Related-Tweakey Impossible Differential Attack on Reduced-Round SKINNY-AEAD M1/M3

Yanhong Fan[1,2], Muzhou Li[1,2], Chao Niu[1,2], Zhenyu Lu[1,2], and Meiqin Wang[1,2(✉)]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, 266237, China
{fanyh, muzhouli, niuchao, luzhenyu}@mail.sdu.edu.cn, mqwang@sdu.edu.cn
[2] Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao, Shandong, 266237, China

Abstract. SKINNY-AEAD is one of the second-round candidates of the Lightweight Cryptography Standardization project held by NIST. SKINNY-AEAD M1 is the primary member of six SKINNY-AEAD schemes, while SKINNY-AEAD M3 is another member with a small tag. In the design document, only security analyses of their underlying primitive SKINNY-128-384 are provided. Besides, there are no valid third-party analyses on SKINNY-AEAD M1/M3 according to our knowledge. Therefore, this paper focuses on constructing the first third-party security analyses on them under a nonce-respecting scenario. By taking the encryption mode of SKINNY-AEAD into consideration and exploiting several properties of SKINNY, we can deduce some necessary constraints on the input and tweakey differences of related-tweakey impossible differential distinguishers. Under these constraints, we can find distinguishers suitable for mounting powerful tweakey recovery attacks. With the help of the automatic searching algorithms based on STP, we find some 14-round distinguishers. Based on one of these distinguishers, we mount a 20-round and an 18-round tweakey recovery attack on SKINNY-AEAD M1/M3. To the best of our knowledge, all these attacks are the best ones so far.

Keywords: Related-tweakey · Impossible differential cryptanalysis · SKINNY-AEAD M1/M3 · Tweakey recovery · SKINNY-128-384

1 Introduction

SKINNY-AEAD schemes are proposed by Beierle *et al.* [5] and accepted as one of the second-round candidates of the NIST Lightweight Cryptography (LWC) Standardization project[1]. There are six schemes for SKINNY-AEAD and SKINNY-AEAD M1 is the primary member. SKINNY-AEAD M3 is the same as M1 except that M3 has smaller length of tag than M1. These two authenticated encryption

[1] https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates

schemes M1/M3 use a mode following the ΘCB3 framework [13] and adopts SKINNY-128-384 [4] as their internal tweakable block cipher. For M1/M3, only security analyses of their underlying primitive SKINNY-128-384 are introduced in the design document [5]. As for M1/M3 themselves, no cryptanalytic results are presented there.

SKINNY-128-384 belongs to the well-known SKINNY family designed by Beierle *et al.* [4]. It adopts 128-bit plaintexts and 384-bit tweakeys. Since its proposal, many cryptanalytic results have been proposed [9,15,20,21,22]. Among all of them, the best attack so far was provided by [9], where a 30-round related-tweakey rectangle attack was mounted with time complexity $2^{361.38}$. As for impossible differential attacks, Liu *et al.* gave a 27-round attack in related-tweakey setting [15] with time complexity 2^{378}, while a 22-round attack in single-tweakey setting was proposed by Tolba *et al.* in [21] with time complexity $2^{373.48}$. Besides, there is a 22-round Demirci-Selçuk meet-in-the-middle attack provided in [20] with time complexity $2^{382.46}$.

From above security analyses of SKINNY-128-384, we can see that there are no constraints on the value or difference of the tweakey. However, when it comes to SKINNY-AEAD, some constraints should be considered in the attack procedure because of the encryption mode, the nonce-respecting scenario and complex tweakey initialization. In [5], the designers depict that SKINNY-AEAD M1/M3 apply a 384-bit tweakey but a 128-bit key, and claim full 128-bit security for key recovery in the nonce-respecting scenario. Thus, only attacks on SKINNY-128-384 with time complexity less than 2^{128} is valid for SKINNY-AEAD M1/M3. Due to this constraints on time complexity, above attacks [9,15,20,21,22] cannot be directly applied to SKINNY-AEAD M1/M3. Zhao *et al.* [22] gave a related-tweakey rectangle attack on SKINNY-AEAD M1. However, the second-round status update document of SKINNY-AEAD [1] pointed that the attack of [22] on SKINNY-AEAD M1 is invalid. To the best of our knowledge, there is no valid third-party analysis of SKINNY-AEAD M1/M3 up to now. This motivates us to evaluate their security considering these restrictions.

In this paper, we mount tweakey recovery attacks by searching for the related-tweakey impossible differential distinguishers. As one of the most popular cryptanalytic methods, the impossible differential attack was proposed in [7]. Biham [6] firstly proposed the related-key attack, where only the relations between pairs of related keys are chosen by the attacker, who does not know the keys themselves. Jakimoski *et al.* [10] proposed the related-key impossible differential attack firstly combining the two aforementioned attacks. In recent years, the related-key/tweakey impossible differential attacks [2,15,19] gave some better cryptanalysis results in block ciphers.

Our Contributions. The main contribution of our work is providing the third-party cryptanalytic results on SKINNY-AEAD M1/M3 by utilizing related-tweakey impossible differentials. All these distinguishers are found with the automatic solver STP after setting several constraints on the input and tweakey differences.

1) **Automatically Searching for Distinguishers of SKINNY-AEAD M1/M3**. Unlike finding distinguishers of SKINNY-128-384 itself, distinguishers exploited here should fulfill several conditions due to the encryption mode of SKINNY-AEAD M1/M3, as well as the complex tweakey initialization. By combining these restrictions with several properties of SKINNY cipher, we can deduce some constraints on the input and tweakey differences of the target related-tweakey impossible differential distinguishers. Based on the method of searching for related-tweakey impossible differentials [15,19], we find some 14-round distinguishers with only one active input and output state differences with the help of STP.
2) **Tweakey Recovery Attacks for SKINNY-AEAD M1/M3**. Based on one 14-round distinguisher, we mount a 20-round and an 18-round tweakey recovery attack under nonce-respecting scenario. All the attacks are applicable to M1 and M3. Our attack results along with [22] are illustrated in Table 1. To our knowledge, they are the best tweakey recovery attacks on SKINNY-AEAD M1/M3 so far.

Table 1. Summary of attack results on SKINNY-AEAD M1/M3

Attack	Cipher	Rounds	Data	Time	Memory	Ref.
RTID	M1/M3	20	$2^{121.6}$ CP	$2^{127.1}$	$2^{124.8}$ bits	Sect. 4
		18	$2^{121.6}$ CP	$2^{125.9}$	$2^{52.8}$ bits	Sect. 5
RTR^{+}	M1	24	$2^{123.0}$ CP	$2^{123.0}$	$2^{121.0}$	[22]

Note: RTID: Related-Tweakey Impossible Differential. CP: Chosen PlaintextS.
RTR: Related-Tweakey Rectangle. $^{+}$: This attack is invalid, as pointed out in [1].

Outline. In Section 2, we give the brief description of SKINNY-AEAD M1/M3, SKINNY-128-384 and properties of SKINNY, and then introduce some notations used in the paper. Section 3 gives the 14-round related-tweakey impossible differential distinguishers that are found by an automatic search algorithm with STP. Based on one of these 14-round distinguishers, Section 4 and 5 discuss a 20-round and an 18-round tweakey recovery attack, respectively. In Section 6, we summarize and conclude our work.

2 Preliminaries

2.1 Description of SKINNY-AEAD M1/M3

In this section, we briefly give some specifications for SKINNY-AEAD M1/M3. The input parameters of SKINNY-AEAD M1/M3 might contain an associated data A, a message M and a nonce N. In our attack scenario, A is set to be empty, and then the message block is handled directly, which can reduce the time complexity of

the tweakey recovery attack. The output parameters of M1/M3 are a ciphertext with the same length as the plaintext and a tag. The tag length of M1 and M3 are 128 and 64 bits, respectively. Fig. 1 illustrates the message encryption part of SKINNY-AEAD M1/M3 without padding.

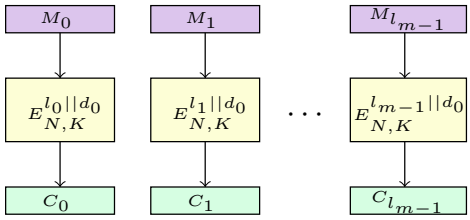

Fig. 1. The message encryption part of SKINNY-AEAD M1/M3 with SKINNY-128-384 without padding. E refers to SKINNY-128-384. For simplicity, we denote the block counter by $l_0, \ldots, l_{m-1}$ but actually refer to the state of the LFSRs serving as a block counter.

Messages are encrypted with SKINNY-128-384 under the 384-bit tweakey *i.e.*, $\text{rev}_{64}(\text{LSFR})||0^{56}||d_0||N||K$. The 384-bit tweakey states can be seen as a group of three 128-bit data that are denoted as $TK1, TK2$, and $TK3$, where $TK_1 \leftarrow \text{rev}_{64}(\text{LSFR})||0^{56}||d_0$, $TK2 \leftarrow N$, and $TK3 \leftarrow K$. In SKINNY-AEAD M1/M3, TK_1, TK_2 and TK_3 store information corresponding to block counter l_i, nonce N and a 128-bit key, respectively. $\text{rev}_{64}(\text{LSFR})$ stores 8-byte values from a 64-bit LFSR. d_0 is a domain separation value and denotes encryption of a full message block without padding, where $d_0 =$ 0x00 for M1, $d_0 =$ 0x08 for M3. $\text{rev}_{64}(\text{LSFR})$ can be updated in a series of blocks in the following way.

The 64-bit state of the LFSR can be expressed as: $x_{63}||x_{62}||\ldots||x_1||x_0$ and initialized to $\text{LFSR}_0 = 0^{63}||1$. The LFSR is updated by the function upd_{64} (*i.e.*, $\text{LFSR}_{t+1} = \text{upd}_{64}(\text{LFSR}_t)$), where upd_{64} is defined as

$$\text{upd}_{64} : x_{63}||x_{62}||\ldots||x_1||x_0 \rightarrow y_{63}||y_{62}||\ldots||y_1||y_0$$

with

$$\begin{aligned} &y_i \leftarrow x_{i-1} \text{ for } i \in \{63, 62, \ldots, 1\}\backslash\{4, 3, 1\}, \\ &y_4 \leftarrow x_3 \oplus x_{63}, \quad y_3 \leftarrow x_2 \oplus x_{63}, \\ &y_1 \leftarrow x_0 \oplus x_{63}, \quad y_0 \leftarrow x_{63}. \end{aligned}$$

Before loaded into the tweakey state, the order of the bytes of the LFSR state should be first reversed by rev_{64} function, where rev_{64} is defined as

$$\text{rev}_{64} : S_7||S_6||S_5||S_4||S_3||S_2||S_1||S_0 \mapsto S_0||S_1||S_2||S_3||S_4||S_5||S_6||S_7,$$

where S_i is an 8-bit value, $0 \leq i \leq 7$.

