Generic Differential Key Recovery Attacks and Beyond

Ling Song[1], Huimin Liu[1], Qianqian Yang[2,3]✉, Yincen Chen[1], Lei Hu[2,3], and Jian Weng[1]

[1] College of Cyber Security, Jinan University, Guangzhou, China
[2] Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
songling.qs@gmail.com,liuhuimin301@gmail.com,yangqianqian@iie.ac.cn,
icsnow98@gmail.com,hulei@iie.ac.cn,cryptjweng@gmail.com

Abstract. At Asiacrypt 2022, a holistic key guessing strategy was proposed to yield the most efficient key recovery for the rectangle attack. Recently, at Crypto 2023, a new cryptanalysis technique–the differential meet-in-the-middle (MITM) attack–was introduced. Inspired by these two previous works, we present three generic key recovery attacks in this paper. First, we extend the holistic key guessing strategy from the rectangle to the differential attack, proposing the generic classical differential attack (GCDA). Next, we combine the holistic key guessing strategy with the differential MITM attack, resulting in the generalized differential MITM attack (GDMA). Finally, we apply the MITM technique to the rectangle attack, creating the generic rectangle MITM attack (GRMA). In terms of applications, we improve 12/13-round attacks on AES-256. For 12-round AES-256, by using the GDMA, we reduce the time complexity by a factor of 2^{62}; by employing the GCDA, we reduce both the time and memory complexities by factors of 2^{61} and 2^{56}, respectively. For 13-round AES-256, we present a new differential attack with data and time complexities of 2^{89} and 2^{240}, where the data complexity is 2^{37} times lower than previously published results. These are currently the best attacks on AES-256 using only two related keys. For KATAN-32, we increase the number of rounds covered by the differential attack from 115 to 151 in the single-key setting using the basic differential MITM attack (BDMA) and GDMA. Furthermore, we achieve the first 38-round rectangle attack on SKINNYe-64-256 by using the GRMA.

Keywords: Differential cryptanalysis, Rectangle attack, Meet-in-the-middle, Key recovery, AES, KATAN, SKINNYe

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 in mounting 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 reveal the non-randomness of the differential.

Since the introduction of differential cryptanalysis, many improvements have been proposed for the key recovery: structures of data [BS92], conditional differentials [KMN10], probabilistic neutral bits [AFK+08], the early abort technique [LKKD08], the probabilistic extensions [SYC+24] and so on. Among the techniques for key recovery, the key guessing strategy has received considerable attention. A common approach is to guess key bits corresponding to S-boxes in a default order as in [SWW21]. To improve complexity, the dynamic key-guessing technique [WWJZ18] and a delicate key-guessing technique [BCF+21] that takes advantage of the structure of the S-box have been introduced. In [BDD+24], an automated tool was developed for the first time to find the best key guessing order for block ciphers that use a bit-permutation as the linear layer. Recently, significant progress was made on the differential key recovery in [BDD+23] where the key is recovered in a meet-in-the-middle (MITM) manner, and the new attack is called the differential MITM attack. In classical differential attacks, pairs of data are constructed first, and for each pair, the attacker identifies the keys, under which the differential is respected. In the differential MITM attack, pairs of data are constructed together with the keys it suggests. The new attack has produced favorable results on block ciphers `SKINNY`-128-384 and `AES`-256. Later, this attack was extended to truncated differentials in [AKM+24].

The rectangle attack [BDK01], which is a variant of the differential attack, combines two differentials and utilizes the non-randomness of quartets. There have been a series of works on the key recovery of rectangle attacks [BDK01, BDK02, ZDM+20, DQSW21], each employing a different key guessing strategy. In [SZY+22, YSZ+24], the previous key recovery attacks are unified into a generic rectangle key recovery attack. Notably, the generic rectangle attack supports any key guessing strategy and can optimize complexity by selecting the most appropriate one.

Motivations. Even though the key guessing strategy received great attention in both differential attacks and rectangle attacks, the strategies differ. In the rectangle attack, it means guessing some key bits before any pairs or quartets are generated, which allows for filtering out some wrong pairs without even generating them. This is a natural approach. Under the guessed key some conditions of the distinguisher can be checked. In rectangle attacks, the number of conditions or filters is doubled as there are two pairs. Therefore, guessing some key bits initially is likely to be advantageous. The key guessing strategy lies in determining which part of the key bits to guess in advance. In contrast, in the differential attack, the attacker first generates the pairs that potentially satisfy the differential and then guesses some key bits. Here, the strategy focuses on the order in which the key bits are guessed.

From now on, let the key guessing strategy refer to the one from the rectangle attack. Although in the differential attack the number of filters under the guessed key bits is not doubled, it is interesting to investigate whether guessing some key bits in advance affects the time complexity of the differential attack. Second, the differential MITM attack employs a fixed key guessing strategy, can it be generalized to support any key guessing strategy? Further, can the MITM technique be integrated into the generic rectangle attack [SZY+22, YSZ+24]? These questions form the starting point of this paper.

Our contributions. Guessing some key bits in advance before any pairs of data are generated is a technique that has been used in the rectangle attack. It plays a core role in optimizing the time complexity. In this paper, we introduce the key guessing strategy from the rectangle attack to the differential attack and also introduce the MITM key recovery to the rectangle attack, resulting in three generic key recovery attacks as follows.

GCDA A generic classical differential attack is proposed that first considers the key guessing strategy. Notably, the key guessing strategy can be any, as in the generic rectangle attack [SZY+22, YSZ+24]. Therefore, the GCDA encompasses the previous differential attack with no key bits guessed in advance.

GDMA A generalized differential meet-in-the-middle attack is proposed that extends the basic differential meet-in-the-middle attack (BDMA) [BDD+23] and allows a flexible key guessing strategy.

GRMA A generic rectangle meet-in-the-middle attack is proposed that incorporates the MITM technique into the rectangle key recovery.

To demonstrate the efficiency of these attacks, we revisit the attacks on AES-256, KATAN-32, and SKINNYe-64-256 v2 using previously published distinguishers. The following results are obtained and comparisons of the results with previous ones are summarized in Table 1.

- Using the GCDA and GDMA, the time complexity of the 12-round attack on AES-256 in the related-key setting can be optimized to 2^{144}. Notably, using the BDMA, the time complexity cannot be reduced below 2^{206}. This improvement is primarily due to the flexible key guessing strategy. Additionally, we extend the attack to 13 rounds, achieving lower data and time complexities than previous works. Specifically, the data complexity is reduced by a factor of 2^{37}. These are the best attacks so far on AES-256 using only two related keys.
- We add various numbers of rounds to a 42-round differential of KATAN-32 and compare the three attacks, *i.e.*, BDMA, GCDA, and GDMA. We confirm some properties of the time complexity of these attacks: the GDMA encompasses the BDMA; GDMA outperforms GCDA under certain conditions (see Section 3.3). Using a 91-round differential from the literature, we improve the differential attack on KATAN-32 from 115 rounds to 151 rounds, marking the best differential attack on KATAN-32 to date.

Table 1: Summary of the cryptanalytic results. RK: related-key. SK: single-key.

Cipher	Rounds	Data	Time	Memory	Setting	Type
AES-256	12	2^{89}	2^{214}	2^{89}	RK	BDMA [BDD+23]
			2^{206}	2^{184}	RK	BDMA [BDD+23]
			$\mathbf{2^{185}}$	$\mathbf{2^{89}}$	RK	GCDA (Section 4.1)
			$\mathbf{2^{144}}$	$\mathbf{2^{184}}$	RK	GDMA (Section 4.1)
			$\mathbf{2^{145}}$	$\mathbf{2^{128}}$	RK	GCDA (Section 4.1)
	13	2^{126}	2^{253}	2^{89}	RK	BDMA [BDF23]
		2^{126}	2^{250}	2^{231}	RK	BDMA [BDF23]
		$\mathbf{2^{89}}$	$\mathbf{2^{248}}$	$\mathbf{2^{89}}$	RK	GCDA (Section 4.1)
		$\mathbf{2^{89}}$	$\mathbf{2^{240}}$	$\mathbf{2^{144}}$	RK	GCDA (App. A.3)
KATAN-32	115	2^{32}	$2^{79.98}$	–	SK	Differential [AL13]
	151		$2^{79.98}$	2^{38}	SK	BDMA (Section 4.2)
SKINNYe-64-256 v2	37	$2^{62.8}$	$2^{240.03}$	$2^{62.8}$	RK	Rectangle [QDW+22]
	38	$\mathbf{2^{65.4}}$	$\mathbf{2^{251.07}}$	$\mathbf{2^{254.8}}$	RK	GRMA (Section 4.3)

– We apply the GRMA to SKINNYe-64-256 and extend the rectangle attack by one round. This is the best attack on SKINNYe-64-256 v2 so far. Note that, using the same distinguisher, the generic rectangle attack in [SZY+22, YSZ+24] cannot cover as many rounds. This confirms the advantage of the MITM technique in certain cases of the rectangle key recovery.

Our work demonstrates that guessing key bits in advance is an excellent complement to existing techniques for differential attacks. Furthermore, allowing flexible key guessing strategies enhances the power of the differential MITM attack. Finally, the MITM technique can be extended to the key recovery of the rectangle attack.

Organization. The rest of the paper is organized as follows. In Section 2, we recall the generic rectangle attack and the differential MITM attack. In Section 3, we introduce three generic key recovery attacks, *i.e.*, the generic classical differential attack (GCDA), the generalized differential MITM attack (GDMA), and the generic rectangle MITM attack (GRMA). In Section 4, we provide new cryptanalytic results on AES-256, KATAN-32, and SKINNYe-64-256 using the newly proposed key recovery attacks. Finally, we conclude the paper in Section 5.

2 Preliminaries

2.1 Notations

E	The block cipher $E = E_f \circ E_m \circ E_b$ with a distinguisher over E_m
n	The block size
k	The key size
k_b (resp. k_f)	The subset of subkey bits that are employed in E_b (resp. E_f)

k'_b (resp. k'_f)	The part of k_b (resp. k_f) guessed in advance
k^*_b (resp. k^*_f)	$k^*_b = k_b \setminus k'_b$ (resp. $k^*_f = k_f \setminus k'_f$)
$\lvert \cdot \rvert$	The size of an object
r_b (resp. r_f)	The dimension of the space spanned by all possible plaintext (resp. ciphertext) differences
r'_b (resp. r'_f)	The number of conditions that can be verified under k'_b (resp. k'_f)
r^*_b (resp. r^*_f)	$r^*_b = r_b - r'_b$ (resp. $r^*_f = r_f - r'_f$)
GCRA	The generic classical rectangle attack [SZY+22]
GCDA	The generic classical differential attack
BDMA	The basic differential MITM attack [BDD+23]
GDMA	The generalized differential MITM attack
GRMA	The generic rectangle MITM attack

2.2 The Generic Rectangle Key Recovery Attack

At Asiacrypt 2022, a generic rectangle key recovery attack was proposed by Song *et al.* [SZY+22]. It contains a generic key recovery algorithm and a strategy for finding the best attack. To make a distinction, we call the key recovery attacks that do not use the MITM technique classical attacks, and then the attack in [SZY+22] is called the generic classical rectangle attack (GCRA).

As shown in Figure 1, given a block cipher E, we treat it as the composition of three sub-ciphers: $E = E_f \circ E_m \circ E_b$. Suppose the probability of the boomerang distinguisher over E_m is $Pr = 2^{-2p}$. When we extend the differential outwards with probability 1, Δx will propagate to the plaintext difference ΔP over E_b^{-1} and Δy will propagate to the ciphertext difference ΔC over E_f. Let all possible ΔP span a space of dimension r_b. Similarly, let all possible ΔC span a space with dimension r_f. Suppose that it requires subkey information k_b (resp. k_f) to verify the difference Δx (resp. Δy) for plaintext (resp. ciphertext) pairs.

In the GCRA, some key bits may be guessed in advance to sieve the data faster. Suppose a part of k_b and k_f, denoted by k'_b, k'_f, is guessed at first, $0 \le |k'_b| \le |k_b|, 0 \le |k'_f| \le |k_f|$. With the guessed subkey bits, an r'_b-bit condition on the top and an r'_f-bit condition on the bottom can be verified. Finally, let $r^*_b = r_b - r'_b$ and $r^*_f = r_f - r'_f$. The specific steps for the algorithm are as follows.

1. Phase of data collection. Collect and store y structures of 2^{r_b} plaintexts. The time complexity of this step is T_0.
2. Phase of extracting key candidates.
   (a) Subkeys guessed. For each data (P_1,C_1), partially encrypt P_1 and partially decrypt C_1 under the guessed subkey bits. There are $y^* = y \cdot 2^{r'_b}$ sub-structures of $2^{r_b *} = 2^{r_b - r'_b}$ plaintexts. The time complexity of this step is T_1.
   (b) Pairs constructed. Insert all the obtained (P^*_1, C^*_1) into a hash table by the inactive bits of P^*_1 or C^*_1 to construct a set of pairs $S = \{(P^*_1, C^*_1, P^*_2, C^*_2)\}$ or $S = \{(P^*_1, C^*_1, P^*_3, C^*_3)\}$. The time complexity is T_2.

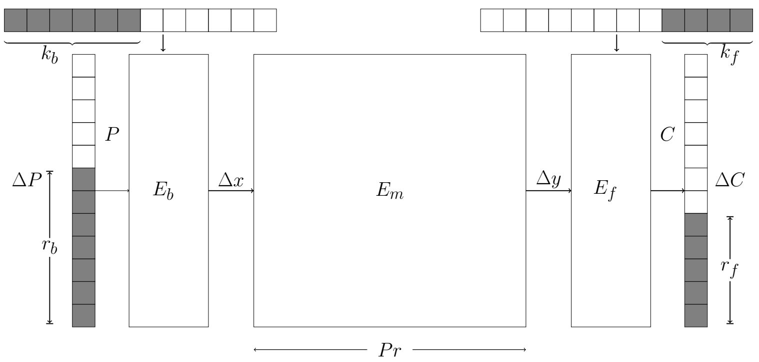

Figure 1: A high-level description of the rectangle/differential MITM attack

(c) Quartets generated. Insert S into a hash table by the inactive bits of C_1^* and C_2^* or P_1^* and P_3^*. Then, generate the quartet for each index.

(d) Quartets processed and key information extracted. Determine the key candidates involved in E_b and E_f and increase the corresponding counters. The time complexity of this step is T_3.

3. Phase of exhaustive search. Guess the remaining unknown key bits according to the key schedule algorithm and exhaustively search over them to recover the correct key. The time complexity of this step is T_4.

