{pmatrix}. \tag{15}$$

Lemma 1 *[6] For any given* SKINNY *S-box S and any two non-zero differences δ_{in} and δ_{out}, the equation $S_i(y) \oplus S_i(y \oplus \delta_{in}) = \delta_{out}$ has one solution on average.*

4.1 Previous Automatic Search Models for Boomerang Distinguishers on SKINNY

On SKINNY, there are several automatic models on searching for boomerang distinguishers. The designers of SKINNY [7] first gave the Mixed-Integer Linear Programming (MILP) model to search for truncated differentials of SKINNY. Later, Liu *et al.* [47] tweaked the model to search for boomerang distinguishers. At EUROCRYPT 2018, Cid *et al.* [28] introduced the the Boomerang Connectivity Table (BCT) to compute the probability of the boomerang distingusher. Later, Song *et al.* [57] studied the probability of SKINNY's boomerang distinguisher with an extended BCT technique. Hadipour *et al.* [41] introduced a heuristic approach to search for a boomerang distinguisher with a set of new tables. They first searched for truncated differential with the minimum number of active S-boxes with an MILP model based on Cid *et al.*'s [29] model. At the same time, the switching effects in multiple rounds were considered. Then, they used the MILP/SAT models to get actual differential characteristic and experimentally evaluated the probability of the middle part. Almost at the same time, Delaune, Derbez and Vavrille [30] proposed a new automatic tool to search for boomerang distinguishers and provided their source code to facilitate follow-up works. They also introduced a sets of tables which help to calculate the probability of the boomerang distinguisher. With the tables to help roughly evaluate the probability, they used an MILP model to search for the upper and lower trails throughout all rounds by automatically handling the middle rounds. Then a CP model was applied to search for the best possible instantiations. Recently, Qin *et al.* [52] combined the key-recovery attack phase and distinguisher searching phase into one uniform automatic model to attack more rounds. Their extended model tweaked the previous models of Hadipour *et al.* [41] and Delaune *et al.* [30] for searching for the entire $(N_b + N_d + N_f)$ rounds of a boomerang attack.

The aim is to find new boomerang distinguishers in the related-tweakey setting that give a key-recovery attack penetrating more rounds.

4.2 Our Model to Determine the Optimal Distinguisher

In Dunkelman *et al.*'s (related-key) sandwich attack framework [37], the N_d-round cipher E_d is considered as $\tilde{E}_1 \circ E_m \circ \tilde{E}_0$, where $\tilde{E}_0$, E_m, $\tilde{E}_1$ contain r_0, r_m, r_1 rounds, respectively. Let $\tilde{p}$ and $\tilde{q}$ be the probabilities of the upper differential used for $\tilde{E}_0$ and the lower differential used for $\tilde{E}_1$. The middle part E_m specifically handles the dependence and contains a small number of rounds. If the probability of generating a right quartet for E_m is t, the probability of the whole N_d-round boomerang distinguisher is $\tilde{p}^2\tilde{q}^2t$. In the following, we use the above symbols in our search model.

Following the previous automatic models [52,30,41], we introduce a uniform automatic model to search for good distinguishers for the new rectangle attack framework in Algorithm 1. We search for the entire $(N_b + N_d + N_f)$ rounds of a boomerang attack by adding new constraints and new objective function, and takes all the critical factors affecting the complexities into account.

In our extended model searching the entire $(N_b + N_d + N_f)$ rounds of a boomerang attack, we use similar notations as [52,30], where X_r^u and X_r^l denote the internal state before `SubCells` in round r of the upper and lower differentials. We only list the variables that appear in our new constraints, i.e. $\texttt{DXU}[r][i]$ $(0 \le r \le N_b + r_0 + r_m, 0 \le i \le 15)$ and $\texttt{DXL}[r][i]$ $(0 \le r \le r_m + r_1 + N_f, 0 \le i \le 15)$ are on behalf of active cells in the internal states, and `KnownEnc` $(0 \le r \le N_b - 1, 0 \le i \le 15)$ is on behalf of the m_b-bit subtweakeys involved in the N_b extended rounds, i.e., $\sum_{0\le r\le N_b-2,\ 0\le i\le 7} \texttt{KnownEnc}[r][i]$ corresponds to the total amount of guessed m_b-bit key in E_b. The constraints in E_b are the same as Qin *et al.*'s [52] model. In the following, we list the differences in our model.

Modelling propagation of cells with known differences in E_f. Since we are going to filter quartets with certain cells of the internal state with fixed differences, we need to model the propagation of fixed differences in E_f. Taking the key-recovery attack on 32-round `SKINNY-128-384` as an example (see Figure 6), the cells with fixed differences are marked by ⊡ and ▣. We define a binary variable $\texttt{DXFixed}[r][i]$ for the i-th cell of X_r and a binary variable $\texttt{DWFixed}[r][i]$ for the i-th cell of W_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$, where $\texttt{DXFixed}[r][i] = 1$ and $\texttt{DWFixed}[r][i] = 1$ indicate that the differences of corresponding cells are fixed. For the first extended round after the lower differential, the difference of each cell is fixed: $\forall\ 0 \le i \le 15, \texttt{DXFixed}[0][i] = 1$.

In the propagation of the fixed differences, after the `SC` operation, only the differences of inactive cells are fixed. In the `ART` operation, the subtweakey differences do not affect whether the differences are fixed. Let permutation $P_{\texttt{SR}} = [0, 1, 2, 3, 7, 4, 5, 6, 10, 11, 8, 9, 13, 14, 15, 12]$ represent the `SR` operation,

$$\texttt{DWFixed}[r][i] = \neg\texttt{DXL}[r_m + r_1 + r][P_{\texttt{SR}}[i]], \forall\ 0 \le r \le N_f - 1, 0 \le i \le 15.$$

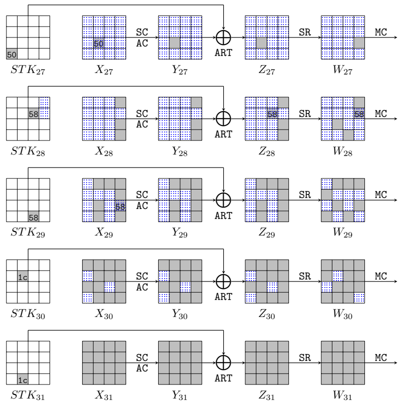

Fig. 6: The cells with fixed differences in N_f-round of the attack on SKINNY-128-384.

The constraints on the impact of the MC operation by Equation (15) on the internal state are given below: $\forall\ 0 \leq r \leq N_f - 2, 0 \leq i \leq 3$,

$$\begin{cases} \texttt{DXFixed}[r+1][i] = \texttt{DWFixed}[r][i] \wedge \texttt{DWFixed}[r][i+8] \wedge \texttt{DWFixed}[r][i+12], \\ \texttt{DXFixed}[r+1][i+4] = \texttt{DWFixed}[r][i], \\ \texttt{DXFixed}[r+1][i+8] = \texttt{DWFixed}[r][i+4] \wedge \texttt{DWFixed}[r][i+8], \\ \texttt{DXFixed}[r+1][i+12] = \texttt{DWFixed}[r][i] \wedge \texttt{DWFixed}[r][i+8]. \end{cases}$$

Modelling cells that could be used to filter quartets in E_f. Note that in our attack framework in Algorithm 1, we guess m'_f-bit k'_f of k_f involved in N_f extended rounds to obtain a $2h_f$-bit filter. To identify smaller m'_f with larger h_f, we define a binary variable DXFilter$[r][i]$ for i-th cell of X_r and a binary variable DWFilter$[r][i]$ for i-th cell of W_r $(0 \leq r \leq N_f - 1, 0 \leq i \leq 15)$, where DXFilter$[r][i] = 1$ and DWFilter$[r][i] = 1$ indicate that the corresponding cells can be used as filters. Note that, the $(n - r_f)$ inactive bits of the ciphertext are also indicated by DWFilter. For each cell in X_r, if the difference is nonzero and fixed, we can choose the cell as filter, i.e. ⊡ $\xrightarrow{\texttt{SC}}$ ■. For ⊡ $\xrightarrow{\texttt{SC}}$ ⊡, the cell is not a filter because it has been used as filter in W_r. The valid valuations of DXFixed, DXL and DXFilter are given in Table 2.