2.2 Specification of the Underlying Primitive of SKINNY-AEAD M1/M3

In the SKINNY-AEAD M1/M3, 56-round SKINNY-128-384 is adopted as their underlying tweakable block cipher. The SKINNY family ciphers follow the TWEAKEY framework from [11] and therefore take a tweakey input. Here, we provide a brief specification of SKINNY-128-384 that is relevant to our related-tweakey impossible differential attacks. For more details about SKINNY, please refer to [4].

Initialization. SKINNY-128-384 has 128-bit internal states and 384-bit tweakey. 128-bit internal states are represented as 4×4 array of cells with each cell being a byte. The tweakey states can be seen as a group of three 4×4 arrays (*i.e.*, $TK1$, $TK2$ and $TK3$). Note that the internal states and tweakey states are loaded row-wise as shown in Fig. 2.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Fig. 2. Cell number of the states in the 4×4 array

Round Function. One encryption round of SKINNY-128-384 contains five operations in the following order: SubCells, AddConstants, AddRoundTweakey, ShiftRows and MixColumns, as shown in Fig. 3 (a).

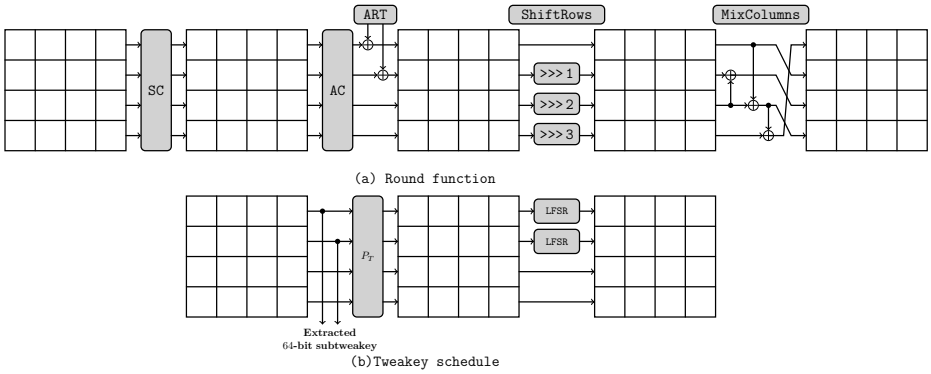

Fig. 3. Round function and tweakey schedule of SKINNY-128-384

1) **SubCells** (SC). An 8-bit S-box is applied to each cell of the internal states. It is a non-linear operation in the round function.

2) **AddConstants** (AC). Three round constants (*i.e.*, c_0, c_1 and c_2) are XORed to the first three cells of the first column of an internal state. c_0 and c_1 are generated by a 6-bit affine LFSR, and $c_2 = \texttt{0x02}$.
3) **AddRoundTweakey** (ART). Only the first two rows of round tweakey are XORed with the first two rows of the corresponding internal state. The round tweakey (tk_i) in the i-th round is defined as $tk_i = TK1_i \oplus TK2_i \oplus TK3_i$. The tweakey arrays are updated by permutation and LFSR operations in the tweakey schedule algorithm as shown in Fig. 3 (b). In the permutation phase, a permutation $P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ on cell positions is applied to $TK1, TK2$ and $TK3$, respectively. In the LFSR update phase, each cell of the first two rows of $TK2$ and $TK3$ are individually updated using an 8-bit LFSR, while there is no LFSR updating on $TK1$. The LFSRs are given in Table 2.

Table 2. The LFSRs used in SKINNY-128-384 to generate round tweakeys

TK	LFSR
$TK2$	$(x_7\|\|x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1\|\|x_0) \rightarrow (x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1\|\|x_0\|\|x_7 \oplus x_5)$
$TK3$	$(x_7\|\|x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1\|\|x_0) \rightarrow (x_0 \oplus x_6\|\|x_7\|\|x_6\|\|x_5\|\|x_4\|\|x_3\|\|x_2\|\|x_1)$

Note: x_7 (resp. x_0) is the MSB (resp. LSB) bit of the cell.

4) **ShiftRow** (SR). The second, third and fourth rows are rotated by 1, 2 and 3 cell positions to the right, respectively.
5) **MixColumns** (MC). Every column of the internal state is multiplied by a binary matrix M below. The inverse MixColumns M^{-1} can be deduced and given as follows.

$$M = \begin{pmatrix} 1\,0\,1\,1 \\ 1\,0\,0\,0 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \end{pmatrix} \qquad M^{-1} = \begin{pmatrix} 0\,1\,0\,0 \\ 0\,1\,1\,1 \\ 0\,1\,0\,1 \\ 1\,0\,0\,1 \end{pmatrix}$$

2.3 Properties of SKINNY

Liu *et al.* [15] proposed some properties of SKINNY block cipher. For the sake of completeness, we give several properties related to our attacks below.

1) The order of ART, SR and MC operations in any round can be changed by first using SR and MC, and then xoring the internal state with an equivalent round tweakey. The equivalent tweakey denoted as tk^{eq} can be computed by $tk^{eq} = MC(SR(tk))$ as given in Fig. 4.
2) Tweakey schedule of SKINNY is linear. Thus, if the differential value injected in the master tweakey is known, we can determine the exact differences in all the other subtweakeys.

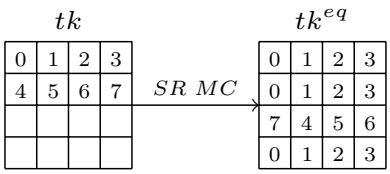

Fig. 4. Obtain tk^{eq} by changing the order of ART, SR and MC

3) Subtweakey difference cancellation. In our attack scenario, a single cell of $TK1$ and $TK2$ is active and the active cell position is in the first and second rows. There is no difference for $TK3$. Hence, we take $TK2$ as an example. As noticed in [4], for a given active tweakey cell, there is only one subtweakey difference cancellation every 30 rounds for $TK2$. Let α_1 and α_2 be nonzero differences of the active cells for $TK1$ and $TK2$, respectively. In the first round, the subtweakey difference is $\alpha_1 \oplus \alpha_2$ for the active cell. For the $(2i+1)$-th round, the subtweakey difference is $\alpha_1 \oplus \mathrm{LFSR}_2^i(\alpha_2)$ by ignoring the permutation P_T, where LFSR_2 has a cycle of 15. Thus one subtweakey cancellation (*i.e.*, $\alpha_1 \oplus {\mathrm{LFSR}_2}^i(\alpha_2) = 0$) can occur every 30 rounds. In SKINNY-128-384, only the first two rows of the round tweakey tk_i are XORed with the internal state in i-th round. Moreover, according to the permutation P_T, the subtweakey cells involved in the i-th round will next appear in $(i+2)$-th round. Therefore, we can deduce that there are three rounds of fully inactive internal states for $TK2$.

2.4 Notations

We give the following notations (see Table 3) which are used in this paper. Some of them follow [8,15].

3 Related-Tweakey Impossible Differential Distinguisher

Due to the encryption mode, nonce-respecting scenario and complex tweakey initialization, the attacks on SKINNY-AEAD M1/M3 should fulfill some constraints. In this section, based on these constraints and several properties of SKINNY cipher, we search for the related-tweakey impossible differential distinguishers. Here, we first give the descriptions of constraints for distinguishers in SKINNY-AEAD M1/M3.

3.1 Constraints of Searching for Distinguishers in SKINNY-AEAD M1/M3

To facilitate the description, we use $\Delta\mathcal{TK}_1^1$, $\Delta\mathcal{TK}_1^2$ and $\Delta\mathcal{TK}_1^3$ to denote tweakey differences of the distinguisher corresponding to $TK1$, $TK2$ and $TK3$ in the

Table 3. Notations related to our attacks

$\Delta_{\mathbb{X}}$ (resp. $\Delta_{\mathbb{Y}}$)	Input (resp. output) differences of the impossible differential.
Δ_{in} (resp. Δ_{out})	Set of all impossible input (resp. output) differences.
r_{Δ}	The number of rounds of the impossible differential.
r_{in} (resp. r_{out})	The number of rounds of the differential $\Delta_{\mathbb{X}} \rightarrow \Delta_{in}$ (resp. $\Delta_{\mathbb{Y}} \rightarrow \Delta_{out}$).
c_{in} (resp. c_{out})	The number of bit-conditions that have to be verified from Δ_{in} to $\Delta_{\mathbb{X}}$ (resp. from Δ_{out} to $\Delta_{\mathbb{Y}}$).
K_{in} (resp. K_{out})	The subset of subtweakey bits involved in the first r_{in} rounds (resp. the last r_{out} rounds).
X_i (resp. Y_i)	The state before SC (resp. AC, ART) in the i-th round.
Z_i (resp. W_i)	The state before SR (resp. MC) in the i-th round.
tk_i	The round tweakey of i-th round.
col(k)	The k-th column, where $1 \leq k \leq 4$.
$\Delta X/Y/Z/W$,	The difference of the state $X/Y/Z/W$.
$\Delta X_i[n]$	The n-th cell difference value of state X in i-th round, where $0 \leq n \leq 15$.
$\Delta X_i[k, \ldots, l]$	The k-th, ..., l-th cell difference value of state X in i-th round, where $0 \leq k, l \leq 15$.

first round, respectively. $\Delta\mathcal{SC}_1$ represents the internal state difference after SC in the first round of the distinguisher. Usually, to achieve powerful impossible differential tweakey recovery attacks, the number of active cells of input/output differences in the distinguisher is constrained to be as small as possible. Besides, to obtain longer distinguishers for SKINNY-128-384, two strategies are often used, as shown in [15,19]: (1) using the subtweakey difference ($\Delta\mathcal{TK}_1^1 \oplus \Delta\mathcal{TK}_1^2 \oplus \Delta\mathcal{TK}_1^3$) to cancel $\Delta\mathcal{SC}_1$; and (2) applying the subtweakey difference cancellation property (*i.e.*, LFSR($\Delta\mathcal{TK}_1^2[p]$) = $\Delta\mathcal{TK}_1^1[p]$), which ensure that there is no active tweakey cell for 3 consecutive rounds (*i.e.*, second, third and fourth round of the distinguisher).