Complexities. The data complexity is $D = y \cdot 2^{r_b} = \sqrt{s}2^{n+1+p}$ where s is the expected number of right quartets and $y = \sqrt{s}2^{n/2-r_b+p}$. The memory complexity is $M = f_M(D, k'_b, k'_f) = D + \min\{D \cdot 2^{r_b^*-1}, D^2 \cdot 2^{r_f^*-n-1}\} + 2^{t+|k_b \cup k_f|-|k'_b \cup k'_f|}$ for storing the data, the pairs and the key counters, where $0 \leq t \leq |k'_b \cup k'_f|$. The time complexity $T = f_T(D, k'_b, k'_f)$ is composed of four parts. The time complexity of collecting data is $T_0 = D$, the time complexity of doing partial encryption and decryption under guessed key bits is

$$T_1 = 2^{|k'_b \cup k'_f|} \cdot D,$$

the time complexity of generating pairs is

$$T_2 = 2^{|k'_b \cup k'_f|} \cdot D \cdot \min\{2^{r_b^*-1}, D \cdot 2^{r_f^*-n-1}\},$$

the time complexity of generating and processing quartet candidates is

$$T_3 = 2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n-2} \cdot \epsilon,$$

where $\epsilon \geq 1$ and its value depends on the concrete situation, and the time complexity of the exhaustive search is $T_4 = 2^{k-h}$, where $h \leq t + |k_b \cup k_f| - |k'_b \cup k'_f|$.

It can be seen that the time complexities are affected by the key guessing strategy k'_b, k'_f. Given a distinguisher, different strategies for guessing key bits may lead to different time complexities. The GCRA, which supports any strategy, is supposed to find an optimal attack with the lowest time complexity using a holistic key guessing strategy.

2.3 The Basic Differential MITM Attack

The basic differential meet-in-the-middle (MITM) attack was first proposed by Boura *et al.* [BDD+23] at Crypto 2023, as depicted in Figure 1. Suppose the probability of the differential distinguisher over E_m is $Pr = 2^{-p}$. The differential MITM attack 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_b, compute the associated $\widetilde{P}^i$ that ensures $E_b(P) \oplus E_b(\widetilde{P}^i) = \Delta x$. There are $2^{|k_b|}$ possible $(i, \widetilde{P}^i)$. Acquire the associated ciphertexts $\widehat{C}^i = E(\widetilde{P}^i)$ and store $(\widehat{C}^i, i)$ in a hash table H.
   (b) Given $C = E(P)$, for each guess j for k_f, we can compute the associated $\widetilde{C}^j$ that ensures $E_f^{-1}(C) \oplus E_f^{-1}(\widetilde{C}^j) = \Delta y$. There are $2^{|k_f|}$ possible $(j, \widetilde{C}^j)$.
   (c) Match $\widetilde{C}^j$ with $\widehat{C}^i$ by looking up the table H. Each collision of $(\widehat{C}^i, \widetilde{C}^j)$ suggests an associated key $k_b = i, k_f = j$, that we will consider as a candidate. The number of expected collisions for one plaintext P is $2^{|k_b|+|k_f|-|k_b \cap k_f|-n}$.
2. Exhaustive search phase.
   (a) Guess the remaining key bits (if there are) and test the guess with additional pairs.

Complexities. The time complexity of this attack can be estimated as

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

where the first term corresponds to the computations done in the upper part E_b and the lower part E_f, the second one to the number of expected key candidates for $k_b \cup k_f$ and the last one to the exhaustive search.

The data complexity of the attack can be roughly estimated as $D = \min\{2^n, 2^{p+\min(|k_b|,|k_f|)}\}$, which may be improved using data structures. The memory complexity is given by $M = 2^{\min(|k_b|,|k_f|)}$, but it can be improved to $2^{\min(|k_b|,|k_f|)-|k_b \cap k_f|}$ by guessing the common key material at the beginning. In particular, [BDD+23] claimed the attack can become much more efficient when the key size of the cipher is bigger than the state size.

3 New Generic Key Recovery Attacks

Inspired by the holistic key guessing strategy of the GCRA [SZY+22], we propose counterparts for the differential attack, *i.e.*, a generic classical differential attack

(GCDA) and a generalized differential MITM attack (GDMA). Our core idea is to enhance key recovery attacks using the holistic key guessing strategy and the MITM technique. By selecting an appropriate strategy (including the type of key recovery attacks, key guessing strategy, etc.), the different terms of the time complexity can be more balanced, resulting in a lower overall time complexity. Further, upon the techniques used in GDMA, we combine the MITM technique with the rectangle attack and propose a generic rectangle MITM attack (GRMA) in return.

3.1 The Generic Classical Differential Attack

In [SZY+22], it was demonstrated for rectangle attacks that guessing some key bits in advance affects the time complexity and that any key guessing strategies should be allowed to find the best attack in terms of the time complexity. Since the rectangle attack is a variant of the differential attack, it is interesting to explore how these strategies can be applied back to the differential attack.

Suppose the differential $\Delta x \to \Delta y$ used in the attack has probability 2^{-p}. Then the data complexity for the attack is $D = 2^{p+1}$ if one right pair satisfying the differential is expected[4]. Other parameters for the key recovery are the same as introduced in Section 2.1 and Figure 1. The attacker first guesses k'_b, k'_f, a part of the involved key k_b and k_f, where $0 \le |k'_b| \le |k_b|$, $0 \le |k'_f| \le |k_f|$. Suppose there are additional r'_b and r'_f filtering bits under the guess of k'_b and k'_f. Let $k^*_b = k_b \setminus k'_b$, $k^*_f = k_f \setminus k'_f$, $r^*_b = r_b - r'_b$, $r^*_f = r_f - r'_f$.

Like the rectangle key recovery algorithm in [SZY+22], a generic differential attack can be given in Alg. 1. In this algorithm, it is assumed that $r_b - 1 \le p$, so multiple plaintext structures [BS92] are used. However, when $r_b - 1 > p$, a partial structure is enough, and this can be handled similarly. For conciseness, Alg. 1 focuses on the former case.

Complexities. The time complexity of Alg. 1 contains five parts:

- $T_0 = D$ for getting the data;
- $T_1 = 2^{|k'_b \cup k'_f|} \cdot D$ for partial encryption and decryption under the guessed key bits;
- $T_2 = 2^{|k'_b \cup k'_f|} \cdot D \cdot 2^{r_b - 1 + r_f - n - r'_b - r'_f}$ for getting the pairs that satisfy the specific filtering conditions;
- $T_3 = 2^{|k_b \cup k_f| + p - n} \cdot \epsilon = D \cdot 2^{|k_b \cup k_f| - n - 1} \cdot \epsilon$ for extracting all the $2^{|k_b \cup k_f| + p - n}$ key candidates, where ϵ depends on the concrete situation;
- $T_4 = 2^{k-h}$ for the exhaustive search, where $n - p \le h \le |k_b \cup k_f|$ when Line 10 uses the counting method while $h = n - p$ when enumerating all candidates is chosen.

As there will be $2^{|k_b \cup k_f| + p - n}$ key candidates on average, T_3 is at least $2^{|k_b \cup k_f| + p - n}$ and thus $\epsilon \ge 1$.

[4] If a bit more right pairs are needed, then D should be increased by a factor.

Algorithm 1: The generic classical differential attack (GCDA)

```
1  $S \leftarrow 2^{p-r_b+1}$ structures, each of $2^{r_b}$ messages.
2  for each possible $k'_b$ and $k'_f$, $0 \le |k'_b| \le |k_b|$, $0 \le |k'_f| \le |k_f|$, do
3      for structure $S[i], 0 \le i < |S|$ do
4          Do partial encryption and decryption for elements in $S[i]$ if $k'_b \cup k'_f \ne \emptyset$.
               // Additional $r'_b$, $r'_f$ filtering bits are obtained, respectively.
5          Store the data into a hash table indexed by the filtering bits.
6          Get $2^{2r_b-1+r_f-n-r'_b-r'_f}$ pairs having fixed differences on the filtering bits.
7          for each of such pairs do
8              Extract $2^{|k^*_b|-r^*_b}$ candidates for $k^*_b$, under which $\Delta x$ can be reached.
9              Extract $2^{|k^*_f|-r^*_f}$ candidates for $k^*_f$, under which $\Delta y$ can be reached.
10             Update the key counters or test directly.
```

The data should be stored. In addition, key counters consume memory if the counting method is used. Thus, the memory complexity is $M = \max\{2^{|k^*_b \cup k^*_f|}, D\}$ or $M = \min\{2^{r_b}, D\}$ depending on Line 10.

Remark 1. We check if the key guessing strategy makes a difference in the time complexity. First, the time complexity depends on the guessed key bits k'_b, k'_f. Additionally, we can compare two typical cases: guessing no key bits and guessing k'_b, k'_f in advance. From the data, $N = 2^{r_b+p-(n-r_f)}$ pairs can be constructed, satisfying the $n - r_f$ bits of the ciphertext difference. If the common guess-and-filter method is then used, it takes a time complexity of $N \cdot 2^{|k'_b \cup k'_f|}$ to get $N \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{-r'_b-r'_f}$ pairs which satisfy $r'_b + r'_f$ additional bit conditions. Since $N \cdot 2^{|k'_b \cup k'_f|}$ is higher than T_1 when $r_b + r_f > n$, the key guessing strategy may matter in certain cases. In Section 4.1, we will see that using different key guessing strategies results in different time complexities for attacks on 12-round AES-256.

3.2 The Generalized Differential MITM Attack

In the original differential MITM attack [BDD+23], *i.e.*, the BDMA, a fixed key guessing strategy is used. Namely, the attacker separately guesses all the k_b and k_f in the MITM phase. Similar to the GCDA, it is beneficial to allow all possible key guessing strategies for the differential MITM attack.

Note that there are $n - r_f$ filtering bits from the fixed ciphertext difference, which are available at no extra cost. However, the BDMA cannot exploit these filtering bits until two sets are matched. When $n - r_f$ is large, whether these filtering bits can be exploited early or not makes a significant difference. The attacks on 12-round AES-256 in Section 4.1 are typical examples to confirm this.

Storing pairs instead of single messages. To exploit the $n - r_f$ filtering bits earlier, we propose to store pairs instead of single messages in the MITM stages, as these filtering bits can only be used for pairs. Then, it is more efficient to perform the MITM stages for all data together in a structure rather than for each single (P, C) one by one when pairs are considered.

The detailed `GDMA`. In Alg. 2, we propose our generalized differential MITM attack (`GDMA`), which allows any possible key guessing strategies and exploits the filtering bits of ciphertext difference early. The notations showed in Alg. 2 are defined similarly. The attacker first guesses k'_b, k'_f, a part of the involved key k_b and k_f, respectively. Let $k'_\cap = k'_b \cap k'_f$. Suppose there are additional r'_b and r'_f filtering bits under the guessed key bits for the upper and lower parts. Let $k^*_b = k_b \setminus k'_b$, $k^*_f = k_f \setminus k'_f$, $r^*_b = r_b - r'_b$, $r^*_f = r_f - r'_f$. For conciseness, Alg. 2 also focuses on the case $r_b - 1 \leq p$ where multiple data structures are used.

Algorithm 2: The generalized differential MITM attack (`GDMA`)

```
1  $S \leftarrow 2^{p-r_b+1}$ structures, each of $2^{r_b}$ messages.
2  for each possible $v_\cap$ for $k'_\cap$ do
3      for structure $S[i]$, $0 \leq i < |S|$ do
4          for each possible $v_b$ for $k'_b \setminus k'_\cap$ do
5              Do partial encryption for data in $S[i]$.
6              Get $2^{2r_b-1+r_f-n-r'_b}$ pairs $(P, \tilde{P})$ satisfying $n - r_f + r'_b$ filtering bits.
7              Store the corresponding $(C, \tilde{C}, v_b)$ in a table $H$.
8          for each possible $v_f$ for $k'_f \setminus k'_\cap$ do
9              Do partial decryption for data in $S[i]$.
10             Get $2^{2r_b-1+r_f-n-r'_f}$ pairs $(C, \hat{C})$ satisfying $n - r_f + r'_f$ filtering bits.
11             for each $v_b \in H(C, \tilde{C})$ do
12                 Get $(P, \tilde{P})$ and $(v_\cap, v_b, v_f)$.
13                 Extract $2^{|k^*_b \cup k^*_f| - r^*_b - r^*_f}$ candidates for $k^*_b \cup k^*_f$ for each pair.
14                 Update the key counters or test directly.
```

Complexities. Without the pivot (P, C), both ciphertexts $(C, \tilde{C})$ are stored in Line 7. Given two random pairs from the same structure, they will match with probability 2^{-2r_b+1} as there are 2^{2r_b-1} pairs in a structure. Therefore, in Line 12, there will be $D \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$ pairs[5], the same as T_2 of the `GCDA`. 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. Similarly, the time complexity of the whole attack has five parts:

[5] $2^{|k'_\cap|} \cdot 2^{p-r_b+1} \cdot 2^{|k'_b \cup k'_f| - |k'_\cap|} \cdot 2^{2r_b-1-r'_b} \cdot 2^{2r_b-1+r_f-n-r'_f} \cdot 2^{-2r_b+1} = D \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$

- $T_0 = D$ for getting the data;
- $T_1 = (2^{|k'_b|} + 2^{|k'_f|}) \cdot D$ for partial encryption and decryption under the guessed key bits;
- $T_2 = D \cdot 2^{|k'_b|} \cdot 2^{r_b-1+r_f-n-r'_b} + D \cdot 2^{|k'_f|} \cdot 2^{r_b-1+r_f-n-r'_f} + D \cdot 2^{|k'_b \cup k'_f|} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$ for getting pairs that satisfy the specific filtering conditions;
- $T_3 = 2^{|k_b \cup k_f|+p-n} \cdot \epsilon = D \cdot 2^{|k_b \cup k_f|-n-1} \cdot \epsilon$ for extracting all the $2^{|k_b \cup k_f|+p-n}$ key candidates, where ϵ depends on the concrete situation;
- $T_4 = 2^{k-h}$ for the exhaustive search, where $n - p \leq h \leq |k_b \cup k_f|$ when Line 14 uses the counting method while $h = n - p$ when enumerating all candidates is chosen.

The data complexity is $D = 2^{p+1}$, the same as the GCDA. The memory complexity comes from the storage of the data, the pairs, and key counters if needed. To save the memory for storing the counters of $k'_\cap$, we store the whole data and count candidates for $(k_b \cup k_f) \setminus k'_\cap$. However, we can also do it the other way around, *i.e.*, store one structure each time and count candidates for $k_b \cup k_f$, if it is more beneficial. Therefore, the memory complexity is $M = \min\left\{\max\left\{D, 2^{|k_b \cup k_f| - |k'_\cap|}\right\}, \max\left\{2^{r_b}, 2^{|k_b \cup k_f|}\right\}\right\}$ if the counting method is used or $M = \max\left\{2^{r_b}, 2^{2r_b-1+r_f-n-|k'_\cap|} \cdot \min\left\{2^{|k'_b|-r'_b}, 2^{|k'_f|-r'_f}\right\}\right\}$ if the enumeration method is used.

