Key Guessing Strategies for Linear Key-Schedule Algorithms in Rectangle Attacks

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] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China.
siweisun.isaac@gmail.com
[3] University of Chinese Academy of Sciences, Beijing, China.
[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. In building boomerang distinguishers, Murphy indicated that two independently chosen differentials for a boomerang may be incompatible. In this paper, we find that similar incompatibility also happens to key-recovery phase. When generating quartets for the rectangle attack on linear key-schedule ciphers, we find that the right quartets which may suggest key candidates have to satisfy some nonlinear relationships. However, some quartets generated always violate these relationships, so that they will never suggest any key candidates. We call those quartets as *nonlinearly incompatible* quartets. Inspired by previous rectangle frameworks, we find that guessing certain key cells before generating quartets may reduce the number of *nonlinearly incompatible* quartets. However, guessing a lot of key cells at once may lose the benefit from the guess-and-filter or early abort technique, which may lead to a higher overall complexity. To get better tradeoff from the two aspects, we build a new rectangle attack framework on linear key-schedule ciphers with the purpose of reducing the overall complexity or attacking more rounds.

The first application is on `SKINNY`. In the tradeoff model, there are many parameters affecting the overall complexity. We build a uniform automatic model on `SKINNY` to identify all the optimal parameters, 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 model, we identify a 32-round key-recovery attack on `SKINNY`-128-384 in related-key setting, which extend 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 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 Introduction

The boomerang attack [49] proposed by Wagner, is an adaptive chosen plaintext and ciphertext attack derived from differential cryptanalysis [14]. 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:

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

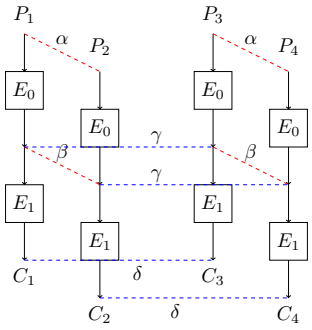

Fig. 1: Boomerang attack.

The adaptive chosen plaintext and ciphertext of boomerang attack can be converted into a chosen-plaintext attack that is known as amplified boomerang attack [34] or rectangle attack [12]. 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

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

The boomerang attack and rectangle attack have been successfully applied to numerous block ciphers, including `Serpent` [12,11], `AES` [16,13], `IDEA` [10], `KASUMI` [28], `Deoxys-BC` [22], etc. Recently, a new variant of boomerang attack was developed and applied to `AES`, named as retracing boomerang attack [26].

There are two steps when applying the boomerang and rectangle attack, i.e., building distinguishers and performing key-recovery attacks. In building distinguishers, Murphy [39] 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.* [15]. To further explore the dependence and increase the probability of boomerang, Biryukov and Khovratovich [16] introduced the *boomerang switch* technique including the *ladder switch* and *S-box switch*. Then, those techniques were generalized and formalized by Dunkelman *et al.* [27,28] as the *sandwich attack*. Recently, Cid *et al.* [21] introduced the boomerang connectivity table (BCT) to clarify the probability around the boundary of boomerang and compute its probability more accurately. Later, various improvements or further studies [17,5,46,50,23] on the BCT technique enrich the boomerang attacks.

Given a distinguisher, we usually need more complicated key-recovery algorithms to identify the right quartets [34,12] when performing the rectangle attack than the boomerang attack. Till now, a series of generalized key-recovery algorithms [12,11,13] 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. On the one hand, it usually allows to determine the security margin of a given cipher more accurately or semi-automatically. On the other hand, more clear understanding on what happens in each step of the algorithm and why such steps are included will come into mind, which will possibly leads to new improvement ideas, such as impossible differential attacks [19,18], linear attacks [29], invariant attacks [8], and meet-in-the-middle attacks [20,25], etc.

Our contributions.

In building boomerang distinguishers, Murphy [39] pointed out that two independently chosen differential characteristics can be incompatible, thus the probability of generating a right quartet can be zero. We find similar incompatibilities in the key-recovery phase of the rectangle attack on ciphers with linear key schedule. When performing rectangle attacks, we usually append 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 differential attack, in rectangle attack we may collect data and generate quartets whose plaintext difference and ciphertext difference meet (α', δ'). Then guess-and-determine or early-abort technique [37] 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 key candidates. In further study, we find the right quartets that suggest key candidates have to meet certain nonlinear relationships. However, many quartets meeting (α', δ') always violate those nonlinear relationships, and thereby never suggest any key candidates. We call those quartets as *nonlinearly incompatible* quartets.

Inspired from the previous rectangle attacks [12,11,51], we find that guessing certain key cells before generating quartets may avoid many *nonlinearly incompatible* quartets in advance. However, guessing a lot of key cells as a whole may lose the advantage of guess-and-determine or early-abort technique [37], 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.

As the first application, we apply the new framework to SKINNY [7]. When evaluating dedicated cipher with the tradeoff framework, we have to identify many attack parameters, such as finding a rectangle distinguisher which is optimal for our new key-recovery attack framework, determining the number and positions of cells of the guessed key before generating quartets, the size of key counters, etc. Hence, in order to launch the optimal key-recovery attack on SKINNY with our framework, we build a uniform automatic model, which is based on a series of automatic tools [23,30,40] on SKINNY proposed recently, to determine a set of optimal parameters affecting the attack complexity or attacked rounds. Note that in the field of automatic cryptanalysis, there are many works focusing on searching distinguishers [38,48,43,35], but only a few works [24,45,40] 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 [40,30]. In addition, for other versions of SKINNY with n-2n or n-3n, one more round is achieved.

As the second application, we perform our new key-recovery framework on reduced ForkSkinny [1], Deoxys-BC-384 [32] and GIFT-64 [4] with some previous proposed distinguishers. All the attacks achieve better complexities than before. At last, we discuss the conversion of our attack framework from related-key setting to single setting. Since our related-key attack framework is on ciphers with linear key-schedule, it is trivial to be converted 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 and achieve slightly better complexity than the previous rectangle attack [11]. 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 attack [12,11,13] introduced before. We briefly recall them with the symbols from [11]. 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 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 [6], respectively. Denote the subset of subkey bits which is involved in E_b as k_b, which affects the difference of the plaintexts by

[6] In [11], the authors also introduced the set of plaintext differences that may cause the difference α after E_b. However, the set highly depends on the differential property of Sbox. For generality and succinctness, we only use the active bits in this paper.

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	[36]
	23	$2^{62.47}$	$2^{125.91}$	2^{124}	ID	RK	[36]
	23	$2^{62.47}$	2^{124}	$2^{77.47}$	ID	RK	[41]
	23	$2^{71.4}$	2^{79}	$2^{64.0}$	ID	RK	[3]
	23	$2^{60.54}$	$2^{120.7}$	$2^{60.9}$	Rectangle	RK	[30]
	24	$2^{61.67}$	$2^{96.83}$	2^{84}	Rectangle	RK	[40]
	25	$2^{61.67}$	$2^{118.43}$	$2^{64.26}$	Rectangle	RK	Sect. C.2
64-192	27	$2^{63.5}$	$2^{165.5}$	2^{80}	Rectangle	RK	[36]
	29	$2^{62.92}$	$2^{181.7}$	2^{80}	Rectangle	RK	[30]
	30	$2^{62.87}$	$2^{163.11}$	$2^{68.05}$	Rectangle	RK	[40]
	31	$2^{62.78}$	$2^{182.07}$	$2^{62.79}$	Rectangle	RK	Sect. C.1
128-256	22	2^{127}	$2^{235.6}$	2^{127}	Rectangle	RK	[36]
	23	$2^{124.47}$	$2^{251.47}$	2^{248}	ID	RK	[36]
	23	$2^{124.41}$	$2^{243.41}$	$2^{155.41}$	ID	RK	[41]
	24	$2^{125.21}$	$2^{209.85}$	$2^{125.54}$	Rectangle	RK	[30]
	25	$2^{124.48}$	$2^{226.38}$	2^{168}	Rectangle	RK	[40]
	25	$2^{120.25}$	$2^{193.91}$	2^{136}	Rectangle	RK	Sect. C.3
	26	$2^{126.53}$	$2^{254.4}$	$2^{128.44}$	Rectangle	RK	Sect. C.4
128-384	27	2^{123}	2^{331}	2^{155}	Rectangle	RK	[36]
	28	2^{122}	$2^{315.25}$	$2^{122.32}$	Rectangle	RK	[53]
	30	$2^{125.29}$	$2^{361.68}$	$2^{125.8}$	Rectangle	RK	[30]
	30	2^{122}	$2^{341.11}$	$2^{128.02}$	Rectangle	RK	[40]
	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	[40]
	28	$2^{118.88}$	$2^{224.76}$	$2^{118.88}$	Rectangle	RK	Sect. E
Deoxys-BC							
128-384	13	2^{127}	2^{270}	2^{144}	Rectangle	RK	[22]
	14	2^{127}	$2^{286.2}$	2^{136}	Rectangle	RK	[51]
	14	$2^{125.2}$	$2^{282.7}$	2^{136}	Rectangle	RK	[52]
	14	$2^{125.2}$	2^{260}	2^{140}	Rectangle	RK	Sect. F
GIFT							
64-128	25	$2^{63.78}$	$2^{120.92}$	$2^{64.1}$	Rectangle	RK	[33]
	26	$2^{60.96}$	$2^{123.23}$	$2^{102.86}$	Differential	RK	[47]
	26	$2^{63.78}$	$2^{112.78}$	$2^{63.78}$	Rectangle	RK	Sect. G

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 $m_f = |k_f|$.

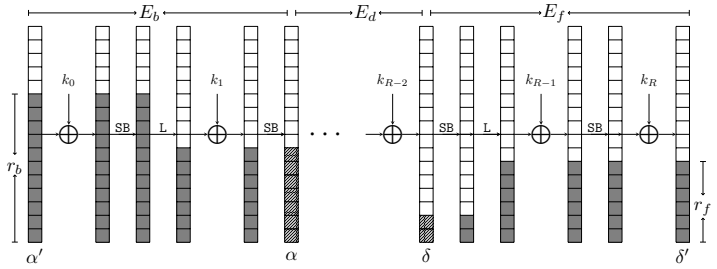

Fig. 2: Framework of rectangle attack on E.

Related-key boomerang and rectangle attacks were proposed by Biham *et al.* in [13]. Assume 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 a boomerang attack is the best known related-key attack on the full versions of `AES-192` and `AES-256`, presented by Biryukov and Khovratovich [16].

As shown by Biham *et al.* [10], when the key schedule is linear, the related-key rectangle attacks is the similar to (with very slightly modification) the single-key rectangle framework. Different from non-linear key schedule, for linear key schedule, the differences between the subkeys of K_1, K_2, K_3 and K_4 are all determined in each round. 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 fixed differences between the subkeys.

In this paper, we focus on the rectangle frameworks on ciphers with linear key schedules. We list the previous frameworks below.

2.1 `Attack I`: Biham-Dunkelman-Keller's attack from EUROCRYPT 2001

At EUROCRYPT 2001, Biham, Dunkelman and Keller introduced the rectangle attack [12] and applied it to the single-key attack on `Serpent` [9]. We trivially convert it to a related-key model with linear key schedule. The procedures are summarized below:

1. Collect and store y structures of 2^{r_b} plaintexts each, by traversing the active bits in each structure.
2. For structure, query the 2^{r_b} plaintexts under K_1, K_2, K_3 and K_4.
3. 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) For each structure, 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 the plaintext P_2 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. Choose

$$y = \sqrt{s} \cdot 2^{n/2-r_b}/\hat{p}\hat{q}, \tag{3}$$

we get about

$$(y \cdot 2^{2r_b})^2 \cdot 2^{-2r_b} \cdot 2^{-n}\hat{p}^2\hat{q}^2 = s \tag{4}$$

quartets for the right $(m_b + m_f)$-bit. Therefore, the total data complexity for the 4 oracles with K_1, K_2, K_3 and K_4 is

$$4y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/\hat{p}\hat{q}. \tag{5}$$

In Step 3, 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 from FSE 2002

At FSE 2002, Biham, Dunkelman and Keller introduced a more generic algorithm to perform the rectangle attack [11] in single-key setting. Later, Liu *et al.* [36] 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 procedures are briefly summarized below and for more details please refer to [11]:

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.
2. Initialize an array of $2^{m_b+m_f}$ counters, where each corresponds to a $(m_b + m_f)$-bit subkey guess.
3. Insert the $y \cdot 2^{r_b}$ 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 to Eq. (12) given at `Attack I`, with the same y given by Eq. (10).

- **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.* [51,53] proposed a new generalized related-key rectangle attacks as shown below:

1. Construct y structures of 2^{r_b} plaintexts each. For 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) For each structure, 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 in the following:

$$S_1 = \{(P_1, C_1, P_2, C_2) : E_{b_{K_1}}(P_1) \oplus E_{b_{K_2}}(P_2) = \alpha\}.$$

   Similarly, build

$$S_2 = \{(P_3, C_3, P_4, C_4) : E_{b_{K_3}}(P_3) \oplus E_{b_{K_4}}(P_4) = \alpha\}.$$

   (c) The size of S_1 and S_2 is $y \cdot 2^{r_b-1}$. 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 to Eq. (12) given by `Attack I`, with the same y given by Eq. (10).

- **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

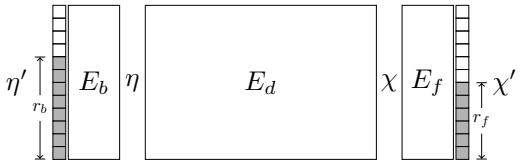

Fig. 3: Differential key-recovery attack on block cipher E.

In the differential cryptanalysis using structures, we collect plaintext-ciphertext pairs by traversing the active bits of plaintext (r_b gray bits in Figure 3) as a structure. Store the structure indexed by the inactive bits of ciphertext ($n - r_f$ white bits in Figure 3) 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 η' and χ', respectively. Using the property of truncated differential of the ciphertext to filter wrong pairs in advance 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 [11] 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 the key candidates for each of the generated quartets, we find 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 satisfy the truncated differential (α', δ') in the plaintext and ciphertext.

For the differential attack, given a pair conforming to (η', χ'), it will only suggest $2^{m_b+m_f-(r_b+r_f)}$ key candidates[7]. However, for the rectangle attack, given

[7] Assume that after partially encrypting the (P_1, P_2) of truncated form η' by E_b, the difference of the corresponding internal states is η with probability of 2^{-r_b}. The same assumption is given for the partial decryption with E_f.

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 of quartets conforming to (α', δ') may never suggest key candidates.

Here is an example of E_b part in Figure 4. 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

$$S(k_{1b} \oplus P_1) \oplus S(k_{2b} \oplus P_2) = \alpha, \tag{6}$$

$$S(k_{3b} \oplus P_3) \oplus S(k_{4b} \oplus P_4) = \alpha. \tag{7}$$

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. (6) on average. Then k_{3b}, k_{4b} are determined. Hence, by Eq. (7), 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 nonlinearly relationships (e.g., Eq. (6) and (7)) will never suggest a key. We call those quartets as *nonlinearly incompatible* quartets. Similar with the *incompatibilities* in building boomerang distinguishers discovered by Murphy [39], *incompatibilities* may also happen in the key-recovery phase of the rectangle attack on ciphers with linear key-schedule.

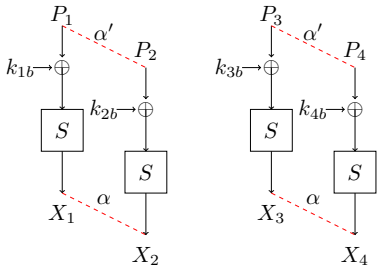

Fig. 4: Nonlinear incompatibility in key-recovery phase.

According to the analysis of the rectangle attack models `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 4, 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. (6) and (7), 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 *nonlinearly incompatible* 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 guess-and-filter or early abort technique [37], which tests the key candidates out 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 *nonlinearly incompatible* quartets, but may also lead to higher overall complexity

To get a better tradeoff, we try to guess all m_b and part of k_f, denoted as k'_f whose size is m'_f. With partial decryption and encryption, we may gain more inactive bits (or bits with fixed differences) from the internal state, which are used to build similar equations as Eq. (6) and (7), to avoid some *nonlinearly incompatible* quartets in advance.

3.1 On the success probability and exhaustive search phase

The success probability given by Selçuk [44] is evaluated by

$$P_s = \Phi(\frac{\sqrt{sS_N} - \Phi^{-1}(1-2^{-h})}{\sqrt{S_N+1}}), \tag{8}$$

where $S_N = \hat{p}^2\hat{q}^2/2^{-n}$ is the signal-to-noise ratio, with an h-bit or higher advantage. This equation is used by previous rectangle attacks [36,53] to estimate the success probability. In the above attack models (`Attack I, II, III`), after generating the k_c-bit key counters, we have to 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` and `Attack 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, this step costs at least $2^{k-(m_b+m_f)}$ with $h = m_b + m_f$.

In `Attack III`, the sizes of key counters are 2^{m_f}, which is smaller than `Attack I` and `Attack II`. Then the complexity of the exhaustive search is $2^{m_b} \times 2^{m_f-h} \times 2^{k-(m_b+m_f)} = 2^{k-h}$ ($h \leq m_f$ due to the size of key counters is 2^{m_f}) for `Attack III`. So the exhaustive search step costs at least 2^{k-m_f}. In certain cases, this cost of exhaustive search step may bound the total time complexity.

3.2 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 attacked rounds. As shown in Figure 5, we guess part of k_f to compute some internal states as filters. We display the new tradeoff of the rectangle attack in Algorithm 1.

