# A Note on Related-Tweakey Impossible Differential Attacks

Xavier Bonnetain and Virginie Lallemand

Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
firstname.name@loria.fr

**Abstract.** In this short note we review the technique proposed at ToSC 2018 by Sadeghi *et al.* for attacks built upon several related-tweakey impossible differential trails. We show that the initial encryption queries are improper and lead the authors to misevaluating a filtering value in the key recovery phase. We identified 4 papers (from Eurocrypt, DCC, ToSC and ePrint) that follow on the results of Sadeghi *et al.*, and in three of them the issue was propagated.

We thus present a careful analysis of these types of attacks and give generic complexity formulas similar to the ones proposed by Boura *et al.* at Asiacrypt 2014. We apply these to the aforementioned papers and provide patched versions of their attacks. The main consequence is an increase in the memory complexity. We show that in many cases (a notable exception being quantum impossible differentials) it is possible to recover the numeric estimates of the flawed analysis, and in all cases we were able to build a correct attack reaching the same number of rounds.

**Keywords:** Impossible Differential Attack · Related-Tweakey · Complexity Analysis

## 1 Impossible Differential Attacks and their Complexity Analysis

We start by recalling the framework of single-key impossible differential attacks for block ciphers, as detailed in [BNS14]. We consider an $n$-bit block cipher with a $k$-bit key $K$. The attack is built around a probability-0 differential distinguisher of $r_d$ rounds that starts with a (set of) n-bit difference(s) $\Delta_X$ and ends with a (set of) n-bit difference(s) $\Delta_Y$, as depicted in Figure 1. Then, $r_b$ rounds are added before this differential and $r_f$ rounds are added after. The set of possible differences at the plaintext size, denoted $\Delta_{in}$, is a set of differences that might lead to $\Delta_X$ after $r_b$ rounds. Similarly, $\Delta_{out}$ represents the set of differences at the output size to which $\Delta_Y$ might propagate after $r_f$ rounds. We let $2^{-c_{in}}$ denote the probability that a difference in $\Delta_{in}$ leads to a difference in $\Delta_X$, and $2^{-c_{out}}$ the probability that a difference in $\Delta_{out}$ leads to a difference in $\Delta_Y$. The set of key bits used to compute if a difference in $\Delta_{in}$ leads to $\Delta_X$ is $k_{in}$, while the corresponding set of key bits in $r_f$ is $k_{out}$.

Given a pair of messages that satisfies $\Delta_{in}$ and $\Delta_{out}$, the probability that a key guess $(k_{in}, k_{out})$ leads to $\Delta_X$ and $\Delta_Y$ (and thus is discarded) is equal to $2^{-c_{in}-c_{out}}$. Not discarding a given key is thus of probability $(1 - 2^{-c_{in}-c_{out}})$, and not discarding a given key when considering $N$ pairs is thus $(1 - 2^{-c_{in}-c_{out}})^N \simeq \exp(-N \times 2^{-c_{in}-c_{out}})$.

The goal of an attacker is to reduce the set of possible keys by at least a factor of 2 in order to keep the final step (of exhaustively testing the remaining keys) of reasonable cost. We introduce the variable $g$ to measure the number of remaining keys and denote by

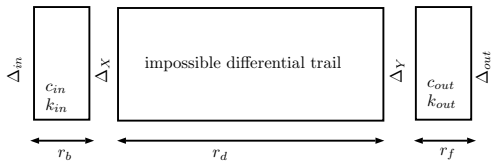


**Figure 1:** Setting and notation for an impossible differential attack on a block cipher.

$N^g_{min}$ the number of pairs satisfying $\Delta_{in}$ and $\Delta_{out}$ such that:

$$(1 - 2^{-c_{in}-c_{out}})^{N^g_{min}} < \frac{1}{2^g}.$$

The previous approximation using the exponential function leads to:

$$N^g_{min} > g \times \ln(2) \times 2^{c_{in}+c_{out}}.$$

To build a given amount of pairs $N$, an attacker organizes their encryption queries into structures (either at the plaintext or at the ciphertext side). Depending on the exact number of pairs that are required, either several structures are encrypted or less than one. These different scenarios make the data complexity be approximated by:

$$D = \max\left\{\min_{\Delta\in\{\Delta_{in},\Delta_{out}\}}\left\{\sqrt{N2^{n+1-|\Delta|}}\right\}, N2^{n+1-|\Delta_{in}|-|\Delta_{out}|}\right\}$$

where $2^{|\Delta_{in}|}$ and $2^{|\Delta_{out}|}$ represent the number of differences in $\Delta_{in}$ and $\Delta_{out}$, respectively. This number of encryption queries must be so that $D \leq 2^n$.

If we let $C_E$ denote the cost of one encryption, the lower bound of the time complexity of the attack ($T$) provided in [BNS14] is:

$$T = \left(D + \left(N + 2^{|k_{in}\cup k_{out}|}\frac{N}{2^{c_{in}+c_{out}}}\right)C'_E + 2^{k-g}\right)C_E,$$

where $C'_E$ is the ratio of the cost of partial encryption to the full encryption. This should satisfy $T \leq 2^k C_E$. The memory complexity corresponds to storing the $N$ pairs.

## 2 Multiple-Tweakey Attack from ToSC 2018

This section recalls the key recovery technique used in the 23-round related-tweakey impossible differential attack on SKINNY-$n$-$2n$ described in [SMB18].

**SKINNY.** As a reminder, SKINNY [BJK+16] is a family of tweakable block ciphers whose variants are denoted SKINNY-$n$-$v$, where $n = 16 \times s$ is the internal state size, $s = 4$ or 8 is the word size and $v$ is the tweakey size (it can be equal to $n, 2n$ or $3n$). The round function of SKINNY is recalled in Figure 2 and relies on 5 operations: `SubCells` (SC) which is an Sbox application on each word, `AddConstants` (AC) which adds a constant on words 0, 4 and 8, `AddRoundTweakey` (ART) which adds the round tweakey to the first and second rows, and `ShiftRows` (SR) and `MixColumns` (MC) which are two linear operations corresponding to a right shift of the rows and a multiplication of each column by a binary matrix, respectively. Further details, in particular on the linear tweakey schedule producing the round tweakeys ($TK1, TK2, \dots$) from the master tweakey states `TK1`, `TK2`, `TK3`, can be found in [BJK+16].