3.3 Comparison

GDMA *versus* BDMA. Let $k'_b = k_b$ and $k'_f = k_f$. Then the GDMA turns out to be almost identical to the BDMA: the time complexities are exactly the same, while the formulas for the memory and data complexities are different. Since $D = 2^{p+1}$ is already minimal, the data complexity of GDMA is not larger than that of BDMA. The memory complexity of GDMA depends on the key guessing strategy, so it is hard to compare the memory complexity of two attacks. Hence, Alg. 2 can be seen as a generalization of BDMA if the time complexity is of the greatest concern.

GDMA *versus* GCDA. If the same key guessing strategy is used, then GDMA and GCDA share the same time complexity parts T_0, T_3 and T_4. Let us look into T_1 and T_2 which are rewritten in Table 2. Using the GDMA, the time complexity T_1 of partial encryption and decryption gets lower. For T_2, however, it depends but GDMA's T_2 is at least the one of GCDA. On the one hand, if the GDMA outperforms the GCDA, then T_1 must be dominant for the GCDA. It is the case for the differential attack on KATAN-32 in Section 4.2 when a 42-round differential is used.

On the other hand, if $r'_b \leq |k'_b|$ and $r'_f \leq |k'_b|$, GDMA will not be worse than GCDA. Usually, this is the case when a relatively large number of rounds are added around the distinguisher.

BDMA, GCDA, *and* GDMA. As there will be $2^{|k_b \cup k_f|+p-n}$ key candidates on average in any way, the following property can be obtained for differential attacks.

Table 2: Time Complexity Comparison of `GCDA` and `GDMA`

	`GCDA`	`GDMA`
T_1	$2^{\|k'_b \cup k'_f\|} \cdot D$	$(2^{\|k'_b\|} + 2^{\|k'_f\|}) \cdot D$
	-	$D \cdot 2^{\|k'_b\|} \cdot 2^{r_b-1+r_f-n-r'_b}$
T_2	-	$D \cdot 2^{\|k'_f\|} \cdot 2^{r_b-1+r_f-n-r'_f}$
	$D \cdot 2^{\|k'_b \cup k'_f\|} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$	$D \cdot 2^{\|k'_b \cup k'_f\|} \cdot 2^{r_b-1+r_f-n-r'_b-r'_f}$

Property 1. When the overall time complexity reaches $2^{|k_b \cup k_f|+p-n}$, the differential key recovery attack cannot be further improved in terms of time complexity.

If the time complexity of a certain stage exceeds this term $2^{|k_b \cup k_f|+p-n}$, there are ways to balance.

– If the time complexity T_4 of the exhaustive search is high, the counting method can be used to select the most likely candidates to test. This reduces T_4 at the cost of increasing the data by a small factor, say 4.
– The holistic key guessing strategy can balance T_1 and T_2.
– If T_3 is large due to a large ϵ, precomputed tables may help to reduce ϵ.

In a nutshell, balancing the different components of the time complexity makes the attack more efficient.

3.4 The Generic Rectangle MITM Attack

From the comparison of the `GCDA` and the `GDMA`, it is known that when a significant number of rounds is added around the distinguisher, the MITM technique is likely to be beneficial. Then, a natural question arises: can the MITM technique enhance the rectangle attack so that more rounds can be attacked in certain cases? Next, we study the combination of the MITM technique with the rectangle attack.

Suppose the boomerang distinguisher has a probability of $P^2 = 2^{-2p}$ and y structures of plaintexts are needed. Note y structures can constitute $2 \cdot \binom{y2^{r_b-1}}{2}$[6] quartets that satisfy the input difference. Then $y = 2^{n/2-r_b+1+p}$ and the data complexity $D = y2^{r_b} = 2^{n/2+p+1}$. Other notations are similar to the ones in `GDMA`. The only difference is that two differentials are used in the rectangle attack. Can we do MITM for pairs of data as in the `GDMA`? Since pairs on the upper and lower parts of the rectangle attack are constructed in different directions, we cannot perform MITM on pairs but on quartets. The generalized rectangle MITM attack is given below with this taken into account.

[6] If both (P_1, P_2) and (P_3, P_4) satisfy the input difference of the distinguisher, then we can form two quartets: (P_1, P_2, P_3, P_4) and (P_1, P_2, P_4, P_3).

Algorithm 3: The generic rectangle MITM attack (`GRMA`)

```
1  $S \leftarrow y = 2^{n/2-r_b+1+p}$ structures, each of $2^{r_b}$ messages.
2  for each possible $v_\cap$ for $k'_\cap$ do
3      for each possible $v_b$ for $k'_b \setminus k'_\cap$ do
4          Do partial encryption for data in $S$.
5          Get $D \cdot 2^{r^*_b-1}$ pairs $(P, \tilde{P})$ satisfying $r'_b$ filtering bits.
6          Generate $D^2 \cdot 2^{2r^*_b-2}$ quartets $(C_1, C_2, C_3, C_4, v_b)$.
7          Store the quartets in a table $H$.
8      for each possible $v_f$ for $k'_f \setminus k'_\cap$ do
9          Do partial decryption for data in $S$.
10         Get $D^2 \cdot 2^{r^*_f-n-1}$ pairs $\left(C, \tilde{C}\right)$ satisfying $n - r_f + r'_f$ filtering bits.
11         Generate $D^4 \cdot 2^{2r^*_f-2n-2} \cdot y^{-2}$ quartets $(C_1, C_2, C_3, C_4, v_b)$.
12         for each $v_b \in H(C_1, C_2, C_3, C_4)$ do
13             Get $(P_1, P_2, P_3, P_4)$ and $(v_\cap, v_b, v_f)$.
14             Extract $2^{|k^*_b \cup k^*_f|-2r^*_b-2r^*_f}$ candidates for $k^*_b \cup k^*_f$ for each quartet.
15             Update the key counters or test directly.
```

In Line 13 of Alg. 3, there are $2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r^*_b+2r^*_f-2n-2}$ matches[7]. In Line 14, further actions may be needed to get the other key bits. Similarly, the time complexity of the whole attack has five parts:

- $T_0 = D$ for getting the data;
- $T_1 = (2^{|k'_b|} + 2^{|k'_f|}) \cdot D$ for partial encryption and decryption under the guessed key bits;
- $T_2 = D^2 \cdot 2^{|k'_b|+2r^*_b-2} + D^4 \cdot 2^{|k'_f|+2r^*_f-2n-2} \cdot y^{-2}$ for getting quartets that satisfy the specific filtering conditions on one side, where y^{-2} is the probability of both pairs falling in the same structure;
- $T_3 = 2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n-2} \cdot \epsilon$ for extracting all the $2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n-2}$ key candidates, where $\epsilon \geq 1$ and its value depends on the concrete situation;
- $T_4 = 2^{k-h}$ for the exhaustive search, where $n - 2p \leq h \leq |k_b \cup k_f|$ when Line 15 uses the counting method while $h = n - 2p$ when enumerating all candidates is chosen.

The data, the quartets on one side, and the key counters should be stored, so the memory complexity is $M = \max\{D, \min\{D^2 \cdot 2^{|k'_b|+2r^*_b-2+2r_f-2n}, D^4 \cdot 2^{|k'_f|+2r^*_f-2n-2} \cdot y^{-2}\}, 2^{k^*_b \cup k^*_f}\}$ when the counting method is used and $M = \max\{D, \min\{D^2 \cdot 2^{|k'_b|+2r^*_b-2+2r_f-2n}, D^4 \cdot 2^{|k'_f|+2r^*_f-2n-2} \cdot y^{-2}\}\}$ when the enumeration method is used in Line 15. Like `BDMA`, the `GRMA` is usually more effective when the ratio k/n is large, which is the inherent limitation of the differential MITM attack itself.

[7] From the set of D plaintexts, there are $D^2 2^{2r_b-2}$ quartets, so a match happens with probability $D^{-2} 2^{2-2r_b}$ and $2^{|k'_b \cup k'_f|} \cdot D^2 2^{2r_b-2-2r'_b} \cdot D^4 2^{2r_f-2n-2r'_f-2} y^{-2} \cdot D^{-2} 2^{2-2r_b} = 2^{|k'_b \cup k'_f|} \cdot D^2 \cdot 2^{2r^*_b+2r^*_f-2n-2}$.

In Section 4.3, we will show with application to SKINNYe-64-256 v2 that when a large number of rounds are added to the distinguished, the GRMA can help to attack more rounds.

4 Applications

In this section, we analyze three block ciphers: AES-256, KATAN-32, and SKINNYe-64-256 using generic key recovery attacks proposed in Section 3. First, we provide improved attacks on 12/13-round AES-256 using the GCDA and GDMA in the related-key setting, which are the best attacks on AES-256 to date using only 2 related keys. Then, for KATAN-32, we use the GDMA and BDMA to extend the attack from 115 rounds to 151 rounds. These improved results demonstrate the advantage of our new key recovery attacks that enjoy the flexibility of key guessing strategies. Lastly, we apply the GRMA to SKINNYe-64-256 and extend the rectangle attack by 1 round, confirming the advantage of the GRMA in certain cases.

4.1 Application to AES-256

In this subsection, we give a brief description of AES-256 and recall the differential MITM attack on 12-round AES-256. We then propose improved attacks on 12-/13-round AES-256 using the GCDA and the GDMA.

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

- 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 $GF(2^8)$.
- AddRoundKey (AK) is an exclusive-or with the round key.

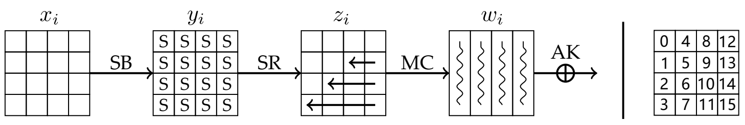

Figure 2: AES round function and the ordering of bytes

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 3. Please refer to [DR02] for more details.

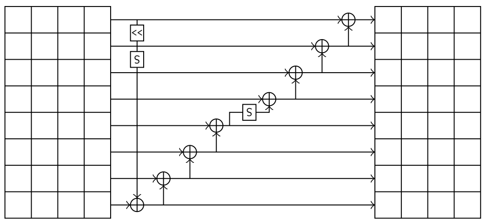

Figure 3: Key schedule of AES-256

Distinguisher. Our differential attacks below are all based on the distinguisher proposed in [BDD+23], as shown in Figure 4. The attacks require a pair of related keys. The attacker chooses two bytes a and b such that the differential transition $b \to a$ 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 $\mathtt{MC}(a, 0, 0, 0)$ to the first column of the round key k_9. Figure 4 displays the attacks on AES-256, where $u_i, 0 \le i \le 14$ are unknown key differences. The differential used for the attacks 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. If it does, the probability of the distinguisher is 2^{-86}. Moreover, the differential in the red dashed rectangle represents a 13-round attack on AES-256. The key difference propagation is depicted in Appendix A.1.

Attacks on AES-256 Reduced to 12 Rounds. In [BDD+23], to show the power of the differential MITM attack, a 12-round attack on AES-256 was proposed. To verify the input difference of the differential, it involves 15 key bytes, namely, k_b 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_f consists of $k_{11}[12, 13, 14, 15]$ and $k_{12}[0, 4, 8, 12]$, from which an extra key byte $k_{10}[12]$ can also be derived. These key bytes are highlighted as red squares in Figure 4. The remaining information of the master key contains 9 bytes. Applying the differential MITM attack presented in Section 2.3, one can have an attack of data, memory, and time[8] complexities $D = 2^{89}$, $M = 2^{89}$ and

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

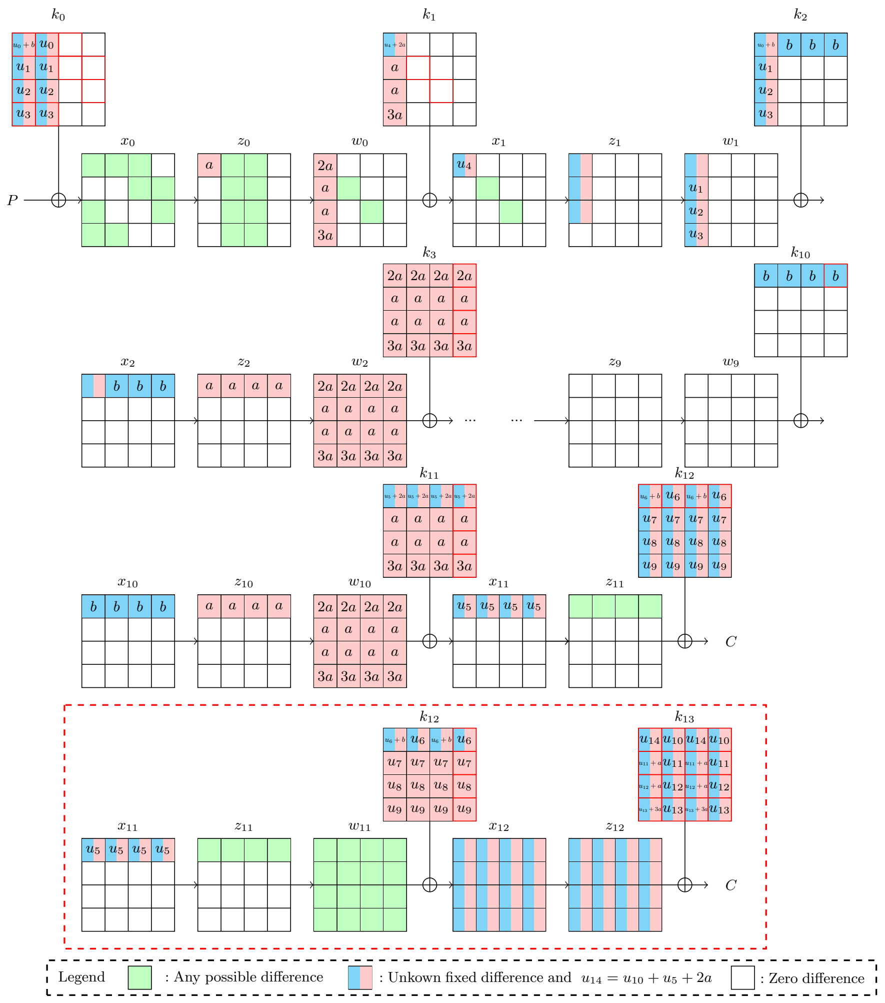

Figure 4: Differential attacks on 12-round and 13-round AES-256 where a, b are chosen and known and the bottom part of the 13-round attack is shown in the dashed square.

