Revisiting the Differential Meet-In-The-Middle Cryptanalysis

Ling Song¹, Qianqian Yang², and Huimin Liu¹

¹ College of Cyber Security, Jinan University, Guangzhou, China
² State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

songling.qs@gmail.com, yangqianqian@iie.ac.cn, liuhuimin301@gmail.com

Abstract. The differential meet-in-the-middle (MITM) attack is a new cryptanalysis technique proposed at Crypto 2023 recently. It led to greatly improved attacks on round-reduced SKINNY-128-384 and AES-256. In this paper, we revisit the differential MITM attack and propose several variants by absorbing techniques widely used in the classical differential attack. In particular, we present a new differential MITM attack that generalizes the basic differential MITM attack in several aspects. As for applications, we make refinements to the 24-round attack on SKINNY-128-384; on 12-round AES-256, we show that the classical differential attack and the generalized differential MITM attack perform better than the basic differential MITM attack.

Keywords: Differential cryptanalysis, Differential meet-in-the-middle cryptanalysis, Key recovery, SKINNY, AES

1 Introduction

Differential cryptanalysis, which was introduced by Biham and Shamir [BS90, BS91], is one of the most powerful cryptanalytic approaches for assessing the security of block ciphers. The basic idea is to exploit non-random propagation of input difference to output difference, i.e., high-probability differentials. The first step to mount a differential attack is to find a high-probability differential covering a large number of rounds. This procedure has been extensively studied and many approaches have been proposed [Mat94, MWGP11, MP13, SHW⁺14, SWW21]. Once an r-round high-probability differential of a certain cipher has been found, one could add some outer rounds and restrict the possible values of key bits in the outer rounds. Indeed, the right key of the outer rounds will allow her to observe the non-randomness of the differential.

Since the introduction of differential cryptanalysis, many improvements have been proposed: structures of data [BS92], conditional differentials [KMN10], probabilistic neutral bits [AFK⁺08], the early abort technique [LKKD08] and refinements in the key recovery process. Recently at Crypto 2023, Boura et al. introduced a new cryptanalysis technique, called the differential meet-in-the-middle (MITM) attack [BDD⁺23]. It provides a novel way to do the key recovery
in a differential attack (or it can be seen as a new way to do MITM attacks). In classical differential attacks, pairs of data that potentially satisfy the differential are constructed, and each key suggested by the data is a candidate whose number of occurrences is counted. The right key is then among the candidates and is likely to be the one that occurs most often. In the differential MITM attack, it constructs pairs of data that potentially satisfy the differential in an MITM manner, which leads to good results on block ciphers SKINNY-128-384 and AES-256. Especially for SKINNY-128-384, the differential MITM attack can cover 24 and even 25 rounds in the single-key setting while the best previous attack reaches 23 rounds only. Besides, the differential MITM attack is elegant and easy to use. In regard to this new cryptanalysis technique, some questions arise. How does the differential MITM attack compare to the classical differential attack? In which case is the differential MITM attack more advantageous?

**Our contribution.** With these questions in mind, we revisit the differential MITM attack and make extensions to it. Also, we make a clearer comparison between the differential MITM attack and the classical differential attack.

1. We propose some basic variants of the differential MITM attack, which allow one to use (partial) structures of plaintexts and flexibly select the number of key candidates for the exhaustive search. We highlight two notions in the computation of time complexity and stress to treat them carefully. As an example, we revisit the differential MITM attack on SKINNY-128-384 and make refinements to it.

2. We give classical differential attacks on 12-round AES-256 in the related-key setting, whose time complexity can reach $2^{145}$. Note the basic differential MITM attack in this case has a time complexity of $2^{208}$.

3. We propose a new differential MITM attack called GDMA, which generalizes the basic differential MITM attack in several aspects. Unlike the basic differential MITM attack, the GDMA does not have to traverse all possible values for the involved key bits of the outer rounds in the MITM phase; it stores pairs of data rather than single messages so that filters on both sides can be exploited in the MITM phase. Consequently, the GDMA allows more balanced complexities and potentially leads to better results.

4. We apply the GDMA to AES-256 and have an attack on 12-round AES-256 whose time complexity is optimized and as good as the classical differential attack.

In comparison, GDMA can be seen as a variant of the classical differential attack by implementing the steps of partial encryption/decryption and construction of pairs in an MITM manner. Whether the GDMA is more advantageous depends on many parameters. In most cases, GDMA is paralleled with the classical differential MITM attack. Roughly speaking, it is more likely to be suitable to the case where a relatively large number of rounds are added around the differential. In such cases, the key size is much larger than the block size. This confirms the analysis with the authors of [BDD+23].
Organization. The rest of the paper is organized as follows. In Section 2, we recall
the differential MITM attack and the classical differential attack. In Section 3, we
propose some basic variants of the differential MITM attack and revisit the attack
on 24-round SKINNY-128-384. Section 4 presents the generalized differential MITM
attack and compares it with the classical differential attack via the applications
to AES-256. Some simple rules for doing key recovery attacks in differential
cryptanalysis are also given. We conclude the paper in Section 5.

2 Preliminaries

2.1 Differential MITM Attacks

We first briefly recall the differential meet-in-the-middle attack. We use the same
notations as in [BDD+23] for convenience.

Given an n-bit block cipher $E$ with a k-bit key, we treat it as the composition
of three sub-ciphers: $E = E_{\text{out}} \circ E_{\text{in}} \circ E_{\text{out}}$, as depicted in Figure 1. For $E_{\text{in}}$ there
is a differential $\Delta x \rightarrow \Delta y$ of probability $2^{-p}$. When we extend the differential
outwards with probability 1, $\Delta x$ will propagate to plaintext difference $\Delta_{\text{in}}$ over
$E_{\text{in}}^{-1}$ and $\Delta y$ will propagate to ciphertext difference $\Delta_{\text{out}}$ over $E_{\text{out}}$. Let all
possible $\Delta_{\text{in}}$ span a space with dimension $\ell_{\text{in}}$. Similarly, let all possible $\Delta_{\text{out}}$
span a space with dimension $\ell_{\text{out}}$. Suppose that it requires subkey information $k_{\text{in}}$
(resp. $k_{\text{out}}$) to verify the difference $\Delta x$ (resp. $\Delta y$) for plaintext (resp. ciphertext)
pairs. The differential attack aims to recover $k_{\text{in}}, k_{\text{out}}$ and further the master key
of $E$ based on the differential.

The differential MITM attack integrates the meet-in-the-middle technique
with the differential attack. It can be divided into two phases.

1. MITM phase.
   Choose $2^p$ plaintexts. For each one:
   (a) Given the plaintext $P$, for each guess $i$ for $k_{\text{in}}$, we can compute the
   associated $\widetilde{P}^i$ that ensures $E_{\text{in}}(P) \oplus E_{\text{in}}(\widetilde{P}^i) = \Delta x$. There are $2^{|k_{\text{in}}|}$
possible $(i, \widetilde{P}^i)$. Acquire the associated ciphertexts $\widetilde{C}^i = E(\widetilde{P}^i)$ with calls
to the encryption oracle and store $(\widetilde{C}^i, i)$ in a hash table $H$.
   (b) Given $C = E(P)$, for each guess $j$ for $k_{\text{out}}$, we can compute the associated
   ciphertext $\widetilde{C}^j$ that ensures $E_{\text{out}}^{-1}(C) \oplus E_{\text{out}}^{-1}(\widetilde{C}^j) = \Delta y$. There are $2^{|k_{\text{out}}|}$
possible $(j, \widetilde{C}^j)$.
   (c) Match $\widetilde{C}^j$ with $\widetilde{C}^i$ by looking up the table $H$. Each collision of $(\widetilde{C}^i, \widetilde{C}^j)$
suggests an associated key guess $k_{\text{in}} = i, k_{\text{out}} = j$, that we will consider
as a candidate. The number of expected collisions for one plaintext $P$ is
$2^{|k_{\text{in}}|+|k_{\text{out}}|-|k_{\text{in}} \cap k_{\text{out}}|-n}$.

2. Exhaustive search phase.
   (a) Guess the remaining key bits (if there are) and test the guess with
   additional pairs.
Complexity. The time complexity of this attack can be estimated as

$$T = 2^p \times \left(2^{|k_{in}|} + 2^{|k_{out}|}\right) + 2^{|k_{in} \cup k_{out}| - n + p} + 2^{k - n + p}, \quad (1)$$

where the first term corresponds to the computations done in the upper part $E_{in}$ and the lower part $E_{out}$, the second one to the number of expected key candidates for $k_{in} \cup k_{out}$ and the last one to the exhaustive search.

The data complexity of this first version of the attack can be roughly estimated as $D = \min \left(2^n, 2^{p+\min(|k_{in}|,|k_{out}|)}\right)$. The memory complexity is given by $M = 2^{\min(|k_{in}|,|k_{out}|)}$, but it can be improved to $2^{\min(|k_{in}|,|k_{out}|) - |k_{in} \cap k_{out}|}$ by guessing the common key material at the beginning.

2.2 Classical Differential Attacks

Since the introduction of the differential attack [BS90,BS91], key recovery attacks were considered alongside. The classical differential key recovery attack mainly proceeds in the following phases.

1. Generate pairs of plaintexts that potentially suggest candidates for $k_{in}$ and $k_{out}$.
2. Extract the key candidates suggested by the pairs.
3. For all or partial candidates for $k_{in}$ and $k_{out}$ together with other needed key bits, test exhaustively until the right master key is found.