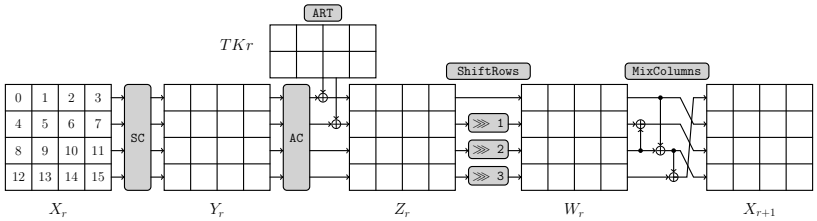


**Figure 2:** Word numbering and round function of SKINNY (modified version of [Jea16]).

**Description of the 23-round related-tweakey impossible differential attack on SKINNY-n-2n of [SMB18].** We reuse the notations from [SMB18]. The attack is based on the 15-round related-tweakey impossible differential trail represented in Figure 4. This distinguisher is positioned between $Y_4$ and $X_{19}$ and a total of 23 rounds is attacked by adding 3 rounds at the top and 5 rounds at the bottom for the key-recovery phase, as detailed in Figure 5. The first round tweakey addition is moved to the end of the round (just before $X_2$) by defining an equivalent tweakey for the first round as $ETK = \mathrm{MC}(\mathrm{SR}(TK_1))$. Since all the operations made before do not rely on secret values, the attacker considers an equivalent plaintext at position $Y_1$ (see Figure 5) that corresponds to the state before the equivalent tweakey addition.

As can be seen in Figure 4, the distinguisher starts after the `SubCells` operation and relies on a difference cancellation between the difference of the internal state and the one in the round tweakey. The difference in the master tweakey is set so that the 3 next rounds are blank rounds, and the last round of the distinguisher also relies on a cancellation between the internal state difference and the round tweakey difference.

The idea of the authors of [SMB18] is to use the set of all the possible distinguishers of this shape instead of only one. The 3 blank rounds require an inactive round tweakey in the round 3 of Figure 4, that translates into the relation $\Delta\texttt{TK1}[1] \oplus LFSR_2(\texttt{TK2}[1]) = \texttt{0xn} \oplus LFSR_2(\texttt{p}) = 0$. There is a total of $2^s - 1$ pairs of such (non-zero) working tweakey differences, that uniquely determine the master tweakey differences together with the internal state input and output differences of the distinguisher.

The authors of [SMB18] define the "Tweakey Differentials Table" (TDT) (reproduced in Figure 3 for the case $s = 4$) to store the $2^s - 1$ sets of valid round tweakey differences in rounds 2, 4, 18, 20 and 22. They use $2^s - 1$ lists $(L_i)$ to store the data corresponding to each distinguisher.

| | $L_1$ | $L_2$ | $L_3$ | $L_4$ | $L_5$ | $L_6$ | $L_7$ | $L_8$ | $L_9$ | $L_{10}$ | $L_{11}$ | $L_{12}$ | $L_{13}$ | $L_{14}$ | $L_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $TDT$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $TK2[7]$ 1 | 9 | 3 | A | 6 | F | 5 | C | 4 | D | 7 | E | 2 | B | 1 | 8 |
| | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| $TK4[1]$ 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ |
| $TK18[7]$ 3 | 7 | F | 8 | E | 9 | 1 | 6 | B | C | 4 | 3 | 5 | 2 | A | D |
| | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| $TK20[1]$ 4 | D | A | 7 | 5 | 8 | F | 2 | 6 | B | C | 1 | 3 | E | 9 | 4 |
| | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| $TK22[0]$ 5 | 8 | 1 | 9 | 2 | A | 3 | B | C | 4 | D | 5 | E | 6 | F | 7 |

**Figure 3:** Screenshot of the TDT from [SMB16].

We transcribe below the beginning of the key recovery procedure proposed in [SMB18] and provide our comments in Section 3.

The attacker queries $2^x$ structures of $2^{|\Delta_{in}|} = 2^{4s}$ messages $Y_1$ taking all the possible values in cells $5, 7, 8$ and $15$ and being fixed in the other positions. A total of $2^{x+8s}$ pairs of messages $(P, \bar{P})$ and their associated $(C, \bar{C})$ are built from these $2^{x+4s+1}$ initial messages $(D = 2^{x+|\Delta_{in}|+1})$. As $|\Delta_{out}| = n$, these pairs are not further filtered on their ciphertext differences and can all be used to discard wrong key guesses.

Once these pairs have been generated, the attacker guesses the value of $ETK[7]$ to compute the difference in the cell $Y_2[7]$ of each pair of plaintexts. By looking for the index $i$ such that $\Delta Y_2[7] = TDT[1][i]$ in the TDT, the attacker deduces in which list $L_i$ to store the pair. This identifies which of the $2^s - 1$ impossible trails is considered for this key guess.

The TDT is also used in round 4, where the attacker needs to check if the correct difference happens at the start of the distinguisher. Namely, it corresponds to checking that $\Delta Y_4[1] = TDT[2][i]$ (and if not, to discard the pair) as this is the required condition to have a cancellation.

We do not transcribe here the remainder of the key recovery as it is standard and not relevant to our discussion.

The claimed complexities of the 23-round attacks are of $D = 2^{62.47}$ chosen plaintexts for SKINNY-64-128, and $D = 2^{124.41}$ for SKINNY-128-256. The time complexities (expressed in number of encryptions) are respectively of $T = 2^{124.21}$ and $T = 2^{243.61}$.

# 3 Analysis of the Key-Recovery in [SMB18]

There are a few interlinked issues in the previous process.