Recall the composition of tweakey used in SKINNY-AEAD M1/M3 in Section 2.1. The highest 64 bits of $\mathcal{TK}_1^1$ serve as a block counter l_i. To reduce the time complexity of tweakey recovery attacks, we set $|A| = 0$[2] and $\Delta\mathcal{TK}_1^1[p] =$ 0x03[3], where p ($0 \leq p \leq 7$) is the position of the active cell. $\mathcal{TK}_1^3$ stores the 128-bit key. To make our attack more practical, we set $\Delta\mathcal{TK}_1^3 = 0$, and then we can mount our attacks after obtaining only one encryption oracle. $\mathcal{TK}_1^2$ stores nonce

[2] $|A|$ denotes the bit length of associated data A. When $|A| = 0$, there is no encryption of associated data in the tweakey recovery attack.

[3] Taking a 20-round tweakey recovery attack as an example, we explain why $\Delta\mathcal{TK}_1^1[p] =$ 0x03. In the tweakey recovery attacks, if $\Delta\mathcal{TK}_1^1[p] =$ 0x03, we will utilize plaintext-ciphertext pairs in the i-th and $(i+1)$-th block of encryptions in SKINNY-AEAD M1/M3, where $i = 8 \cdot ((P_T^{-1}) \circ (P_T^{-1}(p))) + 1$, which needs less time complexity than other values once p is fixed.

N. Since we mount attacks under a nonce-respecting scenario and $\Delta\mathcal{TK}_1^3 = 0$, $\Delta\mathcal{TK}_1^2 \neq 0$.

Combining all the above constraints on the tweakey differences, we can get $\Delta\mathcal{TK}_1^2[p] = \texttt{0x81}$, $\Delta\mathcal{SC}_1[p] = \texttt{0x82}$ under $\Delta\mathcal{TK}_1^1[p] = \texttt{0x03}$, where the p-th cell is the only active one in the corresponding difference of the first round of distinguishers.

3.2 Searching for Related-Tweakey Impossible Differential Distinguisher with STP

In recent years, some works [12,14,16,18] use the Boolean Satisfiability Problem (SAT) [17]/ Satisfiability Modulo Theories (SMT) problem [3] solver STP[4] (Simple Theorem Prover) to search for distinguishers of some ciphers. The application of STP for cryptanalysis is a decision problem to confirm whether there is a solution to a set of equations. To search for related-tweakey impossible differential distinguishers, we should use some equations to describe the differential propagation properties of basic operations in the round function and tweakey schedule. To make it clear, we present all relevant equations for basic operations as follows.

Equations for Basic Operations. In the round function and tweakey schedule of SKINNY-128-384, basic operations consist of XOR, three-branch, Substitution and confusion layer. We give the following properties describing the differential propagation equations.

Property 1 (XOR [14]) Let Δ_{in1} and Δ_{in2} represent two input differences and Δ_{out} denote the output difference for the XOR operation. Then the corresponding equation can be expressed as $\Delta_{out} = \Delta_{in1} \oplus \Delta_{in2}$.

Property 2 (Three-Branch [14]) Let Δ_{in} denote the input difference of this operation, two output differences are represented as Δ_{out1} and Δ_{out2}. Then the relation between them is $\Delta_{out1} = \Delta_{out2} = \Delta_{in}$.

Property 3 (Substitution) Let S be an S-box used in the round function of SKINNY cipher. Define X_{in} as the input value of S. Let Δ_{in} and Δ_{out} represent input and output differences, respectively. Denote v as a flag variable indicating the validity of difference propagation. Then their relations are

$$v = \begin{cases} 0 \text{ (invalid)}, & \text{if } S(X_{in}) \oplus S(X_{in} \oplus \Delta_{in}) \neq \Delta_{out}, \\ 1 \text{ (valid)}, & \text{if } S(X_{in}) \oplus S(X_{in} \oplus \Delta_{in}) = \Delta_{out}, \end{cases}$$

where $v = 1$ represents that there is a value X_{in} that makes the difference propagation $\Delta_{in} \to \Delta_{out}$ valid.

[4] https://github.com/stp/stp

Property 4 (Confusion Layer) Denote M as the confusion matrix. $\overrightarrow{\Delta C_{in}}$ and $\overrightarrow{\Delta C_{out}}$ represent the column-wise input and output difference, respectively. Then the equation is $\overrightarrow{\Delta C_{out}} = M \cdot \overrightarrow{\Delta C_{in}}$.

In this section, we utilize a similar method used in [15,19] to search for related-tweakey impossible differential distinguishers with STP. Firstly, according to the round function and tweakey schedule, we use Property 1∼4 to construct some equations to express the bit-level differential propagation properties between input and output differences. In addition, we should construct some equations representing the constraints on tweakey and state differences. Finally, we can confirm whether an expected impossible differential distinguisher exists through the result returned by the solver STP. If the solver STP returns *Invalid* for the specific input and output active cell positions, it shows that this differential distinguisher is impossible.

Note that, only the active input difference values are fixed as mentioned earlier (*i.e.*, $\Delta\mathcal{TK}_1^1[p] = \texttt{0x03}$, $\Delta\mathcal{TK}_1^2[p] = \texttt{0x81}$ and $\Delta\mathcal{SC}_1[p] = \texttt{0x82}$), but to determine the value of p, we have to traverse all the values of cell position p and active cell positions of output differences as shown in Algorithm 1.

Algorithm 1: Seaching Related-Tweakey Impossible Distinguishers

Input: R: The number of rounds covered by the expected distinguisher.
Output: R-round related-tweakey impossible differential distinguisher or "*No solution*".

▷ p_i: Active cell position of input differences of the distinguisher.
▷ p_o: Active cell position of output differences of the distinguisher.
for $p_i \leftarrow 0$ *to* 7 **do**
  for $p_o \leftarrow 0$ *to* 15 **do**
    ▷ (1) Construct equations to describe differential propagation.
    for $r \leftarrow 1$ *to* $R+1$ **do**
      Use the Property 1∼4 to construct equations for the round function and tweakey schedule of r-th round.
    ▷ (2) Construct equations to represent the distinguisher's restrictive conditions.
    Construct equations to describe the difference values for the active cells of the internal state after SC and tweakey in the first round (*i.e.*, $\Delta\mathcal{SC}_1[p_i] = \texttt{0x82}$, $\Delta\mathcal{TK}_1^1[p_i] = \texttt{0x03}$ and $\Delta\mathcal{TK}_1^2[p_i] = \texttt{0x81}$) and the differential value of the active cell in the output difference of the distinguisher (*i.e.*, $\Delta\mathcal{SC}_{R+1}[p_o] \neq \texttt{0x00}$).
    Input all the equations (1) and (2) into STP solver and solve them.
    if *STP return "Invalid"* **then**
      Return the active cell position (p_i, p_o) of the R-round relate-tweakey impossible differential distinguisher.
Return "*No solution*"

3.3 14-Round Related-Tweakey Impossible Differential Distinguishers

Utilizing Algorithm 1, we find some 14-round distinguishers. Under the related-tweakey model, our impossible differential distinguishers start with one active cell in internal state after SC of the first round and end with one active cell after SC of the 15-th round. Thus, they can be seen as complete 14-round distinguishers for the underlying primitive SKINNY-128-384 in SKINNY-AEAD M1/M3. In the tweakey recovery phase, we select the related-tweakey impossible differential distinguisher with $(p_i, p_o) = (2, 10)$[5], which can obtain a lower attack complexity for both 20-round and 18-round attacks. This distinguisher is illustrated in Fig. 5. The 14-round related-tweakey impossible differential is: $(00\alpha 0|0000|0000|0000) \overset{14r}{\nrightarrow} (0000|0000|00*0|0000)$, where $\alpha =$ 0x82 and $*$ denotes any non-zero difference. An 8.5-round related-tweakey differential in the forward direction (having probability 1) starting at Y_1 (after SC in the first round) is concatenated to a 5.5-round related-tweakey differential (having probability 1) starting from Y_{15} (before the AC and ART in the 15-th round) in the backward direction. The contradiction takes place at X_{10} in the 10-th round.

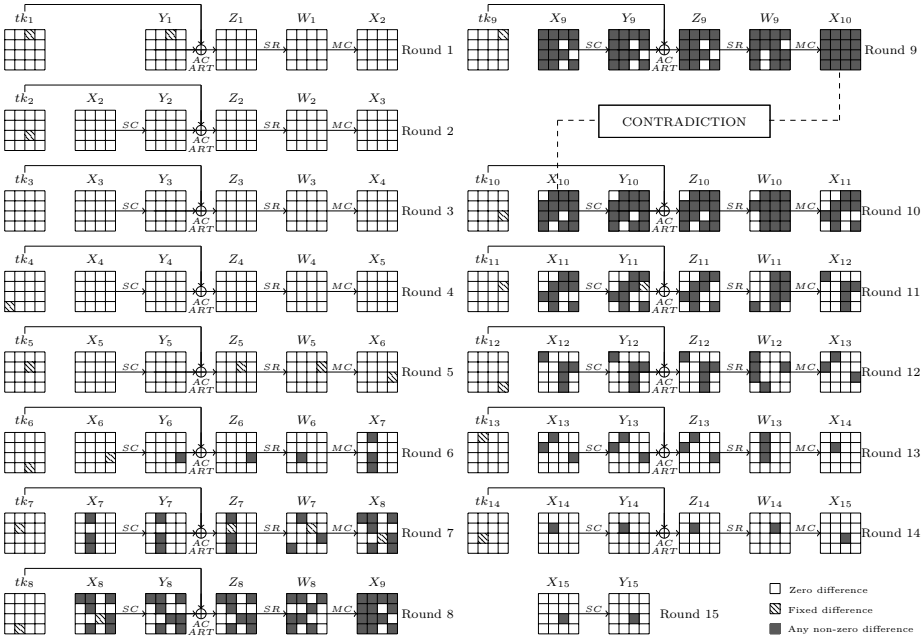

Fig. 5. 14-round related-tweakey impossible differential distinguisher for the underlying primitive SKINNY-128-384 in SKINNY-AEAD M1/M3

[5] p_i and p_o denote the active cell positions corresponding to the input and out differences of the distinguisher, respectively.