The main property of the classical differential attack is that pairs of plaintexts or ciphertexts are generated at first. For an $n$-bit block cipher, there is an $n$-bit filter for $k_{in}$ and $k_{out}$ when we extract candidates for them from pairs of data. Particularly, the $n$ filtering bits are used step by step. On the contrary, in the differential MITM attack, these filtering bits are used together in one step when pairs of data that suggest key candidates are generated.
Usually, the number of occurrences of each suggested key is counted. When the number of right pairs that follow the differential is large enough, the right key is likely to be the one that occurs most often. One can pick one or a list of candidates that are counted most often for exhaustive searches. An extreme case is to test all the candidates as the original differential MITM attack does. The ratio of tested key candidates will affect the success probability of the attack [Sel08, BGT11].

Another common idea to improve differential cryptanalysis is to use plaintext structures. A plaintext structure takes all possible values for the $\ell_{in}$ bits and chooses a constant for the remaining $n - \ell_{in}$ bits. It allows to enjoy the birthday effect and potentially attack more rounds without increasing the data complexity.

### 3 Basic Variants of the Differential MITM Attack

In this section, we first propose two variants of the differential MITM attack. We highlight two notions in the computation of time complexity and stress to treat them carefully. Then we revisit the differential MITM attack on 24-round SKINNY-128-384 from [BDD+23] and design refinements for it.

#### 3.1 Basic Variants

**Basic ideas.** We observe that the differential MITM attack can be refined by absorbing some techniques widely used in the classical differential attack.

Avoid the full code book. In classical differential attacks, the data complexity depends only on the probability $2^{-p}$ of the differential, no matter how many active bits the plaintext has, i.e., regardless of $\ell_{in}$. However, in the differential MITM attack, when $\ell_{in} = n$, a full codebook is required. In this case, we propose to use a partial structure of plaintexts to avoid the requirement of the full codebook\(^3\), as in the classical differential attacks. More generally, when $p + 1 < \ell_{in}$, a partial structure of plaintexts can be used.

Balance the time complexities. In the original differential MITM attack, all the key candidates are tested. In some cases, the time complexity of the exhaustive search may dominate the overall complexity. Instead of testing all candidates, we may choose to test a list of most likely key candidates, which may reduce the overall time complexity by increasing the data by a small factor, say 4. In order to count the occurrences of each possible key candidate, an extra memory is needed.

**Concrete variants.** We incorporate the above ideas and propose two variants DMA-1 and DMA-2 of the differential MITM attack, as shown in Figure 2. The first variant follows the original differential MITM and enumerates every key candidate while the second variant borrows the so-called counting method from the classical differential attack.

---

\(^3\) In [BDD+23], a similar idea was used, which allows for reducing the data without increasing the time complexity in certain cases. However, the partial structure technique is slightly different and more generic.
Differential MITM attack 1 (DMA-1) based on the enumeration method

1. If $\ell_{in} \leq p + 1$:
2. Construct a structures of $2^\ell_{in}$ plaintexts.
3. else:
4. Construct a partial structure of $2^s = 2^{(p+\ell_{in}+1)/2}$ plaintexts.
5. Query for the ciphertexts. Store the plaintext-ciphertext pairs in $S$.
6. For each $(P,C)$ from the structure $S$:
7. Guess $u$ for $k_{\cap}$:
8. Guess $i$ for $k^*_{\cap}$:
9. Compute $P$ from $P$ such that $\Delta x$ is reached under $(u,i)$.
10. // The test below is needed when a partial structure is used.
11. If $P \notin S$, go to Step 8; otherwise, let $\tilde{C} = E(\tilde{P})$.
12. If $(C,\tilde{C})$ have no difference on the $n - \ell_{out}$ bits, store $(\tilde{C},i)$ in a table $H$.
13. Guess $j$ for $k^*_{\out}$:
14. Compute $C$ from $C = E(P)$ such that $\Delta y$ is reached under $(u,j)$.
15. For each $i \in H(\tilde{C})$:
16. Get candidate $(u,i,j)$ and do an exhaustive search.
17. Repeat Step 1~15 about $2^{p-\ell_{in}+1}$ times if $\ell_{in} \leq p + 1$.

Differential MITM attack 2 (DMA-2) based on the counting method

1. If $\ell_{in} \leq p + 1$:
2. Construct $2^{p-\ell_{in}+1}$ structures, each of $2^s = 2^{\ell_{in}}$ plaintexts.
3. else:
4. Construct a partial structure of $2^s = 2^{(p+\ell_{in}+1)/2}$ plaintexts.
5. Query for the ciphertexts. Store the plaintext-ciphertext pairs in $S$.
6. Guess $u$ for $k_{\cap}$:
7. For each structure $S_\ast$ in $S$:
8. For each $(P,C)$ from the structure $S_\ast$:
9. Guess $i$ for $k^*_{\cap}$:
10. Compute $\tilde{P}$ from $P$ such that $\Delta x$ is reached under $(u,i)$.
11. If $\tilde{P} \notin S_\ast$, go to Step 9; otherwise, let $\tilde{C} = E(\tilde{P})$.
12. If $(C,\tilde{C})$ have no difference on the $n - \ell_{out}$ bits, store $(\tilde{C},i)$ in a table $H$.
13. Guess $j$ for $k^*_{\out}$:
14. Compute $\tilde{C}$ from $C = E(P)$ such that $\Delta y$ is reached under $(u,j)$.
15. For each $i \in H(\tilde{C})$:
16. Get candidate $(u,i,j)$ and update the key counters.
17. Test a fraction $2^{-h}$ of key candidates with top counters, $0 \leq h \leq |k^*_{\cap}| + |k^*_{\out}|$.

Figure 2: Two variants of the basic differential MITM attack.

We define $k_{\cap} = k_{in} \cap k_{out}$, $k^*_{in} = k_{in} - k_{\cap}$, and $k^*_{out} = k_{out} - k_{\cap}$. In the MITM phase, the common information of the key $k_{\cap}$ is guessed at first. Also, both variants use structures of plaintexts. When $\ell_{in} > p + 1$, a partial structure is enough due to the birthday effect. Therefore, the data complexity for both
variants is the same, namely,
\[
D = \begin{cases} 
2^{p+1}, & \text{if } \ell_{in} \leq p + 1 \\
2^{(p+\ell_{in}+1)/2}, & \text{otherwise.} 
\end{cases}
\]

Note that when we compute \( \overline{P} \) from \( P \) under the guessed \( k_{in} \), \( \overline{P} \) may not fall in the partial structure. Thus a check is needed. Since a random pair from a structure satisfies the input difference \( \Delta x \) with probability \( 2^{-\ell_{in}} \), a partial structure as described in Figure 2 will lead to about \( 2^p \) pairs satisfying the input difference of the differential. This explains the usage of partial structures.

The DMA-1 variant based on the enumeration method has the following memory and time complexities\(^4\).

\[
\begin{align*}
M_{\text{DMA-1}} &= \max\{2^s, \min\{2|k_{in}^*|, 2|k_{out}^*|\} \times 2^{f_{\text{out}}-n}\}, \\
T_{\text{DMA-1}} &= D \times (2^{|k_{in}|} + 2^{|k_{out}|} + 2|k_{in}|+|k_{out}|-n-f_{\text{extra}}) + 2^{k-n+p+|k_{extra}|},
\end{align*}
\]

where \( f_{\text{extra}} \leq |k_{\gamma}|, |k_{\text{extra}}| \geq 0 \). We will explain later why we introduce the two additional notations \( f_{\text{extra}} \) and \( k_{\text{extra}} \).

In the DMA-2 variant, as there is an extra need for storing the counters of the key, the memory complexity may increase. In order to save the memory for storing the counters of \( k_{\gamma} \), we store the whole data instead and extract \( k_{in}^* \) and \( k_{out}^* \) under each guess of \( k_{\gamma} \), as shown in line 6 of DMA-2. However, we can also do it the other way around if it is more beneficial. Therefore, the memory complexity is

\[
M_{\text{DMA-2}} = \min\{\max\{D, 2^{|k_{in}^*|+|k_{out}^*|}\}, \max\{2^s, 2|k_{in}|\cup k_{out}|\}\}.
\]

The time complexity of the DMA-2 variant is

\[
T_{\text{DMA-2}} = D \times (2^{|k_{in}|} + 2^{|k_{out}|} + 2|k_{in}|+|k_{out}|-n-f_{\text{extra}}) + 2^{k-n+p+|k_{\text{extra}}|} - h,
\]

where \( 0 \leq h \leq |k_{in}^*| + |k_{out}^*| \) can be used to trade the data complexity for lower time complexity.

The reason for introducing \( f_{\text{extra}} \) and \( k_{\text{extra}} \). In the matching and filtering phase of the attack, when looking up the hash table, we combine values for \( k_{in} \) and \( k_{out} \) by matching \( \tilde{C} \) and \( C \). This implies an \( n \)-bit filter. Besides, there might exist redundancy between \( k_{in} \) and \( k_{out} \) that can also act as filters. We denote it as a \( f_{\text{extra}} \)-bit filter. Specifically, in DMA-1 and DMA-2, \( f_{\text{extra}} = |k_{\gamma}| \), namely assume \( k_{\gamma} \) has an explicit form and can be guessed before the MITM phase. In certain cases, even though \( k_{\gamma} \) is not empty, it may not have an explicit form, and thus \( f_{\text{extra}} \) may be smaller than \( |k_{\gamma}| \).

In the exhaustive search phase, to recover the master key, the remaining key information outside \( k_{in} \cup k_{out} \) is required for testing. Theoretically, \( k - |k_{in} \cup k_{out}| \)

---

\(^4\) One may wonder if the time complexity doubles when compared to the original differential MITM attack due to \( D = 2 \cdot 2^p \). It is not true. In the original differential MITM attack, it is assumed implicitly that \( (P, \overline{P}) \) and \( (\overline{P}, P) \) do not appear simultaneously. We do not have such an assumption.
more bits are needed, but for concrete ciphers in practice, \( k - |k_{in} \cup k_{out}| \) bits may not be enough and an extra part of key materials \( k_{extra} \) may be used.

