Improved Differential Meet-In-The-Middle Cryptanalysis

Zahra Ahmadian[1], Akram Khalesi[1], Dounia M'Foukh[2], Hossein Moghimi[1] and María Naya-Plasencia[2]

[1] Shahid Beheshti University, Iran
{z_ahmadian,a_khalesi}@sbu.ac.ir, h.moghimi@mail.sbu.ac.ir
[2] Inria, France
{dounia.mfoukh,maria.naya_plasencia}@inria.fr

Abstract. In this paper, we extend the applicability of differential meet-in-the-middle attacks, proposed at Crypto 2023, to truncated differentials, and in addition, we introduce three new ideas to improve this type of attack: we show how to add longer structures than the original paper, we show how to improve the key recovery steps by introducing some probability in them, and we combine this type of attacks with the state-test technique, that was introduced in the context of impossible differential attacks. Furthermore, we have developed a MILP-based tool to automate the search for a truncated differential-MITM attack with optimized overall complexity, incorporating some of the proposed improvements. Thanks to this, we can build the best known attacks on the cipher `CRAFT`, reaching 23 rounds against 21 previously; we provide a new attack on 23-round `SKINNY-64-192`, and we improve the best attacks on `SKINNY-128-384`.

Keywords: differential meet-in-the-middle cryptanalysis, truncated cryptanalysis, parallel partitioning, state-test technique, tool, `SKINNY`, `CRAFT`

1 Introduction

Symmetric cryptanalysis is crucial for trusting symmetric primitives: the more we cryptanalyze a primitive without success, the more confidence we will have in it. It is essential for determining the security margin of the cryptographic constructions and being able to anticipate problems. Many different cryptanalysis families exist, like differential attacks [10], linear attacks [26], meet-in-the-middle attacks [14], invariant sub-space attacks [24], integral attacks [22]. Often new techniques and variants are proposed to augment these attacks, but proposing new families is less common.

In [12] a new cryptanalysis was proposed: differential meet-in-the-middle (MITM) attack. It combines ideas from differential and MITM attacks, allowing to build the best known attack on the cipher `SKINNY-128-384` in the single-tweak setting [12]. As described by the authors, these attacks could be seen in part as a

new way of performing the key recovery in differential attacks, or as MITM ones where instead of looking for a partial collision at some middle state, we look for states that verify, with certain probability, a differential relation. Because of the MITM nature, the authors proposed in the SKINNY-128-384 scenario, a way of extending the attack using parallel partitioning to gain one round, techniques usually applied in MITM attacks [3,4,11], but not applicable during classical differential attacks. Some questions were left unsolved in the original paper, such as whether these attacks could be seen as just a new way of performing the key recovery part, but were in essence differential attacks. Another interesting question was if they could be combined with truncated differential attacks – as the probability in both directions of a truncated path is often not symmetric, it seemed at first counter-intuitive to apply it. We have considered and answered these two questions, and in addition proposed two additional improvements to the technique: allowing some probability in the key-guessing part, and combining it with the state-test technique, introduced in [13] in the context of impossible differential attacks.

We have applied our new techniques to CRAFT [9], SKINNY-128-384 [8] and SKINNY-64-192 [8], providing the best known attacks in the two first cases, and an attack reaching the highest number of rounds as the best attack for the third. These attacks can be seen in Tab. 1.

This paper is organized as follows: Sec. 2 presents the previous framework of differential MITM attacks. Sec. 3 describes our proposition of combining this attack with truncated differentials and section 4 the newly proposed improvements. Sec. 5 presents our new tool that finds the distinguishers providing the best overall attacks, considering in addition most of the new improvements on the external rounds, and Sec. 6 and Sec. 7 describe the new applications. The paper is ended with a conclusion.

2 Preliminaries: Differential Meet-in-the-Middle

The differential meet-in-the-middle (MITM) technique, introduced in [12], represents a novel approach for the cryptanalysis of symmetric primitives. This attack combines two significant families of symmetric cryptanalysis attacks: the meet-in-the-middle attack and the differential attack. In [12], it is described as both an extension of classical MITM attacks and as a new key recovery method to apply in differential cryptanalysis.

This new technique has been successfully applied to SKINNY-128-384, the 128-bit block cipher variant employing a 384-bit tweakey, achieving the result of breaking 25 out of the 56 rounds in the single-tweakey setting [12]. This application highlights the attack's efficiency by surpassing the best known attack on this cipher by two rounds. Furthermore, another instance provided in [12], involved AES-256, where this technique managed to break 12 rounds of the cipher in the related-key model. This attack requires only 2 related keys while the previous attacks with the same number of related keys achieve a maximum of 10 rounds.

Cipher	Rounds	Time	Data	Memory	Attack	Setting/Model	Ref.
CRAFT	18	$2^{101.7}$	$2^{60.92}$	2^{84}	Rectangle	STK	[18]
	19	$2^{114.68}$	2^{56}	2^{109}	DS-MITM	STK,CP	[25]
	19	$2^{112.61}$	$2^{60.92}$	2^{72}	Rectangle	SK	[31]
	20	$2^{126.96}$	2^{56}	2^{109}	DS-MITM	STK,CP	[25]
	21	$2^{106.53}$	$2^{60.99}$	2^{100}	ID	STK,CP	[19]
	21	$\mathbf{2^{116}}$	$\mathbf{2^{56}}$	$\mathbf{2^{68}}$	**Tr-Diff-MITM**	**STK**	**Sec. 6**
	22	$\mathbf{2^{125}}$	$\mathbf{2^{58}}$	$\mathbf{2^{72}}$	**Tr-Diff-MITM**	**STK**	**Sec. 6**
	23	$\mathbf{2^{125}}$	$\mathbf{2^{60}}$	$\mathbf{2^{68}}$	**Tr-Diff-MITM**	**STK**	**Sec. 6**
SKINNY-64-192	21	$2^{180.01}$	2^{44}	$2^{191.55}$	DS-MITM	STK	[30]
	21	$2^{174.42}$	$2^{62.43}$	2^{168}	ID	STK/CP	[19]
	22	$2^{183.97}$	$2^{47.84}$	$2^{74.84}$	ID	STK	[33]
	23	2^{188}	2^{52}	2^{4}	MITM	STK	[15]
	23	2^{184}	2^{60}	2^{8}	MITM	STK	[7]
	23	2^{188}	2^{28}	2^{4}	MITM	STK	[7]
	23	$\mathbf{2^{188}}$	$\mathbf{2^{56}}$	$\mathbf{2^{104}}$	**Tr-Diff-MITM**	**STK**	**Sec. 7.1**
SKINNY-128-384	23	2^{376}	2^{104}	2^{8}	MITM	STK	[15]
	23	2^{372}	2^{96}	$2^{352.46}$	DS-MITM	STK	[30]
	23	$2^{361.9}$	2^{117}	$2^{118.5}$	Diff-MITM	STK	[12]
	24	$2^{361.9}$	2^{117}	2^{183}	Diff-MITM	STK	[12]
	24	$2^{372.5}$	$2^{122.3}$	$2^{123.8}$	Diff-MITM	STK	[12]
	25	$2^{372.5}$	$2^{122.3}$	$2^{188.3}$	Diff-MITM	STK	[12]
	25	$2^{378.9}$	$\mathbf{2^{117}}$	$\mathbf{2^{165}}$	**Diff-MITM**	**STK**	**Sec. 7.2**
	25	$\mathbf{2^{366}}$	$\mathbf{2^{122.3}}$	$\mathbf{2^{188.3}}$	**Diff-MITM**	**STK**	**Sec. 7.2**

MITM: Meet In The Middle — ID: Impossible Differential
CP: Chosen Plaintext — DS-MITM: Demirci-Selcuk
Diff-MITM: Differential MITM — Tr-Diff-MITM: Truncated Differential MITM
STK: Single-Tweak/Tweakey — SK: Single Key

Table 1. Summary of the best known cryptanalysis on CRAFT, SKINNY-64-192 and SKINNY-128-384 in the single tweak setting (related tweak the strongest setting).

In this section, we will provide an overview of how differential MITM attacks are constructed, along with the original improvements introduced in [12]. One improvement, achieved through a parallel treatment of data partitions, extends the attack for one round more, mostly at no cost, particularly when the key was exclusively added to half of the internal state. Another one is a technique designed to reduce data complexity in the originally full code-book attack scenario.

2.1 Framework of the differential MITM attack

Consider an n-bit cipher E decomposed into three sub-ciphers $E_{out} \circ E_m \circ E_{in}$ of r_{in}, r_m and r_{out} rounds respectively, as depicted in Fig. 1. Finally, suppose that the efficient differential $(\alpha \to \beta)$, covering the r_m middle rounds, with probability

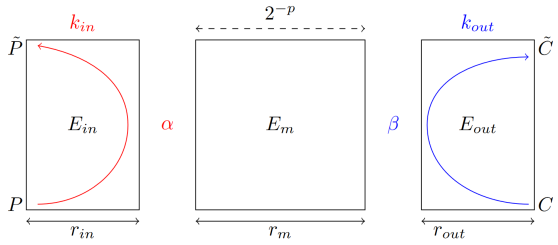

Fig. 1. Framework of the differential meet-in-the-middle attack.

2^{-p} is a distinguisher for E_m. The core idea of the attack involves employing the MITM approach. In other words, given a pair of plaintext/ciphertext (P, C), we independently compute the candidate plaintexts and ciphertexts $\tilde{P}$ and $\tilde{C}$ such that they follow the differential characteristic as summarized on Fig. 1. $\tilde{P}$ is computed from the plaintext P and the difference α for each possible value of the associated key k_{in}, and $\tilde{C}$ from the ciphertext C and the difference β for each possible value of the associated key k_{out}.

Thus the aim of the following procedure is to find a pair of plaintext/ciphertext (P, C) and $(\tilde{P}, \tilde{C})$ such that:

$$\begin{cases} E_{in}(P) \oplus E_{in}(\tilde{P}) & = \alpha \\ E_{out}^{-1}(C) \oplus E_{out}^{-1}(\tilde{C}) & = \beta. \end{cases} \tag{1}$$

Procedure. First, we randomly pick a plaintext P and ask the encryption oracle for its ciphertext C.

- *Upper part:* From P, α and some key information, we aim to compute $\tilde{P}$ such that $E_{in}(P) \oplus E_{in}(\tilde{P}) = \alpha$, if the key guess is correct. Thus, we want to minimize the amount of key information, denoted by k_{in}, needed. For each possible value i of k_{in}, we have a different candidate $\tilde{P}^i$. Thus, we have $2^{|k_{in}|}$ candidates at the end of this step. For each $\tilde{P}^i$, we ask the oracle for its encryption $\hat{C}^i$, and store them in a hash table.
- *Lower part:* Similarly, given C, β and a minimized amount of key information, denoted by k_{out}, we can compute $\tilde{C}^j$ such that $E_{out}^{-1}(C) \oplus E_{out}^{-1}(\tilde{C}^j) = \beta$, if the key guess is correct. We have $2^{|k_{out}|}$ candidates for $\tilde{C}^j$.

Actually, during the procedure before the upper and lower parts, we can first guess the subkey bits of k_{in} and k_{out} in common thanks to the possible linear relations between k_{in} and k_{out} given by the key schedule.

Furthermore, given that $P(\alpha \to \beta) = 2^{-p}$, we have to repeat the procedure 2^p times using 2^p different plaintext/ciphertexts pairs (P_l, C_l) to have a good pair $(P_l, \tilde{P}_l^i)$ and $(C_l, \tilde{C}_l^j)$, satisfying the distinguisher. When this is the case, we will get a collision between a $\hat{C}^i = E(\tilde{P}^i)$ of the upper part and a $\tilde{C}^j$ of the lower part. Each collision (i, j) yields a candidate key.

For each P_l, we initially have $2^{|k_{in}|+|k_{out}|}$ candidate pairs $(\hat{C}_l^i, i)$ and $(\tilde{C}_l^j, j)$ in search of a collision. After matching through the relation $\hat{C}_l^i = \tilde{C}_l^j$, we are left with $2^{|k_{in}|+|k_{out}|-n}$ candidates. Thus, in the end, for each P_l, the number of expected collisions would be $2^{|k_{in}|+|k_{out}|-n-|k_{in}\cap k_{out}|}$.
Complexity. The time complexity for the computations in the upper and lower parts of the procedure is $2^{|k_{in}|+p}$ and $2^{|k_{out}|+p}$, respectively. And as explained above, the number of expected candidate keys is $2^{|k_{in}|+|k_{out}|-n-|k_{in}\cap k_{out}|+p}$. Thus, the time complexity is:

$$2^p \times 2^{|k_{in}\cap k_{out}|}(2^{|k_{in}|-|k_{in}\cap k_{out}|} + 2^{|k_{out}|-|k_{in}\cap k_{out}|} + 2^{|k_{in}|+|k_{out}|-n-2|k_{in}\cap k_{out}|}).$$

With this, we recover $k_{in} \cup k_{out}$. So, if we expect fewer key candidates than the whole key space $\mathcal{K}$ of size 2^k, then we can guess the remaining bits of the master key and test the guess with additional pairs. Thus, we recover the whole key with a complexity smaller than the cost of an exhaustive key search, and the additional cost of $2^{k-|k_{in}\cup k_{out}|} \times \max(1, 2^{|k_{in}|+|k_{out}|-n-|k_{in}\cap k_{out}|+p})$ to be added to the time complexity $\mathcal{T}$. In the case that we need to guess the remaining bits of the master key, specifically if $|k_{in}| + |k_{out}| - n - |k_{in} \cap k_{out}| + p > 0$, the total time complexity would be:

$$\begin{aligned}\mathcal{T} = {} & 2^{p+|k_{in}\cap k_{out}|}(2^{|k_{in}|-|k_{in}\cap k_{out}|} + 2^{|k_{out}|-|k_{in}\cap k_{out}|} + 2^{|k_{in}|+|k_{out}|-n-2|k_{in}\cap k_{out}|}) \\ & + 2^{k-n+p}. \end{aligned} \tag{2}$$

The data complexity is $\mathcal{D} = \min(2^n, 2^{p+\min(|k_{in}|,|k_{out}|)})$, and the memory complexity is $\mathcal{M} = 2^{\min(|k_{in}|-|k_{in}\cap k_{out}|,|k_{out}|-|k_{in}\cap k_{out}|)}$.