$T = 2^{214}$, as

$$\begin{aligned} T &= 2^p \times \left(2^{|k_{in}|} + 2^{|k_{out}|}\right) + 2^{|k_{in} \cup k_{out}| - n + p} + 2^{k-n+p} \\ &= 2^{86} \times \left(2^{120} + 2^{64}\right) + 2^{120+64-128+86} + 2^{256-128+86} \\ &= 2^{206} + 2^{150} + 2^{142} + 2^{214} \approx 2^{214}. \end{aligned} \tag{2}$$

From Equation (2), it is evident that the time for the exhaustive search dominates the overall time complexity. We could turn to the counting method to reduce the time complexity of the exhaustive search. However, the time complexity cannot be reduced below 2^{206}, primarily due to the large k_b in the upper part of the MITM phase. To further reduce the time complexity, it is worthwhile to explore different key guessing strategies.

The first `GCDA` *on 12-round* `AES`*-256.* We apply the `GCDA` to `AES`-256 and put forward the following attack using the same differential trail. The plaintext difference falls in a space of dimension $11{\times}8$ since $\Delta k_0[1] = \Delta k_0[5]$, *i.e.*, $r_b = 11{\times}8$. Since the bottom three bytes of the columns of Δk_{12} are the same, 9 bytes of the ciphertext difference are known, *i.e.*, $r_f = 7 \times 8$. We choose to guess 4 bytes of k_b and 3 bytes of k_f in advance. 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_b^*, k_f^*, *i.e.*, $k_0[0,2,3,4,7,8,9,13,14], k_1[5,10]$ and $k_{11}[12,14,15], k_{12}[0,4]$.
   (c) Do partial encryption and decryption. There are 2 additional byte filters on both sides, *i.e.*, $r_b' = r_f' = 8 \times 2$.
   (d) Construct $2^{88\times2-(2+2+9)\times8} = 2^{72}$ pairs of data. Each pair has i) fixed differences on 7 x_0 bytes at positions $1,5,6,10,11,12,15$, ii) the last three rows of Δz_{11} satisfy the pattern of Δk_{12} and iii) $\Delta x_{11}[8,12]$ are u_5.
   (e) For each pair of data, extract the other bytes of k_b by guessing $\Delta w_0[5,10]$:
      – Compute the two middle columns of Δz_0. From Δx_0 and Δz_0, derive x_0 at the 9 active bytes as well as $k_0[0,2,3,4,7,8,9,13,14]$. Compute $w_0[5,10]$.
      – 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 $k_1[5,10] = w_0[5,10] \oplus x_1[5,10]$.
   (f) For each pair of data, extract the other bytes of k_f :
      – As $\Delta k_{12}[0] = u_6 + b$, $\Delta k_{12}[4] = u_6$, and $\Delta x_{11}[0] = \Delta x_{11}[4] = u_5$, 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 $\Delta k_{11}[14,15,12] \xrightarrow{S} (u_7,u_8,u_9)$, derive $k_{11}[14,15,12]$.
   (g) Update the counters.
   (h) Select the key values with the top 2^{14} counters. Together with the guessed key bytes and all possible values for another 9 bytes outside $k_b \cup k_f$, 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 $\max\{2^{89}, 2^{16\times8}\} = 2^{128}$. The time complexity is

$$T = 2^{56} \times (2^{89} + 2^{72} + 2^{88}) + 2^{142} \approx 2^{145},$$

which is much smaller than the time complexity 2^{206} by the `BDMA`. The expected number of right pairs from the data is 4, so the right key ranks first with a high probability.

The second GCDA *on 12-round* AES*-256.* However, the memory complexity of the above attack is higher than that of the BDMA. If we guess more key bytes in advance, the memory consumption for the counters decreases. Suppose $k_0[0], k_{12}[0,4]$ and $k_0[8,13]$ are also guessed. Then more filters $\Delta z_0[0] = a, 11 \cdot \Delta z_0[8] = 13 \cdot \Delta z_0[9]$, and $\Delta x_{11}[0,4]$ are u_5, *i.e.*, r'_b, r'_f now are both 4 bytes. Consequently, the time complexity becomes

$$T = 2^{96} \times (2^{89} + 2^{40} + 2^{48}) + 2^{182} \approx 2^{185},$$

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 BDMA but a lower time complexity.

The GDMA *on 12-round* AES*-256.* Using the GDMA, let $k'_b = k_3[12,13,14,15]$, $k'_b = (k_{11}[13], k_{12}[8,12])$, and we have $k'_\cap = 0$, $r'_b = 16$, $r'_f = 16$, $M = 2^{|k_b \cup k_f|} = 2^{184}$ and $T = D(2^{32} + 2^{32-1} + 2^{24} + 2^{24-1}) + 2^{128} + 2^{144} = 2^{144}$. Now the overall time complexity is as good as the time complexity of GCDA. If in the attack only single messages are stored instead of pairs, then the $n - r_f = 72$ bit filters cannot be used early and the time complexity is $T = D(2^{32} + 2^{32-1+72} + 2^{24} + 2^{24-1}) + 2^{128} + 2^{144} = 2^{192}$. Even though this attack is more efficient than the BDMA, it is less efficient than the GDMA which stores pairs.

Remark 2. According to Property 1, when the overall time complexity of the attack on 12-round AES-256 based on the differential trail in Figure 4 reaches 2^{144}, there is no room for further improvement. That is, both the GCDA and the GDMA can lead to attacks with the optimal time complexity. However, using the BDMA, the time complexity cannot be lower than 2^{206}. This confirms that allowing flexible key guessing strategies is critical for finding the best attack.

Extending the Attack to 13 Rounds. We use the same differential in Figure 4 and add one more round to the end. In this case, k_b remains the same as in the 12-round attacks, while k_f becomes large. k_f involves 28 round key bytes: $k_{10}[12], k_{11}[12,13,14,15], k_{12}[12,13,14,15]$, $\overline{k_{12}}[0,4,8]$ and k_{13} where $\overline{k_{12}} = \mathtt{MC}^{-1}(k_{12})$. Now $k_b \cup k_f$ covers the whole key information. In addition, $r_b = 88, r_f = 128$.

The BDMA fails due to a large k_f in this case. Again, the flexible key guessing strategy shows its advantage with the following attack. Similar to the GCDA attack on 12-round AES, we also construct a structure of 2^{88} plaintexts under the two related keys, respectively.

1. Let k'_b be $k_3[14,15]$ and k'_f be $k_{10}[12]$, $k_{11}[12 \sim 15]$, $k_{12}[12 \sim 15]$ and $k_{13}[8 \sim 11]$. Guess k'_b, k'_f:
   (a) Initialize counters for all possible values for k^*_b and k^*_f.
   (b) Derive the difference for partial k_b and the whole k_f: From $\Delta k_3[14,15]$ and $k_3[14,15]$, compute u_1, u_2 so $\Delta k_0[1,2,5,6]$ is known; from $k_{10}[12]$ and b compute u_5 so $\Delta k_{11}[0,4,8,12]$ are known; from $k_{11}[12 \sim 15]$, compute $u_6 \sim u_9$ and then Δk_{12} is known; from $k_{12}[12 \sim 15]$, compute $u_{10} \sim u_{14}$ so now Δk_{13} is known. From $k_{13}[8 \sim 11]$ and $k_{12}[12 \sim 15]$ compute $k_{13}[12 \sim 15]$.

(c) Do partial encryption and decryption and then construct pairs of data satisfying 6 filtering bytes: $\Delta x_0[5,6] = 0$, *i.e.*, $r'_b = 16$, and $\Delta x_{12}[1] = \Delta x_{12}[2], \Delta x_{12}[6] = \Delta x_{12}[7], \Delta x_{12}[8] = \Delta x_{12}[11], \Delta x_{12}[12] = \Delta x_{12}[13]$ due to the MDS property of `MC`, *i.e.*, $r'_f = 32$. There will be $2^{88\times 2-6\times 8} = 2^{128}$ pairs of data.

(d) Derive other key bytes $k_{13}[0\sim 7]$,$\overline{k_{12}}[0,4]$, $k_{12}[8\sim 11]$, $k_3[12,13]$ and $k_0[0]$. Now, Δz_{11} and Δx_{12} are determined. Given the input and the output difference of the S-box, there is one solution on average. Therefore, for each pair, we can get one solution for $k_{13}[0\sim 7]$, $\overline{k_{12}}[0,4,8,12]$ and $k_0[0]$. As $\overline{k_{12}}[12]$ must match with the guessed $k_{12}[12\sim 15]$, this is a **1-byte filter** and the number of pairs becomes 2^{120}.
From the key schedule, $k_3[14] = S(k_{12}[10] \oplus k_{12}[14]) \oplus k_{11}[14]$, so we can get $k_{12}[10]$. Similarly, we can get $k_{12}[11]$. And $k_{12}[8] = k_{12}[12] \oplus k_{10}[12]$. As $\overline{k_{12}}[8]$ is also known, we can get $k_{12}[9]$. From the known bytes of k_{11}, k_{12}, we can compute $k_3[12,13]$.

(e) For each pair of data, guess $\Delta w_0[10]$ and recover the remaining key bytes of k_b, k_f.
   i. From $\Delta w_0[10]$, get the difference of the third column of z_0. For the S-boxes of that column, the input difference and the output difference are known, so we can get $k_0[8,13,2,7]$. Similarly, $k_1[10]$ can be known as $\Delta z_1[3]$ is known from u_1, u_2, u_3.
   According to the relation of key bytes, as shown in Appendix A.2 of the paper, from $k_0[0,2,7,8,13]$ and $k_1[10]$, we can get $k_{12}[0,2,5,6,7]$ and a redundant relation which acts as a **1-byte filter**.
   From $k_{12}[5,6,7]$ and $\overline{k_{12}}[4]$, compute $k_{12}[4]$. From $k_{12}[4,6]$, derive $k_0[4,14]$ which helps to compute $\Delta z_0[4,6]$. This is a **1-byte filter** due to the MDS property. Also, $\Delta z_0[5,7]$ and $\Delta w_0[5]$ can be known now, so we can derive $k_0[3,9]$ and $k_1[5]$. From these three bytes, we can derive $k_{12}[1,3]$ and one redundant relation which is also a **1-byte filter**. From $k_{12}[0\sim 3]$ and $\overline{k_{12}}[0]$, we get the last **1-byte filter**.
   Now the whole k_{12} and k_{13} are determined.
   ii. Test the key candidates exhaustively to find the right master key.

Complexities. The data complexity is $D = 2^{89}$. The memory complexity is also 2^{89}. Since $|k'_b| = 16$, $|k'_f| = 104$, we have $T_1 = D\cdot 2^{120} = 2^{209}$, $T_2 = 2^{120+128} = 2^{248}$. In Step (e), from 2^{240} pairs of data, we guess one byte more and get the other 4 filtering bytes. Therefore, 2^{216} key candidates will be suggested finally in a time complexity of 2^{248}. Since $T_4 = 2^{216}$ is not dominant, the overall time complexity is 2^{248}.

13-Round attack with an improved time complexity. In the above attack, several filters are used in the last steps. If some of these filters can be used earlier, the number of pairs may not reach 2^{248} and hopefully the time complexity is lower. We then propose an adapted attack that guesses fewer key bytes in advance and utilizes two precomputed tables so that the number of pairs is at most 2^{240}. This attack has the same data complexity. The time complexity is 2^{240} while the memory complexity is increased to 2^{120} due to the precomputed tables. The detailed attack is given in Appendix A.3.

Comparison and Discussion. On an existing differential, we analyze AES-256 reduced to 12 and 13 rounds using the GCDA and GDMA in the related-key setting. The results are summarized as follows.

– On 12-round AES-256, both the GCDA and GDMA can achieve attacks with the optimal time complexity, which is 2^{62} times lower than that of BDMA.
– Using the GCDA, the attack can be extended to 13 rounds. In this attack, $r'_b \leq |k'_b|$ and $r'_f \leq |k'_b|$, so the GDMA is not worse (not better as well in this case) under the same key guessing strategy, as explained in Section 3.3.
– Using the same differential, the BDMA cannot cover 13 rounds, because of $2^{p+|k_f|} > 2^k$ due to a large k_f. In [BDF23], the authors modified the differential to make k_f smaller, thus enabling a differential MITM attack. However, the differential probability decreased from 2^{-86} to 2^{-126}.

All these results show that allowing flexible key guessing strategies is critical for mounting key recovery attacks efficiently. Choosing between the counting method and the enumerating method for the exhaustive search also impacts the overall efficiency. The attacks on AES-256 illustrate that by combining these methods, we can balance the time complexities of the attack steps to get more efficient attacks.

However, the time complexity of the 13-round attack does not reach $2^{|k_b \cup k_f|+p-n}$. The major reason lies in the nonlinear key schedule. k_b and k_f share some common information, but it is difficult to utilize them early on due to their complex nonlinear relationship. The 13-round attack with a lower time complexity highlights the challenge of effectively leveraging the shared key information.

4.2 Application to KATAN-32

For most block ciphers, a few rounds can be added around a differential in the key recovery attack. However, the block cipher KATAN-32 is an exception. Since it updates only 2 bits in each round, a relatively large number of rounds can be added. In attacks on KATAN-32, we can observe how the BDMA, GCDA, and GDMA perform when the number of rounds increases, and more importantly, whether there is any discernible trend. Additionally, any improvement in its differential attack is of particular interest. We start by recalling the description of the KATAN-32.

Description of KATAN. The KATAN family is composed of three variants with 32-, 48-, and 64-bit block sizes denoted as KATAN-n, $n = 32, 48, 64$, respectively. Here, we briefly revisit the KATAN-32, which is analyzed in the next. The KATAN-32 iterates 254 rounds using two non-linear feedback shift registers (NLFSR) to store and update the plaintext.

Key schedule. The 80-bit master key $K = (k_0, k_1, \cdots, k_{78}, k_{79})$ uses the linear feedback register to generate the new round keys.

$$k_{i+80} = k_i \oplus k_{i+19} \oplus k_{i+30} \oplus k_{i+67}, \ \ 0 \leq i \leq 427. \tag{3}$$

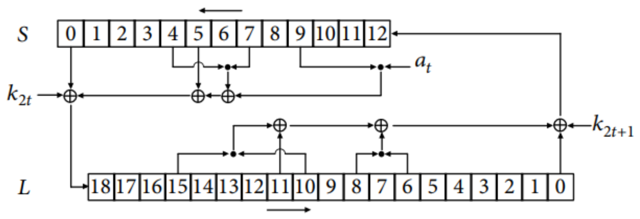

Figure 5: The round function of KATAN-32.