If we take \( f_{extra} = |k| \) and \( k_{extra} = \emptyset \), DMA-1 has the same time complexity as the original differential MITM attack [BDD+23]. This means all the redundancy between \( k_{in} \) and \( k_{out} \) can be used at once as filters when we look up the hash table and that exact \( k - |k_{in} \cup k_{out}| \) more bits are required for recovering the master key. In Appendix A.1, we take the block cipher SPECK as an example to show that a nonempty \( k_{extra} \) may be needed and that \( f_{extra} \) may be smaller than \( |k| \). Our analysis shows that the values for \( k_{extra} \) and \( f_{extra} \) should be taken carefully.

In the following subsection, we revisit the attack on 24-round SKINNY, and make refinements by giving an explicit form of \( k \) and computing \( f_{extra} \) correctly.

### 3.2 Refined Attack on 24-Round SKINNY-128-384

**Description of SKINNY.** SKINNY [BJK+16] is a family of lightweight tweak block ciphers whose tweak key schedule adopts the TWEAKEY framework [JNP14]. Members of SKINNY are denoted by \( \text{SKINNY-} n \cdot tk \), where \( n \in \{64, 128\} \) is the block size and \( tk \in \{n, 2n, 3n\} \) is the tweak key size. The internal states of SKINNY are represented as \( 4 \times 4 \) arrays of cells with each cell being a nibble in case of \( n = 64 \) bits and a byte in case of \( n = 128 \) bits. The tweak key state is seen as a group of \( z \) \( 4 \times 4 \) arrays, where, \( z = tk/n \). The arrays are marked as \( TK_1, (TK_1, TK_2) \) and \( (TK_1, TK_2, TK_3) \) for \( z = 1, 2, 3 \) respectively.

SKINNY iterates a round function for \( N_r \) rounds and each round consists of the following five steps.

1. **SubCells (SC)** - A 4-bit (resp. 8-bit) S-box is applied to all cells when \( n \) is 64 (resp. \( n \) is 128).
2. **AddConstants (AC)** - This step adds constants to the internal state.
3. **AddRoundTweakey (ART)** - The first two rows of the internal state absorb the first two rows of \( TK \), where \( TK = \bigoplus_{i=1}^{z} TK_i \).
4. **ShiftRows (SR)** - Each cell in row \( j \) is rotated to the right by \( j \) cells.
5. **MixColumns (MC)** - Each column of the internal state is multiplied by matrix \( M \). The inverse MixColumns operation employs \( M^{-1} \) instead.

\[
M = \begin{pmatrix}
1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0
\end{pmatrix}, \quad M^{-1} = \begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 \\
1 & 0 & 0 & 1
\end{pmatrix}
\]

The tweak key schedule of SKINNY is linear. The tweak key is first loaded into \( z \times 4 \times 4 \) tweak key states. After each ART step, a cell-wised permutation \( P \) is applied to each tweak key state, where \( P \) is defined as: \( P = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7] \). Then cells in the first two rows of all tweak key states but \( TK_1 \) are individually updated using LFSRs. For complete details of the tweak key scheduling algorithm, one can refer to [BJK+16].
The Original Attack. In \cite{BDD23}, the authors gave a 23-round attack on \textsc{Skinny}-128-384 using the basic differential MITM described in Section 2.1 and then extended it to 24 rounds by a technique called parallel partitions.

The attacks utilize a 13-round differential $\Delta x \rightarrow \Delta y$ with a probability $2^{-105.9}$. The 23-round attack adds five rounds before and after the distinguisher, respectively, as shown in Figure 3, where $k_{in}$ contains 31 subtweakey bytes, $k_{out}$ 32 subtweakey bytes, and $k_{in} \cap k_{out}$ 15 subtweakey bytes. Based on the 23-round attack, the 24-round attack was obtained by appending one more round and has the same time complexity of $2^{361.9}$. For completeness, we recall the original attacks in Appendix A.2.

Motivation for refinements. In the original 24-round attack, the authors mentioned 248 filtering bits, namely, 120 filtering bits (15 bytes) between the two parts of the key, 64 filtering bits from the last subtweakey $K_{23}$, and 64 filtering bits on the last two rows of the ciphertext difference. If so, the time complexity is $2^{425.9}$ instead of the claimed $2^{361.9}$. The inconsistency lies in that another 64 filtering bits are missed in \cite{BDD23}. In fact, on top of the mentioned filters, the difference on the first two rows of ciphertext should be equal as well.

In addition, the 24-round attack does not give an explicit way to exploit all the filters efficiently. If the three sets of filters are used one by one, the time complexity will be high. Suppose the ciphertexts are matched to ensure $\hat{P}$ and $\hat{C}$ are valid plaintext and ciphertext at first, and then the consistency between $k_{in}$ and $k_{out}$ is checked. In this case, the number of candidates for $(k_{in}, k_{out})$ is $2^{|k_{in}| + |k_{out}| - n + p}$, which will be reduced to $2^{|k_{in} \cup k_{out}| - n + p} = 2^{|k_{in} \cap k_{out}| - n + p}$ afterwards. Thus, a large term $2^{|k_{in}| + |k_{out}| - n + p}$ appears which may dominate the overall time complexity. In such MITM procedures, a key point to have a low time complexity is to utilize all the filters before or when the hash table is looked up.

Questions. When we figure out all the filters, we are left to design a method that exploits them cleverly. The 128-bit filter on the ciphertext difference can be considered easily. Then the question is: \textit{How does the differential MITM attack proceed to exploit the $|k_{in} \cap k_{out}|$-bit filter and the 64-bit filter on the last subtweakey of \textsc{Skinny} while or before the hash table is looked up?} Next, we will answer the question and give a refined attack on 24-round \textsc{Skinny}-128-384.

Refined Attacks. We provide solutions to the above question in the following two observations. According to the tweakey schedule of \textsc{Skinny}-128-384, each subtweakey byte is a linear combination of bytes at the same position of the master tweakey states. In addition, the bytes in the tweakey states are permuted simultaneously during the tweakey schedule. Therefore, knowing three subtweakey bytes that are computed from the same master tweakey bytes is equivalent to knowing the three involved master tweakey bytes.

A typical case in the attack on \textsc{Skinny}-128-384 is that for the three master tweakey bytes at the same position, $k_{in}$ consists of two-byte information of them.
and so does $k_{\text{out}}$. Namely, $k_{\gamma} = k_{\text{in}} \cap k_{\text{out}}$ contains one byte for this position. Theoretically, as described in the DMA-1 and DMA-2 (see Figure 2), guessing $k_{\gamma}$ in advance helps to utilize the $|k_{\gamma}|$-bit filter before looking up the hash table. An explicit way to achieve it is given in Observation 1 for SKINNY. For cases other
than $k_{in}$ and $k_{out}$ sharing one-byte information for a position, it is trivial to deal with.

**Observation 1** Let $\mathcal{L}, \mathcal{R}$ be the matrices corresponding to the two LFSRs used in the tweakey schedule and $tk_i \in \mathbb{F}_2^8$, $i \in \{1, 2, 3\}$ unknown vectors. For distinct $i, j, k > 0$, let

\[
\begin{align*}
a &= tk_1 + tk_2 + tk_3, \\
b &= tk_1 + \mathcal{L}^i \cdot tk_2 + \mathcal{R}^i \cdot tk_3, \\
c &= tk_1 + \mathcal{L}^j \cdot tk_2 + \mathcal{R}^j \cdot tk_3, \\
d &= tk_1 + \mathcal{L}^k \cdot tk_2 + \mathcal{R}^k \cdot tk_3.
\end{align*}
\]

Then there exists $m = \mathcal{X} \cdot a + \mathcal{Y} \cdot b = \mathcal{Z} \cdot c + \mathcal{W} \cdot d$, where $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{W}$ are $8 \times 8$ binary matrices.

Without loss of generality, let $\mathcal{W} = \mathcal{I}$. From the equation $m = \mathcal{X} \cdot a + \mathcal{Y} \cdot b = \mathcal{Z} \cdot c + \mathcal{I} \cdot d$, we can get three equations

\[
\begin{align*}
\mathcal{X} + \mathcal{Y} \cdot A_1 + \mathcal{Z} \cdot A_2 &= A_3, \\
\mathcal{X} \cdot B_1 + \mathcal{Z} \cdot B_2 &= B_3.
\end{align*}
\]

For the attack on SKINNY-128-384, $(i, j, k)$ are either $(1, 10, 11)$ or $(1, 9, 10)$. For both cases, we can derive invertible matrices $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ from the three equations via a simple computation.

The 24-round attack starts from a set of $2^{64}$ ciphertexts. The basic MITM phase, which still covers the first 23 rounds, builds a hash table for all elements in the set. Denote the state before the ART of Round 23 as $S_{23}$. Then we have $C \oplus \overline{C} = MC \circ \text{SR}(S_{23} \oplus \overline{S}_{23} \oplus \Delta K_{23})$, which is a 128-bit filter. Also, when the table is looked up, the plaintext $P$ and ciphertext $C$ should match individually. That is, $(\text{SR}^{-1} \circ MC^{-1}(C) \oplus f(k_{in}))[0 : 63] = S_{23} \oplus g(k_{out})[0 : 63]$, where $f(k_{in})$ and $g(k_{out})$ are computed from $k_{in}$ and $k_{out}$ respectively and constitute $K_{23}$. This is a 64-bit filter. The exact expression of $f(k_{in})$ and $g(k_{out})$ can be obtained using Observation 2, which is verified experimentally as well.