Complexity. Choose

$$y = \sqrt{s} \cdot 2^{n/2-r_b}/\hat{p}\hat{q}, \tag{10}$$

we get about

$$(y \cdot 2^{2r_b})^2 \cdot 2^{-2r_b} \cdot 2^{-n}\hat{p}^2\hat{q}^2 = s \tag{11}$$

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

Construct y structures of 2^{r_b} plaintexts each
For structure i ($1 \le i \le y$), 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, which is a part of $(m_b + m'_f)$-bit key* **do**
  Initialize a list of $2^{m_b+m_f-x}$ counters
  for *each of $(m_b + m'_f - x)$-bit key 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**
        The corresponding plaintext $P_2 \in L_2[i]$ under K_2 is computed by
        $$E_b^{-1}(m_b \oplus \Delta K_b, E_b(m_b, P_1) \oplus \alpha) = P_2$$
        $S_1 \leftarrow (P_1, C_1, P_2, C_2)$
      end
      for $(P_3, C_3) \in L_3[i]$ **do**
        Do similar steps as Line 9 and 10 and $S_2 \leftarrow (P_3, C_3, P_4, C_4)$
      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**
      Partially decrypt (C_1, C_2) to get two h_f-bit internal state (X_1, X_2). Assuming they are the first h_f-bit, i.e.:
      $$\begin{cases} X_1[1, \cdots, h_f] = E_f^{-1}(k'_f, C_1) \\ X_2[1, \cdots, h_f] = E_f^{-1}(k'_f \oplus \Delta K_f, C_2) \end{cases} \tag{9}$$
      Assume the inactive bits of δ' are first $n - r_f$ bits. Let $\tau = (X_1[1,\cdots,h_f], X_2[1,\cdots,h_f], C_1[1,\cdots,n-r_f], C_2[1,\cdots,n-r_f])$
      $H[\tau] \leftarrow (P_1, C_1, P_2, C_2)$
    end
    for $(P_3, C_3, P_4, C_4) \in S_2$ **do**
      Partially decrypt (C_3, C_4) to get $X_3[1,\cdots,h_f]$ and $X_4[1,\cdots,h_f]$ as Eq. 9 but under different subkeys of K_3 and K_4.
      Let $\tau' = (X_3[1,\cdots,h_f], X_4[1,\cdots,h_f], C_3[1,\cdots,n-r_f], C_4[1,\cdots,n-r_f])$
      Access $H[\tau']$ to find (P_1, C_1, P_2, C_2) to generate quartet (C_1, C_2, C_3, C_4).
      Determine the key candidates involved in E_f and increase corresponding counters `/* Denote the time as` ε. */
    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 a h-bit or higher advantage. Guess the remaining $k - (m_b + m_f)$ bit keys and combined with the guessed x subkey bits to check the full key.
end

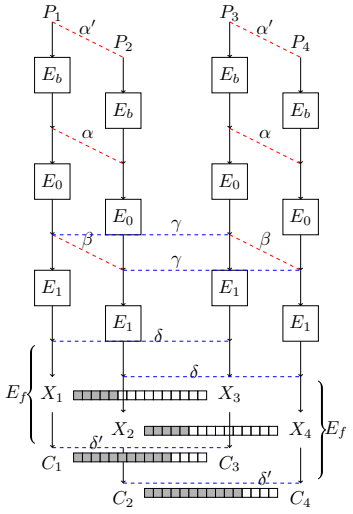

Fig. 5: Filter with internal state.

quartets for the right $(m_b + m_f)$-bit. Therefore, the total data complexity for the 4 oracles with K_1, K_2, K_3 and K_4 is

$$4y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+2}/\hat{p}\hat{q}. \tag{12}$$

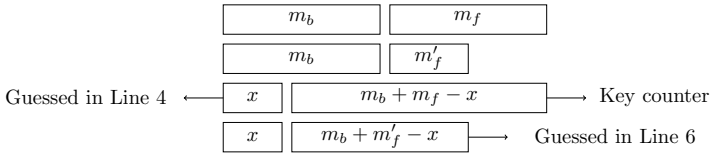

Fig. 6: Diagram for the guessed key in Algorithm 1.

▶ **Time I** (T_1): In Line 7 to 21 of Algorithm 1, the time complexity is about

$$T_1 = 2^{x+m_b+m'_f-x} \cdot y \cdot 2^{r_b} \cdot 2 = \sqrt{s} \cdot 2^{m_b+m'_f+n/2+1}/\hat{p}\hat{q} \tag{13}$$

▶ **Time II** (T_2): In Line 24, we generate about

$$\begin{aligned} &2^{x+m_b+m'_f-x} \cdot y^2 \cdot 2^{2r_b-2(n-r_f)-2h_f} \\ &= s \cdot 2^{m_b+m'_f-2(n-r_f)-2h_f+n}/\hat{p}^2\hat{q}^2 \\ &= s \cdot 2^{m_b+m'_f-n+2r_f-2h_f}/\hat{p}^2\hat{q}^2 \end{aligned} \tag{14}$$

quartets. The time complexity of Line 25 to generate the key counters is

$$T_2 = (s \cdot 2^{m_b+m'_f-n+2r_f-2h_f}/\hat{p}^2\hat{q}^2) \cdot \varepsilon. \tag{15}$$

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

$$T_3 = 2^x \cdot 2^{m_b+m_f-x-h} \cdot 2^{k-(m_b+m_f)} = 2^{k-h}. \tag{16}$$

For choosing h (according to the success probability Eq. (8)), 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 store the structures is

$$2^{m_b+m_f-x} + 4y \cdot 2^{r_b} = 2^{m_b+m_f-x} + \sqrt{s} \cdot 2^{n/2+2}/\hat{p}\hat{q}. \tag{17}$$

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 $\hat{p}\hat{q}$. Hence, taken `SKINNY` as an example, we build an automatic model to determine boomerang distinguishers with optimal key-recovery parameters in the following section.

4 Automatic Model For `SKINNY`

4.1 Description of `SKINNY`

At CRYPTO 2016, Beierle *et al.* proposed the lightweight block cipher `SKINNY` [7]. Denote n as the block size, $\tilde{n}$ as the tweakey size and c as the cell 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. The tweakey state is viewed as z 4×4 square arrays of cells, denoted as $(TK1)$ when $z = 1$, $(TK1, TK2)$ when $z = 2$, and $(TK1, TK2, TK3)$ when $z = 3$. In each round, the state is updated with 5 operations: `SubCells (SC)`, `AddConstants (AC)`, `AddRoundTweakey (ART)`, `ShiftRows (SR)` and `MixColumns (MC)`. `SKINNY` adopts the TWEAKEY framework [31] as shown in Figure 7. We briefly introduce the round function as follows. For more details, please refer to [7].

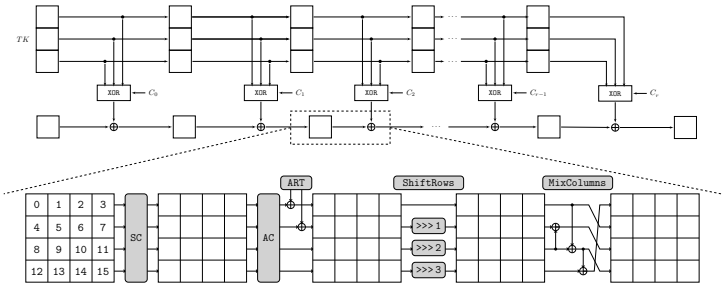

Fig. 7: The framework of `SKINNY`-n-$3n$. (Thanks to https://www.iacr.org/authors/tikz/)

The MC operation adopts non-MDS binary matrix, and we have

$$\texttt{MC}\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \oplus c \oplus d \\ a \\ b \oplus c \\ a \oplus c \end{pmatrix} \quad \text{and} \quad \texttt{MC}^{-1}\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \beta \\ \beta \oplus \gamma \oplus \delta \\ \beta \oplus \delta \\ \alpha \oplus \delta \end{pmatrix}. \tag{18}$$

The equivalent subtweakey in round i is $ETK_i = \texttt{MC} \circ \texttt{SR}(STK_i)$ as Figure 8.

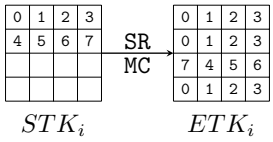

Fig. 8: The relations between the cells of STK_i and ETK_i.

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.*

4.2 Previous automatic models of searching boomerang distinguishers on SKINNY

On SKINNY, there are several automatic models aim at searching boomerang distinguishers. The designers of SKINNY [7] first gave the Mixed-Integer Linear Programming (MILP) model to search truncated differentials of SKINNY. Later, Liu *et al.* [36] tweaked the model to search boomerang distinguishers.

As Dunkelman *et al.*'s (related-key) sandwich attack framework [27], 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 $\tilde{p}^2\tilde{q}^2t$.

Song *et al.* [46] proposed a generalized framework of BCT revisited the Boomerang Connectivity Table (BCT) proposed by Cid *et al.* in [21], and to systematically calculate the probability of a boomerang distinguisher considering the dependence.

Hadipour *et al.* [30] introduced a heuristic approach to search a boomerang distinguisher with a set of new tables. They first searched truncated differential with the minimum number of active S-boxes with an MILP model based on Cid *et al.*'s [22] model. At the same time, the switching effects in multiple rounds were also considered. Then they used the MILP/SAT models to get actual differential characteristic and experimentally evaluate the probability of the middle part. Based on the algorithm proposed in [46], they evaluated the probability of the middle part mathematically.

Almost at the same time, Delaune, Derbez and Vavrille [23] proposed a new automatic tool to search boomerang distinguishers and provide 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 use an MILP model to search for the upper and lower trails throughout all rounds by automatically handling the middle rounds. Then a CP model is applied to search for the best possible instantiations.

Recently, Qin *et al.* [40] combine the key-recovery attack phase and distinguisher searching phase into one uniform automatic model to attack more rounds. Their extended model tweaks the previous models of Hadipour *et al.* [30] and Delaune *et al.* [23] for searching the entire $(N_b + N_d + N_f)$ rounds of a boomerang attack. The aim is to find new boomerang distinguishers in related-tweakey setting that give a key-recovery attack penetrating more rounds.

4.3 Our model to determine a distinguisher

Following the previous automatic models [40,23,30], we introduce a uniform automatic model to search good distinguishers for the new rectangle attack framework in Algorithm 1. We searches 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. The source code of our automatic model is given in https://github.com/key-guess-rectangle/key-guess-rectangle, which is mainly built on the open source by Delaune, Derbez and Vavrille [23].

Different from Qin *et al.*'s [40] uniform automatic key-recovery model, which targets on the rectangle attack framework by Zhao *et al.* [53], our automatic models for Algorithm 1 need 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 [40] model, only the time complexity of (Time II of Zhao *et al.*'s model [53] in Section 2.3) generated 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.

In our extended model searching the entire $(N_b + N_d + N_f)$ rounds of a boomerang attack, we use similar notations as [40,23], 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. $\texttt{DXU}[r][i]$ $(0 \leq r \leq N_b + r_0 + r_m, 0 \leq i \leq 15)$ and $\texttt{DXL}[r][i]$ $(0 \leq r \leq r_m + r_1 + N_f, 0 \leq i \leq 15)$ are on behalf of active cells in the internal states, and `KnownEnc` $(0 \leq r \leq N_b - 1, 0 \leq i \leq 15)$ is on behalf of the m_b-bit subtweakeys involved in the N_b extended rounds, i.e., $\sum_{0 \leq r \leq N_b-2,\ 0 \leq i \leq 7} \texttt{KnownEnc}[r][i]$ corresponds to the total amount of guessed m_b-bit key in E_b. The constraints in E_b are the same to Qin *et al.*'s [40] 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 9), 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 $(0 \leq r \leq N_f - 1, 0 \leq i \leq 15)$, 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 :

$$\texttt{DXFixed}[0][i] = 1, \forall\ 0 \leq i \leq 15.$$

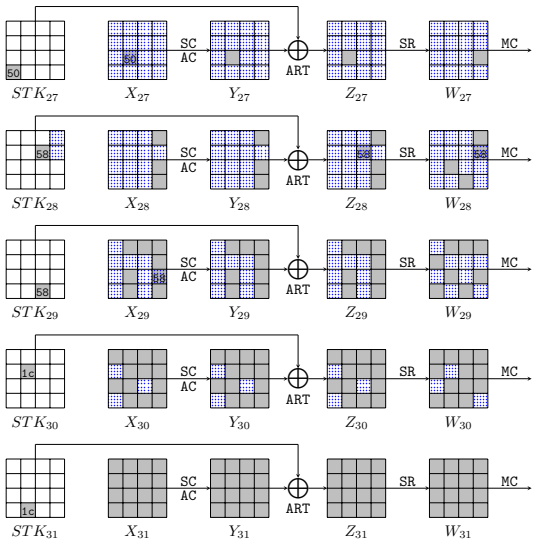

Fig. 9: The cells with fixed differences in N_f-round of the attack on SKINNY-128-384.

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,

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

The constraints on the impact of the MC operation by Equation (18) to 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 $\texttt{DXFilter}[r][i]$ for i-th cell of X_r and a binary variable $\texttt{DWFilter}[r][i]$ for i-th cell of W_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$, where $\texttt{DXFilter}[r][i] = 1$ and $\texttt{DWFilter}[r][i] = 1$ indicate the corresponding cells can be used as filters. Note that, the $(n - r_f)$ inactive bits of the ciphertext are also indicated by $\texttt{DWFilter}$.

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 $\texttt{DXFixed}$, $\texttt{DXL}$ and $\texttt{DXFilter}$ are given in Table 2.

Table 2: All valid valuations of $\texttt{DXFixed}$, $\texttt{DXL}$ and $\texttt{DXFilter}$ for $\texttt{SKINNY}$.

$\texttt{DXFixed}[r][i]$	$\texttt{DXL}[r_m + r_1 + r][i]$	$\texttt{DXFilter}[r][i]$
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, where

$$\texttt{DWFilter}[N_f - 1][i] = \texttt{DWFixed}[N_f - 1][i], \forall\ 0 \le i \le 15.$$

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 $\texttt{MC}$ 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:

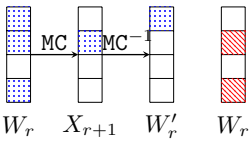

Fig. 10: Example (1).

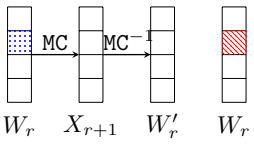

Fig. 11: Example (2).

1. Example (1): Figure 10 shows the propagation of fixed differences, i.e., $\texttt{DWFixed}$ and $\texttt{DXFixed}$, where □ cells denote the unfixed differences. In Figure 10, the differences of $W_r[0, 1, 3]$ are fixed (marked by ⊡). After the $\texttt{MC}$ 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 $\mathtt{MC}^{-1}$ 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 11, 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 $\mathtt{MC}^{-1}$ 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 ▨).

We give some other examples in Figure 12. All valid valuations of DWFixed and DWFilter are given in Supplementary Material A. Note that DXFixed is only used as the intermediate variable to determine DWFilter, since DXFixed is fully determined by DWFixed.

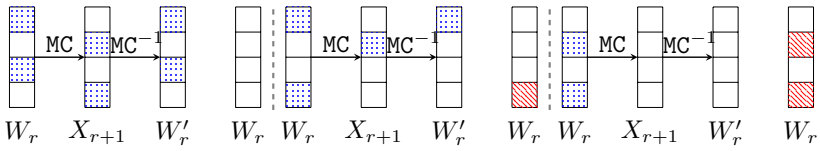

Fig. 12: Some cases of the valuations of DWFixed and DWFilter.

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

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

We define a binary variable $\mathtt{DXisFiter}[r][i]$ for i-th cell of X_r and a binary variable $\mathtt{DWisFiter}[r][i]$ for i-th cell of W_r $(0 \le r \le N_f-1, 0 \le i \le 15)$, where $\mathtt{DXisFiter}[r][i] = 1$ and $\mathtt{DWisFilter}[r][i] = 1$ indicate the corresponding cells are chosen to be filters before generating the quartets. $\forall\ 0 \le r \le N_f-1, 0 \le i \le 15$,

$$\begin{cases} \mathtt{DXisFilter}[r][i] \le \mathtt{DXFilter}[r][i], \\ \mathtt{DWisFilter}[r][i] \le \mathtt{DWFilter}[r][i]. \end{cases}$$

Modeling the guessed subtweakey cells in E_f for generating the quartets. We define a binary variable $\mathtt{DXGuess}[r][i]$ for i-th cell of X_r and a binary

variable $\texttt{DWGuess}[r][i]$ for i-th cell of W_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$, where $\texttt{DXGuess}[r][i] = 1$ and $\texttt{DWGuess}[r][i] = 1$ indicate 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 $0 \le r \le N_f - 1$ and $0 \le i \le 7$.

For the round 0, only cells used to be filters in the internal state need to be known:

$$\texttt{DXGuess}[0][i] = \texttt{DXisFilter}[0][i],\ \forall\ 0 \le i \le 15.$$

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:

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

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}: $\forall\ 0 \le r \le N_b - 2,\ 0 \le i \le 3$

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

We have $\sum_{0 \le r \le N_f - 1,\ 0 \le i \le 7} \texttt{DXGuess}[r][i]$ 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 3.2, the advantage h determines the exhaustive search time, where h should be smaller than the number of key counters, i.e. $h \le m_b + m_f - x$. The x-bit guessed subkey should satisfy $x \le m_b + m'_f$, 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 $(0 \le r \le N_f - 1, 0 \le i \le 15)$, where $\texttt{KnownDec}[r][i] = 1$ indicates 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:

$$\texttt{KnownDec}[0][i] = \texttt{DXL}[r_m + r_1][i], \forall\ 0 \le i \le 15.$$

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}: $\forall\ 0 \le r \le N_b - 2,\ 0 \le i \le 3$

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

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. 3.2, 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 to [23], where $\texttt{DXU}$, $\texttt{DXL}$ and $\texttt{DXU} \wedge \texttt{DXL}$ are on behalf of the $\tilde{p}$, $\tilde{q}$ and t. $\texttt{KnownEnc}$ is on behalf of the m_b, and we don't 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 (14), 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:

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

So we get a uniformed objective:

$$\begin{cases} \text{Minimize} \quad obj \\ \qquad obj \geq T_1, \\ \qquad obj \geq T_2, \\ \qquad obj \geq T_3. \end{cases}$$

4.4 New distinguishers for SKINNY

With our new model, we add such conditions to the automatic searching model in [23] to search for new distinguishers. Due to different parameters have different coefficients in the formula of the time complexity, we give them different weights to model the objective more accurate. 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 [30], 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 to run our model. 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 new truncated upper and lower differentials using the MILP model and get instantiations using the CP model following the open source [23], we obtain new distinguishers for SKINNY-128-384, SKINNY-64-192 and SKINNY-128-256. 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 [46,30,23] 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 [23], which is provided in https://github.com/key-guess-rectangle/key-guess-rectangle. Our source code also automatically draws the rectangle attack pictures including the distinguisher and key-recovery part. In addition, we summarize the previous boomerang distinguishers for a few versions of SKINNY in Table 4.

We list the 23-round boomerang distinguisher for SKINNY-128-384 in Table 5. For more details, we refer to Table 17, 18 and 19 and Table 20, 21, 22 and 23 in Supplementary Material J.

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

Table 4: 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.
64-192	22	$2^{-42.98}$	-	[46]
	22	$2^{-54.94}$	26	[36]
	23	$2^{-55.85}$	29	[30]
	23	$2^{-57.93}$	-	[23]
	22	$2^{-57.73}$	30	[40]
	22	$2^{-57.56}$	31	Ours
128-256	18	$2^{-77.83}$	-	[46]
	18	$2^{-103.84}$	22	[36]
	20	$2^{-85.77}$	-	[23]
	21	$2^{-116.43}$	24	[30]
	19	$2^{-116.97}$	25	[40]
	18	$2^{-108.51}$	25	Ours
	19	$2^{-121.07}$	26	Ours
128-384	22	$2^{-48.30}$	-	[46]
	23	2^{-112}	27	[36]
	23	2^{-112}	28	[53]
	24	$2^{-86.09}$	-	[23]
	25	$2^{-116.59}$	30	[30]
	22	$2^{-101.49}$	30	[40]
	23	$2^{-115.09}$	32	Ours

5 Improved Attacks on SKINNY

In this section, we give an improved attack on round-reduced SKINNY-128-384 using the distinguisher in Section 4.4 with our new rectangle attack framework. For SKINNY-64-192 and SKINNY-128-256, we also use the new distinguishers to achieve one more round than previous. For SKINNY-64-128, we find the distinguisher in [40] is optimal and attack one more round with the new attack framework. The details can refer to Supplementary Material C.