2.2 Improvement: Parallel partitions for layers with partial subkeys

In [12], two methods for improving the original differential MITM attack are proposed. The first method, elaborated upon below, focuses on adding an extra round at the beginning or the end of the attack, in some specific cases.

In the cases where the round key addition only affects a part of the state, as is the case with `SKINNY` [8] or `GIFT` [6] block ciphers, the differential MITM attack can be extended by one round. Suppose $m < n$ bits of the state are affected by the round key addition. The framework of the improvement is given in Fig. 2, where one round is added at the end of the attack covering $r - 1$ rounds of the cipher and we have eliminated all the transformations after the round key addition of round r, if any. Let A and B denote the states before and after adding K_r, respectively. The main idea of the improvement is to only keep the ciphertexts that satisfy the following condition: we fix the $n - m$ bits that are not affected by the key addition in A and B, and compute all the 2^m possible values for A and B. Now, we will repeat the procedure 2^{p-m} times. So, we can apply this improvement without increasing the time complexity, if $p > m$.

Then, from all the 2^m possible values for A, we can proceed with the lower part of the differential MITM attack. We get $2^{|k_{out}|+m}$ possible candidates $(A, \tilde{A}, j)$. In parallel, we do the same for each of the 2^m possible values for B, and we proceed with the upper part of differential MITM attack and we get $2^{|k_{in}|+m}$ possible

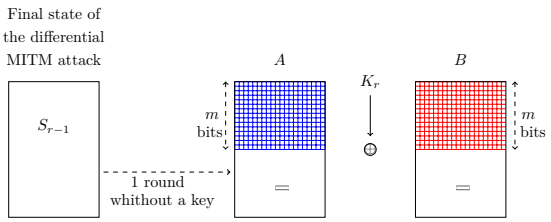

Fig. 2. Framework of the parallel partitioning improvement.

candidates $(B, \tilde{B}, i)$; hence $2^{|k_{in}|+|k_{out}|+2m}$ total candidates which is a factor 2^m higher than before, as we are comparing this to performing the attack without structures 2^m times. Nevertheless, note that we have to match A and B and their associated pairs $\tilde{A}$ and $\tilde{B}$ through the relation $A \oplus B = \tilde{A} \oplus \tilde{B}$. This also yields the value of K_r, already determined by k_{in} and k_{out}, usually. So, the number of expected collisions does not increase: in total, we need to collide on $n - m$ bits of the key-free parts of $\tilde{A}$ and $\tilde{B}$; and we have to collide for both pairs in the m bits once the key is added, providing a total filtering of $(n - m) + 2m = n + m$. As we now have 2^m additional potential candidates, but a 2^{m+n} filter, we expect a total proportion of remaining candidates of 2^{-n}, as in the attack without parallel partitions. In [12], they applied this technique to reach one more round of `SKINNY-128-384` without increasing the time complexity and thus mounting the best known attack on this cipher in the single tweak setting.

2.3 Reducing data needed with imposed conditions

The idea is to fix x bits of the plaintext P and of the associated plaintext $\tilde{P}$ to a specific value, thereby restricting them to a set of 2^{n-x} plaintexts instead of the whole codebook. We can choose the plaintext P as desired but the probability to get a $\tilde{P}$ with the fixed x bits is 2^{-x}. Then, the probability of the attack decreases by 2^x as we have to repeat the procedure 2^x times to get a pair that satisfies the differential characteristic along with this new condition. On the other hand, it means that during the upper and lower part of the attack, we will discard a proportion of 2^x tuples of $(P, \tilde{P}, i)$ and $(C, \tilde{C}, j)$, the ones that do not satisfy the conditions. Thus the data and memory complexities will decrease by 2^x.

Furthermore, considering the attributes of the differential MITM attack, we can derive the following two bounds on the number of fixed bits x:

$$p + x \leq n - x \text{ and } 2^{p+x}(2^{|k_{in}|} + 2^{|k_{out}|}) < 2^k.$$

This technique particularly applies when the whole codebook would be needed, as it is in the differential MITM attack on `SKINNY-128-384` presented in [12].

3 Truncated Differential Meet-in-the-Middle Attack

Truncated differential cryptanalysis was introduced at FSE in 1994 [21] by Knudsen as an extension of differential cryptanalysis and has proven its efficacy by successfully attacking several ciphers which seemed to be secure against differential attacks. It is the case with the KLEIN block cipher, for which its security against bit-wise differential attack had been proved but was broken by some truncated differential attacks [23,28]. Thus the main extension of the differential MITM attack that we will explain in this section is the truncated differential MITM attack. A challenge of building this extension is that since the probability of a truncated differential characteristic is not the same in both directions, then it was not clear how to properly take this propriety into account in the truncated differential MITM extension.

The main idea of truncated differential-MITM attacks is to use, instead of a differential path, a truncated differential path as the underlying distinguisher of the attack. As stated in Appendix A of the longer version of this paper [2], a truncated differential operates based on sets of input and output differences rather than concrete ones. It considers whether a word (typically of the S-box size in the cipher) has a non-zero difference or not, regardless of its concrete value. One advantage of the truncated differential attack is that during the key recovery step, we do not need to know the concrete values of the states just before and after the distinguisher. Consequently, we may need to guess fewer subkey words, potentially allowing us to reach more rounds. Additionally, for certain ciphers, truncated differential distinguishers can reach more rounds than concrete differentials (as in [27]). Finally, the search space for truncated differentials is much smaller than that of concrete differentials, making it easier to deal with an automated method [27].

3.1 Framework of the truncated differential MITM attack

Similar to the differential MITM attack, consider the n-bit cipher E decomposed into three sub-ciphers: $E_{out} \circ E_m \circ E_{in}$, of r_{in}, r_m and r_{out} rounds respectively, as depicted in Fig. 3. Finally, suppose that $(\Delta_{in} \xrightarrow{E_m} \Delta_{out})$ is a truncated distinguisher for E_m with the probability of 2^{-p}, where $|\Delta_{in}| = 2^{\delta_{in}}$ and $|\Delta_{out}| = 2^{\delta_{out}}$. So, according to Def. 2, $(\Delta_{in} \xrightarrow{E_m} \Delta_{out})$ is an efficient differential if $p < n - \delta_{out}$.

We randomly pick a pair of plaintext/ciphertext pair (P, C), and try to generate appropriate candidates for $(\tilde{P}, \tilde{C})$ such that the difference of these two data ensures the truncated difference Δ_{in} at round r_{in}, i.e. $E_{in}(P) \oplus E_{in}(\tilde{P}) \in \Delta_{in}$, and the truncated difference Δ_{out} at round $r_{in} + r_m$, i.e. $E_{out}^{-1}(C) \oplus E_{out}^{-1}(\tilde{C}) \in \Delta_{out}$. The procedures of the upper and lower parts of the attack are as follows.

Upper part: To generate $\tilde{P}$, we guess some key material, denoted by k_{in}. For each P, and each guess i of k_{in}, there are $2^{\delta_{in}}$ distinct candidates $\tilde{P}_l^i$, each corresponding to an input difference $l \in \Delta_{in}$. All the $2^{|k_{in}|+\delta_{in}}$ ciphertexts $\hat{C}_l^i = E_K(\tilde{P}_l^i)$, along with its associated key material i are stored in a hash table.

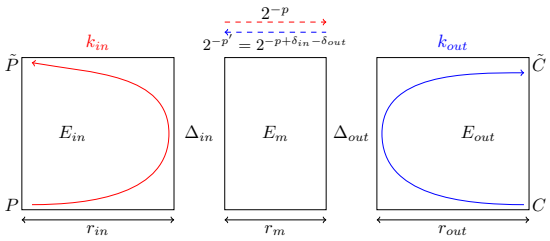

Fig. 3. Framework of the truncated differential meet-in-the-middle attack. The probability of the distinguisher in the forward (backward) direction is 2^{-p} ($2^{-p'}$).

Lower part: In the ciphertext side, given C, for each guess j of k_{out}, we compute all the $2^{\delta_{out}}$ ciphertexts corresponding to the $2^{\delta_{out}}$ possible differences $m \in \Delta_{out}$. So, there would be in total $2^{|k_{out}|} \times 2^{\delta_{out}}$ values for $\tilde{C}_m^j$.

Number of pairs: For the correct key guess, the transition $(\Delta_{in} \xrightarrow{E_m} \Delta_{out})$ will happen with probability 2^{-p}. So, a number of 2^p pairs of data is expected to be tested to find the correct key. For each P and a fixed key material i, we can provide $2^{\delta_{in}}$ differential pairs $(P, \tilde{P})$, each of which corresponds to a specific value of Δ_{in}. So, it is required to repeat the upper and lower parts of the attack for a number of $2^{p-\delta_{in}}$ plaintexts P. On the other hand, despite the concrete differential-MITM attack, the output difference Δ_{out} does not have a specific single value, but it belongs to a set of size $2^{\delta_{out}}$, whole of which should be checked to certainly determine if the event $(\Delta_{in} \xrightarrow{E_m} \Delta_{out})$ has occurred or not.

3.2 Attack complexities

As done in the differential MITM attack, we first guess the possible linear relations between k_{in} and k_{out}, i.e. $k_{in} \cap k_{out}$. During the upper part of the attack, for each guess of $k_{in} - k_{in} \cap k_{out}$, we get $2^{\delta_{in}}$ candidates $\tilde{P}_l^i$ hence $2^{|k_{in}|+\delta_{in}-|k_{in}\cap k_{out}|}$ candidates for $(P, \tilde{P}, i)$. Similarly, in the lower part, we have $2^{|k_{out}|+\delta_{out}-|k_{in}\cap k_{out}|}$ candidates triplets for $(C, \tilde{C}, j)$. So, there is $2^{|k_{in}|+\delta_{in}+|k_{out}|+\delta_{out}-2|k_{in}\cap k_{out}|}$ candidates for (i, j). Let's denote $\hat{C} = E(\tilde{P})$. The matching of $\hat{C}_l^i$ with $\tilde{C}_m^j$ leaves us with $2^{|k_{in}|+\delta_{in}+|k_{out}|+\delta_{out}-2|k_{in}\cap k_{out}|-n}$ candidates. Moreover, similar to the original differential-MITM attack, we can guess the remaining bits of the master key and test the guess with additional pairs. Thus, the time complexity of the attack is:

$$\begin{aligned}\mathcal{T} = {} & 2^{p-\delta_{in}} \times 2^{|k_{in}\cap k_{out}|}(2^{|k_{in}|+\delta_{in}-|k_{in}\cap k_{out}|} + 2^{|k_{out}|+\delta_{out}-|k_{in}\cap k_{out}|}) \\ & + 2^{p-\delta_{in}} \times 2^{|k_{in}\cap k_{out}|}(2^{|k_{in}|+\delta_{in}+|k_{out}|+\delta_{out}-2|k_{in}\cap k_{out}|-n}) \end{aligned} \tag{3}$$

The data and memory complexities are similar to the differential MITM ones.

$$\mathcal{D} = \min(2^n, 2^{p-\delta_{in}+\min(|k_{in}|+\delta_{in}, |k_{out}|+\delta_{out})}), \tag{4}$$

$$\mathcal{M} = \min(2^{|k_{in}|+\delta_{in}-|k_{in}\cap k_{out}|}, 2^{|k_{out}|+\delta_{out}-|k_{in}\cap k_{out}|}). \tag{5}$$

The second line of (3) refers to the number of candidates for $k_{in} \cup k_{out}$ to be tested, which should be less than the whole set $k_{in} \cup k_{out}$. This holds if $p + |k_{in}| + |k_{out}| - |k_{in} \cap k_{out}| - n + \delta_{out} < |k_{in} \cup k_{out}|$, implying that $p < n - \delta_{out}$, which is ensured by an efficient distinguisher (Def. 2).

Remark 1. According to (10), it holds for the reverse transition that $P(\Delta_{out} \xrightarrow{E_m^{-1}} \Delta_{in}) = 2^{-p'}$, where $p' = p + \delta_{out} - \delta_{in}$. Using this equality, one can infer that all the complexities of the reverse attack (the chosen ciphertext scenario) are equivalent to that of the forward attack: equations (3), (4), (5).

4 New improvements to differential MITM attacks

We present in this section three new improvements that can be incorporated into either the truncated or the original variants of differential MITM attacks. These improvements include an extension of the parallel partitioning technique, the state-test technique, and the probabilistic key recovery technique. The extended parallel partitioning technique, built upon the concept initially proposed in [12] (also reviewed in Sec. 2.2 of this paper), expands its range of applicability. The other two techniques focus on minimizing the information needed to be guessed during the key guessing step, which has a direct impact on the complexity. Specifically, the state-test technique, adopted from the impossible differential attacks [13], guesses a word of the state instead of a larger-size key material, thereby decreasing the total information that needs to be guessed. The probabilistic key recovery technique introduces a probability into the key-guessing step to reduce the number of active words in the key recovery parts of the attack, and consequently, the keybits involved.