4 Tweakey Recovery Attack on 20-Round SKINNY-AEAD M1/M3

In this section, we propose a 20-round related-tweakey impossible differential attack on SKINNY-AEAD M1/M3. The whole attack contains two phases (i.e, distinguisher construction phase in Section 3 and tweakey recovery phase in this section). In the distinguisher construction phase, we find some 14-round impossible differential distinguishers under some constraints for the underlying primitive SKINNY-128-384. As a result, we select one distinguisher with (p_i, p_o)=(2,10) to mount our tweakey recovery attack to achieve a lower attack complexity. In the tweakey recovery phase, we construct a 20-round related tweakey recovery attack to derive the right tweakey by filtering out the wrong candidates suggested by the impossible differential distinguisher.

Fig. 6 presents a 20-round tweakey recovery attack on SKINNY-AEAD M1/M3 based on the 14-round related-tweakey impossible differential distinguisher. We extend the distinguisher 4 rounds on the top and 2 rounds at the bottom to cover 6 rounds in the tweakey recovery phase. As discussed in Section 2.3, we can obtain the equivalent plaintext P^{eq} (resp. equivalent tweakey tk_1^{eq}) by applying $P^{eq} = MC \circ SR \circ AC \circ SC(P)$ (resp. $tk_1^{eq} = MC \circ SR(tk_1)$) in the first round. We start our tweakey recovery attack at W_1 since there is no tweakey involved before W_1. In the rest of section, we call the inputs at the position W_1 as the plaintext inputs P^{eq}. The original plaintext P can be recovered by applying $SC^{-1} \circ AC^{-1} \circ SR^{-1} \circ MC^{-1}(P^{eq})$.

The 14-round distinguisher is placed between the 5-th round and the 19-th round as shown in Fig. 6. In the first round, the subtweakey difference $\Delta tk_1[1]$ (*i.e.*, $\Delta tk_1^{eq}[1, 5, 13]$) is a nonzero fixed difference, the other cell-position differences are all zero . The values of fixed differences $tk_1[1]$, $tk_3[0]$, $tk_5[2]$ and $tk_{19}[0]$ affect 20-round attack. These differential values are dependent on each other. If the input tweakey differences of distinguisher are fixed, other subtweakey differences can be derived using the tweakey schedule. The involved active subtweakey differences in the whole attack procedure are shown in Table 4.

Table 4. Active subtweakey difference values related to 20-round tweakey recovery

r	$\Delta TK1_r[p]$	$\Delta TK2_r[p]$	$\Delta TK3_r[p]$	$\Delta tk_r[p]$
1	0x03	0xE0	0x00	$\Delta tk_1[1]$ = 0xE3
3		0xC0		$\Delta tk_3[0]$ = 0xC3
5		0x81		$\Delta tk_5[2]$ = 0x82
19		0xC3		$\Delta tk_{19}[0]$ = 0xC0

Note: $\Delta tk_r[p] = \Delta TK1_r[p] \oplus \Delta TK2_r[p] \oplus \Delta TK3_r[p]$. r and p denote the number of round and active cell position, respectively.

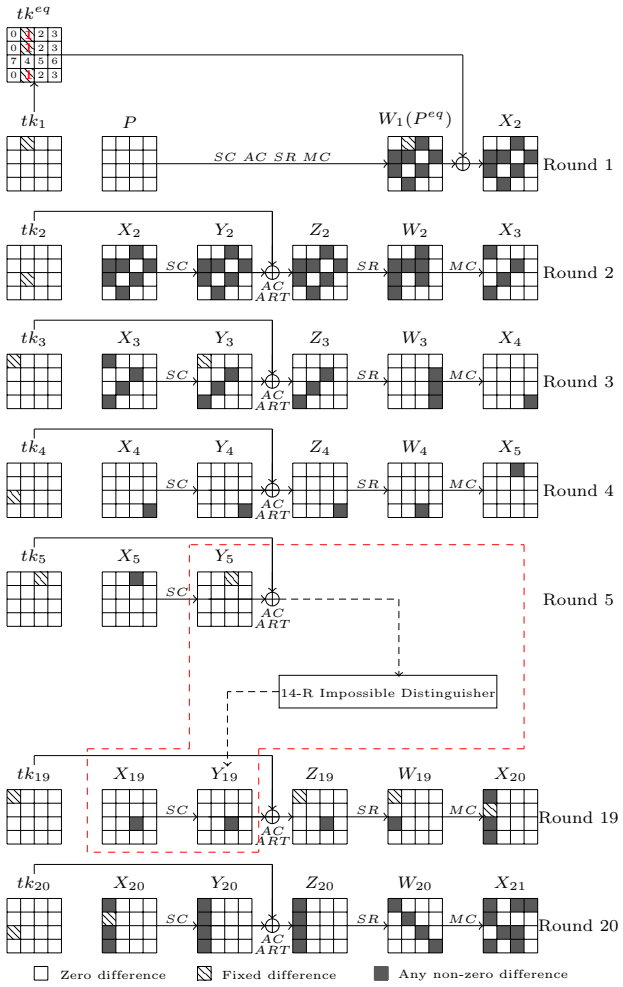

Fig. 6. 20-round tweakey recovery attack for SKINNY-AEAD M1/M3

In the 20-round attack, $r_\Delta = 14$, $r_{in} = 4$, $r_{out} = 2$, $c_{in} = 48$, $|\Delta_{in}| = 56$, $c_{out} = 56$, $|\Delta_{out}| = 64$ and $|K_{in} \cup K_{out}| = 13 \times 8 + 2 \times 8 = 120$. The attack procedure is briefly described in Algorithm 2. Detailed steps of 20-round cryptanalysis are given as follows.

Data Collection. The adversary should construct 2^N pairs of structures S_1 and S_2 at W_1, where each structure contains $2^{|\Delta_{in}|} = 2^{56}$ equivalent plaintexts. For each equivalent plaintext pair $P^{eq} \in S_1$ and $\overline{P^{eq}} \in S_2$, its difference is $P^{eq} \oplus \overline{P^{eq}} = (0, \alpha, *, 0, *, *, 0, *, *, 0, *, 0, 0, *, 0, 0)$, where $\alpha = \Delta tk_1[1] =$ 0xE3 and $*$ denotes any nonzero byte value. Using $2^N \cdot (2 \cdot 2^{|\Delta in}|) = 2^{N+|\Delta_{in}|+1}$ equivalent plain-

Algorithm 2: Tweakey Recovery Attack on 20-Round `SKINNY-AEAD` M1/M3

▷ Data collection;
for 2^N pairs of structures **do**
  for $2^{|\Delta_{in}|}$ equivalent plaintext P_i^{eq} in S_1 **do**
    $\overline{P_i^{eq}} = P_i^{eq} \oplus (0, \alpha, *, 0, *, *, 0, *, *, 0, *, 0, 0, *, 0, 0)$, where $\alpha =$ `0xE3`;
    Choose a nonce N_i that has never been used;
    $\overline{N_i} = N_i \oplus (0, \gamma, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, where $\gamma =$ `0xE0`;
    $P_i = SC^{-1} \circ AC^{-1} \circ SR^{-1} \circ MC^{-1}(P_i^{eq})$;
    $\overline{P_i} = SC^{-1} \circ AC^{-1} \circ SR^{-1} \circ MC^{-1}(\overline{P_i^{eq}})$;
    for m from 0 to 7 by 1 **do**
      $C \leftarrow$ encrypt P_i under (l_m, N_i, Key);
    $C_i \leftarrow$ encrypt P_i under (l_8, N_i, Key);
    $W_{20_i} \leftarrow MC^{-1}(C_i)$;
  for $2^{|\Delta_{in}|}$ plaintext $\overline{P_i}$ in S_2 **do**
    for n from 0 to 8 by 1 **do**
      $\overline{C} \leftarrow$ encrypt $\overline{P_i}$ under $(l_n, \overline{N_i}, Key)$;
    $\overline{C_i} \leftarrow$ encrypt $\overline{P_i}$ under $(l_9, \overline{N}_i, Key)$;
    $\overline{W_{20_i}} \leftarrow MC^{-1}(\overline{C_i})$;
  for $2^{2|\Delta_{in}|}$ pairs **do**
    $\Delta W_{20} \leftarrow W_{20_i} \oplus \overline{W_{20_i}}$;
    if $\Delta W_{20} = (*, 0, 0, 0, 0, *, 0, 0, 0, 0, *, 0, 0, 0, 0, *)$ **then**
      Remain the pair of (P_i, N_i, C_i) and $(\overline{P_i}, \overline{N_i}, \overline{C_i})$;
      ▷Seive 2^{N+16} remaining pairs finally;
▷ Subkey recovery;
Call Algorithm 3 under 2^{N+16} pairs to discard the candidates of $tk_1[0, 1, 2, 3, 4, 5, 6, 7]$, $tk_2[0, 3, 4, 5, 7]$ and $tk_{20}[0, 4]$, and obtain 2^ρ $|K_{in} \cup K_{out}|$-bit remaining subtweakey candidates tk_{rem};
▷ Brute force;
for 2^ρ tk_{rem} **do**
  for 2^8 remaining master key $MK[12]$ **do**
    Compute the master key MK from tk_{rem} and $MK[12]$ using the tweakey schedule;
    Get a random new pair of (P, C) with a nonce value N and a number related to the first block counter l_0;
    $C' \leftarrow$ encrypt P under (l_0, N, MK);
    if $C' = C$ **then**
      Return the MK as the right tweakey.

texts, we can generate $2^{N+2|\Delta_{in}|}$ plaintext pairs $(P, \overline{P})$. In the 20-round tweakey recovery attack, we define the two related-tweakey inputs as $(TK1||TK2\,||TK3)$ and $(\overline{TK1}||\overline{TK2}||\overline{TK3})$. Encrypt the plaintexts P (resp. $\overline{P}$) under $TK1||TK2\,||TK3$ (resp. $\overline{TK1}||\overline{TK2}||\overline{TK3}$) to obtain the corresponding ciphertexts C (resp. $\overline{C}$). In the 20-round attack of SKINNY-AEAD M1/M3, ($TK1$,$TK2$,$TK3$) (resp. $(\overline{TK1}, \overline{TK2}, \overline{TK3})$) is expressed as (l_8, N, Key) (resp. $(l_9, \overline{N}, Key)$), where l_8 (resp. l_9) represents the first 128-bit tweakey value in the 9-th (resp. 10-th) block message encryption, N and $\overline{N}$ denote different nonce values, and Key means the 128-bit key. Note that $l_8 \oplus l_9 = (0, \beta, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ and $N \oplus \overline{N} = (0, \gamma, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, where $\beta =$ 0x03 and $\gamma =$ 0xE0 (see the case of $r = 1$ in Table 4).