**Observation 2** Let $\mathcal{L}, \mathcal{R}$ be the matrices corresponding to the two LFSRs used in the tweakey schedule and $tk_i \in \mathbb{F}_2^8$, $i \in \{1, 2, 3\}$ unknown vectors. For distinct $i, j, k, \ell > 0$, let

\[
\begin{align*}
a &= tk_1 + tk_2 + tk_3, \\
b &= tk_1 + \mathcal{L}^i \cdot tk_2 + \mathcal{R}^i \cdot tk_3, \\
c &= tk_1 + \mathcal{L}^j \cdot tk_2 + \mathcal{R}^j \cdot tk_3, \\
d &= tk_1 + \mathcal{L}^k \cdot tk_2 + \mathcal{R}^k \cdot tk_3, \\
e &= tk_1 + \mathcal{L}^\ell \cdot tk_2 + \mathcal{R}^\ell \cdot tk_3.
\end{align*}
\]

Suppose $a, b, c, d$ are known and there exist matrices $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ such that $\mathcal{X} \cdot a + \mathcal{Y} \cdot b = \mathcal{Z} \cdot c + d$. Then there exists $e = f(a, b) + g(c, d) = \mathcal{U} \cdot a + \mathcal{V} \cdot b + \mathcal{M} \cdot c + d$, where $\mathcal{U}, \mathcal{V}, \mathcal{M}$ are $8 \times 8$ binary matrices.
Revised attacks. Based on the two observations, we give procedures for the 24-round attack such that the time complexity reaches the claimed one in [BDD+23].

1. Ask for the encryption of the whole codebook.
2. Pick $2^{64}$ plaintext/ciphertext pairs $(P_\ell, C_\ell)$ such that $\text{MC}^{-1}(C_\ell)$ is constant on the last two rows.
3. For each possible value $u$ of $k_\cap$:
   (a) Compute all possible tuples $(P_\ell, \tilde{P}_i^\ell, i)$ for each value $i$ of $k_\cap$ and each $P_\ell$ from the structure defined at the previous step such that the state difference after the 6th S-box layer is 0x02. Store $(C_\ell \oplus \tilde{C}_i^\ell, (\text{SR}^{-1} \circ \text{MC}^{-1}(C_\ell) \oplus f(k_\text{in}))[0 : 63], i)$ in a hash table. The memory complexity is $2^{128+64} = 2^{192}$ 320-bit words.
   (b) For each value $j$ of $k_\text{out}$ and each state $S_{23,\ell}$ (the state after SC of Round 23), compute all possible tuples $(S_{23,\ell} \oplus \tilde{S}_{23,\ell}^j, S_{23,\ell} \oplus g(k_\text{out}), j)$ so that the difference on the state before the 19th S-box layer is 0x64 on the four active bytes.
   (c) Check for possible matches on the hash table. The match is now performed on $C_\ell \oplus \tilde{C}_i^\ell = \text{MC} \circ \text{SR}(S_{23,\ell} \oplus \tilde{S}_{23,\ell}^j \oplus \Delta K_{23})$ and $(\text{SR}^{-1} \circ \text{MC}^{-1}(C_\ell) \oplus f(k_\text{in}))[0 : 63] = S_{23,\ell} \oplus g(k_\text{out})[0 : 63]$. This is a 192-bit filter.
   (d) Test candidates for $(k_\text{in}, k_\text{out})$ against very few additional plaintexts.
4. Repeat from Step 2 about $2^{41.9}$ times until the right key is retrieved.

Now $f_{\text{extra}} = |k_\cap| + |K_{23}| = 184$ and thus the time complexity is

$$T = 2^{41.9} \times 2^{120} \times 2^{128+64} \times 2^{136+64} \times 2^{41.9} \times 2^{120+128+64+136+64} = 2^{361.9}.$$
Classical differential attack 1 (CDA-1) where $E_{out} = I, \ell_{in} \leq p + 1$

1. For each of $2^{n-\ell_{in}+1}$ plaintext structures:
2. Guess $k_{in}$;
3. Do partial encryption.
4. Get $2^{\ell_{in}-n-1}$ plaintext pairs that satisfy both $\Delta x$ and $\Delta y$.
5. Update the key counters or test directly.

$D = 2^{p+1}; T = D \times 2^{|k_{in}|} + T_{search}; M = \max\{2^{\ell_{in}}, 2^{|k_{in}|}\}$ or $2^{\ell_{in}}$ according to Step 5.

Classical differential attack 2 (CDA-2) where $E_{out} = I, \ell_{in} \leq p + 1$

1. For each of $2^{p-\ell_{in}+1}$ plaintext structures:
2. Store the data into a hash table according to the ciphertext.
3. Select ciphertext pairs satisfying $\Delta y$. There will be $2^{2\ell_{in}-1-n}$ pairs.
4. For each of such pairs:
5. Extract $2^{\ell_{in}^*}$ candidates for $k_{in}^*$, under which $\Delta x$ can be reached.
6. Update the key counters or test directly.

$D = 2^{p+1}; T = D \times (1 + 2^{\ell_{in}-n-1} + 2^{k_{in}^*} \cdot \epsilon) + T_{search}; M = \max\{2^{\ell_{in}}, 2^{|k_{in}|}\}$ or $2^{\ell_{in}}$ according to Step 6.

Classical differential attack 3 (CDA-3) where $E_{out} = I, \ell_{in} \leq p + 1$

1. For each of $2^{p-\ell_{in}+1}$ plaintext structures:
2. Guess $k'_{in}, 0 \leq |k'_{in}| \leq |k_{in}|$;
3. Do partial encryption.
4. Get $2^{2\ell_{in}^*-n-\ell_{in}}$ pairs having fixed difference on the filtering bits.
5. For each of such pairs:
6. Extract $2^{|k_{in}^*|}$ candidates for $k_{in}^*$, under which $\Delta x$ can be reached.
7. Update the key counters or test directly.

$D = 2^{p+1}; T = D \times (2^{|k'_{in}|} + 2^{|k'_{in}|+\ell_{in}^*} + 2^{k_{in}^*} \cdot \epsilon) + T_{search}; M = \max\{2^{\ell_{in}}, 2^{|k_{in}|}\}$ or $2^{\ell_{in}}$ according to Step 7.

**Figure 4:** Classical differential attacks in the simple case

the time complexity of the exhaustive search varies with the ratio of tested key candidates, we mainly focus on the time complexity of the attack excluding the exhaustive search.

In classical differential attacks, it permits guessing none of the key bits and this corresponds to CDA-2. The key candidates can be obtained via computations or accesses to some precomputed tables. Given a pair of plaintexts from the same structure, it suggests about $2^{|k_{in}^*|}$ candidates for $k_{in}^*$. Suppose extracting this number of candidates takes a time complexity of $2^{|k_{in}^*|-\ell_{in}} \cdot \epsilon, \epsilon \geq 1$. Particularly, if the assumption below holds, $\epsilon$ will be as small as 1.
Assumption 1  Given a pair of plaintexts that suggests $N > 0$ possible candidates for the key, assume these candidates can be extracted with a time complexity as small as $N$.

More generally, assume $k'_\text{in}$, a part of $k_{\text{in}}$, is guessed, where $0 \leq |k'_\text{in}| \leq |k_{\text{in}}|$. Suppose there are additional $\ell'_\text{in}$ filtering bits under the guess of $k'_\text{in}$. Let $k^*_\text{in} = k_{\text{in}} - k'_\text{in}$, $\ell^*_\text{in} = \ell_{\text{in}} - \ell'_\text{in}$. A more general attack named CDA-3 is given in Figure 4 and covers CDA-1 and CDA-2, i.e., CDA-1 and CDA-2 are its special cases.

Generic cases. Generally, some rounds may also be added to the differential in a differential attack and $\ell_{\text{in}} \leq p + 1$ does not necessarily hold. With this in mind, we extend CDA-3 and give the generic classical differential attack named GCDA, as shown in Figure 5, where notations $k_{\text{out}}$, $k^*_\text{out}$ and $\ell_{\text{out}}$, $\ell^*_\text{out}$ are defined similarly. The time complexity consists of four parts:

- $T_0 = 2^{\ell'_\text{in}} |k'_\text{out}| \times D$ for partial encryption and decryption;
- $T_1 = 2^{\ell'_\text{in}} |k'_\text{out}| \times D \times 2^{-s-1+\ell_{\text{out}}-n-\ell^*_\text{in}-\ell_{\text{out}}}$ for getting the pairs that satisfy some filtering conditions where $2^s$ is the structure size;
- $T_2 = 2^{\ell'_\text{in}} |k'_\text{out}| + p - n \cdot \epsilon$ for extracting all the key candidates;
- $T_3 = T_{\text{search}}$ for the exhaustive search, respectively.

The formula for the overall time complexity is

$$T = 2^{\ell'_\text{in}} |k'_\text{out}| \times D (1 + 2^{-s-1+\ell_{\text{out}}-n-\ell^*_\text{in}-\ell_{\text{out}}}) + 2^{\ell'_\text{in}} |k'_\text{out}| + p - n \cdot \epsilon + T_{\text{search}}. \quad (4)$$

Remark 1. When Assumption 1 holds, the basic differential MITM attack is not better than the classical differential attack. Check a special case of the GCDA where $k'_\text{in} = k_{\text{in}}$ and $k'_\text{out} = \emptyset$. The time complexity for this case is

$$T = 2^{\ell'_\text{in}} |k'_\text{out}| \times D (1 + 2^{-s-1+\ell_{\text{out}}-n-\ell_{\text{in}}}) + 2^{\ell'_\text{in}} |k'_\text{out}| + p - n + T_{\text{search}}.$$

As $s - 1 + \ell_{\text{out}} - n - \ell_{\text{in}} < 0$, the above time complexity is not larger than the time complexity in Equation (2).

