Mind the TWEAKEY Schedule: Cryptanalysis on SKINNYe-64-256

Abstract. Designing symmetric ciphers for particular applications becomes a hot topic. At EUROCRYPT 2020, Naito, Sasaki and Sugawara invented the threshold implementation friendly cipher SKINNYe-64-256 to meet the requirement of the authenticated encryption PFB_Plus. Soon, Thomas Peyrin pointed out that SKINNYe-64-256 may lose the security expectation due the new tweakey schedule. Although the security issue of SKINNYe-64-256 is still unclear, Naito *et al.* decided to introduce SKINNYe-64-256 v2 as a response.

In this paper, we give a formal cryptanalysis on the new tweakey schedule of SKINNYe-64-256 and discover unexpected differential cancellations in the tweakey schedule. For example, we find the number of cancellations can be up to 8 within 30 consecutive rounds, which is significantly larger than the expected 3 cancellations. Moreover, we take our new discoveries into rectangle, MITM and impossible differential attacks, and adapt the corresponding automatic tools with new constraints from our discoveries. Finally, we find a 41-round related-tweakey rectangle attack on SKINNYe-64-256 and leave a security margin of 3 rounds only.

As STK accepts arbitrary tweakey size, but SKINNY and SKINNYe-64-256 v2 only support up to $4n$ tweakey size. We introduce a new design of tweakey schedule for SKINNY-64 to further extend the supported tweakey size. We give a formal proof that our new tweakey schedule inherits the security requirement of STK and SKINNY.

Keywords: SKINNY· TWEAKEY · Rectangle · Meet-in-the-middle · Impossible differential

1 Introduction

The design of symmetric cryptographic constructions for important security goals and practical applications becomes more and more popular. Typical algorithms including LowMC [3], MiMC [2], etc., provide efficient implementation for multi-party secure computing (MPC), fully homomorphic encryption (FHE), and zero-knowledge proofs (ZK). Another important topic is to design symmetric ciphers that can be efficiently implemented against side-channel attacks [30,14,48], especially because NIST lightweight cryptography competition optionally takes into account the security of the cryptographic modules against side-channel attack (SCA). Masking is by far the most common countermeasure against SCA [41,52]. Threshold implementation (TI) introduced by Nikova *et al.* [52] is a masking particularly popular for hardware implementation. Several TI-friendly Sboxes [15,38] are proposed. At TCHES 2020, Naito and Sugawara [51] discovered that for recently ciphers such as SKINNY [10] and GIFT [7], the complexity of TI for the linear key schedule function is significantly smaller than the non-linear round function. With this asymmetry, Naito and Sugawara [51] proposed a TBC-based scheme `PFB` which is particularly efficient with TI. To further exploit this asymmetry, at EUROCRYPT 2020, Naito, Sasaki and Sugawara [49] invented tweakable block cipher (TBC) based AE modes `PFB_Plus`, `PFB`w, as well as a new TBC, i.e. `SKINNYe-64-256`, which are very efficient in threshold implementations.

At ASIACRYPT 2014, Jean, Nikolić and Peyrin introduced the `TWEAKEY` framework [43] with the goal to unify the design of tweakable block ciphers and allow to build a primitive with arbitrary tweak and key sizes. It treats the key input and the tweak input in the same way as the tweakey. Towards simplifying the security analysis when the tweakey size is large, Jean *et al.* identified a subclass of `TWEAKEY`, named as `STK` construction, which updates the round tweakey by the use of finite field multiplications on low hamming weight constants. `SKINNY` [10] is a well-known lightweight block cipher family proposed by Beierle *et al.* at CRYPTO 2016, which follows closely the `STK` construction [43]. However, instead of using multiplications by non-zero constants in a finite field adopted by `STK` construction, `SKINNY` updates the tweakey cells by the cheap 4-bit or 8-bit LFSRs (depending on the size of the cell) to minimize the hardware cost, while maintaining the cancellation behavior required by the `STK` construction: for a given position, $z-1$ cancellations can only happen every 15 rounds for `TK`-z[1].

As a concrete `STK`-like design, `SKINNY` only supports `TK`-1/-2/-3, while for `STK` construction, the size of tweakey can be of arbitrary length. However, in practical applications, tweakable block ciphers with large tweakeys may be required, such as the TI-friendly AE modes `PFB_Plus` and `PFB`w proposed by Naito, Sasaki and Sugawara [49]. Without `TK`-4 available for `SKINNY`, Naito *et al.* decided to build the `SKINNYe-64-256` to support $zn = 4n$ tweakey with $n = 64$. In order to inherit the numerous cryptanalytic efforts on `SKINNY-64` [39,33,47,5,32,53,26], `SKINNYe-64-256` does not modify any components to realize TK_1, TK_2, and TK_3, and only find a new LFSR for updating TK_4. With the expectation of keeping a similar security margin with 36-round `SKINNY-64-128` and 40-round `SKINNY-64-192`, the authors decided to keep the same rate for increasing the number of rounds, namely 44 rounds for `SKINNYe-64-256`. However, Thomas Peyrin found that the security claim of `SKINNYe-64-256` may not hold due to the tweakey schedule. Although the authors of `SKINNYe-64-256` were unclear whether this issue causes some attacks against the whole cipher [50, Section 7], they proposed an updated version of `SKINNYe-64-256`, named as `SKINNYe-64-256` v2 in Eprint 2020/542 [50].

Our Contributions. In this paper, we try to clarify the security issue of `SKINNYe-64-256` [49] by delving into its new tweakey schedule. There are some previous works considered the relations of keys, such as the key-bridging technique [35,28]. The relations of subtweakeys for `SKINNY` and `SKINNYe-64-256` are mostly dependent on the LFSR_m updating the cells of the tweakey states. For LFSR_2 used for TK_2 and LFSR_4 used for TK_4 of `SKINNYe-64-256`, both of

[1]For `TK`-z, if the size of internal state is n, the size of tweakey will be zn.

them shift the 4-bit input to the left by 1 bit, while LFSR_2 updates 1 output bit with 1 XOR and LFSR_4 updates 2 output bits with 3 XORs. Suppose for a given cell of TK_2 and TK_4 with the initial value `0x8`, then apply LFSR_2 and LFSR_4 respectively to the given cell for 14 times and we get two sequences, i.e.,

$$[\underline{\mathtt{0x8}}, \underline{\mathtt{0x1}}, \underline{\mathtt{0x2}}, \mathtt{0x4}, \underline{\mathtt{0x9}}, \underline{\mathtt{0x3}}, \mathtt{0x6}, \mathtt{0xd}, \underline{\mathtt{0xa}}, \mathtt{0x5}, \underline{\mathtt{0xb}}, \mathtt{0x7}, \mathtt{0xf}, \mathtt{0xe}, \mathtt{0xc}],$$
$$[\underline{\mathtt{0x8}}, \underline{\mathtt{0x1}}, \underline{\mathtt{0x2}}, \mathtt{0x5}, \underline{\mathtt{0x9}}, \underline{\mathtt{0x3}}, \mathtt{0x7}, \mathtt{0xc}, \underline{\mathtt{0xa}}, \mathtt{0x4}, \underline{\mathtt{0xb}}, \mathtt{0x6}, \mathtt{0xe}, \mathtt{0xf}, \mathtt{0xd}].$$

For example, run LFSR_2 or LFSR_4 on `0x8` for 3 times, we get $\mathrm{LFSR}_2^3(\mathtt{0x8})=\mathtt{0x4}$ and $\mathrm{LFSR}_4^3(\mathtt{0x8})=\mathtt{0x5}$, respectively. Intuitively, the longest common subsequence of the two sequences is [`0x8,0x1,0x2,0x9,0x3,0xa,0xb`] which is highlighted with underlines. In other words, when the initial values (or differences) for a given cell position of TK_2 and TK_4 are `0x8` and TK_1 and TK_3 are set to `0x0`, the difference cancellations can happen 7 times within 15 LFSR applications.

In order to further clarify the cancellation property of the new tweakey schedule, we give a formal analysis of relations of subtweakeys. Since the tweakey schedule of SKINNYe-64-256 is linear, each cell of subtweakeys can be derived via multiplying some cells of the master tweakeys by certain binary matrix $\boldsymbol{A}$, which is determined by cell updating functions, i.e., LFSRs. The differential cancellation behavior means active input leads to zero output by multiplying $\boldsymbol{A}$. We analyze the properties of matrix $\boldsymbol{A}$, especially for the influence of its rank on the cancellations in the differential-like distinguishers, as well as the subtweakey guessing strategy in the key-recovery phase. For the differential cancellation behavior, we find the number of cancellations can be up to 8 within 30 consecutive rounds for SKINNYe-64-256 (a cell is updated by LFSR in every two rounds in SKINNY), which is significantly larger than the expected 3 cancellations. By exploring the properties of $\boldsymbol{A}$ in rectangle attack, meet-in-the-middle (MITM) attack and impossible differential attack, we discover unexpected distinguishers or key-recovery attacks:

- **Related-tweakey rectangle attacks.** The properties can not only extend the rectangle distinguisher significantly, but also improve the key-recovery phase. At EUROCRYPT 2022, Dong *et al.* [32] introduced the attacks on the 25-round SKINNY-64-128 with an 18-round distinguisher as well as the 31-round SKINNY-64-192 with a 22-round distinguisher. With our discoveries on SKINNYe-64-256, we find a 30-round rectangle distinguisher, where the gap between SKINNY-64-192 and SKINNYe-64-256 is significantly increased to 30-22=8 rounds comparing to 22-18=4 rounds between SKINNY-64-128 and SKINNY-64-192. Moreover, in the key-recovery phase, we explore the key relations in detail with the help of matrix $\boldsymbol{A}$, and finally perform a 41-round key-recovery attack on SKINNYe-64-256.
  In order to find the optimal configurations of the rectangle attack, we tweak Dong *et al.*'s automatic model by applying the properties of the new tweakey schedule into the model. Our attack leaves only a 3-round security margin for SKINNYe-64-256, which is significantly reduced comparing to the 11-round and 9-round security margins for SKINNY-64-128 and SKINNY-64-192.

- **MITM attacks in single-tweakey setting.** Not only the differential cancellation property can be used to improve attacks, but also the non-full rank property of $\boldsymbol{A}$. The MITM attack explores two independent chunks that overlap in a match point. Suppose $\boldsymbol{A}$ is of non-full rank, we compute the solution space of $\boldsymbol{Ax} = \boldsymbol{c}$ for given vector $\boldsymbol{c}$. In `SKINNYe-64-256`, $\boldsymbol{x}$ is the master tweakey bits and $\boldsymbol{c}$ is the subtweakey bits that will XORed into the internal state. Denote solution set as $\{\boldsymbol{x} : \boldsymbol{Ax} = \boldsymbol{c}\}$, if it is not empty, then its size will be $|\{\boldsymbol{x} : \boldsymbol{Ax} = \boldsymbol{c}\}| > 1$ due to non-full rank property of $\boldsymbol{A}$. In the MITM, those $\boldsymbol{x} \in \{\boldsymbol{x} : \boldsymbol{Ax} = \boldsymbol{c}\}$ will have the same effect on the internal states, i.e., the vector $\boldsymbol{c}$. When building independent forward and backward chunks in MITM, we may prefix $\boldsymbol{c}$ and $\boldsymbol{c}'$ for these two chunks, then the values in $\{\boldsymbol{x} : \boldsymbol{Ax} = \boldsymbol{c}\}$ and $\{\boldsymbol{y} : \boldsymbol{A}'\boldsymbol{y} = \boldsymbol{c}'\}$ will have independent effects.
  We adapt the previous automatic tools [8,31] for MITM attacks by taking the non-full rank properties of $\boldsymbol{A}$ into the model. Finally, we find 31-round MITM attack on `SKINNYe-64-256`, while previous MITM attacks on `SKINNY-64-128` and `SKINNY-64-192` reach 18 and 23 rounds, respectively. In other words, the gaps of the attacked rounds increase from 23-18=5 rounds between `SKINNY-64-128` and `SKINNY-64-192` to currently 31-23=8 rounds between `SKINNY-64-192` and `SKINNYe-64-256`.
- **Related-tweakey impossible differential attack.** With the differential cancellation properties, we find a 21-round impossible differential for `SKINNYe-64-256` based on a cancellation pattern, while previous impossible differential reaches 16 rounds [47] for `SKINNY-64-192` and 15 rounds [55] for `SKINNY-64-128`, respectively.

Our cryptanalysis proves that `SKINNYe-64-256` does not keep a similar security margin to `SKINNY-64-128` and `SKINNY-64-192` as expected by the designers. The non-trivial properties of the new tweakey schedule can be used to improve the attacks from the distinguishers to key-recovery.

In addition, we also analyze the updated version, i.e., `SKINNYe-64-256` v2 [50], and obtain a 37-round related-tweakey rectangle attack, a 27-round MITM attack, as well as an 18-round impossible differential. Comparing to the attacks on `SKINNY-64-128` and `SKINNY-64-192`, the attacked rounds on `SKINNYe-64-256` v2 keep the same rate as expected by the designers. We summarize results on `SKINNY-64` and `SKINNYe-64-256` and its version 2 in Table 1 and Table 2.

Table 1: Rectangle attacks on `SKINNY-64` and `SKINNYe-64-256` and its version 2

Version	Rounds	Data	Time	Memory	Distinguisher	Setting	Ref.
`SKINNY-64-128`	23/36	$2^{60.54}$	$2^{120.7}$	$2^{60.9}$	19	RK	[39]
	24/36	$2^{61.67}$	$2^{96.83}$	2^{84}	18	RK	[53]
	25/36	$2^{61.67}$	$2^{118.43}$	$2^{64.26}$	18	RK	[32]
`SKINNY-64-192`	29/40	$2^{62.92}$	$2^{181.7}$	2^{80}	23	RK	[39]
	30/40	$2^{62.87}$	$2^{163.11}$	$2^{68.05}$	22	RK	[53]
	31/40	$2^{62.78}$	$2^{182.07}$	$2^{62.79}$	22	RK	[32]
`SKINNYe-64-256`	41/44	$2^{62.24}$	$2^{237.06}$	$2^{62.26}$	30	RK	Sect. 4.3
`SKINNYe-64-256` v2	37/44	$2^{62.8}$	$2^{240.03}$	$2^{62.8}$	26	RK	Sect. C.2

Table 2: MITM attacks on SKINNY-64 and SKINNYe-64-256 and its version 2

Version	Rounds	Data	Time	Memory	Approach	Setting	Ref.
SKINNY-64-128	18/36	2^{16}	2^{124}	2^4	MITM	SK	[40]
SKINNY-64-192	23/40	2^{52}	2^{188}	2^4	MITM	SK	[31]
SKINNYe-64-256	31/44	2^{52}	2^{254}	2^{52}	MITM	SK	Sect. 5.2
SKINNYe-64-256 v2	27/44	2^{52}	2^{252}	2^{52}	MITM	SK	Sect. D.3

Note that STK construction supports arbitrary length of tweakey, but SKINNY and SKINNYe-64-256 v2 supports upto $4n$-bit tweakey. As stated in [49, Page 5]: "... there is no consensus about the adequate tweak size to support". SKINNY with larger tweakey size may be useful in future applications, such as the TI-friendly AE modes PFB_Plus and PFBw with SKINNYe-64-256 v2. Therefore, as another contribution, we propose a uniformed design strategy for tweakey schedule of SKINNY-n-zn for positive integer $z \leq 14$. Our uniformed tweakey schedule satisfies the security requirements of the STK construction with a formal proof. Interestingly, our schedule will be reduced to SKINNY-64 when $z = 1, 2, 3$, and to SKINNYe-64-256 v2 when $z = 4$.

2 Preliminaries

2.1 The TWEAKEY Framework

At ASIACRYPT 2014, Jean *et al.* [43] proposed a generic framework for tweakable block ciphers, named as the TWEAKEY framework. They consider the tweak and key inputs in a unified manner, i.e., tweakey, that can be used to design a tweakable block cipher with any key and any tweak sizes. The TWEAKEY framework uses the tweakey scheduling algorithm. The ciphertext is computed from the plaintext by applying the permutation f iteratively. Each round is composed of three parts, a sub-tweakey extraction function g from the tweakey state, an internal update permutation f and a tweakey state update function h. Based on the TWEAKEY framework, many designs of tweakable block ciphers are proposed, including Deoxys [44], SKINNY [10], and CRAFT [13], etc. Moreover, Jean *et al.* identified a subclass of tweakey for AES-like ciphers named as Superposition TWEAKEY (STK) construction shown in Figure 1. In the STK construction, the n-bit internal state and zn-bit tweakey state (denoted as TK-z) are partitioned into n/c and zn/c c-bit cells respectively. The functions g and h become:

- the function g simply XORs all the z n-bit words of the tweakey state to the internal state (AddRoundTweakey, denoted ART).
- the function h first applies the same cell position permutation function P to each of the z n-bit words of the tweakey state, and then multiply each c-bit cell of the j-th n-bit word by a nonzero coefficient α_j in the finite field $GF(2^c)$ (with $\alpha_i \neq \alpha_j$ for all $1 \leq i \neq j \leq z$).

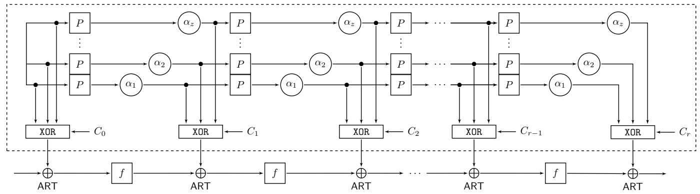

Fig. 1: The `STK` [43]. (Thanks to https://www.iacr.org/authors/tikz/)

2.2 `SKINNY` family and `SKINNYe-64-256`

`SKINNY` is a family of lightweight block cipher proposed by Beierle *et al.* at CRYPTO 2016 [10]. Following the `TWEAKEY` framework and `STK` construction [43], the round function of `SKINNY` that replaces the f function of `STK` in Figure 1 is given in Figure 2. There are six main versions `SKINNY`-n-zn: $n = 64, 128$, $z = 1, 2, 3$. The internal state is viewed as a 4×4 square arrays of cells. The tweakey state is viewed as z 4×4 square arrays of cells, denoted as (TK_1) when $z = 1$, (TK_1, TK_2) when $z = 2$, and (TK_1, TK_2, TK_3) when $z = 3$. Denote the i-th cell of TK_m as $TK_{m,i}$ $(1 \le m \le z,\ 0 \le i \le 15)$. An important difference between the `STK` construction [43] and `SKINNY` is that in the tweakey schedule the cells of the tweakey are updated by LFSRs for `SKINNY` instead of multiplying α_j. As shown in Figure 2, the round function applies 5 transformations: `SubCells (SC)`, `AddConstants (AC)`, `AddRoundTweakey (ART)`, `ShiftRows (SR)` and `MixColumns (MC)`. For the details, please refer to [10].

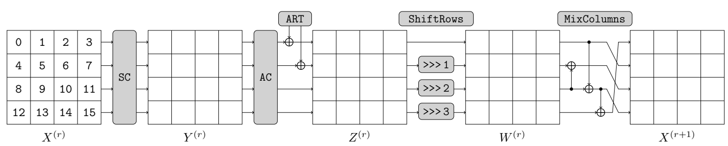

Fig. 2: Round function of `SKINNY`

For the block size $n = 64$, `SKINNY` supports the tweakey sizes up to 192 bits. At EUROCRYPT 2020, to support the TI-friendly AE modes `PFB_Plus` and `PFB`w, Naito, Sasaki, and Sugawara [49] extended the design of `SKINNY-64` to support a 256-bit tweakey and derived `SKINNYe-64-256`, which applies the same round function of `SKINNY` but a new tweakey schedule. However, Thomas Peyrin found that the security claim of `SKINNYe-64-256` may not hold due to the new tweakey schedule. In response, Naito *et al.* decided to propose an updated version of `SKINNYe-64-256`, i.e., `SKINNYe-64-256` v2 in Eprint 2020/542 [50].

New Tweakey Schedule. The 256-bit tweakey state is viewed as 4 4×4 square arrays of nibbles as (TK_1, TK_2, TK_3, TK_4). Denote the tweakey arrays as $TK_1^{(r)}$, $TK_2^{(r)}$, $TK_3^{(r)}$ and $TK_4^{(r)}$ in round r $(r \ge 0)$, where $TK_m^{(0)} = TK_m$ $(1 \le m \le 4)$. For $r \ge 1$, $TK_m^{(r)}$ is generated in two steps.

First, apply the permutation $P = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ on each nibble of all tweakey arrays:

$$TK_{m,i}^{(r)} \leftarrow TK_{m,P[i]}^{(r-1)},\ 1 \le m \le 4,\ 0 \le i \le 15, r \ge 1. \tag{1}$$

Then, apply LFSR_m to update each nibble of the first and second rows of $TK_m^{(r)}$ with $2 \le m \le 4$. The LFSR for $TK_4^{(r)}$ used in SKINNYe-64-256 and SKINNYe-64-256 v2 is different. The LFSRs are given in Table 3.

Table 3: The LFSRs used in SKINNYe-64-256 and SKINNYe-64-256 v2

TK	LFSRs
TK_2	$(x_3\|x_2\|x_1\|x_0) \to (x_2\|x_1\|x_0\|x_3 \oplus x_2)$
TK_3	$(x_3\|x_2\|x_1\|x_0) \to (x_0 \oplus x_3\|x_3\|x_2\|x_1)$
TK_4	$(x_3\|x_2\|x_1\|x_0) \to (x_2\|x_1\|x_2 \oplus x_0\|x_3 \oplus x_2 \oplus x_1)$
TK_4 v2	$(x_3\|x_2\|x_1\|x_0) \to (x_1\|x_0\|x_3 \oplus x_2\|x_2 \oplus x_1)$

In the ART operation, only the first two rows of subtweakey $STK^{(r)}$ are xored to the internal state, where

$$STK_i^{(r)} = TK_{1,i}^{(r)} \oplus TK_{2,i}^{(r)} \oplus TK_{3,i}^{(r)} \oplus TK_{4,i}^{(r)}, 0 \le i \le 7,\ r \ge 0. \tag{2}$$

Lemma 1. *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.*

3 Properties of the Tweakey Schedule of SKINNYe-64-256

In round $r \ge 0$, each of the 64-bit tweakey $TK_m^{(r)}$ ($1 \le m \le 4$) of SKINNYe-64-256 can be represented as a 4×16 binary matrix $\boldsymbol{TK}_m^{(r)}$ $(1 \le m \le 4,\ r \ge 0)$ as

$$\boldsymbol{TK}_m^{(r)} = \begin{pmatrix} x_{m,0}^{(r)}\ x_{m,4}^{(r)}, \dots, x_{m,60}^{(r)} \\ x_{m,1}^{(r)}\ x_{m,5}^{(r)}, \dots, x_{m,61}^{(r)} \\ x_{m,2}^{(r)}\ x_{m,6}^{(r)}, \dots, x_{m,62}^{(r)} \\ x_{m,3}^{(r)}\ x_{m,7}^{(r)}, \dots, x_{m,63}^{(r)} \end{pmatrix},$$

with $x_{m,j}^{(r)} \in \{0,1\}$ $(0 \le j \le 63)$. Denote $\boldsymbol{TK}_m^{(r)}[*, i]$ as the i-th column of the binary matrix $\boldsymbol{TK}_m^{(r)}$. Then $\boldsymbol{TK}_m^{(r)}[*, i]$ is actually the i-th nibble of $TK_m^{(r)}$, *i.e.*, $TK_{m,i}^{(r)}$ $(0 \le i \le 15)$, which is denoted as a binary vector $\boldsymbol{tk}_{m,i}^{(r)} \in \mathbb{F}_2^4$,

$$\boldsymbol{tk}_{m,i}^{(r)} = [x_{m,4i}^{(r)}, x_{m,4i+1}^{(r)}, x_{m,4i+2}^{(r)}, x_{m,4i+3}^{(r)}]^T,\ 0 \le i \le 15,\ 1 \le m \le 4, r \ge 0.$$

Since $TK_m^{(0)} = TK_m$, we also write $\boldsymbol{tk}_{m,i}^{(0)} = [x_{m,4i}, x_{m,4i+1}, x_{m,4i+2}, x_{m,4i+3}]^T$ for simplicity. We can deduce the relations between the subtweakeys transformed from the same nibble of the master tweakey. For TK_1, only the permutation P is applied in each round. Assume P^r means to apply the permutation P for r times. We have $\boldsymbol{tk}_{1,i}^{(r)} = \boldsymbol{tk}_{1,P^r[i]}^{(0)},\ 0 \le i \le 15$.