4.1 Improving the parallel partitioning

As explained in Sec. 2.2, the original parallel partitioning method proposed in [12] effectively extends the attack for one round, in situations where the round key addition impacts only a portion of the cipher's state. This extension has no additional cost in time or data complexities if certain specific conditions are met. In this section, inspired by the structures commonly employed in MITM attacks like initial structures [5] and bicliques [20], we explain how the applicability range can be extended, specifically in two directions: One round extension, in the case of SPN ciphers with a whole-state key addition (applied to `CRAFT` in Sec. 6 of this paper). Additionally, more than one round extension, in the case of SPN ciphers with a partial-state key addition (applied in Sec. 7 of this paper).
The general idea. Without loss of generality, we explain the procedure for the round(s) extensions at the end of the cipher. A general view is shown in Fig. 4. Suppose that the cipher state is formed of W words of size s bits ($n = Ws$). Let A be the final state of the (truncated) differential-MITM attack, and B be the ciphertext, typically one or two rounds after A. Without any extra conditions, there are 2^{Ws} possibilities for A and B values, each. A set of F words,

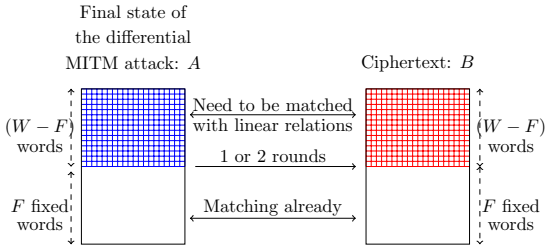

Fig. 4. High-level representation of the proposed structures when added at the end of the attack. They could also be added at the input without loss of generality.

representing independent conditions, enforced at any point within the internal states of the added rounds, would reduce the number of possibilities for each of A and B to $2^{(W-F)s}$. The whole set of these possible values for A and B is called a pair of initial structures of size $2^{(W-F)s}$. These conditions typically involve forcing some words within the internal state into fixed values or imposing linear relationships on specific internal state words. This set of conditions of size Fs bits are selected so that having them, along with k_{in} (resp. k_{out}), allows one to uniquely determine an equivalent of Fs information bits of B (resp. A). Therefore, it makes sense to consider the Fs-bit internal state condition as the starting point for the structures.

As in the previous parallel partitioning method, we next perform the upper and lower procedures for both structures of size $2^{(W-F)s}$ in parallel, generating lists of $(B, \tilde{B}, i)$ and $(A, \tilde{A}, j)$. Therefore, we need to repeat the modified attack $2^{p-(W-F)s}$ times, while the first two terms of time complexity in (2) and (3), would be increased by a factor of $2^{(W-F)s}$, and the third term by a factor of $2^{2(W-F)s}$. So it is required to have efficient sieving on the candidates when merging the two partial lists of solutions, to retain a number well below the exhaustive search. The sieving over the candidate solutions is done in two ways: Firstly, a 2^{-Fs} sieving over $\tilde{B}$ and $\tilde{A}$, in the words involved in the starting point. These words computed from the both sides, (i.e. from the Fs information bits of $\tilde{A}$ and $\tilde{B}$ possibly with the aid of the associated k_{out} and k_{in}, respectively) should return the same values. Secondly, an L-bit sieving by new linear relations between A and B (similarly, $\tilde{A}$ and $\tilde{B}$) independent from those Fs-bit matching over the starting point. To profit from the whole sieving potential, we would need to discover $2(W-F)s$ linear relations between the pairs A and B, as well as between $\tilde{A}$ and $\tilde{B}$. The accurate value of L, upper bounded by $2(W-F)s$, depends on the particular cases. Finally, as far as $(W-F)s \leq p$, this technique imposes no additional costs to the time complexity of the original attack, besides the possible increase in the size of k_{in} and k_{out} if more bits are needed to check the linear relations.

The time complexity of the truncated differential-MITM is then:

$$\mathcal{T} = 2^{p-(W-F)s-\delta_{in}+|k_{in}\cap k_{out}|} \times (2^{(W-F)s} \times 2^{|k_{in}|+\delta_{in}-|k_{in}\cap k_{out}|}$$
$$+2^{(W-F)s} \times 2^{|k_{out}|+\delta_{out}-|k_{in}\cap k_{out}|} + 2^{|k_{in}|+\delta_{in}+|k_{out}|+\delta_{out}+2(W-F)s-Fs-L-2|k_{in}\cap k_{out}|}).$$

A positive side effect of this improvement is that thanks to the linear relations checked in the second sieving, we recover most of the time the whole values of the last subkey bits, which increases the size of the number of bits recovered, as we will see in the applications.

In the following, we provide a brief description of two specific examples illustrating how this technique is applicable for more than one round (for SKINNY), or even when the key is added to the entire state (for CRAFT).

The case of SKINNY. In Sec. 7.1, we add two rounds at the end of our truncated differential MITM attack on SKINNY-64-192 as shown in Fig. 10. For the cipher SKINNY, knowing the first key row of the first added key allows checking of all the linear equations needed to profit from the whole sieving potential. In our attack, we construct structures of size 2^{40}, thus to profit from the whole sieving potential, we want to find 10 linear relations between the pairs A and B, as well as between $\tilde{A}$ and $\tilde{B}$. In our case, we guessed 3 subkey bits of the first key row of the first key which gives us 8 linear equations out of the 10 linear equations we want. And we guess the 2 subkey bits of the third column of the second key to find the 2 remaining linear relations. A similar idea is applied in our improvements of the SKINNY-128-384 attacks of section 7.2.

The case of CRAFT. In our attack of CRAFT in Sec. 6, we can check less than all the linear equations as we have a bigger margin - we do not need to use all the potentially available sieving. Thus we get our linear equation by checking the relation $\texttt{MC}(A) \oplus B = \texttt{MC}(\tilde{A}) \oplus \tilde{B}$ for all the non-fixed words, which erases the unknown key bits. Since we fixed 5 words in A and B, we get 11 linear relations on 4 bits each.

4.2 Probabilistic key recovery technique

Usually, in the key recovery step of the differential attacks, we extend with probability one the differential characteristic for some rounds in both sides. Our second idea for improvement is to force one or more transitions to have zero difference, paying some probability. Thus the number of active words will decrease and fewer key bits will need to be guessed.

Suppose we are in the case of a differential attack. In the classical case, we propagate Δ_{in} and Δ_{out} with probability 1 for r_{in} rounds backward and r_{out} rounds forward respectively, determining the truncated differences that pairs of plaintexts/ciphertexts should verify. Then we will test the pairs verifying this truncated differential and the possible keys that would lead to this differential.

Here instead of extending Δ_{in} and Δ_{out} for r_{in} and r_{out} rounds with probability 1, we allow these transitions to happen with probability $2^{-p_{in}}$ and $2^{-p_{out}}$.

Thus the overall probability for a random pair to follow the differential path is now $2^{-p-p_{in}-p_{out}}$. Therefore we have to repeat the attack $2^{p_{in}+p_{out}}$ more times. However, the number of pairs we keep for each side decreases by $2^{p_{in}}$ for the upper part and of $2^{p_{out}}$ for the lower part, so this is often compensated in the final time complexity. On the other hand, this technique restricts the large diffusion of active nibbles in the upper and lower sides, resulting in smaller sets of k_{in} and k_{out}. We will show an application of this improvement in our attack on `CRAFT` in Sec. 6.

Comparison with differential attacks. In differential attacks, the complexity of the key recovery step is usually quite low, thanks to early abort techniques, and therefore the improvement of using probabilistic key recovery thechnique might be less advantageous, though it could still be applied. Here the number of involved keybits is directly associated to the final complexity.

4.3 Applying the state-test technique

The state-test technique was introduced in [13,16] in the context of impossible differential and MITM attacks respectively, to reduce the amounts of bits guessed in the key guessing step. The main idea is to test a part of the internal state that will uniquely define a partition of the involved key bits instead of guessing these keybits, reducing, therefore, the complexity of the guess.

During the key guessing part, some internal state words needed for computing $\tilde{P}$ or $\tilde{C}$ are smaller than the key materials that affect them. Thus guessing the state words instead of the key bits involved decreases the time complexity. Indeed, if l key or subkey bits are only needed to compute s bits of internal state with no differences but required to compute some internal state with differences. In such cases, we can guess s bits instead of the l key bits as P (or C) is fixed and this will define a disjoint partition of the involved keybits. Thus the time complexity of this part decreases of 2^{l-s}. We use this technique in both applications of Sec. 6 and Sec. 7.

This idea can be easily applied to differential-MITM attacks, unlike to classical differential attacks, as the plaintext for which we guess the keys is fixed, therefore defining a disjoint partition of the involved keybits.

Analysis of the secret information recovered. As instead of recovering direct bits from the key we can recover non-linear equations, we have to be careful when computing the overall complexity that the number of recovered keybits of the actual key is bigger than the number of candidates for the triplets $(P, \tilde{P}, |k_{in} \cup k_{out}|)$ so that we do not have to deal with counters. We will see how to deal with particular cases in the applications.

5 MILP Modeling of the Truncated Differential-MITM Attack

In this section, we describe the automatic MILP-aided method for searching the truncated differential-MITM attack. We first introduce the modeling of the basic attack. Then, we explain how the state-test and probabilistic key recovery techniques can be incorporated into the model. We leave the inclusion of the parallel partitioning method as an open problem for future work. All source codes are available at https://github.com/CraftSkinny.

5.1 MILP model of the basic attack

The set of constraints used in our model can be divided into three parts: constraints describing the distinguisher, constraints associated to k_{in} and k_{out}, and constraints describing the objective function.

Constraints associated with the distinguisher. This set of constraints is derived according to the method given in [27]. Once the model is solved, the approximated values of transition probabilities will be replaced by the accurate ones, given in the Branching Property Tables (BPT) described in Appendix D of [2]. To be more conservative, we can examine the accurate method given in [17], which computes the exact value of the probability for a given path. Moreover, we develop a distinguisher-only model with an accurate DBT, to compute the clustering effect on the differential probability, by summing up the probabilities of all the paths with $(\Delta_{in}, \Delta_{out})$ fixed to that of the optimum solution.

Constraints determining k_{in} and k_{out}. We explain the method for the identification of the set k_{out}. A similar scenario holds for k_{in}, as well. The set k_{out} is determined by two factors: First, all the subkeys involved in the *differential propagation* of Δ_{out} to the ciphertext, and second, all the subkeys involved in the *value determination* of active words of Δ_{out} from the ciphertext. These two concepts have been previously used and modeled in other works such as automation of MITM attacks [29]. In Appendix B of [2], tunify the description of the MILP modeling of the differential propagation and value determination through the linear layer of a given matrix $\mathbf{M}$ with input and output $\mathbf{a}$ and $\mathbf{b}$, respectively.

These two concepts have been denoted in Fig. 5.1, where the differential propagation and value determination of active words are indicated by $\{D_i\}$ and $\{V_i\}$ chains, respectively. The active words in round r, i.e. D_r only depend on the active words in D_{r-1}, however, the words whose values should be determined in round r, i.e. V_r, depend on both D_{r-1} and V_{r-1}, i.e. on $D_{r-1} \vee V_{r-1}$.

There is a nuanced aspect at the starting point of chain $\{V_i\}$. Note that since the truncated differential attack is not dependent on the concrete values of differences, the attacker avoids the need to guess the subkeys required for value determination just before and after the distinguisher. Moreover, this property

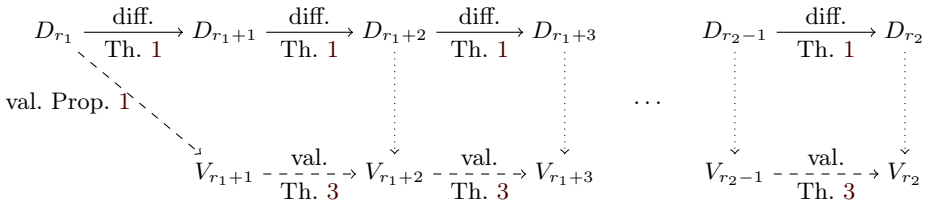

Fig. 5. Differential propagation and value determination in the lower part, where $r_1 = r_{in} + r_m$ and $r_2 = r_{in} + r_m + r_{out}$. The solid arrows show the differential propagation, the dashed ones show the value determination trace and the dotted arrows show the update of V_i by $V_i \vee D_i$, for $r_1 + 2 \leq i \leq r_2$. D_{r_1} corresponds to Δ_{out}.

may partially overspread to the next rounds. For instance, consider a scenario where the differential output of a distinguisher for `Skinny-64-192` is denoted by $\Delta_{out} = (a, b, 0, c)$, and this propagates to $(a + c, a, b, a)$ after the `MixColumn` operation. It's important to note that we do not have to determine the values of all four active words; rather, we only need to determine the values of the second and fourth words, while the other two can remain undetermined. This is because the differences in the second and fourth words are independent of the others, allowing them to have any non-zero value, thanks to the truncated nature of the attack. Property 1 in Appendix B formulates this aspect.

Finally, we have specified the active words, as well as the words whose values need to be determined, over all states of the first r_{in} and the last r_{out} rounds. With this information in place, determining k_{in} and k_{out} becomes straightforward, as it equates to $D \vee V$ at the internal states where the round keys are introduced. All details of the MILP parameters for the two cases studied in this paper, i.e. `CRAFT` and `Skinny-64-192`, is given in Tab. 2 and 3. In both tables, D_i is the input differential state of round $r_1 \leq i \leq r_2$, $D_{r_{in}} = \Delta_{in}$, $D_{r_{in}+r_m} = \Delta_{out}$, $V_{r_{in}} = V_{r_{in}+r_m} = 0$, and $T_i = D_i \vee V_i$. Finally, $T_i[\mathtt{R}_j]$ denotes the j^{th} row of state T_i.