Take the attack on 23-round SKINNY-128-384 as an example where $|k_{\text{in}}| = 248$, $|k_{\text{out}}| = 256$, $|k_{\text{in}} \cap k_{\text{out}}| = 120$ and $\ell_{\text{in}} = \ell_{\text{out}} = 128$. Suppose only $k_{\text{in}}$ is guessed in the GCDA. The partial structure has a size of $2^s = 2^{116.95}$ and the data complexity is $D = 2^{s+1} = 2^{117.95}$, and the time complexity is

$$T = 2^{248} \times 2^{117.95} (1 + 2^{-11.05}) + 2^{361.9} \cdot \epsilon + 2^{361.9}.$$

Therefore, if $\epsilon$ can be as small as 1, the basic differential MITM is not better. However, making $\epsilon$ as small as possible itself is a challenge in classical differential attacks.

4.2 Attacks on AES and Comparisons

Description of AES Advanced Encryption Standard (AES) [DR02] is a block cipher that encrypts 128-bit plaintext with the secret key of sizes 128, 192, and 256 bits. Its internal state can be represented by a $4 \times 4$ matrix whose elements are byte values in a finite field of $GF(2^8)$. As shown in Figure 6, the round function consists of four basic transformations in the following order:
Generic classical differential attack (GCDA)

1. $S \leftarrow \emptyset$
2. If $\ell_{in} \leq p + 1$:
3. $S \leftarrow 2^{p-\ell_{in}+1}$ structures, each of $2^x = 2^{\ell_{in}}$ plaintexts
4. else:
5. $S \leftarrow$ a partial structure of $2^x = 2^{(p+\ell_{in}+1)/2}$ plaintexts
6. Guess $k_{in}^*$ and $k_{out}^*$, $0 \leq |k_{in}^*| \leq |k_{in}|$, $0 \leq |k_{out}^*| \leq |k_{out}|$:
7. For $S_\ast \in S$:
8. Do partial encryption and decryption for elements in $S_\ast$ if $k_{in}^* \cup k_{out}^* \neq \emptyset$.
9. // Additional $\ell_{in}', \ell_{out}'$ filtering bits are obtained, respectively.
10. Store the data into a hash table indexed by the filtering bits.
11. Get $2^{2s-1+\ell_{out}-n-\ell_{in}'}$ pairs having fixed difference on the filtering bits.
12. For each of such pairs:
13. Extract $2^{\ell_{in}'}$ candidates for $k_{in}^*$, under which $\Delta x$ can be reached.
14. Extract $2^{\ell_{out}'}$ candidates for $k_{out}^*$, under which $\Delta y$ can be reached.
15. Update the key counters or test directly.

$-$ When $\ell_{in} \leq p + 1$, $D = 2^{p+1}$; otherwise, $D = 2^{(p+\ell_{in}+1)/2}$.
$-$ $M = \max\{2^{\ell_{in}'} \cup k_{out}', D\}$ or $\min\{D, 2^{\ell_{in}'}\}$ depending on Step 15.

Figure 5: The generic classical differential attack (GCDA).

- **SubBytes (SB)** is a nonlinear substitution that applies the same S-box to each byte of the internal state.
- **ShiftRows (SR)** is a cyclic rotation of the $i$-th row by $i$ bytes to the left, for $i = 0, 1, 2, 3$.
- **MixColumns (MC)** is a multiplication of each column with a Maximum Distance Separable (MDS) matrix over $\text{GF}(2^8)$.
- **AddRoundKey (AK)** is an exclusive-or with the round key.

![Figure 6: AES round function and the ordering of bytes](image)

At the beginning of the encryption, an additional whitening key addition is performed, and the last round does not contain MixColumns. AES-128, AES-192, and AES-256 share the same round function with a different number of rounds: 10, 12, and 14, respectively. AES-256 has a 256-bit key, which is twice as large as the internal state and derives round keys from the master key based on the key schedule illustrated in Figure 7. We refer to [DR02] for more details.
Differential MITM Attack on 12-round AES-256 In [BDD+23], to show the power of the differential MITM attack, a 12-round attack on AES-256 was proposed. This attack requires a pair of related keys. The attacker chooses two bytes $a$ and $b$ such that the differential transition $b \rightarrow a$ through the S-box happens with probability $2^{-6}$. The attacker then injects the difference $b$ on the first byte of the round key $k_8$ and $\text{MC}(a, 0, 0, 0)$ to the first column of the round key $k_9$. Figure 8 displays the attack on AES-256, where $u_i, 0 \leq i \leq 9$ are unknown key differences.
The differential used for the attack starts from columns 0 and 3 of the state $w_0$ and columns 1 and 2 of $z_1$ and stops at state $x_{11}$. The differential holds with a probability of 0.25. When it does, its probability is $2^{-86}$. To verify the input difference of the differential, it involves 15 key bytes, namely, $k_{in}$ contains $k_0[0,2,3,4,7,8,9,13,14], k_1[5,10]$ and $k_3[12,13,14,15]$. Similarly, to verify the output difference of the differential, it requires the information of 8 key bytes, i.e., $k_{out}$ consists of $k_{11}[12,13,14,15]$, and $k_{12}[0,4,8,12]$, from which an extra key byte $k_{10}$ can also be derived. These key bytes are highlighted as red squares in Figure 8. The remaining information of the master key contains 9 bytes. Applying the differential MITM attack presented in Section 2.1, one can have an attack of data, memory, and time\(^5\) complexities $D = 2^{89}$, $M = 2^{89}$ and $T = 2^{214}$, as
\[
T = 2^p \times \left(2^{k_{in}} + 2^{k_{out}}\right) + 2^{k_{in} \cup k_{out}} - n+p + 2^k - n+p \\
= 2^{86} \times (2^{120} + 2^{64}) + 2^{120+64-128+86} + 2^{256-128+86} \\
= 2^{206} + 2^{142} + 2^{214} \approx 2^{214}. \tag{5}
\]

From Equation (5), the time for the exhaustive search dominates the overall time complexity. In this case, DMA-2 in Figure 2 can be applied to lower this term and make the time complexity $2^{88} \times (2^{120} + 2^{64}) + +2^{120+64-128+88} + 2^{256-128+88-10} = 2^{208}$ (there are 4 right pairs) but the memory complexity becomes $2^{k_{in} \cup k_{out}} = 2^{184}$.

**Classical Differential Attacks on 12-round AES-256** We apply the GCDA to AES-256 and give the following attack using the same differential trail. The plaintext difference falls in a space of dimension $11 \times 8$ since $\Delta k_0[1] = \Delta k_0[5]$. The attack starts with preparing a structure of $2^{88}$ plaintexts under both related keys, respectively.

1. Guess $k_3[12,13,14,15], k_{11}[13]$, and $k_{12}[8,12]$:
   (a) Compute differences ($u_0, u_1, u_2, u_3$) from $k_3[12,13,14,15]$ and ($2a, a, a, 3a$); compute $u_5$ from $b$ and $k_{10}[12] = k_{12}[8] \oplus k_{12}[12]$; compute $u_6$ from $a$ and $k_{11}[13]$.
   (b) Initialize counters for all possible values of $k_0[0, 2, 3, 4, 7, 8, 9, 13, 14, 15], k_1[5, 10]$ and $k_{11}[12,14,15], k_{12}[0,4]$.
   (c) Construct $2^{16}$ pairs of structures. Each pair has fixed differences on 7 plaintext bytes at positions 1, 5, 6, 10, 11, 12, 15. For each pair of structures:
      i. Do partial encryptions and decryptions.
      ii. Using a hash table, generate $2^{72 \times 2-9 \times 8-2 \times 8}$ pairs of data such that 1) the last three rows of the ciphertext difference satisfy the pattern of $\Delta x_{12}$ and 2) $\Delta x_{11}[8,12]$ are $u_5$.
      iii. For each pair of data, extract the other bytes of $k_{in}$ by guessing $\Delta w_0[5,10]$.

\(^5\) We contacted the authors of [BDD+23] and confirmed that they mistook the time complexity $2^p \cdot \max\{2^{120}, 2^{64}\} + 2^8 \cdot 56 \cdot 72 = 2^{214}$ for $2^p \cdot \max\{2^{120}, 2^{64}, 2^{72}\} = 2^{206}$. 

17
which is much smaller than the time complexity $2^{208}$.

ii. Among the bytes of $\Delta z_1[0, 1, 2, 3]$ and $\Delta w_1[0, 1, 2, 3]$, four bytes are known. From them, compute $\Delta z_1[1, 2]$. From $\Delta x_1[5, 10]$ and $\Delta z_1[1, 2]$, recover $x_1[5, 10]$ and $k_1[5, 10] = w_0[5, 10] \oplus x_1[5, 10]$.

iv. For each pair of data, extract the other bytes of $k_{out}$:
- As $\Delta k_{12}[0] = u_6 + b$, $\Delta k_{12}[4] = u_6$, and $\Delta z_{11}[0] = \Delta x_{11}[4] = u_5$, compute $\Delta z_{11}[0, 4]$ and derive $z_{11}[0, 4]$, $k_{12}[0, 4]$.
- Let $(u_7, u_8, u_9) = \Delta C[1, 2, 3]$. As $\Delta k_{11}[14, 15, 12]$ for the basic MITM step, test exhaustively to find the right master key.

v. Update the counters.

(d) Select the key value with the largest counter. Together with the guessed key bytes and all possible values for another 9 bytes outside $k_{in} \cup k_{out}$, test exhaustively to find the right master key.