For TK_2, TK_3 and TK_4, after applying the permutation, a LFSR is applied to update each cell of the 1st and 2nd rows in each round, which is equivalent to multiplying the cell by a 4×4 binary matrix. For SKINNYe-64-256 and its version 2, the LFSRs used for TK_2 and TK_3 are the same, whose corresponding matrices are denoted as $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$. The LFSRs used in TK_4 for `SKINNYe-64-256` and version 2 are different, which are denoted as $\boldsymbol{L}_4$ and $\tilde{\boldsymbol{L}}_4$. We have

$$\boldsymbol{L}_2 = \begin{pmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,1\,0\,0 \end{pmatrix}, \quad \boldsymbol{L}_3 = \begin{pmatrix} 1\,0\,0\,1 \\ 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \end{pmatrix}, \quad \boldsymbol{L}_4 = \begin{pmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix}, \quad \tilde{\boldsymbol{L}}_4 = \begin{pmatrix} 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,1\,0\,0 \\ 0\,1\,1\,0 \end{pmatrix}.$$

Since only the first two rows of subtweakey are XORed to the internal state, the tweakey cells involved in the r-th round encryption will be involved again in the $(r+2)$-th round according to $P = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$. For simplicity, we first consider the formulas of subtweakeys for `SKINNYe-64-256`, and for version 2, the formulas are different only for TK_4. Assume $\boldsymbol{L}_m^i$ represents the i-th power of matrix $\boldsymbol{L}_m$ in $GF(2)$ and $\boldsymbol{L}_m^0 = \boldsymbol{I}$ $(2 \le m \le 4)$. Note that the LFSRs for TK_2 and TK_3 in `SKINNY` and the new LFSR for TK_4 in `SKINNYe-64-256` have the same cycle of 15, which lead to $\boldsymbol{L}_m^{15} = \boldsymbol{I}$ $(2 \le m \le 4)$. For `SKINNYe-64-256` v2, although the update function for TK_4 is not a LFSR, it also has a cycle of 15, i.e., $\tilde{\boldsymbol{L}}_4^{15} = \boldsymbol{I}$. In the tweakey schedule, for each nibble of $TK_m^{(r)}$, the LFSR is applied in every two rounds, we deduce: $\forall m \in \{2, 3, 4\}$,

$$\begin{cases} \boldsymbol{tk}_{m,i}^{(r)} = \boldsymbol{L}_m^{\lceil r/2 \rceil} \cdot \boldsymbol{tk}_{m,P^r[i]}^{(0)}, \ 0 \le i \le 7, \\ \boldsymbol{tk}_{m,i}^{(r)} = \boldsymbol{L}_m^{\lfloor r/2 \rfloor} \cdot \boldsymbol{tk}_{m,P^r[i]}^{(0)}, \ 8 \le i \le 15. \end{cases}$$

Denote the nibble $STK_i^{(r)}$ $(0 \le i \le 7)$ as a binary vector $\boldsymbol{stk}_i^{(r)} = (y_{4i}^{(r)}, y_{4i+1}^{(r)}, y_{4i+2}^{(r)}, y_{4i+3}^{(r)})^T$. Then we obtain $\boldsymbol{stk}_i^{(r)} = \bigoplus_{m=1}^{4} \boldsymbol{tk}_{m,i}^{(r)}$ for $0 \le i \le 7$. Considering subtweakey cells $\boldsymbol{stk}_i^{(r)}$ derived from master tweakey, we get

$$\boldsymbol{stk}_i^{(r)} = [\boldsymbol{I}\ \boldsymbol{L_2}^{\lceil r/2 \rceil}\ \boldsymbol{L_3}^{\lceil r/2 \rceil}\ \boldsymbol{L_4}^{\lceil r/2 \rceil}] \cdot \left(\boldsymbol{tk}_{1,P^r[i]}^{(0)}, \boldsymbol{tk}_{2,P^r[i]}^{(0)}, \boldsymbol{tk}_{3,P^r[i]}^{(0)}, \boldsymbol{tk}_{4,P^r[i]}^{(0)}\right)^T. \quad (3)$$

Without losing generality, we analyze the subtweakeys in the even rounds, which are all transformed from the first two rows of master tweakeys. Let $\bar{P} = [8, 9, 10, 11, 12, 13, 14, 15, 2, 0, 4, 7, 6, 3, 5, 1]$ be the inverse permutation of P. For a set $\texttt{Index} = \{r_1, \cdots, r_t\}$ $(|\texttt{Index}| = t)$, which corresponding to a set of subtweakeys $\{STK^{(2r_1)}, STK^{(2r_2)} \cdots, STK^{(2r_t)}\}$, we can get a set of linear equations as

$$\begin{pmatrix} \boldsymbol{stk}_{\bar{P}^{2r_1}[i]}^{(2r_1)} \\ \boldsymbol{stk}_{\bar{P}^{2r_2}[i]}^{(2r_2)} \\ \vdots \\ \boldsymbol{stk}_{\bar{P}^{2r_t}[i]}^{(2r_t)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{L}_2^{r_1} & \boldsymbol{L}_3^{r_1} & \boldsymbol{L}_4^{r_1} \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_2} & \boldsymbol{L}_3^{r_2} & \boldsymbol{L}_4^{r_2} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_t} & \boldsymbol{L}_3^{r_t} & \boldsymbol{L}_4^{r_t} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{tk}_{1,i}^{(0)} \\ \boldsymbol{tk}_{2,i}^{(0)} \\ \boldsymbol{tk}_{3,i}^{(0)} \\ \boldsymbol{tk}_{4,i}^{(0)} \end{pmatrix}, \ 0 \le i \le 7. \quad (4)$$

Because the tweakey schedule only contains the permutation and LFSRs, Equation (4) is linear equation. Denote coefficient matrix as $\boldsymbol{A}$ and its rank as $rank(\boldsymbol{A}) = a$. The image space of $\boldsymbol{A}$ represents the solution space of $\{STK_{\bar{P}^{2r_1}[i]}^{(2r_1)}, STK_{\bar{P}^{2r_2}[i]}^{(2r_2)} \cdots, STK_{\bar{P}^{2r_t}[i]}^{(2r_t)}\}$ with arbitrary $\{\boldsymbol{tk}_{1,i}^{(0)}, \boldsymbol{tk}_{2,i}^{(0)}, \boldsymbol{tk}_{3,i}^{(0)}, \boldsymbol{tk}_{4,i}^{(0)}\}$, whose size is $|Im(\boldsymbol{A})| = 2^a$. Let the kernel space of $\boldsymbol{A}$ be $Ker(\boldsymbol{A}) = \{\boldsymbol{x} \in \mathbb{F}_2^{4t} : \boldsymbol{A}\boldsymbol{x} = 0\}$, then the size of the kernel space is $|Ker(\boldsymbol{A})| = 2^{16-a}$. For example, assuming `Index` $= \{0, 1, 2, 3\}$, we can obtain the equations of $\{STK^{(0)}, STK^{(2)}, STK^{(4)}, STK^{(6)}\}$ as Equation (4). For $i = 0$, there is

$$\begin{pmatrix} \boldsymbol{stk}_0^{(0)} \\ \boldsymbol{stk}_2^{(2)} \\ \boldsymbol{stk}_4^{(4)} \\ \boldsymbol{stk}_6^{(6)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{L}_2^0 & \boldsymbol{L}_3^0 & \boldsymbol{L}_4^0 \\ \boldsymbol{I} & \boldsymbol{L}_2^1 & \boldsymbol{L}_3^1 & \boldsymbol{L}_4^1 \\ \boldsymbol{I} & \boldsymbol{L}_2^2 & \boldsymbol{L}_3^2 & \boldsymbol{L}_4^2 \\ \boldsymbol{I} & \boldsymbol{L}_2^3 & \boldsymbol{L}_3^3 & \boldsymbol{L}_4^3 \end{pmatrix} \begin{pmatrix} \boldsymbol{tk}_{1,0}^{(0)} \\ \boldsymbol{tk}_{2,0}^{(0)} \\ \boldsymbol{tk}_{3,0}^{(0)} \\ \boldsymbol{tk}_{4,0}^{(0)} \end{pmatrix} = \begin{pmatrix}
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0 \\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0 \\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0 \\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \\
1&0&0&0&0&1&0&0&1&0&0&1&0&1&0&0 \\
0&1&0&0&0&0&1&0&1&0&0&0&0&0&1&0 \\
0&0&1&0&0&0&0&1&0&1&0&0&0&1&0&1 \\
0&0&0&1&1&1&0&0&0&0&1&0&1&1&1&0 \\
1&0&0&0&0&0&1&0&1&0&1&1&0&0&1&0 \\
0&1&0&0&0&0&0&1&1&0&0&1&0&1&0&1 \\
0&0&1&0&1&1&0&0&1&0&0&0&1&1&0&0 \\
0&0&0&1&0&1&1&0&0&1&0&0&0&0&1&1 \\
1&0&0&0&0&0&0&1&1&1&1&1&0&1&0&1 \\
0&1&0&0&1&1&0&0&1&0&1&1&1&1&0&0 \\
0&0&1&0&0&1&1&0&1&0&0&1&0&1&1&0 \\
0&0&0&1&0&0&1&1&1&0&0&0&1&0&1&1
\end{pmatrix} \begin{pmatrix} x_{1,0} \\ x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{2,0} \\ x_{2,1} \\ x_{2,2} \\ x_{2,3} \\ x_{3,0} \\ x_{3,1} \\ x_{3,2} \\ x_{3,3} \\ x_{4,0} \\ x_{4,1} \\ x_{4,2} \\ x_{4,3} \end{pmatrix}. \quad (5)$$

The rank of the coefficient matrix $\boldsymbol{A}$ in Equation (5) is 14. Therefore, the size of its kernel space and image space is $|Ker(\boldsymbol{A})| = 2^2$ and $|Im(\boldsymbol{A})| = 2^{14}$.

Let $\boldsymbol{A}_{r_j} = [\boldsymbol{I}\ \boldsymbol{L}_2^{r_j}\ \boldsymbol{L}_3^{r_j}\ \boldsymbol{L}_4^{r_j}]$, which is a 4×16 matrix. Then the coefficient matrix of Equation (4) can be represented as $\boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}} = [\boldsymbol{A}_{r_1}^T\ \boldsymbol{A}_{r_2}^T\ \cdots\ \boldsymbol{A}_{r_t}^T]^T$, which is a $4t \times 16$ matrix. Since $\boldsymbol{L}_i^{15} = \boldsymbol{I}$ for $2 \le i \le 4$, we can assume that all subscripts of $\boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}}$ are mod 15. We call $\boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}}$ a full rank matrix if and only if $rank(\boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}}) = \min\{4t, 16\}$. We find that when $t \ge 4$, certain sets of `Index` lead to non-full rank coefficient matrices. Let $\mathcal{K} = \{0, 1, 2, \cdots, 14\}$, for any subset $\{r_1, r_2, \cdots, r_t\} \subset \mathcal{K}$ and $0 \le r' \le 14$, we have

$$\begin{aligned}
&\boldsymbol{A}_{\{r_1+r', r_2+r', \cdots, r_t+r'\}} \\
&= \begin{pmatrix} \boldsymbol{I} & \boldsymbol{L}_2^{r_1+r'} & \boldsymbol{L}_3^{r_1+r'} & \boldsymbol{L}_4^{r_1+r'} \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_2+r'} & \boldsymbol{L}_3^{r_2+r'} & \boldsymbol{L}_4^{r_2+r'} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_t+r'} & \boldsymbol{L}_3^{r_t+r'} & \boldsymbol{L}_4^{r_t+r'} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{L}_2^{r_1} & \boldsymbol{L}_3^{r_1} & \boldsymbol{L}_4^{r_1} \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_2} & \boldsymbol{L}_3^{r_2} & \boldsymbol{L}_4^{r_2} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_t} & \boldsymbol{L}_3^{r_t} & \boldsymbol{L}_4^{r_t} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{I} & & & \\ & \boldsymbol{L}_2^{r'} & & \\ & & \boldsymbol{L}_3^{r'} & \\ & & & \boldsymbol{L}_4^{r'} \end{pmatrix} \quad (6) \\
&= \boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}} \cdot diag(\boldsymbol{I}, \boldsymbol{L}_2^{r'}, \boldsymbol{L}_3^{r'}\ \boldsymbol{L}_4^{r'}).
\end{aligned}$$

Since $\boldsymbol{L}_2$, $\boldsymbol{L}_3$ and $\boldsymbol{L}_4$ are all 4×4 full rank matrices, $D_{r'} = diag(\boldsymbol{I}, \boldsymbol{L}_2^{r'}, \boldsymbol{L}_3^{r'}\ \boldsymbol{L}_4^{r'})$ is a 16×16 full rank matrix. Then we can deduce that

$$rank(\boldsymbol{A}_{\{r_1+r', r_2+r', \cdots, r_t+r'\}}) = rank(\boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}}). \quad (7)$$

Since the rank of the coefficient matrix is our most concern, we introduce the concept of *rank-equivalent* as follows.

Definition 1 (rank-equivalent). *Given two subsets $x = \{r_1, r_2, \ldots, r_t\}$, $y = \{r_1', r_2', \ldots, r_t'\} \subset \mathcal{K}$, we say x and y are rank-equivalent if there exits an integer r' such that*

$$r_i \equiv r_i' + r' \bmod 15 \textit{ for all } 1 \le i \le t.$$

The rank-equivalence class of the subset x is defined by

$$[x] := \{y \subset \mathcal{K} : x \text{ and } y \text{ are rank-equivalent}\}.$$

From Eq. (7), $rank(\boldsymbol{A}_x) = rank(\boldsymbol{A}_y)$ holds for any rank-equivalent subsets x and y.

Table 4: Rank-equivalence class of non-full rank coefficient matrix for `SKINNYe-64-256`

rank	t	Rank-equivalence class $[\{r_1, r_2, \cdots, r_t\}]$
14	4	[{0,1,2,3}],[{0,1,2,10}],[{0,1,3,4}],[{0,1,3,7}],[{0,1,3,13}],[{0,1,4,5}],[{0,1,4,12}], [{0,1,5,6}],[{0,1,5,8}],[{0,1,5,11}],[{0,1,6,7}],[{0,1,6,10}],[{0,1,6,12}],[{0,1,7,8}], [{0,1,7,9}],[{0,1,11,13}],[{0,2,4,6}],[{0,2,5,7}],[{0,2,5,12}],[{0,2,6,8}],[{0,2,6,11}], [{0,2,7,9}],[{0,2,7,10}],[{0,2,7,11}],[{0,2,9,12}],[{0,3,6,9}],[{0,3,7,10}],[{0,3,7,11}]
15	4	[{0,1,2,4}],[{0,1,2,5}],[{0,1,2,6}],[{0,1,2,7}],[{0,1,2,8}],[{0,1,2,9}],[{0,1,2,11}], [{0,1,2,12}],[{0,1,2,13}],[{0,1,3,5}],[{0,1,3,6}],[{0,1,3,8}],[{0,1,3,9}],[{0,1,3,10}], [{0,1,3,11}],[{0,1,3,12}],[{0,1,4,6}],[{0,1,4,7}],[{0,1,4,8}],[{0,1,4,9}],[{0,1,4,10}], [{0,1,4,11}],[{0,1,4,13}],[{0,1,5,7}],[{0,1,5,9}],[{0,1,5,10}],[{0,1,5,12}],[{0,1,5,13}], [{0,1,6,8}],[{0,1,6,9}],[{0,1,6,11}],[{0,1,6,13}],[{0,1,7,10}],[{0,1,7,11}],[{0,1,7,12}], [{0,1,7,13}],[{0,1,8,10}],[{0,1,8,11}],[{0,1,8,12}],[{0,1,8,13}],[{0,1,9,11}],[{0,1,9,12}], [{0,1,9,13}],[{0,1,10,12}],[{0,1,10,13}],[{0,2,4,7}],[{0,2,4,8}],[{0,2,4,9}],[{0,2,4,10}], [{0,2,4,11}],[{0,2,4,12}],[{0,2,5,8}],[{0,2,5,9}],[{0,2,5,10}],[{0,2,5,11}],[{0,2,6,9}], [{0,2,6,10}],[{0,2,6,12}],[{0,2,7,12}],[{0,2,8,11}],[{0,2,8,12}],[{0,3,6,10}],[{0,3,6,11}]
	5	[{0,1,2,3,7}],[{0,1,2,3,10}],[{0,1,2,3,11}],[{0,1,2,3,13}],[{0,1,2,4,5}],[{0,1,2,4,8}], [{0,1,2,4,10}],[{0,1,2,5,8}],[{0,1,2,5,10}],[{0,1,2,6,9}],[{0,1,2,6,10}],[{0,1,2,6,12}], [{0,1,2,7,10}],[{0,1,2,7,11}],[{0,1,2,7,13}],[{0,1,2,8,10}],[{0,1,2,9,10}],[{0,1,2,9,12}], [{0,1,2,10,11}],[{0,1,2,10,12}],[{0,1,2,10,13}],[{0,1,2,11,13}],[{0,1,3,4,7}],[{0,1,3,4,9}], [{0,1,3,5,6}],[{0,1,3,5,7}],[{0,1,3,5,8}],[{0,1,3,5,12}],[{0,1,3,6,7}],[{0,1,3,6,8}], [{0,1,3,6,12}],[{0,1,3,7,8}],[{0,1,3,7,9}],[{0,1,3,7,10}],[{0,1,3,7,11}],[{0,1,3,7,12}], [{0,1,3,7,13}],[{0,1,3,8,12}],[{0,1,3,10,11}],[{0,1,3,10,13}],[{0,1,3,11,13}],[{0,1,4,5,8}], [{0,1,4,5,10}],[{0,1,4,6,11}],[{0,1,4,6,12}],[{0,1,4,6,13}],[{0,1,4,7,9}],[{0,1,4,8,10}], [{0,1,4,11,13}],[{0,1,5,6,12}],[{0,1,5,7,8}],[{0,1,5,7,12}],[{0,1,5,8,9}],[{0,1,5,8,10}], [{0,1,5,8,11}],[{0,1,5,8,12}],[{0,1,5,8,13}],[{0,1,5,9,11}],[{0,1,5,9,13}],[{0,1,5,11,13}], [{0,1,6,7,12}],[{0,1,6,8,12}],[{0,1,6,9,12}],[{0,1,6,10,12}],[{0,1,6,11,13}],[{0,1,7,10,13}], [{0,1,7,11,13}],[{0,1,8,11,13}],[{0,1,9,11,13}],[{0,2,4,6,11}],[{0,2,4,7,11}],[{0,2,4,8,10}], [{0,2,4,9,12}],[{0,2,5,7,11}],[{0,2,5,8,10}],[{0,2,5,9,12}],[{0,2,6,9,12}]
	6	[{0,1,2,3,7,10}],[{0,1,2,3,7,11}],[{0,1,2,3,7,13}],[{0,1,2,3,10,11}],[{0,1,2,3,10,13}], [{0,1,2,3,11,13}],[{0,1,2,4,5,8}],[{0,1,2,4,5,10}],[{0,1,2,4,8,10}],[{0,1,2,5,8,10}], [{0,1,2,6,9,10}],[{0,1,2,6,9,12}],[{0,1,2,6,10,12}],[{0,1,2,7,10,11}],[{0,1,2,7,10,13}], [{0,1,2,7,11,13}],[{0,1,2,9,10,12}],[{0,1,2,10,11,13}],[{0,1,3,4,7,9}],[{0,1,3,5,6,8}], [{0,1,3,5,6,12}],[{0,1,3,5,7,8}],[{0,1,3,5,7,12}],[{0,1,3,5,8,12}],[{0,1,3,6,7,12}], [{0,1,3,6,8,12}],[{0,1,3,7,8,12}],[{0,1,3,7,10,11}],[{0,1,3,7,10,13}],[{0,1,3,7,11,13}], [{0,1,4,5,8,10}],[{0,1,4,6,11,13}],[{0,1,5,7,8,12}],[{0,1,5,8,11,13}],[{0,1,5,9,11,13}]
	7	[{0,1,2,3,7,10,11}],[{0,1,2,3,7,10,13}],[{0,1,2,3,7,11,13}],[{0,1,2,3,10,11,13}], [{0,1,2,4,5,8,10}],[{0,1,2,6,9,10,12}],[{0,1,2,7,10,11,13}],[{0,1,3,5,6,8,12}], [{0,1,3,5,7,8,12}]
	8	[{0,1,2,3,7,10,11,13}]

For `SKINNYe-64-256`, we compute all the rank-equivalence classes whose corresponding coefficient matrix is non-full rank with Algorithm 2 in Supplementary Material A and list the results in Table 4.

Similarly, for `SKINNYe-64-256` v2, we set $\tilde{\boldsymbol{A}}_{r_j} = [\boldsymbol{I}\ \boldsymbol{L}_{\boldsymbol{2}}^{r_j}\ \boldsymbol{L}_{\boldsymbol{3}}^{r_j}\ \tilde{\boldsymbol{L}}_{\boldsymbol{4}}^{r_j}]$, which is also a 4×16 matrix. Then the coefficient matrix of Equation (4) can be represented as $\tilde{\boldsymbol{A}}_{\{r_1,r_2,\cdots,r_t\}} = [\tilde{\boldsymbol{A}}_{r_1}^T\ \tilde{\boldsymbol{A}}_{r_2}^T\ \cdots\ \tilde{\boldsymbol{A}}_{r_t}^T]^T$, which is a $4t \times 16$ matrix. For arbitrary $\{r_1, r_2, \cdots, r_t\} \subset \mathcal{K}$, the matrix $\tilde{\boldsymbol{A}}_{\{r_1,r_2,\cdots,r_t\}}$ is full rank. That is, when $t \leq 4$, the rank of $\tilde{\boldsymbol{A}}_{\{r_1,r_2,\cdots,r_t\}}$ is $4t$, otherwise the rank is 16.

The Subtweakey Difference Cancellations. For a given active tweakey cell, $z-1$ subtweakey difference cancellation happens every 30 rounds for SKINNY-n-zn [10] with $z = 2, 3$. However, for SKINNYe-64-256, although $z = 4$, we have more cancellations than $z - 1 = 3$. Since the tweakey schedule is linear, the differences of subtweakeys can be computed by the differences injected in the master tweakey with Equation (4). Assume that there is at least one $1 \leq m \leq 4$ that $\Delta TK_{m,i} \neq 0$. Set $\boldsymbol{A}_{[\{r_1,r_2,\cdots,r_t\}]} \cdot [\boldsymbol{tk}^{(0)}_{1,i}, \boldsymbol{tk}^{(0)}_{2,i}, \boldsymbol{tk}^{(0)}_{3,i}, \boldsymbol{tk}^{(0)}_{4,i}]^T = \boldsymbol{0}$, which means the subtweakey difference cancellations happen at $\{STK^{(2r_1)}_{\bar{P}^{2r_1}[i]} \cdots, STK^{(2r_t)}_{\bar{P}^{2r_t}[i]}\}$ if $0 \leq i \leq 7$, or $\{STK^{(2r_1+1)}_{\bar{P}^{2r_1+1}[i]} \cdots, STK^{(2r_t+1)}_{\bar{P}^{2r_t+1}[i]}\}$ if $8 \leq i \leq 15$. When $rank(\boldsymbol{A}_{[\{r_1,r_2,\cdots,r_t\}]}) = 16$, the size of its kernel space is 1. Then $[\boldsymbol{tk}^{(0)}_{1,i}, \boldsymbol{tk}^{(0)}_{2,i}, \boldsymbol{tk}^{(0)}_{3,i}, \boldsymbol{tk}^{(0)}_{4,i}]$ has only one zero solution, which means $\Delta TK_{m,i} = 0$ for all $m = 1, 2, 3, 4$. When $rank(\boldsymbol{A}_{[\{r_1,r_2,\cdots,r_t\}]}) < 16$, we have non-zero solutions for $\Delta TK_{m,i}$, i.e., the subtweakey difference cancellations happen. Obviously, when $t \leq 3$, $rank(\boldsymbol{A}_{[\{r_1,r_2,\cdots,r_t\}]}) = 4t \leq 16$. For $t \geq 4$, we obtain all rank-equivalence classes whose corresponding coefficient matrices are non-full rank from Table 4. So each rank-equivalence class corresponds to a set of positions of the subtweakey difference cancellations. We find several properties of the rank-equivalence classes:

- When $t = 4$, we find the matrix $A_{\{r_1,r_2,r_3,r_4\}}$ with arbitrary $\{r_1, r_2, r_3, r_4\} \subset \mathcal{K}$ is non-full rank. That is, for the given active nibbles in the master key, the subtweakey difference cancellations can happen four times in arbitrary round for every 30 rounds. Especially for $rank(\boldsymbol{A}_{\{0,1,2,3\}}) = 14$ and $|Ker(\boldsymbol{A}_{\{0,1,2,3\}})| = 2^2$, there are 3 non-zero solutions of the difference for the active nibbles of the master tweakey. For SKINNYe-64-256, there can be nine consecutive rounds with fully inactive internal states.
- When $t \geq 5$, for all $[\{r_1, r_2, \cdots, r_t\}]$ in Table 4, $rank(\boldsymbol{A}_{[\{r_1,r_2,\cdots,r_t\}]}) = 15$. For $\boldsymbol{A}_{[\{r_1,r_2,\cdots,r_t\}]} \cdot [\boldsymbol{tk}^{(0)}_{1,i}, \boldsymbol{tk}^{(0)}_{2,i}, \boldsymbol{tk}^{(0)}_{3,i}, \boldsymbol{tk}^{(0)}_{4,i}]^T = \boldsymbol{0}$, there is only one nonzero solution. We find that for some different rank-equivalence classes, the solutions are the same. For example, for rank-equivalence classes $[\{0, 1, 2, 7, 10\}]$ and $[\{0, 1, 3, 11, 13\}]$, when $0 \leq i \leq 7$ we set

$$A_{[\{0,1,2,7,10\}]} \cdot [\boldsymbol{tk}^{(0)}_{1,i}, \boldsymbol{tk}^{(0)}_{2,i}, \boldsymbol{tk}^{(0)}_{3,i}, \boldsymbol{tk}^{(0)}_{4,i}]^T = \boldsymbol{0}, \tag{8}$$

$$A_{[\{0,1,3,11,13\}]} \cdot [\boldsymbol{tk}^{(0)}_{1,i}, \boldsymbol{tk}^{(0)}_{2,i}, \boldsymbol{tk}^{(0)}_{3,i}, \boldsymbol{tk}^{(0)}_{4,i}]^T = \boldsymbol{0}, \tag{9}$$

  where the cancellations happen at $\{STK^{(0)}_i, STK^{(2)}_{\bar{P}^2[i]}, STK^{(4)}_{\bar{P}^4[i]}, STK^{(14)}_{\bar{P}^{14}[i]}, STK^{(20)}_{\bar{P}^{20}[i]}\}$ for Equation (8) and $\{STK^{(0)}_i, STK^{(2)}_{\bar{P}^2[i]}, STK^{(6)}_{\bar{P}^6[i]}, STK^{(22)}_{\bar{P}^{22}[i]}, STK^{(26)}_{\bar{P}^{26}[i]}\}$ for Equation (9). The non-zero solutions of both two linear equations are $\boldsymbol{tk}^{(0)}_{1,i} = [0, 0, 0, 1]^T$, $\boldsymbol{tk}^{(0)}_{2,i} = [0, 1, 1, 1]^T$, $\boldsymbol{tk}^{(0)}_{3,i} = [0, 0, 0, 0]^T$, $\boldsymbol{tk}^{(0)}_{4,i} = [0, 1, 1, 0]^T$. Namely, the cancellations happen at $\{STK^{(0)}_i, STK^{(2)}_{\bar{P}^2[i]}, STK^{(4)}_{\bar{P}^4[i]}, STK^{(6)}_{\bar{P}^6[i]}, STK^{(14)}_{\bar{P}^{14}[i]}, STK^{(20)}_{\bar{P}^{20}[i]}, STK^{(22)}_{\bar{P}^{22}[i]}, STK^{(26)}_{\bar{P}^{26}[i]}\}$ at the same time, which corresponds to the rank-equivalence class $[\{0, 1, 2, 3, 7, 10, 11, 13\}]$.

The situation for $8 \leq i \leq 15$ is the same. Further, we find that for arbitrary $\{r_1, r_2, \cdots, r_t\} \subset \{0,1,2,3,7,10,11,13\}$ $(t \geq 5)$, the solution of $\boldsymbol{A}_{[\{r_1,r_2,\cdots,r_t\}]} \cdot [\boldsymbol{tk}_{1,i}^{(0)}, \boldsymbol{tk}_{2,i}^{(0)}, \boldsymbol{tk}_{3,i}^{(0)}, \boldsymbol{tk}_{4,i}^{(0)}]^T = \mathbf{0}$ is the same to $A_{[\{0,1,2,3,7,10,11,13\}]} \cdot [\boldsymbol{tk}_{1,i}^{(0)}, \boldsymbol{tk}_{2,i}^{(0)}, \boldsymbol{tk}_{3,i}^{(0)}, \boldsymbol{tk}_{4,i}^{(0)}]^T = \mathbf{0}$, which means that there is only one difference cancellation behaviour for those rank-equivalence classes.