**Data generation.** First, the authors mention a total of $2^{x+|\Delta_{in}|+1}$ encryption queries corresponding to $2^x$ structures of $2^{|\Delta_{in}|}$ messages that can form a total of $2^{x+2|\Delta_{in}|}$ pairs. This presuppose that each plaintext is encrypted twice, under two different tweaks.

**TDT.** The TDT misses that the set of relevant tweak differences is actually a vector space (if we add the 0 difference) and not an arbitrary set of values. Such vector space stems from the linearity of the tweakey schedule and of the condition it needs to fulfill. It implies that the queries of the plaintexts under several tweaks can be done efficiently, as will be detailed next section.

**On the "choice" of $i$.** Following the first key guess (of $ETK[7]$), the attacker computes $\Delta Y_2[7]$ for each pair of plaintexts $(P, \bar{P})$ and deduces from it the impossible differential trail that is used for this pair by selecting $i$ and putting the pair in the corresponding list $L_i$, where $i$ is such that $\Delta Y_2[7] = TDT[1][i]$. This means the tweak difference depends on the value of $ETK[7]$. Thus, the approach breaks if $(P, \bar{P})$ are each encrypted using a unique tweak.

**Filtering factor at round 2.** In the initial attack, the first cancellation between $\Delta Y_2[7]$ and $\Delta TK2[7]$ happens with probability 1 as $\Delta TK2[7]$ is chosen with $i$. We showed in the previous point that $i$ cannot be chosen and that $\Delta TK2[7]$ is fixed with the pair, so actually checking this cancellation creates a filter of $2^{-s}$ of the pairs.

Following these observations we propose a new algorithm in the next section and reassess the attacks of [SMB18] together with the attacks proposed in 4 papers that are based on it ([HSE23, HGSE23, BDL20, DNPS23]).