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

We use the 23-round rectangle distinguisher for SKINNY-128-384 given in Table 5, whose probability is $2^{-n}\tilde{p}^2t\tilde{q}^2 = 2^{-128-115.09} = 2^{-243.09}$. Appending 4-round

Table 5: The 23-round related-tweakey boomerang distinguisher for 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

E_b and 5-round E_f, we attack 32-round SKINNY-128-384 as illustrated in Figure 13. 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 $0 \le i \le 3$ according to Figure 8. So we have $r_b = 12 \cdot 8 = 96$ by W_0'. As shown in Figure 13, 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 $k_f' = \{STK_{30}[1,3,5,7], STK_{24}[0,2,3,4,5,6,7]\}$ marked in red indexes and $h_f = \{X_{29}[11], W_{29}[5,7,13,15], W_{30}[5,8,15]\}$ marked in bold. With the above preparation, we give the attack according to Algorithm 1 as follows:

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	$STK_{28}[7]$,$STK_{29}[2,4]$, $STK_{30}[1,3,5,6]$, $STK_{31}[0,1,2,5,6,7]$

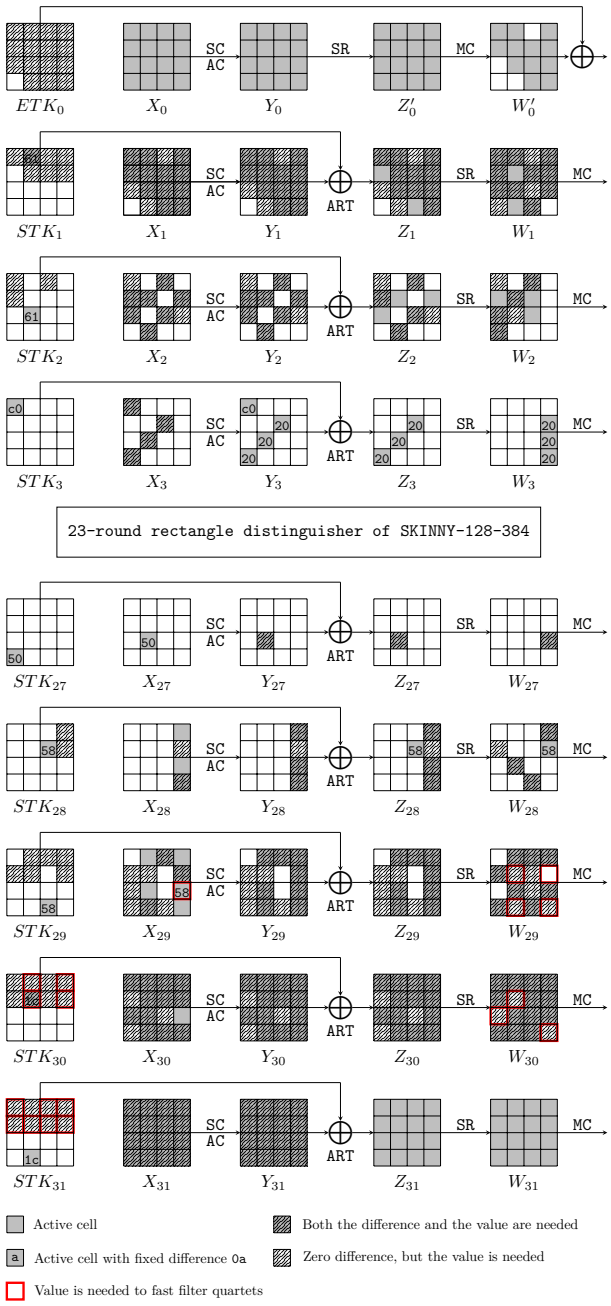

Fig. 13: The 32-round attack against SKINNY-128-384.

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. (10). For each structure, query the 2^{96} ciphertexts by encryptions under K_1, K_2, K_3 and K_4. Hence, the data complexity 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. (12). 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 the other $(m_b + m'_f - x) = (232 - x)$-bit key involved in E_b and E_f:
      i. For 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 Supplementary Material B. 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 $h_f = 56$-bit $\{W_{30}[5, 8, 15], W_{29}[5, 7, 13, 15], X_{29}[11]\}$ of C_1 and $h_f = 56$-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. (13), 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. (15).
      iii. On ε: for each of $s \cdot 2^{347.09}$ quartets, determine the key candidates and increase the corresponding counters. According to Eq. (15), 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 more clear.
         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. (18). 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 $\Delta X_{29}[1] = \Delta X_{29}[9] = \Delta X_{29}[13]$ from Eq. (18). Hence, we obtain $\Delta X_{29}[1]$

and deduce $STK_{29}[1]$ by Lemma 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 $\Delta X_{29}[3] = \Delta X_{29}[7] = \Delta X_{29}[15]$ from Eq. (18). Hence, we obtain $\Delta X_{29}[3, 7]$ and deduce $STK_{29}[3, 7]$ by Lemma 1. 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 $\Delta X_{28}[3] = \Delta X_{28}[11] = \Delta X_{28}[15]$ from Eq. (18). Since $STK_{28}[6]$ can be deduced from the known $ETK_0[0]$, $STK_2[2]$ and $STK_{30}[1]$, 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. (16), $T_3 = 2^{k-h}$.

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	$X_{27}[9]$	$STK_{28}[7]$,$STK_{29}[2, 4]$, $STK_{30}[1, 3, 5, 6]$, $STK_{31}[0, 1, 2, 5, 6, 7]$

In order to balance T_1, T_2, T_3 and memory complexity and achieve a high success probability, we set $s = 1$, $h = 40$ and $x = 128$ ($x \leq m_b + m'_f = 232$, $h \leq m_b + m_f - 2c - x = 320 - x$) with Eq. (8). 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%.

6 Conclusion and Further Disscussion

In this paper, 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. As proof of work, we give a uniform automatic model on `SKINNY` to search 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 rounds than the best previous attacks.

Further discussion.

For `ForkSkinny`, we find that the 21-round distinguisher on `ForkSkinny`-128-256 in [40] is also optimal for our new rectangle attack model. Our attack on 28-round `ForkSkinny-128-256` with 256-bit key reduce the time complexity of [40] by a factor of 2^{22}. For `Deoxys-BC` and `GIFT`, we do not give the automatic models but only apply our rectangle attack framework in Algorithm 1 with the previous distinguishers. Our attack for `Deoxys-BC-384` reduces the time complexity of the best previous 14-round attack [52] by a factor of $2^{22.7}$ with similar data and memory complexities. For `GIFT-64`, our rectangle attack use the same rectangle distinguisher with Ji *et al.* [33], but achieve one more round. Moreover, compared with the best previous attack achieved by differential attack by Sun *et al.* [47], our rectangle attack achieves the same 26 rounds, where the time and memory complexities are reduced by factors of $2^{10.45}$ and 2^{39}, respectively, and the data complexity are increased by a factor of $2^{2.82}$. The details of the attacks can refer to Supplementary Material E, F and G.

For single-key setting, our tradeoff key-recovery model in Section 3 and Zhao *et al.*'s model [51] can be trivially converted into the single-key model by just let 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 [12] and achieving better time complexity (see Supplementary Material H).

To better understand different key-recovery rectangle models, we give an overall analysis of the four attack models in Section 2 and 3. With different parameters, different models perform differently (see Supplementary Material I).

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, Proceedings*, 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. Tim Beyne. Block cipher invariants as eigenvectors of correlation matrices. *J. Cryptol.*, 33(3):1156–1183, 2020.
9. Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In *FSE '98, Proceedings*, pages 222–238.
10. Eli Biham, Orr Dunkelman, and Nathan Keller. New cryptanalytic results on IDEA. In *ASIACRYPT 2006, Proceedings*, pages 412–427.
11. Eli Biham, Orr Dunkelman, and Nathan Keller. New results on boomerang and rectangle attacks. In *FSE 2002, Revised Papers*, volume 2365, pages 1–16.
12. Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack - rectangling the serpent. In *EUROCRYPT 2001, Proceeding*, volume 2045, pages 340–357.
13. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In *EUROCRYPT 2005, Proceedings*, volume 3494, pages 507–525.
14. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology*, 4(1):3–72, 1991.
15. Alex Biryukov, Christophe De Cannière, and Gustaf Dellkrantz. Cryptanalysis of SAFER++. In *CRYPTO 2003, Proceedings*, volume 2729, pages 195–211.
16. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT 2009, Proceedings*, volume 5912, pages 1–18.
17. Christina Boura and Anne Canteaut. On the boomerang uniformity of cryptographic sboxes. *IACR Trans. Symmetric Cryptol.*, 2018(3):290–310, 2018.
18. Christina Boura, Virginie Lallemand, María Naya-Plasencia, and Valentin Suder. Making the impossible possible. *J. Cryptol.*, 31(1):101–133, 2018.
19. Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In *ASIACRYPT 2014, Proceedings, Part I*, volume 8873, pages 179–199.
20. 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.

21. 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.
22. 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.
23. Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs application to SKINNY. *IACR Trans. Symmetric Cryptol.*, 2020(4):104–129, 2020.
24. Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In *CRYPTO 2016, Proceedings, Part II*, pages 157–184.
25. 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.
26. Orr Dunkelman, Nathan Keller, Eyal Ronen, and Adi Shamir. The retracing boomerang attack. In *EUROCRYPT 2020, Proceedings, Part I*, volume 12105, pages 280–309.
27. 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.
28. 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.
29. 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.
30. Hosein Hadipour, Nasour Bagheri, and Ling Song. Improved rectangle attacks on SKINNY and CRAFT. *IACR Cryptol. ePrint Arch.*, 2020:1317, 2020.
31. 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.
32. 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.
33. Fulei Ji, Wentao Zhang, Chunning Zhou, and Tianyou Ding. Improved (related-key) differential cryptanalysis on GIFT. *Accepted to SAC 2020*.
34. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round MARS and serpent. In *FSE 2000, Proceedings*, volume 1978, pages 75–93.
35. Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In *CRYPTO 2015, Proceedings, Part I*, volume 9215, pages 161–185.
36. 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.
37. 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.

38. 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.
39. Sean Murphy. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory*, 57(4):2517–2521, 2011.
40. 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. Cryptology ePrint Archive, Report 2021/656, 2021. https://eprint.iacr.org/2021/656.
41. Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. *IACR Transactions on Symmetric Cryptology*, 2018(3):124–162, 2018.
42. Yu Sasaki. Improved related-tweakey boomerang attacks on Deoxys-BC. In *AFRICACRYPT 2018, Proceedings*, pages 87–106.
43. 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.
44. Ali Aydin Selçuk. On probability of success in linear and differential cryptanalysis. *J. Cryptology*, 21(1):131–147, 2008.
45. 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.
46. 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.
47. 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.
48. 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.
49. David A. Wagner. The boomerang attack. In *FSE '99, Proceedings*, volume 1636, pages 156–170.
50. 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.
51. 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.
52. 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.
53. 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.

Supplementary Material

A All valid valuations of DWFixed and DWFilter for SKINNY

This section give all valid valuations of DWFixed and DWFilter for SKINNY when model which cells that could be used to filter quartets in E_f.

Table 8: All valid valuations of DWFixed and DWFilter for SKINNY.

DWFixed$[r]$				DWFilter$[r]$			
$[i]$	$[i+4]$	$[i+8]$	$[i+12]$	$[i]$	$[i+4]$	$[i+8]$	$[i+12]$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	0	1
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

B The procedure of generating plaintext pairs for SKINNY-128-384

In this section, we explain how to deduce P_2 from P_1 with m_b-bit subkeys involved in E_b. We take the 4-round E_b of 32-round attack on SKINNY-128-384 as example (see Fig. 13), and decompose the whole process into two phases:

1. Partially encrypt P_1 to Y_3 (only active cells) as shown in Fig. 14.
2. Partially decrypt $\bar{Y_3} = Y_3 \oplus \Delta Y_3$ to get P_2 as shown in Fig. 15.

In Phase (1): As shown in Fig. 14, in order to compute $Y_3[0, 6, 9, 12]$ from P_1, we need the values of the cells marked by ⊡ in X_3. With the details of MC and SR, the values of $Z_2[0, 2, 4, 10, 11, 13]$ and corresponding $STK_2[0, 2, 4]$ are needed. Then cell positions in X_2 which need to be known are the same as those in Z_2. Similarly, the values of $Z_1[0-2, 5, 6, 8-11, 13, 15]$ and corresponding $STK_1[0-2, 5, 6]$ are needed. Then $W_0'[0-2, 5, 6, 8-11, 13, 15]$ and $ETK_0[0-$

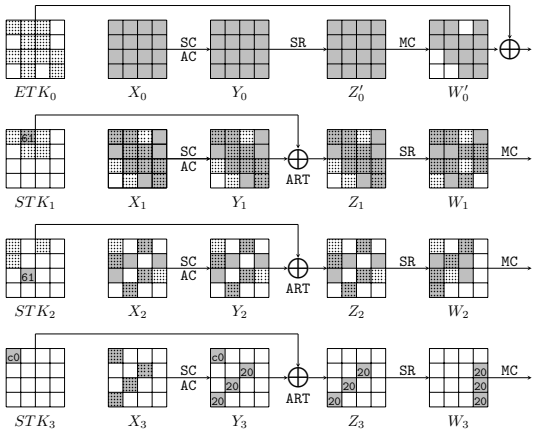

Fig. 14: Phase (1): The 4-round E_b of the attack on SKINNY-128-384.

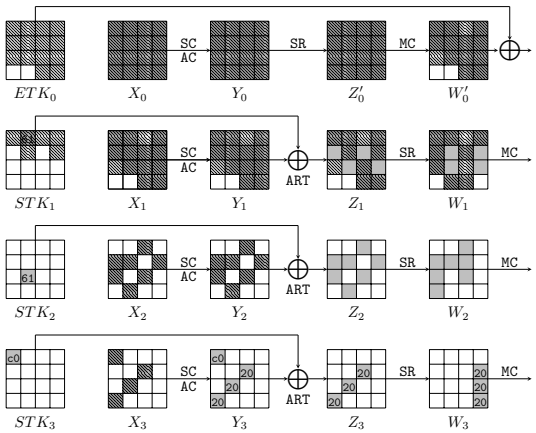

Fig. 15: Phase (2): The 4-round E_b of the attack on SKINNY-128-384.

$2, 5, 6, 8-11, 13, 15]$ are needed to compute $X_1[0-2, 5, 6, 8-11, 13, 15]$. All cells need to be known in Phase (1) are marked by ⊡ and ■.

In Phase (2): As shown in Fig. 15, with $Y_3[0, 6, 9, 12]$ computed in Phase (1), we can compute $\bar{Y_3}[0, 6, 9, 12] = Y_3[0, 6, 9, 12] \oplus \alpha[0, 6, 9, 12]$. So the differences of the active cells in $\Delta X_3[0, 6, 9, 12]$ are determined. Therefore, we compute the differences of active cells in $\Delta Y_2[2, 4, 5, 7, 8, 10, 13]$ from ΔX_3 through the linear operations SR and MC. Then we need compute the values of $Y_2[2, 4, 5, 7, 8, 10, 13]$ from P_1 as Phase (1), where the values of $Z_1[0-3, 5, 7, 8, 10, 14, 15]$ and $STK_1[0-3, 5, 7]$ are needed. Suppose we have known the values of $Y_2[2, 4, 5, 7, 8, 10, 13]$, we can compute the values of $\bar{Y_2}[2, 4, 5, 7, 8, 10, 13]$ and deduce the $\Delta Y_1[0, 1, 3-7, 9-11, 14, 15]$. So in total, the values of $Y_1[0-11, 14, 15]$ are needed. We need $ETK_0[0-11, 14, 15]$ and $W_0'[0-11, 14, 15]$ to deduce $Y_1[0-11, 14, 15]$ and $\Delta X_1[0, 1, 3-7, 9-11, 14, 15]$, as well as $\Delta W_0'[0, 1, 3-7, 9-11, 14, 15]$. So for P_2, we obtain $\bar{W_0'}[0, 1, 3-7, 9-11, 14, 15] = W_0'[0, 1, 3-7, 9-11, 14, 15] \oplus \Delta W_0'[0, 1, 3-7, 9-11, 14, 15]$. All cells need to be known in Phase (1) are marked by ▧ and ▩.

Totally, $m_b = 18 \times 8 = 144$ key bits are involved in the above two phases, and involved in E_b.

C Improved Attacks on other versions of SKINNY

C.1 Improved attack on 31-round SKINNY-64-192

We use the 22-round rectangle distinguisher for SKINNY-64-192 given in Sect. 4.3, whose probability is $2^{-n}\tilde{p}^2t\tilde{q}^2 = 2^{-64-57.56} = 2^{-121.56}$. Appending 4-round E_b and 5-round E_f, we attack 31-round SKINNY-64-192, as illustrated in Fig. 16. There are $r_b = 15 \cdot 4 = 60$ by W_0', $m_b = 19 \cdot 4 = 76$, $r_f = 16 \cdot 4 = 64$, $m_f = 24 \cdot 4 = 96$. $h_f = 8 \cdot 4 = 32$ and $m_f' = 11 \cdot 4 = 44$. Due to the tweakey schedule, we can deduce $STK_{28}[1, 4, 5, 7]$ from $ETK_0[2, 10, 8, 0]$, $STK_2[4, 3, 1, 2]$ and $STK_{30}[0, 6, 3, 1]$. So there are only $(m_f - 4c) = 80$-bit subtweakey unknown in E_f. As shown in Fig. 16, there are $k_f' = \{STK_{29}[1, 3, 5, 7], STK_{24}[0, 2, 3, 4, 5, 6, 7]\}$ and $h_f = \{X_{28}[11], W_{28}[5, 7, 13, 15], W_{29}[5, 8, 15]\}$.

The attack follows Algorithm 1 in Sect. 3.2 and is similar to Sect. 5.1, where $y = \sqrt{s} \cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2t\tilde{q}^2} = \sqrt{s} \cdot 2^{0.78}$. Hence, the data complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2t\tilde{q}^2} = \sqrt{s} \cdot 2^{62.78}$.

According to Eq. (13), $T_1 = \sqrt{s} \cdot 2^{m_b+m_f'+n/2+1}/\sqrt{\tilde{p}^2t\tilde{q}^2} = \sqrt{s} \cdot 2^{76+44+32+1+28.78} = \sqrt{s} \cdot 2^{181.78}$.