The data complexity is still $2^{89}$. The data and the counters should be stored, so the memory complexity is $\max\{2^{89}, 2^{16 \times 8}\} = 2^{128}$. The time complexity is

$$T = 2^{56} \times 2^{16} \times (2^{72} + 2^{56} + 2^{72} + 2^{72}) \approx 2^{145},$$

which is much smaller than the time complexity $2^{208}$ by the differential MITM attack. The expected number of right pairs from the data is 4, so the right key ranks first with a high probability.

An alternative attack. However, the memory complexity of the above attack is higher than the differential MITM attack. In fact, if we guess more key bytes in advance, the memory consumption for the counters decreases. Suppose $k_0[0], k_{12}[0, 4]$ and $k_0[2, 3]$ are also guessed. Then for each pair of structures, $2^{72 \times 2 \times 3 \times 8 - 5 \times 8} = 2^{32}$ pairs of data can be generated in Step (ii). Consequently, the time complexity becomes

$$T = 2^{96} \times 2^{16} \times (2^{72} + 2^{32} + 2^{32} + 2^{72}) \approx 2^{185},$$

and the memory complexity decreases to $\max\{2^{89}, 2^{11 \times 8}\} = 2^{89}$. This attack has the same data and memory complexities as the basic differential MITM attack but a lower time complexity.

Remark 2. In the attack on AES-256, the number of key candidates in total is $2^{144}$. In the classical differential attack, these candidates can be extracted with a time complexity as small as $2^{144}$, which means Assumption 1 holds for this attack. Recall that in this case, the classical differential attack is not worse than the basic differential MITM attack. Our dedicated attacks on AES also confirm this.

In the differential MITM attack, we could split the time complexity into three parts: $T_{mitm} = D \times (2^{k_{in}} + 2^{k_{out}})$ for the basic MITM step, $T_{candidates} = 2^{k_{in}} + |k_{out} - n||k_{extra}|$ for accessing the hash table to get the key candidates, and $T_{search} = 2^{k_{in} + p + |k_{extra}|}$ for the exhaustive search. When $T_{search}$ is dominant, we
could turn to DMA-2 and adjust it. When $T_{mitm}$ is much larger than $T_{candidates}$ (this is the case for the attack on AES-256 due to $2^{208} > 2^{144}$), guessing fewer key bits than max$\{|k_{in}|, |k_{out}|\}$ might be beneficial, but how the key is guessed is fixed in the differential MITM attack. On the contrary, the strategy for guessing the key is flexible in the classical differential attack. To relax this limitation, we propose generalized differential MITM attacks in the next subsection.

### 4.3 Generalized Differential MITM Attack

In this subsection, inspired by the results on AES-256, we propose generalized differential MITM attacks, which enjoy much greater flexibility than the basic differential MITM.

**Guessing A Subset of $k_{in}$ and $k_{out}$** The differential MITM attack on 12-round AES-256 is not as good as the classical differential attacks. One limitation of the differential MITM is that the way of guessing key bits is fixed. Can it be generalized? Inspired by the GCDA, a variant of the differential MITM attack, named DMA-3, is proposed in Figure 9. It allows one to guess a subset of the keys in the upper and lower parts, respectively. When the upper part and the lower part meet in Step 15, pairs of data that satisfy some of the filtering conditions, are obtained. Thus, like the GCDA, it may need to take further actions to extract the remaining information of $k_{in} \cup k_{out}$ using the remaining filters.

We apply DMA-3 to AES-256 reduced to 12 rounds. Let $k'_{in} = \emptyset$, $k'_{out} = k_{out}$. Then $k^*_{in} = k_{in}$ contains 15 bytes, $\ell^*_{in} = 88$, $\ell^*_{out} = 0$. $M = 2^{184}$, and the time complexity is $T = 2^{176} + 2^{112} + 2^{144}\cdot \epsilon + T_{search}$ and $\epsilon = 1$.

**Remark 3.** We could not make the first term lower and the time complexity is still higher than the the time complexity of classical differential attacks. Note that there are $8 \times 9$ filtering bits for the ciphertext difference. However, in the differential MITM attack, these filtering bits cannot be used for the upper part. To overcome this, we propose a variant of the differential MITM attack, named DMA-4, in Figure 10. DMA-4 stores pairs of data so that filters of the ciphertext difference can be used for the upper part even before the match.

**Storing Pairs Instead of Single Messages** In DMA-4, the MITM procedure is carried out for all the data in a structure rather than for a single $(P, C)$. Without the pivot $(P, C)$, both ciphertexts $C, \tilde{C}$ are stored in Step 11. Then a match happens in Step 15 with probability $2^{-2s+1}$ as there are $2^{2s-1}$ pairs in total for a structure. From the formula of the time complexity, this variant is equivalent to the original differential MITM attack.

**Combining Both** The generalized differential MITM attack, named GDMA can be obtained by combining DMA-3 and DMA-4, as shown in Figure 11. It supports guessing a subset of $k_{in}$ and $k_{out}$, as well as exploiting the filtering bits of the
Differential MITM attack 3 (DMA-3)

1. If $\ell_{in} \leq p + 1$:
   - Construct $2^{p-\ell_{in}+1}$ structures, each of $2^2 = 2^{\ell_{in}}$ plaintexts.
2. Construct $2^{p-\ell_{in}+1}$ structures, each of $2^2 = 2^{\ell_{in}}$ plaintexts.
3. else:
   - Construct a partial structure of $2^4 = 2^{(\ell_{in} + 1)/2}$ plaintexts.
5. Guess $u$ for $k'_c \subset k_c$:
6. For each structure $S_i$ in $S$:
   - For each $(P, C)$ from the structure $S_i$:
     - Guess $i$ for $k'_i$ in and let $k'_i = k_i - k_i' - k_i''$.
     - Compute $2^{\ell_{in}} \bar{P}$ from $P$ such that $\ell_{in}$ condition bits are satisfied.
   - If $P \notin S_i$, go to Step 9; otherwise, let $\bar{C} = E(\bar{P})$.
   - If $(\bar{C}, \bar{C})$ have no difference on the $n - \ell_{out}$ bits, store $(\bar{C}, i)$ in a table $H$.
   - Guess $j$ for $k'_j$ out and let $k'_j = k'_o - k'_j' - k'_j''$.
   - Compute $2^{\ell_{out}} \bar{C}$ from $C$ such that $\ell_{out}$ more bits of $\Delta Y$ are satisfied.
   - For each $i \in H(\bar{C})$:
     - Get $(\bar{P}, P)$ and $(u, i, j)$. Extract $2^{k'_i + k'_o - \ell_{in} - \ell_{out}}$ candidates for $k'_i$ and $k'_o$ for each pair.
     - Update the key counters or test directly.

   - When $\ell_{in} \leq p + 1$, $D = 2^{p+1}$; otherwise, $D = 2^{(p+\ell_{in}+1)/2}$.
   - Depending on Step 17, $M = \min\{\max\{2^4, 2^{k'_i + k'_o - \ell_{in}}\}, \max\{2^4, 2^{k'_i + k'_o - \ell_{out}}\}\}$ or $\max\{2^4, \min\{2^{k'_i + k'_o - \ell_{in}}, 2^{k'_i + k'_o - \ell_{out}}\} \times 2^{\ell_{out} - n}\}$.

- $T_0 = D \times (2^{k'_i + k'_o} + |k'_i| + |k'_o|)$ for partial encryption and decryption;
- $T_1 = D \times (2^{k'_i + k'_o} + s + \ell_{out} - n - 1 - \ell_{in}) + D \times \frac{1}{2^{k'_i + k'_o} + |k'_i| + |k'_o|} + s + \ell_{out} - n - 1 - \ell_{in}$
- $\times 2^{k'_i + k'_o - n - 1 - \ell_{in} - \ell_{out}}$ for getting pairs that satisfying some conditions;
- $T_2 = 2^{k'_i + k'_o + p - n} \cdot c.$

The GCDA covers all the variants from DMA-1 to DMA-4. This attack is comparable with the GCDA in Figure 5. Since $T_2, T_3$ of the time complexity for both attacks are the same if the key bits to be guessed are the same, we focus on the comparison of $T_0, T_1$. When $|k'_i| = 0$ or $|k'_o| = 0$, it becomes a classical differential attack as the meet-in-the-middle phase disappears. Therefore, we restrict $|k'_i| > 0$ and $|k'_o| > 0$ for the GCDA.

Figure 9: The differential MITM attack that allows any possible key guessing strategy.
Differential MITM attack 4 (DMA-4)

1. If $\ell_{in} \leq p + 1$:
2. Construct $2^{p-\ell_{in}+1}$ structures, each of $2^s = 2^{2\ell_{in}}$ plaintexts.
3. else:
4. Construct a partial structure of $2^s = 2^{(p+\ell_{in}+1)/2}$ plaintexts.
5. Query for the ciphertexts. Store the plaintext-ciphertext pairs in $S$.
6. Guess $u$ for $k_{\gamma}$:
7. For each structure $S_u$ in $S$:
   8. Guess $i$ for $k'_{in}$ where $k_{in} - k_{\gamma} = k'_{in}$;
   10. Get $2^{2s-1-\ell_{in}+\ell_{out}-n}$ pairs $(P, \tilde{P})$ satisfying $\ell_{in} + \ell_{out} - n$ condition bits.
11. Store $(C, \tilde{C}, i)$ in a table $H$.
12. Guess $j$ for $k'_{out}$ where $k_{out} - k_{\gamma} = k'_{out}$;
14. Get $2^{2s-1-n}$ pairs $(C, \tilde{C})$ satisfying $\Delta y$.
15. For each $i \in H(C, \tilde{C})$:
   16. Get $(u, i, j)$.
17. Update the key counters or test directly.