Constraints associated with the objective function. The model should minimize the time complexity given in (3) while preserving the efficient distinguisher constraint given in (9). Let the time complexity be dominated by the integer-valued variable u. the set of constraints associated with these parameters would be defined as follows.

$$\begin{aligned} \min\ & u \\ s.t.\ & p < n - \delta_{out} \\ & p + |k_{in}| < u \\ & p + |k_{out}| + \delta_{out} - \delta_{in} < u \\ & |k_{in} \cup k_{out}| + p + \delta_{out} - n < u \end{aligned} \tag{6}$$

Table 2. `Skinny-64-192` MILP parameters.

	Lower Part		**Upper Part**	
	Parameters	Description	Parameters	Description
Rounds	r_1	$r_{in} + r_m$	r_1	0
$(r_1 \leq i \leq r_2)$	r_2	$r_{in} + r_m + r_{out} - 1$	r_2	r_{in}
Differential	**M**	$\texttt{MC}$	**M**	Inverse $\texttt{MC}$
Propagation	**a**	$\texttt{SR}(D_i)$	**a**	D_i
(Theorem 1)	**b**	D_{i+1}	**b**	$\texttt{SR}(D_{i-1})$
Value	**M**	Inverse $\texttt{MC}$	**M**	$\texttt{MC}$
Determination	**a**	V_{i+1}	**a**	$\texttt{SR}(V_{i-1})$
(Theorem 2, Prop. 1)	**b**	$\texttt{SR}(T_i)$	**b**	T_i
Involved Subkeys	$\lvert k_{out}\rvert$	$\sum_{i=r_1+1}^{r_2} T_i[\texttt{R}_0, \texttt{R}_1]$	$\lvert k_{in}\rvert$	$\sum_{i=r_1}^{r_2-2} T_i[2] + (\bigvee_{j\neq 2} T_i[\texttt{R}_j])$

Table 3. `CRAFT` MILP parameters.

	Lower Part		**Upper Part**	
	Parameters	Description	Parameters	Description
Rounds	r_1	$r_{in} + r_m$	r_1	0
$(r_1 \leq i \leq r_2)$	r_2	$r_{in} + r_m + r_{out} - 1$	r_2	r_{in}
Differential	**M**	$\texttt{MC}$	**M**	Inverse $\texttt{MC}$
Propagation	**a**	D_i	**a**	$\texttt{P}^{-1}(D_i)$
(Theorem 1)	**b**	$\texttt{P}^{-1}(D_{i+1})$	**b**	D_{i-1}
Value	**M**	Inverse $\texttt{MC}$	**M**	$\texttt{MC}$
Determination	**a**	$\texttt{P}^{-1}(V_{i+1})$	**a**	V_{i-1}
(Theorem 2, Prop. 1)	**b**	T_i	**b**	$\texttt{P}^{-1}(T_i)$
Involved subkeys	$\lvert k_{out}\rvert$	$\sum_{i=r_1+1}^{r_2} (\bigvee_{i\ even} \texttt{P}^{-1}(T_i)) + \sum_{i=r_1+1}^{r_2} (\bigvee_{i\ odd} \texttt{P}^{-1}(T_i))$	$\lvert k_{in}\rvert$	$\sum_{i=r_1}^{r_2-2} (\bigvee_{i\ even} (T_i)) + \sum_{i=r_1}^{r_2-2} (\bigvee_{i\ odd} (T_i))$

5.2 MILP model of the improved attack

State-test enhanced attack. In this section, we propose a general method for MILP modeling of the state-test technique introduced in Sec. 4.3, for reducing the guessed key material on each side. This technique specifically influences the constraints related to the value determination (Theorem 2), in such a way that if a word of the internal state is preferred to be guessed, the value-determination trace corresponding only to this specific word should be aborted, then. To MILP model this event, for each state word whose value is supposed to be determined, we define a new binary variable s, indicating whether the corresponding word should undergo the state-test ($s = 1$) or not ($s = 0$). Then, the value-determination constraints would be as regular (Theorem 2) if $s = 0$ or aborted if $s = 1$. Theorem 3, identifies the required constraints corresponding to the value-determination for state-test enhanced attack.

Finally, in the objective function constraints (6), $|k_{in}|$ should be replaced by $|k_{in}| + s_{in}$ where $s_{in} = \sum_{0\leq i\leq r_{in}-2} Hw(\mathbf{s}_i)$, and $|k_{out}|$ should be replaced by $|k_{out}| + s_{out}$ where $s_{out} = \sum_{r_{in}+r_m+1\leq i\leq r_{in}+r_m+r_{out}-1} Hw(\mathbf{s}_i)$.

Probabilistic key recovery enhanced attack. In order to incorporate this technique into the model, it suffices to replace the differential propagation constraints generated according to Theorem 1 by the MILP model of the probabilistic truncated differential propagation given in [27]. In the upper part, this model is used for the inverse of `MixColumn` matrix as $\mathbf{M}$, and in the lower part, with `MixColumn` matrix. Then, the constraint $p + |k_{in}| < u$ of (6) should be modified as $p + p_{out} + |k_{in}| < u$, and $p + |k_{out}| + \delta_{out} - \delta_{in} < u$ should be modified to $p + p_{in} + |k_{out}| + \delta_{out} - \delta_{in} < u$.

6 Application on 23-round `CRAFT`

`CRAFT` is a lightweight, tweakable block cipher designed with the goal of protecting implementations against differential fault analysis while also providing strong security guarantees within the related-tweak model. The specification of this cipher is provided in Appendix C.1.

Security claim. In [9], the authors presented optimum differentials for 13 and 14 rounds of `CRAFT` and claimed that using those differentials the attacker cannot have a successful single-tweak differential attack on 22 rounds. The best previous known attack on `CRAFT` [19] is a 21-round impossible differential attack with time complexity of $2^{106.53}$, data complexity of $2^{60.99}$ and memory complexity of 2^{100}.

Using truncated differential-MITM, enhanced with parallel partitioning for whole-state key addition ciphers, probabilistic key guessing, and the state-test techniques we managed to provide the best attack on `CRAFT` [9], improving by 2 rounds the best known attack, as detailed in Tab. 1.

6.1 An attack on 23 rounds of `CRAFT`

The truncated differential-MITM attack proposed in this section is composed of a 22-round core attack followed by one additional round using the parallel partitioning method. We conducted an automatic search for the optimum 22-round core attack on `CRAFT`, enhanced with the state-test and probabilistic key guessing techniques, based on the MILP method proposed in Sec. 5. We set $r_m = 11$, $r_{in} = 6$, and $r_{out} = 5$. The 11-round distinguisher and the core 22-round attack are represented in Figures 6 and 7 respectively, with the following parameters:

p	p_{in}	p_{out}	s_{in}	s_{out}	δ_{in}	δ_{out}	$\lvert k_{in}\rvert$	$\lvert k_{out}\rvert$
44	16	12	16	12	16	16	32	32

where s_{in} and s_{out} is the number of state-test bits to guess. The clustering effect was also examined which has a negligible impact. Since the matrix M used in the `MixColumn` operation is involutory, and the input and output differences of the distinguisher are the same, the propagation of active nibbles in the upper and lower parts is very similar. Thus, as shown in Tab. 4, the subkey words needed on both parts have many words in common thanks to the key schedule

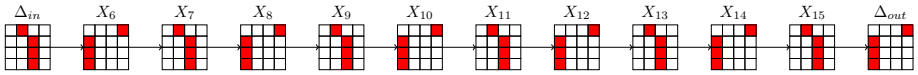

Fig. 6. CRAFT 11-round truncated differential distinguisher with $p = 44$

Subkey	k_{in}	k_{out}	$\lvert k_{in} \cap k_{out} \rvert$
Even	$K_0[1, 6, 10, 14, 15]$	$K_0[6, 10, 14]$	3 nibbles
Odd	$K_1[4, 8, 12]$	$K_1[4, 8, 12, 13, 3 \oplus 11 \oplus 15]$	3 nibbles

Table 4. Subkey nibbles guessed during the 23-round attack of `CRAFT`.

and the position of the odd and even subkeys, which allows us to efficiently apply the parallel partitioning technique. The parallel key guessing steps of this attack benefit from the state-test technique from Sec. 4.3 and the probabilistic key recovery technique of Sec. 4.2.

Probabilistic key recovery technique. It is imposed that the difference during the `MixColumn` transition on nibbles $X_2[2], X_3[0], X_4[2], X_5[0], Y_{17}[0], Y_{18}[2]$ and $Y_{19}[0]$ is null. This decreases the overall probability by a factor of $2^{p_{in}+p_{out}} = 2^{16+12}$, but the number of active words, and consequently $|k_{in}|$ and $|k_{out}|$, also decreases dramatically. This attack is easy to adapt to 21 rounds, as we will show in Sec 6.2.

State-test technique We use the state-test technique to test four and three nibbles of the internal state in the upper and lower parts respectively, which are given in Appendix E. This avoids guessing more subkey nibbles.

Parallel guessing step. From a pair (P, Y_{22}), we need to know the values of the active nibbles before the S-boxes, in green in Fig. 7, to be able to compute the associated value of $\tilde{P}$ for each k_{in} and of $\tilde{Y}_{22}$ for each k_{out}. We will first describe the procedure over the 22 first rounds, and then show how to deal with the last round with parallel partitions.

1. We first guess the $6 \times 4 = 24$ subkey bits of $(k_{in} \cap k_{out})$ given in Tab. 4. We then fix $F = 5$ nibbles $Y_{22}[1, 6, 10, 14, 15]$, the words denoted by `F` in Fig. 8 and since we know the subkey nibbles $K_{22}[6, 10, 14]$ for both the upper and lower parts, and of $K_{22}[1, 15]$ for the upper part, then we also have the fixed values of the ciphertext $C[1, 6, 10, 14, 15]$.
2. As 5 nibbles from C and Y_{22} are fixed, we can build a pair of structures of size $2^{4(16-5)} = 2^{44}$ for Y_{22} and for the ciphertext C. We decrypt the 2^{44} ciphertexts C_s and we obtain the associated plaintexts P_s.
3. For each of the $2^{|k_{in}-k_{in}\cap k_{out}|} \times 2^{s_{in}} = 2^{8+16} = 2^{24}$ possible values i of k_{in} and the state-test relations, and each of the 2^{44} P_s from the structure defined at the previous step, compute all possible tuples $(P_s, \tilde{P}_s, i)$ such that they generate Δ_{in} after 6 rounds and they follow the probabilistic differential

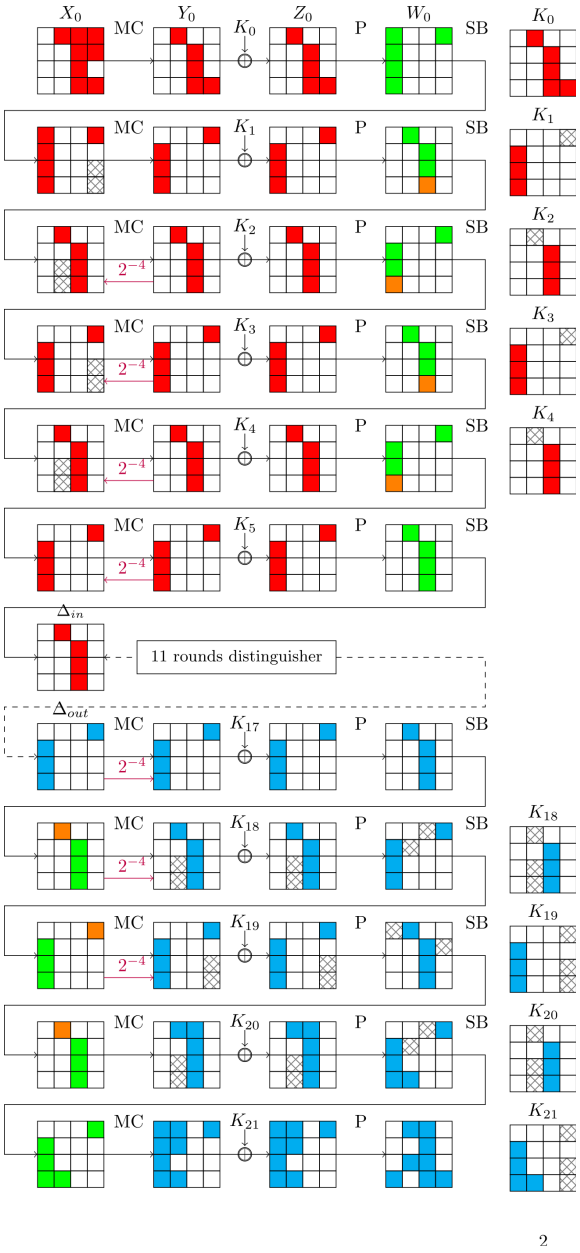

Fig. 7. The 22-round core part of the 23-round attack on `CRAFT` not including the structures. Differential propagation in the upper (lower) part has been shown in red (blue). The state-test nibble is shown in orange. The gray-striped nibbles are whose values are no longer required thanks to the state-tests, except K_{21} that the XOR of gray-striped nibbles are required.

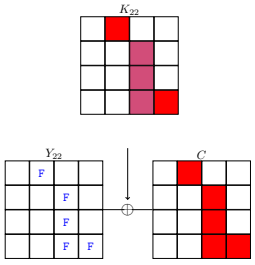

Fig. 8. The last round of the 23-round attack on `CRAFT`, using parallel partitioning technique. The red nibbles are known or can be computed from the fixed values and k_{in} in the upper part of the attack and the blue nibbles are known or can be computed from the fixed values in the lower part of the attack as $Y_{22} = \texttt{MC}(\texttt{SB}(W_{21}))$. The purple subkey nibbles are known in both the upper and lower parts.

transitions in the first rounds. There are 2^{24} possible values for k_{in}, and it needs to be done for each $2^{\delta_{in}} = 2^{16}$ possible differences in Δ_{in}, but we expect only a proportion of $2^{-p_{in}} = 2^{-16}$ to verify the upward path, so the expected number of total solutions per P_s will be $2^{24+16-16}$, and if we use rebound-like techniques as shown in the original paper [12] the cost of this step would be equal to the number of solutions: $2^{44+24} = 2^{68}$. We store them in a hash table.