According to Eq. (15), $T_2 = s \cdot 2^{m_b+m_f'-2h_f+n}/(\tilde{p}^2t\tilde{q}^2) \cdot \varepsilon = s \cdot 2^{76+44-64+64+57.56}$. $\varepsilon = s \cdot 2^{177.56} \cdot \varepsilon$.

According to Eq. (16), $T_3 = 2^{k-h}$, where $h \leq m_b + m_f - 4c - x = 156 - x$.

According to Eq. (17), the memory complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2t\tilde{q}^2} + 2^{m_b+m_f-4c-x} = \sqrt{s} \cdot 2^{62.78} + 2^{156-x}$.

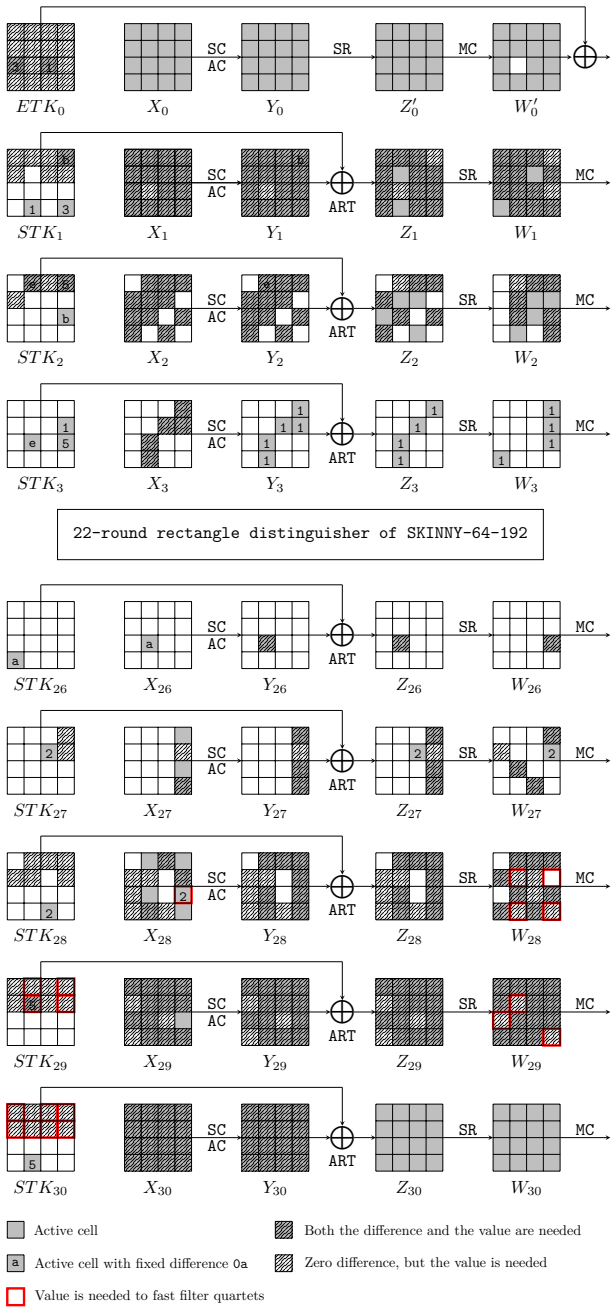

Fig. 16: The 31-round attack against SKINNY-64-192.

The tweakey recovery process with time ε. The tweakey recovery process is same with Sect. 5.1. For each of the $s \cdot 2^{177.56}$ quartets, we determine the key candidates and increase the corresponding counters with the following steps:

1. **In round 30:** guessing $STK_{30}[1]$ and together with k'_f as shown in Table 9, we compute $Z_{29}[6, 14]$ and peel off round 30. Then $\Delta Y_{29}[6]$ and $\Delta X_{29}[14]$ are deduced. For the 3rd column of X_{29} of (C_1, C_3), we can obtain $\Delta X_{29}[6] = \Delta X_{29}[14]$ from Eq.(18). Hence, we obtain $\Delta X_{29}[6]$ and deduce $STK_{29}[6]$ by Lemma 1. Similarly, we deduce $STK'_{29}[6]$ for (C_2, C_4). Since $\Delta STK_{29}[6]$ is fixed, we get a 4-bit filter. $s \cdot 2^{177.56} \cdot 2^4 \cdot 2^{-4} = s \cdot 2^{177.56}$ quartets remain.
2. **In round 29:** guessing $STK_{29}[0]$, we compute $Z_{28}[1, 9, 13]$. Then $\Delta Y_{28}[1]$ and $\Delta X_{28}[9, 13]$ are deduced. For the 2nd column of X_{28} of (C_1, C_3), we can obtain $\Delta X_{28}[1] = \Delta X_{28}[9] = \Delta X_{28}[13]$ from Eq.(18). Since $STK_{28}[1]$ can be deduced from the known $ETK_0[2]$, $STK_2[4]$ and $STK_{30}[0]$, we can compute $X_{28}[1]$ and $\Delta X_{28}[1]$. For both (C_1, C_3) and (C_2, C_4), $\Delta X_{28}[1] = \Delta X_{28}[13]$ and $\Delta X_{29}[9] = \Delta X_{29}[13]$ are two 4-bit filter. $s \cdot 2^{177.56} \cdot 2^4 \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{165.56}$ quartets remain.
3. Guessing $STK_{29}[2, 4]$, we compute $Z_{28}[3, 7, 15]$ and peel off round 29. Then $\Delta Y_{28}[3, 7]$ and $\Delta X_{28}[15]$ are deduced. For the 4th column of X_{28} of (C_1, C_3), we can obtain $\Delta X_{28}[3] = \Delta X_{28}[7] = \Delta X_{28}[15]$ from Eq.(18). Hence, we obtain $\Delta X_{28}[3]$ and deduce $STK_{28}[3]$ by Lemma 1. Similarly, we deduce $STK'_{28}[3]$ for (C_2, C_4), which is a 4-bit filter. Since $STK_{28}[7]$ is deduced from last steps, we can compute $\Delta X_{28}[7]$,where is a 4-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{165.56} \cdot 2^8 \cdot 2^{-4} \cdot 2^{-8} = s \cdot 2^{161.56}$ quartets remain.
4. **In round 28:** guessing $STK_{28}[2]$, we compute $Z_{27}[3, 11, 15]$. Then $\Delta Y_{27}[3]$ and $\Delta X_{27}[11, 15]$ are deduced. For the 4th column of X_{27} of (C_1, C_3), we can obtain $\Delta X_{27}[3] = \Delta X_{27}[11] = \Delta X_{27}[15]$ from Eq.(18). Hence, we obtain $\Delta X_{27}[3]$ and deduce $STK_{27}[3]$ by Lemma 1. Similarly, we deduce $STK'_{27}[3]$ for (C_2, C_4), which is a 4-bit filter. For both (C_1, C_3) and (C_2, C_4), $\Delta X_{27}[11] = \Delta X_{27}[15]$ is a 4-bit filter. $s \cdot 2^{161.56} \cdot 2^4 \cdot 2^{-4} \cdot 2^{-8} = s \cdot 2^{153.56}$ quartets remain.
5. **In round 27:** guessing $STK_{27}[7]$, we compute $X_{27}[9]$. $\Delta X_{27}[9] =$ `0x50` is a 4-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{153.56} \cdot 2^4 \cdot 2^{-8} = s \cdot 2^{149.56}$ quartets remain.

So for each quartet, $\varepsilon = 2^4 \cdot \frac{4}{31} + 2^4 \cdot \frac{4}{31} + 2^{-12} \cdot 2^8 \cdot \frac{4}{31} + 2^{-16} \cdot 2^4 \cdot \frac{4}{31} + 2^{-24} \cdot 2^4 \cdot \frac{4}{31} \approx 2^{2.05}$ and $T_2 = s \cdot 2^{179.61}$.

In order to balance T_1, T_2, T_3 and memory complexity and achieve a high success probability, we set $s = 1$, $h = 24$ and $x = 116$ ($x \leq m_b + m'_f = 120$, $h \leq 156 - x$). We have $T_1 = 2^{181.78}, T_2 = 2^{179.61}$ and $T_3 = 2^{168}$.

In total, for the 31-round attack on `SKINNY-64-192`, the data complexity is $2^{62.78}$, the memory complexity is $2^{62.79}$, and the time complexity is $2^{182.07}$. The success probability is about 66.6%.

C.2 Improved attack on 25-round `SKINNY-64-128`

The 18-round rectangle distinguisher for `SKINNY-64-128` produced by our automatic model is the same to [40], whose probability is $2^{-n}\tilde{p}^2 t\tilde{q}^2 = 2^{-64-55.34} =$

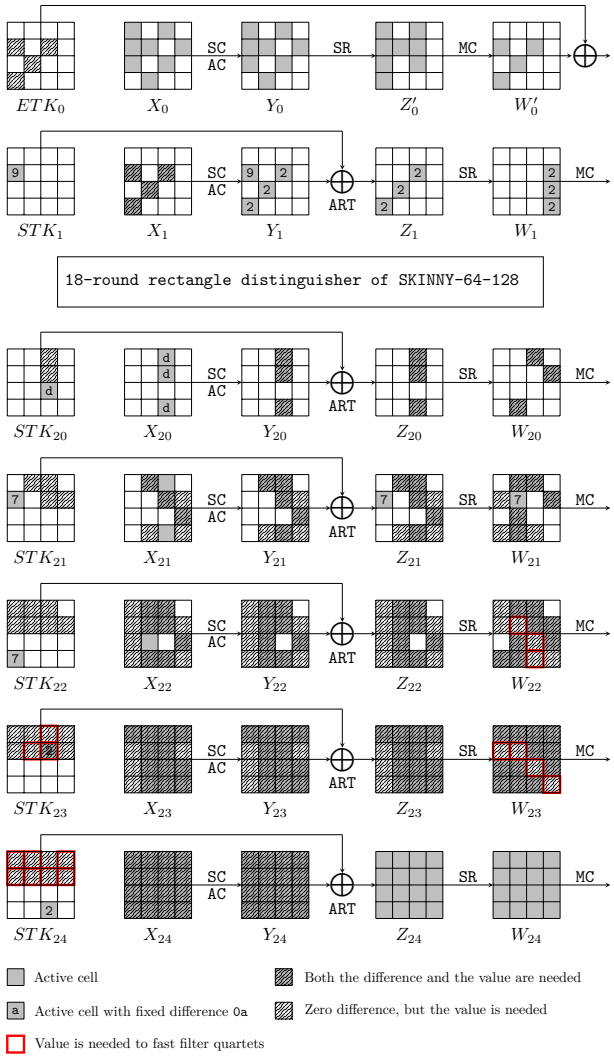

Fig. 17: The 25-round attack against SKINNY-64-128.

Table 9: Tweakey recovery for 31-round SKINNY-64-192, where the red bytes are among k'_f or obtained in the last steps.

Step	Internal state	Involved subtweakeys
1	$Z_{29}[6]$	$STK_{30}[7]$
	$Z_{29}[14]$	$STK_{30}[1]$
2	$Z_{28}[1]$	$STK_{29}[5]$, $STK_{30}[6]$
	$Z_{28}[9]$	$STK_{29}[7]$, $STK_{30}[2,4]$
	$Z_{28}[13]$	$STK_{29}[0]$, $STK_{30}[3,4]$
3	$Z_{28}[3]$	$STK_{29}[7]$, $STK_{30}[4]$
	$Z_{28}[7]$	$STK_{29}[4]$, $STK_{30}[3,5,6]$
	$Z_{28}[15]$	$STK_{29}[2]$, $STK_{30}[1,6]$
4	$Z_{27}[3]$	$STK_{28}[7]$, $STK_{29}[4]$, $STK_{30}[3,5,6]$
	$Z_{27}[11]$	$STK_{28}[5]$, $STK_{29}[0,6]$, $STK_{30}[1,3,4,7]$
	$Z_{27}[15]$	$STK_{28}[2]$, $STK_{29}[1,6]$, $STK_{30}[0,5,7]$
5	$X_{26}[9]$	$STK_{27}[7]$,$STK_{28}[2,4]$, $STK_{29}[1,3,5,6]$, $STK_{30}[0,1,2,5,6,7]$

$2^{-119.34}$. Appending 2-round E_b and 5-round E_f, we attack 25-round SKINNY-64-128, as illustrated in Fig. 17. There are $r_b = 4 \cdot 4 = 16$ by W'_0, $m_b = 3 \cdot 4 = 12$, $r_f = 16 \cdot 4 = 64$, $m_f = 29 \cdot 4 = 116$, $h_f = 7 \cdot 4 = 28$ and $m'_f = 10 \cdot 4 = 40$. Due to the tweakey schedule, we can deduce $STK_{22}[5,6]$ from $ETK_0[6,4]$ and $STK_{24}[3,5]$, and deduce $STK_{20}[6]$ from $ETK_0[6]$ and $STK_{24}[3]$. So there are only $(m_f - 3c) = 104$-bit subtweakey unknown in E_f. As shown in Fig. 17, there are $k'_f = \{STK_{23}[2,5,6], STK_{24}[0,1,3,4,5,6,7]\}$ and $h_f = \{W_{22}[5,10,14], W_{23}[4,5,10,15]\}$.

The attack follows Algorithm 1 in Sect. 3.2, where $y = \sqrt{s} \cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2 t \tilde{q}^2} = \sqrt{s} \cdot 2^{43.67}$. Hence, the data complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2 t \tilde{q}^2} = \sqrt{s} \cdot 2^{61.67}$.

According to Eq. (13), $T_1 = \sqrt{s} \cdot 2^{m_b+m'_f+n/2+1}/\sqrt{\tilde{p}^2 t \tilde{q}^2} = \sqrt{s} \cdot 2^{12+40+32+1+27.67} = \sqrt{s} \cdot 2^{112.67}$.

According to Eq. (15), $T_2 = s \cdot 2^{m_b+m'_f-2h_f+n}/(\tilde{p}^2 t \tilde{q}^2) \cdot \varepsilon = s \cdot 2^{12+40-56+64+55.34}$. $\varepsilon = s \cdot 2^{115.34} \cdot \varepsilon$.

According to Eq. (16), $T_3 = 2^{k-h}$, where $h \leq m_b + m_f - 12 - x = 116 - x$.

According to Eq. (17), the memory complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2 t \tilde{q}^2 2} + 2^{m_b+m_f-12-x} = \sqrt{s} \cdot 2^{61.67} + 2^{116-x}$.

The tweakey recovery process with time ε. For each of the $s \cdot 2^{115.34}$ quartets, we determine the key candidates and increase the corresponding counters with the following steps:

1. **In round 24:** guessing $STK_{24}[2]$ and together with other known subtweakeys as shown in Table 10, we partially decrypt one round to compute $Z_{23}[3,15]$ and peel off round 24. Then $\Delta Y_{23}[3]$ and $\Delta X_{23}[15]$ are deduced. For the 4th column of X_{23} of (C_1, C_3), we can obtain $\Delta X_{23}[3] = \Delta X_{23}[15]$ from Eq.(18). Hence, we obtain $\Delta X_{23}[3]$ and deduce $STK_{23}[3]$ by Lemma 1. Similarly, we deduce $STK'_{23}[3]$ for (C_2, C_4). Since $\Delta STK_{23}[3]$ is fixed, we get a 4-bit filter. $s \cdot 2^{115.34} \cdot 2^4 \cdot 2^{-4} = s \cdot 2^{115.34}$ quartets remain.
2. **In round 23:** guessing $STK_{23}[1]$, we compute $Z_{22}[2,14]$. Then $\Delta Y_{22}[2]$ and $\Delta X_{22}[14]$ are deduced. For the 3rd column of X_{22} of (C_1, C_3), we can obtain

$\Delta X_{22}[2] = \Delta X_{22}[14]$. Hence, we obtain $\Delta X_{22}[2]$ and deduce $STK_{22}[2]$ by Lemma 1. Similarly, we deduce $STK'_{22}[2]$ for (C_2, C_4), which is a 4-bit filter. $s \cdot 2^{115.34} \cdot 2^4 \cdot 2^{-4} = s \cdot 2^{115.34}$ quartets remain.

3. Guessing $STK_{23}[7]$, we compute $Z_{22}[6]$. Since $STK_{22}[6]$ is deduced in data collection process, $X_{22}[6]$ is deduced. Due to Eq.(18), $\Delta X_{22}[6] = \Delta X_{22}[14]$ is a 4-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{115.34} \cdot 2^4 \cdot 2^{-4} \cdot 2^{-4} = s \cdot 2^{111.34}$ quartets remain.
4. Guessing $STK_{23}[0]$, we compute $Z_{22}[5, 9, 13]$. Since $STK_{22}[5]$ is deduced in data collection process, $X_{22}[5]$ is deduced. Due to Eq.(18), $\Delta X_{22}[5] = \Delta X_{22}[9] \oplus \Delta X_{22}[13] \oplus$ `0x7`is a 4-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{111.34} \cdot 2^4 \cdot 2^{-4} \cdot 2^{-4} = s \cdot 2^{107.34}$ quartets remain.
5. **In round 22:** guessing $STK_{22}[1]$, we compute $Z_{21}[2, 14]$. Then $\Delta Y_{21}[2]$ and $\Delta X_{21}[14]$ are deduced. For the 3rd column of X_{21} of (C_1, C_3), we can obtain $\Delta X_{21}[2] = \Delta X_{21}[14]$. Hence, we obtain $\Delta X_{21}[2]$ and deduce $STK_{21}[2]$. Similarly, we deduce $STK'_{21}[2]$ for (C_2, C_4), which is a 4-bit filter. $s \cdot 2^{107.34} \cdot 2^4 \cdot 2^{-4} = s \cdot 2^{107.34}$ quartets remain.
6. Guessing $STK_{22}[7]$ and $STK_{23}[4]$, we compute $Z_{21}[6]$. Then $\Delta Y_{21}[6]$ is deduced. For the 3rd column of X_{21} of (C_1, C_3), we can obtain $\Delta X_{21}[6] = \Delta X_{21}[14]$. Hence, we obtain $\Delta X_{21}[6]$ and deduce $STK_{21}[6]$. Similarly, we deduce $STK'_{21}[6]$ for (C_2, C_4), which is a 4-bit filter. Thereafter, **in round 21**, we compute $Z_{20}[2]$ from $X_{21}[6]$. Then, $\Delta Y_{20}[2]$ is deduced. Due to $\Delta X_{20}[2] =$ `0xd`, we deduce $STK_{20}[2]$ for (C_1, C_3). Similarly, we deduce $STK'_{20}[2]$ for (C_2, C_4), which is a 4-bit filter. $s \cdot 2^{107.34} \cdot 2^8 \cdot 2^{-4} \cdot 2^{-4} = s \cdot 2^{107.34}$ quartets remain.
7. Guessing $STK_{22}[0]$ and $STK_{21}[1]$, we compute $Z_{20}[14]$ and deduce $X_{20}[14]$. $\Delta X_{20}[14] =$ `0xd` is a 4-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{107.34} \cdot 2^8 \cdot 2^{-4} \cdot 2^{-4} = s \cdot 2^{107.34}$ quartets remain.
8. Guessing $STK_{22}[4]$ and $STK_{21}[7]$, we compute $Z_{20}[4]$. Since $STK_{20}[6]$ is deduced in data collection process, we deduce $X_{20}[6]$. $\Delta X_{20}[6] =$ `0xd` is a 4-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{107.34} \cdot 2^8 \cdot 2^{-4} \cdot 2^{-4} = s \cdot 2^{107.34}$ quartets remain.