Remark. It is worth noting that there are some rank-equivalence classes $[\{r_1, r_2, \cdots, r_t\}]$ in Table 4, where $\{r_1, r_2, \cdots, r_t\}$ is not directly the subset of $\{0,1,2,3,7,10,11,13\}$ but corresponds to the same difference cancellation behaviour. Taking the rank-equivalence class $[\{0,1,2,6,9\}]$ as an example, we can assume $A_{[\{0,1,2,6,9\}]} \cdot [\bar{\boldsymbol{tk}}_{1,i}^{(0)}, \bar{\boldsymbol{tk}}_{2,i}^{(0)}, \bar{\boldsymbol{tk}}_{3,i}^{(0)}, \bar{\boldsymbol{tk}}_{4,i}^{(0)}]^T = \mathbf{0}$, and obtain $\bar{\boldsymbol{tk}}_{1,i}^{(0)} = [0,0,0,1]^T$, $\bar{\boldsymbol{tk}}_{2,i}^{(0)} = [1,1,1,1]^T$, $\bar{\boldsymbol{tk}}_{3,i}^{(0)} = [0,0,0,0]^T$, $\bar{\boldsymbol{tk}}_{4,i}^{(0)} = [1,1,1,0]^T$. Applying the same solution, we can also deduce $A_{[\{0,1,2,6,9,10,12,14\}]} \cdot [\bar{\boldsymbol{tk}}_{1,i}^{(0)}, \bar{\boldsymbol{tk}}_{2,i}^{(0)}, \bar{\boldsymbol{tk}}_{3,i}^{(0)}, \bar{\boldsymbol{tk}}_{4,i}^{(0)}]^T = \mathbf{0}$. Similarly, for arbitrary $\{r_1, r_2, \cdots, r_t\} \subset \{0,1,2,6,9,10,12,14\}$ $(t \geq 5)$, we deduce that there is only one difference cancellation behaviour. Further, due to rank-equivalence class in Definition 1, there is $[\{0,1,2,3,7,10,11,13\}] = [\{0,1,2,6,9,10,12,14\}]$. The two sets $\{0,1,2,3,7,10,11,13\}$ and $\{0,1,2,6,9,10,12,14\}$ only represent the difference cancellations starting from different rounds every 15 rounds for `TK`-z, and actually show the same difference cancellation behaviour.

In summary, there are only two kinds of the difference cancellation behaviours:

- For rank-equivalence class $[\{0,1,2,4,5,8,10\}]$, the subtweakey difference cancellations happen 7 times in the fixed positions for the given active nibble of the master key in every 30 rounds. Assuming $A_{[\{0,1,2,4,5,8,10\}]} \cdot [\boldsymbol{tk}_{1,i}^{(0)}, \boldsymbol{tk}_{2,i}^{(0)}, \boldsymbol{tk}_{3,i}^{(0)}, \boldsymbol{tk}_{4,i}^{(0)}]^T = \mathbf{0}$, we can compute the only one nonzero solution, where $\boldsymbol{tk}_{1,i}^{(0)} = [0,0,0,0]^T$, $\boldsymbol{tk}_{2,i}^{(0)} = [1,0,0,0]^T$, $\boldsymbol{tk}_{3,i}^{(0)} = [0,0,0,0]^T$, $\boldsymbol{tk}_{4,i}^{(0)} = [1,0,0,0]^T$.
- For rank-equivalence class $[\{0,1,2,3,7,10,11,13\}]$, the subweakey difference cancellations happen 8 times in the fixed positions every 30 rounds. Assuming $A_{[\{0,1,2,3,7,10,11,13\}]} \cdot [\boldsymbol{tk}_{1,i}^{(0)}, \boldsymbol{tk}_{2,i}^{(0)}, \boldsymbol{tk}_{3,i}^{(0)}, \boldsymbol{tk}_{4,i}^{(0)}]^T = \mathbf{0}$, the nonzero solution is $\boldsymbol{tk}_{1,i}^{(0)} = [0,0,0,1]^T$, $\boldsymbol{tk}_{2,i}^{(0)} = [0,1,1,1]^T$, $\boldsymbol{tk}_{3,i}^{(0)} = [0,0,0,0]^T$, $\boldsymbol{tk}_{4,i}^{(0)} = [0,1,1,0]^T$.

For `SKINNYe-64-256 v2`, there is $rank(\tilde{\boldsymbol{A}}_{\{r_1,r_2,r_3,r_4\}}) = 16$ for arbitrary $\{r_1, r_2, r_3, r_4\} \subset \mathcal{K}$. That is, at most three difference cancellations can happen every 30 rounds for a given active tweakey nibble and there can be seven rounds of fully inactive internal states at most.

Key Guessing Strategy Based on the Relations of Subtweakeys. In key-recovery attacks, several rounds are added before and after the distinguisher and the involved subtweakeys should be guessed to recover the master tweakey. We can use the relations of subtweakeys to get more accurate and efficient key guessing strategy following similar idea of the key-bridge technique [35,28]. For example, assume that a set of subtweakeys $\{\boldsymbol{stk}^{(2r_1)}_{\bar{P}^{2r_1}[i]}, \boldsymbol{stk}^{(2r_2)}_{\bar{P}^{2r_2}[i]} \cdots, \boldsymbol{stk}^{(2r_t)}_{\bar{P}^{2r_t}[i]}, \boldsymbol{stk}^{(2r_{t+1})}_{\bar{P}^{2r_{t+1}}[i]}\}$ derived from the same i-th ($0 \leq i \leq 7$) nibble of the master tweakey are involved in the key-recovery phase. Suppose $rank(\boldsymbol{A}_{\{r_1,r_2,\cdots,r_t\}}) = a$ and $rank(\boldsymbol{A}_{\{r_1,r_2,\cdots,r_{t+1}\}}) = b$ ($b > a$). The number of possible values for $\{\boldsymbol{stk}^{(2r_1)}_{\bar{P}^{2r_1}[i]}, \boldsymbol{stk}^{(2r_2)}_{\bar{P}^{2r_2}[i]} \cdots, \boldsymbol{stk}^{(2r_t)}_{\bar{P}^{2r_t}[i]}\}$ is $|Im(\boldsymbol{A}_{\{r_1,r_2,\cdots,r_t\}})| = 2^a$. After we guessed $\{\boldsymbol{stk}^{(2r_1)}_{\bar{P}^{2r_1}[i]}, \boldsymbol{stk}^{(2r_2)}_{\bar{P}^{2r_2}[i]} \cdots, \boldsymbol{stk}^{(2r_t)}_{\bar{P}^{2r_t}[i]}\} \in Im(\boldsymbol{A}_{\{r_1,r_2,\cdots,r_t\}})$, the number of possible guesses for the last nibble $\boldsymbol{stk}^{(2r_{t+1})}_{\bar{P}^{2r_{t+1}}[i]}$ will be 2^{b-a}.

4 Rectangle Attacks on SKINNYe-64-256 and Its Version 2

4.1 Preliminary for Boomerang and Rectangle Attacks

The boomerang attack proposed by Wagner [62] is a differential-based attack, which uses two short differential characteristics instead of one long characteristic as shown in Figure 3. The boomerang attack is developed into the amplified boomerang attack [45] and rectangle attack [19], which require only chosen plaintext queries. To clarify the probability of boomerang, Biryukov *et al.* [21] introduced the *boomerang switch* technique, which is generalized by Dunkelman *et al.* [36] as the *sandwich attack*. In the attack, the cipher E_d is considered as $\tilde{E}_1 \circ E_m \circ \tilde{E}_0$, where $\tilde{p}$ and $\tilde{q}$ are the probability of the differentials used for the r_0-round $\tilde{E}_0$ and r_1-round $\tilde{E}_1$. The middle part r_m-round E_m handles the dependence of the two short differentials. If the probability of generating a right quartet for E_m is ξ, the probability of the whole rectangle distinguisher is $2^{-n}\tilde{p}^2\tilde{q}^2\xi$. Then, Cid *et al.* [25] introduced the boomerang connectivity table (BCT) to clarify the probability around the boundary of boomerang and compute its probability more accurately. Further, various studies or improvements [23,60,63,26] on BCT technique enrich boomerang attacks.

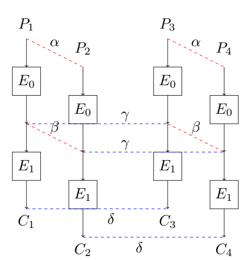

Fig. 3: Boomerang attack

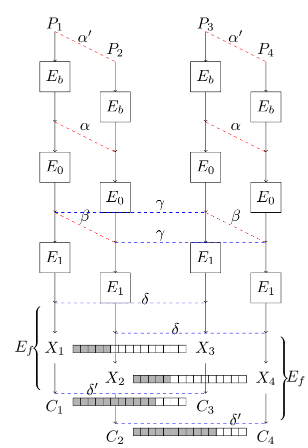

Fig. 4: Rectangle attack on E

.

Related-key boomerang and rectangle attacks were proposed by Biham *et al.* [20]. As shown in Figure 4, the cipher E is decomposed into $E_f \circ E_d \circ E_b$, where $E_d = E_1 \circ E_0$ is the related-key rectangle distinguisher and E_b and E_f are the extended rounds before and after the distinguisher. Assuming we use a related-key differential $\alpha \to \beta$ over E_0 under a key difference ΔK and $\delta \to \gamma$ over E_1 under a key difference ∇K. 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 $K_4 = K_1 \oplus \Delta K \oplus \nabla K$. Denote r_b as the number of unknown bits in the difference α' of plaintexts. Let k_b be the set of subkey bits that involved in E_b while encrypting the plaintext to the known difference α and decrypting to get the corresponding plaintext. Denote the number $m_b = |k_b|$. Similarly, we have r_f and $m_f = |k_f|$ for E_f.

There are several key-recovery frameworks of rectangle attacks [19,18,17,47] in both single-key setting and related-key setting. As shown by Biham *et al.* [17], when the key schedule is linear (e.g. `SKINNY`), the differences between the subkeys of K_1, K_2, K_3 and K_4 are all determined in each round. Exploring this property, Dong *et al.* [32] proposed a new related-key rectangle attack for ciphers with linear key schedule (see Supplementary Material B.1). They try to guess all k_b and part of k_f, denoted as k'_f before generating quartets. Then with partial decryption, they may gain h_f inactive bits (or bits with fixed differences) from the internal state as filters. They also built a uniform automatic tool to search for the entire rectangle key-recovery attack on `SKINNY`, which is based on a series of automatic tools [39,26,53].

4.2 Automatic Search for Related-Tweakey Rectangle Attacks for `SKINNYe-64-256` and its Version 2

We apply Dong *et al.*'s automatic tool [32] by modifying the constraints of the subtweakey to include more differential cancellation behaviours studied in Section 3. For simplicity, we put Dong *et al.*'s automatic tool in Supplementary Material B.2, and only list the differences of the modelling here.

In previous automatic models [10,26,39] for `SKINNY`-n-zn ($z = 1, 2, 3$), for a given cell position in the tweakey schedule, the number of cancellations can only be $z - 1$ within 30 consecutive rounds. The constraints for the cancellations are given by the designers of `SKINNY` [12, Page 52], e.g., for 0-th nibble of the master tweakey within the 30 consecutive rounds:

$$\begin{aligned} \mathrm{LANE}_0 - stk_0^{(0)} \geq 0,\ \mathrm{LANE}_0 - stk_{P^{2i}[0]}^{(2i)} \geq 0,\ 1 \leq i \leq 14, \\ stk_0^{(0)} + stk_{P^2[0]}^{(2)} + \cdots + stk_{P^{28}[0]}^{(28)} - 15 \cdot \mathrm{LANE}_0 \geq -(z-1), \end{aligned} \tag{10}$$

where the binary variable LANE_0 is 0 only if $TK_{m,0} = 0$ for all $1 \leq m \leq z$, and the binary variable $stk_{P^r[0]}^{(r)}$ is 0 if and only if the nibble $STK_{P^r[0]}^{(r)}$ is inactive. Similar constraints are applied to other nibble positions.

However, for `SKINNYe-64-256`, although $z = 4$, we have more cancellations than $z - 1 = 3$ according to Section 3. The possible number and positions of cancellations are diverse, which needs to be modeled by new constraints for the upper and lower differentials besides Constraint (10). According to Section 3, the automatic models are divided into two cases according to different subtweakey difference cancellation behaviours to search for the distinguisher suitable for Dong *et al.*'s rectangle attack framework:

- $t \le 4$: When $t \le 3$, the rank of $\boldsymbol{A}_{\{r_1,\cdots,r_t\}}$ is $4t \le 16$. When $t = 4$, the matrix $\boldsymbol{A}_{\{r_1,r_2,r_3,r_4\}}$ is non-full rank. That is, when $t \le 4$, $rank(\boldsymbol{A}_{\{r_1,\cdots,r_t\}}) < 16$. For a given active nibble in the master key, the subtweakey difference cancellations can happen at most four times in arbitrary 30 rounds. In this case, we only need to modify the last constraint of Eq. (10) to be ($z = 4$):

$$stk_0^{(0)} + stk_{P^2[0]}^{(2)} + \cdots + stk_{P^{28}[0]}^{(28)} - 15 \cdot \mathrm{LANE}_0 \ge -z.$$

- $t > 4$: There are only two kinds of the difference cancellation behaviours in Section 3, i.e., $[\{0,1,2,4,5,8,10\}]$ and $[\{0,1,2,3,7,10,11,13\}]$. For the rank-equivalence class $[\{0,1,2,4,5,8,10\}]$, we fixed the positions of difference cancellations for the i-th active nibble of the master tweakey to build the model. For each $0 \le r' \le 14$, we set the subtweakey differences to 0 in $\{2r', 2(r'+1) \bmod 30, 2(r'+2) \bmod 30, 2(r'+4) \bmod 30, 2(r'+5) \bmod 30, 2(r'+8) \bmod 30, 2(r'+10) \bmod 30\}$ rounds when $0 \le i \le 7$, and in $\{2r'+1, (2(r'+1)+1) \bmod 30, (2(r'+2)+1) \bmod 30, (2(r'+4)+1) \bmod 30, (2(r'+5)+1) \bmod 30, (2(r'+8)+1) \bmod 30, (2(r'+10)+1) \bmod 30\}$ rounds when $8 \le i \le 15$ to run the model. Similar for case $[\{0,1,2,3,7,10,11,13\}]$.

Searching with different automatic models, we select a 30-round related-tweakey (RTK) boomerang distinguisher for SKINNYe-64-256 in Table 5, where the difference cancellation behaviour $[\{0,1,2,3,7,10,11,13\}]$ is used both in the upper and lower differentials. We also experimentally verify the probabilities of the middle part of the distinguishers, and list details of the distinguisher, the experimental results and full figures in Table 14, Table 16 and Figure 14 in Supplementary Material C.1 and H. Our source codes are based on the open source in [26,32], which is provided in https://github.com/skinny64/Skinny64-256.

For SKINNYe-64-256 v2, we find a 26-round related-tweakey boomerang distinguisher in Table 13 and 15 in Supplementary Material C.1.

4.3 Rectangle Attack on 41-round SKINNYe-64-256

We use the 30-round rectangle distinguisher for SKINNYe-64-256 in Table 5, whose probability is $2^{-n}\tilde{p}^2\xi\tilde{q}^2 = 2^{-64-56.47} = 2^{-120.47}$. The attack follows the Dong *et al.*'s rectangle attack framework [32], which is also given in Algorithm 3 in Supplementary Material B.1 for completeness. Adding 4-round E_b and 7-round E_f, we attack 41-round SKINNYe-64-256, as illustrated in Figure 5. For simplicity, let $STK_{j_1,j_2}^{(i)}$ be the j_1-th and j_2-th nibble of the i-th round STK. In the first round, we use subtweakey $ETK^{(0)} = \mathtt{MC} \circ \mathtt{SR}(STK^{(0)})$ instead of $STK^{(0)}$, and there is $ETK_i^{(0)} = ETK_{i+4}^{(0)} = ETK_{i+12}^{(0)} = STK_i^{(0)}$ for

Table 5: The 30-round RTK boomerang distinguisher for `SKINNYe-64-256`.

$r_0 = 12$, $r_m = 5$, $r_1 = 13$, $\tilde{p} = 2^{-3.46}$, $\xi = 2^{-30.95}$, $\tilde{q} = 2^{-9.30}$, $\tilde{p}^2\xi\tilde{q}^2 = 2^{-56.47}$
$\Delta TK_1 =$ `0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0`
$\Delta TK_2 =$ `0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0`
$\Delta TK_3 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0`
$\Delta TK_4 =$ `0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0`
$\Delta X^{(0)} =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4`
$\nabla TK_1 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1`
$\nabla TK_2 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8`
$\nabla TK_3 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0`
$\nabla TK_4 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8`
$\nabla X^{(30)} =$ `0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1`

$0 \le i \le 3$, and $ETK_8^{(0)} = STK_7^{(0)}$, $ETK_9^{(0)} = STK_4^{(0)}$, $ETK_{10}^{(0)} = STK_5^{(0)}$, $ETK_{11}^{(0)} = STK_6^{(0)}$. Construct the structures at $\bar{W}^{(0)}$ and $r_b = 12 \cdot 4 = 48$. The cells need to be guessed in E_b are $k_b = \{STK_{0,2,4}^{(2)}, STK_{0-3,5-7}^{(1)}, STK_{0-7}^{(0)}\}$ and $m_b = 18{\cdot}4 = 72$. In E_f, we have $r_f = 16{\cdot}4 = 64$ and $m_f = 45{\cdot}4 = 180$ where $k_f = \{STK_{3,7}^{(34)}, STK_{2-4,7}^{(35)}, STK_{1-7}^{(36)}, STK_{0-7}^{(37)}, STK_{0-7}^{(38)}, STK_{0-7}^{(39)}, STK_{0-7}^{(40)}\}$. The subtweakey cells guessed in advance are marked by red boxes, which are $k'_f = \{STK_{3,6,7}^{(37)}, STK_{0-2,4-7}^{(38)}, STK_{0-7}^{(39)}, STK_{0-7}^{(40)}\}$, and we have $m'_f = 26 \cdot 4 = 104$. Then, we get 7 cells in the internal states (marked by red boxes in $W^{(37)}$ and $W^{(36)}$) as additional filters with the guessed m'_f-bit key, i.e., $h_f = 7 \cdot 4 = 28$ as $\{W_{6,11,15}^{(36)}, W_{5,6,11,12}^{(37)}\}$.

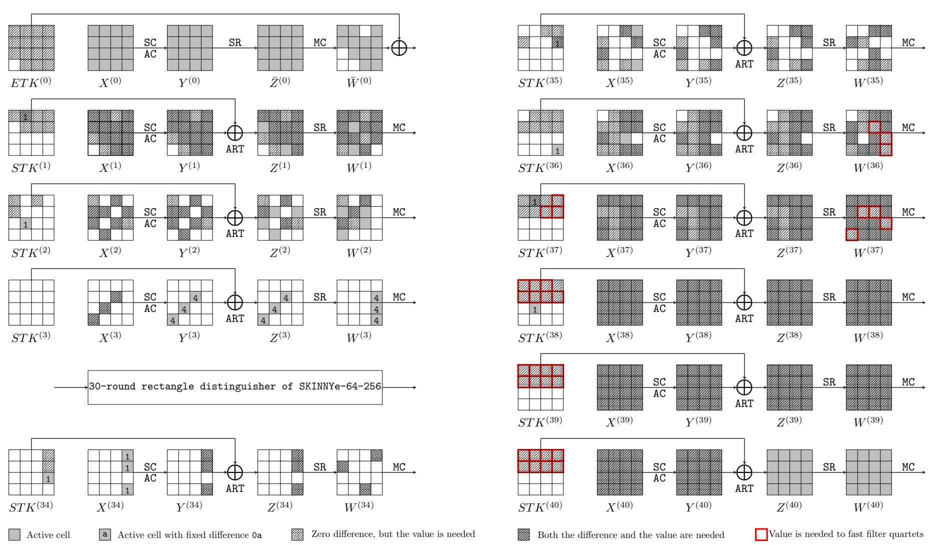

Fig. 5: The 41-round attack against `SKINNYe-64-256`.

Key bridges. To further accelerate our attack, we identify some tweakey relations in E_b and E_f according to the analysis in Section 3 . We list the subtweakeys transformed from the i-th ($0 \le i \le 15$) nibble of the master key $TK_m^{(0)}$ ($1 \le m \le 4$) in Table 6. For example in line 0 of Table 6, there are 5 subtweakeys in k_b and k_f transformed from the 0-th nibbles of $TK_m^{(0)}$, where $\left(\boldsymbol{stk}_0^{(0)}, \boldsymbol{stk}_2^{(2)}, \boldsymbol{stk}_4^{(36)}, \boldsymbol{stk}_6^{(38)}, \boldsymbol{stk}_5^{(40)}\right)^T = \boldsymbol{A}_{\{0,1,3,4,5\}} \cdot \left(\boldsymbol{tk}_{1,0}^{(0)}, \boldsymbol{tk}_{2,0}^{(0)}, \boldsymbol{tk}_{3,0}^{(0)}, \boldsymbol{tk}_{4,0}^{(0)}\right)^T$. Since $rank(\boldsymbol{A}_{\{0,1,3,4,5\}}) = 16$, the number of possible values of $\{ETK_0^{(0)} = STK_0^{(0)}, STK_2^{(2)}, STK_4^{(36)}, STK_6^{(38)}, STK_5^{(40)}\}$ is $|Im(A_{\{0,1,3,4,5\}})| = 2^{16}$. Similarly, the number of possible values of $\{ETK_0^{(0)}, STK_2^{(2)}, STK_6^{(38)}, STK_5^{(40)}\} \in k_b \cup k'_f$ is $|Im(A_{\{0,1,4,5\}})| = 2^{14}$. In total, the key size involved in E_b and E_f is only 224-bit due to the key relations although $m_b + m_f = 72 + 180 = 252$, denoted as $|k_b \cup k_f| = 2^{224}$. Similarly, we have $|k_b \cup k'_f| = 2^{170}$ although $m_b + m'_f = 72 + 104 = 176$.

Table 6: Relations of the subtweakeys involved in the 41-round attack on SKINNYe-64-256, where the subtweakeys marked in bold are among k'_f.

i	k_b	k_f		
0	$ETK_0^{(0)}, STK_2^{(2)}$	$STK_4^{(36)}, \boldsymbol{STK_6^{(38)}}, \boldsymbol{STK_5^{(40)}}$	$\lvert Im(A_{\{0,1,3,4,5\}})\rvert = 2^{16}$	$\lvert Im(A_{\{0,1,4,5\}})\rvert = 2^{14}$
1	$ETK_1^{(0)}, STK_0^{(2)}$	$STK_2^{(36)}, \boldsymbol{STK_4^{(38)}}, \boldsymbol{STK_6^{(40)}}$	$\lvert Im(A_{\{0,1,3,4,5\}})\rvert = 2^{16}$	$\lvert Im(A_{\{0,1,4,5\}})\rvert = 2^{14}$
2	$ETK_2^{(0)}, STK_4^{(2)}$	$STK_6^{(36)}, \boldsymbol{STK_5^{(38)}}, \boldsymbol{STK_3^{(40)}}$	$\lvert Im(A_{\{0,1,3,4,5\}})\rvert = 2^{16}$	$\lvert Im(A_{\{0,1,4,5\}})\rvert = 2^{14}$
3	$ETK_3^{(0)}$	$STK_7^{(34)}, STK_1^{(36)}, \boldsymbol{STK_0^{(38)}}, \boldsymbol{STK_2^{(40)}}$	$\lvert Im(A_{\{0,2,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{0,4,5\}})\rvert = 2^{12}$
4	$ETK_9^{(0)}$	$STK_5^{(36)}, STK_3^{(38)}, \boldsymbol{STK_7^{(40)}}$	$\lvert Im(A_{\{0,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{0,5\}})\rvert = 2^{8}$
5	$ETK_{10}^{(0)}$	$STK_3^{(34)}, STK_7^{(36)}, \boldsymbol{STK_1^{(38)}}, \boldsymbol{STK_0^{(40)}}$	$\lvert Im(A_{\{0,2,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{0,4,5\}})\rvert = 2^{12}$
6	$ETK_{11}^{(0)}$	$STK_3^{(36)}, \boldsymbol{STK_7^{(38)}}, \boldsymbol{STK_1^{(40)}}$	$\lvert Im(A_{\{0,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{0,4,5\}})\rvert = 2^{12}$
7	$ETK_8^{(0)}$	$\boldsymbol{STK_2^{(38)}}, \boldsymbol{STK_4^{(40)}}$	$\lvert Im(A_{\{0,4,5\}})\rvert = 2^{12}$	$\lvert Im(A_{\{0,4,5\}})\rvert = 2^{12}$
8	$STK_2^{(1)}$	$STK_4^{(35)}, \boldsymbol{STK_6^{(37)}}, \boldsymbol{STK_5^{(39)}}$	$\lvert Im(A_{\{1,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{1,4,5\}})\rvert = 2^{12}$
9	$STK_0^{(1)}$	$STK_2^{(35)}, STK_4^{(37)}, \boldsymbol{STK_6^{(39)}}$	$\lvert Im(A_{\{1,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{1,5\}})\rvert = 2^{8}$
10		$STK_5^{(37)}, \boldsymbol{STK_3^{(39)}}$	$\lvert Im(A_{\{4,5\}})\rvert = 2^{8}$	$\lvert Im(A_{\{5\}})\rvert = 2^{4}$
11	$STK_7^{(1)}$	$STK_0^{(37)}, \boldsymbol{STK_2^{(39)}}$	$\lvert Im(A_{\{1,4,5\}})\rvert = 2^{12}$	$\lvert Im(A_{\{1,5\}})\rvert = 2^{8}$
12	$STK_6^{(1)}$	$\boldsymbol{STK_3^{(37)}}, \boldsymbol{STK_7^{(39)}}$	$\lvert Im(A_{\{1,4,5\}})\rvert = 2^{12}$	$\lvert Im(A_{\{1,4,5\}})\rvert = 2^{12}$
13	$STK_3^{(1)}$	$STK_7^{(35)}, STK_1^{(37)}, \boldsymbol{STK_0^{(39)}}$	$\lvert Im(A_{\{1,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{1,5\}})\rvert = 2^{8}$
14	$STK_5^{(1)}$	$STK_3^{(35)}, \boldsymbol{STK_7^{(37)}}, \boldsymbol{STK_1^{(39)}}$	$\lvert Im(A_{\{1,3,4,5\}})\rvert = 2^{15}$	$\lvert Im(A_{\{1,4,5\}})\rvert = 2^{12}$
15	$STK_1^{(1)}$	$STK_2^{(37)}, \boldsymbol{STK_4^{(39)}}$	$\lvert Im(A_{\{1,4,5\}})\rvert = 2^{12}$	$\lvert Im(A_{\{1,5\}})\rvert = 2^{8}$
			$\lvert k_b \cup k_f\rvert = 2^{224}$	$\lvert k_b \cup k'_f\rvert = 2^{170}$

The details of our attack are given as follows:

1. Construct $y = \sqrt{s} \cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2\xi\tilde{q}^2} = \sqrt{s} \cdot 2^{12.24}$ structures of $2^{r_b} = 2^{48}$ plaintexts each. For each structure, encrypt the 2^{48} plaintexts under the four related tweakeys K_1, K_2, K_3 and K_4 to get corresponding ciphertexts and store the plaintext-ciphertext pairs in L_1, L_2, L_3 and L_4. The data and memory complexity here is both $\sqrt{s} \cdot 2^{n/2+2}/\sqrt{\tilde{p}^2\xi\tilde{q}^2} = \sqrt{s} \cdot 2^{62.24}$.
2. Guess 2^x possible values of $k_b \cup k'_f$ ($2^x \le |k_b \cup k'_f|$):
   (a) Initialize $|k_b \cup k_f|/2^x = 2^{224-x}$ counters with memory cost 2^{224-x}.
   (b) Guess all the remaining $|k_b \cup k'_f|/2^x = 2^{170-x}$ possible values in $k_b \cup k'_f$:
      i. For each structure, partially encrypt each plaintext P_1 under the guessed values of k_b to $Y_{6,9,12}^{(3)}$. After xoring the known difference α, partially decrypt it to get the plaintext P_2. Do the same for each P_3 to get P_4. Store the pairs in S_1 and S_2, whose sizes are $y \cdot 2^{r_b} = \sqrt{s} \cdot 2^{60.24}$.

ii. For each element in S_1, partially decrypt (C_1, C_2) under guessed k'_f to get $W^{(36)}_{6,11,15} \| W^{(37)}_{5,6,11,12}$. Insert the element in S_1 into a hash table H indexed by the h_f = 28-bit $W^{(36)}_{6,11,15} \| W^{(37)}_{5,6,11,12}$ of C_1 and h_f = 28-bit $\tilde{W}^{(36)}_{6,11,15} \| \tilde{W}^{(37)}_{5,6,11,12}$ of C_2. For each element in S_2, partially decrypt (C_3, C_4) under guessed k'_f to get the $2h_f = 56$ 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 $2h_f = 56$ bits. The time complexity here is $T_1 = \sqrt{s} \cdot 2^{|k_b \cup k'_f| + n/2 + 1} / \sqrt{\tilde{p}^2 \xi \tilde{q}^2} = \sqrt{s} \cdot 2^{170+32+1+28.24} = \sqrt{s} \cdot 2^{231.24}$. We get $s \cdot 2^{|k_b \cup k'_f| - 2h_f - n + 2r_f} / (\tilde{p}^2 \xi \tilde{q}^2) = s \cdot 2^{170-56-64+128+56.47} = s \cdot 2^{234.47}$ quartets.

iii. For each of the $s \cdot 2^{234.47}$ quartets, determine the key candidates step by step, whose time complexity is ε:

A: In round 38, guess 2^4 possible values of $STK^{(38)}_3$. As shown in Table 7, with other guessed k'_f together, we compute $Z^{(37)}_{0,12}$ and deduce $\Delta Y^{(37)}_0$ and $\Delta X^{(37)}_{12}$. For the 1st column of $X^{(37)}$ of (C_1, C_3), we obtain $\Delta X^{(37)}_0 = \Delta X^{(37)}_{12}$ by property of `MC`, and deduce $STK^{(37)}_0$ by Lemma 1. Similarly, we deduce $STK'^{(37)}_0$ for (C_2, C_4). Then the fixed $\Delta STK^{(37)}_0 = STK^{(37)}_0 \oplus STK'^{(37)}_0$ is a 4-bit filter. $s \cdot 2^{234.47} \cdot 2^4 \cdot 2^{-4} = s \cdot 2^{234.47}$ quartets remain.