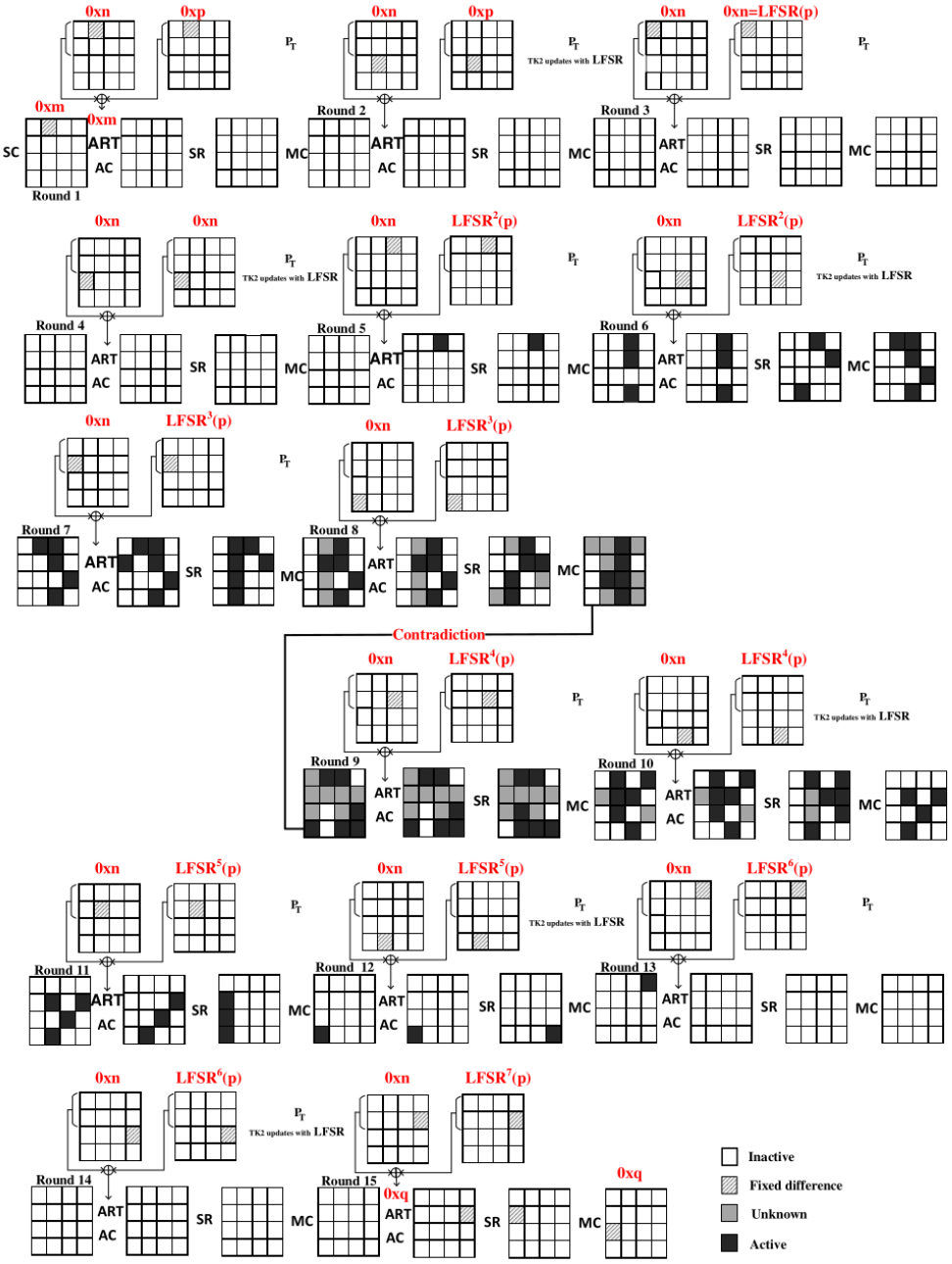


**Figure 4:** Screenshot of the 15-round related-tweakey impossible distinguisher for SKINNY-$n$-$2n$ from [SMB18].

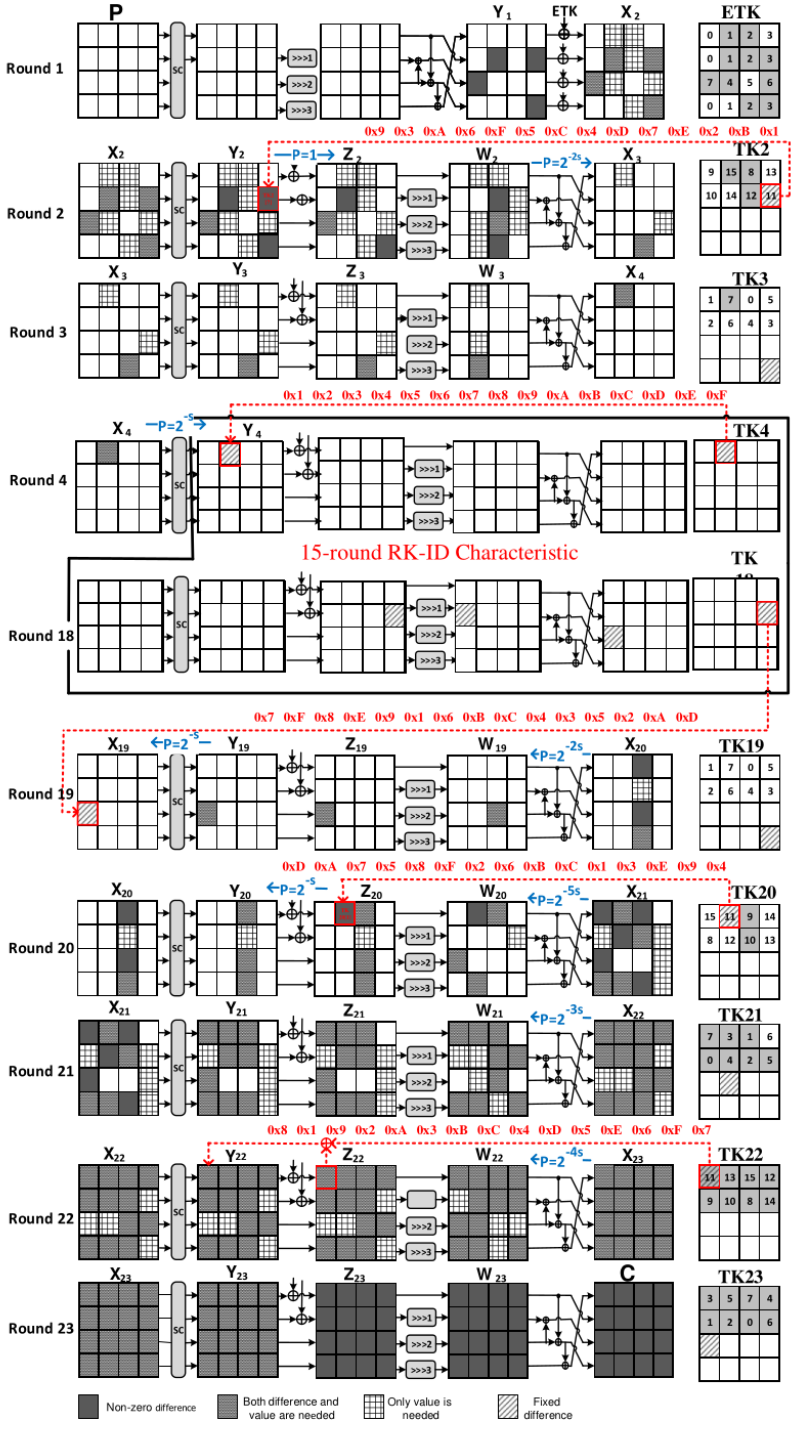


**Figure 5:** Screenshot of the 23-round related-tweakey impossible attack for SKINNY-$n$-$2n$ from [SMB18].

# 4 Our Corrected Key-Recovery

In this section we describe a technique to organize an attack taking advantage of a valid set of impossible differential trails based on several tweakey differences. We start by discussing how to modify the 23-round attack of [SMB18] and next provide generic complexity formulas.

## 4.1 Principle

The encryption queries must allow to take advantage of the set of $2^s - 1$ impossible differential distinguishers. As a reminder, the queries the attacker makes are for ciphertexts corresponding to a given plaintext value $P$, under the secret master key $K$ and the (master) tweak value $T$. For the previous attack on SKINNY-n-2n the queries have the form $(P, \Delta\texttt{TK1}, \Delta\texttt{TK2})$ (where actually the tweakeys are made of key material only).

As both the tweakey schedule and the condition to obtain the 3 blank rounds are linear, the set of relevant tweak differences is actually a vector space. The attacker can thus exploit this mathematical pattern by querying structures of plaintexts under a set of tweakey values that take all possible values on the affine coset of the space of tweak differences. Similar to the original attack presented in [SMB18], the considered plaintext values are organized in one structure where all the possible values for the 4 active cells of $Y_1$ are spanned while the other cells are fixed to a given value. What is different is that these $2^{4s}$ messages are each encrypted under *a set of* $2^s$ *tweakeys*.

By pairing any two different messages encrypted under any two different master tweakeys, an attacker obtains a valid message difference and a valid tweakey difference. A total of approximately $2^{4s+s} \times 2^{4s+s} \times 2^{-1} \approx 2^{10s-1}$ such (unordered) pairs are obtained from $2^{4s+s}$ initial encryption queries. This is thus repeated $2^x$ times to get enough pairs for the attack.

Note that this change can be seen as increasing the size of the structure by using the additional degree of freedom on the tweakey.

In summary, the first point of Section 3 is fixed with this new data generation, the second point (TDT use) is replaced by the subspace of tweakeys, which also fixes the value of the previous "$i$" of point 3. The filtering factor (point 4) needs to be paid.

## 4.2 Generic Formulas

We consider the same notation as previously and in particular denote by $|\Delta_{in}|$ the number of active bits at the plaintext side, by $|\Delta_{out}|$ the number of active bits at the ciphertext side and by $n$ the block size. We introduce $|\Delta_t|$ to denote the number of active tweakey bits. We assume here for simplicity that all the $2^{|\Delta_t|}$ tweak differences correspond to a valid impossible differential trail, but the formulas can easily be adapted to cover more complex cases.