So for each quartet, $\varepsilon = 2^4 \cdot \frac{4}{25} \cdot 3 + 2^{-4} \cdot 2^4 \cdot \frac{4}{25} + 2^{-8} \cdot 2^4 \cdot \frac{4}{25} + 2^{-8} \cdot 2^8 \cdot \frac{4}{25} \cdot 3 \approx 2^{3.06}$ and $T_2 = s \cdot 2^{118.4}$.

In order to balance T_1, T_2, T_3 and memory complexity and achieve a high success probability, we set $s = 1$, $h = 24$ and $x = 52$ ($x \leq m_b + m'_f = 52$, $h \leq 112 - x$). We have $T_1 = 2^{112.67}, T_2 = 2^{118.4}$ and $T_3 = 2^{104}$. The memory complexity is $2^{61.67} + 2^{64} \approx 2^{62.79}$.

In total, for the 25-round attack on `SKINNY-64-128`, the data complexity is $2^{61.67}$, the memory complexity is $2^{64.26}$, and the time complexity is $2^{118.43}$. The success probability is about 76.9%.

C.3 Improved attack on 25-round `SKINNY-128-256`

We use the 18-round rectangle distinguisher for `SKINNY-128-256` given in Sect. 4.3, whose probability is $2^{-n}\tilde{p}^2 t \tilde{q}^2 = 2^{-128-108.51} = 2^{-236.51}$. Appending 3-round E_b

Table 10: Tweakey recovery for 25-round SKINNY-64-128, where the red bytes are among k'_f or obtained in the last steps.

Step	Internal state	Involved subtweakeys
1	$Z_{23}[3]$	$STK_{24}[7]$
	$Z_{23}[15]$	$STK_{24}[2]$
2	$Z_{22}[2]$	$STK_{23}[6]$, $STK_{24}[7]$
	$Z_{22}[14]$	$STK_{23}[1]$, $STK_{24}[0,5]$
3	$Z_{22}[6]$	$STK_{23}[7]$, $STK_{24}[2,4,5]$
4	$Z_{22}[5]$	$STK_{23}[6]$, $STK_{24}[1,4,7]$
	$Z_{22}[9]$	$STK_{23}[7]$, $STK_{24}[2,4]$
	$Z_{22}[13]$	$STK_{23}[0]$, $STK_{24}[3,4]$
5	$Z_{21}[2]$	$STK_{22}[6]$, $STK_{23}[7]$, $STK_{24}[2,4,5]$
	$Z_{21}[14]$	$STK_{22}[1]$, $STK_{23}[0,5]$, $STK_{24}[3,4,6]$
6	$Z_{21}[6]$	$STK_{22}[7]$, $STK_{23}[2,4,5]$, $STK_{24}[0,1,3,5,6]$
	$Z_{20}[2]$	$STK_{21}[6]$, $STK_{22}[7]$, $STK_{23}[2,4,5]$, $STK_{24}[0,1,3,5,6]$
7	$Z_{20}[14]$	$STK_{21}[1]$, $STK_{22}[0,5]$, $STK_{23}[3,4,6]$, $STK_{24}[1,2,4,5,7]$
8	$Z_{20}[6]$	$STK_{21}[7]$, $STK_{22}[2,4,5]$, $STK_{23}[0,1,3,5,6]$, $STK_{24}[0,1,2,3,4,5,6,7]$

and 4-round E_f, we attack 25-round SKINNY-128-256, as illustrated in Fig. 18. There are $r_b = 4 \cdot 8 = 32$ by W'_0, $m_b = 5 \cdot 8 = 40$, $r_f = 12 \cdot 8 = 96$, $m_f = 21 \cdot 8 = 168$, $h_f = 7 \cdot 8 = 56$ and $m'_f = 3 \cdot 8 = 24$. Due to the tweakey schedule, we can deduce $STK_{22}[6]$ from $ETK_0[0]$ and $STK_{24}[5]$. So there are only $(m_f - c) = 160$-bit subtweakey unknown in E_f. As shown in Fig. 18, there are $k'_f = \{STK_{24}[2,5,6]\}$ and $h_f = \{W_{24}[5,10,14], W_{25}[4,5,10,15]\}$.

The attack follows Algorithm 1 in Sect. 3.2, where $y = \sqrt{s} \cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2 t \tilde{q}^2} = \sqrt{s} \cdot 2^{86.25}$. Hence, the data complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2 t \tilde{q}^2} = \sqrt{s} \cdot 2^{120.25}$.

According to Eq. (13), $T_1 = \sqrt{s} \cdot 2^{m_b+m'_f+n/2+1}/\sqrt{\tilde{p}^2 t \tilde{q}^2} = \sqrt{s} \cdot 2^{40+24+64+1+54.25} = \sqrt{s} \cdot 2^{183.25}$.

According to Eq. (15), $T_2 = s \cdot 2^{m_b+m'_f-2h_f+n}/(\tilde{p}^2 t \tilde{q}^2) \cdot \varepsilon = s \cdot 2^{40+24-112+128+108.51} \cdot \varepsilon = s \cdot 2^{188.51} \cdot \varepsilon$.

According to Eq. (16), $T_3 = 2^{k-h}$, where $h \leq m_b + m_f - c - x = 200 - x$.

According to Eq. (17), the memory complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2 t \tilde{q}^2} + 2^{m_b+m_f-c-x} = \sqrt{s} \cdot 2^{120.25} + 2^{200-x}$.

The tweakey recovery process with time ε. For each of the $s \cdot 2^{188.51}$ quartets, we determine the key candidates and increase the corresponding counters with the following steps:

1. **In round 24:** for the 4th column of X_{24} of (C_1, C_3), we can obtain $\Delta X_{24}[3] = \Delta X_{24}[15]$ from $\Delta W_{23}[15] = 0$. Since $\Delta Y_{24}[3]$ and $\Delta X_{24}[15]$ are known, we deduce $STK_{24}[3]$ by Lemma 1. Similarly, we deduce $STK'_{24}[3]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{188.51} \cdot 2^{-8} = s \cdot 2^{180.51}$ quartets remain.
2. Guessing $STK_{24}[1]$, we compute $Z_{23}[2,14]$. Then $\Delta Y_{23}[2]$ and $\Delta X_{23}[14]$ are deduced. For the 3rd column of X_{23} of (C_1, C_3), we can obtain $\Delta X_{23}[2] = \Delta X_{23}[14]$. Hence, we obtain $\Delta X_{23}[2]$ and deduce $STK_{23}[2]$ by Lemma 1. Similarly, we deduce $STK'_{23}[2]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{180.51} \cdot 2^8 \cdot 2^{-8} = s \cdot 2^{180.51}$ quartets remain.

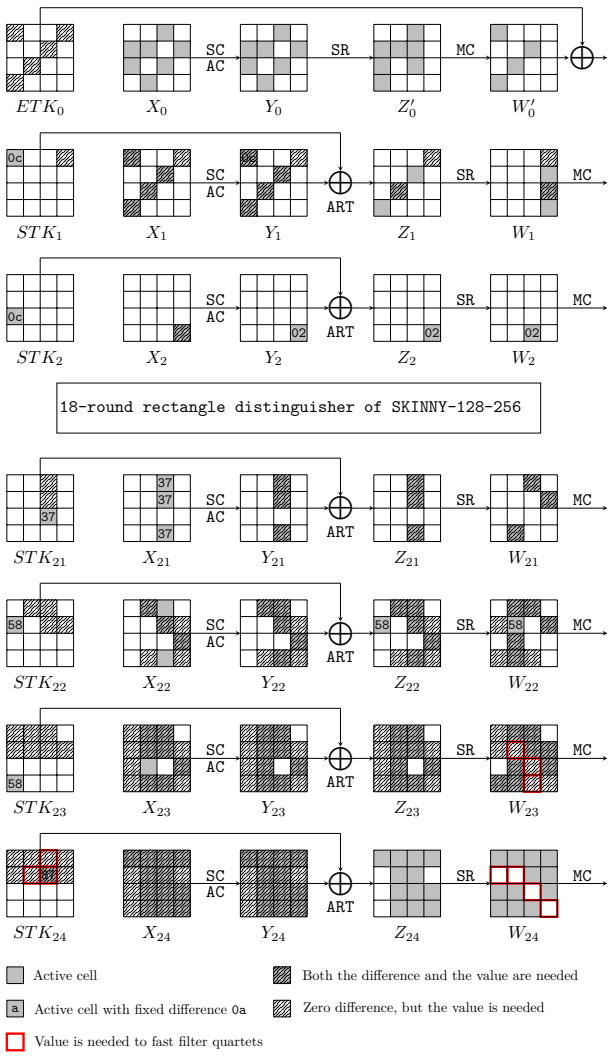

Fig. 18: The 25-round attack against SKINNY-128-256.

3. Guessing $STK_{24}[7]$, we compute $Z_{23}[6]$. Then $\Delta Y_{23}[6]$ is deduced. For the 3rd column of X_{23} of (C_1, C_3), we can obtain $\Delta X_{23}[6] = \Delta X_{23}[14]$. Hence, we obtain $\Delta X_{23}[6]$ and deduce $STK_{23}[6]$ by Lemma 1. Similarly, we deduce $STK'_{23}[6]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{180.51} \cdot 2^8 \cdot 2^{-8} = s \cdot 2^{180.51}$ quartets remain.
4. Guessing $STK_{24}[0]$, we compute $Z_{23}[5, 9, 13]$. Then $\Delta Y_{23}[5]$ and $\Delta X_{23}[9, 13]$ are deduced. For the 2nd column of X_{23} of (C_1, C_3), we can obtain $\Delta X_{23}[5] = \Delta X_{23}[9] \oplus \Delta X_{23}[13] \oplus \texttt{0x58}$. Hence, we obtain $\Delta X_{23}[5]$ and deduce $STK_{23}[5]$ by Lemma 1. Similarly, we deduce $STK'_{23}[5]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{180.51} \cdot 2^8 \cdot 2^{-8} = s \cdot 2^{180.51}$ quartets remain.
5. **In round 23:** guessing $STK_{23}[1]$, we compute $Z_{22}[2, 14]$. Then $\Delta Y_{22}[2]$ and $\Delta X_{22}[14]$ are deduced. For the 3rd column of X_{22} of (C_1, C_3), we can obtain $\Delta X_{22}[2] = \Delta X_{22}[14]$. Hence, we obtain $\Delta X_{22}[2]$ and deduce $STK_{22}[2]$. Similarly, we deduce $STK'_{22}[2]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{180.51} \cdot 2^8 \cdot 2^{-8} = s \cdot 2^{180.51}$ quartets remain.
6. Guessing $STK_{23}[7]$ and $STK_{24}[4]$, we compute $Z_{22}[6]$. Since $STK_{22}[6]$ from $ETK_0[0]$ and $STK_{24}[5]$, then $X_{22}[6]$ is deduced. Hence, $\Delta X_{23}[6] = \Delta X_{23}[14]$ is an 8-bit filter for both (C_1, C_3) and (C_2, C_4). Thereafter, **in round 22**, we compute $Z_{21}[2]$ from $X_{22}[6]$. Since $\Delta X_{21}[2] = \texttt{0x37}$, we deduce $STK_{21}[2]$ for (C_1, C_3) and $STK_{21}[2]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{180.51} \cdot 2^{16} \cdot 2^{-8} \cdot 2^{-16} = s \cdot 2^{172.51}$ quartets remain.
7. Guessing $STK_{23}[0]$ and $STK_{22}[1]$, we compute $Z_{21}[14]$ and deduce $X_{21}[14]$. $\Delta X_{21}[14] = \texttt{0x37}$ is an 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{172.51} \cdot 2^{16} \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{172.51}$ quartets remain.
8. Guessing $STK_{23}[4]$ and $STK_{22}[7]$, we compute $Z_{21}[6]$. Since $\Delta X_{21}[6] = \texttt{0x37}$, we deduce $STK_{21}[6]$ for (C_1, C_3) and $STK_{21}[6]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{172.51} \cdot 2^{16} \cdot 2^{-8} = s \cdot 2^{180.51}$ quartets remain.

So for each quartet, $\varepsilon = \frac{4}{25} + 2^{-8} \cdot 2^8 \cdot \frac{4}{25} \cdot 4 + 2^{-8} \cdot 2^{16} \cdot \frac{4}{25} + 2^{-16} \cdot 2^{16} \cdot \frac{4}{25} \cdot 2 \approx 2^{5.4}$ and $T_2 = s \cdot 2^{193.91}$.

In order to balance T_1, T_2, T_3 and memory complexity and achieve a high success probability, we set $s = 1$, $h = 80$ and $x = 64$ ($x \leq m_b + m'_f = 64$, $h \leq 200 - x$). We have $T_1 = 2^{183.25}, T_2 = 2^{193.91}$ and $T_3 = 2^{176}$. The memory complexity is $2^{120.25} + 2^{136} \approx 2^{136}$.

In total, for the 25-round attack on SKINNY-128-256, the data complexity is $2^{120.25}$, the memory complexity is 2^{136}, and the time complexity is $2^{193.91}$. The success probability is about 83.8%.

C.4 Improved attack on 26-round SKINNY-128-256

We use the 19-round rectangle distinguisher for SKINNY-128-256 given in Sect. 4.3, whose probability is $2^{-n}\tilde{p}^2 t \tilde{q}^2 = 2^{-128-121.07} = 2^{-249.07}$. Appending 3-round E_b and 4-round E_f, we attack 26-round SKINNY-128-256, as illustrated in Fig. 19. There are $r_b = 9 \cdot 8 = 72$ by W'_0, $m_b = 11 \cdot 8 = 88$, $r_f = 12 \cdot 8 = 96$, $m_f = 21 \cdot 8 = 168$, $h_f = 8 \cdot 8 = 64$ and $m'_f = 4 \cdot 8 = 32$. Due to the tweakey schedule, we can deduce $STK_{22}[2, 6]$ from $ETK_0[8, 0]$ and $STK_{24}[4, 5]$. So there

Table 11: Tweakey recovery for 25-round SKINNY-128-256, where the red bytes are among k'_f or obtained in the last steps.

Step	Internal state	Involved subtweakeys
2	$Z_{23}[2]$	$STK_{24}[6]$
	$Z_{23}[14]$	$STK_{24}[1]$
3	$Z_{23}[6]$	$STK_{24}[7]$
4	$Z_{23}[5]$	$STK_{24}[6]$
	$Z_{23}[9]$	$STK_{24}[7]$
	$Z_{23}[13]$	$STK_{24}[0]$
5	$Z_{23}[2]$	$STK_{23}[6]$, $STK_{24}[7]$
	$Z_{22}[14]$	$STK_{23}[1]$, $STK_{24}[0,5]$
6	$Z_{22}[6]$	$STK_{23}[7]$, $STK_{24}[2,4,5]$
	$Z_{21}[2]$	$STK_{22}[6]$, $STK_{23}[7]$, $STK_{24}[2,4,5]$
7	$Z_{21}[14]$	$STK_{22}[1]$, $STK_{23}[0,5]$, $STK_{24}[3,4,6]$
8	$Z_{21}[6]$	$STK_{22}[7]$, $STK_{23}[2,4,5]$, $STK_{24}[0,1,3,5,6]$

are only $(m_f - 2c) = 152$-bit subtweakey unknown in E_f. As shown in Fig. 19, there are $k'_f = \{STK_{24}[2,3,5,6]\}$ and $h_f = \{W_{24}[5,10,14], W_{25}[4,5,10,15]\}$.

The attack follows Algorithm 1 in Sect. 3.2, where $y = \sqrt{s}\cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2 t\tilde{q}^2} = \sqrt{s}\cdot 2^{52.53}$. Hence, the data complexity is $\sqrt{s}\cdot 2^{n/2+2}/\sqrt{\tilde{p}^2 t\tilde{q}^2} = \sqrt{s}\cdot 2^{126.53}$.

According to Eq. (13), $T_1 = \sqrt{s}\cdot 2^{m_b+m'_f+n/2+1}/\sqrt{\tilde{p}^2 t\tilde{q}^2} = \sqrt{s}\cdot 2^{88+32+64+1+60.53} = \sqrt{s}\cdot 2^{245.53}$.

According to Eq. (15), $T_2 = s\cdot 2^{m_b+m'_f-2h_f+n}/(\tilde{p}^2 t\tilde{q}^2)\cdot\varepsilon = s\cdot 2^{88+32-128+128+121.07}\cdot\varepsilon = s\cdot 2^{241.07}\cdot\varepsilon$.

According to Eq. (16), $T_3 = 2^{k-h}$, where $h \le m_b + m_f - 2c - x = 240 - x$.

According to Eq. (17), the memory complexity is $\sqrt{s}\cdot 2^{n/2+2}/\sqrt{\tilde{p}^2 t\tilde{q}^2} + 2^{m_b+m_f-2c-x} = \sqrt{s}\cdot 2^{126.53} + 2^{240-x}$.

The tweakey recovery process with time ε. For each of the $s\cdot 2^{241.07}$ quartets, we determine the key candidates and increase the corresponding counters with the following steps:

1. **In round 25:** guessing $STK_{25}[1]$, we compute $Z_{24}[2,14]$. Then $\Delta Y_{24}[2]$ and $\Delta X_{24}[14]$ are deduced. For the 3rd column of X_{24} of (C_1, C_3), we can obtain $\Delta X_{24}[2] = \Delta X_{24}[14]$. Hence, we obtain $\Delta X_{24}[2]$ and deduce $STK_{24}[2]$ by Lemma 1. Similarly, we deduce $STK'_{24}[2]$ for (C_2, C_4), which is an 8-bit filter. $s\cdot 2^{241.07}\cdot 2^8\cdot 2^{-8} = s\cdot 2^{241.07}$ quartets remain.
2. Guessing $STK_{25}[7]$, we compute $Z_{24}[6]$. Then $\Delta Y_{24}[6]$ is deduced. For the 3rd column of X_{24} of (C_1, C_3), we can obtain $\Delta X_{24}[6] = \Delta X_{24}[14]$. Hence, we obtain $\Delta X_{24}[6]$ and deduce $STK_{24}[6]$ by Lemma 1. Similarly, we deduce $STK'_{24}[6]$ for (C_2, C_4), which is an 8-bit filter. $s\cdot 2^{241.07}\cdot 2^8\cdot 2^{-8} = s\cdot 2^{241.07}$ quartets remain.
3. Guessing $STK_{25}[0]$, we compute $Z_{24}[5,9,13]$. Then $\Delta Y_{24}[5]$ and $\Delta X_{24}[9,13]$ are deduced. For the 2nd column of X_{24} of (C_1, C_3), we can obtain $\Delta X_{24}[5] = \Delta X_{24}[9] \oplus \Delta X_{24}[13] \oplus$ 0x82. Hence, we obtain $\Delta X_{24}[5]$ and deduce $STK_{24}[5]$ by Lemma 1. Similarly, we deduce $STK'_{24}[5]$ for (C_2, C_4), which is an 8-bit filter. $s\cdot 2^{241.07}\cdot 2^8\cdot 2^{-8} = s\cdot 2^{241.07}$ quartets remain.