B: In round 37, guessing 2^4 possible values of $STK^{(37)}_2$, following Table 7 we compute $Z^{(36)}_{3,15}$ and deduce $\Delta Y^{(36)}_3$ and $\Delta X^{(36)}_{15}$. For the 4-th column of $X^{(36)}$ of (C_1, C_3), we deduce $\Delta X^{(36)}_3 = \Delta X^{(36)}_{15}$ by `MC` and deduce $STK^{(36)}_3$. Since the number of possible values[2] of $STK^{(36)}_3$ is only 2^3 as shown in Table 7, which acts as a filter of $2^3/2^4 = 2^{-1}$. Similarly, we deduce $STK'^{(36)}_3$ for (C_2, C_4). Then the fixed $\Delta STK^{(36)}_3$ is a 4-bit filter. $s \cdot 2^{234.47} \cdot 2^4 \cdot 2^{-1} \cdot 2^{-4} = s \cdot 2^{233.47}$ quartets remain.

C: Guessing 2^4 possible values of $STK^{(37)}_4$, we compute $Z^{(36)}_7$ and deduce $\Delta Y^{(36)}_7$. For the 4-th column of $X^{(36)}$ of (C_1, C_3), we can obtain $\Delta X^{36}_7 = \Delta X^{(36)}_{15}$ by `MC`. With the known $\Delta X^{(36)}_{15}$ in **step B**, we deduce $STK^{(36)}_7$. The number of possible values of $STK^{(36)}_7$ is 2^3, which can act as a filter of $2^3/2^4 = 2^{-1}$. Similarly, we deduce $STK'^{(36)}_7$ for (C_2, C_4). Then the fixed $\Delta STK^{(36)}_7$ is a 4-bit filter. $s \cdot 2^{233.47} \cdot 2^4 \cdot 2^{-1} \cdot 2^{-4} = s \cdot 2^{232.47}$ quartets remain.

---

[2] The number of possible values of $STK^{(36)}_3$ is computed via Table 6. For example, in line 6 of Table 6, $\{ETK^{(0)}_{11}, STK^{(38)}_7, STK^{(40)}_1\} \in k_b \cup k'_f$ derived from the 6-th nibble have already been guessed, so the number of possible values of $STK^{(36)}_3$ is $|Im(A_{\{0,3,4,5\}})| / |Im(A_{\{0,4,5\}})| = 2^{15-12} = 2^3$. Similarly, we compute all the number of possible values for subtweakey cells involved in the guess and filter procedure, which are listed in Table 7.

D: Guessing 2^4 possible values of $STK_1^{(37)}$, we compute $Z_{6,10,14}^{(36)}$. Then $\Delta Y_6^{(36)}$ and $\Delta X_{10,14}^{(36)}$ are deduced. For the 3rd column of $X^{(36)}$ of (C_1, C_3), we can obtain $\Delta X_6^{(36)} = \Delta X_{10}^{(36)} \oplus \Delta X_{14}^{(36)}$ by MC and deduce $STK_6^{(36)}$. The number of possible values of $STK_6^{(36)}$ is 2^2, which acts as a filter of $2^2/2^4 = 2^{-2}$. Similarly, we deduce $STK_6'^{(36)}$ for (C_2, C_4) and $\Delta STK_6^{(36)}$ can act as a 4-bit filter. $s \cdot 2^{232.47} \cdot 2^4 \cdot 2^{-2} \cdot 2^{-4} = s \cdot 2^{230.47}$ quartets remain.

E: In round 36, guessing $2^4 \times 2^2 \times 2^2$ possible values for ($STK_5^{(37)}$, $STK_2^{(36)}$, $STK_4^{(36)}$), we compute $Z_{3,7,15}^{(35)}$ and deduce $\Delta Y_{3,7}^{(35)}$ and $\Delta X_{15}^{(35)}$. For the 4-th column of $X^{(35)}$ of (C_1, C_3), we can obtain $\Delta X_3^{(35)} = \Delta X_7^{(35)} = \Delta X_{15}^{(35)}$ by MC and deduce $STK_3^{(35)}$ and $STK_7^{(35)}$. Both the numbers of possible values of $STK_3^{(35)}$ and $STK_7^{(35)}$ are 2^3, which acts as two filters of $2^3/2^4 = 2^{-1}$. Similarly, we deduce $STK_3'^{(35)}$ and $STK_7'^{(35)}$ for (C_2, C_4). Then the fixed $\Delta STK_3^{(35)}$ and $\Delta STK_7^{(35)}$ can act as two 4-bit filters. Thereafter, in round 34, we deduce $Z_3^{(34)}$ from $Z_7^{(35)}$ and $STK_7^{(35)}$. Since $STK_3^{(34)}$ only has one possible value[3], we deduce $X_3^{(34)}$. So $\Delta X_3^{(34)} = \texttt{0x1}$ acts a 4-bit filter both for (C_1, C_3) and (C_2, C_4). $s \cdot 2^{230.47} \cdot 2^8 \cdot 2^{-1} \cdot 2^{-1} \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{220.47}$ quartets remain.

F: Guessing $2^3 \times 2^3 \times 2^3 \times 2^3$ possible values of ($STK_1^{(36)}$, $STK_5^{(36)}$, $STK_2^{(35)}$, $STK_4^{(35)}$), compute $Z_{7,15}^{(34)}$ and deduce $X_{15}^{(34)}$. Since $STK_7^{(34)}$ only has one possible value, we can deduce $X_7^{(34)}$. $\Delta X_7^{(34)} = \texttt{0x1}$ and $\Delta X_{15}^{(34)} = \texttt{0x1}$ are two 4-bit filters for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{220.47} \cdot 2^{12} \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{216.47}$ quartets remain.

So for each quartet, $\varepsilon = 2^4 \cdot \frac{4}{41} + 2^4 \cdot \frac{4}{41} + 2^{-1} \cdot 2^4 \cdot \frac{4}{41} + 2^{-2} \cdot 2^4 \cdot \frac{4}{41} + 2^{-4} \cdot 2^8 \cdot \frac{4}{41} + 2^{-14} \cdot 2^{12} \cdot \frac{4}{41} \approx 2^{2.56}$ and $T_2 = s \cdot 2^{234.47} \cdot \varepsilon = s \cdot 2^{237.03}$.

(c) (Exhaustive search) Select the top $|k_b \cup k_f| \cdot 2^{-x-h} = 2^{224-x-h}$ hits in the counter as the key candidates. Guess the remaining $k - 224 = 32$-bit key to check the full key, and $T_3 = 2^{k-h}$.

Set $s = 1$, $h = 32$ and $x = 168$ ($x \leq 170$, $h \leq 224 - x$). We have $T_1 = 2^{231.24}$, $T_2 = 2^{237.03}$ and $T_3 = 2^{224}$. The memory complexity is $2^{62.24} + 2^{56} \approx 2^{62.26}$. In total, for the 41-round attack on SKINNYe-64-256, the data complexity is $2^{62.24}$, the memory complexity is $2^{62.26}$, and the time complexity is $2^{237.06}$. The success probability is about 70.6%.

In addition, for SKINNYe-64-256 v2 we give a 37-round related-tweakey rectangle attack (given in the Supplementary Material C.2) based on the 26-round related-tweakey boomerang distinguisher (Table 15). The data complexity is $2^{62.8}$, the memory complexity is $2^{62.8}$, and the time complexity is $2^{240.03}$. The success probability is about 66.3%.

[3] As shown in line 5 of Table 6, with $STK_7^{(36)}$ deduced in **step C** and other cells guessed in $k_b \cup k_f'$, the number of possible values is only 1 for $STK_3^{(34)}$.

Table 7: Tweakey recovery for 41-round `SKINNYe-64-256`. The red cells are among k'_f or gained in the previous steps. `D`/`G`: deduced/guessed subtweakeys.

Step	State	Involved subtweakeys	Number of values
A	$Z_0^{(37)}$ $Z_{12}^{(37)}$	$STK_4^{(38)}, STK_5^{(39)}, STK_{0,6,7}^{(40)}$ $STK_3^{(38)}, STK_{2,7}^{(39)}, STK_{1,4,6}^{(40)}$	`G`: $STK_3^{(38)} : 2^4$ `D`: $STK_0^{(37)} : 2^4$
B	$Z_3^{(36)}$ $Z_{15}^{(36)}$	$STK_7^{(37)}, STK_4^{(38)}, STK_{3,5,6}^{(39)}, STK_{0-2,6,7}^{(40)}$ $STK_2^{(37)}, STK_{1,6}^{(38)}, STK_{0,5,7}^{(39)}, STK_{2-6}^{(40)}$	`G`: $STK_2^{(37)} : 2^4$ `D`: $STK_3^{(36)} : 2^3$
C	$Z_7^{(36)}$	$STK_4^{(37)}, STK_{3,5,6}^{(38)}, STK_{0-2,6,7}^{(39)}, STK_{0-7}^{(40)}$	`G`: $STK_4^{(37)} : 2^4$ `D`: $STK_7^{(36)} : 2^3$
D	$Z_6^{(36)}$ $Z_{10}^{(36)}$ $Z_{14}^{(36)}$	$STK_7^{(37)}, STK_{2,4,5}^{(38)}, STK_{0,1,3,5,6}^{(39)}, STK_{0-7}^{(40)}$ $STK_4^{(37)}, STK_{3,5}^{(38)}, STK_{0,2,6,7}^{(39)}, STK_{1-4,6,7}^{(40)}$ $STK_1^{(37)}, STK_{0,5}^{(38)}, STK_{3,4,6}^{(39)}, STK_{1,2,4,5,7}^{(40)}$	`G`: $STK_1^{(37)} : 2^4$ `D`: $STK_6^{(36)} : 2^2$
E	$Z_3^{(35)}$ $Z_7^{(35)}$ $Z_{15}^{(35)}$ $Z_3^{(34)}$	$STK_7^{(36)}, STK_4^{(37)}, STK_{3,5,6}^{(38)}, STK_{0-2,6,7}^{(39)}, STK_{0-7}^{(40)}$ $STK_4^{(36)}, STK_{3,5,6}^{(37)}, STK_{0-2,6,7}^{(38)}, STK_{0-7}^{(39)}, STK_{0-7}^{(40)}$ $STK_2^{(36)}, STK_{1,6}^{(37)}, STK_{0,5,7}^{(38)}, STK_{2-6}^{(39)}, STK_{0-7}^{(40)}$ $STK_7^{(35)}, STK_4^{(36)}, STK_{3,5,6}^{(37)}, STK_{0-2,6,7}^{(38)}, STK_{0-7}^{(39)}, STK_{0-7}^{(40)}$	`G`: $STK_5^{(37)} : 2^4$ `G`: $STK_2^{(36)} : 2^2$ `G`: $STK_4^{(36)} : 2^2$ `D`: $STK_3^{(35)} : 2^3$ `D`: $STK_7^{(35)} : 2^3$ `D`: $STK_3^{(34)} : 2^0$
F	$Z_7^{(34)}$ $Z_{15}^{(34)}$	$STK_4^{(35)}, STK_{3,5,6}^{(36)}, STK_{0-2,6,7}^{(37)}, STK_{0-7}^{(38)}, STK_{0-7}^{(39)}, STK_{0-7}^{(40)}$ $STK_2^{(35)}, STK_{1,6}^{(36)}, STK_{0,5,7}^{(37)}, STK_{2-6}^{(38)}, STK_{0-7}^{(39)}, STK_{0-7}^{(40)}$	`G`: $STK_1^{(36)} : 2^3$ `G`: $STK_5^{(36)} : 2^3$ `G`: $STK_2^{(35)} : 2^3$ `G`: $STK_4^{(35)} : 2^3$ `D`: $STK_7^{(34)} : 2^0$

5 MITM Attacks on `SKINNYe-64-256` and its Version 2

The three-subset meet-in-the-middle attack was proposed by Bogdanov and Rechberger [22] and was well summarized by Isobe [42]. Several important techniques significantly enhance and enrich the MITM methodology, including the *splice-and-cut* technique [6], initial structure [58,57], (indirect-)partial matching [58,57], sieve-in-the-middle [24], match-box technique [37], and dissection [29], etc. Recently, several automatic tools [27,56,8,9,59] on MITM attacks are presented. At CRYPTO 2021, Dong *et al.* [31] developed the MILP model for MITM key-recovery attack on `SKINNY`, and the related definitions and symbols of MITM attack are given in Supplementary Material D.1.

5.1 MILP Model for MITM Attacks on `SKINNYe-64-256`

The MITM attack cuts the whole cipher into forward and backward computation chunks, which overlaps in a common intermediate round (i.e. matching point). In each of the chunks, the computation involves a set of key bits, such that they can be computed over all values of the involved key bits independently from the key bits involved in the other chunk (the distinct key bits are called neutral words). In order to perform a successful MITM attack, both the number of possible values of forward and backward neutral words (denoted as degrees of freedom of forward and backward neutral words) must be larger than 1. In Dong *et al.*'s model [31], the degree of freedom for forward and backward neutral words is counted in cells, which is denoted as DoF^+ and DoF^-, respectively. Then, they add the constraint $\mathrm{DoF}^+ \geq 1$ and $\mathrm{DoF}^- \geq 1$ in the model to ensure the number of possible values of the neutral words to be larger than 1. The reason behind is that for SKINNY when $\mathrm{DoF}^+ = 0$ or $\mathrm{DoF}^- = 0$, the linear constraints for the neutral words will lead to a full-rank linear system (i.e., the Equation (25) or (26) in Supplementary Material D.1 will have only one solution), which leads to the size of neutral words to be 1 and fails the MITM attack.

When applying Dong *et al.*'s model to SKINNYe-64-256, the situation becomes different due to the new tweakey schedule. For example, assume the initial degrees of freedom from the key schedule path as $\lambda^+_{\mathtt{KSA}} = \lambda^-_{\mathtt{KSA}} = 4$ and the consumed degrees of freedom $l^+_{\mathtt{KSA}} = l^-_{\mathtt{KSA}} = 4$. In this case we will get $\mathrm{DoF}^+ = 4 - 4 = 0, \mathrm{DoF}^- = 4 - 4 = 0$, which will never be the solutions of Dong *et al.*'s MILP model (invalidate the constraints $\mathrm{DoF}^+ \geq 1, \mathrm{DoF}^- \geq 1$). However, when building the linear constraint systems for the neutral key words as Equation (25) and (26), the coefficient matrices (denoted as $\boldsymbol{A}$) can be of non-full rank even though $\mathrm{DoF}^+ = 0$ or $\mathrm{DoF}^- = 0$. More precisely, according to Section 3 and Table 4, those coefficient matrices with $rank = 14$ will make the degree of freedom for the neutral words be 2^2, which therefore validates the MITM attack. In order to cover all the possible solutions for the MITM attacks on SKINNYe-64-256, we build two different kinds of models:

- The first one maintain similar constraints from Dong *et al.*'s model including the constraints $\mathrm{DoF}^+ \geq 1,\ \mathrm{DoF}^- \geq 1$.
- The second one takes the property in Section 3 into account. We only consider those coefficient matrices with $rank = 14$ in Table 4, and discard those with $rank = 15$ since they will lead to attacks with almost exhaustive attack. Let st^+, st^- be the starting rounds of the degrees of freedom consumption for forward and backward chunks. Suppose for forward chunk and backward chunk, the equivalence classes used are $[\{r_1^+, r_2^+, r_3^+, r_4^+\}]$ and $[\{r_1^-, r_2^-, r_3^-, r_4^-\}]$ (selected from equivalence classes with $rank = 14$ in Table 4), respectively. Then, the following constraints will be added to replace $\mathrm{DoF}^+ \geq 1,\ \mathrm{DoF}^- \geq 1$:
$$\begin{cases} \mathrm{DoF}^+ = 0,\ \mathrm{DoF}^- = 0, \\ \sum_{i=0}^{15} \delta^{(r^+)}_{\mathtt{KSA}}[i] = 1,\ \text{for } r^+ = st^+ + 2r_j^+ \text{ and } j = 1,2,3,4, \\ \sum_{i=0}^{15} \delta^{(r^+)}_{\mathtt{KSA}}[i] = 0, \text{otherwise}, \\ \sum_{i=0}^{15} \delta^{(r^-)}_{\mathtt{KSA}}[i] = 1,\ \text{for } r^+ = st^- + 2r_j^- \text{ and } j = 1,2,3,4, \\ \sum_{i=0}^{15} \delta^{(r^-)}_{\mathtt{KSA}}[i] = 0, \text{otherwise}, \end{cases} \tag{11}$$
  where Boolean variable $\delta^{(r^{+/-})}_{\mathtt{KSA}}[i] = 1$ if and only if the degrees of freedom of the forward/backward neutral words are consumed in i-th cell in round r^+ or round r^-. We build different MILP models for parameters $(st^+, st^-, r_i^+, r_i^-)$ with $i = 1,2,3,4$.

The other parts of the model for SKINNYe-64-256 similar with Dong *et al.*'s model can refer to Supplementary Material D.2 and the source code is provided in https://github.com/skinny64/Skinny64-256. We run different kinds of the models to find optimal solutions.

5.2 MITM Attack on 31-Round SKINNYe-64-256

The best MITM attack we discover is based on the rank-equivalence class $[\{0,1,2,3\}]$ for both forward and backward chunk, which is a 31-round single-key attack on SKINNYe-64-256 as shown in Figure 10 in Supplementary Material D.3. The starting states are $\mathcal{S}^{\texttt{ENC}} = X^{(1)}$ and $\mathcal{S}^{\texttt{KSA}} = (TK_1^{(1)}, TK_2^{(1)}, TK_3^{(1)}, TK_4^{(1)})$. The matching point is between $Z^{(16)}$ and $X^{(17)}$, which forms a 1-cell filter. There are 4 ■ cells and 4 ■ cells in $\mathcal{S}^{\texttt{KSA}}$, so $\lambda^+_{\texttt{KSA}} = \lambda^-_{\texttt{KSA}} = 4$ cells. The highlevel procedures of the MITM key-recovery attack on SKINNYe-64-256 are given as follows:

1. Assign arbitrary compatible values to all bytes except those that depend on the neutral bytes.
2. Collecting a structure of plaintext-ciphertext pairs and store them in a table H.
3. Compute the solution spaces of ■ and ■ (neutral words for the forward and backward computations) under the constraints on them.
4. For all possible values of ■, compute forward to the matching point to get a table L_1, whose indices are the values for matching and the elements are the values of ■.
5. For all possible values of ■, compute backward to the matching point to get a table L_2, whose indices are the values for matching and the elements are the values of ■.
6. Check whether there is a match on indices between L_1 and L_2. If there is a partial-matching, check for a full-state match. In case none of them are fully matched, repeat the procedure by changing values assigned in Step 1 till find a full match.

To peform the detailed attack, we firstly need to compute the solution space of ■ and ■ cells. For the forward computation with ■, we need to make the Equation (12) hold:

$$\begin{cases} TK_{1,2}^{(6)} \oplus TK_{2,2}^{(6)} \oplus TK_{3,2}^{(6)} \oplus TK_{4,2}^{(6)} = a_1, \\ TK_{1,4}^{(8)} \oplus TK_{2,4}^{(8)} \oplus TK_{3,4}^{(8)} \oplus TK_{4,4}^{(8)} = a_2, \\ TK_{1,6}^{(10)} \oplus TK_{2,6}^{(10)} \oplus TK_{3,6}^{(10)} \oplus TK_{4,6}^{(10)} = a_3, \\ TK_{1,5}^{(12)} \oplus TK_{2,5}^{(12)} \oplus TK_{3,5}^{(12)} \oplus TK_{4,5}^{(12)} = a_4, \end{cases} \tag{12}$$

where a_1, a_2, a_3, a_4 are constants. Assume $\{TK_{1,2}^{(6)}, TK_{2,2}^{(6)}, TK_{3,2}^{(6)}, TK_{4,2}^{(6)}\}$ represents the master tweakey and $\mathfrak{c}^+ = (a_1, a_2, a_3, a_4) \in \mathbb{F}_2^{16}$. Using the notations in Section 3, the Equation (12) can be rewritten as

$$\boldsymbol{A}_{\{0,1,2,3\}} \cdot [\boldsymbol{tk}_{1,2}^{(6)}, \boldsymbol{tk}_{2,2}^{(6)}, \boldsymbol{tk}_{3,2}^{(6)}, \boldsymbol{tk}_{4,2}^{(6)}]^T = \mathfrak{c}^+. \tag{13}$$

For Equation (13), we can know $rank(\boldsymbol{A}_{\{0,1,2,3\}}) = 14$ from Table 4 and the size of its image space $|Im(\boldsymbol{A})| = 2^{14}$. Therefore, if $\mathfrak{c}^+ \in Im(\boldsymbol{A})$, there will be 2^2 solutions of Equation (13). Otherwise, it will have no solution. Before the attack, we precompute the image space of $\boldsymbol{A}_{\{0,1,2,3\}}$ and store it in a table $\texttt{Im}^+_{\boldsymbol{A}_{\{0,1,2,3\}}}$. In other words, for given $\mathfrak{c}^+ \in Im(\boldsymbol{A})$, the consumed degrees of freedom of neutral words for ■ is 14 bits. Therefore, the remaining degrees of freedom for ■ is $4\lambda^{+}_{\texttt{KSA}} - 14 = 16 - 14 = 2$ bits, i.e., the solution space of ■ neutral words is 2^2.

Similarly, for the backward computation with ■ cells, there needs Equation (14) to hold:

$$\begin{cases} TK^{(23)}_{1,0} \oplus TK^{(23)}_{2,0} \oplus TK^{(23)}_{3,0} \oplus TK^{(23)}_{4,0} = b_1, \\ TK^{(25)}_{1,2} \oplus TK^{(25)}_{2,2} \oplus TK^{(25)}_{3,2} \oplus TK^{(25)}_{4,2} = b_2, \\ TK^{(27)}_{1,4} \oplus TK^{(27)}_{2,4} \oplus TK^{(27)}_{3,4} \oplus TK^{(27)}_{4,4} = b_3, \\ TK^{(29)}_{1,6} \oplus TK^{(29)}_{2,6} \oplus TK^{(29)}_{3,6} \oplus TK^{(29)}_{4,6} = b_4. \end{cases} \tag{14}$$

Let $\mathfrak{c}^- = (b_1, b_2, b_3, b_4) \in \mathbb{F}_2^{16}$ be constants, there is

$$\boldsymbol{A}_{\{0,1,2,3\}} \cdot [\boldsymbol{tk}^{(23)}_{1,0}, \boldsymbol{tk}^{(23)}_{2,0}, \boldsymbol{tk}^{(23)}_{3,0}, \boldsymbol{tk}^{(23)}_{4,0}]^T = \mathfrak{c}^-. \tag{15}$$

Precompute the solution space of $\mathfrak{c}^-$ in table $\mathtt{Im}^-_{\boldsymbol{A}_{\{0,1,2,3\}}}$ before the attack. The consumed degrees of freedom of neutral words for the backward computation is 14 bits. Therefore, the remaining degrees of freedom for the backward computation is $4\lambda^{-}_{\texttt{KSA}} - 14 = 16 - 14 = 2$ bits, i.e., the solution space of ■ neutral words for the backward computation is 2^2.

We give the details of 31-round MITM key-recovery attack in Algorithm 1. The data and memory complexity is 2^{52}, which is bounded by Line 3 in Algorithm 1. According to the Equation (27), we can get the time complexity is about 2^{254}.

In addition, we give a 27-round MITM attack on SKINNYe-64-256 v2 (see Supplementary Material D.3). The time complexity is about 2^{252}, the data and memory complexity is 2^{52}.

6 Related-Tweakey Impossible Differential Attacks

The impossible differential attack is proposed by Biham *et al.* [16] and Knudsen [46] independently. It uses a differential with probability zero to act as a distinguisher, named as the impossible differential. With several rounds appended before and after the impossible differential distinguisher, one partially encrypts/decrypts a given pair by a candidate key to the input and output of the distinguisher. The key candidate that leads to the impossible differential will be the wrong one and will be rejected. This technique provides a sieving of the key space and the remaining candidates can be tested by exhaustive search. There are several works analyzed the security of SKINNY family against the impossible differential attacks [47,61,5,55,33], in both single-tweakey and related-tweakey setting.