Table 2: All valid valuations of DXFixed, DXL and DXFilter for SKINNY.

DXFixed$[r][i]$	DXL$[r_m + r_1 + r][i]$	DXFilter$[r][i]$
0	1	0
1	0	0
1	1	1

In the last round, W_{N_f-1} can be computed from the ciphertexts, and the cells with fixed differences of W_{N_f-1} can be used as filters, i.e., the $(n - r_f)$ inactive bits: $\forall\ 0 \leq i \leq 15, \texttt{DWFilter}[N_f - 1][i] = \texttt{DWFixed}[N_f - 1][i]$.

Since we extend N_f rounds with probability 1 at the bottom of the distinguisher, then the differences of W_r are propagated to X_{r+1} with probability 1 with the `MC` operation, and there will be more cells of W_r with fixed differences than the cells of X_{r+1} with fixed differences. Hence, these extra cells with fixed differences in W_r can act as filters. We give two examples of how to determine which cells of W_r can be used for filtering:

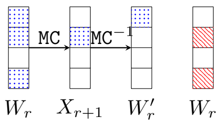

Fig. 7: Example (1).

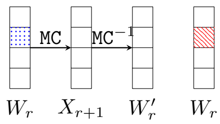

Fig. 8: Example (2).

1. Example (1): Figure 7 shows the propagation of fixed differences, i.e., `DWFixed` and `DXFixed`, where □ cells denote the unfixed differences. In Figure 7, the differences of $W_r[0, 1, 3]$ are fixed (marked by ⊡). After the `MC` operation, only the difference of $X_{r+1}[1]$ is fixed. Since there are three cells with fixed differences in W_r but only one cell with fixed difference in X_{r+1}, we can use two cells of W_r as filters (the one cell of fixed difference in X_{r+1} has been used in the `SC` computation). To determine which cells acting as filters, we apply the $\texttt{MC}^{-1}$ operation to X_{r+1} and get fixed difference of $W'_r[0]$, which means if $\Delta X_{r+1}[1]$ is fixed, then $\Delta W_r[0]$ will be certainly fixed. Since $X_{r+1}[1]$ has been used as filter in the `SC` computation, $W_r[0]$ will not act as filter redundantly. Hence, only $W_r[1, 3]$ can be used as filters (marked by ⧅).
2. Example (2): In Figure 8, only the difference of $W_r[1]$ is fixed, which is marked by ⊡. After applying the `MC` operation, all the differences of X_{r+1} are unfixed. So applying the $\texttt{MC}^{-1}$ operation to X_{r+1}, all the differences of W'_r are unfixed. Hence, the difference of $W_r[1]$ need to be fixed, which can be used for filtering (marked by ⧅).

All valid valuations of `DWFixed` and `DWFilter` please refer to Section B of the full version of the paper. Note that `DXFixed` is only used as the intermediate variable to determine `DWFilter`, since `DXFixed` is fully determined by `DWFixed`.

Denoting the sets of all possible valuations listed in Table 2 and Table 8 in the full version of the paper by $\mathbb{P}_i$ and $\mathbb{Q}_i$, there are

$$\begin{cases} (\texttt{DXFixed}[r][i], \texttt{DXL}[r_m + r_1 + r][i], \texttt{DXFilter}[r][i]) \in \mathbb{P}_i, \forall\ 0 \leq r \leq N_f - 1, 0 \leq i \leq 15, \\ (\texttt{DWFixed}[r][i], \texttt{DWFixed}[r][i+4], \texttt{DWFixed}[r][i+8], \texttt{DWFixed}[r][i+12], \\ \texttt{DWFilter}[r][i], \texttt{DWFilter}[r][i+4], \texttt{DWFilter}[r][i+8], \texttt{DWFilter}[r][i+12]) \in \mathbb{Q}_i, \\ \hfill \forall\ 0 \leq r \leq N_f - 2, 0 \leq i \leq 3. \end{cases}$$

We define a binary variable $\texttt{DXisFilter}[r][i]$ for i-th cell of X_r and a binary variable $\texttt{DWisFilter}[r][i]$ for i-th cell of W_r $(0 \leq r \leq N_f - 1, 0 \leq i \leq 15)$, where $\texttt{DXisFilter}[r][i] = 1$ and $\texttt{DWisFilter}[r][i] = 1$ indicate that the corresponding cells are chosen as filters before generating quartets. $\forall\, 0 \leq r \leq N_f - 1, 0 \leq i \leq 15$, $\texttt{DXisFilter}[r][i] \leq \texttt{DXFilter}[r][i], \texttt{DWisFilter}[r][i] \leq \texttt{DWFilter}[r][i]$.

Modeling the guessed subtweakey cells in E_f for generating the quartets. We define a binary variable $\texttt{DXGuess}[r][i]$ for i-th cell of X_r and a binary variable $\texttt{DWGuess}[r][i]$ for i-th cell of W_r $(0 \leq r \leq N_f - 1, 0 \leq i \leq 15)$, where $\texttt{DXGuess}[r][i] = 1$ and $\texttt{DWGuess}[r][i] = 1$ indicate that the corresponding cells need to be known in decryption from ciphertexts to the cells acting as filters. So whether $STK_r[i]$ should be guessed is also identified by $\texttt{DXGuess}[r][i]$, where $0 \leq r \leq N_f - 1$ and $0 \leq i \leq 7$.

For the round 0, only cells used to be filters in the internal state need to be known: $\forall\, 0 \leq i \leq 15, \texttt{DXGuess}[0][i] = \texttt{DXisFilter}[0][i]$.

From round 0 to round $N_f - 1$, the cells in W_r need to be known involve two types: cells to be known from X_r over the SR operation, and cells used to be filters in W_r:

$$\texttt{DWGuess}[r][i] = \texttt{DWisFilter}[r][i] \vee \texttt{DXGuess}[r][P_{\texttt{SR}}[i]],\ \forall\, 0 \leq r \leq N_f - 1,\ 0 \leq i \leq 15.$$

In round 0 to round $N_f - 2$, the cells in X_{r+1} need to be known involve two types: cells to be known from W_r over the MC operation, and cells used to be filters in X_{r+1}: $\forall\, 0 \leq r \leq N_b - 2,\ 0 \leq i \leq 3$

$$\begin{cases} \texttt{DXGuess}[r+1][i] = \texttt{DWGuess}[r][i+12] \vee \texttt{DXisFilter}[r+1][i], \\ \texttt{DXGuess}[r+1][i+4] = \texttt{DWGuess}[r][i] \vee \texttt{DWGuess}[r][i+4] \vee \texttt{DWGuess}[r][i+8] \vee \\ \qquad \texttt{DXisFilter}[r+1][i+4], \\ \texttt{DXGuess}[r+1][i+8] = \texttt{DWGuess}[r][i+4] \vee \texttt{DXisFilter}[r+1][i+8], \\ \texttt{DXGuess}[r+1][i+12] = \texttt{DWGuess}[r][i+4] \vee \texttt{DWGuess}[r][i+8] \vee \texttt{DWGuess}[r][i+12] \vee \\ \qquad \texttt{DXisFilter}[r+1][i+12]. \end{cases}$$

We have $\sum_{0 \leq r \leq N_f - 1,\ 0 \leq i \leq 7} \texttt{DXGuess}[r][i]$ to indicate the m'_f-bit key guessed for generating quartets.