The probability to keep a key is, as in Section 1,

$$(1 - 2^{-c_{in}-c_{out}})^N \simeq \exp\left(-N \times 2^{-c_{in}-c_{out}}\right),$$

where $N$ is the number of pairs with an input difference in $\Delta_{in}$, an output difference in $\Delta_{out}$ and a tweakey difference in $\Delta_t$. Again we can choose the number of pairs $N^g_{min}$ such that $N^g_{min} > g \times \ln(2) \times 2^{c_{in}+c_{out}}$ to get $(1 - 2^{-c_{in}-c_{out}})^{N^g_{min}} < \frac{1}{2^g}$.

To build these pairs, the attacker organizes the (plaintext or ciphertext) queries in $2^x$ structures of $2^{|\Delta_{in}|+|\Delta_t|}$ (resp. $2^{|\Delta_{out}|+|\Delta_t|}$) encryption queries each, corresponding to the encryption of all the possible plaintexts (resp. ciphertexts) with varying values on the active positions (and a fixed value on the other cells) under all the $2^{|\Delta_t|}$ master tweakey differences of the linear subspace.

If more than one full structure is necessary ($2^x \geq 1$), with $2^{x+|\Delta_{in}|+|\Delta_t|}$ encryptions, the attacker can approximately build $2^x \times \binom{2^{|\Delta_{in}|+|\Delta_t|}}{2} \approx 2^{x+2|\Delta_{in}|+2|\Delta_t|-1}$ pairs with the correct $\Delta_{in}$ and $\Delta_t$. Otherwise, if only a portion of $2^x < 1$ of a structure is required, with $2^{x+|\Delta_{in}|+|\Delta_t|}$ encryptions the attacker can build around $\binom{2^{x+|\Delta_{in}|+|\Delta_t|}}{2} \approx 2^{2(x+|\Delta_{in}|+|\Delta_t|)-1}$ pairs with the correct[1] $\Delta_{in}$ and $\Delta_t$.

Only the pairs with a ciphertext difference lying in $\Delta_{out}$ are of interest, so we have:

$$N = \begin{cases} 2^{x+2|\Delta_{in}|+2|\Delta_t|-1-(n-|\Delta_{out}|)} & \text{if } 2^x \geq 1 \\ 2^{2x+2|\Delta_{in}|+2|\Delta_t|-1-(n-|\Delta_{out}|)} & \text{if } 2^x \leq 1. \end{cases}$$

Consequently, $2^x$ and thus the data complexity $D$ should be chosen so that:

$$D = \begin{cases} 2^{x+|\Delta_{in}|+|\Delta_t|} \approx g\ln(2)2^{c_{in}+c_{out}-|\Delta_{in}|-|\Delta_t|+1+n-|\Delta_{out}|} & \text{if } 2^x \geq 1 \\ 2^{x+|\Delta_{in}|+|\Delta_t|} \approx \min_{\Delta\in\{\Delta_{in},\Delta_{out}\}}\{\sqrt{g\ln(2)2^{c_{in}+c_{out}+1+n-|\Delta|}}\} & \text{if } 2^x \leq 1. \end{cases}$$

There is finally the cost of guessing and filtering the pairs. This approach is identical to the single-tweak case. Using early-abort and the heuristic from [BNS14], we can estimate it to cost

$$N2^{|k_{in}\cup k_{out}|-c_{in}-c_{out}}.$$

The exhaustive search of the remaining key bits costs $2^{k-g}$.

### 4.3 Differences with [SMB18]

If there are $2^{|\Delta_t|}$ possible tweak differences, assuming the initial attack uses at least $2^{|\Delta_t|}$ structures, the correction is:

- Instead of encrypting each plaintext under two different tweaks, it is encrypted under $2^{|\Delta_t|}$ tweaks.
- The tweak constraint is no longer free in our model, and divides the number of pairs by $2^{|\Delta_t|}$ (that is, in general formulas, we need to add $|\Delta_t|$ to $c_{in}$ or $c_{out}$).

Overall, the main change is the number of pairs, which is significantly increased. Every other parameters, including the data complexity, can be reused as-is. Interestingly, the part of the key recovery after checking the tweak constraints are identical. Thus, in some regimes, the estimated time cost is not changed, and the only difference is the larger memory footprint, as pairs need to be stored in memory for filtering.

## 5 Impact on Concrete Key-Recoveries

Including the original [SMB18], we identified 5 articles [SMB18, BDL20, DNPS23, HSE23, HGSE23] that build upon this technique. Among them, 3 explicitly reuse the key-recovery technique, while 2 apply the original SKINNY related-tweakey distinguisher in different contexts. All theses attacks are against variants of SKINNY. For clarity, we express the attack parameters in function of $s$.

[1] Note that a refinement of these approximations can be obtained to take into account the fact that the tweak must be active, by multiplying the previous formula by $\frac{2^{|\Delta_t|}-1}{2^{|\Delta_t|}}$. In particular we will use this in the case $|\Delta_t| = 1$, as it corresponds to a factor $\frac{1}{2}$.

## 5.1 Revisiting the Original Article [SMB18]

In addition to the 23-round attack that we detailed in Section 2, Sadeghi *et al.* proposed a 19-round related tweakey impossible differential attack on SKINNY-n-n (see [SMB18, Appendix A]) that uses the same technique. By applying the formulas from Section 4.2 to both these attacks (aiming for the same success probability as in [SMB18]), we obtain the patched parameters and complexities as presented in Table 1.