For SKINNYe-64-256, applying the properties of subtweakey difference cancellations introduced in Section 3 we construct a 21-round related-tweakey impossible differential. The impossible differential is placed at Round 3 to Round 24 as illustrated in Figure 12 in Supplementary Material E, which has only one active nibble in the master tweakey. For the active 9-th nibble of the master tweakey, the subtweakey difference cancellations happen 6 times at Round 5, 7, 9, 13, 15 and 21, where $(\boldsymbol{stk}^{(5)}_4, \boldsymbol{stk}^{(7)}_6, \boldsymbol{stk}^{(9)}_5, \boldsymbol{stk}^{(13)}_7, \boldsymbol{stk}^{(15)}_1, \boldsymbol{stk}^{(21)}_4)^T = \boldsymbol{A}_{\{3,4,5,7,8,11\}} \cdot$

Algorithm 1: The MITM key-recovery attack on `SKINNYe-64-256`

1 Compute the $\mathtt{Im}^{+}_{\boldsymbol{A}_{\{0,1,2,3\}}}$ and $\mathtt{Im}^{-}_{\boldsymbol{A}_{\{0,1,2,3\}}}$

2 $X^{(0)}_{0,10,14} \leftarrow 0$, $X^{(1)}_{2,3,9,11,13} \leftarrow 0$, $X^{(2)}_{2,8,10,12} \leftarrow 0$, $Y^{(2)}_{1} \leftarrow 0$, $X^{(3)}_{1,9} \leftarrow 0$, $Y^{(4)}_{0} \leftarrow 0$.

3 Collecting structure of plaintext-ciphertext pairs and store them in table H, which traverses the non-constant 16-3=13 cells in the plaintext

4 **for** *All possible values of the* ■ *cells in* $(TK^{(1)}_1, TK^{(1)}_2, TK^{(1)}_3, TK^{(1)}_4)$ **do**

5   Compute all other unknown Gray cells according to the values assigned in step 2 and step 3.

6   **for** $(a_1, a_2, a_3, a_4) \in \mathtt{Im}^{+}_{\boldsymbol{A}_{\{0,1,2,3\}}}$ *and* $(b_1, b_2, b_3, b_4) \in \mathtt{Im}^{-}_{\boldsymbol{A}_{\{0,1,2,3\}}}$ **do**

7   Derive the solution space of the ■ cells by Eq. (12) and store it in T_1.

8   Derive the solution space of the ■ cells by Eq. (14) and store it in T_2.

9   Initialize L to be an empty hash table.

10   **for** *the values in* T_2 **do**

11   Compute $X^{(17)}_{11}$ along the backward computation path: $X^{(4)} \to X^{(0)} \to E_K(X^{(0)}) \to X^{(17)}$ by accessing H

12   Insert relative information into L indexed by $X^{(17)}_{11}$

13   **end**

14   **for** *the values in* T_1 **do**

15   Compute $Z^{(16)}_{7}$ and $Z^{(16)}_{11}$ along the forward computation path: $X^{(1)} \to Z^{(16)}$

16   **for** *Candidate keys in* $L[Z^{(16)}_{7} \oplus Z^{(16)}_{11}]$ **do**

17   Test the guessed key with several plaintext-ciphertext pairs.

18   **end**

19   **end**

20   **end**

21 **end**

$\left(\boldsymbol{tk}^{(0)}_{1,9}, \boldsymbol{tk}^{(0)}_{2,9}, \boldsymbol{tk}^{(0)}_{3,9}, \boldsymbol{tk}^{(0)}_{4,9}\right)^T = \mathbf{0}$. With $rank(\boldsymbol{A}_{\{3,4,5,7,8,11\}}) = rank(\boldsymbol{A}_{\{0,1,2,4,5,8\}}) = 15$, we find the only one nonzero solution, i.e., $\boldsymbol{tk}^{(0)}_{1,9} = [0,0,0,0]^T$, $\boldsymbol{tk}^{(0)}_{2,9} = [1,1,1,1]^T$, $\boldsymbol{tk}^{(0)}_{3,9} = [0,0,0,0]^T$ and $\boldsymbol{tk}^{(0)}_{4,9} = [1,1,1,0]^T$. Then the subtweakey differences in each round can be deduced. The impossible differential is represented as

$$(\mathtt{0010\ 0000\ 0000\ 0000}) \nrightarrow (\mathtt{0000\ 0000\ 0001\ 0000}).$$

Remark. With the difference cancellation properties, we find a 21-round impossible differential for `SKINNYe-64-256` based on a cancellation pattern, while previous impossible differentials reach 16 rounds [47] for `SKINNY-64-192` and 15 rounds [55] for `SKINNY-64-128`, respectively.

For `SKINNYe-64-256` v2, the impossible differential is 18 rounds, which is in the expectation of the designers (see Supplementary Material E).

7 A Proposal for Tweakey Schedule of `SKINNY` Family

At ASIACRYPT 2014, Jean *et al.* [43] introduced the `STK` construction as shown in Figure 1, which absorbs arbitrary length of tweakey. It updates each cell of the tweakey states by multiplying a non-zero α_j. For SKINNY-n-zn, the tweakey cells are updated by dedicated chosen lightweight LFSRs, which guarantees at most $z-1$ cancellations within 30 consecutive rounds. However, SKINNY family [10] only gives instances for $z = 1, 2, 3$. SKINNYe-64-256 [49] extends z to 4, but fails to satisfy its expected security claim[4]. In the updated version SKINNYe-64-256 v2 [50], the designers fixed the issue and claimed that the LFSR for TK_4 is the only one to ensure at most 3 cancellations after exhaustively testing 2^{16} choices. It is not trivial to extend SKINNY to support arbitrary length of tweakey with similar *subtweakey difference cancellation property* to STK construction: for a given cell position, $z-1$ cancellations can only happen every 15 rounds for TK-z (or every 30 rounds for SKINNY-n-zn).

As stated by Naito *et al.* [49, Page 5] that PFB_Plus "... give new insight to TBC designers considering that there is no consensus about the adequate tweak size to support". It is interesting to consider a uniformed tweakey schedule to extend SKINNY to support larger tweakey size, while obeying the property of STK construction, which may have potential application, such as SKINNYe-64-256 v2 in TI-friendly constructions.

For general $z \leq 14$, the output nibbles can be represented by linear combinations of the input nibbles as in equation (4), i.e.,

$$\begin{pmatrix} \boldsymbol{stk}^{(2\times 0)}_{\bar{P}^{2\times 0}[i]} \\ \boldsymbol{stk}^{(2\times 1)}_{\bar{P}^{2\times 1}[i]} \\ \vdots \\ \boldsymbol{stk}^{(2\times 14)}_{\bar{P}^{2\times 14}[i]} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{L}_2^0 & \cdots & \boldsymbol{L}_z^0 \\ \boldsymbol{I} & \boldsymbol{L}_2^1 & \cdots & \boldsymbol{L}_z^1 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I} & \boldsymbol{L}_2^{14} & \cdots & \boldsymbol{L}_z^{14} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{tk}^{(0)}_{1,i} \\ \boldsymbol{tk}^{(0)}_{2,i} \\ \vdots \\ \boldsymbol{tk}^{(0)}_{z,i} \end{pmatrix}, \ 0 \leq i \leq 7. \tag{16}$$

To satisfy the subtweakey difference cancellation property, the coefficient matrix in (16) must satisfy the 'block-MDS' property [43], i.e.,

$$\det \begin{pmatrix} \boldsymbol{I} & \boldsymbol{L}_2^{r_1} & \cdots & \boldsymbol{L}_z^{r_1} \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_2} & \cdots & \boldsymbol{L}_z^{r_2} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{I} & \boldsymbol{L}_2^{r_z} & \cdots & \boldsymbol{L}_z^{r_z} \end{pmatrix} \neq 0 \tag{17}$$

for all $0 \leq r_1 < r_2 < \cdots < r_z \leq 14$. In other words, the goal of our design is to choose $\boldsymbol{L}_i$'s such that the 'block-MDS' property is guaranteed. Although the coefficient matrix in (16) has a block Vandermonde form, there is no simple formula to compute the determinant of its squared sub-matrices for general $\boldsymbol{L}_i$'s. When the $\boldsymbol{L}_i$'s are pairwise commutable, a formula can be deduced for squared block Vandermonde matrices, which we refer to Supplementary Material F.

[4] Similar issue happens to Lilliput-AE [1], one of the first-round candidates at the NIST competition, specifies TBCs with up to $z = 7$. However, they also ignored the rationale of the original tweakey framework to ensure the security, and were actually attacked practically [34].

7.1 The Choice of $\boldsymbol{L}_i$

Our construction can be viewed as an extension of the generator matrices of Reed-Solomon codes to the block matrix form. Specifically, denoting $\boldsymbol{L}_1 = \boldsymbol{I}$, and we choose the $\boldsymbol{L}_i$'s to be consecutive powers of a matrix $\boldsymbol{L}$, i.e.,

$$\{\boldsymbol{L}_i\}_{1 \le i \le z} = \{\boldsymbol{L}^{\alpha+1}, \cdots, \boldsymbol{L}^{\alpha+z}\} \tag{18}$$

for some integer $\alpha \in [-z, -1]$. Then we can show that the 'block-MDS' property is guaranteed if the matrix $\boldsymbol{L}$ satisfies specific property.

Proposition 1. *Suppose $\boldsymbol{L}$ is a 4×4 matrix over $GF(2)$ such that the characteristic polynomial $p_{\boldsymbol{L}}(\lambda)$ is a primitive polynomial of degree 4 over $GF(2)$. Then $\boldsymbol{L}$ has cycle 15, and for any integer α,*

$$\det \begin{pmatrix} (\boldsymbol{L}^{\alpha+1})^{r_1} & (\boldsymbol{L}^{\alpha+2})^{r_1} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_1} \\ (\boldsymbol{L}^{\alpha+1})^{r_2} & (\boldsymbol{L}^{\alpha+2})^{r_2} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_2} \\ \vdots & \vdots & \ddots & \vdots \\ (\boldsymbol{L}^{\alpha+1})^{r_z} & (\boldsymbol{L}^{\alpha+2})^{r_z} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_z} \end{pmatrix} \neq 0 \tag{19}$$

for all $0 \le r_1 < r_2 < \cdots < r_z \le 14$.

Proof. Let $\lambda_i, 1 \le i \le 4$, be the eigenvalues of $\boldsymbol{L}$, then λ_i is primitive in $GF(2^4)$ and $\boldsymbol{L}^r$ has eigenvalues $\lambda_i^r, 1 \le i \le 4$. For $1 \le r < 15$, we have $\lambda_i^r \neq 1, 1 \le i \le 4$, and thus $\boldsymbol{L}^r \neq \boldsymbol{I}$. For $r = 15$, note that $p_{\boldsymbol{L}}(\lambda) \mid (\lambda^{15} - 1)$, and by the Cayley–Hamilton theorem (see Section 9 of [54]) we have $p_{\boldsymbol{L}}(\boldsymbol{L}) = \boldsymbol{0}$. Then it follows that $\boldsymbol{L}^{15} - \boldsymbol{I} = \boldsymbol{0}$.

To show the determinant is nonzero, we observe that

$$\begin{pmatrix} (\boldsymbol{L}^{\alpha+1})^{r_1} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_1} \\ (\boldsymbol{L}^{\alpha+1})^{r_2} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_2} \\ \vdots & \ddots & \vdots \\ (\boldsymbol{L}^{\alpha+1})^{r_z} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_z} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}^{\alpha r_1} & & & \\ & \boldsymbol{L}^{\alpha r_2} & & \\ & & \ddots & \\ & & & \boldsymbol{L}^{\alpha r_z} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{L}^{r_1} & \cdots & (\boldsymbol{L}^{r_1})^z \\ \boldsymbol{L}^{r_2} & \cdots & (\boldsymbol{L}^{r_2})^z \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}^{r_z} & \cdots & (\boldsymbol{L}^{r_z})^z \end{pmatrix}. \tag{20}$$

Then it suffices to show that $\det \left((\boldsymbol{L}^{r_i})^j \right)_{1 \le i,j \le z} \neq 0$ for all $0 \le r_1 < r_2 < \cdots < r_z \le 14$, which we refer to Supplementary Material F. $\square$

Construction of L**.** One simple way to construct $\boldsymbol{L}$ is to take $\boldsymbol{L}$ to be the companion matrix of a primitive polynomial. For example, for the primitive polynomial $\lambda^4 + \lambda + 1$, we can take $\boldsymbol{L}$ to be the companion matrix

$$\boldsymbol{L} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}. \tag{21}$$

It can be readily checked that the characteristic polynomial $p_{\boldsymbol{L}}(\lambda) = \lambda^4 + \lambda + 1$ and thus the eigenvalues of $\boldsymbol{L}$ are distinct primitive elements in $GF(2^4)$. On the other hand, taking companion matrices of primitive polynomials is not the only way to obtain $\boldsymbol{L}$. In fact, we perform an exhaustive search of all binary 4×4 binary matrices, and find totally 1344 distinct $\boldsymbol{L}$ whose characteristic polynomial is primitive over $GF(2)$.

An Example for $z = 4$. Taking $\alpha = -2$ and $\boldsymbol{L}$ equals to that in (21), then

$$\{\boldsymbol{L}_i\}_{1 \le i \le 4} = \{\boldsymbol{L}^{-1}, \boldsymbol{L}^0, \boldsymbol{L}^1, \boldsymbol{L}^2\}. \tag{22}$$

Without loss of generality, let

$$\boldsymbol{L}_2 = \boldsymbol{L}^1 = \begin{pmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,1\,0\,0 \end{pmatrix}, \boldsymbol{L}_3 = \boldsymbol{L}^{-1} = \begin{pmatrix} 1\,0\,0\,1 \\ 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \end{pmatrix}, \boldsymbol{L}_4 = \boldsymbol{L}^2 = \begin{pmatrix} 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,1\,0\,0 \\ 0\,1\,1\,0 \end{pmatrix}. \tag{23}$$

Considering the LFSRs defined by (23), we found that the LFSRs for TK_2 and TK_3 coincide with the original LFSRs in SKINNY, and the LFSRs for TK_4 coincides with that constructed in SKINNYe-64-256 v2. *From this point of view, our construction can be viewed as a natural extension of the original SKINNY-64 and SKINNYe-64-256 v2.*

For general $z \le 14$, the 'block-MDS' property of our construction guarantees there are at most $z-1$ difference cancellations every 15 rounds for TK-z (or every 30 rounds SKINNY-n-zn). We derive the lower bounds of the number of active S-boxes for SKINNY-n-zn ($z \le 14$) with our construction of the tweakey schedule (see Supplementary Material G). The results (see Table 19) show that our new tweakey schedule for TK-z ($z \le 14$) leads to a natural increase of the bounds compared to TK-1, TK-2 and TK-3 in [10] and TK-4 in [50].

Efficiency Considerations. How to choose $\boldsymbol{L}_i$'s to optimize the implementation efficiency is also an important issue. As pointed in [49], one direction of optimization is to minimize the total number of XORs required by the LFSRs. For $z = 4$, the LFSRs constructed through (23) require only 4 XORs totally, i.e., $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$ require only 1 XOR respectively, and $\boldsymbol{L}_4$ requires 2 XORs. Note that in [49] it was proved that there is no secure LFSRs for TK_4 with only a single XOR, therefore the LFSRs constructed through (23) is optimal with respect to the number of XORs. For all $4 \le z \le 7$, we enumerate all possible $\boldsymbol{L}$ and α, and give the optimal number of XORs required in our construction in Table 8.

Another direction of optimization is to minimize the circuit area of the LFSRs. In our construction, all $\boldsymbol{L}_i$'s are powers of a matrix $\boldsymbol{L}$, therefore a minimal area implementation can be supported by instantiating only one circuit of $\boldsymbol{L}$ and computing each $\boldsymbol{L}_i$ iteratively. For example, for $z = 4$ we take $\alpha = -1$ and $\boldsymbol{L}_2 = \boldsymbol{L}, \boldsymbol{L}_3 = \boldsymbol{L}^2, \boldsymbol{L}_4 = \boldsymbol{L}^3$. Then $\boldsymbol{L}_2, \boldsymbol{L}_3$ and $\boldsymbol{L}_4$ can be computed by repeating $\boldsymbol{L}$ in $1, 2$ and 3 times respectively, and the total latency is as 6 times as that of a single $\boldsymbol{L}$. On the other hand, we propose an area-latency trade-off to reduce the latency by slightly increasing the area. Again we take $z = 4$ and $\boldsymbol{L}_2 = \boldsymbol{L}, \boldsymbol{L}_3 = \boldsymbol{L}^2$ and $\boldsymbol{L}_4 = \boldsymbol{L}^3$ for an example. In this case we instantiate a circuit of $\boldsymbol{L}$ and a circuit of $\boldsymbol{L}^2$. Then taking $\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4 \in GF(2)^4$ as inputs,

Table 8: Optimal number of XORs required in our construction.

z	$\boldsymbol{L}$	$\{\boldsymbol{L}_i\}_{2\le i\le z}$	Number of XORs	Total XORs
4	$\begin{pmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&1&0&0 \end{pmatrix}$	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2}\}$	$\{1,1,2\}$	4
5	$\begin{pmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&1&0&0 \end{pmatrix}$	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2},\boldsymbol{L}^{-2}\}$	$\{1,1,2,3\}$	7
6	$\begin{pmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&1&0&0 \end{pmatrix}$	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2},\boldsymbol{L}^{-2},\boldsymbol{L}^{3}\}$	$\{1,1,2,3,3\}$	10
7	$\begin{pmatrix} 0&1&0&0\\ 0&0&1&0\\ 0&0&0&1\\ 1&1&0&0 \end{pmatrix}$	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2},\boldsymbol{L}^{-2},\boldsymbol{L}^{3},\boldsymbol{L}^{4}\}$	$\{1,1,2,3,3,5\}$	15

the output states $\boldsymbol{L}_2\boldsymbol{x}_2, \boldsymbol{L}_3\boldsymbol{x}_3, \boldsymbol{L}_4\boldsymbol{x}_4$ can be computed in two steps, i.e., firstly compute $\boldsymbol{L}\boldsymbol{x}_2$ and $\boldsymbol{L}^2\boldsymbol{x}_4$, then compute $\boldsymbol{L}(\boldsymbol{L}^2\boldsymbol{x}_4)$ and $\boldsymbol{L}^2\boldsymbol{x}_3$. As a result, the total latency is reduced by a third at the cost of double area[5]. In Table 9 we list the area-latency trade-off for our construction for $4 \le z \le 7$.

Table 9: The area-latency trade-off for our construction.

z	$\{\boldsymbol{L}_i\}_{2\le i\le z}$	Instantiated circuit	Area	Latency
4	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3}\}$	$\{\boldsymbol{L}\}$	1	6
	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{2}\}$	2	2
5	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4}\}$	$\{\boldsymbol{L}\}$	1	10
	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2},\boldsymbol{L}^{-2}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{-1}\}$	2	3
	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3}\}$	3	2
6	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4},\boldsymbol{L}^{5}\}$	$\{\boldsymbol{L}\}$	1	15
	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4},\boldsymbol{L}^{5}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{2}\}$	2	4
	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4},\boldsymbol{L}^{5}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{4}\}$	3	3
	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4},\boldsymbol{L}^{5}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4}\}$	4	2
7	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4},\boldsymbol{L}^{5},\boldsymbol{L}^{6}\}$	$\{\boldsymbol{L}\}$	1	21
	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2},\boldsymbol{L}^{-2},\boldsymbol{L}^{3},\boldsymbol{L}^{-3}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{-1}\}$	2	6
	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{3},\boldsymbol{L}^{4},\boldsymbol{L}^{5},\boldsymbol{L}^{6}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{2},\boldsymbol{L}^{4}\}$	3	3
	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2},\boldsymbol{L}^{-2},\boldsymbol{L}^{3},\boldsymbol{L}^{-3}\}$	$\{\boldsymbol{L},\boldsymbol{L}^{-1},\boldsymbol{L}^{2},\boldsymbol{L}^{-2}\}$	4	2

A more scalable construction. Our construction can be naturally extended to choose $c \times c$ ($c \ge 4$) matrices $\boldsymbol{L}_i$'s such that the 'block-MDS' property in (17) is satisfied. For large c, it is not always essential for the index r_i's in (17) reaching up to $2^c - 2$. For example, SKINNY uses cheap 8-bit LFSRs (i.e., $c = 8$) that only needs a single XOR to reduce the implementation cost, and therefore the 'block-MDS' property in (17) can be guaranteed for r_i's up to 14 [11]. To cover this consideration, in the following we present a scalable construction that addresses general $c \times c$ sub-matrices and a general upper constraint r on the r_i's.

Let $c \ge 4$ and $z \le r \le 2^c - 2$ be three positive integers. Again we choose the $\boldsymbol{L}_i$'s to be consecutive powers of a $c \times c$ matrix $\boldsymbol{L}$ such that $\boldsymbol{L}_1 = \boldsymbol{I}$, i.e., $\{\boldsymbol{L}_i\}_{1\le i\le z} = \{\boldsymbol{L}^{\alpha+1}, \cdots, \boldsymbol{L}^{\alpha+z}\}$ for some integer $\alpha \in [-z, -1]$. Then the fol-

[5] The area of the trade-off implementation mainly includes the circuit for $\boldsymbol{L}$ and $\boldsymbol{L}^2$ and two 4-bit registers. In area optimization implementation, the area is the circuit of $\boldsymbol{L}$ and one 4-bit register. Assume the registers bound the area, we can say trade-off method costs double area.

lowing property can be proved similar to Proposition 1 by making a weaker assumption on the matrix $\boldsymbol{L}$.

Proposition 2. *Suppose $\boldsymbol{L}$ is a $c \times c$ matrix over $GF(2)$ such that its eigenvalues $\{\lambda_i\}_{1 \le i \le c}$ satisfy $\lambda_i \neq 0$ and $\lambda_i^j \neq 1$ for $1 \le i \le c, 1 \le j \le r$. Then for any integer α, it has*

$$\det \begin{pmatrix} (\boldsymbol{L}^{\alpha+1})^{r_1} & (\boldsymbol{L}^{\alpha+2})^{r_1} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_1} \\ (\boldsymbol{L}^{\alpha+1})^{r_2} & (\boldsymbol{L}^{\alpha+2})^{r_2} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_2} \\ \vdots & \vdots & \ddots & \vdots \\ (\boldsymbol{L}^{\alpha+1})^{r_z} & (\boldsymbol{L}^{\alpha+2})^{r_z} & \cdots & (\boldsymbol{L}^{\alpha+z})^{r_z} \end{pmatrix} \neq 0 \tag{24}$$

for all $0 \le r_1 < r_2 < \cdots < r_z \le r$.

Proof. The proof is similar to the determinant statement in Proposition 1, which we refer to Supplementary Material F. □

If we choose $c \times c$ matrix $\boldsymbol{L}$ as in Proposition 1 such that its characteristic polynomial is a primitive polynomial of degree c over $GF(2)$, then clearly the requirements of $\boldsymbol{L}$ in Proposition 2 can be met. Thus Proposition 2 makes a weaker assumption on $\boldsymbol{L}$ due to the relaxation of the constraint r. In the following, we focus on the specific choice of $\boldsymbol{L}$ for $c = 8$ (i.e., 8-bit LFSR).

Table 10: The choices of $\boldsymbol{L}$ for $c = 8$ and $r = 254$. The $\boldsymbol{L}$'s are found by enumerating all 8×8 matrices that require small number of XOR's for the update.

z	$\boldsymbol{L}$	$\{\boldsymbol{L}_i\}_{2 \le i \le z}$	Number of XORs	Total XORs
3	$\begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 1&0&0&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}\}$	$\{2, 2\}$	4
4	$\begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 1&0&0&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2\}$	$\{2, 2, 4\}$	8
5	$\begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 1&0&0&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2, \boldsymbol{L}^{-2}\}$	$\{2, 2, 4, 5\}$	13
6	$\begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 1&0&0&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2, \boldsymbol{L}^{-2}, \boldsymbol{L}^3\}$	$\{2, 2, 4, 5, 7\}$	20
7	$\begin{pmatrix} 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 1&0&0&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2, \boldsymbol{L}^{-2}, \boldsymbol{L}^3, \boldsymbol{L}^{-3}\}$	$\{2, 2, 4, 5, 7, 9\}$	29

Construction of 8×8 matrix $\boldsymbol{L}$. For $c = 8$, taking $\boldsymbol{L}$ to be the companion matrix of a primitive polynomial may still be the simplest way to meet the conditions in Proposition 2. However, a primitive polynomial of degree 8 over $GF(2)$ has at least 5 nonzero terms[6], thus any $\boldsymbol{L}$ constructed via this approach needs at least 4 `XOR`'s for the update. In fact, it is possible to find $\boldsymbol{L}$ with primitive characteristic polynomial which needs less `XOR`'s for the update. In Table 10 we list the choice of such $\boldsymbol{L}$ for $3 \leq z \leq 7$. The characteristic polynomial of the $\boldsymbol{L}$ in Table 10 is $\lambda^8+\lambda^6+\lambda^5+\lambda^3+1$, which is a primitive polynomial over $GF(2)$. Thus the 'block-MDS' property in (17) can be satisfied for all $0 \leq r_1 < r_2 < \cdots < r_z \leq 2^c - 2 = 254$.

On the other hand, if we are interested in r that is strictly less than $2^c - 2$, it is possible to find $\boldsymbol{L}$ satisfying the conditions in Proposition 2 which requires even less `XOR`'s for the update. For example, for $r = 14$ (adopted by `SKINNY` [10]), we give in Table 11 the choices of $\boldsymbol{L}$ for $3 \leq z \leq 7$ such that the number of `XOR`'s are optimized. We note that in Table 11, the characteristic polynomial of the $\boldsymbol{L}$ is $\lambda^8+\lambda^2+1$ which is not primitive. Nevertheless, $\boldsymbol{L}$ can still meet the conditions in Proposition 2 for $r = 14$, and thus the 'block-MDS' property is satisfied for $0 \leq r_1 < r_2 < \cdots < r_z \leq r = 14$. Besides, it can be easily checked that the LFSR's for $z = 3$ coincides with the 8-bit LFSR constructed in `SKINNY` [10]. From this perspective, Proposition 2 can provide a theoretic support for the construction of the the 8-bit LFSR in `SKINNY`.

Table 11: The choices of $\boldsymbol{L}$ for $c = 8, r = 14$. The $\boldsymbol{L}$'s are found by enumerating all 8×8 matrices that require small number of `XOR`'s.

z	$\boldsymbol{L}$	$\{\boldsymbol{L}_i\}_{2\leq i\leq z}$	Number of `XOR`s	Total `XOR`s
3	$\begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&1&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}\}$	$\{1,1\}$	2
4	$\begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&1&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2\}$	$\{1,1,2\}$	4
5	$\begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&1&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2, \boldsymbol{L}^{-2}\}$	$\{1,1,2,2\}$	6
6	$\begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&1&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2, \boldsymbol{L}^{-2}, \boldsymbol{L}^3\}$	$\{1,1,2,2,3\}$	9
7	$\begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1\\ 1&0&1&0&0&0&0&0 \end{pmatrix}$	$\{\boldsymbol{L}, \boldsymbol{L}^{-1}, \boldsymbol{L}^2, \boldsymbol{L}^{-2}, \boldsymbol{L}^3, \boldsymbol{L}^{-3}\}$	$\{1,1,2,2,3,4\}$	13

[6] In Table 18 of Supplementary Material F we listed all primitive polynomials of degree 8 over $GF(2)$.

For the area-latency trade-off, the scalable construction still utilizes consecutive powers of a matrix $\boldsymbol{L}$, and thus Table 9 is applicable to general c.