Modelling the advantage h in the key-recovery attack. In our Algorithm 1 in Section 3.1, the advantage h determines the exhaustive search time, where h should be smaller than the number of key counters, i.e. $h \leq m_b + m_f - x$. The x-bit guessed subkey should satisfy $x \leq m_b + m'_f$, and also determine the size of memory $2^{m_b + m_f - x}$ to store the key counters. So we need a balance between x and h to achieve a low time and memory complexities. We define an integer variable Adv for h and an integer variable x. To describe m_f (not m'_f here), we define a binary variable $\texttt{KnownDec}[r][i]$ for i-th cell of Y_r $(0 \leq r \leq N_f - 1, 0 \leq i \leq 15)$, where $\texttt{KnownDec}[r][i] = 1$ indicates that the corresponding cell should be known in the decryption from ciphertext to the position of known δ. Then whether

$STK_r[i]$ should be guessed is also identified by $\texttt{KnownDec}[r][i]$, where $0 \le r \le N_f - 1$ and $0 \le i \le 7$. In the first round extended after the distinguisher, only the active cells need to be known: $\forall\ 0 \le i \le 15, \texttt{KnownDec}[0][i] = \texttt{DXL}[r_m + r_1][i]$.

In round 1 to round $N_f - 1$, the cells in Y_{r+1} need to be known involve two types: cells to be known from W_r over the MC and SB operation, and active cells in X_{r+1}: $\forall\ 0 \le r \le N_b - 2,\ 0 \le i \le 3$

$$\begin{cases} \texttt{KnownDec}[r+1][i] = \texttt{DXL}[r_m + r_1 + r + 1][i] \vee \texttt{KnownDec}[r][P_{\texttt{SR}}[i+12]], \\ \texttt{KnownDec}[r+1][i+4] = \texttt{DXL}[r_m + r_1 + r + 1][i+4] \vee \texttt{KnownDec}[r][P_{\texttt{SR}}[i]] \vee \\ \qquad \texttt{KnownDec}[r][P_{\texttt{SR}}[i+4]] \vee \texttt{KnownDec}[r][P_{\texttt{SR}}[i+8]], \\ \texttt{KnownDec}[r+1][i+8] = \texttt{DXL}[r_m + r_1 + r + 1][i+8] \vee \texttt{KnownDec}[r][P_{\texttt{SR}}[i+4]], \\ \texttt{KnownDec}[r+1][i+12] = \texttt{DXL}[r_m + r_1 + r + 1][i+12] \vee \texttt{KnownDec}[r][P_{\texttt{SR}}[i+4]] \vee \\ \qquad \texttt{KnownDec}[r][P_{\texttt{SR}}[i+8]] \vee \texttt{KnownDec}[r][P_{\texttt{SR}}[i+12]]. \end{cases}$$

We have $\sum_{0 \le r \le N_f - 1,\ 0 \le i \le 7} \texttt{KnownDec}[r][i]$ to indicate the m_f-bit key.

The objective function. As in Sect. 3.1, the time complexities of our new attack framework involve three parts: **Time I** (T_1), **Time II** (T_2) and **Time III** (T_3). We need to balance those time complexities T_1, T_2 and T_3.

The constraints for probability $\tilde{p}^2 t \tilde{q}^2$ of the boomerang distinguisher are same as [30], where DXU, DXL and $\texttt{DXU} \wedge \texttt{DXL}$ are on behalf of $\tilde{p}$, $\tilde{q}$ and t. KnownEnc is on behalf of m_b, and we do not repeat the details here. To describe T_1, we have:

$$\begin{aligned} T_1 = & \sum_{0 \le r \le r_0 - 1,\ 0 \le i \le 15} w_0 \cdot \texttt{DXU}[N_b + r][i] + \sum_{0 \le r \le r_1 - 1,\ 0 \le i \le 15} w_1 \cdot \texttt{DXL}[r_m + r][i] + \\ & \sum_{0 \le r \le r_m - 1,\ 0 \le i \le 15} w_m \cdot (\texttt{DXU}[N_b + r_0 + r][i] \wedge \texttt{DXL}[r][i]) + \\ & \sum_{0 \le r \le N_b - 2,\ 0 \le i \le 7} w_{m_b} \cdot \texttt{KnownEnc}[r][i] + \sum_{0 \le r \le N_f - 1,\ 0 \le i \le 7} w_{m_f} \cdot \texttt{DXGuess}[r][i] + c_{T_1}, \end{aligned}$$

where c_{T_1} indicates the constant factor $2^{n/2+1}$, and w_0, w_1, w_m, w_{m_b}, w_{m_f} are weights factors discussed later.

For describing T_2 (let $\varepsilon = 1$), we have:

$$\begin{aligned} T_2 = & \sum_{0 \le r \le r_0 - 1,\ 0 \le i \le 15} 2w_0 \cdot \texttt{DXU}[N_b + r][i] + \sum_{0 \le r \le r_1 - 1,\ 0 \le i \le 15} 2w_1 \cdot \texttt{DXL}[r_m + r][i] + \\ & \sum_{0 \le r \le r_m - 1,\ 0 \le i \le 15} 2w_m \cdot (\texttt{DXU}[N_b + r_0 + r][i] \wedge \texttt{DXL}[r][i]) + \\ & \sum_{0 \le r \le N_b - 2,\ 0 \le i \le 7} w_{m_b} \cdot \texttt{KnownEnc}[r][i] + \sum_{0 \le r \le N_f - 1,\ 0 \le i \le 7} w_{m_f} \cdot \texttt{DXGuess}[r][i] - \\ & \sum_{0 \le r \le N_f - 1,\ 0 \le i \le 15} w_{h_f} \cdot (\texttt{DXisFilter}[r][i] + \texttt{DWisFilter}[r][i]) + c_{T_2}, \end{aligned}$$

where $\sum_{0 \le r \le N_f - 1,\ 0 \le i \le 15} w_{h_f} \cdot (\texttt{DXisFilter}[r][i] + \texttt{DWisFilter}[r][i])$ corresponds to the total filter $2(n - r_f) + 2h_f$ according to Equation (10), and c_{T_2} indicates a constant factor 2^n.

For T_3, we have $T_3 = c_{T_3} - \texttt{Adv}$, where $c_{T_3} = \tilde{n}$ for SKINNY-n-$\tilde{n}$.
For the advantage h and x, we have constraints:

$$\begin{cases} \texttt{x} \leq \sum_{0\leq r\leq N_b-2,\ 0\leq i\leq 7} \texttt{KnownEnc}[r][i] + \sum_{0\leq r\leq N_f-1,\ 0\leq i\leq 7} \texttt{DXGuess}[r][i], \\ \texttt{Adv} + \texttt{x} \leq \sum_{0\leq r\leq N_b-2,\ 0\leq i\leq 7} \texttt{KnownEnc}[r][i] + \sum_{0\leq r\leq N_f-1,\ 0\leq i\leq 7} \texttt{KnownDec}[r][i]. \end{cases}$$

So we get a uniformed objective:

$$\text{Minimize } obj,\ obj \geq T_1,\ obj \geq T_2,\ obj \geq T_2. \quad (16)$$

4.3 Comparisons between Qin *et al.*'s Model and Ours

Different from Qin *et al.*'s [52] uniform automatic key-recovery model, which is about the rectangle attack framework by Zhao *et al.* [64], our automatic model for Algorithm 1 needs additional constraints to determine h_f-bit internal states acting as filters and m'_f-bit subtweakey needed to guess in the N_f extended rounds. Moreover, in Qin *et al.*'s [52] model, only the time complexity of (Time II of Zhao *et al.*'s model [64] in Section 2.3) generating quartets is considered. However, in our model we have to consider more time complexity constraints, i.e., Time I, Time II and Time III in Algorithm 1. All these differences lead to better attacks than Qin *et al.*'s attacks. Especially we gain 32-round attack on SKINNY-128-384, while Qin *et al.*'s model only achieves 30 rounds.