Round function. A 32-bit plaintext $X = (x_0, x_1, \cdots, x_{30}, x_{31})$ is divided into two parts with 13 and 19 bits, respectively. At round t, the two parts are denoted by $S_t = (s_t, s_t + 1, \cdots, s_{t+11}, s_{t+12})$ and $L_t = (l_t, l_t + 1, \cdots, l_{t+17}, l_{t+18})$. When $t = 0$, the plaintext is loaded as $s_i = x_i, 0 \leq i \leq 12$ and $l_i = x_{13+i}, 0 \leq i \leq 18$. When $0 \leq t \leq 253$, the round function depicted in Figure 5 is defined as follows:

$$\begin{aligned} s_{t+13} &= l_t \oplus l_{t+11} \oplus l_{t+6} \cdot l_{t+8} \oplus l_{t+10} \cdot l_{t+15} \oplus k_{2t+1}, \\ l_{t+19} &= s_t \oplus s_{t+5} \oplus s_{t+4} \cdot s_{t+7} \oplus s_{t+9} a_t \oplus k_{2t}, \end{aligned} \tag{4}$$

where a_t is a round constant updated by the relation equation $a_t = a_t - 3 \oplus a_{t-5} \oplus a_{t-7} \oplus a_{t-8}, (t \geq 8)$ with the initial value $(a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (1, 1, 1, 1, 1, 1, 1, 0)$. According to the Equation 4, we can get the expression of l_t, s_t in the decryption direction:

$$\begin{aligned} l_t &= s_{t+13} \oplus l_{t+11} \oplus l_{t+6} l_{t+8} \oplus l_{t+10} l_{t+15} \oplus k_{2t+1}, \\ s_t &= l_{t+19} \oplus s_{t+5} \oplus s_{t+4} s_{t+7} \oplus s_{t+9} a_t \oplus k_{2t}. \end{aligned} \tag{5}$$

Distinguisher. Our attacks are based on the 42-round and the 91-round distinguishers proposed in [JRS22] and [AL13] with probability 2^{-12} and $2^{-31.98}$:

$$\begin{aligned} \Delta_{in}^{42} &= 0x08020040, \ \Delta_{out}^{42} = 0x00200420, \\ \Delta_{in}^{91} &= 0x1006a880, \ \Delta_{out}^{91} = 0x00400000. \end{aligned}$$

When the attacker conducts a key attack on KATAN, it is very time-consuming to determine the best key guessing strategy manually. It becomes even harder when the added rounds increase. Therefore, we build a MILP model to find the best key guessing strategy automatically. This model follows the same modules as [SZY+22], and its detailed description is postponed to Appendix B.1. Given a differential, the number of rounds before the differential, the number of rounds after the differential, and the type of attack (*i.e.*, BDMA, GCDA, or GDMA), the model outputs the minimal time complexity of the attack and the parameters that lead to the attack.

Comparison and Discussion. Using the key recovery model, we can change the objective function according to the key recovery algorithm to find the best-attacking parameters such that the time complexity of the attack is optimal. Based on the key recovery model and the 42-round distinguisher, we apply the GCDA, GDMA, and BDMA to KATAN-32 from 94 to 137 rounds and compare their time complexities. The results are illustrated in Figure 6 and analyzed as follows.

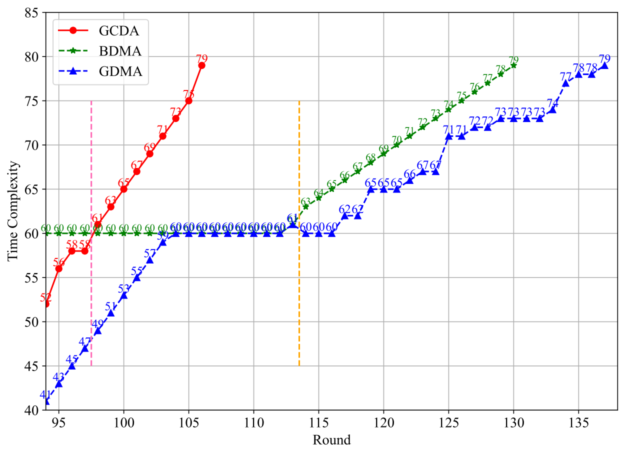

Figure 6: The time complexity of three attacks.

- From attacks on KATAN-32 reduced to 94 to 97 rounds before the pink dashed line as shown in Figure 6, both GCDA and GDMA have a better performance than the BDMA. The last part 2^{k-n+p} of the BDMA's time complexity is dominated, while the GCDA and GDMA can use the counting method mentioned in Section 3.3 to reduce the exhaustive search complexity.
- Between the pink and orange dashed lines, T_1 becomes a large one in the GCDA's time complexity as the key guessed increases. So, the GCDA is worse than the GDMA and BDMA. When $|k_b \cup k_f|$ is not the full key space, the GDMA has a lower time complexity than the BDMA. As the number of rounds increases, $|k_b \cup k_f| = 80$, which reaches the full key space. The dominant terms of the BDMA and GDMA are the same.

– After the orange dashed line, the first term $(2^{|k_b|} + 2^{|k_f|}) \cdot 2^p$ of BDMA and $T_1 = (2^{|k'_b|} + 2^{|k'_f|}) \cdot D$ of the GDMA turn to dominant terms. The GDMA has a lower time complexity than the BDMA. Because the key guessing strategy of the BDMA is fixed and the GDMA allows the attacker to guess a part of key information.
– The GDMA always performs better than the GCDA in every round of attacks on KATAN-32. As explained in Section 3.3, T_1 of the GCDA is a dominant one. The partial encryption and decryption time complexity T_1 of the GDMA is lower than that of GCDA.

Attacks on KATAN-32 Reduced to 151 Rounds. We apply the BDMA and GDMA to attack 151-round KATAN-32 based on the 91-round distinguisher with 29-round E_b and 31-round E_f, as shown in Table 3. In Table 3, ? denotes a bit with the unknown difference, 0 and 1 represent the specific difference value.

BDMA. The parameters are $p = 31.98$, $n = 32$, $|k_b| = 41$, $|k_f| = 38$. The key information is listed as follows:

$$\begin{aligned}
k_b &= k_0, k_1, \cdots, k_{36}, k_{37}, k_{40}, k_{44}, k_{45},\\
k_f &= k_{252}, k_{260}, k_{262}, k_{264}, k_{266}, k_{268}, k_{269}, k_{270}, k_{272}, k_{273}, \cdots, k_{300}, k_{301}.
\end{aligned}$$

The time complexity is

$$\begin{aligned}
T &= 2^{31.98+41} + 2^{31.98+38} + 2^{41+38-0.02} + 2^{80-32+31.98}\\
&= 2^{72.98} + 2^{69.98} + 2^{78.98} + 2^{79.98} \approx 2^{79.98}
\end{aligned}$$

with data and memory complexities $D = 2^{32}, M = 2^{38}$.

GDMA. The best guessing parameters are $|k'_b| = 33$, $|k'_f| = 29$, $r'_b = 25$, $r'_f = 22$ and $r_b = r_f = 32$. Namely, guessing 33-bit k'_b and 29-bit k'_f obtains 25 and 22 filters, respectively. The filtering bits in the backward and forward extended rounds are marked in blue and red in Table 3. The subkey bits guessed are:

$$\begin{aligned}
k'_b &= k_0, k_1, \cdots, k_{24}, k_{25}, k_{27}, k_{28}, k_{29}, k_{30}, k_{32}, k_{33}, k_{36},\\
k'_f &= k_{264}, k_{268}, k_{272}, k_{274}, k_{275}, k_{276}, k_{278}, k_{279}, k_{280}, k_{282}, k_{283}, \cdots, k_{300}, k_{301}.
\end{aligned}$$

The data complexity and memory complexity are $D = 2^{32}, M = 2^{70}$. The time complexity is

$$\begin{aligned}
T = &\, 2^{32+33} + 2^{32+33+32-25-1} + 2^{32+29} + 2^{32+29+32-22-1}\\
&+ 2^{32+33+29+32-25-22-1} + 2^{79-0.02} + 2^{80+31.98-32}\\
= &\, 2^{65} + 2^{71} + 2^{61} + 2^{70} + 2^{78} + 2^{78.98} + 2^{79.98} \approx 2^{79.98}.
\end{aligned}$$

Although the GDMA and BDMA have a slight advantage over traversing all keys, our attacks primarily demonstrate that using new generic differential attacks can significantly increase the number of rounds attacked, from 115 to 151. In the attack, the time complexity of the exhaustive search is dominant. Since the differential has a probability slightly larger than 2^{-n}, there is no opportunity to trade data for time by using the counting method. Consequently, `BDMA` and `GDMA` share the same overall time complexity.

In addition, we present alternative 151-round attacks on `KATAN`-32 by extending 33 before and 27 rounds after the differential, respectively. More details are provided in Appendix B.2. In this setting, the `GDMA` still recovers the key with a time complexity of $2^{79.98}$, while the `BDMA` fails. This outcome underscores the importance of flexibly guessing key information for successful key recovery attacks, highlighting the greater applicability of the `GDMA`.

Table 3: The $151(29+91+31)$-round attack on the `KATAN`-32. For round t, Δ_t is the t-th round difference, $0 \leq t \leq 151$. The blue and red denote the filters by guessing a part of key information k'_b and k'_f.

Δ_0	???? ???? ???? ???? ???? ???? ???? ????	Δ_{out}	0000 0000 0100 0000 0000 0000 0000 0000
Δ_1	???? ???? ???? ???? ???? ???? ???? ????	Δ_{121}	0000 0000 1000 0000 0000 0000 0000 0001
Δ_2	???? ???? ???? ???? ???? ???? ???? ????	Δ_{122}	0000 0001 0000 0000 0000 0000 0000 0010
Δ_3	???? ???? ???? ???? ???? ???? ???? ????	Δ_{123}	0000 0010 0000 0000 0000 0000 0000 010?
Δ_4	???? ???? ???? ???? ???? ???? ???? ????	Δ_{124}	0000 0100 0000 0000 0000 0000 0000 10?0
Δ_5	???? ???? ???? ???? ???? ???? ???? ????	Δ_{125}	0000 1000 0000 ?000 0000 0000 0001 0?01
Δ_6	???? ???? ???? ???? ???? ???? ???? ????	Δ_{126}	0001 0000 000? 0000 0000 0000 0010 ?01?
Δ_7	???? ???? ???? ???? ???? ???? ???? ????	Δ_{127}	0010 0000 00?0 ?000 0000 0000 010? 01?0
Δ_8	???? ???? ???? ???? ???? ???? ???? ????	Δ_{128}	0100 0000 0?0? 0000 0000 0000 10?0 1?00
Δ_9	???? ???? ???? ???? ???? ???? ???? ???0	Δ_{129}	1000 0000 ?0?0 ?000 0000 0001 0?01 ?000
Δ_{10}	???? ???? ???? 1??? ???? ???? ???? ??0?	Δ_{130}	0000 000? 0?0? ?000 0000 0010 ?01? 0001
Δ_{11}	???? ???? ???1 ???? ???? ???? ???? ?0?1	Δ_{131}	0000 00?0 ?0?? ?000 0000 010? 01?0 001?
Δ_{12}	???? ???? ??1? ???? ???? ???? ???? 0?11	Δ_{132}	0000 0?0? 0??? ?000 0000 10?0 1?00 01?0
Δ_{13}	???? ???? ?1?? ???? ???? ???? ???0 ?110	Δ_{133}	0000 ?0?0 ???? 1000 0001 0?01 ?000 1?0?
Δ_{14}	???? ???? 1??? 0??? ???? ???? ??0? 1101	Δ_{134}	000? 0?0? ???1 ?000 0010 ?01? 0001 ?0??
Δ_{15}	???? ???1 ???0 1??? ???? ???? ?0?1 1010	Δ_{135}	00?0 ?0?? ??1? ?000 010? 01?0 001? 0???
Δ_{16}	???? ??1? ??01 ???? ???? ???? 0?11 0101	Δ_{136}	0?0? 0??? ?1?? ?000 10?0 1?00 01?0 ????
Δ_{17}	???? ?1?? ?01? 0??? ???? ???0 ?110 1010	Δ_{137}	?0?0 ???? 1??? ?001 0?01 ?000 1?0? ????
Δ_{18}	???? 1??? 01?0 0??? ???? ??0? 1101 0101	Δ_{138}	0?0? ???1 ???? ?010 ?01? 0001 ?0?? ????
Δ_{19}	???1 ???0 1?00 0??? ???? ?0?1 1010 1010	Δ_{139}	?0?? ??1? ???? ?10? 01?0 001? 0??? ????
Δ_{20}	??1? ??01 ?000 1??? ???? 0?11 0101 0100	Δ_{140}	0??? ?1?? ???? ?0?0 1?00 01?0 ???? ????
Δ_{21}	?1?? ?01? 0001 0??? ???0 ?110 1010 1000	Δ_{141}	???? 1??? ???? ??01 ?000 1?0? ???? ????
Δ_{22}	1??? 01?0 0010 0??? ??0? 1101 0101 0001	Δ_{142}	???1 ???? ???? ?01? 0001 ?0?? ???? ????
Δ_{23}	???0 1?00 0100 0??? ?0?1 1010 1010 0010	Δ_{143}	??1? ???? ???? ?1?0 001? 0??? ???? ????
Δ_{24}	??01 ?000 1000 0??? 0?11 0101 0100 0100	Δ_{144}	?1?? ???? ???? ??00 01?0 ???? ???? ????
Δ_{25}	?01? 0001 0000 0??0 ?110 1010 1000 1000	Δ_{145}	1??? ???? ???? ?000 1?0? ???? ???? ????
Δ_{26}	01?0 0010 0000 0?0? 1101 0101 0001 0000	Δ_{146}	???? ???? ???? ?001 ?0?? ???? ???? ????
Δ_{27}	1?00 0100 0000 00?1 1010 1010 0010 0000	Δ_{147}	???? ???? ???? ?01? 0??? ???? ???? ????
Δ_{28}	?000 1000 0000 0?11 0101 0100 0100 0000	Δ_{148}	???? ???? ???? ?1?0 ???? ???? ???? ????
Δ_{in}	0001 0000 0000 0110 1010 1000 1000 0000	Δ_{149}	???? ???? ???? ??0? ???? ???? ???? ????
..	....	Δ_{150}	???? ???? ???? ?0?? ???? ???? ???? ????
..	....	Δ_{151}	???? ???? ???? ???? ???? ???? ???? ????

4.3 Application on SKINNYe-64-256 v2

In this subsection, we apply our GRMA to SKINNYe-64-256 to obtain a 38-round rectangle attack.

Description of SKINNYe-64-256 v2. SKINNYe-64-256 v2 [NSS20a] is one of the variants of SKINNY [BJK+16]. Similar to SKINNY, SKINNYe-64-256 v2 employs the TWEAKEY framework and the STK construction [JNP14], with modifications to the f function in STK. SKINNY supports block sizes $n \in \{64, 128\}$, and for $n = 64$, the tweakey sizes $tk \in \{64, 128, 192\}$. To support TI-friendly AE modes, Naito *et al.* extended SKINNY-64 to create SKINNYe-64-256, which features a 256-bit tweakey and a similar tweakey schedule [NSS20b]. Due to security concerns raised by Thomas Peyrin, an updated version, SKINNYe-64-256 v2, was proposed in 2020 [NSS20a].