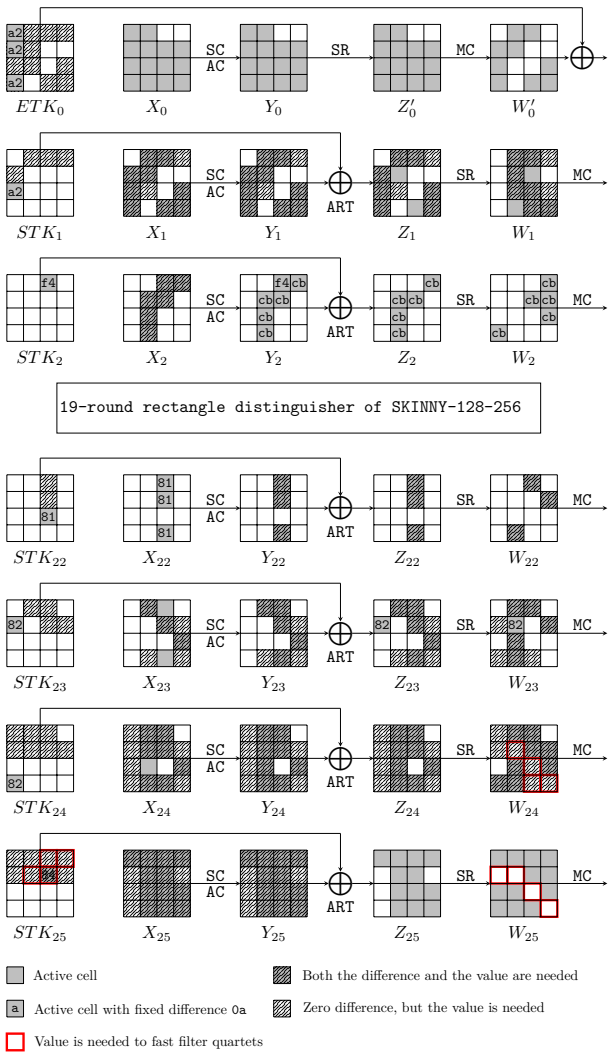

Fig. 19: The 26-round attack against SKINNY-128-256.

4. **In round 24:** guessing $STK_{24}[1]$, we compute $Z_{23}[2,14]$. Then $\Delta Y_{23}[2]$ and $\Delta X_{23}[14]$ are deduced. For the 3rd column of X_{23} of (C_1, C_3), we can obtain $\Delta X_{23}[2] = \Delta X_{23}[14]$. Hence, we obtain $\Delta X_{23}[2]$ and deduce $STK_{23}[2]$. Similarly, we deduce $STK'_{23}[2]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{241.07} \cdot 2^8 \cdot 2^{-8} = s \cdot 2^{241.07}$ quartets remain.
5. Guessing $STK_{24}[7]$ and $STK_{25}[4]$, we compute $Z_{23}[6]$. Then $\Delta Y_{23}[6]$ is deduced. For the 3rd column of X_{23} of (C_1, C_3), we can obtain $\Delta X_{23}[6] = \Delta X_{23}[14]$. Hence, we obtain $\Delta X_{23}[6]$ and deduce $STK_{23}[6]$. Similarly, we deduce $STK'_{23}[6]$ for (C_2, C_4), which is an 8-bit filter. Thereafter, **in round 23**, we compute $Z_{22}[2]$ from $X_{23}[6]$. Since $STK_{22}[2]$ can be deduced from known subtweakeys, $\Delta X_{22}[2] =$ `0x81` is an 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{241.07} \cdot 2^{16} \cdot 2^{-8} \cdot 2^{-16} = s \cdot 2^{233.07}$ quartets remain.
6. Guessing $STK_{24}[0]$ and $STK_{23}[1]$, we compute $Z_{22}[14]$ and deduce $X_{22}[14]$. $\Delta X_{22}[14] =$ `0x81` is an 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{233.07} \cdot 2^{16} \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{233.07}$ quartets remain.
7. Guessing $STK_{24}[4]$ and $STK_{23}[7]$, we compute $Z_{22}[6]$. Since $STK_{22}[6]$ can be deduced from known subtweakeys, $\Delta X_{22}[6] =$ `0xd` is an 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{233.07} \cdot 2^{16} \cdot 2^{-16} = s \cdot 2^{233.07}$ quartets remain.

So for each quartet, $\varepsilon = 2^8 \cdot \frac{4}{26} \cdot 4 + 2^{16} \cdot \frac{4}{26} + 2^{-8} \cdot 2^{16} \cdot \frac{4}{26} + 2^{-8} \cdot 2^{16} \cdot \frac{4}{26} \approx 2^{13.33}$ and $T_2 = s \cdot 2^{254.4}$.

In order to balance T_1, T_2, T_3 and memory complexity and achieve a high success probability, we set $s = 1$, $h = 24$ and $x = 112$ ($x \leq m_b + m'_f = 112$, $h \leq 240 - x$). We have $T_1 = 2^{245.53}, T_2 = 2^{254.4}$ and $T_3 = 2^{232}$. The memory complexity is $2^{126.53} + 2^{128} \approx 2^{128.44}$.

In total, for the 26-round attack on `SKINNY-128-256`, the data complexity is $2^{126.53}$, the memory complexity is $2^{128.44}$, and the time complexity is $2^{254.4}$. The success probability is about 69.8%.

Table 12: Tweakey recovery for 26-round `SKINNY-128-256`, where the red bytes are among k'_f or obtained in the last steps.

Step	Internal state	Involved subtweakeys
1	$Z_{24}[2]$	$STK_{25}[6]$
	$Z_{24}[14]$	$STK_{25}[1]$
2	$Z_{24}[6]$	$STK_{25}[7]$
3	$Z_{24}[5]$	$STK_{25}[6]$
	$Z_{24}[9]$	$STK_{25}[7]$
	$Z_{24}[13]$	$STK_{25}[0]$
4	$Z_{23}[2]$	$STK_{24}[6]$, $STK_{25}[7]$
	$Z_{23}[14]$	$STK_{24}[1]$, $STK_{25}[0, 5]$
5	$Z_{23}[6]$	$STK_{24}[7]$, $STK_{25}[2, 4, 5]$
	$Z_{22}[2]$	$STK_{23}[6]$, $STK_{24}[7]$, $STK_{25}[2, 4, 5]$
6	$Z_{22}[14]$	$STK_{23}[1]$, $STK_{24}[0, 5]$, $STK_{25}[3, 4, 6]$
7	$Z_{22}[6]$	$STK_{23}[7]$, $STK_{24}[2, 4, 5]$, $STK_{25}[0, 1, 3, 5, 6]$

D Early abort technique with table

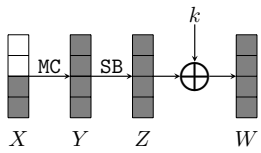

Fig. 20: Reducing complexity of ε.

At CT-RSA 2008, Lu, Kim, Keller and Dunkelman [37] introduced the early abort technique for impossible differential attack. They partially check whether a candidate pair could produce the expected difference by guessing only a small fraction of the unknown required subkey bits at a time. Since some useless pairs can be discarded before the next guess for a different fraction of the required round subkey bits, they reduce the computational workload for an attack.

Take Fig. 20 as an example of the the early abort technique adopted in differential attack. W is the ciphertext and gray bytes are active. Guessing three bytes of k, e.g. $k[0,1,2]$, to partially decrypt a pair to get $\Delta Y[0,1,2]$. Since $\Delta X[0,1] = 0$, with property of MC, we get a filter of 2^{-8}. Then, guess $k[3]$ for the left pairs. The time complexity ε is bounded by the first guess of $k[0,1,2]$, i.e., $\varepsilon = 2^{24}$.

Early abort technique with table. Using the hash tables to improve the early abort technique is a tradeoff between the time and space. With table, we reduce ε from 2^{24} to 1 in this example. We prepare a hash table T_ε,

1. For each of the 2^{64} values of (W, W'),
2. For each of the 2^{32} key of k, decrypt (W, W') to get ΔX. If $\Delta X[0,1] = 0$, store k in T_ε indexed by (W, W').

Under each index of T_ε, there expect $2^{32-16} = 2^{16}$ values for k. When using T_ε to perform the key recovery attack, we use each ciphertext pair (W, W') to access T_ε to get the possible key candidates indexed by (W, W'). Here, ε becomes one memory access. To prepare the T_ε, we need a time complexity of 2^{96} and a memory complexity of 2^{80}.

When applying the Early abort technique with table to our rectangle attack, we have to prepare the table T_ε for quartets. Suppose the quartets are (W_1, W_2, W_3, W_4), where (X_1, X_3) and (X_2, X_4) meet the conditions that the first two bytes are inactive. Then we construct T_ε as follows:

1. For each of the 2^{128} values of (W_1, W_2, W_3, W_4),
2. For each of the 2^{32} key of k, decrypt (W_1, W_3) to get $\Delta X = X_1 \oplus X_3$ and $\Delta X' = X_2 \oplus X_4$. If $\Delta X[0,1] = \Delta X'[0,1] = 0$, store k in T_ε indexed by (W_1, W_2, W_3, W_4).

Under each index, there expect $2^{32-16-16} = 1$ key candidate for k. Then in the key-recovery phase, for each (W_1, W_2, W_3, W_4), we just access the T_ε to find the candidate key. To prepare the T_ε, we need a time complexity of 2^{160} and a memory complexity of 2^{128}.

E Application to ForkSkinny

ForkSkinny is designed by Andreeva *et al.* [1], which is the internal primitive of ForkAE [2], a 2nd round candidate in the NIST LWC project. ForkSkinny applies the round function of SKINNY $R_{\mathtt{init}}$ times to the plaintext P, then forks the state, and computes two ciphertexts independently: applying $R_{\mathtt{I}}$ rounds in the first branch to get C_I and $R_{\mathtt{II}}$ rounds in the second branch to get C_{II}. The subkeys are generated by extending the tweakey schedule to produce $R_{\mathtt{init}}+R_{\mathtt{I}}+R_{\mathtt{II}}$ subkeys. At ToSC 2020, Bariant, David and Leurent [6] gave the related-key impossible differential attack on (7+19=) 26-round reduced ForkSkinny-128-256, where $R_{\mathtt{init}} = 7$, $R_{\mathtt{I}} = 27$, $R_{\mathtt{II}} = 19$.

By tweaking our automatic model on SKINNY into ForkSkinny, we find that the 21-round distinguisher on ForkSkinny-128-256 used in [40] is also the optimal distinguisher for our new rectangle attack model. The probability of the 21-round distinguisher with $R_{\mathtt{init}} = 17$, $R_{\mathtt{I}} = 27$ and $R_{\mathtt{II}} = 4$ is $2^{-n}p^2tq^2 = 2^{-128-105.77} = 2^{-233.77}$. With our new attack framework, we reduce the time complexity of the previous 28-round attack ($R_{\mathtt{init}} = 7$, $R_{\mathtt{I}} = 27$, $R_{\mathtt{II}} = 21$) on ForkSkinny-128-256 with 256-bit key.

28-round attack on ForkSkinny-128-256 with 256-bit key. Appending 3-round E_b and 4-round E_f to the 21-round distinguisher, we attack 28-round ForkSkinny-128-256, as illustrated in Figure 21. There are $r_b = 8 \cdot 8 = 64$ by W'_0, $m_b = 10 \cdot 8 = 80$, $r_f = 12 \cdot 8 = 96$ and $m_f = 17 \cdot 8 = 136$. Due to the tweakey schedule, we can deduce $STK_{52}[2, 6]$ from $ETK_0[5, 2]$ and $STK_{54}[4, 5]$. So there are only $(m_f - 2c) = 120$-bit subtweakey unknown in E_f. From the automatic search model, we get $h_f = 7 \cdot 8 = 56$ and $m'_f = 3 \cdot 8 = 24$. where $k'_f = \{STK_{54}[0, 4, 5]\}$ and $h_f = \{X_{26}[13], W_{26}[8, 9], W_{25}[4, 10, 11, 14]\}$.

The attack follows Algorithm 1 in Section 3.2, where $y = \sqrt{s} \cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2t\tilde{q}^2} = \sqrt{s} \cdot 2^{52.88}$. Hence, the data complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2t\tilde{q}^2} = \sqrt{s} \cdot 2^{118.88}$.

According to Eq. (13), $T_1 = \sqrt{s} \cdot 2^{m_b+m'_f+n/2+1}/\sqrt{\tilde{p}^2t\tilde{q}^2} = \sqrt{s} \cdot 2^{80+24+64+1+52.88} = \sqrt{s} \cdot 2^{221.88}$.

According to Eq. (15), $T_2 = s \cdot 2^{m_b+m'_f-2h_f+n}/(\tilde{p}^2t\tilde{q}^2) \cdot \varepsilon = s \cdot 2^{80+24-112+128+105.77} \cdot \varepsilon = s \cdot 2^{225.77} \cdot \varepsilon$.

According to Eq. (16), $T_3 = 2^{k-h}$, where $h \leq m_b + m_f - 2c - x = 200 - x$.

According to Eq. (17), the memory complexity is $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2t\tilde{q}^2} + 2^{m_b+m_f-2c-x} = \sqrt{s} \cdot 2^{118.88} + 2^{200-x}$.

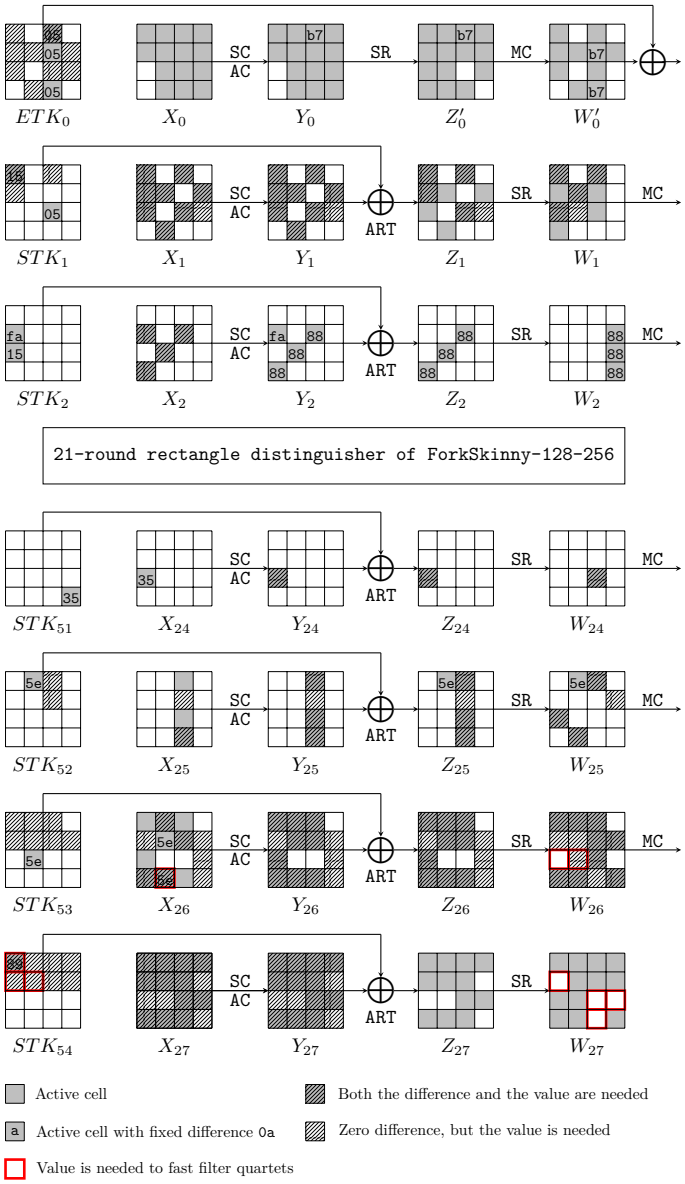

Fig. 21: The 28-round attack against `ForkSkinny-128-256`

The tweakey recovery process with time ε. For each of the $s \cdot 2^{225.77}$ quartets, we determine the key candidates and increase the corresponding counters with the following steps:

1. **In round 27:** for the 3rd column of X_{27} of (C_1, C_3), we can obtain $\Delta X_{27}[2] \oplus \Delta X_{27}[14] = \Delta W_{26}[14] = 0$. Then we obtain $\Delta X_{27}[2]$ and deduce $STK_{54}[2]$ by Lemma 1. Similarly, we deduce $STK'_{54}[2]$ for (C_2, C_4), which an 8-bit filter. $s \cdot 2^{225.77} \cdot 2^{-8} = s \cdot 2^{217.77}$ quartets remain.
2. Guessing $STK_{54}[6]$, we compute $Z_{26}[5]$. Then $\Delta Y_{26}[5]$ is deduced. Since $\Delta X_{26}[5] = \texttt{0x5e}$, we deduce $STK_{53}[5]$ for (C_1, C_3) and $STK'_{53}[5]$ for (C_2, C_4), which is an 8-bit filter. $s \cdot 2^{217.77} \cdot 2^{8} \cdot 2^{-8} = s \cdot 2^{217.77}$ quartets remain.
3. Guessing $STK_{54}[3]$ and together with known subtweakeys as in Table 13, we compute $Z_{26}[0, 8, 12]$. Then $\Delta X_{26}[8, 12]$ and $\Delta Y_{26}[0]$ are deduced. For the 1st column of X_{26} of (C_1, C_3), we can obtain $\Delta X_{26}[0] = \Delta X_{26}[8] = \Delta X_{26}[12]$. Hence, we obtain $\Delta X_{26}[0]$ and deduce $STK_{53}[0]$ by Lemma 1. Similarly, we deduce $STK'_{53}[0]$ for (C_2, C_4), which is an 8-bit filter. $\Delta X_{26}[8] = \Delta X_{26}[12]$ is an 8-bit filter both for (C_1, C_3) and (C_2, C_4). $s \cdot 2^{217.77} \cdot 2^{8} \cdot 2^{-8} \cdot 2^{-16} = s \cdot 2^{201.77}$ quartets remain.
4. Guessing $STK_{54}[1, 7]$, we compute $Z_{26}[2, 6, 14]$ and peel off round 27. Then $\Delta X_{26}[14]$ and $\Delta Y_{26}[2, 6]$ are deduced. For the 3rd column of X_{26} of (C_1, C_3), we can obtain $\Delta X_{26}[2] = \Delta X_{26}[6] = \Delta X_{26}[14]$. Hence, we obtain $\Delta X_{26}[2, 6]$ and deduce $STK_{53}[2, 6]$ by Lemma 1. Similarly, we deduce $STK'_{53}[2, 6]$ for (C_2, C_4), which is a 16-bit filter. $s \cdot 2^{201.77} \cdot 2^{16} \cdot 2^{-16} = s \cdot 2^{201.77}$ quartets remain.
5. **In round 26:** Guessing $STK_{53}[1, 4]$, we compute $Z_{25}[2, 10, 14]$. Then $\Delta X_{25}[10, 14]$ and $\Delta Y_{25}[2]$ are deduced. For the 3rd column of X_{25} of (C_1, C_3), we can obtain $\Delta X_{25}[2] = \Delta X_{25}[10] = \Delta X_{25}[14]$. Since $STK_{52}[2]$ can be deduced from $ETK_0[5]$ and $STK_{54}[4]$, we can compute $X_{25}[2]$ and $\Delta X_{25}[2]$. Then $\Delta X_{25}[2] = \Delta X_{25}[14]$ and $\Delta X_{25}[10] = \Delta X_{25}[14]$ are two 8-bit filters both for (C_1, C_3) and (C_2, C_4). $s \cdot 2^{201.77} \cdot 2^{16} \cdot 2^{-16} \cdot 2^{-16} = s \cdot 2^{185.77}$ quartets remain.
6. Guessing $STK_{53}[7]$, we compute $Z_{24}[8]$ and deduce $X_{24}[8]$. $\Delta X_{24}[8] = \texttt{0x35}$ is an 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{185.77} \cdot 2^{8} \cdot 2^{-16} = s \cdot 2^{177.77}$ quartets remain.