4.4 New Distinguishers for SKINNY

With our new model, we add such conditions to the automatic searching model in [30,52] to search for new distinguishers. Due to that different parameters have different coefficients in the formula of the time complexity, we give them different weights to model the objective more accurately. For SKINNY, the maximum probability in the DDT table both for 4-bit S-box and 8-bit S-box is 2^{-2}. Then considering the switching effects similar to [41], we adjust the weight $w_{h_f} = 2w_{m_b} = 2w_{m_f} = 4w_0 = 4w_1 = 8w_m = 8$ for $c = 4$ and $w_{h_f} = 2w_{m_b} = 2w_{m_f} = 8w_0 = 8w_1 = 16w_m = 16$ for $c = 8$. Similarly, the constants c_{T_1} and c_{T_2} are set to 33 and 64 for $c = 4$, and to 65 and 128 for $c = 8$. We use different N_b, N_d and N_f. N_b is chosen from 2 to 4 and N_f is 4 or 5 usually. N_d is chosen based on experience, which is shorter than previous longest distinguishers.

By searching for new truncated upper and lower differentials using the MILP model and get instantiations using the CP model following the open source [30], we obtain new distinguishers for SKINNY-128-384, SKINNY-64-192 and SKINNY-128-256. For SKINNY-64-128, we find the distinguisher in [52] is optimal. To get more accurate probabilities of the distinguishers, we calculate the probability $\tilde{p}$ and $\tilde{q}$ considering the clustering effect. For the middle part, we evaluate the probability using the method in [57,41,30] and experimentally verify the probability. The experiments use one computer equipped with one RTX 2080 Ti and the results of our experiments are listed in Table 3. Our source codes are based on the open source by Delaune, Derbez and Vavrille [30], which is provided in https://github.com/key-guess-rectangle/key-guess-rectangle.

Table 3: Experiments on the middle part of boomerang distinguishers for SKINNY.

Version	N_d	r_m	Probability t	Complexity	Time
64-192	22	6	$2^{-17.88}$	2^{30}	21.9s
128-384	23	3	$2^{-20.51}$	2^{31}	30.6s
128-256	18	4	$2^{-35.41}$	2^{40}	16231.8s
128-256	19	4	$2^{-26.71}$	2^{35}	481.2s

We list the 23-round boomerang distinguisher for SKINNY-128-384 in Table 4. For more details of the boomerang distinguishers for other versions of SKINNY, we refer to Section J of the full version of the paper. In addition, we summarize the previous boomerang distinguishers for a few versions of SKINNY in Table 5.

Table 4: The 23-round related-tweakey boomerang distinguisher on SKINNY-128-384.

$r_0 = 11$, $r_m = 3$, $r_1 = 9$, $\tilde{p} = 2^{-32.18}$, $t = 2^{-20.51}$, $\tilde{q} = 2^{-15.11}$, $\tilde{p}^2 t \tilde{q}^2 = 2^{-115.09}$
$\Delta TK1 =$ 00, 00, 00, 00, 00, 00, 00, 00, 24, 00, 00, 00, 00, 00, 00, 00
$\Delta TK2 =$ 00, 00, 00, 00, 00, 00, 00, 00, 07, 00, 00, 00, 00, 00, 00, 00
$\Delta TK3 =$ 00, 00, 00, 00, 00, 00, 00, 00, e3, 00, 00, 00, 00, 00, 00, 00
$\Delta X_0 =$ 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 20
$\nabla TK1 =$ 00, 8a, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
$\nabla TK2 =$ 00, 0c, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
$\nabla TK3 =$ 00, 7f, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
$\nabla X_{23} =$ 00, 00, 00, 00, 00, 00, 00, 00, 00, 50, 00, 00, 00, 00, 00, 00

5 Improved Attacks on SKINNY

In this section, we give the first 32-round attack on SKINNY-128-384 using the distinguisher in Section 4.4 with our new rectangle attack framework. We also give improved attacks on other versions (n-$2n$ and n-$3n$). For more details, please refer to Section D in the full version of the paper.

5.1 Improved Attack on 32-round SKINNY-128-384

We use the 23-round rectangle distinguisher for SKINNY-128-384 given in Table 4, whose probability is $2^{-n}\tilde{p}^2 t \tilde{q}^2 = 2^{-128-115.09} = 2^{-243.09}$. Prepending 4-round

Table 5: Summary of related-tweakey boomerang distinguishers for SKINNY. N_d is the round of distinguishers; $N_b + N_d + N_f$ is the total attacked round.

Version	N_d	Probability $\tilde{p}^2\tilde{q}^2t$	$N_b + N_d + N_f$	Ref.
64-128	17	$2^{-29.78}$	-	[57]
	17	$2^{-48.72}$	21	[47]
	19	$2^{-51.08}$	23	[41]
	19	$2^{-54.36}$	-	[30]
	18	$2^{-55.34}$	24	[52]
	18	$2^{-55.34}$	25	Ours
64-192	22	$2^{-42.98}$	-	[57]
	22	$2^{-54.94}$	26	[47]
	23	$2^{-55.85}$	29	[41]
	23	$2^{-57.93}$	-	[30]
	22	$2^{-57.73}$	30	[52]
	22	$2^{-57.56}$	31	Ours
128-256	18	$2^{-77.83}$	-	[57]
	18	$2^{-103.84}$	22	[47]
	20	$2^{-85.77}$	-	[30]
	21	$2^{-116.43}$	24	[41]
	19	$2^{-116.97}$	25	[52]
	18	$2^{-108.51}$	25	Ours
	19	$2^{-121.07}$	26	Ours
128-384	22	$2^{-48.30}$	-	[57]
	23	2^{-112}	27	[47]
	23	2^{-112}	28	[64]
	24	$2^{-86.09}$	-	[30]
	25	$2^{-116.59}$	30	[41]
	22	$2^{-101.49}$	30	[52]
	23	$2^{-115.09}$	32	Ours

E_b and appending 5-round E_f, we attack 32-round SKINNY-128-384 as illustrated in Figure 9. As introduced in Section 2, the numbers of active bits of the plaintext and ciphertext are denoted as r_b and r_f, and the numbers of subkey bits involved in E_b and E_f are denoted as m_b and m_f. In the first round, we use subtweakey $ETK_0 = \texttt{MC} \circ \texttt{SR}(STK_0)$ instead of STK_0, and there is $ETK_0[i] = ETK_0[i+4] = ETK_0[i+12] = STK_0[i]$ for $0 \le i \le 3$. So we have $r_b = 12 \cdot 8 = 96$ by W_0'. As shown in Figure 9, the ▨ cells are needed to be guessed in E_b, including 3 ▨ cells in STK_2, 7 ▨ cells in STK_1, 8 ▨ cells in ETK_0. Hence, $m_b = 18 \cdot 8 = 144$. In the E_f, we have $r_f = 16 \cdot 8 = 128$ and $m_f = 24 \cdot 8 = 192$. There are 7 cells in STK_{31} and 4 cells STK_{30} marked by red boxes to be guessed in advance, i.e., $m_f' = 11 \cdot 8 = 88$. Then, we get 8 cells in the internal states (marked by red boxes in W_{30}, W_{29} and X_{29}) as additional filters with the guessed m_f'-bit key, i.e., $h_f = 8 \cdot 8 = 64$. Due to the tweakey schedule, we deduce $STK_{28}[3, 7]$ from $ETK_0[1, 0]$, $STK_2[0, 2]$ and $STK_{30}[7, 1]$. So there are only $(m_f - 2c) = 176$-bit subtweakey unknown in E_f after m_b-bit key is guessed in E_b.