SKINNY, SKINNYe-64-256, and SKINNYe-64-256 v2 share the same round function, applying five transformations:

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 the matrix M whose branch number is only 2.

Tweakey schedule of SKINNYe-*64-256 v2.* The 256-bit tweakey state is viewed as 4×4 square arrays of nibbles, denoted as (TK^1, TK^2, TK^3, TK^4). The tweakey arrays in round r ($r \geq 0$) are represented as TK_r^1, TK_r^2, TK_r^3, and TK_r^4, where $TK_0^m = TK^m$ ($1 \leq m \leq 4$). For $r \geq 1$, TK_r^m is generated in two steps:

- First, apply the permutation $P = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ to each nibble of all tweakey arrays: $TK_r^{m,i} \leftarrow TK_{(r-1)}^{m,P[i]}$, where $1 \leq m \leq 4, 0 \leq i \leq 15$ and $r \geq 1$.
- Then, apply $LFSR_r$ to update each nibble of the first and second rows of TK_r^m for $2 \leq m \leq 4$. For the details, please refer to [NSS20a].

Distinguisher of SKINNYe-64-256 v2. At Asiacrypt 2022, Qin *et al.* [QDW+22] proposed a 26-round related-key rectangle distinguisher of SKINNYe-64-256 v2 with a probability of $2^{-57.6}$. The distinguisher is detailed in Appendix C. By modifying this distinguisher, we obtain a new version that is more suitable for GRMA. Specifically, we allow $b \to 3$ in the SC operation in round 5 with a probability of 2^{-2}. Therefore, the probability of this modified distinguisher is $P^2 = 2^{-57.6-4} = 2^{-61.6}$.

We add 6 rounds forward and backward to attack the 38-round SKINNYe-64-256 v2, as shown in Figure 7. The corresponding parameters are: $r_b = r_f = 16 \times 4 = 64$, $|k_b| = (8 \times 3 + 6 + 2) \times 4 = 128$, and $|k_f| = (2 + 6 + 8 \times 3) \times 4 = 128$.

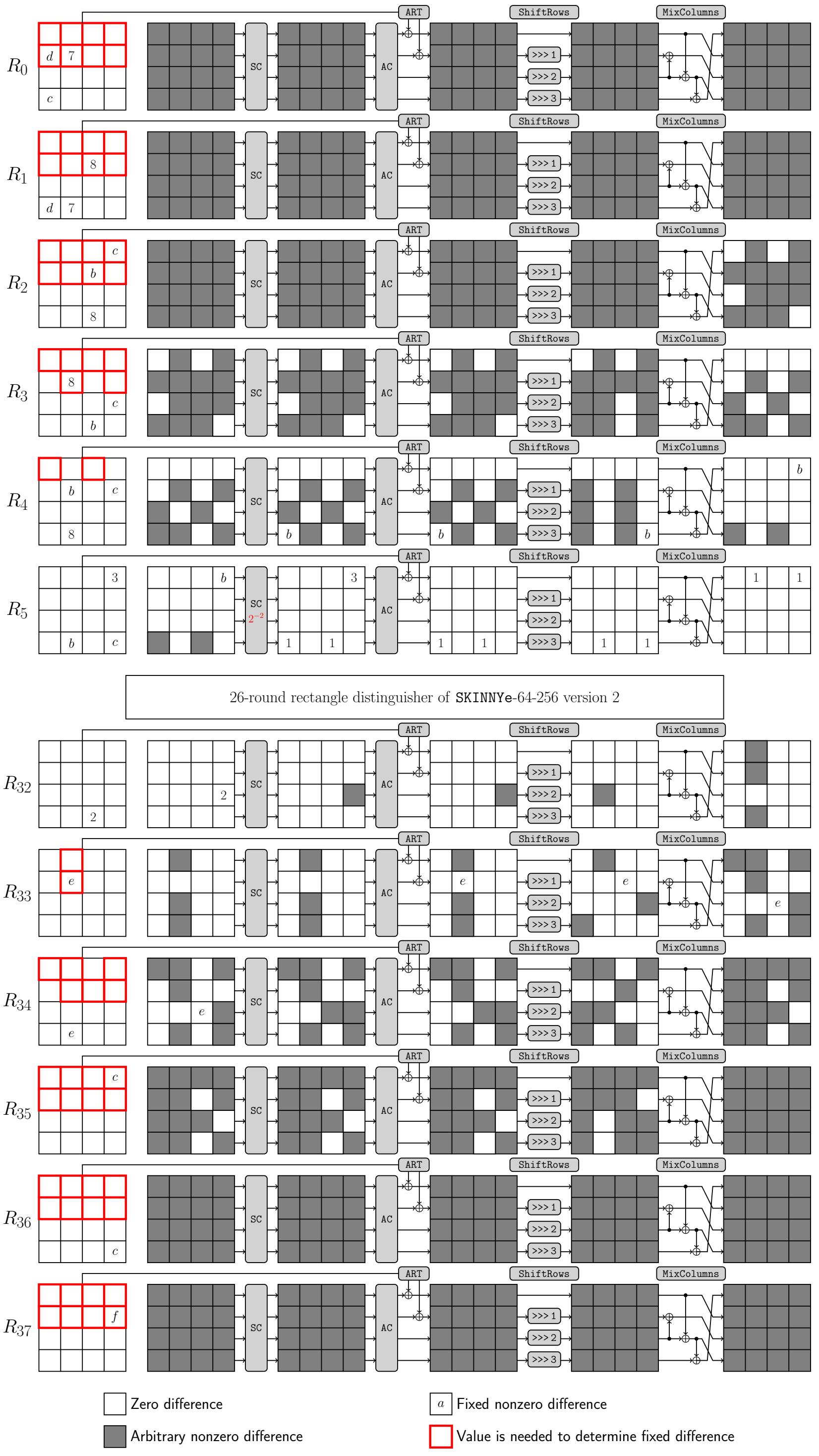

Figure 7: Rectangle meet-in-the-middle attack on 38-round SKINNYe-64-256 v2

Parameters and Complexities. We apply GRMA to the above distinguisher. The attack is illustrated in Figure 7, and the parameters of this attack are $|k'_b| = |k_b| = 128$ and $|k'_f| = |k_f| = 128$. Therefore, all the involved keys are guessed, and all the unknown differentials in E_b and E_f can be determined.

Given a boomerang distinguisher with probability P^2, the number of quartets satisfying the input difference of the distinguisher should be at least $P^{-2}2^n$. For the sake of clarity, we denote $2^{-2p} = P^2$, then $p = 30.8$. In our attack, $r_b = n$, so we use partial structures of plaintexts for data collection. By collecting D plaintext-ciphertext pairs, $(D^2)^2 \times 2^{-2n}$ quartets will satisfy the input difference. Therefore, $D = 2^{3n/4} \cdot 2^{p/2} = 2^{63.4}$.

Under the related-key setting, the calculation of time complexity is slightly different from that in the single-key setting introduced in Section 3.4. Additionally, the data of our attack will always belong to the same partial structure, so y^{-2} is omitted in T_2. We use D_R to represent the data required under the related-key setting. The complexities of our new attack are as follows:

- The data complexity is $D_R = 4 \cdot D = 2^{3n/4+p/2+2} = 2^{65.4}$.
- The memory complexity is $M = D^2 \cdot 2^{|k_b|+2r_f-2n} = 2^{254.8}$.
- The time complexity:

$$\begin{aligned}
T_0 &= 4 \cdot D_R,\\
T_1 &= (2^{|k_b|} + 2^{|k_f|}) \cdot D_R = 2^{194.4},\\
T_2 &= D^2 \cdot 2^{|k_b|} + D^4 \cdot 2^{|k_f|-2n} = 2^{255.32},\\
T_3 &= 2^{|k_b \cup k_f|} \cdot D^2 \cdot 2^{-2n} = 2^{254.8},\\
T_4 &= 2^{k-h} = 2^{256-(n-2p)} = 2^{253.6}.
\end{aligned}$$

The time complexity of our attack is $\frac{1}{38} \times (T_0 + T_1 + T_2 + T_3 + T_4) \approx 2^{251.07}$ 38-round encryption.

Comparison and Discussion. In our rectangle MITM attack, we modify one cell in the input difference. Compared with the original rectangle key recovery, the probability of the distinguisher is reduced by 2^{-4}, and the k_b has two fewer cells in E_b.

According to the GRMA algorithm, MITM attacks are more effective in situations where subkey bits are balanced on both sides. Therefore, the modified distinguisher is more suitable for the GRMA. By adding 6 rounds on both sides of the distinguisher, the number of the subkey bits is: $|k_b| = |k_f| = 128$. Using the GRMA, we achieve a 38-round rectangle attack on SKINNYe-64-256, which is one more round than the previous attack. It is worth noting that for this distinguisher, the GCRA could not provide an effective 38-round rectangle attack.

5 Conclusion

In this paper, we revisit the generic rectangle key recovery attack [SZY+22] and the differential MITM attack recently proposed in [BDD+23]. Inspired by the holistic strategy and the MITM technique, we propose three new generic key recovery attacks: the classical differential attack, the differential MITM attack, and the rectangle MITM attack, respectively denoted as `GCDA`, `GDMA`, and `GRMA`. By selecting an appropriate strategy (including the type of key recovery attacks, key guessing strategy, etc.), the different terms of the time complexity can be more balanced, resulting in a lower overall time complexity. For application, we apply our new key recovery algorithms to the `AES`-256, `KATAN`-32, and `SKINNYe`-64-256 block ciphers. For `AES`-256, we provide better results on a 12-round differential attack using the `GCDA` and the `GDMA`, and introduce a new 13-round differential attack with reduced data and time complexities. We apply the `GDMA`, `GCDA`, and `BDMA` to `KATAN`-32, building a dedicated model for key guessing with MILP. With the aid of MILP-based tools, we improve the differential attacks on `KATAN`-32 from 115 rounds to 151 rounds. For `SKINNYe`-64-256, we achieve the first 38-round rectangle attack using the `GRMA`, which extends the previous attack by 1 round.

Acknowledgement. The authors would like to thank anonymous reviewers for their helpful comments and suggestions. The work of this paper was supported by the National Natural Science Foundation of China (Grants 62132008, 62372213, 62202460, 62332007 and U22B2028). Jian Weng is also supported by the Major Program of Guangdong Basic and Applied Research Project under Grant No. 2019B030302008, Science and Technology Major Project of Tibetan Autonomous Region of China under Grant No. XZ202201ZD0006G, Guangdong Provincial Science and Technology Project under Grant No. 2021A0505030033, National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education.

References

AFK$^+$08. Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of Salsa, ChaCha, and Rumba. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 470–488. Springer, 2008.

AKM$^+$24. Zahra Ahmadian, Akram Khalesi, Dounia M'foukh, Hossein Moghimi, and María Naya-Plasencia. Improved differential meet-in-the-middle cryptanalysis. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 280–309. Springer, 2024.

AL13. Martin R Albrecht and Gregor Leander. An all-in-one approach to differential cryptanalysis for small block ciphers. In *Selected Areas in Cryptography:*

19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers 19, pages 1–15. Springer, 2013.

BCF+21. Marek Broll, Federico Canale, Antonio Flórez-Gutiérrez, Gregor Leander, and María Naya-Plasencia. Generic framework for key-guessing improvements. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 453–483. Springer, 2021.

BDD+23. Christina Boura, Nicolas David, Patrick Derbez, Gregor Leander, and María Naya-Plasencia. Differential meet-in-the-middle cryptanalysis. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 240–272. Springer, 2023.

BDD+24. Christina Boura, Nicolas David, Patrick Derbez, Rachelle Heim Boissier, and María Naya-Plasencia. A generic algorithm for efficient key recovery in differential attacks - and its associated tool. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 217–248. Springer, 2024.

BDF23. Christina Boura, Patrick Derbez, and Margot Funk. Related-key differential analysis of the aes. *IACR Transactions on Symmetric Cryptology*, 2023(4):215–243, 2023.

BDK01. Eli Biham, Orr Dunkelman, and Nathan Keller. The rectangle attack—rectangling the Serpent. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 340–357. Springer, 2001.

BDK02. Eli Biham, Orr Dunkelman, and Nathan Keller. New results on boomerang and rectangle attacks. In *International Workshop on Fast Software Encryption*, pages 1–16. Springer, 2002.

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 *Annual International Cryptology Conference*, pages 123–153. Springer, 2016.

BS90. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

BS91. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.

BS92. Eli Biham and Adi Shamir. Differential cryptanalysis of the full 16-round DES. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California,* *USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496. Springer, 1992.

DQSW21. Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. *IACR Cryptol. ePrint Arch.*, page 856, 2021.

DR02. Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

JNP14. Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2014.

JRS22. Amit Jana, Mostafizar Rahman, and Dhiman Saha. Deepand: In-depth modeling of correlated and gates for nlfsr-based lightweight block ciphers. Cryptology ePrint Archive, Paper 2022/1123, 2022. https://eprint.iacr.org/2022/1123.

KMN10. Simon Knellwolf, Willi Meier, and María Naya-Plasencia. Conditional differential cryptanalysis of NLFSR-based cryptosystems. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010.

LKKD08. Jiqiang Lu, Jongsung Kim, Nathan Keller, and Orr Dunkelman. Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1. In Tal Malkin, editor, *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*, volume 4964 of *Lecture Notes in Computer Science*, pages 370–386. Springer, 2008.

Mat94. Mitsuru Matsui. On correlation between the order of S-boxes and the strength of DES. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1994.

MP13. Nicky Mouha and Bart Preneel. A proof that the ARX cipher Salsa20 is secure against differential cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 328, 2013.

MWGP11. Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

NSS20a. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight authenticated encryption mode suitable for threshold implementation. Cryptology ePrint Archive, Paper 2020/542, 2020. https://eprint.iacr.org/2020/542.

NSS20b. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight authenticated encryption mode suitable for threshold implementation. In *Annual*

International Conference on the Theory and Applications of Cryptographic Techniques, pages 705–735. Springer, 2020.

QDW$^+$22. Lingyue Qin, Xiaoyang Dong, Anyu Wang, Jialiang Hua, and Xiaoyun Wang. Mind the tweakey schedule: cryptanalysis on skinnye-64-256. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 287–317. Springer, 2022.

SHW$^+$14. Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

SWW21. Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the SAT method. *IACR Trans. Symmetric Cryptol.*, 2021(1):269–315, 2021.

SYC$^+$24. Ling Song, Qianqian Yang, Yincen Chen, Lei Hu, and Jian Weng. Probabilistic extensions: a one-step framework for finding rectangle attacks and beyond. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 339–367. Springer, 2024.