**Table 1:** Claimed and patched parameters and costs for the attacks of [SMB18].

| Attack | Version | $\lvert\Delta_{in}\rvert$ | $c_{in}$ | $\lvert\Delta_{out}\rvert$ | $c_{out}$ | $\lvert\Delta_t\rvert$ | $x$ | $D$ | $N$ | $T$ |
|---|---|---|---|---|---|---|---|---|---|---|
| [SMB18] | 64-64 | 4s | 4s | 9s | 8s | 1 | 44.3 | $2^{61.3}$ | $2^{48.3}$ | $2^{62.83}$ |
| | 128-128 | | | | | | 89.47 | $2^{122.47}$ | $2^{97.47}$ | $2^{124.43}$ |
| | 64-128 | 4s | 3s | 16s | 16s | 1 | 45.47 | $2^{62.47}$ | $2^{77.47}$ | $2^{124.21}$ |
| | 128-256 | | | | | | 91.40 | $2^{124.41}$ | $2^{155.41}$ | $2^{243.61}$ |
| patch | 64-64 | 4s | 4s | 9s | 9s | s | 41.3 | $2^{61.3}$ | $2^{52.3}$ | $2^{62.83}$ |
| | 128-128 | | | | | | 82.47 | $2^{122.47}$ | $2^{105.47}$ | $2^{124.43}$ |
| | 64-128 | 4s | 4s | 16s | 16s | s | 42.47 | $2^{62.47}$ | $2^{81.47}$ | $2^{124.21}$ |
| | 128-256 | | | | | | 84.40 | $2^{124.40}$ | $2^{163.40}$ | $2^{243.61}$ |

Note that it's the error in the $c_{in}/c_{out}$ value that creates the error in $N$.

## 5.2 Attack Presented at Eurocrypt 2023 [HSE23]

The article [HSE23] presents a new CP-based method to search for impossible-differential, integral and zero-correlation attacks. Among the impossible differential attacks that are presented, only two attacks are related-tweakey attacks that use the framework from [SMB18]. The two are almost-identical, with distinct targets: SKINNY-64-192 and SKINNY-128-384. They are both detailed in the full version of the article [HSE22, Appendix F.4].

As with the previous application, we need to consider $|\Delta_t| = s$ and an increased $c_{in}$. The changes in the attack parameters are detailed in Table 2. While the increase of $c_{in}$ and thus of $N$ impacts the key-recovery detailed in [HSE23], its first step is to tackle the tweak difference. This first filtering step has a negligible cost compared with later steps and hence, the overall cost of the pair filtering is the same. In the end, the time and data complexities of our patched version matches the ones obtained with the flawed technique.

**Table 2:** Claimed and patched parameters and costs for the attacks from [HSE23, Table 1]. † The caption of [HSE22, Fig. 10] claims $c_{in} = 4s$. This is however inconsistent with the computations the page before and likely a typo.

| Attack | Version | $\lvert\Delta_{in}\rvert$ | $c_{in}$ | $\lvert\Delta_{out}\rvert$ | $c_{out}$ | $\lvert\Delta_t\rvert$ | $x$ | $D$ | $N$ | $T$ |
|---|---|---|---|---|---|---|---|---|---|---|
| [HSE23] | 64-192 | 4s | 3s† | 16s | 16s | 1 | 46.64 | $2^{63.64}$ | $2^{78.64}$ | $2^{183.26}$ |
| | 128-384 | | | | | | 91.99 | $2^{124.99}$ | $2^{155.99}$ | $2^{362.61}$ |
| patch | 64-192 | 4s | 4s | 16s | 16s | s | 43.64 | $2^{63.64}$ | $2^{82.64}$ | $2^{183.26}$ |
| | 128-384 | | | | | | 84.99 | $2^{124.99}$ | $2^{163.99}$ | $2^{362.61}$ |

### 5.3 Follow-up of the Eurocrypt Article [HGSE23]

A follow-up of the previous article was posted on ePrint [HGSE23] in November 2023. Among other things, it proposes an extension of the previous CP model that covers bit-oriented ciphers and that does not require the attacker to set the contradiction point of the impossible differential distinguisher.

The authors applied this improved model to various variants of SKINNY (SKINNY, ForkSKINNY and SKINNYe-v2) and proposed 15 impossible differential attacks. As with the previous article, the key-recovery technique of [SMB18] is explicitly used. For most of the attacks the time cost is unaffected. Still, for 2 of them the updated costs are slightly above what was initially claimed. Our results are summarized in Table 3.

We communicated our observations to the authors of the preprint who confirmed our findings and added a discussion in the published version of their article (see Appendix A of [HGSE24]).

### 5.4 Cryptanalysis of ForkSKINNY [BDL20]

In [BDL20], Bariant *et al.* proposed two attacks on ForkSKINNY-128-256. The first one, which attacks the 128-bit key version, reuses as-is the attack against 19-rounds SKINNY-128-128 from [SMB18] to attack 24-round ForkSKINNY. Thus, the issues of the previous section also convey to this attack, and the correction is exactly the same.

The second attack, which targets the 256-bit key version, relies on an extension of the distinguisher described in [SMB18] on SKINNY-128-256. By taking advantage of ForkSKINNY's structure, the authors add 3 blank rounds to the distinguisher and are able to attack 26 rounds of ForkSKINNY with $r_i = 7$, $r_0 = 27$ and $r_1 = 19$ rounds.

Given that their distinguisher has a distinct input difference, the authors made their own key recovery. The proposed data generation step begins with the encryption of each structure under several keys, as in the correction we propose.

Their distinguisher requires an initial tweakey difference that makes the round tweakeys $TK_6$ anf $TK_9$ inactive which corresponds to selecting $\delta \in \mathbb{F}_2^s$ so that $\delta \oplus LFSR^{15}(\delta) = 0$. For both $s = 4$ and $s = 8$ there are 15 such solutions $\delta$ (and hence there are 15 related-key impossible differential trails for 18 rounds), and as the condition is linear the solutions form a linear subspace.

We agree with the parameters and with the complexity analysis made by the authors.

### 5.5 Quantum Impossible Differential Attacks [DNPS23]

The article [DNPS23] proposes a framework for quantum impossible attacks, and gives as an application a quantum attack heavily inspired by [SMB18], but with 2 less rounds in the output. Unfortunately, they reuse as-is the parameters for the input, that is, there is no tweak variation and a cancellation is free. While the authors use these flawed parameters to estimate the number of pairs, they also describe a detailed key-recovery, which is correct (and in particular is inconsistent with the claimed value for $c_{in}$). The authors also seek enough filtering to directly obtain the correct $k_{in} \cup k_{out}$ subkey.