4. Similarly, for each of the $2^{|k_{out}-k_{in}\cap k_{out}|} \times 2^{s_{out}} = 2^{8+12} = 2^{20}$ possible values j of k_{out} and the state-test relations, and each of the 2^{44} states $Y_{22,s}$, compute all possible tuples $(Y_{22}, \tilde{Y}_{22}, j)$ such that there is the Δ_{out} difference on the state before the 17th S-box layer. It needs to be done for each $2^{\delta_{out}} = 2^{16}$ possible difference in Δ_{out} but we expect only a proportion of $2^{-p_{out}} = 2^{-12}$ to verify the downward path, so the expected number of total solutions per $Y_{22,s}$ will be $2^{20+16-12}$. As the previous step, we expect a cost and a number of solutions of $2^{44+24} = 2^{68}$. For each solution, check for possible matches on the hash table. The match is performed on two quantities:
   - The values of $\tilde{Y}_{22}[1, 6, 10, 14, 15]$ can be fully computed from $\tilde{C}[1, 6, 10, 14, 15]$ and the guessed k_{in}: 20-bit filter;
   - The linear relations between Y_{22} and C and between $\tilde{Y}_{22}$ and $\tilde{C}$, i.e. $Y_{22} \oplus C = \tilde{Y}_{22} \oplus \tilde{C}$, excluding the nibbles we could fully compute as they were already used in the 20 bit filter: 44-bit filter (11 equations on 4 bits each);
5. Repeat from Step 1 until the right key is retrieved: as the structures check 2^{44} plaintexts, we need to repeat all this procedure $2^{44-16+16+12-44} = 2^{12}$, to expect having a pair of data that verifies the middle distinguisher and the external transitions 2^{-56}.

The data complexity of the attack is 2^{64} as we ask the whole codebook to the oracle. But, applying the improved data technique from [12], we can fix

$\frac{64-56}{2} = 4$ bits of the ciphertexts to reduce the data needed without increasing the time complexity, hence the data complexity will become $\mathcal{D} = 2^{60}$.

The memory complexity is determined by Step 3 where 2^{68} words of $64 \times 2 + 16 + 8 = 152$ bits each are stored in the hash table, so $\mathcal{M} = 2^{68}$.

The time complexity so far is

$$2^{12}2^{24}\left(2^{44}2^{24}2^{16-16} + 2^{44}2^{20}2^{16-12} + 2^{68+68-20-44}\right) = 2^{108}.$$

But the attack is not yet finished. Indeed, we have recovered $5 \times 4 = 20$ bits of K_1, as well as 64 bits of K_0, as the unguessed bits of $K_0 = K_{22}$ would be revealed as a side effect of the second sieving, explained in Step. 4. In addition, we recover $7 \times 4 = 28$ bits of information in key bits due to the 7 nibbles of state-tests, so $64 + 20 + 28 = 112$ information-bits of the key in total, which is bigger than the number of candidates, i.e. 108. The big question now is how to determine the whole key from each candidate because of the complex form of the state test equations.

How to recover the whole key. The whole key K_0 is known. Considering this and rewriting the state-test equations given in Appendix E, we recover the following values and relations. From the first equation we can derive the value of $K_1[3]$, giving us $K_1[11] \oplus K_1[15]$ from the guesses, and we choose to write $K_1[15]$ as a function of $K_1[11]$. Then we rewrite the equations given on rounds 4 and 18 as a function of some variables $x_1, \ldots, x_{24}$ which depend only on the plaintext, the ciphertext, the guessed values of the state tests, the nibbles of K_0 and those nibbles of K_1 from k_{in} and k_{out}. We obtain the following equations:

$$\begin{aligned}
\text{Equation 4} \ : & SB(K_1[2] \oplus SB(SB(K_1[0] \oplus x_1) \oplus SB(K_1[14] \oplus x_2) \oplus SB(K_1[7] \\
& \oplus x_3) \oplus x_4) \oplus SB(SB(K_1[1] \oplus x_5) \oplus SB(K_1[10] \oplus x_6) \oplus x_7) \\
& \oplus SB(SB(K_1[2] \oplus x_8) \oplus x_9)) \oplus SB(K_1[5] \oplus SB(SB(K_1[5] \oplus x_{10}) \\
& \oplus x_{11}) \oplus x_{12}) \oplus x_{13} = 0, \\
\text{Equation 18} \ : & SB(K_1[2] \oplus K_1[10] \oplus K_1[14] \oplus SB(SB(K_1[7] \oplus K_1[11] \oplus x_{14}) \\
& \oplus SB(SB(K_1[0] \oplus x_{15}) \oplus SB(K_1[14] \oplus x_{16}) \oplus x_{17})) \oplus SB(SB(K_1[1] \oplus K_1[9] \\
& \oplus x_{18}) \oplus SB(K_1[10] \oplus x_{19}) \oplus x_{20}) \oplus SB(SB(K_1[2] \oplus K_1[10] \oplus K_1[14] \\
& \oplus x_{21}) \oplus x_9) \oplus SB(K_1[5] \oplus SB(SB(K_1[5] \oplus x_{22}) \oplus x_{11}) \oplus x_{23}) \oplus x_{24} = 0.
\end{aligned}$$

During the attack procedure, we stock the candidates we find after the matching in a table of size 2^s with $s \geq 100$, and sort this table based on $x_1, \ldots, x_{24}$. We will have 2^{96} groups of candidates of size $2^x = 2^{s-96}$ with the same variables. Since $x_1, \cdots, x_{24}$ define equations 4 and 18, each candidate in one group will have the same set of solutions for those two equations. Thus, for each group, we can calculate and store the list of the 2^{20} solutions for equation 4 and 18. And in parallel, for each of the 2^x candidates in the group, we calculate the 2^{16} possible solutions of equations 3 and 19. For each of these 2^{16} solutions, we get one match with the stored list of 2^{20} solutions from equation 4 and 18 with respect to the nibbles $K_1[0, 1, 9, 11, 14]$. Thus, for each element in the group, we can find 2^{16}

possibilities for $K_1[0,1,2,5,6,7,9,10,11,14,15]$ giving us the whole key. Finally, the overall time cost of this will be:

$$\mathcal{T} = 2^{108-s} 2^{96} \left(2^{20} + 2^x 2^{16}\right),$$

which equals to 2^{124} if $x > 4$ and if $x = 4$ it equals to 2^{125} of small computations for recovering the whole key, which is lower than exhaustive search. And the memory complexity is 2^s as we stock the candidates in a table of size 2^s. For $s = 101$, we get a time complexity of $2^{124.58}$, and many trade-offs are possible.

6.2 Other Attacks on CRAFT and conclusion

The attack described above can be applied to fewer than 23 rounds straightforwardly by subsequently removing one round from each side and adapting the structures to the known key nibbles. These results can be seen in Table 1. In this case, the data will be smaller, as we can apply the data reduction idea from [12].

It is worth pointing out that the authors did not expect differential attacks to reach 22 rounds with the best paths they found. Given that these paths reached 13 rounds, and the distinguisher used in our attack reaches 11 (two less rounds), it makes us deduce that truncated MITM attacks seem to be much more performant on CRAFT than differential attacks, and the most performant attack, to the best of our knowledge. We believe this is the case because of its alternated key schedule and the existence of iterative truncated paths, both with period two. The number of rounds reached is still far from the full version, but we expect further attacks with these techniques and better dealing with the state-test relations to reach more rounds.

7 Applications: SKINNY-64-192 and SKINNY-128-384

In this section, we provide two applications on two variants of SKINNY. Using a truncated path, state-test technique and the enhanced parallel partitioning method over two rounds, we provide a new attack on 23-round SKINNY-64-192, with slightly better time and lower data than the best known attack. In order to illustrate the importance of our improved parallel partitioning method, we have improved the previous differential MITM attack on SKINNY-128-384, that provided the best attacks on the single tweakey setting, and we have managed to slightly reduce their time or data complexity thanks to the structures, providing the best current attack on SKINNY-128-384 in the single tweakey setting. The specification of this cipher is given in C.2. In the proposed attacks, we use the modified round key addition in the upper part where $U_r = \texttt{MC}(\texttt{SR}(K_r))$ is Xored to the output state of the MixColumn operation. This shows the fact that (truncated) differential MITM attacks work well on reduced-round variants of the SKINNY constructions. As further work, we plan to automatize the tool including more evolved structures, as the ones used in the MITM attacks from [3,4,11], and we expect we might be able to reach more rounds in both variants.

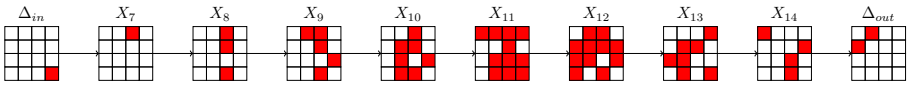

Fig. 9. 9-round truncated differential characteristic of probability 2^{52} for SKINNY-64-192.

7.1 Attack on 23-round SKINNY-64-192

Since SKINNY key schedule is linear, it would be an efficient approach to guess some subkey bits and retrieve the whole key after guessing enough independent round key bits. Moreover, the key schedule makes the evaluation of the dimension of any set of round key nibbles easy since a round key nibble depends on exactly three master key nibbles, $\mathbf{TK1}[i], \mathbf{TK2}[i]$, and $\mathbf{TK3}[i]$, for a specific $i \in \{0, \ldots, 15\}$.

An attack on SKINNY-64-192 without parallel partitioning. We first propose a 21-round truncated differential-MITM attack which is followed by two additional rounds using the parallel partitioning method, resulting in a 23-round attack. We conducted an automatic search for the optimum 21-round core attack on SKINNY-64-192, enhanced with the state-test key guessing technique, based on the MILP method proposed in Sec. 5. We set $r_m = 9$, $r_{in} = 6$, and $r_{out} = 6$. The core attack is represented in Fig. 14 in Appendix F, with the following parameters:

Rounds	p	s_{in}	s_{out}	δ_{in}	δ_{out}	$\lvert k_{in}\rvert$	$\lvert k_{out}\rvert$
22	52	4	4	4	8	128	116

The clustering effect increases p to 51.78, which is not very significant. Although we have included the probabilistic key recovery method in our search, the optimum solution returned $p_{in} = p_{out} = 0$, meaning that a deterministic key guessing is more efficient for SKINNY-64-192.

Tab. 9 in Appendix F describes all the subkey nibbles of k_{in} and k_{out} needed in the 21-round attack of SKINNY-64-192 which are also reflected in Fig. 14 (this information is also included in the needed key nibbles for the 23-round full attack in Table 10). It also indicates which nibble of the master key each subkey nibble of k_{in} and k_{out} depends on, and presents the total number of linear relations in each nibble of the master key, given k_{in} and k_{out}. In this way, we can determine the number of common linear relations, i.e. the size of the intersection $|k_{in} \cap k_{out}|$ which is $15 \times 4 = 60$ bits.

State-test technique. We could use the state-test technique to reduce $|k_{in}|$ and $|k_{out}|$ by testing the 3 and 2 respective nibbles of Tab. 5, instead of guessing the 5 and 4 respective subkeys nibbles described on the table. Thanks to this technique, the number of bits in k_{in} could be reduced by 8 bits, and the number

of bits in k_{out} could also be reduced by 8 bits. The optimal time complexity is nevertheless reached when we only consider one state-test technique for each part (the non-crossed ones in Tab. 5).

Wanted nibble	RoundKey nibbles involved	Nibbles needed from the precedent state
~~$X_4[2]$~~	$K_3[2]$, $K_2[7]$	$K_3[2] \oplus \mathtt{SC}(X_3[2]) \oplus \mathtt{SC}(X_3[15]) \oplus \mathtt{SC}(K_2[7] \oplus \mathtt{SC}(X_2[7]) \oplus \mathtt{SC}(X_2[10]))$
~~$X_4[10]$~~	$K_3[5]$, $K_2[7]$	$K_3[5] \oplus \mathtt{SC}(X_3[5]) \oplus \mathtt{SC}(K_2[7] \oplus \mathtt{SC}(X_2[7]) \oplus \mathtt{SC}(X_2[10]))$
$X_5[9]$	$K_4[4]$, $K_3[6]$	$K_4[4] \oplus \mathtt{SC}(X_4[4]) \oplus \mathtt{SC}(K_3[6] \oplus \mathtt{SC}(X_3[6]) \oplus \mathtt{SC}(X_3[9]))$
~~$X_{17}[5]$~~	$K_{17}[5]$,$K_{18}[6]$	$K_{17}[5] \oplus \mathtt{SC}^{-1}(X_{18}[10]) \oplus \mathtt{SC}^{-1}(X_{18}[14]) \oplus \mathtt{SC}^{-1}(K_{18}[6] \oplus \mathtt{SC}^{-1}(X_{19}[7]) \oplus \mathtt{SC}^{-1}(X_{19}[11]) \oplus \mathtt{SC}^{-1}(X_{19}[15]))$
$X_{16}[5]$	$K_{16}[5]$,$K_{17}[6]$	$K_{16}[5] \oplus \mathtt{SC}^{-1}(X_{17}[10]) \oplus \mathtt{SC}^{-1}(X_{17}[14]) \oplus \mathtt{SC}^{-1}(K_{17}[6] \oplus \mathtt{SC}^{-1}(X_{18}[7]) \oplus \mathtt{SC}^{-1}(X_{18}[11]) \oplus \mathtt{SC}^{-1}(X_{18}[15]))$

Table 5. Non-linear relations available in the state-test technique on the 21 and 23 rounds of `SKINNY-64-192`. The crossed cells will not be used in the 23-round attack.