8 Conclusion

The unexpected cancellations in the new tweakey schedule of SKINNYe-64-256 significantly enhances several attacks on SKINNYe-64-256 when compared to that on SKINNY-64-128 and SKINNY-64-192, and leaves a security margin of 3 rounds in related-tweakey setting. Moreover, we give some cryptanalysis results on the updated version 2, which indicates that the current version satisfies the security claims of the designers. At last, we introduce a uniformed design strategy for the tweakey schedule of SKINNY-n-zn ($z \leq 14$), and prove that it satisfies the security requirements of the STK construction.

References

1. Alexandre Adomnicai, Thierry P. Berger, Christophe Clavier, Julien Francq, Paul Huynh, Virginie Lallemand, Kévin Le Gouguec, Marine Minier, Léo Reynaud, and Gëel Thomas. Lilliput-AE: a new lightweight tweakable block cipher for authenticated encryption with associated data. *Submission to NIST Lightweight Cryptography Project*, 2019.
2. Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT 2016, Proceedings, Part I*, volume 10031 of *LNCS*, pages 191–219.
3. Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT 2015, Proceedings, Part I*, volume 9056 of *LNCS*, pages 430–454.
4. Gianira N. Alfarano, Christof Beierle, Takanori Isobe, Stefan Kölbl, and Gregor Leander. Shiftrows alternatives for AES-like ciphers and optimal cell permutations for Midori and Skinny. *IACR Trans. Symmetric Cryptol.*, 2018(2):20–47, 2018.
5. Ralph Ankele, Subhadeep Banik, Avik Chakraborti, Eik List, Florian Mendel, Siang Meng Sim, and Gaoli Wang. Related-key impossible-differential attack on reduced-round SKINNY. In *ACNS 2017*, volume 10355, pages 208–228.
6. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In *SAC 2008*, volume 5381 of *LNCS*, pages 103–119.
7. 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.
8. Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic search of meet-in-the-middle preimage attacks on aes-like hashing. In *EUROCRYPT 2021, Part I*, volume 12696, pages 771–804.
9. Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. MITM meets guess-and-determine: Further improved preimage attacks against AES-like hashing. *IACR Cryptol. ePrint Arch.*, page 575, 2021.
10. 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.

11. Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. SKINNY-AEAD and SKINNY-Hash v1.0. *Submission to NIST Lightweight Cryptography Project*, 2019.
12. 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. Cryptology ePrint Archive, Report 2016/660, 2016.
13. Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. *IACR Trans. Symmetric Cryptol.*, 2019(1):5–45.
14. Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR ToSC*, 2020(S1):295–349.
15. Tim Beyne and Begül Bilgin. Uniform first-order threshold implementations. In *SAC 2016*, volume 10532 of *LNCS*, pages 79–98.
16. Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In *EUROCRYPT '99, Proceeding*, volume 1592 of *LNCS*, pages 12–23.
17. Eli Biham, Orr Dunkelman, and Nathan Keller. New cryptanalytic results on IDEA. In *ASIACRYPT 2006, Proceedings*, pages 412–427.
18. Eli Biham, Orr Dunkelman, and Nathan Keller. New results on boomerang and rectangle attacks. In *FSE 2002, Revised Papers*, volume 2365, pages 1–16.
19. Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack - rectangling the serpent. In *EUROCRYPT 2001, Proceeding*, volume 2045, pages 340–357.
20. Eli Biham, Orr Dunkelman, and Nathan Keller. Related-key boomerang and rectangle attacks. In *EUROCRYPT 2005, Proceedings*, volume 3494, pages 507–525.
21. Alex Biryukov and Dmitry Khovratovich. Related-key cryptanalysis of the full AES-192 and AES-256. In *ASIACRYPT 2009*, volume 5912, pages 1–18.
22. Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In *SAC 2010*, volume 6544 of *LNCS*, pages 229–240.
23. Christina Boura and Anne Canteaut. On the boomerang uniformity of cryptographic sboxes. *IACR Trans. Symmetric Cryptol.*, 2018(3):290–310.
24. Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In *CRYPTO 2013, Part I*, volume 8042, pages 222–240.
25. 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.
26. Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs application to SKINNY. *IACR ToSC*, 2020(4):104–129.
27. Patrick Derbez and Pierre-Alain Fouque. Automatic search of Meet-in-the-Middle and Impossible Differential attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016 Proceedings, Part II*, volume 9815, pages 157–184.
28. Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In *EUROCRYPT 2013*, volume 7881, pages 371–387.

29. Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *CRYPTO 2012*, volume 7417, pages 719–740.
30. Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - towards side-channel secure authenticated encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.
31. Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In *CRYPTO 2021, Proceedings, Part III*, volume 12827 of *LNCS*, pages 278–308.
32. Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. EUROCRYPT 2022.
33. Orr Dunkelman, Senyang Huang, Eran Lambooij, and Stav Perle. Single tweakey cryptanalysis of reduced-round SKINNY-64. In *CSCML 2020, Proceedings*, volume 12161, pages 1–17. Springer, 2020.
34. Orr Dunkelman, Nathan Keller, Eran Lambooij, and Yu Sasaki. A practical forgery attack on Lilliput-AE. *J. Cryptol.*, 33(3):910–916, 2020.
35. Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In *ASIACRYPT 2010, Proceedings*, pages 158–176.
36. 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.
37. Thomas Fuhr and Brice Minaud. Match box meet-in-the-middle attack against KATAN. In *FSE 2014*, volume 8540 of *LNCS*, pages 61–81.
38. Si Gao, Arnab Roy, and Elisabeth Oswald. Constructing TI-friendly substitution boxes using shift-invariant permutations. In *CT-RSA 2019, Proceedings*, volume 11405 of *LNCS*, pages 433–452.
39. Hosein Hadipour, Nasour Bagheri, and Ling Song. Improved rectangle attacks on SKINNY and CRAFT. *IACR Trans. Symmetric Cryptol.*, 2021(2):140–198.
40. Jialiang Hua, Tai Liu, Yulong Cui, Lingyue Qin, Xiaoyang Dong, and Huiyong Cui. Low-data cryptanalysis on SKINNY block cipher. *The Computer Journal*, 2022.
41. Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO 2003*, volume 2729, pages 463–481.
42. Takanori Isobe. A single-key attack on the full GOST block cipher. In *FSE 2011*, volume 6733 of *LNCS*, pages 290–305.
43. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In *ASIACRYPT 2014*, volume 8874, pages 274–288.
44. Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. Submission to CAESAR : Deoxys v1.41, October 2016.
45. John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified boomerang attacks against reduced-round MARS and Serpent. In *FSE 2000*, volume 1978, pages 75–93.
46. L. R. Knudsen. Deal - a 128-bit block cipher. *Complexity*, 1998.
47. Guozhen Liu, Mohona Ghosh, and Ling Song. Security analysis of SKINNY under related-tweakey settings. *IACR Trans. Symmetric Cryptol.*, 2017(3):37–72.
48. Bart Mennink. Beyond birthday bound secure fresh rekeying: Application to authenticated encryption. In *ASIACRYPT 2020*, volume 12491, pages 630–661.
49. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight authenticated encryption mode suitable for threshold implementation. In *EUROCRYPT 2020, Proceedings, Part II*, volume 12106 of *LNCS*, pages 705–735.
50. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight authenticated encryption mode suitable for threshold implementation. *Cryptol. ePrint Arch.*, 2020.

51. Yusuke Naito and Takeshi Sugawara. Lightweight authenticated encryption mode of operation for tweakable block ciphers. *IACR TCHES*, 2020(1):66–94, 2020.
52. Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS 2006, Proceedings*, volume 4307 of *LNCS*, pages 529–545.
53. Lingyue Qin, Xiaoyang Dong, Xiaoyun Wang, Keting Jia, and Yunwen Liu. Automated search oriented to key recovery on ciphers with linear key schedule applications to boomerangs in SKINNY and ForkSkinny. *IACR ToSC*, 2021(2):249–291.
54. Joseph J Rotman. *Advanced modern algebra*. American Mathematical Soc., 2010.
55. Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. *IACR ToSC*, 2018(3):124–162, 2018.
56. Yu Sasaki. Integer linear programming for three-subset meet-in-the-middle attacks: Application to GIFT. In *IWSEC 2018*, volume 11049, pages 227–243.
57. Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to whirlpool. In *FSE 2011*, volume 6733 of *LNCS*, pages 378–396.
58. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In *EUROCRYPT 2009*, volume 5479, pages 134–152.
59. André Schrottenloher and Marc Stevens. Simplified MITM modeling for permutations: New (quantum) attacks. *IACR Cryptol. ePrint Arch.*, page 189, 2022.
60. Ling Song, Xianrui Qin, and Lei Hu. Boomerang connectivity table revisited. application to SKINNY and AES. *IACR ToSC*, 2019(1):118–141.
61. Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. Impossible differential cryptanalysis of reduced-round SKINNY. In *AFRICACRYPT 2017, Proceedings*, volume 10239, pages 117–134.
62. David A. Wagner. The boomerang attack. In *FSE '99*, volume 1636, pages 156–170.
63. Haoyang Wang and Thomas Peyrin. Boomerang switch in multiple rounds. application to AES variants and Deoxys. *IACR ToSC*, 2019(1):142–169.

Supplementary Material

A The algorithm of computing the equivalence classes with non-full rank coefficient matrix for SKINNYe-64-256

The algorithm of computing the equivalence classes with non-full rank coefficient matrix for SKINNYe-64-256 is given in Algorithm 2.

Algorithm 2: Computing the equivalence classes with non-full rank coefficient matrix

---

Input: $\mathcal{K} = \{0, 1, 2, \cdots, 14\}$
$H \leftarrow [\,]$
for *all* $\{r_1, r_2, \cdots, r_t\} \subset \mathcal{K}$ `/* about` 2^{15} `iterations */`
do
    Compute the rank of $\boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}}$, $\quad Flag \leftarrow 0$
    if $rank(\boldsymbol{A}_{\{r_1, r_2, \cdots, r_t\}}) \neq \min\{4t, 16\}$ **then**
        for $0 \leq r' \leq 14$ **do**
            $B \leftarrow \{(r_1 + r') \bmod 15, (r_2 + r') \bmod 15, \cdots, (r_t + r') \bmod 15\}$
            if $B \in H$ **then**
                $Flag \leftarrow 1$
                break
            end
        end
    end
    if *Flag* = *0* **then**
        $H \leftarrow \{r_1, r_2, \cdots, r_t\}$
    end
end
Output: H

B Automatic Rectangle Attack by Dong *et al.* At EUROCRYPT 2022

This section gives the new rectangle attack framework in [32] and briefly introduces their automatic model.

B.1 Dong *et al.*'s Rectangle Attack Framework [32]

At EUROCRYPT 2022, Dong *et al.* [32] introduce a new related-key rectangle attack on ciphers with linear key schedule, which is listed in Algorithm 3.

Algorithm 3: Related-key rectangle attack with linear key-schedule [32]

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

B.2 Dong *et al.*'s Model to Determine the Optimal Distinguisher

Following the previous automatic models [53,26,39], Dong *et al.* introduced a uniform automatic model to search for the entire $(N_b + N_d + N_f)$ rounds of the new rectangle attack framework in Algorithm 3, by adding new constraints and new objective function. The notations used in the new constraints are listed below:

- X^u_r and X^l_r: the internal state before SubCells in round r of the upper and lower differentials
- W^l_r: the internal state before MixColumns in round r of the lower differential
- $\mathtt{DXU}[r][i]$: active cells in the internal states X^u_r $(0 \le r \le N_b + r_0 + r_m, 0 \le i \le 15)$
- $\mathtt{DXL}[r][i]$: active cells in the internal states X^l_r $(0 \le r \le r_m + r_1 + N_f, 0 \le i \le 15)$
- $\mathtt{KnownEnc}[r][i]$: the cells involved in the m_b-bit subtweakeys in the N_b extended rounds $(0 \le r \le N_b - 1, 0 \le i \le 15)$
- $\mathtt{DXFixed}[r][i]$: the cells with fixed differences in X^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{DWFixed}[r][i]$: the cells with fixed differences in W^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{DXFilter}[r][i]$: the cells can be used as filters in X^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{DWFilter}[r][i]$: the cells can be used as filters in W^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{DXisFilter}[r][i]$: the cells chosen as filters in X^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{DWisFilter}[r][i]$: the cells chosen as filters in W^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{DXGuess}[r][i]$: the cells need to know in the decryption from ciphertexts to the filters in X^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{DWGuess}[r][i]$: the cells need to know in the decryption from ciphertexts to the filters in W^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- $\mathtt{KnownDec}[r][i]$: the cells need to know in the decryption from ciphertext to the position of known δ in Y^l_r $(0 \le r \le N_f - 1, 0 \le i \le 15)$
- Adv: the advantage h
- x: the bit size of K_x in Algorithm 3

B.2.1 Modelling propagation of cells with known differences in E_f.

Since certain cells of the internal state with fixed differences can be used to filter quartets, the propagation of fixed differences are needed to be modelled in E_f. For the first extended round after the lower differential, the difference of each cell is fixed: $\forall\ 0 \le i \le 15, \mathtt{DXFixed}[0][i] = 1$.

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

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

The constraints on the impact of the MC operation on the internal state are given below: $\forall\ 0 \le r \le N_f - 2, 0 \le i \le 3$,

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

B.2.2 Modelling cells that could be used to filter quartets in E_f. In Algorithm 3, m'_f-bit k'_f of k_f involved in N_f extended rounds are guessed to obtain a $2h_f$-bit filter. For each cell in X^l_r, if the difference is nonzero and fixed, it can be chosen as a filter If the difference is fixed as zero, the cell is not a filter because it has been used as filter in W^l_r. The valid valuations of DXFixed, DXL and DXFilter are given in Table 12.

Table 12: All valid valuations of DXFixed, DXL and DXFilter for 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, $W^l_{N_f-1}$ can be computed from the ciphertexts, and the cells with fixed differences of $W^l_{N_f-1}$ can be used as filters, i.e., the $(n - r_f)$ inactive bits: $\forall\ 0 \le i \le 15, \texttt{DWFilter}[N_f - 1][i] = \texttt{DWFixed}[N_f - 1][i]$.

Since the N_f rounds is extended with probability 1 at the bottom of the distinguisher, then the differences of W^l_r are propagated to X^l_{r+1} with probability 1 with the MC operation, and there will be more cells of W^l_r with fixed differences than the cells of X^l_{r+1} with fixed differences. Hence, these extra cells with fixed differences in W^l_r can act as filters. The details and all valid valuations of DWFixed and DWFilter please refer to the full version of [32]. Note that DXFixed is only used as the intermediate variable to determine DWFilter, since DXFixed is fully determined by DWFixed. Denoting the sets of all possible valuations listed in Table 12 and Table 8 in the full version of [32] by $\mathbb{P}_i$ and $\mathbb{Q}_i$, there are

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

Then, $\forall\ 0 \le r \le N_f - 1, 0 \le i \le 15$, there have $\texttt{DXisFilter}[r][i] \le \texttt{DXFilter}[r][i]$ and $\texttt{DWisFilter}[r][i] \le \texttt{DWFilter}[r][i]$.

B.2.3 Modeling the guessed subtweakey cells in E_f for generating the quartets. For the round 0, only cells used to be filters in the internal state need to be known: $\forall\ 0 \leq i \leq 15, \texttt{DXGuess}[0][i] = \texttt{DXisFilter}[0][i]$.

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

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

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

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

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

B.2.4 Modelling the advantage h in the key-recovery attack. In Algorithm 3, the advantage h determines the exhaustive search time, where h should be smaller than the number of key counters, i.e. $h \leq m_b + m_f - x$. The x-bit guessed subkey should satisfy $x \leq m_b + m'_f$, and also determine the size of memory $2^{m_b + m_f - x}$ to store the key counters. So there needs a balance between x and h to achieve a low time and memory complexities. In the first round extended after the distinguisher, only the active cells need to be known: $\forall\ 0 \leq i \leq 15, \texttt{KnownDec}[0][i] = \texttt{DXL}[r_m + r_1][i]$.

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

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

So $\sum_{0 \leq r \leq N_f - 1,\ 0 \leq i \leq 7} \texttt{KnownDec}[r][i]$ indicates the m_f-bit key.

B.2.5 The objective function. The time complexities of the attack framework in Algorithm 3 involve three parts: **Time I (T_1)**, **Time II (T_2)** and **Time III (T_3)**.

The constraints for probability $\tilde{p}^2 t \tilde{q}^2$ of the boomerang distinguisher are same as [26], where DXU, DXL and DXU $\wedge$ DXL are on behalf of $\tilde{p}$, $\tilde{q}$ and t. KnownEnc is on behalf of m_b.To describe T_1, there is:

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

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.

For describing T_2 (let $\varepsilon = 1$), there is:

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

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

For T_3, there is $T_3 = c_{T_3} - \mathtt{Adv}$, where $c_{T_3} = \tilde{n}$ for SKINNY-n-$\tilde{n}$.

For the advantage h and x, there are:

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

So the uniformed objective is

$$\text{Minimize } obj,\ obj \ge T_1,\ obj \ge T_2,\ obj \ge T_2.$$

C The Boomerang Distinguishers and Rectangle Attack on SKINNYe-64-256 and its version 2

C.1 The details of boomerang distinguishers of SKINNYe-64-256 and its version 2

In this section, we give the 26-round related-tweakey boomerang distinguisher for SKINNYe-64-256 v2 in Table 13. The differentials of the two boomerangs for SKINNYe-64-256 and its version 2 are listed in Table 14 and Table 15.

Table 13: The 26-round RTK boomerang distinguisher for SKINNYe-64-256 v2.

$r_0 = 12$, $r_m = 6$, $r_1 = 8$, $\tilde{p} = 2^{-19.05}$, $\xi = 2^{-19.5}$, $\tilde{q} = 1$, $\tilde{p}^2\xi\tilde{q}^2 = 2^{-57.6}$
$\Delta TK_1 =$ `0, 6, 0, 2, 0, 0, 0, 0, 0, 0, 0, d, 0, 0, 0, 0`
$\Delta TK_2 =$ `0, 9, 0, 8, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0`
$\Delta TK_3 =$ `0, c, 0, b, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0`
$\Delta TK_4 =$ `0, a, 0, 9, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0`
$\Delta X^{(0)} =$ `0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0`
$\nabla TK_1 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2`
$\nabla TK_2 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6`
$\nabla TK_3 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9`
$\nabla TK_4 =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1`
$\nabla X^{(26)} =$ `0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0`

Especially, we also experimentally verify the probabilities of the middle part similar with [32,53]. The experiments use one computer equipped with one RTX 2080 Ti and the results are listed in Table 16. The source code of the experiments can refer to https://github.com/skinny64/Skinny64-256.

C.2 Rectangle Attack on 37-round SKINNYe-64-256 v2

We use the 26-round related-tweakey rectangle distinguisher for SKINNYe-64-256 v2 given in Table 13, whose probability is $2^{-n}\tilde{p}^2\xi\tilde{q}^2 = 2^{-64-57.6} = 2^{-121.6}$.

Adding 4-round E_b and 7-round E_f, we attack 37-round SKINNYe-64-256 v2 as illustrated in Figure 6, where $r_b = 12\cdot 4 = 48$, $m_b = 18\cdot 4 = 72$, $r_f = 16\cdot 4 = 64$ and $m_f = 40\cdot 4 = 160$. With $k'_f = \{STK^{(33)}_{1,4,5}, STK^{(34)}_{0,1,3,5-7}, STK^{(35)}_{0-7}, STK^{(36)}_{0-7}\}$ and $X^{(32)}_{10}\|W^{(32)}_{4,5,13}\|W^{(33)}_{7,9,13}$ as internal filters, we have $m'_f = 25\cdot 4 = 100$ and $h_f = 7\cdot 4 = 28$. For SKINNYe-64-256 v2, the LFSR used for $TK4$ is different from the one in SKINNYe-64-256. According to the property of $\tilde{\boldsymbol{A}}_{\{r_0,r_1,\cdots,r_{t-1}\}}$ analyzed in Section 3, we list the relations of the subtweakeys involved in E_b and E_f in Table 17.

In the data collection process, there is $y = \sqrt{s}\cdot 2^{n/2-r_b}/\sqrt{\tilde{p}^2\xi\tilde{q}^2} = \sqrt{s}\cdot 2^{12.8}$ structures and the data complexity is $\sqrt{s}\cdot 2^{n/2+2}/\sqrt{\tilde{p}^2\xi\tilde{q}^2} = \sqrt{s}\cdot 2^{62.8}$. The time complexity of generating quartets is $T_1 = \sqrt{s}\cdot|k_b\cup k'_f|\cdot 2^{n/2+1}/\sqrt{\tilde{p}^2\xi\tilde{q}^2} =$

Table 14: The differentials of the 30-round distinguisher for `SKINNYe-64-256`, where R12 to R16 denote $r_m = 5$-round middle part, u satisfies DDT[`0x4`][u] > 0 and DDT[u][`0x1`] > 0, and v and w satisfy DDT[`0x2`][v] > 0, DDT[v][w] > 0 and DDT[w][`0x1`] > 0.

	Upper differential	Lower differential
R0	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4 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 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 0,0,1,0,0,0,0,0	
R2-R10	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	
R11	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 0,0,0,0,0,0,0,1	
R12	0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0 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
R13-R15	`middle part`	`middle part`
R16	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0	*,0,0,0,0,*,*,*,*,0,*,0,0,0,0,* 2,0,0,0,0,2,2,2,2,0,2,0,0,0,0,2 0,0,0,0,0,0,0,0
R17		0,0,0,0,2,0,0,0,0,0,0,2,0,0,2,0 0,0,0,0,v,0,0,0,0,0,0,v,0,0,v,0 0,0,0,0,0,0,0,0
R18		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,w,0,0 0,0,0,0,0,0,0,0
R19		w,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 1,0,0,0,0,0,0,0
R20-R28		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
R29		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 0,0,0,1,0,0,0,0

Table 15: The differentials of the 26-round distinguisher for SKINNYe-64-256 v2, where R12 to R17 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	0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0 0,9,0,8,0,0,0,0,0,0,0,0,0,0,0,0 0,9,0,8,0,0,0,0	
R1-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,d,0,0,0,0,0	
R8	0,0,d,0,0,0,d,0,0,0,0,0,0,0,d,0 0,0,2,0,0,0,u,0,0,0,0,0,0,0,v,0 0,0,2,0,0,0,6,0	
R9	0,v,0,0,0,0,0,0,0,0,0,$u \oplus$ 0x6,0,0,0,0 0,4,0,0,0,0,0,0,0,0,0,4,0,0,0,0 0,0,0,0,4,0,0,0	
R10	0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0 0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0 0,0,0,0,e,2,0,0	
R11	0,0,0,0,0,0,0,0,0,0,e,0,0,0,0,0 0,0,0,0,0,0,0,0,0,0,9,0,0,0,0,0 0,0,0,0,0,0,9,0	
R12	0,0,0,9,0,0,0,0,0,0,0,0,0,0,0,9 0,0,0,*,0,0,0,0,0,0,0,0,0,0,0,* 0,0,0,4,0,0,c,0	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,0
R13-R16	middle part	middle part
R17	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- 0,0,0,0,0,0,0,8	0,*,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 0,8,0,0,0,0,0,0
R18-R24		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
R25		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 0,0,0,0,0,0,2,0

Table 16: Experiments on the middle part of boomerang distinguishers for SKINNYe-64-256 and its version 2.

Version	N_d	r_m	Probability ξ	Complexity	Time
SKINNYe-64-256	30	5	$2^{-30.95}$	2^{40}	8405s
SKINNYe-64-256 v2	26	6	$2^{-19.50}$	2^{34}	132s

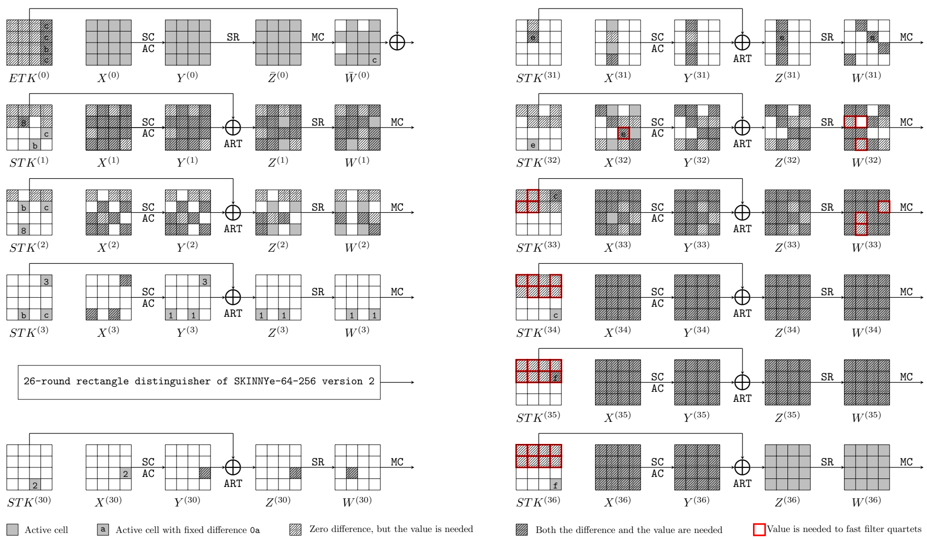

Fig. 6: The 37-round attack against SKINNYe-64-256 version 2.

$\sqrt{s} \cdot 2^{72+100+32+1+28.8} = \sqrt{s} \cdot 2^{233.8}$. We get $s \cdot |k_b \cup k'_f| \cdot 2^{-2h_f-n+2r_f}/(\tilde{p}^2\xi\tilde{q}^2) = s \cdot 2^{72+100-56-64+128+57.6} = s \cdot 2^{237.6}$ quartets. The memory complexity is $\sqrt{s} \cdot 2^{62.8} + |k_b \cup k_f|/2^x = \sqrt{s} \cdot 2^{62.8} + 2^{212-x}$. For each of the $s \cdot 2^{237.6}$ quartets, the key-recovery process is as follows, whose time complexity is ε:

1. **In round 34:** guessing 2^4 possible values of $STK_2^{(34)}$, we compute $Z_{3,15}^{(33)}$ together with guessed k'_f. Then $\Delta Y_3^{(33)}$ and $\Delta X_{15}^{(33)}$ are deduced. For the 4th column of $X^{(33)}$ of (C_1, C_3), we obtain $\Delta W_{15}^{(32)} = \Delta X_3^{(33)} \oplus \Delta X_{15}^{(33)} = 0$. Hence, we obtain $\Delta X_3^{(33)}$ and deduce $STK_3^{(33)}$ by Lemma 1. Similarly, we deduce $STK_3'^{(33)}$ for (C_2, C_4). $\Delta STK_3^{(33)}$ can act as a 4-bit filter. $s \cdot 2^{237.6} \cdot 2^4 \cdot 2^{-4} = s \cdot 2^{237.6}$ quartets remain.
2. Guessing 2^4 possible values of $STK_4^{(34)}$, we compute $Z_{33}[7]$ and peel off round 34, 35 and 36. Then $\Delta Y_7^{(33)}$ is deduced. For the 4th column of $X^{(30)}$ of (C_1, C_3), we obtain $\Delta W_{11}^{(32)} = \Delta X_7^{(33)} \oplus \Delta X_{15}^{(33)} = 0$, where $\Delta X_{15}^{(33)}$ is obtained in step 1. Hence, we obtain $\Delta X_7^{(33)}$ and deduce $STK_7^{(33)}$ by Lemma 1. Similarly, we deduce $STK_7'^{(33)}$ for (C_2, C_4). Then fixed $\Delta STK_7^{(33)}$ can act as a 4-bit filter. $s \cdot 2^{237.6} \cdot 2^4 \cdot 2^{-4} = s \cdot 2^{237.6}$ quartets remain.