SZY$^+$22. Ling Song, Nana Zhang, Qianqian Yang, Danping Shi, Jiahao Zhao, Lei Hu, and Jian Weng. Optimizing rectangle attacks: A unified and generic framework for key recovery. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 410–440. Springer, 2022.

WWJZ18. Ning Wang, Xiaoyun Wang, Keting Jia, and Jingyuan Zhao. Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. *Sci. China Inf. Sci.*, 61(9):098103:1–098103:3, 2018.

YSZ$^+$24. Qianqian Yang, Ling Song, Nana Zhang, Danping Shi, Libo Wang, Jiahao Zhao, Lei Hu, and Jian Weng. Optimizing rectangle and boomerang attacks: A unified and generic framework for key recovery. *Journal of Cryptology*, 37(2):1–62, 2024.

ZDM$^+$20. Boxin Zhao, Xiaoyang Dong, Willi Meier, Keting Jia, and Gaoli Wang. Generalized related-key rectangle attacks on block ciphers with linear key schedule: applications to SKINNY and GIFT. *Designs, Codes and Cryptography*, 88(6):1103–1126, 2020.

Supplementary Materials

A More Details of the Key Schedule of AES-256

A.1 Differences of Round Key Bytes

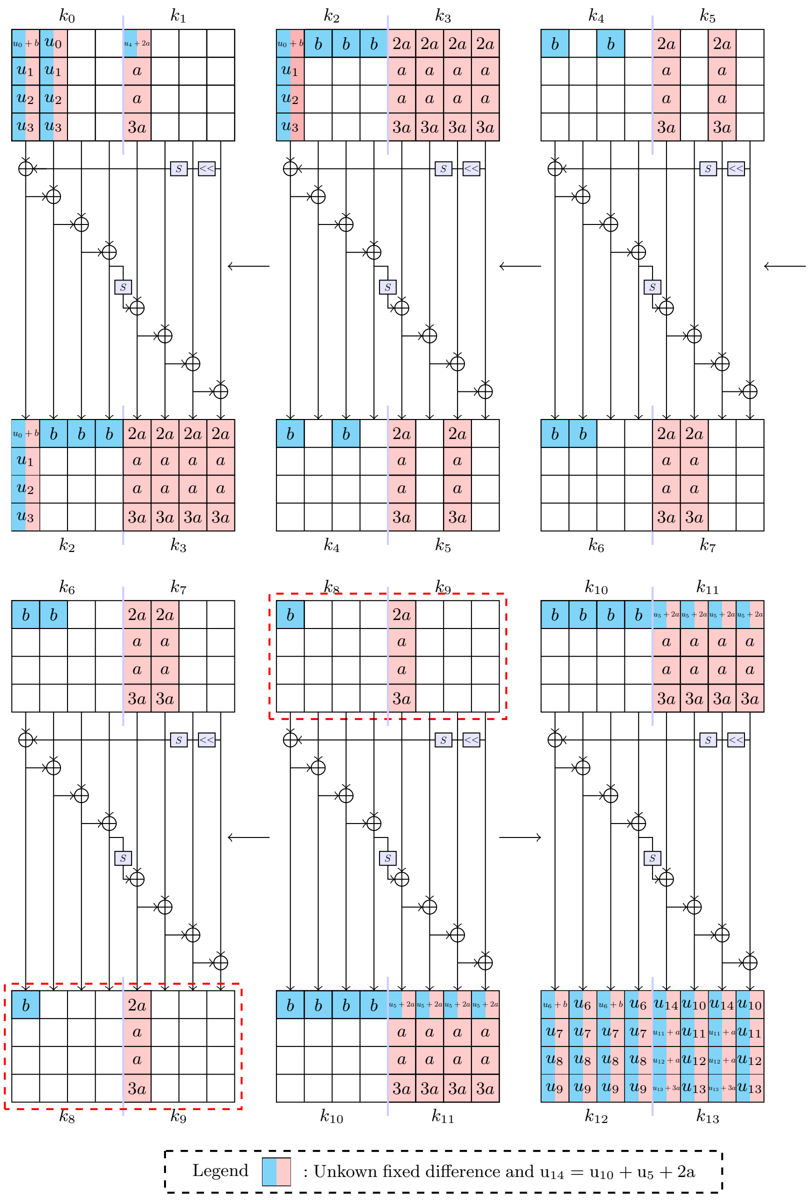

Figure 8: The key difference propagation about AES-256.

A.2 Relationships of Subkey Bytes in the 13-Round Attack

In the 13-round attack on AES-256, we use the relationship of subkey bytes listed as follows to derive some key bytes from known ones. The blue bytes are knonn or guessed, the red bytes are obtained by the key schedule and each $\blacktriangleleft$ marks a redundant relation, *i.e.*, a 1-byte filter. For simplicity, the use of round constants is omitted.

$$
\begin{aligned}
k_3[12] &= S(k_{10}[12]) \oplus k_{11}[12];\\
k_3[14] &= k_{11}[13] \oplus S(k_{12}[14] \oplus k_{12}[10]);\\
k_3[15] &= k_{11}[15] \oplus S(k_{12}[15] \oplus k_{12}[11]);\\
\overline{k_{12}}[8] &= MC^{-1}(k_{12}[8], k_{12}[9], k_{12}[10], k_{12}[11]);\\
k_3[13] &= k_{11}[13] \oplus S(k_{12}[13] \oplus k_{12}[9]);\\
k_0[8] &= k_{12}[0] \oplus k_{12}[8] \oplus S(k_{12}[13] \oplus k_{12}[9]) \oplus S(k_{11}[13] \oplus k_{12}[13] \oplus k_{12}[9]);\\
k_0[13] &= k_{12}[5] \oplus k_{12}[13] \oplus S(k_{13}[14] \oplus k_{13}[10] \oplus k_{13}[6] \oplus k_{13}[2]);\\
k_0[7] &= k_{12}[7] \oplus S(k_{13}[12] \oplus k_{13}[8]) \oplus S(k_{13}[12] \oplus k_{13}[8] \oplus k_{13}[4] \oplus k_{13}[0])\\
&\quad \oplus S(S(k_{10}[12]) \oplus k_{11}[12]);\\
k_0[0] &= k_{12}[0] \oplus S(k_{13}[13] \oplus k_{13}[9]) \oplus S(k_{13}[13] \oplus k_{13}[5]) \oplus S(k_{13}[13] \oplus k_{13}[9]\\
&\quad \oplus k_{13}[5] \oplus k_{13}[1]) \oplus S(k_{13}[13] \oplus k_{13}[9] \oplus S(k_{12}[13] \oplus k_{12}[9])) \oplus S(k_{13}[13]\\
&\quad \oplus k_{13}[5] \oplus S(k_{12}[13] \oplus k_{12}[5])); \ \blacktriangleleft\\
k_0[2] &= k_{12}[2] \oplus S(k_{13}[15] \oplus k_{13}[11]) \oplus S(k_{13}[15] \oplus k_{13}[7]) \oplus S(k_{13}[15] \oplus k_{13}[11]\\
&\quad \oplus k_{13}[7] \oplus k_{13}[3]) \oplus S(k_{13}[15] \oplus S(k_{12}[15])) \oplus S(k_{13}[15] \oplus k_{13}[11]\\
&\quad \oplus S(k_{12}[15] \oplus k_{12}[11])) \oplus S(k_{13}[15] \oplus k_{13}[7] \oplus S(k_{12}[15] \oplus k_{12}[7]));\\
k_1[10] &= k_{13}[10] \oplus k_{13}[2] \oplus S(k_{13}[14] \oplus k_{13}[10]) \oplus S(k_{12}[14] \oplus k_{12}[10] \oplus k_{12}[6]\\
&\quad \oplus k_{12}[2]);\\
\overline{k_{12}}[4] &= MC^{-1}(k_{12}[7], k_{12}[6], k_{12}[5], k_{12}[4]);\\
k_0[4] &= k_{12}[4] \oplus S(k_{13}[13] \oplus k_{13}[9]) \oplus S(k_{13}[13] \oplus k_{13}[9] \oplus k_{13}[5] \oplus k_{13}[1])\\
&\quad \oplus S(k_{13}[13] \oplus k_{13}[9] \oplus S(k_{12}[13] \oplus k_{12}[9]));\\
k_0[14] &= k_{12}[14] \oplus k_{12}[6] \oplus S(k_{13}[15] \oplus k_{13}[11] \oplus k_{13}[7] \oplus k_{13}[3]);\\
k_0[3] &= k_{12}[3] \oplus S(k_{13}[12] \oplus k_{13}[8]) \oplus S(k_{13}[12] \oplus k_{13}[4]) \oplus S(k_{13}[12] \oplus k_{13}[8]\\
&\quad \oplus k_{13}[4] \oplus k_{13}[0]) \oplus S(k_{13}[12] \oplus S(k_{12}[12])) \oplus S(k_{13}[12] \oplus k_{13}[8] \oplus S(k_{12}[12]\\
&\quad \oplus k_{12}[8])) \oplus S(k_{13}[12] \oplus k_{13}[4] \oplus S(k_{12}[12] \oplus k_{12}[4]));\\
k_0[9] &= k_{12}[9] \oplus k_{12}[1] \oplus S(k_{13}[14] \oplus k_{13}[10] \oplus k_{13}[6] \oplus k_{13}[2]);\\
&\quad \oplus S(k_{13}[14] \oplus k_{13}[10] \oplus S(k_{12}[14] \oplus k_{12}[10]));\\
k_1[5] &= k_{13}[5] \oplus S(k_{12}[13]) \oplus S(k_{12}[13] \oplus k_{12}[5]) \oplus S(S(k_{13}[14] \oplus k_{13}[10])\\
&\quad \oplus k_{12}[13]) \ \blacktriangleleft\\
\overline{k_{12}}[0] &= MC^{-1}(k_{12}[0], k_{12}[1], k_{12}[2], k_{12}[3]) \ \blacktriangleleft
\end{aligned}
$$

A.3 An Alternative Attack on 13-Round AES-256 with $T = 2^{240}$

1. Let k'_b be $k_3[14,15]$ and k'_f be $k_{10}[12]$, $k_{11}[12 \sim 15]$, $k_{12}[12,13]$ and $k_{13}[8,9]$. Guess k'_b, k'_f:
   (a) Initialize counters for all possible values for k^*_b and k^*_f.
   (b) Derive the difference for partial k_b and k_f: From $\Delta k_3[14,15]$ and $k_3[14,15]$, compute u_1, u_2 so $\Delta k_0[1,2,5,6]$ is known; from $k_{10}[12]$ and b compute u_5 so $\Delta k_{11}[0,4,8,12]$ are known; from $k_{11}[12 \sim 15]$, compute $u_6 \sim u_9$ and then Δk_{12} is known; from $k_{12}[12,13]$, compute u_{10}, u_{11}, u_{14} so now the first two rows of Δk_{13} are known. From $k_{13}[8,9]$ and $k_{12}[12,13]$ compute $k_{13}[12,13]$.
   (c) Do partial encryption and decryption and then construct pairs of data satisfying 3 filtering bytes: $\Delta x_0[5,6] = 0$, *i.e.*, $r'_b = 16$, and $\Delta x_{12}[12] = \Delta x_{12}[13]$ due to the MDS property of MC, *i.e.*, $r'_f = 8$. There will be $2^{88\times2-3\times8} = 2^{152}$ pairs of data.
   (d) For each pair of data, recover the remaining key bytes of k_b, k_f.
      i. Calculate $\Delta z_{11}[12]$ from $\Delta x_{12}[12]$ and $\Delta k_{12}[12]$. From the known input difference and output difference of the S-box, recover $z_{11}[12]$.
      ii. Look up table H_1 using $\Delta k_{12}[14,15]$, $\mathtt{MC}_1^{-1}(k_{12}[12,13], x_{12}[12,13])$, $\Delta z_{11}[12]$, $C[10,11,14,15]$, $\tilde{C}[10,11,14,15]$, $\Delta x_{12}[2,11]$, $k_{11}[10,15]$, $\Delta C[9,12]$ and get one solution for $k_{12}[14,15]$ and $k_{13}[10,11,14,15]$.
      iii. Now the whole k_{13} can be known from Δx_{12}, Δz_{12} and the ciphertext. Also, $k_3[12 \sim 15], k_{12}[8 \sim 15]$ and $\overline{k_{12}}[0,4]$ are known. Calculate Δk_0 and derive $k_0[0]$.
      iv. Look up table H_2 using $k_0[0], P[8,13]$, $\tilde{P}[8,13]$ and five values from known bytes of k_{12}, k_{13} and get one solution for $\Delta w_0[10]$, $k_0[8,13]$, $k_1[10]$ and $k_{12}[0,5]$.
      v. Recover the other key bytes as in the original attack.
      vi. Test the key candidates exhaustively to find the right master key.

Complexities. This attack has a time complexity of 2^{240}, which is reduced by a factor of 2^8 at a cost of precomputed tables of size 2^{144}. The data complexity remains 2^{89}.

Look-up Tables H_1 and H_2. These two tables take a memory complexity of 2^{144} and preparing them takes a time complexity 2^{160}.

To build H_1, we try all possible values for the following 20 bytes:

$$\Delta k_{12}[14,15], k_{12}[14,15], \Delta z_{11}[12], \mathtt{MC}_1^{-1}(k_{12}[12,13], x_{12}[12,13]),$$
$$C[10,11,14,15], \tilde{C}[10,11,14,15], \Delta x_{12}[2,11], k_{11}[14,15], \Delta C[9,12],$$

and compute

$$\Delta k_{13}[2,3,6,7,10,11,14,15], \Delta z_{12}[10,15], z_{12}[10,15], k_{13}[10,15],$$
$$k_{13}[11,14], \Delta x_{12}[6,7], \Delta x_{12}[14,15], x_{12}[14,15], z_{11}[12], k_{12}[14,15].$$

These deduced bytes should satisfy

$$\begin{aligned}
\Delta x_{12}[6] &= \Delta x_{12}[7],\\
z_{11}[12] &= \texttt{MC}_2^{-1}(k_{12}[14,15], x_{12}[14,15]) \oplus \texttt{MC}_1^{-1}(k_{12}[12,13], x_{12}[12,13]).
\end{aligned}$$

Store the remaining cases in a hash table indexed by

$$\begin{aligned}
&\Delta k_{12}[14,15], \Delta z_{11}[12], \texttt{MC}_1^{-1}(k_{12}[12,13], x_{12}[12,13]),\\
&C[10,11,14,15], \tilde{C}[10,11,14,15], \Delta x_{12}[2,11], k_{11}[14,15], \Delta C[9,12],
\end{aligned}$$

and each index will map to one item of

$$k_{12}[14,15], k_{13}[10,11,14,15].$$

Building this table takes a time complexity of 2^{160} and this table takes a memory complexity of 2^{144}.