As quantum computing is not the focus of this paper, we refer to [DNPS23] for details and formulas. In particular, the quantum framework can be used with our key recovery, it only amounts in changing some parameters in their formulas.

We can patch the attack by adding $|\Delta_t| = s$ to the initial $|\Delta_{in}|$ (each plaintext is encrypted with all tweaks) and $c_{in}$. We can then reuse their formulas to obtain an updated quantum cost. This cost corresponds to what the authors would have obtained had they used correct parameters.

**Table 3:** Claimed and updated parameters and costs for the attacks from [HGSE23]. Patched variants keep the same data complexity, reoptimized variants change it to minimize the time.

| Cipher | rounds | Version | $\lvert\Delta_{in}\rvert$ | $c_{in}$ | $\lvert\Delta_{out}\rvert$ | $c_{out}$ | $\lvert\Delta_t\rvert$ | $x$ | $D$ | $N$ | $T$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ForkSKINNY 64-192 | 28 | [HGSE23, G.1] | $13s$ | $11s$ | $14s$ | $14s$ | 1 | 8 | $2^{60}$ | $2^{104}$ | $2^{169.6}$ |
| | | patch | $13s$ | $13s$ | $14s$ | $14s$ | $2s$ | 1.2 | $2^{61.2}$ | $2^{112}$ | $2^{169.6}$ |
| | | [HGSE23, G.2] | $6s$ | $5s$ | $16s$ | $16s$ | 1 | 38 | $2^{62}$ | $2^{86}$ | $2^{123.73}$ |
| | | patch | $6s$ | $6s$ | $16s$ | $16s$ | $s$ | 35 | $2^{63}$ | $2^{90}$ | $2^{123.73}$ |
| | 30 | [HGSE23, G.4] | $6s$ | $5s$ | $16s$ | $16s$ | 1 | 38 | $2^{62}$ | $2^{86}$ | $2^{123.73}$ |
| | | patch | $6s$ | $6s$ | $16s$ | $16s$ | $s$ | 35 | $2^{63}$ | $2^{90}$ | $2^{123.73}$ |
| | 32 | [HGSE23, G.3] | $13s$ | $12s$ | $16s$ | $16s$ | 1 | 10 | $2^{62}$ | $2^{114}$ | $2^{186.27}$ |
| | | patch | $13s$ | $13s$ | $16s$ | $16s$ | $s$ | 7 | $2^{63}$ | $2^{118}$ | $2^{186.27}$ |
| ForkSKINNY 128-256 | 20 | [HGSE23, G.8] | $5s$ | $4s$ | $10s$ | $7s$ | 1 | 61 | $2^{101}$ | $2^{93}$ | $2^{102.2}$ |
| | | patch | $5s$ | $5s$ | $10s$ | $7s$ | $s$ | 54 | $2^{102}$ | $2^{101}$ | $2^{107.26}$ |
| | | reoptimized | $5s$ | $5s$ | $10s$ | $7s$ | $s$ | 53.2 | $2^{101.2}$ | $2^{100.2}$ | $2^{106.5}$ |
| | 24 | [HGSE23, G.6] | $8s$ | $6s$ | $8s$ | $8s$ | 1 | 54.4 | $2^{118.4}$ | $2^{118.4}$ | $2^{123.17}$ |
| | | patch | $8s$ | $7s$ | $8s$ | $8s$ | $s/2$ | 51.4 | $2^{119.4}$ | $2^{122.4}$ | $2^{126.83}$ |
| | | reoptimized | $8s$ | $7s$ | $8s$ | $8s$ | $s/2$ | 50 | $2^{118}$ | $2^{121}$ | $2^{126.27}$ |
| | | [HGSE23, G.7] | $8s$ | $7s$ | $12s$ | $12s$ | 1 | 62.7 | $2^{126.7}$ | $2^{158.7}$ | $2^{246.62}$ |
| | | patch | $8s$ | $8s$ | $12s$ | $12s$ | $s/2$ | 59.7 | $2^{127.7}$ | $2^{162.7}$ | $2^{246.62}$ |
| | 26 | [HGSE23, G.5] | $13s$ | $12s$ | $9s$ | $9s$ | 1 | 23.6 | $2^{127.6}$ | $2^{175.6}$ | $2^{238.5}$ |
| | | patch | $13s$ | $13s$ | $9s$ | $9s$ | $s/2$ | 20.6 | $2^{128.6}$ | $2^{179.6}$ | $2^{238.5}$ |
| ForkSKINNY 128-288 | 26 | [HGSE23, G.12] | $s$ | 0 | $8s$ | $8s$ | 1 | 116.6 | $2^{124.5}$ | $2^{68.6}$ | $2^{126.74}$ |
| | | patch | $s$ | $s$ | $8s$ | $8s$ | $s/2$ | 113.6 | $2^{125.6}$ | $2^{72.6}$ | $2^{126.74}$ |
| | 28 | [HGSE23, G.10] | $8s$ | $7s$ | $5s$ | $5s$ | 1 | 60.8 | $2^{124.8}$ | $2^{100.8}$ | $2^{126.68}$ |
| | | patch | $8s$ | $8s$ | $5s$ | $5s$ | $s/2$ | 57.8 | $2^{125.8}$ | $2^{104.8}$ | $2^{126.67}$ |
| | | [HGSE23, G.11] | $7s$ | $6s$ | $13s$ | $13s$ | 1 | 70.9 | $2^{126.9}$ | $2^{158.9}$ | $2^{277.23}$ |
| | | patch | $7s$ | $7s$ | $13s$ | $13s$ | $s/2$ | 67.9 | $2^{127.9}$ | $2^{162.9}$ | $2^{277.23}$ |
| | 31 | [HGSE23, G.9] | $8s$ | $7s$ | $16s$ | $16s$ | 1 | 62.5 | $2^{126.5}$ | $2^{190.5}$ | $2^{280.5}$ |
| | | patch | $8s$ | $8s$ | $16s$ | $16s$ | $s/2$ | 59.5 | $2^{127.5}$ | $2^{194.5}$ | $2^{280.5}$ |
| SKINNY 128-288 | 23 | [HGSE23, H.2] | $7s$ | $6s$ | $5s$ | $5s$ | 1 | 64.8 | $2^{120.8}$ | $2^{88.8}$ | $2^{126.73}$ |
| | | patch | $7s$ | $7s$ | $5s$ | $5s$ | $s$ | 57.8 | $2^{121.8}$ | $2^{96.8}$ | $2^{126.73}$ |
| | 26 | [HGSE23, H.1] | $9s$ | $8s$ | $16s$ | $16s$ | 1 | 50 | $2^{122}$ | $2^{194}$ | $2^{286.38}$ |
| | | patch | $9s$ | $9s$ | $16s$ | $16s$ | $s$ | 43 | $2^{123}$ | $2^{202}$ | $2^{286.38}$ |
| SKINNYe-v2 | 31 | [HGSE23, H.3] | $12s$ | $11s$ | $16s$ | $16s$ | 1 | 14 | $2^{62}$ | $2^{110}$ | $2^{251.14}$ |
| | | patch | $12s$ | $12s$ | $16s$ | $16s$ | $s$ | 11 | $2^{63}$ | $2^{114}$ | $2^{251.14}$ |