After obtaining ciphertexts C and $\overline{C}$, we can get W_{20} and $\overline{W_{20}}$ directly respectively, as shown in line[12] and line[17] of Algorithm 2. For each ciphertext pair, check whether the condition in line[20] is satisfied and remain it if true. In total, $2^{N+2|\Delta_{in}|-(128-|\Delta_{out}|)-32} = 2^{N+16}$ plaintext-ciphertext pairs are remained in the phase of data collection. Time complexity of this phase is $(2^N \cdot (2^{|\Delta_{in}|} \cdot (2 \cdot \frac{1}{20} + 9 + \frac{1}{2} \cdot \frac{1}{20} + 10 + \frac{1}{2} \cdot \frac{1}{20}))) \approx 2^{N+60.3}$ times of 20-round encryptions, and the memory complexity is

$$\begin{aligned}\mathcal{M}_{col} &= 2^{|\Delta_{in}|} \cdot (2 \cdot 128) + 2 \cdot 2^{|\Delta_{in}|} \cdot (3 \cdot 128) + 2 \cdot 2^{N+16} \cdot (3 \cdot 128) \\ &= (2^{|\Delta_{in}|-0.59} + 2^{|\Delta_{in}|+1} + 2^{N+17}) \cdot (3 \cdot 128) \text{ bits.}\end{aligned}$$

Tweakey Recovery. The tweakey recovery procedure of the 20-round attack is briefly described in Algorithm 3. We will show the detailed attack procedure as follows. In the attack process of SKINNY-AEAD M1/M3, the value of $(l_8, l_9, N, \overline{N})$ is known, in the following section, we focus on guessing the value of subtweakey tk_i ($i \in \{1, 2, 20\}$) to deduce the master tweakey corresponding to Key[6].

1) Line[1,2,3] in Algorithm 3. Guess $tk_{20}[4]$ and compute $\Delta X_{20}[4]$ under 2^{N+16} remaining pairs. According to Fig. 6, get $\Delta X_{20}[4] = SC^{-1}(Z_{20}[4] \oplus tk_{20}[4] \oplus c^1_{20}) \oplus SC^{-1}(\overline{Z_{20}[4]} \oplus tk_{20}[4] \oplus c^1_{20})$, where $c^1_{20} =$ 0x02 is the 20-th round constant corresponding to c_1. Checking if $\Delta X_{20}[4] = \Delta tk_{19}[0] =$ 0xC0 will lead to an 8-bit filter and generally sieve $2^{N+16-8} = 2^{N+8}$ plaintext-ciphertext pairs. The time complexity of this step is $2^8 \cdot 2 \cdot 2^{N+16} \cdot \frac{1}{16} \cdot \frac{1}{20} \approx 2^{N+16.7}$ 20-round encryptions.
2) Line[4,5,6] in Algorithm 3. Guess $tk_{20}[0]$ and compute $\Delta X_{20}[0, 4, 8, 12]$ under 2^{N+8} remaining pairs from the previous step. According to Fig. 6, compute $\Delta X_{20}[0] = SC^{-1}(Z_{20}[0] \oplus tk_{20}[0] \oplus c^0_{20}) \oplus SC^{-1}(\overline{Z_{20}[0]} \oplus tk_{20}[0] \oplus c^0_{20})$, $\Delta X_{20}[4] = SC^{-1}(Z_{20}[4] \oplus tk_{20}[4] \oplus c^1_{20}) \oplus SC^{-1}(\overline{Z_{20}[4]} \oplus tk_{20}[4] \oplus c^1_{20})$, $\Delta X_{20}[8] = SC^{-1}(Z_{20}[8] \oplus c^2_{20}) \oplus SC^{-1}(\overline{Z_{20}[8]} \oplus c^2_{20})$ and $\Delta X_{20}[12] = SC^{-1}(Z_{20}[12]) \oplus SC^{-1}(\overline{Z_{20}[12]})$, where $c^0_{20} =$ 0x0B, $c^1_{20} =$ 0x02, and $c^2_{20} =$ 0x02 are the 20-th round constants corresponding to c_0, c_1 and c_2, respectively. Based on the properties of MC operations on col(1) of W_{19}, we have the conditions $\Delta X_{20}[0] = \Delta X_{20}[12]$ and $\Delta X_{20}[4] \oplus \Delta X_{20}[8] = \Delta X_{20}[12]$. Check

[6]We define the master tweakey corresponding to Key as master key MK.

Algorithm 3: Subtweakey Recovery Procedure of the 20-Round Attack

```
for 2^8 tk_20[4] do
    for 2^{N+16} remaining pairs do
        Compute ΔX_20[4] and use the condition (i.e., ΔX_20[4] is a fixed
         difference) to get 2^{N+16-8} = 2^{N+8} remaining pairs;
    for 2^8 tk_20[0] do
        for 2^{N+8} remaining pairs do
            Compute ΔX_20[0,4,8,12] and use the conditions
             ΔX_20[0] = ΔX_20[12] and ΔX_20[4] ⊕ ΔX_20[8] = ΔX_20[12] to
             obtain 2^{N+8-16} = 2^{N-8} remaining pairs;
        for 2^16 tk_1[3,5] do
            for 2^{N-8} remaining pairs do
                Compute ΔW_2[4,8] and use the condition ΔW_2[4] = ΔW_2[8] to
                 choose 2^{N-8-8} = 2^{N-16} remaining pairs;
            for 2^24 tk_1[1,2,7] do
                for 2^{N-16} remaining pairs do
                    Compute ΔW_2[2,6,10] and use the conditions
                     ΔW_2[6] = ΔW_2[10] and ΔW_2[2] = ΔW_2[10] to sieve
                     2^{N-16-16} = 2^{N-32} remaining pairs;
                for 2^16 tk_1[0] tk_2[0] do
                    for 2^{N-32} remaining pairs do
                        Compute ΔY_3[0] and use the condition (i.e., ΔY_3[0] is a
                         fixed difference) to get 2^{N-32-8} = 2^{N-40} remaining
                         pairs;
                    for 2^16 tk_1[6] tk_2[4] do
                        for 2^{N-40} remaining pairs do
                            Compute ΔW_3[7,11,15] and use the conditions
                             ΔW_3[7] = ΔW_3[11] and ΔW_3[11] = ΔW_3[15] to
                             obtain 2^{N-40-16} = 2^{N-56} remaining pairs;
                        for 2^32 tk_1[4] tk_2[3,5,7] do
                            for 2^{N-56} remaining pairs do
                                Compute ΔY_5[2] and use the condition (i.e.,
                                 ΔY_5[2] is a fixed difference) to filter out the
                                 wrong subkeys.
```

whether the conditions are satisfied or not, which can act as a 16-bit filter to choose $2^{N+8-16} = 2^{N-8}$ remaining pairs. In this step, the time complexity is $2^{16} \cdot 2 \cdot 2^{N+8} \cdot \frac{4}{16} \cdot \frac{1}{20} \approx 2^{N+18.7}$ 20-round encryptions.

3) Line[7,8,9] in Algorithm 3. Guess $tk_1[3,5]$ and compute $\Delta W_2[4,8]$ under 2^{N-8} remaining pairs from step 2). Base on $tk^{eq}[7] = tk_1[3]$ and $tk^{eq}[10] = tk_1[5]$, we can compute $\Delta W_2[4] = SC(P^{eq}[7] \oplus tk_1[3]) \oplus SC(\overline{P^{eq}[7]} \oplus tk_1[3])$ and $\Delta W_2[8] = SC(P^{eq}[10] \oplus tk_1[5]) \oplus SC(\overline{P^{eq}[10]} \oplus tk_1[5])$. According to the properties of MC^{-1} operations on col(1) of X_3, we have $\Delta W_2[4] = \Delta W_2[8]$. Checking if $\Delta W_2[4] = \Delta W_2[8]$ will lead to an 8-bit filter and select $2^{N-8-8} = 2^{N-16}$ remaining pairs. In this step, the time complexity is $2^{32} \cdot 2 \cdot 2^{N-8} \cdot \frac{2}{16} \cdot \frac{1}{20} \approx 2^{N+17.7}$ 20-round encryptions.

4) Line[10,11,12] in Algorithm 3. Guess $tk_1[1,2,7]$ and compute $\Delta W_2[2,6,10]$ under 2^{N-16} remaining pairs from the previous step. Base on $tk^{eq}[2] = tk_1[2]$, $tk^{eq}[5] = tk_1[1]$ and $tk^{eq}[8] = tk_1[7]$, we can compute $\Delta W_2[2] = SC(P^{eq}[2] \oplus tk_1[2]) \oplus SC(\overline{P^{eq}[2]} \oplus tk_1[2])$, $\Delta W_2[6] = SC(P^{eq}[5] \oplus tk_1[1]) \oplus SC(\overline{P^{eq}[5]} \oplus tk_1[1] \oplus \Delta tk_1[1])$ and $\Delta W_2[10] = SC(P^{eq}[8] \oplus tk_1[7]) \oplus SC(\overline{P^{eq}[8]} \oplus tk_1[7])$. In the light of the properties of MC^{-1} operations on col(3) of X_3, we have the conditions $\Delta W_2[2] = \Delta W_2[10]$ and $\Delta W_2[6] = \Delta W_2[10]$. Checking whether the conditions are satisfied or not will act as a 16-bit filter and sieve $2^{N-16-16} = 2^{N-32}$ remaining pairs. The time complexity of this step is $2^{56} \cdot 2 \cdot 2^{N-16} \cdot \frac{3}{16} \cdot \frac{1}{20} \approx 2^{N+34.3}$ 20-round encryptions.