The table H_2 can be built similarly. Assume $k_{12}[8 \sim 15]$ and k_{13} are known. From the relationship of key bytes in Appendix A.2, we have

$$\begin{aligned}
k_0[8] &= k_{12}[0] \oplus known_1,\\
k_0[13] &= k_{12}[5] \oplus known_2,\\
k_0[0] &= k_{12}[0] \oplus S(S(k_{12}[5] \oplus known_3) \oplus known_4) \oplus known_5.
\end{aligned}$$

We try all possible values for

$$\Delta w_0[10], k_0[0], P[8,13], \tilde{P}[8,13], known_1 \sim known_5,$$

and compute $k_{12}[0,5], k_1[10]$. Then test if

$$k_0[0] = k_{12}[0] \oplus S(S(k_{12}[5] \oplus known_3) \oplus known_4) \oplus known_5$$

holds. For those values satisfying this condition, store them in a hash table indexed by $k_0[0], P[8,13], \tilde{P}[8,13], known_1 \sim known_5$ and each index maps to one item of $\Delta w_0[10], k_{12}[0,5], k_1[10], k_{12}[0,5]$ on average. In this way, the table H_2 is built with time and memory complexities of 2^{88} and 2^{80}, respectively.

B More Results of KATAN-32

B.1 A Key recovery model for Finding the Best Attacking Parameters

In this subsection, we briefly introduce a model as a supplement of our new key recovery algorithm. The model will assist us in finding the best attacking parameters such that the corresponding attack's time complexity is optimal.

Algorithm 4: Optimal key guessing strategy searching

```
1: input: a differential distinguisher with $(\Delta x, \Delta y, p)$, i.e., the input difference, the output difference, and the probability; the differences of the extend rounds.
2: output: $k'_b, k'_f, r'_b, r'_f$ and the minimal time complexity $T$.
3: // Difference propagation.
4: Model the differentials $\Delta P \xleftarrow{E_b^{-1}} \Delta x$ and $\Delta y \xrightarrow{E_f} \Delta C$. Using the binary variables to mark the state differences is fixed or not. Compute $r_b$ and $r_f$.
5: // Value Propagation.
6: To verifying $\Delta x$ and $\Delta y$ differences, the state values are needed. 0/1 variables are used to mark the state values and the subkey $k_b$ and $k_f$.
7: // Guess-and-determine.
8: Model the relation between the filter bits and the subkey bits. The internal state bits are determined when the corresponding subkey bits are guessed. If an internal state bit resulting from some active bits is determined and should have a fixed difference, then a filter is obtained. Compute $r'_b$ and $r'_f$.
9: // Key relationship.
10: According to the key schedule, model the relation between round key bits. Compute $|k'_b \cup k'_f|$.
11: //Computing time complexity.
12: Model $T \geq T_s$, which is every part of the time complexity.
13: // Objective function.
14: MINIMIZE $T$.
```

B.2 An alternative 151-round attack on KATAN-32

We provide an alternative 151-round attack on KATAN-32 with 33-round E_b and 27-round E_f based on the 91-round distinguisher, as shown in Table 4.

BDMA. The parameters are $|k_b| = 49$ and $|k_f| = 30$. The key information is listed as follows:

$$k_b = k_0, k_1, \cdots, k_{44}, k_{45}, k_{48}, k_{52}, k_{53},$$
$$k_f = k_{260}, k_{268}, k_{270}, k_{272}, k_{274}, k_{276}, k_{277}, k_{278}, k_{280}, k_{281}, \cdots, k_{300}, k_{301}.$$

According to the Equation 1, the time complexity is

$$\begin{aligned} T &= 2^{31.98} \cdot (2^{49} + 2^{30}) + 2^{49+30-32+31.98} + 2^{80-32+31.98} \\ &= 2^{80.98} + 2^{61.98} + 2^{78.98} + 2^{79.98} \approx 2^{80.98} \end{aligned}$$

with data and memory complexities $D = 2^{32}, M = 2^{30}$.

GDMA. The best parameters are $|k'_b| = 39$, $|k'_f| = 21$, $r'_b = 24$, $r'_f = 18$, $r_b = 32$ and $r_f = 29$. The filtering bits are marked in blue and red in Table 4.

$$k'_b = k_0, k_1, \cdots, k_{28}, k_{29}, k_{31}, k_{32}, k_{33}, k_{35}, k_{36}, k_{37}, k_{38}, k_{40}, k_{41},$$
$$k'_f = k_{272}, k_{276}, k_{280}, k_{282}, k_{283}, k_{284}, k_{286}, k_{287}, k_{288}, k_{290}, k_{291}, \cdots, k_{300}, k_{301}.$$

With these parameters, one can have an attack of data, memory, and time complexities $D = 2^{32}$, $M = 2^{63}$ and $T = 2^{79.98}$, as

$$\begin{aligned} T &= 2^{32+39} + 2^{32+39+32-24-3-1} + 2^{32+21} + 2^{32+21+32-18-3-1} \\ &+ 2^{32+37+32+21-25-18-3-1} + 2^{79-0.02} + 2^{80+31.98-32} \\ &= 2^{71} + 2^{75} + 2^{53} + 2^{63} + 2^{78} + 2^{78.98} + 2^{80+31.98-32} \approx 2^{79.98}. \end{aligned}$$

Table 4: The 151(33 + 91 + 27)-round attack on the KATAN-32. For round t, $\varDelta_t$ is the t-th round difference, $0 \le t \le 151$. The blue and red denote the filters by guessing a part of key information k'_b and k'_f.

$\varDelta_0$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{32}$	?000 1000 0000 0?11 0101 0100 0100 0000
$\varDelta_1$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{in}$	0001 0000 0000 0110 1010 1000 1000 0000
$\varDelta_2$	???? ???? ???? ???? ???? ???? ???? ????	..	....
$\varDelta_3$	???? ???? ???? ???? ???? ???? ???? ????	..	....
$\varDelta_4$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{out}$	0000 0000 0100 0000 0000 0000 0000 0000
$\varDelta_5$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{125}$	0000 0000 1000 0000 0000 0000 0000 0001
$\varDelta_6$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{126}$	0000 0001 0000 0000 0000 0000 0000 0010
$\varDelta_7$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{127}$	0000 0010 0000 0000 0000 0000 0000 010?
$\varDelta_8$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{128}$	0000 0100 0000 0000 0000 0000 0000 10?0
$\varDelta_9$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{129}$	0000 1000 0000 ?000 0000 0000 0001 0?01
$\varDelta_{10}$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{130}$	0001 0000 000? 0000 0000 0000 0010 ?01?
$\varDelta_{11}$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{131}$	0010 0000 00?0 ?000 0000 0000 010? 01?0
$\varDelta_{12}$	???? ???? ???? ???? ???? ???? ???? ????	$\varDelta_{132}$	0100 0000 0?0? 0000 0000 0000 10?0 1?00
$\varDelta_{13}$	???? ???? ???? ???? ???? ???? ???? ???0	$\varDelta_{133}$	1000 0000 ?0?0 ?000 0000 0001 0?01 ?000
$\varDelta_{14}$	???? ???? ???? 1??? ???? ???? ???? ??0?	$\varDelta_{134}$	0000 000? 0?0? ?000 0000 0010 ?01? 0001
$\varDelta_{15}$	???? ???? ???1 ???? ???? ???? ???? ?0?1	$\varDelta_{135}$	0000 00?0 ?0?? ?000 0000 010? 01?0 001?
$\varDelta_{16}$	???? ???? ??1? ???? ???? ???? ???? 0?11	$\varDelta_{136}$	0000 0?0? 0??? ?000 0000 10?0 1?00 01?0
$\varDelta_{17}$	???? ???? ?1?? ???? ???? ???? ???0 ?110	$\varDelta_{137}$	0000 ?0?0 ???? 1000 0001 0?01 ?000 1?0?
$\varDelta_{18}$	???? ???? 1??? 0??? ???? ???? ??0? 1101	$\varDelta_{138}$	000? 0?0? ???1 ?000 0010 ?01? 0001 ?0??
$\varDelta_{19}$	???? ???1 ???0 1??? ???? ???? ?0?1 1010	$\varDelta_{139}$	00?0 ?0?? ??1? ?000 010? 01?0 001? 0???
$\varDelta_{20}$	???? ??1? ??01 ???? ???? ???? 0?11 0101	$\varDelta_{140}$	0?0? 0??? ?1?? ?000 10?0 1?00 01?0 ????
$\varDelta_{21}$	???? ?1?? ?01? 0??? ???? ???0 ?110 1010	$\varDelta_{141}$	?0?0 ???? 1??? ?001 0?01 ?000 1?0? ????
$\varDelta_{22}$	???? 1??? 01?0 0??? ???? ??0? 1101 0101	$\varDelta_{142}$	0?0? ???1 ???? ?010 ?01? 0001 ?0?? ????
$\varDelta_{23}$	???1 ???0 1?00 0??? ???? ?0?1 1010 1010	$\varDelta_{143}$	?0?? ??1? ???? ?10? 01?0 001? 0??? ????
$\varDelta_{24}$	??1? ??01 ?000 1??? ???? 0?11 0101 0100	$\varDelta_{144}$	0??? ?1?? ???? ?0?0 1?00 01?0 ???? ????
$\varDelta_{25}$	?1?? ?01? 0001 0??? ???0 ?110 1010 1000	$\varDelta_{145}$	???? 1??? ???? ??01 ?000 1?0? ???? ????
$\varDelta_{26}$	1??? 01?0 0010 0??? ??0? 1101 0101 0001	$\varDelta_{146}$	???1 ???? ???? ?01? 0001 ?0?? ???? ????
$\varDelta_{27}$	???0 1?00 0100 0??? ?0?1 1010 1010 0010	$\varDelta_{147}$	??1? ???? ???? ?1?0 001? 0??? ???? ????
$\varDelta_{28}$	??01 ?000 1000 0??? 0?11 0101 0100 0100	$\varDelta_{148}$	?1?? ???? ???? ??00 01?0 ???? ???? ????
$\varDelta_{29}$	?01? 0001 0000 0??0 ?110 1010 1000 1000	$\varDelta_{149}$	1??? ???? ???? ?000 1?0? ???? ???? ????
$\varDelta_{30}$	01?0 0010 0000 0?0? 1101 0101 0001 0000	$\varDelta_{150}$	???? ???? ???? ?001 ?0?? ???? ???? ????
$\varDelta_{31}$	1?00 0100 0000 00?1 1010 1010 0010 0000	$\varDelta_{151}$	???? ???? ???? ?01? 0??? ???? ???? ????

C The 26-Round Distinguisher of SKINNYe-64-256 v2

Table 5: The differentials of the 26-round distinguisher for `SKINNYe`-64-256 v2, where $R12$ to $R17$ denote $r_m = 6$-round middle part, u satisfies $DDT[0\text{x}1][u] > 0$ and $DDT[u \oplus 0\text{x}9][0\text{xb}] > 0$, v satisfies $DDT[0\text{x}1][v] > 0$ and $DDT[v][0\text{xb}] > 0$ [QDW+22].

	Upper differential	Lower differential
R_0	$0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0$ $0,9,0,8,0,0,0,0,0,0,0,0,0,0,0,0$ $0,9,0,8,0,0,0,0$	
$R_1 - R_6$	$0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,0,0,0$	
R_7	$0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,d,0,0,0,0,0$	
R_8	$0,0,d,0,0,0,0,d,0,0,0,0,0,0,d,0$ $0,0,2,0,0,0,u,0,0,0,0,0,0,0,v,0$ $0,0,2,0,0,0,6,0$	
R_9	$0,v,0,0,0,0,0,0,0,0,0,u \oplus 0x6,0,0,0,0$ $0,4,0,0,0,0,0,0,0,0,0,4,0,0,0,0$ $0,0,0,0,4,0,0,0$	
R_{10}	$0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,e,2,0,0$	
R_{11}	$0,0,0,0,0,0,0,0,0,0,e,0,0,0,0,0$ $0,0,0,0,0,0,0,0,0,0,9,0,0,0,0,0$ $0,0,0,0,0,0,9,0$	
R_{12}	$0,0,0,9,0,0,0,0,0,0,0,0,0,0,0,9$ $0,0,0,*,0,0,0,0,0,0,0,0,0,0,0,*$ $0,0,0,4,0,0,c,0$	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- $0,0,0,0,0,0,0,0$
$R_{13} - R_{16}$	middle part	
R_{17}	-,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- -,-,-,-,-,-,-,-,-,-,-,-,-,-,-,- $0,0,0,0,0,0,0,8$	$0,*,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,8,0,0,0,0,0,0$
$R_{18} - R_{24}$		$0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,0,0,0$
R_{25}		$0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0$ $0,0,0,0,0,0,2,0$

Table 6: The 26-round related-tweakey boomerang distinguisher for `SKINNYe`-64-256 v2 [QDW+22].

$r_0 = 12, r_m = 6, r_1 = 8, P_d = 2^{57.6}$		
ΔTK_1	$=$	$0, 6, 0, 2, 0, 0, 0, 0, 0, 0, 0, d, 0, 0, 0, 0$
ΔTK_2	$=$	$0, 9, 0, 8, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0$
ΔTK_3	$=$	$0, c, 0, b, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0$
ΔTK_4	$=$	$0, a, 0, 9, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0$
$\Delta X^{(0)}$	$=$	$0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0$
∇TK_1	$=$	$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2$
∇TK_2	$=$	$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6$
∇TK_3	$=$	$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9$
∇TK_4	$=$	$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1$
$\nabla X^{(26)}$	$=$	$0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0$

Table 7: Differential distribution table of 4-bit Sbox of `SKINNY`

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	4	4	4	4	0	0	0	0
2	0	4	0	4	0	4	4	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2
4	0	0	4	0	0	0	2	2	0	0	0	4	2	2	0	0
5	0	0	4	0	0	0	2	2	0	0	4	0	2	2	0	0
6	0	2	0	2	2	0	0	2	2	0	2	0	0	2	2	0
7	0	2	0	2	2	0	0	2	0	2	0	2	2	0	0	2
8	0	0	0	0	4	4	0	0	0	0	0	0	2	2	2	2
9	0	0	0	0	4	4	0	0	0	0	0	0	2	2	2	2
a	0	0	0	0	0	4	4	0	2	2	2	2	0	0	0	0
b	0	4	0	4	0	0	0	0	0	0	0	0	2	2	2	2
c	0	0	4	0	0	0	2	2	4	0	0	0	0	0	2	2
d	0	0	4	0	0	0	2	2	0	4	0	0	0	0	2	2
e	0	2	0	2	2	0	0	2	0	2	0	2	0	2	2	0
f	0	2	0	2	2	0	0	2	2	0	2	0	2	0	0	2