Attack steps We describe now the core attack on 21 rounds of `SKINNY-64-192`. The guesses needed for this attack are given in Tab. 9 and the state test equations to guess are given in Tab. 5.

1. Ask for the encryption of the whole codebook (we will explain later how to apply the data reduction of Sec. 2.3 to this case).
2. Pick one plaintext/ciphertext pair (P, C).
3. First we guess the 44 subkey bit common relations shared between k_{in} and k_{out} given by Tab. 9.
4. Compute all possible tuples $(P, \tilde{P}, i)$ for each value i of the remaining 84 bits of k_{in} such that the difference after the 6th S-box layer is according to Δ_{in} of Fig. 9. At the end of this step, we have 2^{84+4} possible candidates. For all $\tilde{P}$, compute $E(\tilde{P}) = \hat{C}$ and store them in a hash table.
5. Similarly, for each value j of the remaining 72 bits of k_{out}, compute all possible tuples $(C, \tilde{C}, j)$ so that the difference on the state before the 15th S-box layer is according to Δ_{out} of Fig. 9. At the end of this step, we have 2^{72+8} possible candidates for the tuples $(C, \tilde{C}, j)$. And check for possible matches on the hash table. The match is performed on both the new ciphertext $\hat{C}$ and $\tilde{C}$ so that ($\tilde{P}$,$\tilde{C}$) is a valid plaintext/ciphertext pair.
6. Repeat from Step 1 until the right key is retrieved.

Adjustment for the number of candidate triplets. If we consider what complexity such an attack would have, we obtain:

$$2^{52-4}2^{44}\left(2^{128-44+4} + 2^{116-44+8} + 2^{168-64}\right) = 2^{196},$$

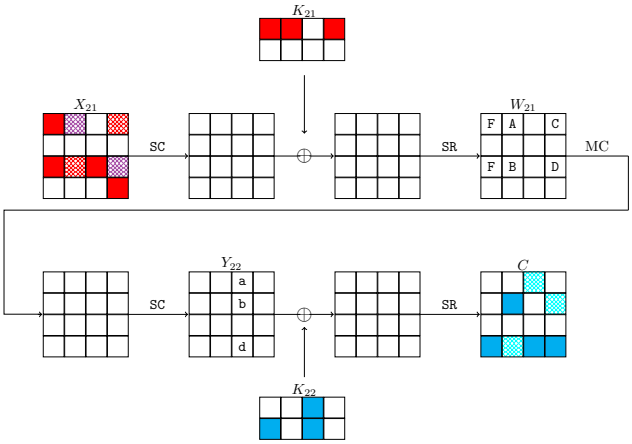

Fig. 10. The two last rounds of the 23-round attack on `SKINNY-64-192`, using parallel partitioning technique with fixed values for $X[1] \oplus X[11]$, $X[3] \oplus X[9]$, $Y[2] \oplus Y[13]$ and $Y[7] \oplus Y[13]$.

that exceeds the exhaustive search.

This is a possible side effect of the state test technique, that allows less sieving regarding the keybits. To compensate this, we will consider less state-test equations: we will guess in addition $K_2[7]$, that will determine because of the two first state-test equations $K_3[2]$ and $K_3[5]$; and also $K_{18}[6]$ and $K_{17}[6]$ that will determine because of the fourth and fifth state test equations $K_{17}[5]$ and $K_{16}[5]$. These three words guesses provide an additional sieving of 6 words, so 2^{-24}. The complexity will then be below exhaustive search:

$$2^{52-4}2^{68}\left(2^{128+4-68+4} + 2^{116+8-68+8} + 2^{132-64}\right) = 2^{184}.$$

Attack on 23-round of SKINNY-64-192 Now we will explain how to extend for two rounds this 21-round attack thanks to the improvement with the parallel partitioning method. In addition to the 21-round attack, we guess words $K_{22}[6]$, $K_{21}[0]$, $K_{21}[3]$ and $K_{21}[1]$, as they will be needed to sieve with respect to all the available potential during the parallel partitioning. In Fig. 10, we give the scheme of the two last rounds of the attack. We represent in red the internal state and the subkey words that we will use to build the parallel partitions and compute the upper part of the attack. Similarly, we represent in blue the internal state and the subkey words used to build the initial structure and compute the lower part of the attack. The remaining attack procedure of the 23-round attack is similar to the 21-round one:

1. We fix the values of the words F, $A \oplus B$ and $C \oplus D$ of W_{21} in Fig. 10, and of the words $a \oplus d$ and $b \oplus d$ of Y_{22} in Fig. 10.

2. We guess the 76 bits of linear common relations of $(k_{in} \cap k_{out})$ given in Tab. 10.
3. Then for each value i of the remaining 56+4=60 bits of $(k_{in} - k_{in} \cap k_{out})$, we can compute from the values of step 1, the fixed nibbles $C[5, 12, 14, 15]$, $C[2] \oplus C[13]$ and $C[7] \oplus C[13]$ as shown in Fig. 10. Then we have 2^{40} possible states for C as we have imposed the values of 6 nibbles thus there are 10 nibbles that can take all possible values. For all the possible values for C, we compute the ciphertext and get the corresponding plaintext P.
4. Compute all possible tuples $(P, \tilde{P}, i)$ for each $2^{|k_{in} - k_{in} \cap k_{out}|} \times 2^{s_{in}} = 2^{56+4}$ values i of $k_{in} - |k_{in} \cap k_{out}|$ and the state test relation, and each P from the structure defined at the previous step, such that they generate Δ_{in} after six rounds. At the end of this step, we have $2^{40} \times 2^{60+4} = 2^{104}$ possible candidates, and store them in a hash table.
5. Similarly, for each of the $2^{|k_{out} - k_{in} \cap k_{out}|} \times 2^{s_{out}} = 2^{52+4}$ possible values j of $(k_{out} - k_{in} \cap k_{out})$ and state-test relations, we can compute from the values of step 1, the fixed values $X_{21}[0, 8, 10, 15]$, $X_{21}[1] \oplus X_{21}[11]$ and $X_{21}[3] \oplus X_{21}[9]$, in blue in Fig. 10. We then pick all the 2^{40} possible states of X_{21} and we compute the 2^{40} possible $W_{20} = \mathtt{MC}^{-1}(X_{21})$.
6. For each $2^{52+4} = 2^{56}$ values j of $(k_{out} - k_{in} \cap k_{out})$ and state-test relation and for each value of state W_{20}, compute all possible tuples $(W_{20}, \tilde{W}_{20}, j)$ so that there is a difference on the state before the 15th S-box layer on both nibbles. At the end of this step we have $2^{40} \times 2^{56+8} = 2^{104}$ possible candidates for the tuples $(W_{20}, \tilde{W}_{20}, j)$.
7. Check for possible matches on the hash table. The match is performed on two quantities:
   - The values of the nibbles $\tilde{X}_{21}[8]$ and $\tilde{X}_{21}[15]$ can be fully computed from $\tilde{C}[2] \oplus \tilde{C}[13]$,$\tilde{C}[7] \oplus \tilde{C}[13]$ and the guessed k_{in} and similarly the values of the nibbles $\tilde{C}[5]$,$\tilde{C}[12]$, $\tilde{C}[14]$ and $\tilde{C}[15]$ can be fully computed from $\tilde{X}_{21}[0]$, $\tilde{X}_{21}[10]$, $\tilde{X}_{21}[1] \oplus \tilde{X}_{21}[11]$, $\tilde{X}_{21}[3] \oplus \tilde{X}_{21}[9]$ and the guessed k_{out}: a $6 \times 4 = 24$ bit filter;
   - The linear relations between X_{21} and C and between $\tilde{X}_{21}$ and $\tilde{C}$: a 80-bit filter (10×2 equations on 4 bits each), in Table 11 from supplementary material.
8. Repeat from Step 1 with different values for the fixed nibbles until the right key is retrieved.

Linear relations to match with the parallel partitioning improvement. At the end of the attack procedure we have to match X_{21} and C and their associated pair $\tilde{X}_{21}$ and $\tilde{C}$. After two rounds of `SKINNY-64-192`, if we know the 4 first subkey nibbles then the relations between the words of X_{21} and C and $\tilde{X}_{21}$ and $\tilde{C}$, of Fig. 10, are linear. In our application, we have guessed the subkey nibbles $K_{21}[0]$,$K_{21}[1]$ and $K_{21}[3]$. We do not know the subkey nibble $K_{21}[2]$ but we have guessed the subkey nibbles $K_{22}[2]$ and $K_{22}[6]$ and thus we obtain the linear relations of Column 3. To match both sides of the equations when the subkey word is not completely determined, we add on each side the subkey information known by each side, respectively. Thus the relations are linear as we can compute

each side of the equations independently. The linear relations (given for the pair (X_{21},C), but that are also true for $(\tilde{X}_{21},\tilde{C})$) to match are given in Tab. 11 of Appendix F.

The time complexity of this step will be:

$$\mathcal{T} = 2^{52-4-40}2^{76}\left(2^{56+4+4}2^{40} + 2^{52+4+8}2^{40} + 2^{104+104-24-40-40}\right) = 2^{188}.$$

To reduce the data complexity of our attack, we use the improvement of [12] to impose the value of $\frac{64-48}{2} = 8$ bits of the ciphertexts. Thus we fix the nibbles $C[6]$, $C[1] \oplus C[13]$, $\tilde{C}[6]$ and $\tilde{C}[1] \oplus \tilde{C}[13]$, which is a 8 bits condition. Moreover, we compute and build a stored table with the data needed to perform the attack to avoid doing the decryption during the upper part of the attack. Finally the data complexity of this attack is $\mathcal{D} = 2^{64-8} = 2^{56}$. The memory complexity is determined by Step 6 storing $\mathcal{M} = 2^{104}$ words of $64 + 64 = 128$.

Thanks to the truncated differential, we manage to extend the characteristic for more rounds than if it was a concrete differential characteristic since the subkeys around the characteristic are not needed in the attack. Moreover in the case of `SKINNY`, we can use, with little cost, the parallel partitioning improvement to reach two more rounds and thus reach the same best number of rounds in the single tweak setting as the best known attack.

7.2 Improved attacks on 25-round `SKINNY-128-384`

We consider the 24-round and the 25-round attack given in [12] in section 3.4 and 3.5 respectively. Each uses a different differential. By considering the core attack on 23 rounds that is extended by one round in the paper, we will apply our improved structure with a two-round extension, reducing the data complexity of the best known attacks, that cover 25 rounds. By considering the 25-round attack also considered in the paper, and including the last key recovery round in the final structure, reaching now 2 rounds instead of 1, we are able to reduce the overall time complexity, as we have fewer key bits to guess, providing the 25-round `SKINNY-128-384` attacks with the lowest complexity. These results are shown in Table 1, compared to all the previous results.

Reducing the data needed. The application is quite straightforward given the previous attack, so we will omit the details here and just provide the image representing the attack in appendix G.

We consider exactly the same core over 23 rounds as in the attack from [12], but we will add two rounds instead of one. For this, we will additionally guess nibbles 5, 2, and 4 from K_{24} in k_{in}, and nibbles 14 and 11 from K_{23} in K_{out}. All these guesses add equations in the common part of the key to guess. This will allow us to build structures of size 2^{64}, with 2 fixed words from the first and the last columns, 3 from the second, and one from the third – see Fig. 15. The full filtering can be applied as we can rewrite all equations as linear ones in the unknown parts of the key. The time complexity becomes:

$$\mathcal{T} = 2^{105.9-64}2^{120+40}\left(2^{128-16}2^{64} + 2^{136-24}2^{64} + 2^{128-16+64+136-24+64-128-64}\right)$$

So we have $\mathcal{T} = 2^{201.9}\left(2^{176} + 2^{176} + 2^{160}\right) = 2^{378.9}$, and the memory complexity, with $x = (128 - 105.9)/2 = 11.05$ becomes $2^{176-11.05} = 2^{165}$, and data $\mathcal{D} = 2^{128-11.05} = 2^{117}$, providing the best data complexity for a 25-round attack, as can be seen in Tab. 1.

Reducing the time. We consider the attack from [12] on 25 rounds, which added 4 rounds at the top and 5 rounds at the bottom of a 15-round distinguisher, plus one added through a structure. As the bottleneck term is the five rounds at the lower part, where we guess 8 more keybits than the upper part, we propose our attack with a 2-round structure, yielding a configuration of 4+15+4+2 rounds. As we are guessing 1 less word from k_{out} (all but the word 12 from K_{23}) than before, as we add 7 words needed in order to verify linearly the equations in the structure and to build small structures, all the complexities stay the same, but the time reduces of about a factor of 2^8 as shown in Tab. 1. We can build now 4 fixed relations between the input and the output of the structure, given by equations $A_1 + C_1$, $A_2 + C_2$, $A_3 + C_3$, $A_4 + C_4$, $B_1 + C_1$, $B_2 + C_2$, $B_3 + C_3$ from Fig. 16, and the structures will have a size of 2^{72}. We do not guess the word 12 from round K_{23} anymore, which reduces by 1 word the lower guess and leaves the number of common keybits the same. The time complexity becomes:

$$\mathcal{T} = 2^{116.5-72}2^{120}\left(2^{128+72} + 2^{136-64+56+72} + 2^{128+72+136-64+56+72-72-128}\right),$$

giving $\mathcal{T} = 2^{164.5}\left(2^{200} + 2^{200} + 2^{200}\right) = 2^{366}$, while previous best time was $2^{372.5}$. Applying the same idea as in the previous attack for reducing the data and memory needed, we obtain, without modifying the time and memory complexities a data of also $2^{128-x} = 2^{122.25}$, as $x = \frac{128-116.5}{2} = 5.75$, with a memory complexity of $2^{194.25}$, that stays the same.