5) Line[13,14,15] in Algorithm 3. Guess $tk_1[0]$, $tk_2[0]$ and compute $\Delta Y_3[0]$ under 2^{N-32} remaining pairs from step 4). Base on $tk^{eq}[0] = tk_1[0]$, we can compute $\Delta Y_3[1] = (SC((SC(P^{eq}[0] \oplus tk_1[0]) \oplus tk_2[0] \oplus c_2^0) \oplus SC(P^{eq}[10] \oplus tk_1[5]) \oplus SC(P^{eq}[13] \oplus tk_1[1]))) \oplus (SC((SC(\overline{P^{eq}[0]} \oplus tk_1[0]) \oplus tk_2[0] \oplus c_2^0) \oplus SC(\overline{P^{eq}[10]} \oplus tk_1[5]) \oplus SC(\overline{P^{eq}[13]} \oplus tk_1[1] \oplus \Delta tk_1[1]))))$, where $c_2^0 =$ `0x03` is the second round constant corresponding to c_0. As shown in Fig. 6, $\Delta Y_3[0] = \Delta tk_3[0]$ is a fixed difference that is `0xC3`. Checking if $\Delta Y_3[0] =$ `0xC3` will act as an 8-bit filter and sieve $2^{N-32-8} = 2^{N-40}$ remaining pairs. In this step, the time complexity is $2^{72} \cdot (2 \cdot 2^{N-32}) \cdot (\frac{3}{16} + \frac{1}{16} \cdot \frac{1}{2}) \cdot \frac{1}{20} \approx 2^{N+34.5}$ 20-round encryptions.

6) Line[16,17,18] in Algorithm 3. Guess $tk_1[6]$, $tk_2[4]$ and compute $\Delta W_3[7, 11, 15]$ under 2^{N-40} remaining pairs from the step 5). Based on $tk_1[0,2,5]$, $tk_2[0]$ determined by steps 3) to 5) and $tk_2[2]$ derived from $tk_{20}[4]$, we can compute $\Delta W_3[7] = (SC(SC(P^{eq}[2] \oplus tk_1[2]) \oplus tk_2[2])) \oplus (SC(SC(\overline{P^{eq}[2]} \oplus tk_1[2]) \oplus tk_2[2]))$, $\Delta W_3[11] = SC((SC(P^{eq}[4] \oplus tk_1[0]) \oplus tk_2[4]) \oplus SC(P^{eq}[11] \oplus tk_1[6])) \oplus SC((SC(\overline{P^{eq}[4]} \oplus tk_1[0]) \oplus tk_2[4]) \oplus SC(\overline{P^{eq}[11]} \oplus tk_1[6]))$ and $\Delta W_3[15] = SC((SC(P^{eq}[0] \oplus tk_1[0]) \oplus tk_2[0]) \oplus SC(P^{eq}[10] \oplus tk_1[5])) \oplus SC((SC(\overline{P^{eq}[0]} \oplus tk_1[0]) \oplus tk_2[0]) \oplus SC(\overline{P^{eq}[10]} \oplus tk_1[5]))$. Based on the properties of MC^{-1} operations on col(4) of X_4, we have the conditions $\Delta W_3[7] = \Delta W_3[11]$ and $\Delta W_3[11] = \Delta W_3[15]$. Checking whether the conditions are satisfied or not will act as a 16-bit filter and sieve $2^{N-40-16} = 2^{N-56}$ remaining pairs. The time complexity of this step is $2^{88} \cdot (2 \cdot 2^{N-40}) \cdot (\frac{5}{16} + \frac{3}{16}) \cdot \frac{1}{20} \approx 2^{N+43.7}$ 20-round encryptions.

7) Line[19,20,21] in Algorithm 3. Guess $tk_1[4]$, $tk_2[3,5,7]$ and compute $\Delta Y_5[2]$ under 2^{N-56} remaining pairs from step 6). As shown in Fig. 6, $\Delta Y_5[2]$ is

a fixed difference that equals to $\Delta tk_5[2] = \texttt{0x82}$. Checking if $\Delta Y_5[2] = \Delta tk_5[2] = \texttt{0x82}$ can act as an 8-bit filter and discard the wrong subtweakeys. The time complexity of this step is $2^{120} \cdot (2 \cdot 2^{N-56}) \cdot (\frac{11}{16} + \frac{7}{16} + \frac{3}{16} + \frac{1}{16}) \cdot \frac{1}{20} \approx 2^{N+61.1}$ 20-round encryptions.

In step 7), the number of the wrong subtweakeys $tk_1[0, 1, 2, 3, 4, 5, 6, 7]$, $tk_2[0, 3, 4, 5, 7]$ and $tk_{20}[0, 4]$ left is

$$2^{\rho} = (2^{|K_{in} \cup K_{out}|} - 1) \times (1 - 2^{-8})^{2^{N-56}} = (2^{120} - 1) \times (1 - 2^{-8})^{2^{N-56}},$$

where we set $N = 64.6$ and $\rho = 117.8$.

Brute Force. The values of master key $MK[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]$ can be derived from tk_{rem} using the tweakey schedule. For the remaining master key $MK[12]$, we traverse 2^8 candidate values. For each guessed value of $MK[12]$, we verify the master key by one plaintext-ciphertext pair under a number related to the first block counter l_0, given nonce value N and $MK[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15]$. In this step, the time complexity is $2^{\rho} \times 2^8 = 2^{\rho+8}$ 20-round encryptions.

Complexity Computation. The 20-round attack above on `SKINNY-AEAD` M1/M3 requires a data complexity of

$$\mathcal{D} = 2^N \cdot (2^{|\Delta_{in}|} + 2^{|\Delta_{in}|}) = 2^{N+57} = 2^{64.6+57} = 2^{121.6}$$

chosen plaintexts. The total time complexity is the summation of the time consumption in all the steps. When $N = 64.6$ and $\rho = 117.8$, the total time complexity is

$$\begin{aligned}\mathcal{T} = {} & 2^{N+60.3} + 2^{N+16.7} + 2^{N+18.7} + 2^{N+17.7} + 2^{N+34.3} + 2^{N+34.5} \\ & + 2^{N+43.7} + 2^{N+61.1} + 2^{\rho+8} \approx 2^{127.1}\end{aligned}$$

20-round encryptions. In the attack procedure, we should store the wrong candidates of guessing tweakey and the pairs of equivalent plaintext, ciphertext and nonce. The total memory complexity is

$$\begin{aligned}\mathcal{M} = {} & \mathcal{M}_{col} + 2 \cdot (2^{N+8} + 2^{N-8} + 2^{N-16} + 2^{N-32} + 2^{N-40} + 2^{N-56}) \cdot (3 \cdot 128) \\ & + 2^{\rho} \cdot 128 = 2^{90.2} + 2^{82.2} + 2^{124.8} \approx 2^{124.8} \text{ bits.}\end{aligned}$$

5 Tweakey Recovery Attack on 18-Round `SKINNY-AEAD` M1/M3

We propose an 18-round related-tweakey impossible differential attack on `SKINNY-AEAD` M1/M3 in this section. The 14-round distinguisher with $(p_i, p_o) = (2, 10)$ is

placed between the third round and 17-th round. Based on the 14-round related-tweakey impossible differential distinguisher, an 18-round attack on SKINNY-AEAD M1/M3 is shown in Fig. 7 of Appendix A. We extend the distinguisher 2 rounds on the top and 2 rounds at the bottom to cover 4 rounds in the tweakey recovery phase. In the first round, the subtweakey difference $\Delta tk_1[0]$ (*i.e.*, $\Delta tk_1^{eq}[0,4,12]$) is a nonzero fixed difference, the other cell-position differences are all zero. The involved active subtweakey differences (*i.e.*, the values of $\Delta tk_1[0]$, $\Delta tk_3[2]$ and $\Delta tk_{17}[0]$) in the attack procedure are given in Table 5.

Table 5. Active subtweakey difference values related to 18-round tweakey recovery

r	$\Delta TK1_r[p]$	$\Delta TK2_r[p]$	$\Delta TK3_r[p]$	$\Delta tk_r[p]$
1		0xC0		$\Delta tk_1[0] =$ 0xC3
3	0x03	0x81	0x00	$\Delta tk_3[2] =$ 0x82
17		0xC3		$\Delta tk_{17}[0] =$ 0xC0

Note: $\Delta tk_r[p] = \Delta TK1_r[p] \oplus \Delta TK2_r[p] \oplus \Delta TK3_r[p]$. r and p denote the number of round and active cell position, respectively.

In the 18-round attack, $r_\Delta = 14$, $r_{in} = 2$, $r_{out} = 2$, $c_{in} = 0$, $|\Delta_{in}| = 8$, $c_{out} = 56$, $|\Delta_{out}| = 64$, and $|K_{in} \cup K_{out}| = 4 \times 8 + 2 \times 8 = 48$. Algorithm 4 briefly describes the 18-round attack process. We will give the detailed attack steps as follows.