**Optimization.** We also propose an improved variant, that contains the following changes:

- The rejection probability of a key is estimated more precisely, using the natural logarithm,
- As the first filtering step in their key recovery does not depend on any key guess, it is moved to the pair generation part.

Our results are summarized in Table 4.

**Table 4:** Parameters and costs for the attacks. $s = 8$ bits. Note that the "no QRAM" attack requires classical pair generation in $2^{128}$ classical time and queries. † Using the formulas from [DNPS23] we obtain $2^{114.46}$. We believe this was a typo.

| Attack | $\lvert\Delta_{in}\rvert + \lvert\Delta_t\rvert$ | $c_{in}$ | $\lvert\Delta_{out}\rvert$ and $c_{out}$ | $N$ | Pair generation cost (with QRAM) | Pair filtering cost | |
|---|---|---|---|---|---|---|---|
| | | | | | | With QRAM | no QRAM |
| Original | $4s$ | $3s$ | $9s$ | $2^{103.17}$ | $2^{119.17}$ | $2^{104.32}$ | $2^{117.46}$† |
| Direct patch | $5s$ | $4s$ | $9s$ | $2^{111.17}$ | $2^{121.84}$ | $2^{112.32}$ | $2^{122.70}$ |
| Optimized | $5s$ | $4s$ | $8s$ | $2^{102.64}$ | $2^{118.64}$ | $2^{103.79}$ | $2^{114.15}$ |

## 6 Conclusion

This short note discusses a problem in the key recovery procedure that has been proposed in [SMB18] and that was later reused in [HSE23, HGSE23, BDL20] and [DNPS23]. It also suggests a technique to fix this procedure, that we applied on all the identified attacks.

Shortly after noticing the problem, we shared an early version of this note with the authors of the 5 papers. Notably, the authors of [HGSE23] confirmed our results and provided a short discussion of the impact of our fix on their automated tool in the final version of their article, published at ToSC 2024 issue 1 (see [HGSE24, Appendix A]).

## Acknowledgments

The authors would like to thank the authors of [SMB18, HSE23, HGSE23, BDL20] and [DNPS23] for their valuable feedback.

This work has been partially supported by the French Agence Nationale de la Recherche through the OREO project under Contract ANR-22-CE39-0015, and through the France 2030 program under grant agreement No. ANR-22-PECY-0010 CRYPTANALYSE.

## References

[BDL20] Augustin Bariant, Nicolas David, and Gaëtan Leurent. Cryptanalysis of forkciphers. IACR Trans. Symm. Cryptol., 2020(1):233–265, 2020.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, CRYPTO 2016, Part II, volume 9815 of LNCS, pages 123–153. Springer, Heidelberg, August 2016.

[BNS14] Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In Palash Sarkar and Tetsu Iwata, editors, ASIACRYPT 2014, Part I, volume 8873 of LNCS, pages 179–199. Springer, Heidelberg, December 2014.

[DNPS23] Nicolas David, María Naya-Plasencia, and André Schrottenloher. Quantum impossible differential attacks: Applications to aes and skinny. Designs, Codes and Cryptography, pages 1–29, 2023.

[HGSE23] Hosein Hadipour, Simon Gerhalter, Sadegh Sadeghi, and Maria Eichlseder. Improved search for integral, impossible-differential and zero-correlation attacks: Application to ascon, forkskinny, skinny, mantis, PRESENT and qarmav2. Cryptology ePrint Archive, Report 2023/1701, version 20231123:125414, 2023. https://eprint.iacr.org/archive/2023/1701/20231123:125414.

[HGSE24] Hosein Hadipour, Simon Gerhalter, Sadegh Sadeghi, and Maria Eichlseder. Improved search for integral, impossible differential and zero-correlation attacks: Application to ascon, forkskinny, skinny, mantis, present and qarmav2. IACR Transactions on Symmetric Cryptology, 2024(1):234–325, Mar. 2024.

[HSE22] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible differential, zero-correlation, and integral attacks. Cryptology ePrint Archive, Report 2022/1147, 2022. https://eprint.iacr.org/2022/1147.

[HSE23] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In Carmit Hazay and Martijn Stam, editors, EUROCRYPT 2023, Part IV, volume 14007 of LNCS, pages 128–157. Springer, Heidelberg, April 2023.

[Jea16] Jérémy Jean. TikZ for Cryptographers. https://www.iacr.org/authors/tikz/, 2016.

[SMB16] Sadegh Sadeghi, Tahere Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. Cryptology ePrint Archive, Report 2016/1120, 2016. https://eprint.iacr.org/2016/1120.

[SMB18] Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. IACR Trans. Symm. Cryptol., 2018(3):124–162, 2018.