Table 17: Relations of the subtweakeys involved in the 37-round attack on SKINNYe-64-256 v2, where the subtweakeys marked in bold are among k'_f.

i	k_b	k_f		
0	$ETK_0^{(0)}, STK_2^{(2)}$	$STK_0^{(32)}, STK_2^{(34)}, \boldsymbol{STK_4^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,1,1,2,3\}})\rvert = 2^{16}$	$\lvert Im(\tilde{A}_{\{0,1,3\}})\rvert = 2^{12}$
1	$ETK_1^{(0)}, STK_0^{(2)}$	$STK_1^{(32)}, \boldsymbol{STK_0^{(34)}}, \boldsymbol{STK_2^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,1,1,2,3\}})\rvert = 2^{16}$	$\lvert Im(\tilde{A}_{\{0,1,2,3\}})\rvert = 2^{16}$
2	$ETK_2^{(0)}$	$STK_4^{(34)}, \boldsymbol{STK_6^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{0,3\}})\rvert = 2^{8}$
3	$ETK_3^{(0)}$	$STK_3^{(32)}, \boldsymbol{STK_7^{(34)}}, \boldsymbol{STK_1^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,1,2,3\}})\rvert = 2^{16}$	$\lvert Im(\tilde{A}_{\{0,2,3\}})\rvert = 2^{12}$
4	$ETK_9^{(0)}$	$\boldsymbol{STK_6^{(34)}}, \boldsymbol{STK_5^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{0,2,3\}})\rvert = 2^{12}$
5	$ETK_{10}^{(0)}, STK_3^{(2)}$	$STK_5^{(32)}, \boldsymbol{STK_3^{(34)}}, \boldsymbol{STK_7^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,1,1,2,3\}})\rvert = 2^{16}$	$\lvert Im(\tilde{A}_{\{0,1,2,3\}})\rvert = 2^{16}$
6	$ETK_{11}^{(0)}$	$STK_6^{(32)}, \boldsymbol{STK_5^{(34)}}, \boldsymbol{STK_3^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,1,2,3\}})\rvert = 2^{16}$	$\lvert Im(\tilde{A}_{\{0,2,3\}})\rvert = 2^{12}$
7	$ETK_8^{(0)}$	$STK_7^{(32)}, \boldsymbol{STK_1^{(34)}}, \boldsymbol{STK_0^{(36)}}$	$\lvert Im(\tilde{A}_{\{0,1,2,3\}})\rvert = 2^{16}$	$\lvert Im(\tilde{A}_{\{0,2,3\}})\rvert = 2^{12}$
8	$STK_2^{(1)}$	$STK_2^{(33)}, \boldsymbol{STK_4^{(35)}}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{1,3\}})\rvert = 2^{8}$
9	$STK_0^{(1)}$	$STK_1^{(31)}, STK_0^{(33)}, \boldsymbol{STK_2^{(35)}}$	$\lvert Im(\tilde{A}_{\{1,1,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{1,3\}})\rvert = 2^{8}$
10	$STK_4^{(1)}$	$\boldsymbol{STK_4^{(33)}}, \boldsymbol{STK_6^{(35)}}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$
11	$STK_7^{(1)}$	$STK_7^{(33)}, \boldsymbol{STK_1^{(35)}}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{1,3\}})\rvert = 2^{8}$
12		$STK_6^{(33)}, \boldsymbol{STK_5^{(35)}}$	$\lvert Im(\tilde{A}_{\{2,3\}})\rvert = 2^{8}$	$\lvert Im(\tilde{A}_{\{3\}})\rvert = 2^{4}$
13	$STK_3^{(1)}$	$STK_5^{(31)}, STK_3^{(33)}, \boldsymbol{STK_7^{(35)}}$	$\lvert Im(\tilde{A}_{\{1,1,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{1,3\}})\rvert = 2^{8}$
14	$STK_5^{(1)}$	$\boldsymbol{STK_5^{(33)}}, \boldsymbol{STK_3^{(35)}}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$
15	$STK_1^{(1)}$	$\boldsymbol{STK_1^{(33)}}, \boldsymbol{STK_0^{(35)}}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$	$\lvert Im(\tilde{A}_{\{1,2,3\}})\rvert = 2^{12}$
			$\lvert k_b \cup k_f\rvert = 2^{212}$	$\lvert k_b \cup k'_f\rvert = 2^{172}$

3. **In round 33:** guessing 2^4 possible values of $STK_2^{(33)}$, we compute $Z_{3,11,15}^{(32)}$. Then $\Delta Y_3^{(32)}$ and $\Delta X_{11,15}^{(32)}$ are deduced. For the 4th column of $X^{(32)}$ of (C_1, C_3), we can obtain $\Delta X_3^{(32)} = \Delta X_{11}^{(32)} = \Delta X_{15}^{(32)}$. Hence, we obtain $\Delta X_3^{(32)}$ and deduce $STK_3^{(32)}$. Similarly, we deduce $STK_3'^{(32)}$ for (C_2, C_4). Then fixed $\Delta STK_3^{(32)}$ which is a 4-bit filter. For both (C_1, C_3) and (C_2, C_4), $\Delta X_{11}^{(32)} = \Delta X_{15}^{(32)}$ is a 4-bit filter. $s \cdot 2^{237.6} \cdot 2^4 \cdot 2^{-4} \cdot 2^{-4} \cdot 2^{-4} = s \cdot 2^{229.6}$ quartets remain.

4. Guessing 2^4 possible values of $STK_0^{(33)}$ and 2^4 possible values of $STK_6^{(33)}$, we compute $Z_{1,5,13}^{(32)}$ and peel off round 33. Since $STK_1^{(32)}$ and $STK_5^{(32)}$ has only one solution, we can compute $X_{1,5,13}^{(32)}$. For the 4th column of $X^{(32)}$, $\Delta X_1^{(32)} = \Delta X_5^{(32)} = \Delta X_{13}^{(32)}$ is a 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{229.6} \cdot 2^4 \cdot 2^4 \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{221.6}$ quartets remain.

5. **In round 32:** guessing 2^4 possible values of $STK_6^{(32)}$, 2^4 possible values of $STK_7^{(32)}$ and together with the one solution of $STK_0^{(32)}$, we compute $Z_{1,5,9,13}^{(31)}$ and peel off round 32. Then $\Delta X_{9,13}^{(31)}$ are deduced. Since there is only one solution of $STK_1^{(31)}$, we can compute $X_1^{(31)}$ and $\Delta X_1^{(31)}$. For the 2nd column of $X^{(31)}$, we can obtain $\Delta X_1^{(31)} = \Delta X_9^{(31)} = \Delta X_{13}^{(31)}$, which is a 8-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{221.6} \cdot 2^4 \cdot 2^4 \cdot 2^{-8} \cdot 2^{-8} = s \cdot 2^{213.6}$ quartets remain.

6. **In round 31:** With the only one solution of $STK_5^{(31)}$, we decrypt one round to get $X_{11}^{(30)}$. $\Delta X_{11}^{(30)} =$ `0x2` is a 4-bit filter for both (C_1, C_3) and (C_2, C_4). $s \cdot 2^{213.6} \cdot 2^{-8} = s \cdot 2^{205.6}$ quartets remain.

So for each quartet, $\varepsilon = 2^4 \cdot \frac{4}{37} + 2^4 \cdot \frac{4}{37} + 2^4 \cdot \frac{4}{37} + 2^{-8} \cdot 2^8 \cdot \frac{4}{37} + 2^{-16} \cdot 2^8 \cdot \frac{4}{37} + 2^{-24} \cdot \frac{4}{37} \approx 2^{2.41}$ and $T_2 = s \cdot 2^{237.6} \cdot \varepsilon = s \cdot 2^{240.01}$.

Set the excepted number of right quartets $s = 1$, the advantage $h = 24$ and $x = 168$ ($x \leq 172$, $h <= 212 - x$). Then we have $T_1 = 2^{233.8}$, $T_2 = 2^{240.01}$ and $T_3 = 2^{232}$. In total, the data complexity is $2^{62.8}$, the memory complexity is $2^{62.8}$, and the time complexity is $2^{240.03}$. The success probability is about 66.3%.

D Automatic MITM on `SKINNYe-64-256`

D.1 Definitions and Symbols of MITM Attack

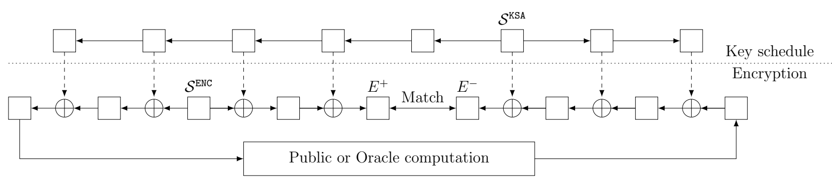

Fig. 7: A high-level overview of the MITM attacks [31]

The high-level overview of the MITM attacks introduced by Dong *et al.* is shown in Figure 7 and the notations are listed below.

- $\mathcal{S}^{\texttt{ENC}}$: starting state in the encryption data path (contains n w-bit cells)
- $\mathcal{S}^{\texttt{KSA}}$: starting state in the key schedule data path (contains $\bar{n}$ w-bit cells)
- E^+: ending state of the forward computation
- E^-: ending state of the backward computation
- $\mathcal{N}$: $\mathcal{N} = \{0, 1, \cdots, n-1\}$
- $\overline{\mathcal{N}}$: $\overline{\mathcal{N}} = \{0, 1, \cdots, \bar{n}-1\}$
- $\mathcal{B}^{\texttt{ENC}}$: subset of $\mathcal{N}$, index of Blue cells in $\mathcal{S}^{\texttt{ENC}}$, visualized by ■ cells.
- $\mathcal{B}^{\texttt{KSA}}$: subset of $\overline{\mathcal{N}}$, index of Blue cells in $\mathcal{S}^{\texttt{KSA}}$, visualized by ■ cells.
- $\mathcal{R}^{\texttt{ENC}}$: subset of $\mathcal{N}$, index of Red cells in $\mathcal{S}^{\texttt{ENC}}$, visualized by ■ cells.
- $\mathcal{R}^{\texttt{KSA}}$: subset of $\overline{\mathcal{N}}$, index of Red cells in $\mathcal{S}^{\texttt{KSA}}$, visualized by ■ cells.
- $\mathcal{G}^{\texttt{ENC}}$: subset of $\mathcal{N}$, index of Gray cells in $\mathcal{S}^{\texttt{ENC}}$, visualized by ■ cells.
- $\mathcal{G}^{\texttt{KSA}}$: subset of $\overline{\mathcal{N}}$, index of Gray cells in $\mathcal{S}^{\texttt{KSA}}$, visualized by ■ cells.
- $\mathcal{M}^+$: subset of $\mathcal{N}$, index of cells in E^+ which can be computed in the forward computation
- $\mathcal{M}^-$: subset of $\mathcal{N}$, index of cells in E^- which can be computed in the backward computation
- λ^+: $\lambda^+ = \mid \mathcal{B}^{\texttt{ENC}} \mid + \mid \mathcal{B}^{\texttt{KSA}} \mid$, the initial degrees of freedom for the forward computation, which is the number of ■ in $(\mathcal{S}^{\texttt{ENC}}, \mathcal{S}^{\texttt{KSA}})$.

- λ^-: $\lambda^- = \mid \mathcal{R}^{\texttt{ENC}} \mid + \mid \mathcal{R}^{\texttt{KSA}} \mid$, the initial degrees of freedom for the backward computation, which is the number of $\blacksquare$ in $(\mathcal{S}^{\texttt{ENC}}, \mathcal{S}^{\texttt{KSA}})$.
- $\lambda^+_{\texttt{ENC}}$: $\lambda^+_{\texttt{ENC}} = |\mathcal{B}^{\texttt{ENC}}|$, the initial degrees of freedom from the encryption data path for the forward computation.
- $\lambda^+_{\texttt{KSA}}$: $\lambda^+_{\texttt{KSA}} = |\mathcal{B}^{\texttt{KSA}}|$, the initial degrees of freedom from the key schedule data path for the forward computation.
- $\lambda^-_{\texttt{ENC}}$: $\lambda^-_{\texttt{ENC}} = |\mathcal{R}^{\texttt{ENC}}|$, the initial degrees of freedom from the encryption data path for the backward computation.
- $\lambda^-_{\texttt{KSA}}$: $\lambda^-_{\texttt{KSA}} = |\mathcal{R}^{\texttt{KSA}}|$, the initial degrees of freedom from the key schedule data path for the backward computation.
- DoM: the degrees of matching
- f_i^+: a function maps $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ to a word
- f_i^-: a function maps $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$ to a word
- $\boldsymbol{f}^+$: $\boldsymbol{f}^+ = (f_1^+, \cdots, f_{l^+}^+)$, l^+ constraints on the neutral words for the forward computation
- $\boldsymbol{f}^-$: $\boldsymbol{f}^- = (f_1^-, \cdots, f_{l^-}^-)$, l^- constraints on the neutral words for the backward computation
- DoF$^+$: the degrees of freedom for the forward computation
- DoF$^-$: the degrees of freedom for the backward computation

From $(\mathcal{S}^{\texttt{ENC}}, \mathcal{S}^{\texttt{KSA}})$ leading to E^+ and E^- are the forward and backward computations respectively. The cells of $(\mathcal{S}^{\texttt{ENC}}, \mathcal{S}^{\texttt{KSA}})$ are partitioned into different subsets with different meanings. The cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ and $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$ are the neutral words for the forward and backward computations respectively. The matching point is between E^+ and E^-, DoM $= m$ if $E^+[\mathcal{M}^+]$ and $E^-[\mathcal{M}^-]$ form an m-cell filter. If the values of $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}])$ are fixed, for any fixed $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathbb{F}_2^{w \cdot l^+}$ and $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathbb{F}_2^{w \cdot l^-}$, the neutral words $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ and $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$ fulfill the following systems of equations:

$$\begin{cases} f_1^+(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) = a_1 \\ f_2^+(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) = a_2 \\ \qquad\qquad \cdots \quad \cdots \\ f_{l^+}^+(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) = a_{l^+} \end{cases} \tag{25}$$

and

$$\begin{cases} f_1^-(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) = b_1 \\ f_2^-(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) = b_2 \\ \qquad\qquad \cdots \quad \cdots \\ f_{l^-}^-(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) = b_{l^-} \end{cases} \tag{26}$$

If there are $2^{w \cdot (\lambda^+ - l^+)}$ and $2^{w \cdot (\lambda^- - l^-)}$ solutions of Equation (25) and (26) respectively, we can get DoF$^+ = \lambda^+ - l^+$ and DoF$^- = \lambda^- - l^-$, which are the *degrees of freedom* for the forward and backward computations. The overall time complexity of MITM attack is

$$\begin{aligned} &(2^w)^{\overline{n} - \mathrm{DoF}^+ - \mathrm{DoF}^-}((2^w)^{\mathrm{DoF}^+} + (2^w)^{\mathrm{DoF}^-} + (2^w)^{\mathrm{DoF}^+ + \mathrm{DoF}^- - m}) \\ \approx &(2^w)^{\overline{n} - min\{\mathrm{DoF}^+, \mathrm{DoF}^-, m\}} \end{aligned} \tag{27}$$

In order to find the MITM key-recovery attacks, Dong *et al.* [31] mentioned that the degrees of freedom for the forward or backward computation from $\mathcal{S}^{\texttt{ENC}}$ should be used up while the degrees of freedom for the forward or backward computation from $\mathcal{S}^{\texttt{KSA}}$ cannot be depleted. That means the degrees of freedom from $\lambda^+_{\texttt{ENC}}$ and $\lambda^-_{\texttt{ENC}}$ must be consumed, so the remaining degrees of freedom are $\lambda^+_{\texttt{KSA}}$ and $\lambda^-_{\texttt{KSA}}$ for forward and backward computations respectively. If there are $l^+_{\texttt{KSA}}$ and $l^-_{\texttt{KSA}}$ degrees of freedom consumed from $\lambda^+_{\texttt{KSA}}$ and $\lambda^-_{\texttt{KSA}}$, then we can get $\mathrm{DoF}^+ = \lambda^+ - \lambda^+_{\texttt{ENC}} - l^+_{\texttt{KSA}} = \lambda^+_{\texttt{KSA}} - l^+_{\texttt{KSA}}$ and $\mathrm{DoF}^- = \lambda^- - \lambda^-_{\texttt{ENC}} - l^-_{\texttt{KSA}} = \lambda^-_{\texttt{KSA}} - l^-_{\texttt{KSA}}$.

D.2 Programming the MITM attacks on `SKINNYe-64-256` with MILP

Based on Dong *et al.*'s MILP model on `SKINNY-n-3n`, we briefly show how to build the MILP model for `SKINNYe-64-256`. We use the same coloring scheme, each cell of a state is represented by two 0-1 variables (x_i^S, y_i^S).

1. Gray: $(x_i^S, y_i^S) = (1, 1)$, predefined constant.
2. Red: $(x_i^S, y_i^S) = (0, 1)$, the values are determined by ■ cells and neutral words for backward computation.
3. Blue: $(x_i^S, y_i^S) = (1, 0)$, the values are determined by ■ cells and neutral words for forward computation.
4. White: $(x_i^S, y_i^S) = (1, 1)$, dependent on ■ cells in the backward computation or ■ cells in the forward computation.

For the starting states, we introduce variables α_i and β_i for each cell of $(\mathcal{S}^{\texttt{ENC}}, \mathcal{S}^{\texttt{KSA}})$, where $\alpha_i = 1$ if and only if the cell is ■ and $\beta_i = 1$ if and only if the cell is ■. Therefore, we can compute the initial degrees of freedom by $\lambda^+_{\texttt{ENC}} = \sum_i \alpha_i^{\texttt{ENC}}, \lambda^+_{\texttt{KSA}} = \sum_i \alpha_i^{\texttt{KSA}}, \lambda^-_{\texttt{ENC}} = \sum_i \beta_i^{\texttt{ENC}}, \lambda^-_{\texttt{KSA}} = \sum_i \beta_i^{\texttt{KSA}}$.

For the ending states, we assume the matching only happens at the `Mixcolumns`, for each pair of columns of E^+ and E^-, we introduce a variable m_i to indicate the degrees of matching in column i which can be constrained by the number of ■, ■ and ■ cells. The total degrees of matching DoM can be computed by $\mathrm{DoM} = \sum_{i=0}^{3} m_i$. For more details, we refer to [31].

Then attribute propagation rules for each operation of `SKINNY-64-256` need to be built, and the consumption of the degrees of freedom need to be recorded. The rules for XOR are shown in Figure 8. As mentioned in [31], we can use the rules of XOR to build the rules of `Mixcolumns` because it contains only XOR operation. `SubCells` and `AddConstants` do not change the color of input cell so we can ingore them. We can easily build the rules of `ShiftRows` because it just permutes the colors scheme of the input state. In order to build the rules for `AddRoundTweakey` of `SKINNY-n-3n`, Dong *et al.* introduced `3-XOR-RULE` to model the rules for three XORs as a whole which can avoid missing important coloring patterns. Similarly, we can build the rules for `AddRoundTweakey` of `SKINNYe-64-256` by introducing `4-XOR-RULE` which is the whole of four XORs. We show how to model the rule for `4-XOR`$^+$`-RULE`, which is the rule of `4-XOR-RULE` for the forward computation. We use $\delta^-_{\texttt{ENC}}$ and $\delta^-_{\texttt{KSA}}$ to denote the consumed degrees of freedom from $\lambda^-_{\texttt{ENC}}$ and $\lambda^-_{\texttt{KSA}}$ respectively, and we should consume $\lambda^-_{\texttt{ENC}}$

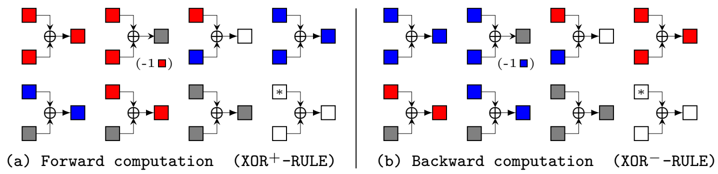

Fig. 8: Rules for XOR ("*" represents the cell can be any color)

first as far as possible. Some valid coloring patterns are shown in Figure 9. Then we need to convert the rule to linear inequalities. Let $\#a$, $\#b$, $\#c$, $\#d$, $\#e$ be the input cells and $\#f$ be the output cell. Then, the set of rules 4-XOR^{+}-RULE restricts $(x^{\#a}, y^{\#a}, x^{\#b}, y^{\#b}, x^{\#c}, y^{\#c}, x^{\#d}, y^{\#d}, x^{\#e}, y^{\#e}, x^{\#f}, y^{\#f}, \delta^{-}_{\texttt{ENC}}, \delta^{-}_{\texttt{KSA}})$ to subsets of $\mathbb{F}_2^{14}$, which can be described by a system of linear inequalities by using the convex hull computation method.Similarly, the rule of 4-XOR^{-}-RULE which is the rule of 4-XOR-RULE for the backward computation can be built by the same way.

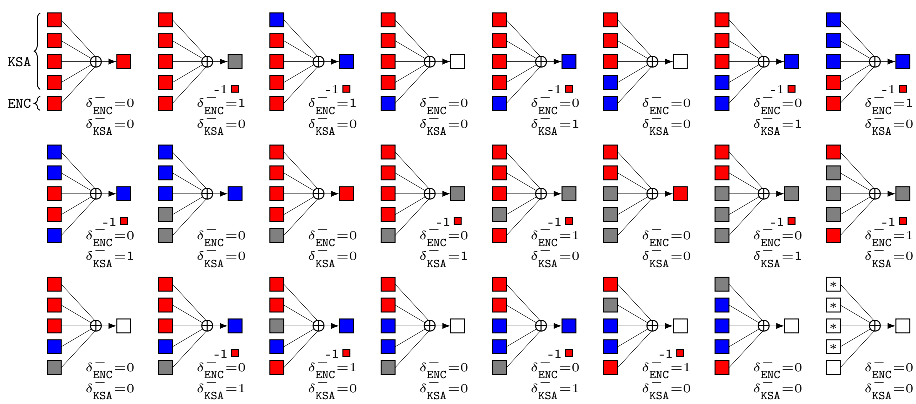

Fig. 9: 4-XOR^{+}-RULE ("*" represents the cell can be any color)

Let $l^{+}_{\texttt{ENC}}$, $l^{+}_{\texttt{KSA}}$ and $l^{-}_{\texttt{ENC}}$, $l^{-}_{\texttt{KSA}}$ be the accumulated degrees of freedom that have been consumed in the backward and forward computations in the encryption and key schedule data paths. We need to let $\lambda^{+}_{\texttt{ENC}} - l^{+}_{\texttt{ENC}} = 0, \lambda^{-}_{\texttt{ENC}} - l^{-}_{\texttt{ENC}} = 0$ to make sure that the degrees of freedom from the encryption data paths have been used up. Therefore, we can get $\mathrm{DoF}^{+} = \lambda^{+}_{\texttt{KSA}} - l^{+}_{\texttt{KSA}}, \mathrm{DoF}^{-} = \lambda^{-}_{\texttt{KSA}} - l^{-}_{\texttt{KSA}}$. Because the time complexity is given by Equation (27), we need to maximize the value of $min\{\mathrm{DoF}^{+}, \mathrm{DoF}^{-}, \mathrm{DoM}\}$ to find the optimal attacks. We introduce an auxiliary variable v_{obj} and impose the constraints

$$\{v_{obj} \leq \mathrm{DoF}^{+}, v_{obj} \leq \mathrm{DoF}^{-}, v_{obj} \leq \mathrm{DoM}\}. \tag{28}$$

D.3 MITM Attack on SKINNYe-64-256 and version 2

The 31-round MITM attack on SKINNYe-64-256 is given in Figure 10.

D.3.1 MITM attack on 27-round SKINNYe-64-256 v2 As mentioned in Section 3, the corresponding coefficient matrices of equivalence classes for SKINNYe-64-256 v2 are all full rank. So we can use Dong *et al.*'s method directly which adds constraints $\mathrm{DoF}^+ \geq 1, \mathrm{DoF}^- \geq 1$ to the MILP model to search attacks. Finally, we find a 27-round MITM attack as shown in Figure 11.

The starting states are $X^{(1)}$ and $(TK_1^{(1)}, TK_2^{(1)}, TK_3^{(1)}, TK_4^{(1)})$, and the matching is between $Z^{(14)}$ and $X^{(15)}$. We can get $\lambda^+_{\mathtt{KSA}} = 4, \lambda^-_{\mathtt{KSA}} = 4$ and DoM$=1$. For the forward computation, we require Equation (29) hold:

$$\tilde{\boldsymbol{A}}_{\{0,1,2\}} \cdot [\boldsymbol{tk}^{(6)}_{1,0}, \boldsymbol{tk}^{(6)}_{2,0}, \boldsymbol{tk}^{(6)}_{3,0}, \boldsymbol{tk}^{(6)}_{4,0}]^T = \mathfrak{c}^+. \tag{29}$$

Hence, $l^+_{\mathtt{KSA}} = 3$, and $\mathrm{DoF}^+ = 4 - 3 = 1$. For the backward computation, we require Equation (30) hold:

$$\tilde{\boldsymbol{A}}_{\{0,1,2\}} \cdot [\boldsymbol{tk}^{(21)}_{1,4}, \boldsymbol{tk}^{(21)}_{2,4}, \boldsymbol{tk}^{(21)}_{3,4}, \boldsymbol{tk}^{(21)}_{4,4}]^T = \mathfrak{c}^-. \tag{30}$$

Similarly, We can get $\mathrm{DoF}^- = 1$. The whole MITM attack on SKINNYe-64-256 v2 is shown in Algorithm 4. The time complexity is about 2^{252}, the data and memory complexity is 2^{52}.

E The Related-tweakey Impossible Differential for SKINNYe-64-256 and its version 2

The 21-round related-tweakey impossible differential of SKINNYe-64-256 is given in Figure 12.