As shown in Table 6, we have $k'_f = \{STK_{30}[1,3,5,7], STK_{31}[0,2,3,4,5,6,7]\}$ marked in red indexes and $h_f = \{X_{29}[11], W_{29}[5,7,13,15], W_{30}[5,8,15]\}$ marked in bold. Finally, we give the attack according to Algorithm 1 as follows:

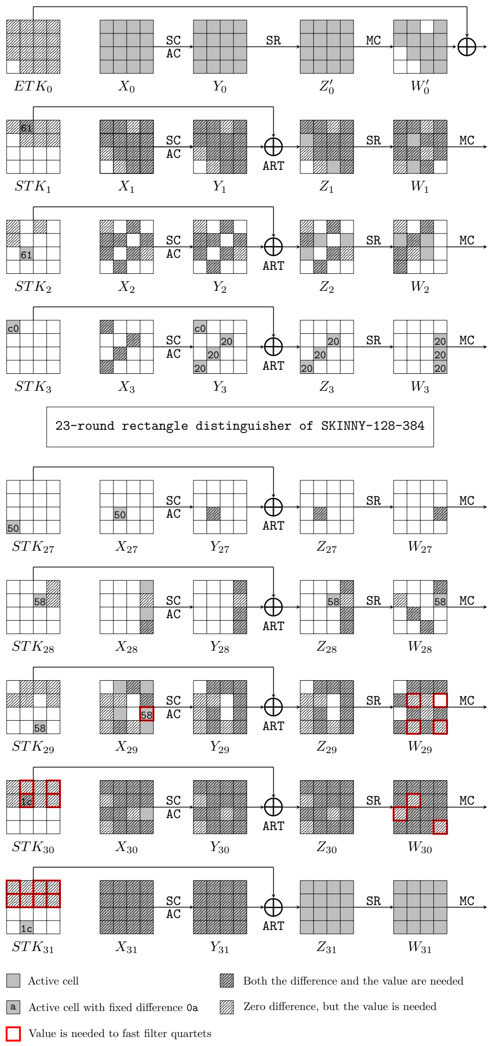

Fig. 9: The 32-round attack against `SKINNY-128-384`.

1. Construct $y = \sqrt{s} \cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2 t \tilde{q}^2} = \sqrt{s} \cdot 2^{25.54}$ structures of $2^{r_b} = 2^{96}$ plaintexts each according to Eq. (7). For each structure, query the 2^{96} ciphertexts by encryptions under K_1, K_2, K_3 and K_4. Hence, the data com-

Table 6: Internal state used for filtering and involved subtweakeys for 32-round SKINNY-128-384.

	Round	Filter	Involved subtweakeys
1	30	$\boldsymbol{\Delta W_{30}[5]=0}$	$STK_{31}[5]$
2		$\boldsymbol{\Delta W_{30}[8]=0}$	$STK_{31}[4]$
3		$\boldsymbol{\Delta W_{30}[15]=0}$	$STK_{31}[3]$
4	29	$\boldsymbol{\Delta W_{29}[5]=0}$	$STK_{30}[5]$, $STK_{31}[0,6,7]$
5		$\boldsymbol{\Delta W_{29}[7]=0}$	$STK_{30}[7]$, $STK_{31}[2,4,5]$
6		$\Delta W_{29}[10]=0$	$STK_{30}[6]$, $STK_{31}[1,7]$
7		$\boldsymbol{\Delta W_{29}[13]=0}$	$STK_{30}[1]$, $STK_{31}[0,5]$
8		$\boldsymbol{\Delta W_{29}[15]=0}$	$STK_{30}[3]$, $STK_{31}[2,7]$
9		$\boldsymbol{\Delta X_{29}[11]=}$ `0x58`	$STK_{30}[5]$, $STK_{31}[0,6]$
10	28	$\Delta W_{28}[5]=0$	$STK_{29}[5]$, $STK_{30}[0,6,7]$, $STK_{31}[1,2,3,4,7]$
11		$\Delta W_{28}[11]=0$	$STK_{29}[7]$, $STK_{30}[2,4]$, $STK_{31}[1,3,5,6]$
12		$\Delta W_{28}[13]=0$	$STK_{29}[1]$, $STK_{30}[0,5]$, $STK_{31}[3,4,6]$
13		$\Delta W_{28}[15]=0$	$STK_{29}[3]$, $STK_{30}[2,7]$, $STK_{31}[1,4,6]$
14	27	$\Delta W_{27}[7]=0$	$STK_{28}[7]$, $STK_{29}[2,4,5]$, $STK_{30}[0,1,3,5,6]$, $STK_{31}[0,1,2,3,4,5,6,7]$
15		$\Delta W_{27}[15]=0$	$STK_{28}[3]$, $STK_{29}[2,7]$, $STK_{30}[1,4,6]$, $STK_{31}[0,3,5,6,7]$
16		$\Delta X_{27}[9]=$ `0x50`	$STK_{28}[7]$,$STK_{29}[2,4]$, $STK_{30}[1,3,5,6]$, $STK_{31}[0,1,2,5,6,7]$

plexity is $\sqrt{s}\cdot 2^{n/2+2}/\sqrt{\tilde{p}^2 t\tilde{q}^2} = \sqrt{s}\cdot 2^{123.54}$ according to Eq. (8). The memory complexity in this step is also $\sqrt{s}\cdot 2^{123.54}$.

2. Guess x-bit key (part of the k_b and k'_f involved in E_b and E_f):
   (a) Initialize a list of $2^{m_b+m_f-2c-x} = 2^{320-x}$ counters. The memory complexity in this step is 2^{320-x}.
   (b) Guess $(m_b + m'_f - x) = (232 - x)$-bit key involved in E_b and E_f:
      i. In each structure, we partially encrypt P_1 under m_b-bit subkey to the positions of known differences of Y_3, and partially decrypt it to the plaintext P_2 (within the same structure) after xoring the known difference α. The details can refer to Section C in the full version of the paper. Do the same for each P_3 to get P_4. Store the pairs in S_1 and S_2. Totally, $m_b = 18\cdot 8 = 144$-bit key are involved.
      ii. The size of S_1 and S_2 is $y\cdot 2^{r_b} = \sqrt{s}\cdot 2^{121.54}$. For each element in S_1, with $m'_f = 88$-bit k'_f, we can obtain $2h_f = 2\cdot 64 = 128$ internal state bits as filters. So partially decrypt (C_1, C_2) in S_1 with k'_f to get $\{W_{30}[5,8,15], W_{29}[5,7,13,15], X_{29}[11]\}$ as filters. Insert the element in S_1 into a hash table H indexed by the $h_f =$ 64-bit $\{W_{30}[5,8,15], W_{29}[5,7,13,15], X_{29}[11]\}$ of C_1 and $h_f =$ 64-bit $\{\bar{W}_{30}[5,8,15], \bar{W}_{29}[5,7,13,15], \bar{X}_{29}[11]\}$ of C_2. For each element (C_3, C_4) in S_2, partially decrypt it with k'_f to get the $2h_f = 128$ internal state bits, and check against H to find the pairs (C_1, C_2), where (C_1, C_3) and (C_2, C_4) collide at the $2h_f = 128$ bits. According to Eq. (9), the data collection process needs $T_1 = \sqrt{s}\cdot 2^{m_b+m'_f+n/2+1}/\sqrt{\tilde{p}^2 t\tilde{q}^2} = \sqrt{s}\cdot 2^{144+88+64+1+57.54} = \sqrt{s}\cdot 2^{354.54}$. We get $s\cdot 2^{m_b+m'_f-2h_f-n+2r_f}/(\tilde{p}^2 t\tilde{q}^2) = s\cdot 2^{144+88-128+128+115.09} = s\cdot 2^{347.09}$ quartets according to Eq. (11).