8 Conclusion

We have implemented a tool, based on MILP modeling, that finds the distinguishers that produce the best overall attack complexities when considering the key-guessing rounds and their two related improvements. The inclusion of the structures is left as an open problem for the future.

We have been able to apply the variety of results to `CRAFT`, `SKINNY-64-192` and `SKINNY-128-384`. For `CRAFT` we managed to improve by two rounds the previous attacks with a truncated version of differential MTIM. For `SKINNY-128-384` we managed to improve the complexities of the best known attacks in the single tweak setting, and for `SKINNY-64-192` we matched the same number of rounds while the time stays comparable and the memory and data are worse.

We have in particular shown that differential MITM attacks have a different nature than differential attacks that allow them to be combined with MITM-related techniques that can not be combined with differential attacks, like the parallel partitions that are closely related to initial structures and bicliques. Actually, we leave as an open problem to produce a tool that will combine differential MITM attacks with parallel partitioning technique [5] and bicliques [20] over more rounds, as the one in [15].

In addition, we showed that differential MITM attacks can be easily combined with the state-test technique, thanks to the fact that each key is tested for a fixed data. In differentials attacks it would be much harder to apply.

Acknowledgements. This research has been partially funded by the European Union (ERC-2023-COG, SoBaSyC, 101125450).

References

1. Ahmadian, Z., Khalesi, A., M'foukh, D., Moghimi, H., Naya-Plasencia, M.: Truncated differential attacks: New insights and 10-round attacks on qarma. Cryptology ePrint Archive (2023)
2. Ahmadian, Z., Khalesi, A., M'foukh, D., Moghimi, H., Naya-Plasencia, M.: Improved differential meet-in-the-middle cryptanalysis. Cryptology ePrint Archive, Paper 2024/351 (2024), https://eprint.iacr.org/2024/351, https://eprint.iacr.org/2024/351
3. Aoki, K., Sasaki, Y.: Preimage attacks on one-block md4, 63-step MD5 and more. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers. Lecture Notes in Computer Science, vol. 5381, pp. 103–119. Springer (2008)
4. Aoki, K., Sasaki, Y.: Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 70–89. Springer (2009)
5. Aoki, K., Sasaki, Y.: Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5677, pp. 70–89. Springer (2009)
6. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 321–345. Springer (2017)
7. Bao, Z., Guo, J., Shi, D., Tu, Y.: Superposition meet-in-the-middle attacks: Updates on fundamental security of aes-like hashing. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022,

Proceedings, Part I. Lecture Notes in Computer Science, vol. 13507, pp. 64–93. Springer (2022)
8. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016)
9. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol. **2019**(1), 5–45 (2019)
10. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer (1990)
11. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. Lecture Notes in Computer Science, vol. 7073, pp. 344–371. Springer (2011)
12. Boura, C., David, N., Derbez, P., Leander, G., Naya-Plasencia, M.: Differential meet-in-the-middle cryptanalysis. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 240–272. Springer (2023)
13. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: Applications to clefia, camellia, lblock and simon. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 179–199. Springer (2014)
14. Diffie, W., Hellman, M.E.: Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6), 74–84 (1977)
15. Dong, X., Hua, J., Sun, S., Li, Z., Wang, X., Hu, L.: Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12827, pp. 278–308. Springer (2021)
16. Dunkelman, O., Sekar, G., Preneel, B.: Improved meet-in-the-middle attacks on reduced-round DES. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4859, pp. 86–100. Springer (2007)
17. Eichlseder, M., Leander, G., Rasoolzadeh, S.: Computing expected differential probability of (truncated) differentials and expected linear potential of (multidimensional) linear hulls in spn block ciphers. In: Progress in Cryptology–

INDOCRYPT 2020: 21st International Conference on Cryptology in India, Bangalore, India, December 13–16, 2020, Proceedings 21. pp. 345–369. Springer (2020)
18. Hadipour, H., Bagheri, N., Song, L.: Improved rectangle attacks on SKINNY and CRAFT. IACR Trans. Symmetric Cryptol. **2021**(2), 140–198 (2021)
19. Hadipour, H., Sadeghi, S., Eichlseder, M.: Finding the impossible: Automated search for full impossible-differential, zero-correlation, and integral attacks. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 128–157. Springer (2023)
20. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: Attacks on skein-512 and the SHA-2 family. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 244–263. Springer (2012)
21. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings. Lecture Notes in Computer Science, vol. 1008, pp. 196–211. Springer (1994)
22. Knudsen, L.R., Wagner, D.A.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2365, pp. 112–127. Springer (2002)
23. Lallemand, V., Naya-Plasencia, M.: Cryptanalysis of KLEIN. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8540, pp. 451–470. Springer (2014)
24. Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A cryptanalysis of printcipher: The invariant subspace attack. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6841, pp. 206–221. Springer (2011)
25. Ma, Z., Li, M., Chen, S.: Meet-in-the-middle attacks on round-reduced CRAFT based on automatic search. IET Inf. Secur. **17**(3), 534–543 (2023)
26. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993)
27. Moghaddam, A.E., Ahmadian, Z.: New automatic search method for truncated-differential characteristics application to midori, SKINNY and CRAFT. Comput. J. **63**(12), 1813–1825 (2020)
28. Rasoolzadeh, S., Ahmadian, Z., Salmasizadeh, M., Aref, M.R.: An improved truncated differential cryptanalysis of KLEIN. Tatra Mountains Mathematical Publications **67**(1), 135–147 (2016)
29. Shi, D., Sun, S., Derbez, P., Todo, Y., Sun, B., Hu, L.: Programming the demirciselçuk meet-in-the-middle attack with constraints. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11273, pp. 3–34. Springer (2018)

30. Shi, D., Sun, S., Song, L., Hu, L., Yang, Q.: Exploiting non-full key additions: Full-fledged automatic demirci-selçuk meet-in-the-middle cryptanalysis of SKINNY. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 67–97. Springer (2023)
31. Song, L., Zhang, N., Yang, Q., Shi, D., Zhao, J., Hu, L., Weng, J.: Optimizing rectangle attacks: A unified and generic framework for key recovery. In: Agrawal, S., Lin, D. (eds.) 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. Lecture Notes in Computer Science, vol. 13791, pp. 410–440. Springer (2022)
32. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, PartI. Lecture Notes in Computer Science, vol. 8873, pp. 158–178. Springer (2014)
33. Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round SKINNY. In: Joye, M., Nitaj, A. (eds.) Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10239, pp. 117–134 (2017)

Supplementary Material

A Preliminaries on truncated differential characteristics

We first briefly review the truncated differential distinguisher. In the following, consider a block cipher $E : F_2^n \times F_2^k \to F_2^n$, with an n-bit block and a k-bit key. The input and output difference variables of E are denoted by ΔX and ΔY, respectively.

Definition 1 (Differential Probability). *For a block cipher E_K, with the input and output concrete differences $\alpha, \beta \in F_2^n$, the differential probability of $(\alpha \xrightarrow{E} \beta)$ is defined as:*

$$P(\alpha \xrightarrow{E} \beta) = P(\Delta Y = \beta | \Delta X = \alpha) = P[E_K(x) \oplus E_K(x \oplus \alpha) = \beta] \tag{7}$$

The differential $(\alpha \xrightarrow{E} \beta)$ is called an efficient distinguisher if $P(\alpha \xrightarrow{E} \beta) \gg 2^{-n}$.

Definition 2 (Truncated Differential Probability). *For a block cipher E_K, with the input and output truncated differences $\Delta_{in}, \Delta_{out} \subseteq F_2^n$, the truncated differential probability of $(\Delta_{in} \xrightarrow{E} \Delta_{out})$ is defined as:*

$$\begin{aligned} P(\Delta_{in} \xrightarrow{E} \Delta_{out}) &= P(\Delta Y \in \Delta_{out} | \Delta X \in \Delta_{in}) \\ &= P[E_K(x) \oplus E_K(x \oplus \alpha) \in \Delta_{out} | \alpha \in \Delta_{in}] \end{aligned} \tag{8}$$

The truncated differential $(\Delta_{in} \xrightarrow{E} \Delta_{out})$ is called efficient if

$$P(\Delta_{in} \xrightarrow{E} \Delta_{out}) > P(\Delta_{in} \xrightarrow{PRP} \Delta_{out}) = \frac{|\Delta_{out}|}{2^n}. \tag{9}$$

In both cases, an efficient (truncated) differential can be used for distinguishing the block cipher E from a Pseudo Random Permutation (PRP). In the context of the concrete differential attack, it remains true that $P(\alpha \xrightarrow{E} \beta) = P(\beta \xrightarrow{E^{-1}} \alpha)$. However, it doesn't hold for truncated differentials. In [1], it is shown that for a block cipher E with a truncated input/output differential pair $(\Delta_{in}, \Delta_{out})$:

$$P(\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in}) = P(\Delta_{in} \xrightarrow{E} \Delta_{out}) \frac{|\Delta_{in}|}{|\Delta_{out}|} \tag{10}$$

B Theorems

In the followings, $\mathbf{r}_i$ and $\mathbf{c}_i$ refers to the i^{th} row and column of $\mathbf{M}$, respectively, and $Hw(\cdot)$ is the Hamming weight operation.

Theorem 1 (Differential Propagation Constraints). *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be the binary vectors indicating the truncated differences at the input and output of the* $c \times c$ *matrix* $\mathbf{M}$*. The subsequent two sets of constraints specify the MILP model for the activation of* $\mathbf{b}$*, given the activation of* $\mathbf{a}$*.*

$$\begin{cases} \mathbf{c}_i \cdot \mathbf{b} \geq Hw(\mathbf{c}_i) a_i & , \quad i = 0, \ldots, c-1 \\ b_i \leq \mathbf{r}_i \cdot \mathbf{a} & , \quad i = 0, \ldots, c-1 \end{cases} \tag{11}$$

Proof. We should consider two aspects for determining the set of constraints:

1. Constraints ensuring the activeness of $\mathbf{b}$ depending on the activeness of $\mathbf{a}$. Since $\mathbf{b} = \mathbf{M}\mathbf{a} = \sum_{i=0}^{c-1} \mathbf{c}_i a_i$, those entries of $\mathbf{b}$ would be affected by a_i whose corresponding entry in $\mathbf{c}_i$ is non-zero. This yields a set of constraints of the form $b_j \geq a_i$, where the j-th entry of $\mathbf{c}_i$ is nonzero. Summing up this set of constraints, they all can be summarized in the following inequality.

$$\mathbf{c}_i \cdot \mathbf{b} \geq Hw(\mathbf{c}_i) a_i \quad , \quad i = 0, \ldots, 3 \tag{12}$$

2. Constraints preventing the undesired activeness of $\mathbf{b}$. We have $b_i = \mathbf{r}_i \cdot \mathbf{a} = \sum_{j=0}^{c} (\mathbf{r}_i)_j a_j$. If none of a_j present in this linear description with a non-zero coefficient are active, then b_i should not be activated. This is equivalent to the following constraint:

$$b_i \leq \mathbf{r}_i \cdot \mathbf{a} \quad , \quad i = 0, \ldots, 3 \tag{13}$$

Theorem 2 (Value Determination Constraints). *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be the binary vectors indicating the value required/known words in the input and output of the* $c \times c$ *matrix* $\mathbf{M}$*. The subsequent sets of the MILP constraints specify the required* $\mathbf{a}$ *for determination of a given* $\mathbf{b}$*.*

$$\begin{cases} \mathbf{r}_i \cdot \mathbf{a} \geq Hw(\mathbf{r}_i) b_i & , \quad i = 0, \ldots, c-1 \\ a_i \leq \mathbf{c}_i \cdot \mathbf{b} & , \quad i = 0, \ldots, c-1 \end{cases} \tag{14}$$

Proof. We should consider two aspects for determining the set of constraints:

1. Constraints ensuring the activeness of $\mathbf{a}$ depending on the activeness of $\mathbf{b}$. We have $b_i = \mathbf{r}_i \cdot \mathbf{a} = \sum_{j=0}^{c} (\mathbf{r}_i)_j a_j$. So, if $b_i = 1$, all the a_j present in this linear description with a non-zero coefficient should be set to 1, i.e. we should have $a_j \geq b_i$ if the j-th entry of $\mathbf{r}_i$ is non-zero. The following inequality also establishes such conditions.

$$\mathbf{r}_i \cdot \mathbf{a} \geq Hw(\mathbf{r}_i) b_i \quad , \quad i = 0, \ldots, 3 \tag{15}$$

2. Constraints preventing the undesired activeness of $\mathbf{a}$. If

$$a_i \leq \mathbf{c}_i \cdot \mathbf{b} \quad , \quad i = 0, \ldots, 3 \tag{16}$$

Theorem 3. *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be the binary vectors indicating the value required/known words in the input and output of the* $c \times c$ *matrix* $\mathbf{M}$*. Suppose* $\mathbf{s}$ *is the binary vector indicating that the corresponding words in* $\mathbf{b}$ *are tested* ($s_i = 1$) *or not* ($s_i = 0$)*. The subsequent sets of the MILP constraints specify the required* $\mathbf{a}$ *for determination of a given* $\mathbf{b}$*, in the presence of* $\mathbf{s}$*.*

$$\begin{cases} s_i \leq b_i & , \quad i = 0, \ldots, c-1 \\ \mathbf{r}_i \cdot \mathbf{a} \geq Hw(\mathbf{r}_i)(b_i - s_i) & , \quad i = 0, \ldots, c-1 \\ a_i \leq \mathbf{c}_i \cdot (\mathbf{b} - \mathbf{s}) & , \quad i = 0, \ldots, c-1 \end{cases} \tag{17}$$

Proof. We should consider three aspects for determining the set of constraints:

1. Constraints preventing the undesired activeness of $\mathbf{s}$. Obviousely, the value required/known words ($b_i = 1$) should be given the chance of turning into a state-test word ($s_i = 1$). In other words, for words with $b_i = 0$, s_i must be 0. This leads us to the following constraint:

$$s_i \leq b_i \quad , \quad i = 0, \ldots, 3 \tag{18}$$

2. Constraints ensuring the activeness of $\mathbf{a}$ depending on the activeness of $\mathbf{b}$. This set of constraints is the counterpart of (12). If $s_i = 0$, it should follows (12) without any change. On the other hand, if $s_i = 1$, the value determination trace arising from this word, should be terminated, equivalently b_i should be considered zero. both cases are equivalent to replacing b_i with $b_i - s_i$ in (12), resulting the following constraint:

$$\mathbf{r}_i \cdot \mathbf{a} \geq Hw(\mathbf{r}_i)(b_i - s_i) \quad , \quad i = 0, \ldots, 3 \tag{19}$$

3. Constraints preventing the undesired activeness of $\mathbf{a}$. The same discussion as above is valid for this set of constraints along with its counterpart (13). So, we have:

$$a_i \leq \mathbf{c}_i \cdot (\mathbf{b} - \mathbf{s}) \quad , \quad i = 0, \ldots, 3 \tag{20}$$

Value Determination at the edges of the Distinguisher

Property 1. Suppose that $\mathbf{a}$ is the output truncated difference of the distinguisher at round $r_m + r_{in}$ with A defined as $A = \{j | a_j = 0\}$. Let $\mathbf{b} = M\mathbf{a}$, M' be matrix M excluding columns with indexes from A, and $B = \{i | \mathbf{r}'_i$ is independent of the other rows of $M'\}$. Then, the value of b_i, where $i \in B$ is not needed to be determined.