- When $\ell_{in} \leq p + 1$,
- $D = 2^{p+1}$; otherwise, $D = 2^{(p+\ell_{in}+1)/2}$.
- Depending on Step 17, $M = \min\{\max\{D, 2^{|k_{in}\cup k_{out}| - |k_{\gamma}|}\}, \max\{2^s, 2^{|k_{in}\cup k_{out}|}\}\}$
  or $\max\{2^s, \min\{2^{2s-1-\ell_{in}+\ell_{out}-n} + |k'_{in}|, 2^{2s-1-n} + |k'_{out}|\}\}$.
- The time complexity is
\[
T = D \cdot (2^{|k_{in}|} + 2^{|k_{out}| + s - \ell_{in} + \ell_{out} - n - 1} + 2^{|k_{out}| + s - n - 1} + 2^{2s-1-n}) + 2^{|k_{in}\cup k_{out}| + p - n} + T_{search}
\]
\[
\approx D \cdot (2^{|k_{in}|} + 2^{|k_{out}|}) + 2^{|k_{in}\cup k_{out}| + p - n} + T_{search}
\]

Figure 10: The differential MITM attack that stores pairs of data.

Let us look into $T_0$ and $T_1$. For the partial encryption and decryption, $T_0^{GDMA}$ is smaller than $T_0^{GDMA}$. For GDMA, $T_1$ has three terms:

\[
T_{1,0}^{GDMA} = D \cdot 2^{|k'_{in}| + |k'_{in}| + s - \ell_{in} + \ell_{out} - n - 1},
\]

\[
T_{1,1}^{GDMA} = D \cdot 2^{|k'_{in}| + |k'_{out}| + s - \ell_{in} + \ell_{out} - n - 1 - \ell_{out}},
\]

\[
T_{1,2}^{GDMA} = D \cdot 2^{|k'_{in}| + |k'_{in}| + |k'_{out}| + s + \ell_{in} - n - 1 - \ell_{in}' - \ell_{out}'.
\]

Note that $T_{1,2}^{GDMA} = T_1^{GDMA}$. When $\ell_{in}' < |k'_{in}|$ and $\ell_{out}' < |k'_{out}|$, $T_{1,1}^{GDMA}$ and $T_{1,0}^{GDMA}$ are smaller than $T_{1,2}^{GDMA}$.

Applying the GDMA to AES-256. Using the GDMA, let $k'_{in} = k_{12}^{[12, 13, 14, 15]}$, $k'_{out} = (k_{11}^{[13]}, k_{12}^{[8, 12]}$, and we have $M = 2^{|k_{in}\cup k_{out}|} = 2^{184}$, $T = D(2^{32} + 2^{32} + 2^{24} + 2^{24} + 2^{28} + 2^{144}) + T_{search} = 2^{144}$. Now the overall time complexity is as good as the time complexity of GDMA. The comparison of key recovery attacks on 12-round AES-256 is shown in Table 1.
Generalized differential MITM attack (GDMA)

1. If \( \ell_{in} \leq p + 1 \):
   - Construct \( 2^{p-\ell_{in}+1} \) structures, each of \( 2^\ell \) plaintexts.
2. else:
   - Construct a partial structure of \( 2^s = 2^{(p+\ell_{in}+1)/2} \) plaintexts.
3. Query for the ciphertexts. Store the plaintext-ciphertext pairs in \( S \).
4. Guess \( u_i \) for \( k_i \), and let \( \ell_i^* = \ell_i - k_i^* \).
5. For each structure \( S_e \) in \( S \):
   - Guess \( i \) for \( k_i^* \mid \{ |k_i^{\prime} | \rangle > 0 \) and let \( \ell_i^{\prime} = \ell_i - k_i - k_i^{\prime} \).
   - Do partial encryption (and decryption).
   - Get \( 2^{2^s-1-\ell_i + \ell_i^{\prime} + n} \) pairs \((P, \hat{P})\) satisfying \( \ell_i^* + \ell_i^{\prime} - n \) condition bits.
6. Store \((C, \hat{C}, i)\) in a table \( H \).
7. Guess \( j \) for \( k_i^{\prime} \mid \{ |k_i^{\prime} | \rangle > 0 \) and let \( \ell_j^{\prime} = \ell_j - k_j - k_j^{\prime} \).
8. Do partial decryption (and encryption).
9. Get \( 2^{2^s+1-\ell_i + \ell_j + n} \) pairs \((P, \hat{P})\) satisfying \( n - \ell_j^* \) bits of \( \Delta y \).
10. For each \( i \in H(C, \hat{C}) \):
    - Get \((P, \hat{P})\) and \((u, i, j)\). Extract \( 2^{\ell_i + \ell_j + n} \) candidates for \( k_i^* \) and \( k_j^* \) for each pair.
11. Update the key counters or test directly.

- When \( \ell_{in} \leq p + 1 \), \( D = 2^{p+1} \); otherwise, \( D = 2^{(p+\ell_{in}+1)/2} \).
- Depending on Step 17, \( M = \min \{ \max \{ D, 2^{|k_{in}^{\prime} \cup k_{out}^{\prime} | - |k_i^{\prime} |} \}, \max \{ M_0, 2^{|k_{in}^{\prime} \cup k_{out}^{\prime} |} \} \} \) or \max \{ 2^s, \min \{ 2^{2^s-1-\ell_{in} + \ell_i^{\prime} + n - \ell_i^*}, 2^{2^s-1-n+\ell_i^{\prime} + \ell_i^*} \} \}.
- The time complexity
  \[
  T = D \cdot (2^{2^s-1-\ell_{in} + \ell_i^{\prime} + \ell_i^* + n} + 2^{2^s-1-\ell_{in} + \ell_i^{\prime} + \ell_i^* + n} + 2^{2^s-1-n+\ell_i^{\prime} + \ell_i^*} + 2^{2^s-1-n+\ell_i^{\prime} + \ell_i^*}) + 2^{|k_{in}^{\prime} \cup k_{out}^{\prime} |} \cdot \epsilon + T_{search}
  \]

Figure 11: Generalized differential MITM attack.

<table>
<thead>
<tr>
<th>Data</th>
<th>Time</th>
<th>Memory</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 2^{214} )</td>
<td>( 2^{39} )</td>
<td>DMA-1 [BDD+23]</td>
<td></td>
</tr>
<tr>
<td>( 2^{208} )</td>
<td>( 2^{184} )</td>
<td>DMA-2</td>
<td></td>
</tr>
<tr>
<td>( 2^{176} )</td>
<td>( 2^{184} )</td>
<td>DMA-3</td>
<td></td>
</tr>
<tr>
<td>( 2^{144} )</td>
<td>( 2^{184} )</td>
<td>GDMA</td>
<td></td>
</tr>
<tr>
<td>( 2^{145} )</td>
<td>( 2^{128} )</td>
<td>GCDA</td>
<td></td>
</tr>
<tr>
<td>( 2^{185} )</td>
<td>( 2^{89} )</td>
<td>GCDA</td>
<td></td>
</tr>
</tbody>
</table>

**Remark 4.** Given a differential and the numbers of added rounds, the term \( 2^{|k_{in} \cup k_{out} | + p - n} \) is fixed and the time complexity can never be smaller than it. In either the classical differential attack or the differential MITM attack, if the overall time complexity is as small as \( 2^{|k_{in} \cup k_{out} | + p - n} \), the time complexity of the


attack is optimized. We can see that the time complexity of the 12-round attack on AES-256 is already optimal for both GCDA and GDMA.

4.4 Comparison and Discussion

With GCDA and GDMA proposed, the comparison between the differential MITM attack and the classical differential attack becomes more apparent. In essence, GDMA can be seen as a variant of GCDA by implementing the partial encryption/decryption and construction of pairs in an MITM manner. The following are some suggestions for a differential attack.

- Choose GCDA in the following cases: $E_{in} = I$ or $E_{out} = I$; $k'_m \cup k'_n$ or $k'_m \cup k'_n$ is empty.
- When $2|k'_m \cup k'_n \cup k'_{out}| \cdot D > 2^k$, choose GDMA as GDMA has a lower complexity for doing partial encryption/decryption.
- In other cases, compute the complexities for both and choose the better one.
  - When $\ell'_{in} \leq |k'_m \cup k'_{in}|$ and $\ell'_{out} \leq |k'_m \cup k'_{out}|$, choose GDMA if the time complexity is of the greatest concern.

4.5 Application to GIFT-64

Besides AES-256, we compare the GCDA and GDMA on another cipher GIFT. GIFT is a lightweight block cipher proposed by Banik et al. at CHES’ 2017 [BPP+17]. According to the block sizes, GIFT has two versions GIFT-64 and GIFT-128. Both versions use a 128-bit master key. We revisit the differential attack on 26-round GIFT-64 from [SWW21] (see Figure 13) as this attack covers the largest number of rounds to date.

The attack exploits an 18-round related-key differential trail of probability $2^{-p} = 2^{-58}$ and adds three and five rounds before and after the differential trail, respectively. The parameters of the attack are as follows: $\ell_{in} = 56$, $\ell_{out} = 64$; $k_{in}$ contains 40 key bits while $k_{out}$ consists of 112 key bits. In addition, $k_{in} \cap k_{out} = k_{in}$. The attack in [SWW21] fits in with the GCDA well, where none of the key bits is guessed and $\epsilon = 2^{23.9} \cdot \frac{1}{16 \cdot 26} = 2^{15.2}$ 26-round encryptions. Using $2^{3.96}$ pairs of structures, the attack can be mounted with data and time complexities $2^{59.96}$ and $2^{123.23}$. If the full $k_{in}$ is guessed in advance, an attack can be mounted with similar complexities.