iii. On ε: for each of $s \cdot 2^{347.09}$ quartets, determine the key candidates and increase the corresponding counters. According to Eq. (11), this step needs $T_2 = s \cdot 2^{347.09} \cdot \varepsilon$. We refer the readers to Table 7 to make the following guess-and-filter steps clearer.

A. **In round 31:** guessing $STK_{31}[1]$ and together with k'_f as shown in Table 7, we compute $Z_{30}[6,14]$ and peel off round 31. Then $\Delta Y_{30}[6]$ and $\Delta X_{30}[14]$ are deduced. For the 3rd column of X_{30} of (C_1, C_3), we obtain $\Delta X_{30}[6] = \Delta X_{30}[14]$ from Eq. (15). Hence, we obtain $\Delta X_{30}[6]$ and deduce $STK_{30}[6]$ by Lemma 1. Similarly, we deduce $STK'_{30}[6]$ for (C_2, C_4). Since $\Delta STK_{30}[6]$ is fixed, we get an 8-bit filter. $s \cdot 2^{347.09} \cdot 2^8 \cdot 2^{-8} = s \cdot 2^{347.09}$ quartets remain.

B. **In round 30:** guessing $STK_{30}[0]$, we compute $Z_{29}[1,9,13]$ as shown in Table 7. Then $\Delta Y_{29}[1]$ and $\Delta X_{29}[9,13]$ are deduced. For the 2nd column of X_{29} of (C_1, C_3), we can obtain $\Delta X_{29}[1] = \Delta X_{29}[9] = \Delta X_{29}[13]$. Hence, we obtain $\Delta X_{29}[1]$ and deduce $STK_{29}[1]$. Similarly, we deduce $STK'_{29}[1]$ for (C_2, C_4), which is an 8-bit filter. For both (C_1, C_3) and (C_2, C_4), $\Delta X_{29}[9] = \Delta X_{29}[13]$ is an 8-bit filter. $s \cdot 2^{347.09} \cdot 2^8 \cdot 2^{-8} \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{331.09}$ quartets remain.

C. Guessing $STK_{30}[2,4]$, we compute $Z_{29}[3,7,15]$ and peel off round 30. Then $\Delta Y_{29}[3,7]$ and $\Delta X_{29}[15]$ are deduced. For the 4th column of X_{29} of (C_1, C_3), we can obtain $\Delta X_{29}[3] = \Delta X_{29}[7] = \Delta X_{29}[15]$. Hence, we obtain $\Delta X_{29}[3,7]$ and deduce $STK_{29}[3,7]$. Similarly, we deduce $STK'_{29}[3,7]$ for (C_2, C_4), which is a 16-bit filter. $s \cdot 2^{331.09} \cdot 2^{16} \cdot 2^{-16} = s \cdot 2^{331.09}$ quartets remain.

D. **In round 29:** guessing $STK_{29}[2,5]$, we compute $Z_{28}[3,11,15]$. Then $\Delta Y_{28}[3]$ and $\Delta X_{28}[11,15]$ are deduced. For the 4th column of X_{28} of (C_1, C_3), we can obtain $\Delta X_{28}[3] = \Delta X_{28}[11] = \Delta X_{28}[15]$. Since $STK_{28}[3]$ can be deduced from the known $ETK_0[1]$, $STK_2[0]$ and $STK_{30}[7]$, we can compute $X_{28}[3]$ and $\Delta X_{28}[3]$. For both (C_1, C_3) and (C_2, C_4), $\Delta X_{28}[3] = \Delta X_{28}[15]$ and $\Delta X_{28}[11] = \Delta X_{28}[15]$ are two 8-bit filter. $s \cdot 2^{331.09} \cdot 2^{16} \cdot 2^{-16} \cdot 2^{-16} = s \cdot 2^{315.09}$ quartets remain.

E. Guessing $STK_{29}[4]$, we decrypt two rounds to get $X_{27}[9]$ with known $STK_{28}[7]$. **In round 27**, $\Delta X_{27}[9] = \texttt{0x50}$ is an 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{315.09} \cdot 2^8 \cdot 2^{-16} = s \cdot 2^{307.09}$ quartets remain.

So for each quartet, $\varepsilon = 2^8 \cdot \frac{4}{32} + 2^8 \cdot \frac{4}{32} + 2^{-16} \cdot 2^{16} \cdot \frac{4}{32} + 2^{-16} \cdot 2^{16} \cdot \frac{4}{32} + 2^{-32} \cdot 2^8 \cdot \frac{8}{32} \approx 2^{6.01}$ and $T_2 = s \cdot 2^{353.1}$.

(c) (Exhaustive search) Select the top $2^{m_b + m_f - 2c - x - h} = 2^{320 - x - h}$ hits in the counter as the key candidates. Guess the remaining $k - (m_b + m_f - 2c) = 64$-bit key to check the full key. According to Eq. (12), $T_3 = 2^{k-h}$.

In order to balance T_1, T_2, T_3 and memory complexity and achieve a high success probability, we set the excepted number of right quartets $s = 1$, the advantage $h = 40$ and $x = 208$ ($x \leq m_b + m'_f = 232$, $h \leq m_b + m_f - 2c - x =$

$320 - x$) with Eq. (14). Then we have $T_1 = 2^{354.54}$, $T_2 = 2^{353.1}$ and $T_3 = 2^{344}$. In total, the data complexity is $2^{123.54}$, the memory complexity is $2^{123.54}$, and the time complexity is $2^{354.99}$. The success probability is about 82.1%.

Table 7: Tweakey recovery for 32-round `SKINNY-128-384`, where the red bytes are among k'_f or obtained in the previous steps.

Step	Internal state	Involved subtweakeys
A	$Z_{30}[6]$	$STK_{31}[7]$
	$Z_{30}[14]$	$STK_{31}[1]$
B	$Z_{29}[1]$	$STK_{30}[5]$, $STK_{31}[6]$
	$Z_{29}[9]$	$STK_{30}[7]$, $STK_{31}[2,4]$
	$Z_{29}[13]$	$STK_{30}[0]$, $STK_{31}[3,4]$
C	$Z_{29}[3]$	$STK_{30}[7]$, $STK_{31}[4]$
	$Z_{29}[7]$	$STK_{30}[4]$, $STK_{31}[3,5,6]$
	$Z_{29}[15]$	$STK_{30}[2]$, $STK_{31}[1,6]$
D	$Z_{28}[3]$	$STK_{29}[7]$, $STK_{30}[4]$, $STK_{31}[3,5,6]$
	$Z_{28}[11]$	$STK_{29}[5]$, $STK_{30}[0,6]$, $STK_{31}[1,3,4,7]$
	$Z_{28}[15]$	$STK_{29}[2]$, $STK_{30}[1,6]$, $STK_{31}[0,5,7]$
E	$X_{27}[9]$	$STK_{28}[7]$,$STK_{29}[2,4]$, $STK_{30}[1,3,5,6]$, $STK_{31}[0,1,2,5,6,7]$

6 Conclusion and Further Disscussion

We introduce a new key-recovery framework for the rectangle attacks on ciphers with linear schedule with the purpose of reducing the overall complexity or attacking more rounds. We give a uniform automatic model on `SKINNY` to search for distinguishers which are more proper for our key-recovery framework. With the new rectangle distinguishers, we give new attacks on a few versions of `SKINNY`, which achieve 1 or 2 more rounds than the best previous attacks.