So for each quartet, $\varepsilon = \frac{4}{28} + 2^{-8} \cdot 2^{8} \cdot \frac{4}{28} \cdot 2 + 2^{-24} \cdot 2^{16} \cdot \frac{4}{28} \cdot 2 + 2^{-40} \cdot 2^{8} \cdot \frac{4}{28} \approx 2^{-1.22}$. So $T_2 = s \cdot 2^{224.55}$.

In order to balance T_1, T_2, T_3 and memory complexity as well as achieve a high success probability, we set $s = 1$, $h = 48$ and $x = 96$ ($x \leq m_b + m'_f = 104$, $h \leq 200 - x$) with Eq. (8). Then we have $T_1 = 2^{221.88}, T_2 = 2^{224.55}$ and $T_3 = 2^{208}$.

In total, the data complexity is $2^{118.88}$, the memory complexity is $2^{118.88}$, and the time complexity is $2^{224.76}$. The success probability is about 84.0%.

F Application to Deoxys-BC-384

Deoxys-BC is the internal tweakable block cipher of Deoxys-II [32], which is among the final portfolio of CAESAR competition*. We only consider the ver-

*https://competitions.cr.yp.to/caesar-submissions.html

Table 13: Tweakey recovery for 28-round `Forkskinny-128-256`, where the red bytes are among k'_f or obtained in the last steps.

Step	Internal state	Involved subtweakeys
2	$Z_{26}[5]$	$STK_{54}[6]$
3	$Z_{26}[0]$	$STK_{54}[4]$
	$Z_{26}[8]$	$STK_{54}[6]$
	$Z_{26}[12]$	$STK_{54}[3]$
4	$Z_{26}[2]$	$STK_{54}[7]$
	$Z_{26}[6]$	$STK_{54}[7]$
	$Z_{26}[14]$	$STK_{54}[1]$
5	$Z_{25}[2]$	$STK_{53}[6]$, $STK_{54}[7]$
	$Z_{25}[10]$	$STK_{53}[4]$, $STK_{54}[3,5]$
	$Z_{25}[14]$	$STK_{53}[1]$, $STK_{54}[0,5]$
6	$Z_{24}[8]$	$STK_{52}[6]$, $STK_{53}[1,7]$, $STK_{54}[0,2,4,5]$

sion `Deoxys-BC-384` here. The framework is similar to `SKINNY`-n-$3n$ as shown in Figure 7, by replacing the `SKINNY`'s round function with `AES`'s in Figure 22.

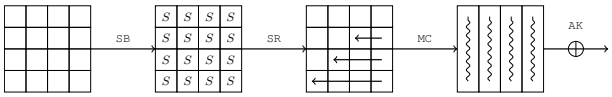

Fig. 22: Round function of `Deoxys-BC-384`

There have been many cryptanalysis results [22,42,51,52] on reduced `Deoxys-BC`. The best attack on reduced `Deoxys-BC-384` is a 14-round attack with time $2^{282.7}$ and data $2^{125.2}$ by Zhao *et al.* [52] at INDOCRYPT 2019. In this section, we improve this attack with `Attack VI`. The distinguisher, the attack map (shown in Figure 23) and the way to choose plaintexts are all the same to [52]. We have the parameters: $\hat{p}\hat{q} = 2^{-59.2}$, $r_b = m_b = 12 \times 8 = 96$, $r_f = 96$, $m_f = 96 + 40 = 136$. According to Eq. (12), the data complexity is $\sqrt{s} \cdot 2^{64+2+59.2} = \sqrt{s} \cdot 2^{125.2}$. According to Eq. (13), $T_1 = \sqrt{s} \cdot 2^{m'_f} \cdot 2^{96+64+1+59.2} = \sqrt{s} \cdot 2^{m'_f} \cdot 2^{220.2}$. According to Eq. (15), $T_2 = s \cdot 2^{m'_f - 2h_f} \cdot 2^{96-128+2\times 96+59.2\times 2} \cdot \varepsilon = s \cdot 2^{m'_f - 2h_f} \cdot 2^{278.4} \cdot \varepsilon$.

As shown in Fig. 23, we guess the first column of $\mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(STK_{14})$ and the 5th byte of $\mathtt{SR}^{-1} \circ \mathtt{MC}^{-1}(STK_{13})$ to compute the internal states marked by red boxes of X_{13} as filters. Therefore, we get $2h_f = 8\times 8 = 64$ with $m'_f = 32+8 = 40$. Then, $T_2 = s \cdot 2^{254.4} \cdot \varepsilon$ and $T_1 = \sqrt{s} \cdot 2^{260.2}$.

To reduce the complexity of ε, we use the *early abort technique with table* technique in Section D to compute the key candidates for each quartet. We prepare two hash tables T_ε for the last two columns of X_{14} in the same way with Section D, which costs 2×2^{160} time and 2×2^{128} memory. By accessing T_ε, we can fast locate the key candidates by two table accesses for each quartet.

According to Eq. (16), $T_3 = 2^{384-h}$, where $m_b + m_f - x - h \geq 0$, i.e., $232 - x - h \geq 0$ and $x \leq m_b + m'_f = 136$. Let $h = 130$, then $T_3 = 2^{254}$. Let $x = 92$, then the memory complexity of storing the key counters is $2^{m_b + m_f - x} = 2^{140}$. Choose $s = 1$, the success probability is 51.2%. The overall time complexity

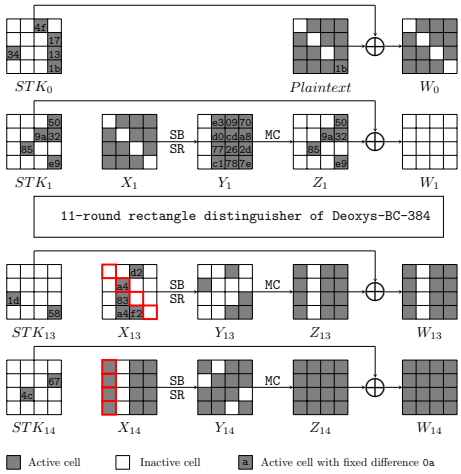

Fig. 23: Rectangle attack on 14-round reduced `Deoxys-BC-384`.

is about 2^{260} (previous $2^{282.7}$). The memory complexity is 2^{140} and the data complexity is the same to Zhao *et al.* [52].

G Application to `GIFT-64`

`GIFT` [4] was proposed by Banik *et al.* at CHES 2017. `GIFT` has an SPN structure. There are two versions for `GIFT` according to the block size *i.e.*, `GIFT-64` and `GIFT-128`. Both two versions adopt a 128-bit key. The numbers of rounds for `GIFT-64` and `GIFT-128` are 28 and 40, respectively. In this paper, we only consider `GIFT-64`. There are three operations in each round function, *i.e.*, `SubCells`, `PermBits` and `AddRoundKey`, whose detials are defined as follows:

1. `SubCells`: Apply 16 4-bit Sboxes (Table 14) in parallel to every nibble of the internal state of `GIFT-64`.

Table 14: The Sbox of `GIFT`

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$GS(x)$	1	a	4	c	6	f	3	9	2	d	b	7	5	0	8	e

2. `PermBits` : Linear bit permutations $b_{P(i)} \leftarrow b_i,\ \forall i \in \{0, 1, ...n-1\}$, where the $P(i)$s are

$$P_{64}(i) = 4\lfloor \frac{i}{16} \rfloor + 16\left(3\lfloor \frac{i \bmod 16}{4} \rfloor + (i \bmod 4) \bmod 4\right) + (i \bmod 4).$$

3. `AddRoundKey` : The round keys RK is $n/2$-bit, which is extracted from the key state, where $n = 64$ for `GIFT-64`. Let $RK = U||V = u_{s-1}...u_0||v_{s-1}v_0$, where $s = n/4$. For `GIFT-64`, the round key is XORed to the state as

$$b_{4i+1} \leftarrow b_{4i+1} \oplus u_i,\ b_{4i} \leftarrow b_{4i} \oplus v_i,\ \forall i \in \{0, ..., 15\}.$$

A single bit "1" and a 6-bit constant C are XORed into the internal state at positions $n-1$, 23, 19, 15, 11, 7 and 3 respectively.

The 128-bit master key is initialized as $K = k_7||k_6||...||k_0$, where $|k_i| = 16$. For `GIFT-64`, the round key $RK = U||V = k_1||k_0$. The key state is updated as follows,

$$k_7||k_6||...||k_0 \leftarrow (k_1 \ggg 2)||(k_0 \ggg 12)||...||k_3||k_2.$$

For more details of GIFT, we refer to [4].

At SAC 2020, Ji *et al.* [33] proposed a 20-round related-key boomerang distinguisher $(\alpha, \delta) =$ (`00 00 00 00 00 00 a0 00`, `04 00 00 00 01 20 10 00`) with

$$\Delta K = \texttt{0004 0000 0000 0800 0000 0000 0000 0010},$$
$$\nabla K = \texttt{2000 0000 0000 0000 0800 0000 0200 0800}.$$

The probability is $\hat{p}^2\hat{q}^2 r = 2^{-58.557}$. Based on it, they gave a 25-round related-key rectangle attack on `GIFT-64` with $2^{120.9}$ time and $2^{63.78}$ data and $2^{64.1}$ memory.

A 26-round attack on `GIFT-64`. For `GIFT-64`, we append one additional round at the bottom of Ji *et al.*'s 25-round attack to attack 26-round `GIFT-64` (see Table 15). The 128-bit key of `GIFT-64` is divided into eight 16-bit k_i $(0 \le i \le 7)$. In each round, a 32-bit $k_{2j+1}\|k_{2j}$ $(0 \le j \le 3)$ key or its bit permuted key $k'_{2j+1}\|k'_{2j}$ are XORed into the state. In our 26-round attack, the involved keys of E_b are 24 bits of $RK_1 = k_1\|k_0$ and 6 bits of $RK_2 = k_3\|k_2$, i.e., $m_b = 30$. In E_f, the involved keys are 32-bit $RK_{26} = k'_3\|k'_2$ and 24 bits of $RK_{25} = k'_1\|k'_0$ and 8 bits of $RK_{24} = k'_7\|k'_6$, i.e., $m_f = 32 + 24 + 8 = 64$.

Moreover, there are some key relations in E_b and E_f. In detail, for RK_1, 24-bit $k_0[0, 1, 2, 3, 4, 5, 6, 7, 12, 13, 14, 15]$ and $k_1[0, 1, 2, 3, 4, 5, 6, 7, 12, 13, 14, 15]$ are involved. For RK_{25}, 24-bit $k_0[4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$ and $k_1[0, 1, 2, 3, 4, 5, 6, 7, 12, 13, 14, 15]$. So $k_1[8, 9, 10, 11]$ does not need to be guessed and only $32 - 4 = 28$-bit $k_1\|k_0$ is involved in RK_1 and RK_{25}. So totally 28-bit $k_1\|k_0$, 32-bit $k_3\|k_2$ and 8-bit $k_7\|k_6$ are involved in E_b and E_f. Hence, $28+32+8 = 68$-bit key (instead of $m_b + m_f = 94$-bit key) are involved in E_b and E_f.

When applying `Attack VI`, we have $r_b = 44$, $r_f = 64$. According to Eq. (12), the data complexity is $\sqrt{s}\cdot 2^{63.28}$. According to Eq. (13), $T_1 = \sqrt{s}\cdot 2^{m_b+m'_f+62.28} = \sqrt{s}\cdot 2^{m'_f+92.28}$. According to Eq. (15), $T_2 = (s\cdot 2^{30+m'_f-64+2\times 64-2h_f+58.557})\cdot\varepsilon = s\cdot 2^{m'_f-2h_f}\cdot 2^{152.557}\cdot\varepsilon$. We choose m'_f to balance T_1 and T_2. Since in E_b, 6-bit $k_3\|k_2$ has been guessed. So to peel off the last round, we need to guess an additional $32 - 6 = 26$-bit key of $RK_{26} = k_3\|k_2$. Moreover, in E_b, 24-bit key of $k_1\|k_0$ has been guessed and with the above analysis, we need to guess only $28 - 24 = 4$-bit key of $RK_{25} = k'_1\|k'_0$ to peel off the round 25. Hence,

we need to guess $m'_f = 26 + 4 = 30$-bit key to filter the wrong quartets using the difference of internal state ΔX_{25} and we get $h_f = 49$. Therefore, we get $T_1 = \sqrt{s} \cdot 2^{30+92.28} = \sqrt{s} \cdot 2^{112.28}$ and $T_2 = s \cdot 2^{30-2\times 49} \cdot 2^{152.557} \cdot \varepsilon = s \cdot 2^{84.557} \cdot \varepsilon$.

According to Eq. (16), $T_3 = 2^{k-h}$. Since the total number of key bits involved in E_b and E_f is 68, h has to meet $68 - x - h \geq 0$ and $x \leq m_b + m'_f = 60$. According to Eq. (17), the memory complexity is $2^{68-x} + \sqrt{s} \cdot 2^{63.28}$. We choose $x = 8$, $h = 20$ and $s = 2$ in order to achieve a good success probability of 72.5% according to Eq. (8). Note that ε is about 4 Sbox computations by the early abort technique. At last, we have a time complexity of $2^{112.78}$ 26-round encryptions, a data complexity of $2^{63.78}$ and a memory complexity of $2^{63.78}$.

Table 15: The 26-round related-key rectangle attack on `GIFT-64`. ΔX_i and ΔY_i are the input and output differences of the Sbox layer of round i, ΔZ_i is the output difference of the linear layer of round i.

input	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
ΔY_1	??0?	1??0	01??	?0??	1?0?	?1?0	0???	?0??	??0?	???0	0???	?0??	??0?	???0	0???	?0??
ΔZ_1	????	????	????	????	0000	0000	0000	0000	11??	????	????	????	????	11??	????	????
ΔX_2	????	????	????	????	0000	0000	0000	0000	11??	????	????	????	????	11??	????	????
ΔY_2	0?01	00?0	000?	?000	0000	0000	0000	0000	0100	00?0	000?	?000	?000	0100	00?0	000?
ΔZ_2	????	0000	?1??	0000	0000	0000	0000	0000	0001	0000	0000	0000	0000	0000	0000	?1??
ΔX_3	????	0000	?1??	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	?1??
ΔY_3	1000	0000	0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0010
ΔZ_3	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0010	1010	0000	0000	0000
$\Delta X_4(\alpha)$	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	1010	0000	0000	0000
...	.	.	.	.	.	.	.	.	.	.						
$\Delta X_{24}(\delta)$	0000	0100	0000	0000	0000	0000	0000	0000	0000	0001	0010	0000	0001	0000	0000	0000
ΔY_{24}	0000	???1	0000	0000	0000	0000	0000	0000	0000	????	????	0000	????	0000	0000	0000
ΔZ_{24}	00?0	0000	00??	0?00	0001	0000	?00?	00?0	?000	0000	??00	000?	0?00	0000	0??0	?000
ΔX_{25}	00?0	0000	00??	0?00	0001	0000	?00?	00?0	?010	0000	??00	000?	0?00	0000	0??0	?000
ΔY_{25}	????	0000	????	????	????	0000	????	????	????	0000	????	????	????	0000	????	????
ΔZ_{25}	??0?	??0?	??0?	??0?	???0	???0	???0	???0	0???	0???	0???	0???	?0??	?0??	?0??	?0??
ΔX_{26}	??0?	??0?	??0?	??0?	???0	???0	???0	???0	0???	0???	0???	0???	?0??	?0??	?0??	?0??
ΔY_{26}	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
ΔZ_{26}	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????
output	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????	????

H Application in single-key setting

In Section 3, we introduce the tradeoff key-recovery model for the rectangle attack on cipher with linear key schedule. Our new related-key model and Zhao *et al.*'s model [51] can be trivially converted into the single-key model by just let the differences of the keys be 0. Choose

$$y = \sqrt{s} \cdot 2^{n/2-r_b+1.5}/\hat{p}\hat{q}, \tag{19}$$

we get about

$$\binom{y \cdot 2^{2r_b-1}}{2} \cdot 2^{-2r_b} \cdot 2^{-n}\hat{p}^2\hat{q}^2 \approx s \tag{20}$$

quartets for the right $(m_b + m_f)$-bit subkey, where s is the expected number of right quartets. Therefore, the data complexity is

$$y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{n/2+1.5}/\hat{p}\hat{q}. \tag{21}$$

- Time I (T_1): The time complexity of generating quartets is about

$$T_1 = 2^{m_b+m'_f} \cdot y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{m_b+m'_f+\frac{n}{2}+1.5}/\hat{p}\hat{q} \tag{22}$$

- Time II (T_2): We generate about

$$2^{m_b+m'_f} \cdot 2^{2(n-r_f+h_f)} \cdot \binom{y \cdot 2^{r_b-1-2(n-r_f+h_f)}}{2} = s \cdot 2^{m_b+m'_f-n+2r_f-2h_f}/\hat{p}^2\hat{q}^2$$

quartets. The time complexity of generating the key counters is

$$T_2 = (s \cdot 2^{m_b+m'_f-n+2r_f-2h_f}/\hat{p}^2\hat{q}^2) \cdot \varepsilon. \tag{23}$$

- Time III (T_3):

$$T_3 = 2^x \cdot 2^{m_b+m_f-x-h} \cdot 2^{k-(m_b+m_f)} = 2^{k-h}. \tag{24}$$

For chossing h (according to the success probability Eq. (8)), the conditions $m_b + m_f - x - h \geq 0$ and $x \leq m_b + m'_f$ have to be satisfied, which gives a guide to choose x.