For SKINNYe-64-256 v2, the subtweakey cancellations only can happen three times every 30 rounds. We find a 18-round related-tweakey impossible differential placed at Round 4 to Round 22 as shown in Figure 13. For the active 3-th nibble of the master tweakey, the subtweakey cancellations happen at Round 6, 8 and 10, where

$$\begin{pmatrix} \boldsymbol{stk}_0^{(6)} \\ \boldsymbol{stk}_2^{(8)} \\ \boldsymbol{stk}_4^{(10)} \end{pmatrix} = \tilde{\boldsymbol{A}}_{\{3,4,5\}} \cdot \begin{pmatrix} \boldsymbol{tk}_{1,3}^{(0)} \\ \boldsymbol{tk}_{2,3}^{(0)} \\ \boldsymbol{tk}_{3,3}^{(0)} \\ \boldsymbol{tk}_{4,3}^{(0)} \end{pmatrix} = \mathbf{0}.$$

With $rank(\tilde{\boldsymbol{A}}_{\{3,4,5\}}) = 12$, there are totally $(2^4 - 1) = 15$ nonzero solutions of $[\boldsymbol{tk}^{(0)}_{1,3}, \boldsymbol{tk}^{(0)}_{2,3}, \boldsymbol{tk}^{(0)}_{3,3}, \boldsymbol{tk}^{(0)}_{4,3}]$. Let $\Delta^{(r)}$ denote the difference of subtweakey $STK^{(r)}_{\bar{P}^{2r}[3]}$ in round r. The 18-round impossible differential can be represented as

$$(0\Delta^{(4)}\mathtt{00}\ \mathtt{0000}\ \mathtt{0000}\ \mathtt{0000}) \nrightarrow (\mathtt{0000}\ \mathtt{0000}\ \mathtt{00N0}\ \mathtt{0000}),$$

where $\Delta^{(4)}$ denotes a fixed non-zero difference and N denotes an arbitrary non-zero difference.

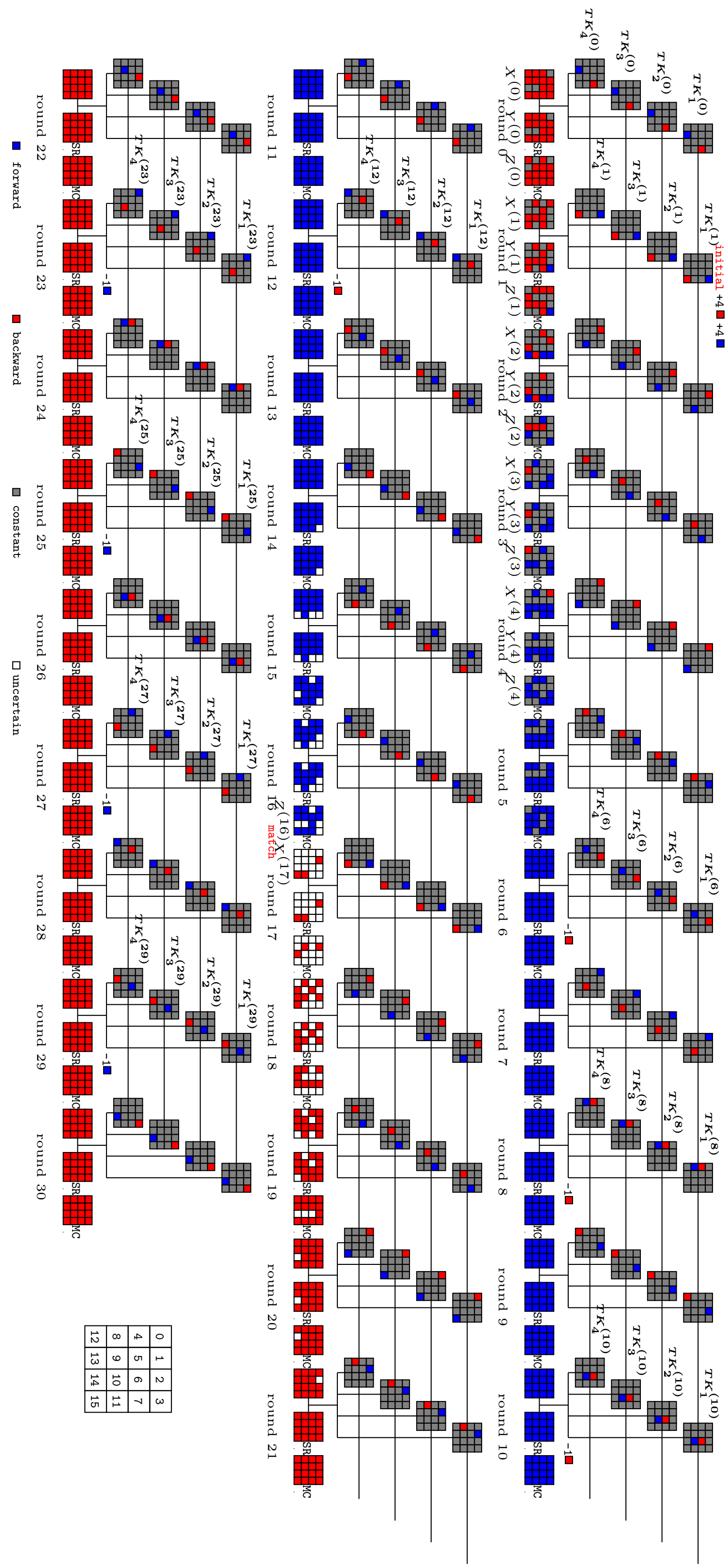

Fig. 10: The 31-round MITM attack on SKINNYe-64-256.

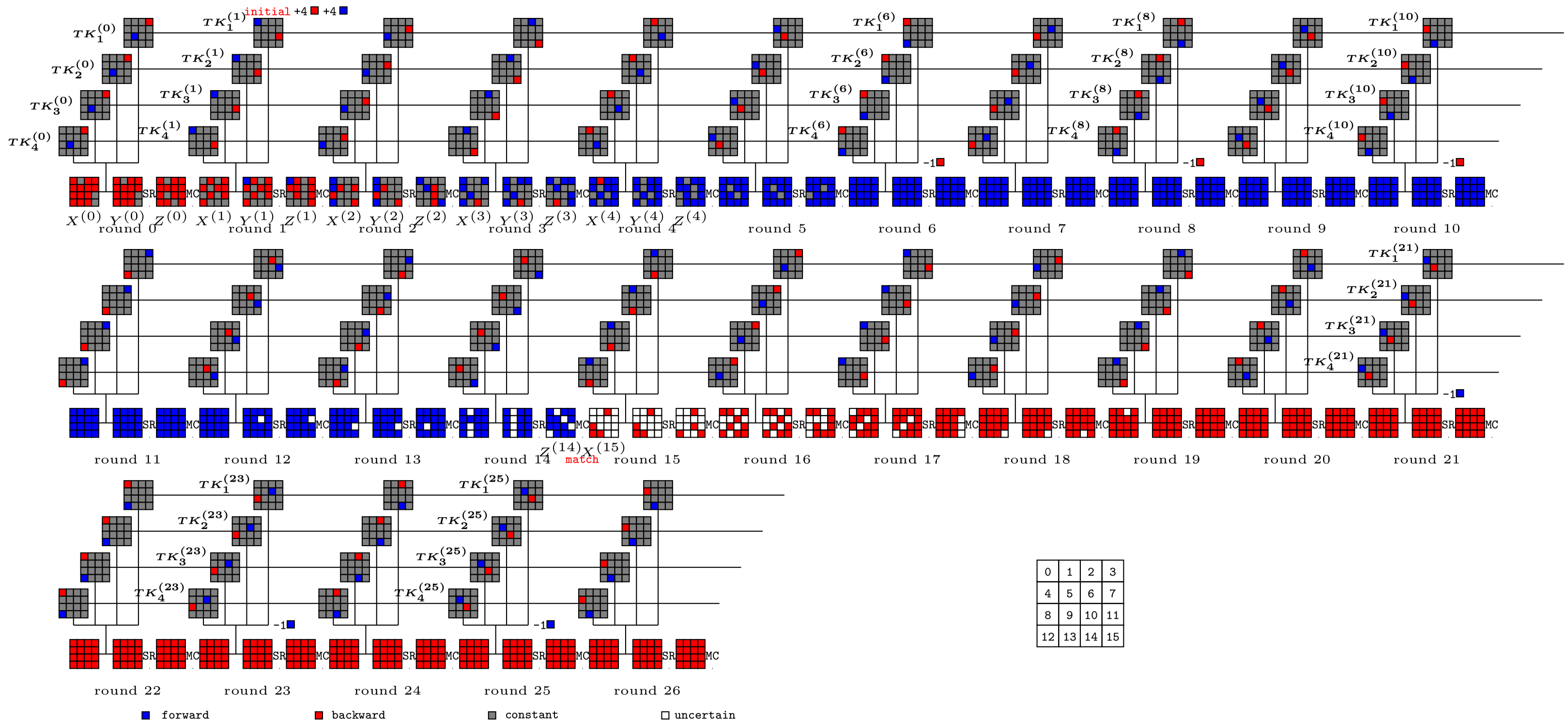

Fig. 11: The 27-round MITM attack on SKINNYe-64-256 v2.

Algorithm 4: The MITM key-recovery attack on SKINNYe-64-256 v2

```
1  X^(0)_{1,11,15} ← 0, X^(1)_{0,8,10,14,15} ← 0, X^(2)_{3,9,11,13} ← 0, Y^(2)_7 ← 0, X^(3)_{2,10} ← 0, Y^(4)_1 ← 0
2  Collecting structure of plaintext-ciphertext pairs and store them in table H,
     which traverses the non-constant 16-3=13 cells in the plaintext
3  for All possible values of the ■ (gray) cells in (TK^(1)_1, TK^(1)_2, TK^(1)_3, TK^(1)_4) do
4      Compute all other unknown Gray cells according to the values assigned in
         step 1 and step 3.
5      for (a_1, a_2, a_3, b_1, b_2, b_3) ∈ F_2^{6×4} do
6          Derive the solution space of the ■ (blue) cells by Equation (29) and store it
             in a table T_1.
7          Derive the solution space of the ■ (red) cells by Equation (30) and store it
             in a table T_2.
8          Initialize L to be an empty hash table.
9          for the values in T_2 do
10             Compute X^(15)_8 along the backward computation path:
                 X^(4) → X^(0) → E_K(X^(0)) → X^(15) by accessing H
11             Insert relative information into L indexed by X^(15)_8
12         end
13         for the values in T_1 do
14             Compute Z^(14)_4 and Z^(14)_8 along the forward computation path:
                 X^(1) → Z^(14)
15             for Candidate keys in L[Z^(14)_4 ⊕ Z^(14)_8] do
16                 Test the guessed key with several plaintext-ciphertext pairs
17             end
18         end
19     end
20 end
```

F The Proofs of Proposition 1 and Proposition 2

F.1 The Determinant of Block Vandermonde matrix

Lemma 2. *Suppose $\boldsymbol{L}_1, \boldsymbol{L}_2, \cdots, \boldsymbol{L}_n$ are pairwise commutable square matrices over a field $\mathbb{K}$. Then the determinant of the block Vandermonde matrix can be computed by*

$$\det \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{L}_1 & \boldsymbol{L}_2 & \cdots & \boldsymbol{L}_n \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{L}_1^{n-1} & \boldsymbol{L}_2^{n-1} & \cdots & \boldsymbol{L}_n^{n-1} \end{pmatrix} = \prod_{1 \le j < i \le n} \det(\boldsymbol{L}_i - \boldsymbol{L}_j). \tag{31}$$

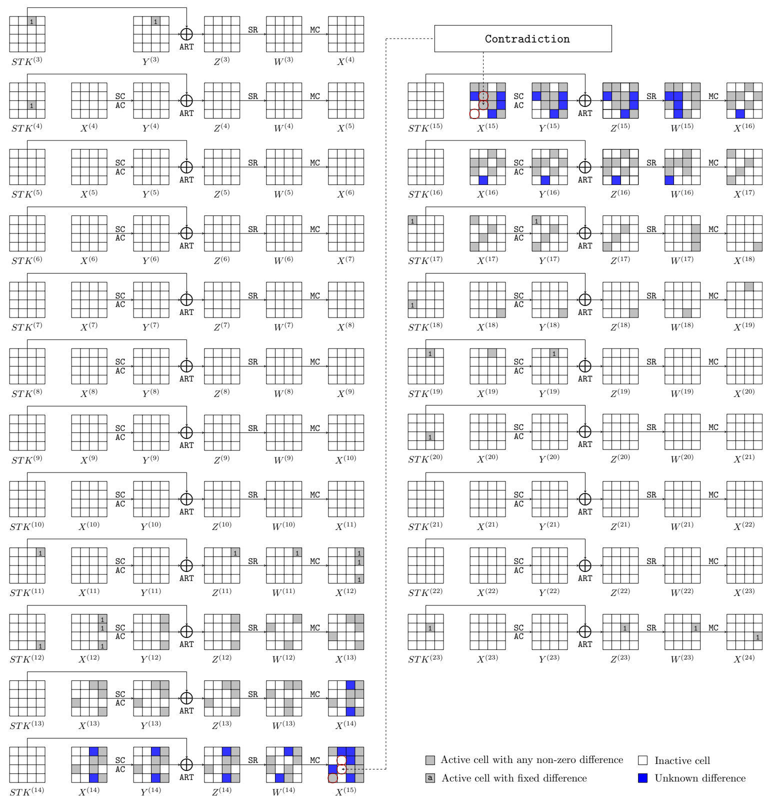

Fig. 12: The 21-round related-tweakey impossible differential of `SKINNYe-64-256`.

Proof. Observe that

$$\begin{pmatrix} \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{L}_1 & \cdots & \boldsymbol{L}_n \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}_1^{n-1} & \cdots & \boldsymbol{L}_n^{n-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & & \\ \boldsymbol{L}_n & \boldsymbol{I} & \\ \vdots & \vdots & \ddots \\ \boldsymbol{L}_n^{n-1} & \boldsymbol{L}_n^{n-2} & \cdots \boldsymbol{I} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{L}_2 - \boldsymbol{L}_1 & \cdots & \boldsymbol{L}_n - \boldsymbol{L}_1 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{L}_2^{n-1} - \boldsymbol{L}_2^{n-2}\boldsymbol{L}_1 & \cdots & \boldsymbol{L}_n^{n-1} - \boldsymbol{L}_n^{n-2}\boldsymbol{L}_1 \end{pmatrix}.$$

Thus it has

$$\det \begin{pmatrix} \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{L}_1 & \cdots & \boldsymbol{L}_n \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}_1^{n-1} & \cdots & \boldsymbol{L}_n^{n-1} \end{pmatrix} = \det \begin{pmatrix} \boldsymbol{L}_2 - \boldsymbol{L}_1 & \cdots & \boldsymbol{L}_n - \boldsymbol{L}_1 \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}_2^{n-1} - \boldsymbol{L}_2^{n-2}\boldsymbol{L}_1 & \cdots & \boldsymbol{L}_n^{n-1} - \boldsymbol{L}_n^{n-2}\boldsymbol{L}_1 \end{pmatrix}$$

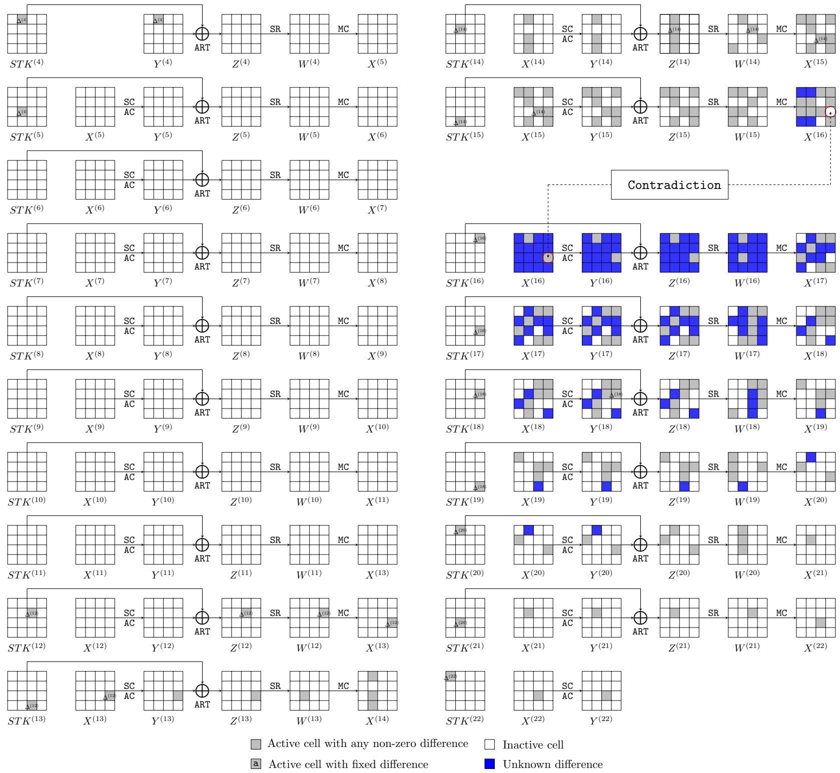

Fig. 13: The 18-round related-tweakey impossible differential of SKINNYe-64-256 v2

$$= \det \begin{pmatrix} \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{L}_2 & \cdots & \boldsymbol{L}_n \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}_2^{n-2} & \cdots & \boldsymbol{L}_n^{n-2} \end{pmatrix} \cdot \prod_{2 \leq j \leq n} \det(\boldsymbol{L}_j - \boldsymbol{L}_1).$$

Then the lemma follows directly by induction since

$$\det \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{L}_{n-1} & \boldsymbol{L}_{n-2} \end{pmatrix} = \det(\boldsymbol{L}_{n-1} - \boldsymbol{L}_{n-2}). \tag{32}$$

F.2 The Proofs of Proposition 1 and Proposition 2

The determinant statements in Proposition 1 and Proposition 2 follows directly from the following lemma. Let $c \geq 4$ and $0 < z \leq r \leq 2^c - 2$ be postive integers.

Lemma 3. *Suppose $\boldsymbol{L}$ is a $c \times c$ matrix over $GF(2)$ such that its eigenvalues $\{\lambda_i\}_{1\leq i\leq c}$ satisfy $\lambda_i \neq 0$ and $\lambda_i^j \neq 1$ for $1 \leq i \leq c, 1 \leq j \leq r$, then*

$$\det \begin{pmatrix} \boldsymbol{L}^{r_1} & \cdots & (\boldsymbol{L}^{r_1})^z \\ \boldsymbol{L}^{r_2} & \cdots & (\boldsymbol{L}^{r_2})^z \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}^{r_z} & \cdots & (\boldsymbol{L}^{r_z})^z \end{pmatrix} \neq 0 \tag{33}$$

for all $0 \leq r_1 < r_2 < \cdots < r_z \leq r$.

Proof. Using the formula in Lemma 2, we have

$$\det \begin{pmatrix} \boldsymbol{L}^{r_1} & \cdots & (\boldsymbol{L}^{r_1})^z \\ \boldsymbol{L}^{r_2} & \cdots & (\boldsymbol{L}^{r_2})^z \\ \vdots & \ddots & \vdots \\ \boldsymbol{L}^{r_z} & \cdots & (\boldsymbol{L}^{r_z})^z \end{pmatrix} = \prod_{1\leq j<i\leq z} \det(\boldsymbol{L}^{r_i} - \boldsymbol{L}^{r_j}). \tag{34}$$

Since the eigenvalues $\{\lambda_i\}_{1\leq i\leq c}$ satisfy $\lambda_i \neq 0$ and $\lambda_i^j \neq 1$ for $1 \leq i \leq c, 1 \leq j \leq r$, then it has $\det(\boldsymbol{L}) \neq 0$ and $\det(\boldsymbol{L}^t - \boldsymbol{I}) \neq 0$ for $1 \leq t \leq r$. Therefore we have

$$\det(\boldsymbol{L}^{r_i} - \boldsymbol{L}^{r_j}) = \det(\boldsymbol{L}^{r_j}) \cdot \det(\boldsymbol{L}^{r_i - r_j} - \boldsymbol{I}) \neq 0 \tag{35}$$

for all $0 \leq r_j < r_i \leq r$, and thus the lemma holds. □

F.3 List of Primitive Polynomials of degree 8

Table 18: Primitive polynomials of degree 8 over $GF(2)$.

Number of nonzero terms	Primitive Polynomials
5	$1+x^2+x^3+x^4+x^8, 1+x+x^3+x^5+x^8, 1+x^2+x^3+x^5+x^8$ $1+x^2+x^3+x^6+x^8, 1+x+x^5+x^6+x^8, 1+x^2+x^5+x^6+x^8$ $1+x^3+x^5+x^6+x^8, 1+x^4+x^5+x^6+x^8, 1+x+x^2+x^7+x^8$ $1+x^2+x^3+x^7+x^8, 1+x^3+x^5+x^7+x^8, 1+x+x^6+x^7+x^8$
7	$1+x+x^2+x^3+x^4+x^6+x^8, 1+x+x^2+x^3+x^6+x^7+x^8$ $1+x+x^2+x^5+x^6+x^7+x^8, 1+x^2+x^4+x^5+x^6+x^7+x^8$

G The Security Analysis of our New Tweakey Schedule for SKINNY Family

Lower bounds on the number of active Sboxes. The designers of `SKINNY` evaluated the tight bounds of the number of active S-boxes by MILP models in [10]. Then Alfarano *et al.* updated tight bounds of the number of active S-boxes for the single-tweakey setting [4]. The authors of `SKINNYe-64-256` and its version 2 extended the [10]'s model to derive the lower bounds in [49,50]. They also derived the tight bounds of the number of linear active Sboxes. Since the difference cancellations can happen more than 3 times for SKINNYe-64-256, we reevaluate the lower bounds of active S-boxes by extending the MILP model, which is similar with Sect 4.2. In addition, for SKINNY-n-zn with our proposition of the tweakey schedules ($z \leq 14$), we also use the MILP models by modifying the constraints of subtweakey difference cancellations to search the lower bounds of active S-boxes. All our results of the lower bounds of the number of active S-boxes are given in Table 19.

The results of these word-oriented models determines a lower bound on the number of differential active Sboxes. As pointed out by the designers in [10], if the Sbox can be chosen independently for every cell and every round, the bound is tight. However, in the related-tweakey settings, it may be hard to achieve the lower bounds as expected. So, the results we derived in Table 19 are only lower bounds and not tight, where the actual bounds of active Sboxes may be higher. We makes some experiments on the proved tight bounds in Table 20.

H Automatically Produced Figures of the Rectangle Attack on SKINNYe-64-256 and its version 2

Based on the open source of Delaune et al. [26], we automatically produce the figures including the key-recovery phase and the distinguishers for the rectangle attack on SKINNYe-64-256 and its version 2.

The full figure on 41-round key-recovery attack on SKINNYe-64-256 is shown in Figure 14.

The full figure on 37-round key-recovery attack on SKINNYe-64-256 v2 is shown in Figure 15.

We also refer the readers to https://github.com/skinny64/Skinny64-256/tree/main/article_boom/pic to see the larger figures.

Table 19: Lower bounds on the number of active Sboxes, where the bounds with upper bar are upper bounds. The bounds for SK[10], TK-1[10], TK-2[10], TK-3[10] and SK Lin[10] are given by the designers of SKINNY. SK[4] is updated by Alfarano *et al.*. TK-4 v2 [50] and SK Lin'[49] are given by the designers of SKINNYe-64-256 and its version 2. We reevaluate the bounds for TK-4 and search for TK-z ($5 \leq z \leq 14$).

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
SK[10]	1	2	5	8	12	16	26	36	41	46	51	55	58	61	66
TK-1[10]	0	0	1	2	3	6	10	13	16	23	32	38	41	45	49
TK-2[10]	0	0	0	0	1	2	3	6	9	12	16	21	25	31	35
TK-3[10]	0	0	0	0	0	0	1	2	3	6	10	13	16	19	24
TK-4	0	0	0	0	0	0	0	0	0	0	1	2	3	6	9
TK-4 v2 [50]	0	0	0	0	0	0	0	0	1	2	3	6	9	12	16
TK-5	0	0	0	0	0	0	0	0	0	0	1	2	3	6	9
TK-6	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3
TK-7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
TK-8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TK-9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TK-10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TK-11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TK-12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TK-13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TK-14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SK Lin[10]	1	2	5	8	13	19	25	32	38	43	48	52	55	58	64

	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
SK[10]	75	82	88	92	96	102	108	$\overline{114}$	$\overline{116}$	$\overline{124}$	$\overline{132}$	$\overline{138}$	$\overline{136}$	$\overline{148}$	$\overline{158}$
SK[4]	75	82	88	92	96	102	108	112	116	124	128	132	136	142	148
TK-1[10]	54	59	62	66	70	75	79	83	85	88	95	102	$\overline{108}$	$\overline{112}$	$\overline{120}$
TK-2[10]	40	43	47	52	57	59	64	67	72	75	82	85	88	92	96
TK-3[10]	27	31	35	43	45	48	51	55	58	60	65	72	77	81	85
TK-4	13	16	19	21	24	29	32	35	39	43	46	49	53	55	58
TK-4 v2 [50]	19	21	24	30	35	39	41	43	46	50	54	58	62	66	72
TK-5	13	16	19	21	24	29	32	35	39	43	46	49	53	55	58
TK-6	6	9	12	15	17	20	23	26	29	32	36	39	42	46	51
TK-7	2	3	6	9	10	13	17	20	22	25	28	30	34	37	40
TK-8	0	1	2	4	4	6	10	13	15	18	21	24	26	29	33
TK-9	0	0	0	1	2	3	4	6	8	11	14	17	21	22	24
TK-10	0	0	0	0	0	1	2	3	4	6	8	11	14	17	21
TK-11	0	0	0	0	0	0	0	1	2	3	4	6	8	11	14
TK-12	0	0	0	0	0	0	0	0	0	1	2	3	4	6	8
TK-13	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4
TK-14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2
SK Lin[10]	70	76	80	85	90	96	102	107	$\overline{110}$	$\overline{118}$	$\overline{122}$	$\overline{128}$	$\overline{136}$	$\overline{141}$	$\overline{143}$
SK Lin'[49]	70	76	80	85	90	96	102	107	110	115	121	127	130	135	141

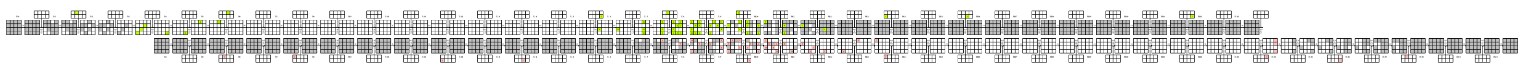

Fig. 14: 41-round rectangle attack on SKINNYe-64-256. We encourage the readers to https://github.com/skinny64/Skinny64-256/tree/main/article_boom/pic to see the larger figures.

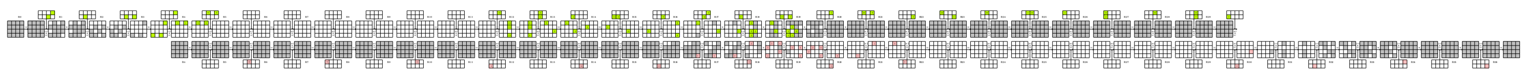

Fig. 15: 37-round rectangle attack on SKINNYe-64-256 v2. We encourage the readers to https://github.com/skinny64/Skinny64-256/tree/main/article_boom/pic to see the larger figures.

Table 20: Proved lower bounds on the number of active Sboxes, where $\boldsymbol{TK4}$ and $\boldsymbol{TK4\ v2}$ use the tweakey schedules in [49] and [50], where the bounds marked by bold have gaps with the results in Table 19.

	6	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
$\boldsymbol{TK4}$	0	0	0	0	0	1	2	3	6	**10**	**14**	**18**	**22**	**25**	**30**
$\boldsymbol{TK4\ v2}$	0	0	0	1	2	3	6	9	12	16	**21**	**26**	**29**	**34**	**37**