Further discussion. For `ForkSkinny`, `Deoxys-BC` and `GIFT`, we do not give the automatic models but only apply our new rectangle attack framework in Algorithm 1 with the previous distinguishers. For `ForkSkinny`, we find that the 21-round distinguisher on `ForkSkinny`-128-256 in [52] is also optimal for our new rectangle attack model. Our attack on 28-round `ForkSkinny-128-256` with 256-bit key reduces the time complexity of [52] by a factor of 2^{22}. For `Deoxys-BC-384`, our attack reduces the time complexity of the best previous 14-round attack [63] by a factor of $2^{22.7}$ with similar data complexity. For `GIFT-64`, our rectangle attack uses the same rectangle distinguisher with Ji *et al.* [44], but achieves one more round. Moreover, compared with the best previous attack achieved by differential attack by Sun *et al.* [58], our rectangle attack achieves the same 26 rounds. The details can refer to Section F, G, H of the full version of the paper.

For single-key setting, our tradeoff key-recovery model in Section 3 and Zhao *et al.*'s model [62] can be trivially converted into the single-key model by just letting the differences of the keys be 0. We also give an attack on 10-round `Serpent` reusing the rectangle distinguisher by Biham, Dunkelman and Keller [13] and achieving better time complexity (see Section I in the full version of the paper).

Overall analysis of the four attack models. To better understand different key-recovery rectangle models, we give an overall analysis of the four attack models in Section 2 and 3. There are some differences in the four models:

- The `Attack I` of Section 2.1 guesses all the $(m_b + m_f)$-bit key at once and generates the quartets;
- The `Attack II` of Section 2.2 does not guess the key involved in E_b and E_f when generating quartets, and uses hash tables in the key-recovery process.
- The `Attack III` of Section 2.3 only guesses m_b-bit key in E_b to generate quartets and the key-recovery process is just a guess and filter process.
- Our new attack of Section 3.1 guesses m_b-bit key in E_b and m'_f-bit key in E_f to generate quartets, which increases the time of generating quartets but reduces the number of quartets to be checked in the key-recovery process.

For all the attack models, the data complexities are the same, which depend on the the probability of the rectangle distinguisher and the expected number of right quartets s. To analyze different time complexities, we first compare time complexities of the key-recovery process. Suppose, $\hat{p}\hat{q} = 2^{-t}$ and s is small and ignored, we approximate the four complexities to be

$$\begin{cases} \texttt{Attack I} : T_{\texttt{I}} = 2^{m_b+m_f+n/2+t+2}, & (17)\\ \texttt{Attack II} : T_{\texttt{II}} = 2^{m_b+r_b+2r_f-n+2t} + 2^{m_f+2r_b+r_f-n+2t}, & (18)\\ \texttt{Attack III} : T_{\texttt{III}} = 2^{m_b+2r_f-n+2t} \cdot \varepsilon, & (19)\\ \texttt{Attack IV} : T_{\texttt{IV}} = 2^{m_b+2r_f-n+m'_f-2h_f+2t} \cdot \varepsilon. & (20) \end{cases}$$

To compare $T_{\texttt{II}}$ and $T_{\texttt{III}}$, when $\varepsilon \leq 2^{r_b}$, the complexity of `Attack III` is lower than `Attack II`. In the key-recovery process of `Attack III`, the early abort technique [49] is usually applied to make the ε very small, i.e., the key-recovery phase on 32-round `SKINNY-128-384`.

To compare $T_{\texttt{III}}$ and $T_{\texttt{IV}}$, when $m'_f - 2h_f \leq 0$, the complexity of `Attack IV` is lower than `Attack III`. For the attack where an h_f-bit filter with an m'_f-bit guessed subkey satisfy $m'_f - 2h_f \leq 0$, `Attack IV` is better than `Attack III`.

To compare $T_{\texttt{I}}$, $T_{\texttt{II}}$ and $T_{\texttt{III}}$, we assume that the probability $\hat{p}^2\hat{q}^2$ is larger than 2^{-n} but the gap is small. Then $n/2+t$ can be approximated by n and $2t \approx n$. Thereafter, the complexities can be further estimated as $2^{m_b+m_f+n+2}$ for `Attack I`, $2^{m_b+r_b+2r_f} + 2^{m_f+2r_b+r_f}$ for `Attack II` and $2^{m_b+2r_f} \cdot \varepsilon$ for `Attack III`. When $2^{2r_f} \cdot \varepsilon < 2^{m_f+n+2}$, the complexity of `Attack III` is lower than `Attack I`. When $r_b + 2r_f < m_f + n + 2$ and $2r_b + r_f < m_b + n + 2$, the complexity of `Attack II` is lower than `Attack I`.

Hence, different models perform differently for different parameters.

Future work. Generally, the model is suitable for most block ciphers with linear key schedule. In fact, we also apply our method to `CRAFT` [8] and `Saturnin` [26]. For `CRAFT`, we find a better rectangle attack. However, the attack is inferior to the attack proposed in [40]. For `Saturnin`, we failed to get any improved attack. We plan to further investigate how to improve the current attacks by applying a more complicated key-bridging technique [36]. For example, in the 32-round attack on `SKINNY`, "we deduce $STK_{28}[3,7]$ from $ETK_0[1,0]$, $STK_2[0,2]$ and $STK_{30}[7,1]$". The current automatic model does not cover the key-bridging technique. Future work is to adopt this technique into the automatic model to find more effective key relations.

Acknowledgments. This work is supported by National Key R&D Program of China (2018YFA0704701, 2018YFA0704704), the Major Program of Guangdong Basic and Applied Research (2019B030302008), Major Scientific and Technological Innovation Project of Shandong Province, China (2019JZZY010133), Natural Science Foundation of China (61902207, 61772519, 62032014, 62072270) and the Chinese Major Program of National Cryptography Development Foundation (MMJJ20180101, MMJJ20180102).

References

1. 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 *ASIACRYPT 2019, Proceedings, Part II*, pages 153–182.
2. Elena Andreeva, Virginie Lallemand, Antoon Purnal, Reza Reyhanitabar, Arnab Roy, and Damian Vizár. ForkAE v. *Submission to NIST Lightweight Cryptography Project*, 2019.
3. Ralph Ankele, Subhadeep Banik, Avik Chakraborti, Eik List, Florian Mendel, Siang Meng Sim, and Gaoli Wang. Related-key impossible-differential attack on reduced-round SKINNY. In *ACNS 2017*, volume 10355, pages 208–228.
4. Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In *CHES 2017, Proceedings*, volume 10529, pages 321–345.
5. Achiya Bar-On, Orr Dunkelman, Nathan Keller, and Ariel Weizman. DLCT: A new tool for differential-linear cryptanalysis. In *EUROCRYPT 2019, Proceedings, Part I*, volume 11476, pages 313–342.
6. Augustin Bariant, Nicolas David, and Gaëtan Leurent. Cryptanalysis of Forkciphers. *IACR Trans. Symmetric Cryptol.*, 2020(1):233–265, 2020.
7. 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 *CRYPTO 2016, Proceedings, Part II*, pages 123–153.
8. 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.
9. Tim Beyne. Block cipher invariants as eigenvectors of correlation matrices. *J. Cryptol.*, 33(3):1156–1183, 2020.

10. Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In *FSE '98, Proceedings*, pages 222–238.
11. Eli Biham, Orr Dunkelman, and Nathan Keller. New cryptanalytic results on IDEA. In *ASIACRYPT 2006, Proceedings*, pages 412–427.
12. Eli Biham, Orr Dunkelman, and Nathan Keller. New results on boomerang and rectangle attacks. In *FSE 2002, Revised Papers*, volume 2365, pages 1–16.
13. Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack - rectangling the serpent. In *EUROCRYPT 2001, Proceeding*, volume 2045, pages 340–357.
14. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In *EUROCRYPT 2005, Proceedings*, volume 3494, pages 507–525.
15. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.
16. Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In *CRYPTO 2003, Proceedings*, volume 2729, pages 195–211.
17. Alex Biryukov, Luan Cardoso dos Santos, Daniel Feher, Vesselin Velichkov, and Giuseppe Vitto. Automated truncation of differential trails and trail clustering in ARX. Cryptology ePrint Archive, Report 2021/1194.
18. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT 2009*, volume 5912, pages 1–18.
19. Alex Biryukov and Ivica Nikolic. Automatic search for related-key differential characteristics in byte-oriented block ciphers: Application to aes, camellia, khazad and others. In *EUROCRYPT 2010, Proceedings*, pages 322–344.
20. Alex Biryukov and Vesselin Velichkov. Automatic search for differential trails in ARX ciphers. In *CT-RSA 2014*, volume 8366, pages 227–250.
21. Xavier Bonnetain, Léo Perrin, and Shizhu Tian. Anomalies and vector space search: Tools for S-box analysis. In *ASIACRYPT 2019, Part I*, volume 11921, pages 196–223.
22. 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.
23. Christina Boura and Anne Canteaut. On the boomerang uniformity of cryptographic sboxes. *IACR Trans. Symmetric Cryptol.*, 2018(3):290–310, 2018.
24. Christina Boura, Virginie Lallemand, María Naya-Plasencia, and Valentin Suder. Making the impossible possible. *J. Cryptol.*, 31(1):101–133, 2018.
25. Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In *ASIACRYPT 2014, Part I*, volume 8873, pages 179–199.
26. Anne Canteaut, Sébastien Duval, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, Thomas Pornin, and André Schrottenloher. Saturnin: a suite of lightweight symmetric algorithms for post-quantum security. *IACR Trans. Symmetric Cryptol.*, 2020(S1):160–207, 2020.
27. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In *CRYPTO 2013, Proceedings, Part I*, volume 8042, pages 222–240.
28. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. Boomerang connectivity table: A new cryptanalysis tool. In *EUROCRYPT 2018, Proceedings, Part II*, volume 10821, pages 683–714.
29. Carlos Cid, Tao Huang, Thomas Peyrin, Yu Sasaki, and Ling Song. A security analysis of Deoxys and its internal tweakable block ciphers. *IACR Trans. Symmetric Cryptol.*, 2017(3):73–107, 2017.

30. Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs application to SKINNY. *IACR Trans. Symmetric Cryptol.*, 2020(4):104–129, 2020.
31. Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In *CRYPTO 2016, Part II*, pages 157–184.
32. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In *EUROCRYPT 2013, Proceedings*, pages 371–387.
33. Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. Cryptology ePrint Archive, Report 2021/856, 2021. https://ia.cr/2021/856.
34. Orr Dunkelman, Sebastiaan Indesteege, and Nathan Keller. A differential-linear attack on 12-round Serpent. In *INDOCRYPT 2008*, volume 5365, pages 308–321.
35. Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. The retracing boomerang attack. In *EUROCRYPT 2020, Part I*, volume 12105, pages 280–309.
36. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In *ASIACRYPT 2010, Proceedings*, pages 158–176.
37. 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, Proceedings*, volume 6223, pages 393–410.
38. Orr Dunkelman, Nathan Keller, and Adi Shamir. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. *J. Cryptology*, 27(4):824–849, 2014.
39. Antonio Flórez-Gutiérrez and María Naya-Plasencia. Improving key-recovery in linear attacks: Application to 28-round PRESENT. In *EUROCRYPT 2020, Proceedings, Part I*, pages 221–249.
40. 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.
41. Hosein Hadipour, Nasour Bagheri, and Ling Song. Improved rectangle attacks on SKINNY and CRAFT. *IACR Trans. Symmetric Cryptol.*, 2021(2):140–198, 2021.
42. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In *ASIACRYPT 2014, Proceedings, Part II*, volume 8874, pages 274–288.
43. Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Submission to CAESAR : Deoxys v1.41, October 2016. http://competitions.cr.yp.to/round3/deoxysv141.pdf.
44. Fulei Ji, Wentao Zhang, Chunning Zhou, and Tianyou Ding. Improved (related-key) differential cryptanalysis on GIFT. *Accepted to SAC 2020*.
45. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round MARS and Serpent. In *FSE 2000*, volume 1978, pages 75–93.
46. Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In *CRYPTO 2015, Part I*, volume 9215, pages 161–185.
47. Guozhen Liu, Mohona Ghosh, and Ling Song. Security analysis of SKINNY under related-tweakey settings. *IACR Transactions on Symmetric Cryptology*, 2017(3):37–72, 2017.
48. Meicheng Liu, Xiaojuan Lu, and Dongdai Lin. Differential-linear cryptanalysis from an algebraic perspective. In *CRYPTO 2021, Part III*, volume 12827, pages 247–277.

49. Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman. Improving the efficiency of impossible differential cryptanalysis of reduced camellia and MISTY1. In *CT-RSA 2008, Proceedings*, volume 4964, pages 370–386.
50. Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Inscrypt 2011, Revised Selected Papers*, pages 57–76.
51. Sean Murphy. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory*, 57(4):2517–2521, 2011.
52. 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.
53. Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. *IACR Transactions on Symmetric Cryptology*, 2018(3):124–162, 2018.
54. Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In *EUROCRYPT 2017, Proceedings, Part III*, volume 10212, pages 185–215.
55. Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *J. Cryptology*, 21(1):131–147, 2008.
56. 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 *ASIACRYPT 2018, Proceedings, Part II*, volume 11273, pages 3–34.
57. Ling Song, Xianrui Qin, and Lei Hu. Boomerang connectivity table revisited. application to SKINNY and AES. *IACR Transactions on Symmetric Cryptology*, 2019(1):118–141, 2019.
58. Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Trans. Symmetric Cryptol.*, 2021(1):269–315, 2021.
59. 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 *ASIACRYPT 2014, Proceedings, Part I*, pages 158–178.
60. David A. Wagner. The boomerang attack. In *FSE '99, Proceedings*, volume 1636, pages 156–170.
61. 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.
62. Boxin Zhao, Xiaoyang Dong, and Keting Jia. New related-tweakey boomerang and rectangle attacks on Deoxys-BC including BDT effect. *IACR Trans. Symmetric Cryptol.*, 2019(3):121–151, 2019.
63. Boxin Zhao, Xiaoyang Dong, Keting Jia, and Willi Meier. Improved related-tweakey rectangle attacks on reduced-round Deoxys-BC-384 and Deoxys-I-256-128. In *INDOCRYPT 2019, Proceedings*, pages 139–159.
64. 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.