Application to `Serpent` in single-key setting. We reuse the rectangle distinguisher by Biham, Dunkelman and Keller [12] and launch the rectangle attack in chosen-ciphertext setting. E_b and E_f consist of 9 rounds and 0 round, respectively. The rectangle distinguisher E' is 8-round, including a 4-round E_0 and a 4-round E_1. $r_b = m_b = 20$, $r_f = m_f = 76$, $n = 128$, $\hat{p} = 2^{-25.4}$ and $\hat{q} = 2^{-34.9}$.

According to Eq. (19), $y = \sqrt{s} \cdot 2^{n/2-r_b+1.5}/\hat{p}\hat{q} = \sqrt{s} \cdot 2^{49.8}$. Hence, the data complexity is $\sqrt{s} \cdot 2^{125.8}$ with Eq. (21). According to Eq. (22), $T_1 = \sqrt{s} \cdot 2^{m'_f} \cdot 2^{20+64+1.5+25.4+34.9} = \sqrt{s} \cdot 2^{m'_f} \cdot 2^{145.8}$. According to Eq. (23), $T_2 = s \cdot 2^{m'_f-2h_f} \cdot 2^{20-128+2\times76+120.6} \cdot \varepsilon = s \cdot 2^{m'_f-2h_f} \cdot 2^{164.6}\varepsilon$. According to Eq. (24), $T_3 = 2^{k-h}$, where $m_b + m_f - x - h \geq 0$, i.e., $h \leq 96 - x$. Obviously, $T_3 \geq 2^{160}$, and when $x = 0$ and $h = 96$, $T_3 = 2^{160}$. The memory complexity is $\sqrt{s} \cdot 2^{125.8} + 2^{96-x}$.

We set $s = 2$, $h = 92$ and $x = 0$ with Eq. (8), and the success probability is 71.4%. Then we have $T_3 = 2^{164}$. We can also balance T_1 and T_2 by adapting m'_f and h_f. For `Serpent`, once we guess a 4-bit key for one active S-box, we gain two 4-bit filters considering both sides of the rectangle. We choose $m'_f = 12$ and $2h_f = 24$. Then $T_1 = \sqrt{2} \cdot 2^{12+145.8} \approx 2^{158.3}$ and $T_2 = 2^{12-24+164.6} \cdot \varepsilon = 2^{152.6} \cdot \varepsilon$. The total complexity is bounded by $T_3 = 2^{164}$ and our attack achieve better time complexity than [11] as shown in Table 16.

Table 16: Rectangle attacks on 10-round `Serpent`.

Key size	Time	Data	Memory	Approach	Ref.
256	$2^{173.8}$	$2^{126.3}$	$2^{126.3}$	rectangle	[11]
256	2^{164}	$2^{126.3}$	$2^{126.3}$	rectangle	ours

I Overall analysis of the four attack models

In the above three models in Sect. 2 and our new model in Sect. 3.2, there are some differences:

- The `Attack I` model of Sect. 2.1 guesses all the $(m_b + m_f)$-bit key at once and generates the quartets;
- The `Attack II` model of Sect. 2.2 does not guess the key involved in E_b and E_f when generating quartets, and uses the hash tables in the key-recovery process.
- The `Attack III` model of Sect. 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 model of Sect. 3.2 guesses m_b-bit key in E_b and m'_f-bit key in E_f $(m'_f \leq m_f)$ 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 s. However, the time complexities are different.

Comparison on the time complexity. To analysis different time complexities, we first compare time complexities in the key-recovery process. Suppose, $\hat{p}\hat{q} = 2^{-t}$ and s is small and ignored, we approximate the four complexities to be

$$\begin{cases} \texttt{Attack I} : T_{\texttt{I}} = 2^{m_b+m_f+n/2+t+2}, & (25) \\ \texttt{Attack II} : T_{\texttt{II}} = 2^{m_b+r_b+2r_f-n+2t} + 2^{m_f+2r_b+r_f-n+2t}, & (26) \\ \texttt{Attack III} : T_{\texttt{III}} = 2^{m_b+2r_f-n+2t} \cdot \varepsilon, & (27) \\ \texttt{Attack IV} : T_{\texttt{IV}} = 2^{m_b+2r_f-n+m'_f-2h_f+2t} \cdot \varepsilon. & (28) \end{cases}$$

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`, usually an early abort technique [37] is applied, which makes the ε very small, for example 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`. So for ciphers in which can found h_f-bit filter with m'_f-bit guessed subkey satisfying $m'_f - 2h_f \leq 0$, `Attack IV` is better than `Attack III`.

To compare $T_{\mathtt{I}}$, $T_{\mathtt{II}}$ and $T_{\mathtt{III}}$, 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.

J The differentials of boomerang distinguishers of SKINNY

This section gives the related-tweakey boomerang distinguisher and the differentials of the boomerang distinguishers searched in Sect. 4.4. For each round of the differentials, we list the input/output differences of the S-box, as well as the subtweakey differences. In the following tables, the differences are given in hexadecimal, "*" denote arbitrary nonzero difference, "-" denote arbitrary difference.

Table 17: The 22-round related-tweakey boomerang distinguisher for `SKINNY-64-192`.

$r_0=10$, $r_m=6$, $r_1=6$, $\tilde{p}=2^{-19.84}$, $t=2^{-17.88}$, $\tilde{q}=1$, $\tilde{p}^2t\tilde{q}^2=2^{-57.56}$
$\Delta TK1$ = f, 0, 0, 0, 0, 0, 0, a, 0, 0, 0, 0, 0, 0, 0, 5
$\Delta TK2$ = d, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 9
$\Delta TK3$ = b, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, d
ΔX_0 = 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0
$\nabla TK1$ = 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
$\nabla TK2$ = 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0
$\nabla TK3$ = 0, 0, 0, 0, 0, 0, 0, 0, 0, d, 0, 0, 0, 0, 0, 0
∇X_{22} = 0, 0, 0, 0, 0, 0, 0, 0, 0, a, 0, 0, 0, 0, 0, 0

Table 18: The 18-round related-tweakey boomerang distinguisher for `SKINNY-128-256`.

$r_0=6$, $r_m=4$, $r_1=8$, $\tilde{p}=2^{-4}$, $t=2^{-35.41}$, $\tilde{q}=2^{-32.55}$, $\tilde{p}^2t\tilde{q}^2=2^{-108.51}$
$\Delta TK1$ = 00, 00, f0, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
$\Delta TK2$ = 00, 00, f8, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
ΔX_0 = 00, 00, 02, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
$\nabla TK1$ = 00, 00, 00, 00, 00, 00, 00, 00, ed, 00, 00, 00, 00, 00, 00, 00
$\nabla TK2$ = 00, 00, 00, 00, 00, 00, 00, 00, 36, 00, 00, 00, 00, 00, 00, 00
∇X_{18} = 00, 00, 37, 00, 00, 00, 37, 00, 00, 00, 00, 00, 00, 37, 00, 00

Table 19: The 19-round related-tweakey boomerang distinguisher for SKINNY-128-256.

$r_0 = 9$, $r_m = 4$, $r_1 = 6$, $\tilde{p} = 2^{-41.84}$, $t = 2^{-26.71}$, $\tilde{q} = 2^{-5.34}$, $\tilde{p}^2 t \tilde{q}^2 = 2^{-121.07}$
$\Delta TK1 =$ 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 6f, 00, 00, 00, 00, 00
$\Delta TK2 =$ 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 9b, 00, 00, 00, 00, 00
$\Delta X_0 =$ cb, 00, 00, 00, 00, 00, 00, cb, 00, 00, cb, 00, 00, 00, 00, 00
$\nabla TK1 =$ 80, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
$\nabla TK2 =$ 2a, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00
$\nabla X_{19} =$ 00, 00, 81, 00, 00, 00, 81, 00, 00, 00, 00, 00, 00, 81, 00, 00

K Figures on SKINNY automatically produced including distinguishers and key-recovery phase

Our automatic model can automatically produce the figures including the key-recovery phase and the distinguishers.

The figure on 25-round key-recovery attack on SKINNY-64-128 is shown in Figure 24.

The figure on 31-round key-recovery attack on SKINNY-64-192 is shown in Figure 25.

The figure on 25-round key-recovery attack on SKINNY-128-256 is shown in Figure 26.

The figure on 26-round key-recovery attack on SKINNY-128-256 is shown in Figure 27.

The figure on 32-round key-recovery attack on SKINNY-128-384 is shown in Figure 28.

The figures are too long, hence too small to look at. We also refer the readers to https://github.com/key-guess-rectangle/key-guess-rectangle/tree/main/ArticleTails/pic to see the large figures.

Table 20: The differentials of the 23-round distinguisher for SKINNY-128-384, where R11 to R13 denote r_m = 3-round middle part, u satisfies DDT[0x20][u] > 0 and DDT[u][0x5b] > 0, v satisfies DDT[0x5b][v] > 0 and DDT[v][0xc0] > 0, $w_{1/2}$ satisfies DDT[v][$w_{1/2}$] > 0 and DDT[$w_{1/2}$][0x04] > 0, u' satisfies DDT[0x02][u'] > 0 and DDT[0x42][u'] > 0, v' satisfies DDT[u'][v'] > 0 and DDT[v'][0x50] > 0.

	Upper differential	Lower differential
R0	0,0,0,0,0,0,0,01,0,0,0,0,0,0,0,20 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u 0,0,0,0,0,0,0,0	
R1	0,0,u,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,5b,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,5b,0,0,0,0,0	
R2-R6	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0	
R7	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,5b,0,0	
R8	0,0,0,0,0,0,0,0,0,0,5b,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,v,0,0,0,0,0 0,0,0,0,0,0,0,0	
R9	v,0,0,0,0,0,0,0,v,0,0,0,v,0,0,0 w_1,0,0,0,0,0,0,0,w_2,0,0,0,c0,0,0,0 0,0,0,c0,0,0,0,0	
R10	w_1,0,w_2,0,w_1,0,0,c0,0,0,w_2,0,w_1,0,w_2,c0 04,0,04,0,04,0,0,04,0,0,04,0,04,0,04,04 0,0,0,0,0,0,0,0	
R11	0,04,0,04,04,0,04,0,0,04,0,0,0,0,04,0 0,*,0,*,*,0,*,0,0,*,0,0,0,0,*,0 0,0,0,0,0,0,0,61	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0
R12	middle part	middle part
R13	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,68,0,0,0,0,0,0	0,*,0,*,*,0,0,*,0,0,0,*,*,0,0,0 0,02,0,42,02,0,0,02,0,0,0,02,42,0,0,0 0,0,0,0,0,0,0,0
R14		0,0,0,0,0,02,0,42,02,0,0,0,0,0,0,42 0,0,0,0,0,u',0,58,u',0,0,0,0,0,0,u' 0,0,0,0,0,0,0,58
R15		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u',0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,v',0 0,0,0,0,0,0,0,0
R16		0,v',0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,50,0,0,0,0,0,0
R17-R21		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0
R22		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,50,0,0,0

Table 21: The differentials of the 22-round distinguisher for SKINNY-64-192, where R10 to R15 denote r_m = 6-round middle part, u satisfies DDT[0x1][u] > 0 and DDT[$u \oplus$ 0x9][0xb] > 0, v satisfies DDT[0x1][v] > 0 and DDT[v][0xb] > 0.

	Upper differential	Lower differential
R0	1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0 9,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0 9,0,0,0,0,0,0,8	
R1-R4	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0	
R5	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,1,0,0,0,0,0	
R6	0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0 0,0,8,0,0,0,u,0,0,0,0,0,0,0,v,0 0,0,8,0,0,0,9,0	
R7	0,v,0,0,0,0,0,0,0,0,0,$u \oplus$ 0x9,0,0,0,0 0,b,0,0,0,0,0,0,0,0,0,b,0,0,0,0 0,0,0,0,b,0,0,0	
R8	0,0,0,0,0,b,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,e,0,0,0,0,0,0,0,0,0,0 0,0,0,0,5,e,0,0	
R9	0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,0 0,0,0,0,0,0,2,0	
R10	0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,2 0,0,0,*,0,0,0,0,0,0,0,0,0,0,0,* 0,0,0,3,0,0,1,0	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0
R11-R14	middle part	middle part
R15	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,6	0,*,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,a,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,a,0,0,0,0,0,0
R16-R20		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0
R21		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,a,0,0,0

Table 22: The differentials of the 18-round distinguisher for SKINNY-128-256, where R6 to R9 denote r_m = 4-round middle part, u satisfies DDT$[u][$0x9b$] > 0$, v satisfies DDT$[v][u] > 0$, w_1 satisfies DDT$[w_1][v] > 0$ and DDT$[$0x30/80/a0$][w_1] > 0$, w_2 satisfies DDT$[w_2][v] > 0$ and DDT$[$0x30$][w_2] > 0$, w_3 satisfies DDT$[w_3][v] > 0$, DDT$[w_3][$0x56$] > 0$ and DDT$[$0x10/30$][w_3] > 0$.

	Upper differential	Lower differential
R0	0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,8,0,0,0,0,0	
R1-R3	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0	
R4	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,10,0	
R5	0,0,0,0,0,0,0,0,0,0,10,0,0,0,0 0,0,0,0,0,0,0,0,0,0,40,0,0,0,0 0,0,0,0,0,0,0,0	
R6	0,40,0,0,0,0,0,0,0,40,0,0,0,40,0,0 0,*,0,0,0,0,0,0,0,*,0,0,0,*,0,0 0,0,0,0,0,30,0,0	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0
R7-R8	middle part	middle part
R9	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0	*,0,0,0,0,0,*,0,0,*,0,0,*,*,*,0 30,0,0,0,0,0,90,0,0,30,0,0,10,30,80,0 0,0,0,30,0,0,0,0
R10		0,80,0,10,30,0,0,30,0,0,0,a0,30,0,0,0 0,w_1,0,w_3,w_1,0,0,w_2,0,0,0,w_1,w_3,0,0,0 0,0,0,0,0,0,0,0
R11		0,0,0,0,0,w_1,0,w_3,w_2,0,0,0,0,0,0,w_3 0,0,0,0,0,v,0,56,v,0,0,0,0,0,0,v 0,0,0,0,0,0,0,56
R12		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,v,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,u,0 0,0,0,0,0,0,0,0
R13		0,u,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,9b,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,9b,0,0,0,0,0,0
R14-R16		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0
R17		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,37,0,0,0,0,0

Table 23: The differentials of the 19-round distinguisher for SKINNY-128-256, where R9 to R12 denote r_m = 4-round middle part, u satisfies DDT[0xcb][u] > 0 and DDT[u][0x58] > 0, v satisfies DDT[0xb0][v] > 0 and DDT[v][0xd0] > 0, $w_{1/2}$ satisfies DDT[v][$w_{1/2}$] > 0 and DDT[$w_{1/2}$][0x04] > 0, v' satisfies DDT[0x20][v'] > 0 and DDT[v'][0x80] > 0.

	Upper differential	Lower differential
R0	cb,0,0,0,0,0,0,cb,0,0,cb,0,0,0,0,0 u,0,0,0,0,0,0,u,0,0,u,0,0,0,0,0 0,0,0,0,0,0,0,0	
R1	0,0,0,0,u,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,58,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,58,0,0,0	
R2-R4	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0	
R5	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,b0,0,0	
R6	0,0,0,0,0,0,0,0,0,0,b0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,v,0,0,0,0,0 0,0,0,0,0,0,0,0	
R7	v,0,0,0,0,0,0,0,v,0,0,0,v,0,0,0 w_1,0,0,0,0,0,0,0,w_2,0,0,0,d0,0,0,0 0,0,0,d0,0,0,0,0	
R8	w_1,0,w_2,0,w_1,0,0,d0,0,0,w_2,0,w_1,0,w_2,d0 04,0,04,0,04,0,0,04,0,0,04,0,04,0,04,04 0,0,0,0,0,0,0,0	
R9	0,04,0,04,04,0,04,0,0,04,0,0,0,0,04,0 0,*,0,*,*,0,*,0,0,*,0,0,0,0,*,0 0,0,0,0,0,0,0,61	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0
R10-R11	middle part	middle part
R12	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0	0,0,0,0,0,*,0,*,*,0,0,0,0,0,0,* 0,0,0,0,0,20,0,20,20,0,0,0,0,0,0,20 0,0,0,0,0,0,0,20
R13		0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,v',0 0,0,0,0,0,0,0,0
R14		0,v',0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,80,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,80,0,0,0,0,0,0
R15-R17		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0
R18		0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,81,0,0,0,0,0

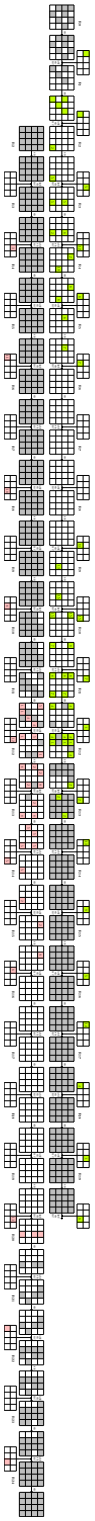

Fig. 24: 25-round key-recovery attack on SKINNY-64-128

Fig. 25: 31-round key-recovery attack on SKINNY-64-192

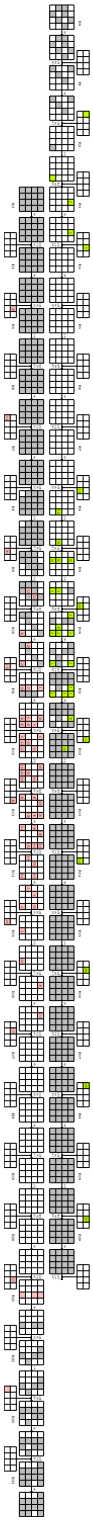

Fig. 26: 25-round key-recovery attack on SKINNY-128-256

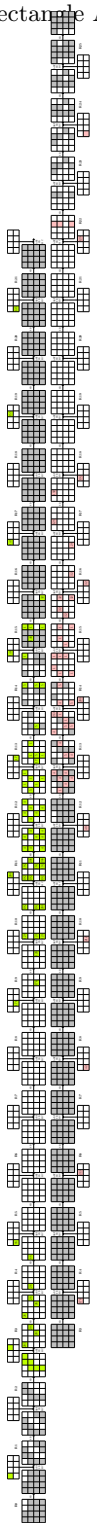

Fig. 27: 26-round key-recovery attack on SKINNY-128-256

Fig. 28: 32-round key-recovery attack on SKINNY-128-384