Data Collection. 2^N pairs of structures S_1 and S_2 at W_1 should be constructed, where each structure contains $2^{|\Delta_{in}|} = 2^8$ equivalent plaintexts. For each equivalent plaintext pair $P^{eq} \in S_1$ and $\overline{P^{eq}} \in S_2$, its difference is $P^{eq} \oplus \overline{P^{eq}} = (\alpha, 0, 0, 0, \alpha, 0, 0, 0, 0, 0, 0, 0, \alpha, 0, 0, *)$, where $\alpha = \Delta tk_1[0] =$ 0xC3 and $*$ denotes any nonzero difference. We use $2^N \cdot (2 \cdot 2^{|\Delta in|}) = 2^{N+|\Delta_{in}|+1}$ equivalent plaintexts to construct $2^{N+2|\Delta_{in}|}$ plaintext pairs $(P, \overline{P})$. Under (l_0, N, Key) and $(l_1, \overline{N}, Key)$, we use the line[10] and line[14] in Algorithm 4 to compute W_{18} and $\overline{W_{18}}$, respectively. Note that $l_0 \oplus l_1 = (\beta, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ and $N \oplus \overline{N} = (\gamma, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, where $\beta =$ 0x03 and $\gamma =$ 0xC0 (see the case of $r = 1$ in Table 5). For each ciphertext pair, check whether the condition in line[17] is satisfied and remain it if true. In total, $2^{N+2|\Delta_{in}|-(128-|\Delta_{out}|)-32} = 2^{N-80}$ plaintext-ciphertext pairs are remained in the step of data collection. The time complexity of this step is $(2^N \cdot (2^{|\Delta_{in}|} \cdot (2 \cdot \frac{1}{18} + 1 + \frac{1}{2} \cdot \frac{1}{18} + 2 + \frac{1}{2} \cdot \frac{1}{18}))) \approx 2^{N+9.7}$ times of 18-round encryptions, and the memory complexity is

$$\begin{aligned}\mathcal{M}_{col} &= 2^{|\Delta_{in}|} \cdot (2 \cdot 128) + 2 \cdot 2^{|\Delta_{in}|} \cdot (3 \cdot 128) + 2 \cdot 2^{N-80} \cdot (3 \cdot 128) \\ &= (2^{|\Delta_{in}|-0.59} + 2^{|\Delta_{in}|+1} + 2^{N-79}) \cdot (3 \cdot 128) \text{ bits.}\end{aligned}$$

Algorithm 4: Tweakey Recovery Attack on 18-Round `SKINNY-AEAD` M1/M3

▷ Data collection;
for 2^N pairs of structures **do**
  for $2^{|\Delta_{in}|}$ equivalent plaintext P_i^{eq} in S_1 **do**
    $\overline{P_i^{eq}} = P_i^{eq} \oplus (\alpha, 0, 0, 0, \alpha, 0, 0, 0, 0, 0, 0, 0, \alpha, 0, 0, *)$, where $\alpha =$ `0xC3`;
    Choose a nonce N_i that has never been used;
    $\overline{N_i} = N_i \oplus (\gamma, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, where $\gamma =$ `0xC0`;
    $P_i = SC^{-1} \circ AC^{-1} \circ SR^{-1} \circ MC^{-1}(P_i^{eq})$;
    $\overline{P_i} = SC^{-1} \circ AC^{-1} \circ SR^{-1} \circ MC^{-1}(\overline{P_i^{eq}})$;
    $C_i \leftarrow$ encrypt P_i under (l_0, N_i, Key);
    $W_{18_i} \leftarrow MC^{-1}(C_i)$;
  for $2^{|\Delta_{in}|}$ plaintext $\overline{P_i}$ in S_2 **do**
    $\overline{C} \leftarrow$ encrypt $\overline{P_i}$ under $(l_0, \overline{N_i}, Key)$;
    $\overline{C_i} \leftarrow$ encrypt $\overline{P_i}$ under $(l_1, \overline{N}_i, Key)$;
    $\overline{W_{18_i}} \leftarrow MC^{-1}(\overline{C_i})$;
  for $2^{2|\Delta in|}$ pairs **do**
    $\Delta W_{18} \leftarrow W_{18_i} \oplus \overline{W_{18_i}}$;
    if $\Delta W_{18} = (*, 0, 0, 0, 0, *, 0, 0, 0, 0, *, 0, 0, 0, 0, *)$ **then**
      Remain the pair of (P_i, N_i, C_i) and $(\overline{P_i}, \overline{N_i}, \overline{C_i})$;
      ▷Seive 2^{N-80} remaining pairs finally;
▷ Subkey recovery;
for 2^8 $tk_{18}[4]$ **do**
  for 2^{N-80} remaining pairs **do**
    Compute $\Delta X_{18}[4]$ and use the condition (*i.e.*, $\Delta X_{18}[4]$ is a fixed difference) to get $2^{N-80-8} = 2^{N-88}$ remaining pairs;
  for 2^8 $tk_{18}[0]$ **do**
    for 2^{N-88} remaining pairs **do**
      Compute $\Delta X_{18}[0, 4, 8, 12]$ and use the conditions $\Delta X_{18}[0] = \Delta X_{18}[12]$ and $\Delta X_{18}[4] \oplus \Delta X_{18}[8] = \Delta X_{18}[12]$ to obtain $2^{N-88-16} = 2^{N-104}$ remaining pairs;
    for 2^{32} $tk_1[2, 3, 7]$ $tk_2[2]$ **do**
      for 2^{N-104} remaining pairs **do**
        Compute $\Delta Y_3[2]$ and use the condition (*i.e.*, $\Delta Y_3[2]$ is a fixed difference) to filter out the wrong subkeys. Finally, 2^ρ $|K_{in} \cup K_{out}|$-bit subtweakey candidates tk_{rem} have been left;
▷ Brute force;
for 2^ρ tk_{rem} **do**
  for 2^{80} remaining master key $MK[0, 1, 4, 5, 6, 11, 12, 13, 14, 15]$ **do**
    Compute the master key MK from tk_{rem} and $MK[0, 1, 4, 5, 6, 11, 12, 13, 14, 15]$ using the tweakey schedule algorithm;
    Get a random new pair of (P, C) with a nonce value N and a number related to the first block counter l_0;
    $C' \leftarrow$ encrypt P under (l_0, N, MK) ;
    if $C' = C$ **then**
      Return the MK as the right key.

Tweakey Recovery. In the attack process of SKINNY-AEAD M1/M3, we are concerned about guessing the value of subtweakey tk_i ($i \in \{1, 2, 18\}$) to deduce the master tweakey corresponding to Key.

1) Line[21,22,23] in Algorithm 4. Guess $tk_{18}[4]$ and compute $\Delta X_{18}[4]$ under 2^{N-80} remaining pairs, where $\Delta X_{18}[4] = SC^{-1}(Z_{18}[4] \oplus tk_{18}[4] \oplus c_1) \oplus SC^{-1}(\overline{Z_{18}[4]} \oplus tk_{18}[4] \oplus c^1_{18})$ as shown in Fig. 7 , where $c^1_{18} =$ 0x03 is the 18-th round constant corresponding to c_1. Checking if $\Delta X_{18}[4] = \Delta tk_{17}[0] =$ 0xC0 will lead to an 8-bit filter and generally sieve $2^{N-80-8} = 2^{N-88}$ plaintext-ciphertext pairs. The time complexity of this step is $2^8 \cdot 2 \cdot 2^{N-80} \cdot \frac{1}{16} \cdot \frac{1}{18} \approx 2^{N-79.2}$ times of 18-round encryptions.
2) Line[24,25,26] in Algorithm 4. Guess $tk_{18}[0]$ and compute $\Delta X_{18}[0, 4, 8, 12]$ under 2^{N-88} remaining pairs from the previous step. According to Fig. 7, compute $\Delta X_{18}[0] = SC^{-1}(Z_{18}[0] \oplus tk_{18}[0] \oplus c^0_{18}) \oplus SC^{-1}(\overline{Z_{18}[0]} \oplus tk_{18}[0] \oplus c^0_{18})$, $\Delta X_{18}[4] = SC^{-1}(Z_{18}[4] \oplus tk_{18}[4] \oplus c^1_{18}) \oplus SC^{-1}(\overline{Z_{18}[4]} \oplus tk_{18}[4] \oplus c^1_{18})$, $\Delta X_{18}[8] = SC^{-1}(Z_{18}[8] \oplus c^2_{18}) \oplus SC^{-1}(\overline{Z_{18}[8]} \oplus c^2_{18})$, and $\Delta X_{18}[12] = SC^{-1}(Z_{18}[12]) \oplus SC^{-1}(\overline{Z_{18}[12]})$, where $c^0_{18} =$ 0x0A, $c^1_{18} =$ 0x03, and $c^2_{18} =$ 0x02 are the 18-th round constants corresponding to c_0, c_1, and c_2, respectively. Based on the properties of MC operations on col(1) of W_{17}, we have the conditions $\Delta X_{18}[0] = \Delta X_{18}[12]$ and $\Delta X_{18}[4] \oplus \Delta X_{18}[8] = \Delta X_{18}[12]$. Check whether the conditions are satisfied or not, which can act as a 16-bit filter to choose $2^{N-88-16} = 2^{N-104}$ remaining pairs. In this step, the time complexity is $2^{16} \cdot 2 \cdot 2^{N-88} \cdot \frac{4}{16} \cdot \frac{1}{18} \approx 2^{N-77.2}$ 18-round encryptions.
3) Line[27,28,29] in Algorithm 4. Guess $tk_1[2, 3, 7]$, $tk_2[2]$ and compute $\Delta Y_3[2]$ under 2^{N-104} remaining pairs from step 2). As shown in Fig. 7, $\Delta Y_3[2]$ is a fixed difference that equals to $\Delta tk_3[2] =$ 0x82. Checking if $\Delta Y_3[2] =$ 0x82 can act as an 8-bit filter and discard the wrong subtweakeys. The time complexity of this step is $2^{48} \cdot (2 \cdot 2^{N-104}) \cdot (\frac{3}{16} + \frac{1}{16}) \cdot \frac{1}{18} \approx 2^{N-61.2}$ 18-round encryptions.

In step 3), the number of the wrong subtweakeys $tk_1[2, 3, 7]$, $tk_2[2]$ and $tk_{20}[0, 4]$) left is

$$2^{\rho} = (2^{|K_{in} \cup K_{out}|} - 1) \times (1 - 2^{-8})^{2^{N-104}} = (2^{48} - 1) \times (1 - 2^{-8})^{2^{N-104}},$$

where we set $N = 112.6$ and $\rho = 45.8$.

Brute Force. The values of master key $MK[2, 3, 7, 8, 9, 10]$ can be derived from tk_{rem} using the tweakey schedule. For the remaining master key $MK[0, 1, 4, 5, 6, 11, 12, 13, 14, 15]$, we exhaustively traverse 2^{80} candidate values. We verify the master key by one plaintext-ciphertext pair under (l_0,N, MK). In this step, the time complexity is $2^{\rho} \times 2^{80} = 2^{\rho+80}$ 18-round encryptions.