In this case, the basic different MITM attack does not work as $|k_{out}|$ is too large. Can the GDMA be applied? In GDMA, $k'_m \cup k'_{out}$ should not be empty. The same applies to the other side. There would be $2^{59.96} + 56 \cdot 2^{k'_{m} + |k'_{out}| - \ell'_{out}}$ possible $(C, \tilde{C})$ in the MITM phase. As the last three rounds are fully active, it requires guessing a lot of key bits before reaching some filters. In fact, more than 40 key bits should be guessed before reaching a filter, i.e., $\ell'_{out} > 0$. As a result, $|k'_m \cup k'_{out}|$ should be less than 12 as $2^{59.96} + 56 \cdot 2^{k'_{m} + |k'_{out}| - \ell'_{out}} < 2^{128}$. Let $k'_m = k_{in}$, $|k'_{out}| > 0$, and $|k'_{m}| + |k'_{out}| < 12$. Then $T_2$, $T_3$ remain as in the GCDA but $T_1$ is larger. However, $T_2$ is the dominant term. Therefore, a generalized differential MITM attack is possible, but it is not better than the classical differential attack.
5 Conclusion

In this paper, we revisit the differential MITM attack which was proposed recently in [BDD+23]. Inspired by the commonly-used techniques used in the classical differential attack, we propose several variants of the differential MITM attack and present a new one by generalizing the basic differential MITM attack in several aspects. As for applications, we revise the 24-round attack on SKINNY-128-384 and provide better results on 12-round AES-256 using the classical differential attack and the generalized differential MITM attack. Unlike the basic differential MITM attack, the generalized differential MITM attack supports various strategies for guessing the key. Future work would be to build an automatic system for searching for the best guessing strategy.

References


A Supplementary Materials

A.1 Differential MITM Attack on SPECK

In this subsection, we take SPECK as an example to show that an extra key material \( k_{\text{extra}} \) might be needed for recovering the master key.

*Description of SPECK.* **SPECK** [BSS+13] is a family of lightweight block ciphers based on addition, rotation, and XOR operations. Its members **SPECK**\( n \) are characterized by the block size \( 2^n \) and key size \( mn \). Figure 12 provides a schematic view of the round function and the key schedule of **SPECK**. The round function of **SPECK** is defined as:

\[
(x_{i+1}, y_{i+1}) = \mathcal{R}_{ki}(x_i, y_i) = ((x_i \gg a) \oplus y_i) \oplus k_i, \quad (y_i \ll b) \oplus ((x_i \gg a) \oplus k_i),
\]

where \( k_i \) is the round key, \( a, b \) are constants. The key schedule takes an initial \( m \)-word master key \((l_m, \ldots, l_0, k_0)\) and from it generates a sequence of round key words \( k_0, k_1, \ldots \) via:

\[
l_{i+m-1} = (k_i \oplus (l_i \gg a)) \oplus i, \quad k_{i+1} = (k_i \ll b) \oplus l_{i+m-1}.
\]

The key schedule has a property that if \( m \) consecutive round keys of **SPECK** are known, we can efficiently invert the key schedule to determine the original \( m \) master key words.

*Differential attacks on SPECK.* In [Din14], Dinur mounted differential attacks on **SPECK** using existing differential trails. As the first round of **SPECK** acts as a whitening layer, it can be covered for free. Take **SPECK**32-64 as an example.
Given an $r$-round differential trail of probability $2^{-p}$, where $p$ is smaller than but very close to $2^n$, $1 + r + 4$ rounds are attacked in [Din14]. The attack extracts candidates for the last four round keys. With four consecutive round keys, the master key can be computed by performing the key schedule backward. Further, test the correctness of the master key by trial encryptions.

In the differential MITM attack, adding four rounds after the differential trail does not work but it is possible to attack $1 + 2 + r + 2$ rounds. In this case, $k_{in}$ contains the first two round keys and $k_{out}$ consists of the last two round keys. The time complexity for the attack excluding the exhaustive search would be $2^p \times (2^{32} + 2^{32}) + 2^{32+32-32+p}$. The number of candidates for the four round keys is about $2^{32+p}$, which is smaller than but very close to $2^k$ if $p$ is close to $k$. Note that these four round keys are not consecutive and it is hard to derive the master key from them. We find that neither the original differential MITM attack nor DMA-1 works, but DMA-2 does. The idea is to shortlist the candidates and guess two neighboring round keys (i.e., $|k_{extra}| = 32$) so that deriving the master key becomes easy and the time complexity would not exceed $2^k$. This requires $k - 2n + p + |k_{extra}| - h < k$, i.e., only a fraction $2^{-h}$, $h > p$ of the candidates are selected for exhaustive search. To make the right key fall in the shortlist with high probability, the data complexity needs to be doubled or tripled.

In the case where $p$ is small, say $p < 16$, i.e., the differential is of very high probability, it is possible to attack $1 + 3 + r + 3$ rounds. That is, both $k_{in}$ and $k_{out}$ have three round keys and $k_{in} \cap k_{out}$ has two words. As the relation between $k_{in}$ and $k_{out}$ is highly complicated, $k_{in} \cap k_{out}$ has no explicit form. Thus, the $|k_{in} \cap k_{out}|$-bit filter can be used only when combinations of $(k_{in}, k_{out})$ are formed, which makes $f_{extra} = 0$.
A.2 Original Differential MITM Attacks on SKINNY-128-384

The 23-Round Attack We almost copy the core attack against 23-round SKINNY-128-384 from [BDD+23] below.

1. Ask for the encryption of the whole codebook.
2. Randomly pick one plaintext/ciphertext pair \((P, C)\).
3. For each possible value \(i\) of \(k_{in}\), compute the tuple \((P, \tilde{P}, i)\) so that the difference on the state after the 6th S-box layer is exactly the input difference \(\Delta x\) of the distinguisher. Get \(\tilde{C} = E(\tilde{P})\) from the codebook. Store all \((\tilde{C}, i)\) in a hash table. This step requires guessing \(k_{in}\).
4. Similarly, for each possible value \(j\) of \(k_{out}\), compute the tuple \((C, \tilde{C}, j)\) so that the difference on the state before the 19th S-box layer is exactly the output difference \(\Delta y\) of the distinguisher.
5. For each of \(\tilde{C}\), check for possible matches on the hash table. The match is performed on both the new ciphertext \((i.e., \tilde{C} = \tilde{C})\) as well as on the linear relations between the subkey bytes of the upper and lower guess.
6. Each match leads to a (full) key candidate that can be tried against very few additional plaintexts.
7. Repeat from Step 2 until the right key is retrieved.

The data complexity of this attack is \(2^{128}\) as it uses the full codebook. The memory complexity is determined by the hash table, which takes \(2^{211+8} = 2^{248}\). The time complexity is

\[
\mathcal{T} = 2^{105.9} \times (2^{248} + 2^{256}) + 2^{248+256−128+105.9−15+8} + 2^{384−128+105.9} = 2^{361.9}.
\]

The 24-Round Attack The attack on 23-round SKINNY-128-384 can be extended to an attack on 24 rounds without increasing the overall time complexity. The attack adds one more round to the end and exploits the fact that the round key is only applied to the first two rows of the internal state.

1. Ask for the encryption of the whole codebook.
2. Pick \(2^{64}\) plaintext/ciphertext pairs \((P_{\ell}, C_{\ell})\) such that \(MC^{-1}(C_{\ell})\) is constant on the last two rows.
3. Compute all possible tuples \((P_{\ell}, \tilde{P}_{\ell}, i)\) for each value \(i\) of \(k_{in}\) and each \(P_{\ell}\) from the structure defined at the previous step such that the state difference after the 6th S-box layer is 0x02. Store them in a hash table. The memory complexity is \(2^{248+64} = 2^{312}\) 504-bit words.
4. For each value \(j\) of \(k_{out}\) and each state \(S_{23, \ell}\) (the state after SC of Round 23) coherent with Step 2 \((i.e., 2^{64}\) states, one for each possible value of the subtweakey of Round 23), compute all possible tuples \((S_{23, \ell}, \tilde{S}_{23, \ell}, j)\) so that the difference on the state before the 19th S-box layer is 0x64 on the four active bytes.
5. Check for possible matches on the hash table. The match is now performed on three quantities:
the difference between the last states: \( C \oplus \tilde{C} = M_C \circ SR(S_{23} \oplus \tilde{S}_{23}) \). This is a 64-bit filter because the difference is zero on the two last rows since \( MC^{-1}(C) \) is constant on these rows.

- the filter on the keys (from key schedule equations): a 120-bit filter.
- the filter on the subtweak \( K_{23} \). Indeed, since \( k_{\text{in}} \cup k_{\text{out}} \) generates the master key, \( K_{23} \) can be rewritten as \( f(k_{\text{in}}) \oplus g(k_{\text{out}}) \) where \( f \) and \( g \) are both linear and, because of the linearity of all the operations, the equation \( C = MC(SR(S_{23} \oplus K_{23})) \) can thus be rewritten as \( C \oplus MC(SR(f(k_{\text{in}})) = MC(SR(S_{23} \oplus k_{\text{out}}))) \). This represents a 64-bit filter.

6. Each match leads to a (full) key candidate that can be tried against very few additional plaintexts.

7. Repeat from Step 2 until the right key is retrieved.

Since \( 2^{64} \) plaintext/ciphertexts are involved in the basic MITM step, the memory complexity is increased by \( 2^{64} \). In Step 5, only \( 64 + 120 + 64 = 248 \) filtering bits are mentioned, leading to a time complexity \( 2^{41.9} \times 2^{248+64+256+64-120-64} = 2^{425.9} \), but the authors claimed the time complexity remains \( 2^{361.9} \).
Figure 13: Differential attack on 26-round GIFT-64 [SWW21].