Complexity Computation. For the 18-round attack on SKINNY-AEAD M1/M3, the data complexity is :

$$\mathcal{D} = 2^N \cdot (2^{|\Delta_{in}|} + 2^{|\Delta_{in}|}) = 2^N \cdot 2 \cdot 2^{|\Delta_{in}|} = 2^{N+9} = 2^{112.6+9} = 2^{121.6}$$

chosen plaintexts, the time complexity is :

$$\mathcal{T}_{cal} = 2^{N+9.7} + 2^{N-79.2} + 2^{N-77.2} + 2^{N-61.2} + 2^{\rho+80} \approx 2^{125.9}$$

18-round encryptions, and the memory complexity is :

$$\begin{aligned}\mathcal{M} &= \mathcal{M}_{col} + 2 \cdot (2^{N-88} + 2^{N-104}) \cdot (3 \cdot 128) + 2^{\rho} \cdot 128 \\ &= 2^{42.2} + 2^{34.2} + 2^{52.8} \approx 2^{52.8} \text{ bits.}\end{aligned}$$

6 Conclusion

In this paper, we analyze the security of SKINNY-AEAD M1/M3 using the related-tweakey impossible differential attack. Firstly, based on the encryption mode, nonce-respecting scenario and complex tweakey initialization of M1/M3, we propose some constraints to search for longer distinguishers and mount tweakey recovery attacks with less time complexity. Then, according to the constraints given in Section 3.1, we search for the related-tweakey impossible differential distinguishers with the help of automatic searching algorithm based on STP. As a result, we find some 14-round distinguishers. Finally, based on one of these distinguishers, 20-round and 18-round tweakey recovery attacks are conducted under a nonce-respecting scenario. To our knowledge, all the attacks are the best tweakey recovery attacks on these two ciphers so far.

The underlying primitive SKINNY-128-384 of SKINNY-AEAD M1/M3 have 128-bit security, while the block cipher SKINNY-128-384 has 384-bit security. Previous attack results (related-tweakey impossible differential, meet-in-the-middle, related-tweakey rectangle) cannot directly be used in SKINNY-AEAD M1/M3. This paper only investigates the related-tweakey impossible differential attacks on M1/M3. The security of SKINNY-AEAD M1/M3 against other attacks deserves further analysis.

Acknowledgements

This work has been supported by the National Natural Science Foundation of China (Grant No. 62032014 and Grant No. 62002204), the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025).

References

1. SKINNY-AEAD and SKINNY-Hash: NIST LWC second-round candidate status update (2020), https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/SKINNY-AEAD_and_SKINNY-Hash_status_update.pdf

2. Ankele, R., Banik, S., Chakraborti, A., List, E., Mendel, F., Sim, S.M., Wang, G.: Related-key impossible-differential attack on reduced-round skinny. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10355, pp. 208–228. Springer (2017), https://doi.org/10.1007/978-3-319-61204-1_11
3. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 825–885. IOS Press (2009), https://doi.org/10.3233/978-1-58603-929-5-825
4. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016), https://doi.org/10.1007/978-3-662-53008-5_5
5. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: SKINNY-AEAD and skinny-hash. IACR Trans. Symmetric Cryptol. **2020**(S1), 88–131 (2020), https://doi.org/10.13154/tosc.v2020.iS1.88-131
6. Biham, E.: New types of cryptanalytic attacks using related keys. J. Cryptol. **7**(4), 229–246 (1994), https://doi.org/10.1007/BF00203965
7. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Lecture Notes in Computer Science, vol. 1592, pp. 12–23. Springer (1999), https://doi.org/10.1007/3-540-48910-X_2
8. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to clefia, camellia, lblock and simon. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. vol. 8873, pp. 179–199 (2014), https://doi.org/10.1007/978-3-662-45611-8_10
9. Hadipour, H., Bagheri, N., Song, L.: Improved rectangle attacks on SKINNY and CRAFT. IACR Trans. Symmetric Cryptol. **2021**(2), 140–198 (2021), https://doi.org/10.46586/tosc.v2021.i2.140-198
10. Jakimoski, G., Desmedt, Y.: Related-key differential cryptanalysis of 192-bit key AES variants. In: Matsui, M., Zuccherato, R.J. (eds.) Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003, Revised Papers. Lecture Notes in Computer Science, vol. 3006, pp. 208–221. Springer (2003), https://doi.org/10.1007/978-3-540-24654-1_15
11. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 274–288. Springer (2014), https://doi.org/10.1007/978-3-662-45608-8_15

12. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 161–185. Springer (2015), https://doi.org/10.1007/978-3-662-47989-6_8
13. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Joux, A. (ed.) Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer (2011), https://doi.org/10.1007/978-3-642-21702-9_18
14. Li, M., Hu, K., Wang, M.: Related-tweak statistical saturation cryptanalysis and its application on QARMA. IACR Trans. Symmetric Cryptol. **2019**(1), 236–263 (2019), https://doi.org/10.13154/tosc.v2019.i1.236-263
15. Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings (long paper). IACR Trans. Symmetric Cryptol. **2017**(3), 37–72 (2017), https://doi.org/10.13154/tosc.v2017.i3.37-72
16. Liu, Y., Wang, Q., Rijmen, V.: Automatic search of linear trails in ARX with applications to SPECK and chaskey. In: Manulis, M., Sadeghi, A., Schneider, S.A. (eds.) Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings. Lecture Notes in Computer Science, vol. 9696, pp. 485–499. Springer (2016), https://doi.org/10.1007/978-3-319-39555-5_26
17. Longo, G., Zilli, M.V.: Complexity of theorem-proving procedures : some general properties. RAIRO Theor. Informatics Appl. **8**(3), 5–18 (1974), https://doi.org/10.1051/ita/197408R300051
18. Niu, C., Li, M., Sun, S., Wang, M.: Zero-correlation linear cryptanalysis with equal treatment for plaintexts and tweakeys. In: Paterson, K.G. (ed.) Topics in Cryptology - CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17-20, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12704, pp. 126–147. Springer (2021), https://doi.org/10.1007/978-3-030-75539-3_6
19. Sadeghi, S., Mohammadi, T., Bagheri, N.: Cryptanalysis of reduced round SKINNY block cipher. IACR Trans. Symmetric Cryptol. **2018**(3), 124–162 (2018), https://doi.org/10.13154/tosc.v2018.i3.124-162
20. Shi, D., Sun, S., Derbez, P., Todo, Y., Sun, B., Hu, L.: Programming the demirci-selçuk meet-in-the-middle attack with constraints. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11273, pp. 3–34. Springer (2018), https://doi.org/10.1007/978-3-030-03329-3_1
21. Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round SKINNY. In: Joye, M., Nitaj, A. (eds.) Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10239, pp. 117–134 (2017), https://doi.org/10.1007/978-3-319-57339-7_7
22. Zhao, B., Dong, X., Meier, W., Jia, K., Wang, G.: Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT. Des. Codes Cryptogr. **88**(6), 1103–1126 (2020), https://doi.org/10.1007/s10623-020-00730-1

A 18-Round Related-Tweakey Impossible Differential Attack for SKINNY-AEAD M1/M3

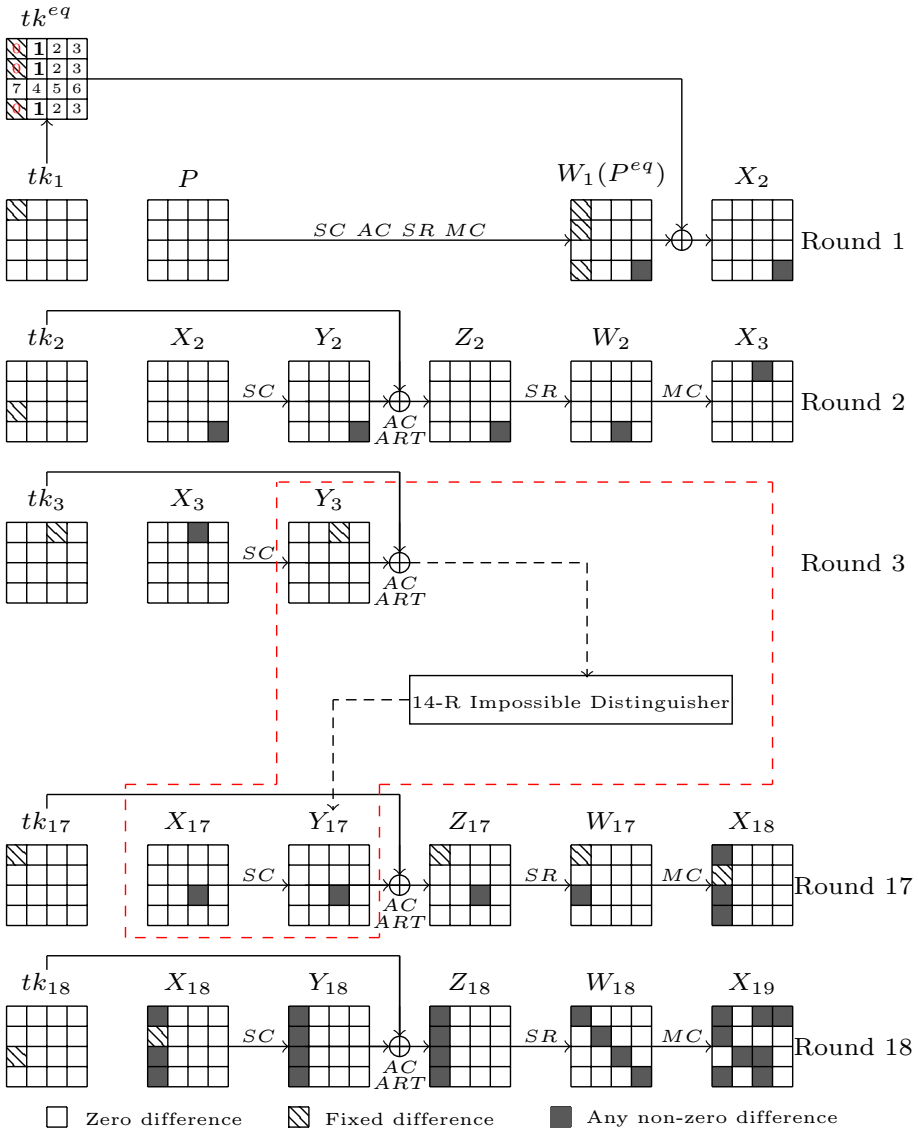

Fig. 7. 18-round related-tweakey impossible differential attack for SKINNY-AEAD M1/M3