Then, one can derive the set of feasible points $(\mathbf{a}, \mathbf{b}')$, where $b'_i = 0$ if $i \in B$, defined in Property 1, otherwise $b'_i = b_i$. Then the set of constraints describing this set of feasible points can be constructed, using the method proposed in [32].

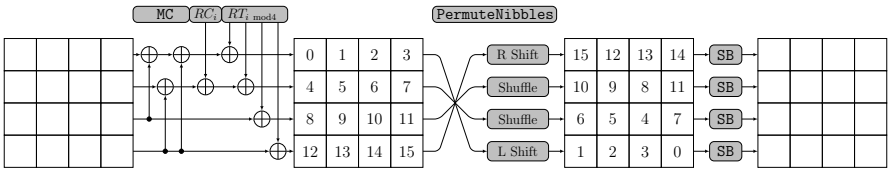

Fig. 11. `CRAFT` round function [9]

C Specification of the Ciphers

C.1 Description of `CRAFT`

`CRAFT` is a lightweight tweakable block cipher made out of involutory building blocks [9]. It consists of a 64-bit block, a 128-bit key, and a 64-bit tweak. The state is seen as a 4×4 matrix of 4-bits cells (nibbles) and is denoted by I. Row 0 is considered the uppermost one and column 0 is taken to be the leftmost one. The numbering of the words inside the state matrix is as follows.

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

The round function of `CRAFT` is depicted in Fig. 11. By initializing the state with the plaintext, the cipher iterates 31 round functions ($\mathcal{R}_i$, $0 \leq i \leq 30$), followed by one more round $\mathcal{R}'_{31}$) to compute the ciphertext. The round operations are defined as follows:

1. `MixColumn` (`MC`): The following binary matrix M is multiplied with each column of the state. Note that $M = M^{-1}$

$$M = \begin{pmatrix} 1\,0\,1\,1 \\ 0\,1\,0\,1 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{pmatrix}.$$

2. $\texttt{AddConstant}_i$ ($\texttt{ARC}_i$) A pair of round constants RC_i is Xored to nibbles 4 and 5 of the state at round i.
3. $\texttt{AddTweakey}_i$ ($\texttt{ATK}_i$): The round tweakey $RT_i = TK_{i \bmod 4}$, described as follows, is Xored to the cipher state.

$$TK_0 = K_0 \oplus T, \quad TK_1 = K_1 \oplus T, \quad TK_2 = K_0 \oplus \texttt{Q}(T), \quad TK_3 = K_1 \oplus \texttt{Q}(T).$$

where $K = (K_0||K_1)$ is the 128-bit secret key, and $\texttt{Q}(\cdot)$ is the permutation:

$$\texttt{Q} = [12, 10, 15, 5, 14, 8, 9, 2, 11, 3, 7, 4, 6, 0, 1, 13].$$

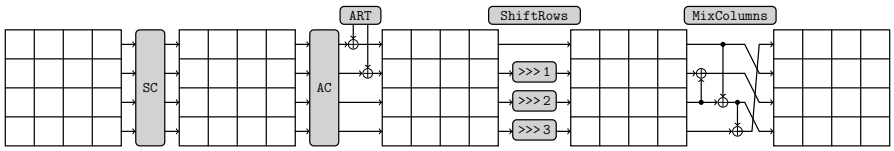

Fig. 12. Round function of SKINNY-64-192 [8].

4. PermuteNibbles (P): An involutory permutation P is applied to nibbles of the cipher state.

$$\mathtt{P} = [15, 12, 13, 14, 10, 9, 8, 11, 6, 5, 4, 7, 1, 2, 3, 0].$$

5. SubBox (SB): The following 4-bit involutory S-box S is applied to each nibble of the state.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S(x)	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

Finally, the round functions of CRAFT are defined as $\mathcal{R}_i = \mathtt{SB} \circ \mathtt{PN} \circ \mathtt{ATK}_i \circ \mathtt{ARC}_i \circ \mathtt{MC}$ for $i = 0, \ldots, 30$, and $\mathcal{R}_{31} = \mathtt{ATK}_i \circ \mathtt{ARC}_i \circ \mathtt{MC}$.

C.2 Description of SKINNY-64-192

SKINNY, designed by Beierle et al. [8], is a family of tweakable lightweight block ciphers denoted by SKINNY-n-t, where $n = 64$ or 128 is the block size, and $t = n, 2n$, and $3n$ is the tweakey [8] (key plus tweak) size. The exact specification of the cipher depends on the block and key sizes. Here, we consider SKINNY-64-192. The cipher follows an SPN structure with a very compact S-box, a linear layer based on a sparse non-MDS binary matrix, and a lightweight tweakey schedule. The state is seen as a 4×4 matrix of 4-bits cells. Row 0 is considered the uppermost one and column 0 is taken to be the leftmost one. The numbering of the words inside the state matrix also follows that of CRAFT, explained in Sec. C.1.

The round function of SKINNY is depicted in Fig. 12, which is iterated 40 times for SKINNY-64-192. One round of SKINNY is composed of five operations applied in the following order: SubCells, AddConstants, AddRoundTweakey, ShiftRows and MixColumns.

1. SubCells (SC). A 4-bit S-box S, described as follows, is applied to every cell of the cipher's internal state.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S(x)	c	6	9	0	1	a	2	b	3	8	5	d	4	e	7	f
$\mathtt{S}^{-1}(x)$	3	4	6	8	c	a	1	e	9	2	5	7	0	b	d	f

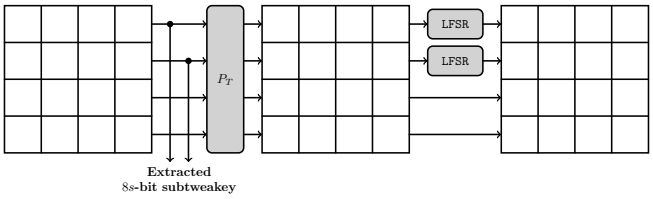

Fig. 13. The tweakey schedule in SKINNY. Each tweakey word **TK1**, **TK2**, and **TK3** follows a similar transformation update, except that no LFSR is applied on $TK1$.

2. AddConstants (AC). A round constant is added to the internal state of the cipher.
3. AddRoundTweakey (ART). In case of SKINNY-64-192, where $t = 3n$, the first and second rows of the three tweakey arrays **TK1**, **TK2**, and **TK3** are extracted from the tweakey with respect to the tweakey schedule and Xored to the first two rows of the internal state, respecting the cell positions inside the arrays.
4. ShiftRows (SR). The rows of the cipher internal state array are rotated to the right. More precisely, the second, third, and fourth cell rows are rotated by 1, 2 and 3 positions to the right, respectively.
5. MixColumns (MC). Each column of the cipher internal state array is multiplied by the following binary matrix $\mathbf{M}$ (or $\mathbf{M}^{-1}$ for decryption):

$$\mathbf{M} = \begin{pmatrix} 1\,0\,1\,1 \\ 1\,0\,0\,0 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \end{pmatrix} \text{ and } \mathbf{M}^{-1} = \begin{pmatrix} 0\,1\,0\,0 \\ 0\,1\,1\,1 \\ 0\,1\,0\,1 \\ 1\,0\,0\,1 \end{pmatrix}.$$

The tweakey schedule used in SKINNY-64-192 updates **TK1**, **TK2**, and **TK3** as illustrated in Fig. 13. First, the cell-wised permutation

$$P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$$

is applied on the tweakey arrays. Moreover, every cell of the first and second rows of **TK2** and **TK3** are individually updated with LFSRs given in Tab. 6, where x_0 is the least significant bit of the cell.

TK	**LFSR**		
TK2	$(x_3 \,\|\|\, x_2 \,\|\|\, x_1 \,\|\|\, x_0)$	$\rightarrow$	$(x_2 \,\|\|\, x_1 \,\|\|\, x_0 \,\|\|\, x_3 \oplus x_2)$
TK3	$(x_3 \,\|\|\, x_2 \,\|\|\, x_1 \,\|\|\, x_0)$	$\rightarrow$	$(x_3 \oplus x_0 \,\|\|\, x_3 \,\|\|\, x_2 \,\|\|\, x_1)$

Table 6. The LFSRs used in the tweakey schedule SKINNY-64-192.

D Branching Property Tables

Definition 3 (Branching Property Table (BPT)). *Let M be a matrix over $\mathbb{F}_{2^m}$ of size $c \times c$. The BPT of M is a $2^c \times 2^c$ table, such that $BPT(\mathbf{a}, \mathbf{b})$ represents transition probability $P(\mathbf{a} \xrightarrow{M} \mathbf{b})$, where $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^c$ are the input and output truncated differential vectors. In more details,*

$$BPT(\mathbf{a}, \mathbf{b}) = \log_2(Pr_{\mathbf{x}}\{Tr(M \cdot \mathbf{x}) = \mathbf{b} | Tr(\mathbf{x}) = \mathbf{a}\}) \tag{21}$$

where $Tr(\cdot)$ is the truncation operator, and the probability is taken over all uniformly distributed $\mathbf{x}$ over $\mathbb{F}_{2^m}^c$.

Tables 7 and 8 display the BPTs for the `SKINNY-64-192` and `CRAFT MixColumn` matrices, respectively. Each entry $(\mathbf{a}, \mathbf{b})$ in these tables represents the base-2 logarithm of the transition probability. The impossible transitions are indicated by a `"-"` in the tables.

Table 7. BPT for `SKINNY MixColumn` Matrix

in/out	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x1	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-
0x2	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-
0x3	-	-	-	-3.907	-	-	-	-	-	-	-	-0.09954	-	-	-	-
0x4	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-
0x5	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-
0x6	-	-	-	-	-	-	-	-	-	-3.907	-	-0.09954	-	-	-	-
0x7	-	-7.814	-	-4.0064	-	-	-	-	-	-4.0064	-	-0.19907	-	-	-	-
0x8	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-
0x9	-	-	-	-	-	-3.907	-	-	-	-	-	-	-	-0.09954	-	-
0xa	-	-	-	-	-	-	-3.907	-	-	-	-	-	-	-	-	-0.09954
0xb	-	-	-	-	-	-	-	-4.106	-	-	-	-	-	-	-3.9996	-0.1990
0xc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
0xd	-	-	-	-	-	-	-	-3.907	-	-	-	-	-	-	-	-0.09954
0xe	-	-	-	-	-7.814	-	-4.006	-	-	-	-	-	-	-4.0064	-	-0.1990
0xf	-	-	-	-	-	-7.913	-	-4.106	-	-	-	-	-7.814	-4.106	-4.0064	-0.2986

Table 8. BPT for `CRAFT` `MixColumn` Matrix

in/out	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x1	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-
0x2	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-
0x3	-	-	-	-	-	-	-	-3.907	-	-	-	-	-	-	-	-0.09954
0x4	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
0x5	-	-	-	-	-	-	-	-	-	-3.907	-	-	-	-0.09954	-	-
0x6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-
0x7	-	-	-	-7.814	-	-	-	-4.0064	-	-	-	-4.0064	-	-	-	-0.19907
0x8	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-
0x9	-	-	-	-	-	-3.907	-	-	-	-	-	-	-	-0.09954	-	-
0xa	-	-	-3.907	-	-	-	-	-	-	-	-0.09954	-	-	-	-	-
0xb	-	-	-	-	-	-	-	-4.0064	-	-	-	-	-	-	-	-0.09268
0xc	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-
0xd	-	-7.814	-	-	-	-4.0064	-	-	-	-4.0064	-	-	-	-0.19907	-	-
0xe	-	-	-	-	-	-	-3.907	-	-	-	-	-	-	-	-0.09954	-
0xf	-	-	-	-7.913	-	-	-	-4.106	-	-	-	-3.9996	-	-	-	-0.1922

E Equations used in the State-test technique of CRAFT

In the attack of CRAFT, we use the state-test technique to test the 7 nibbles of the state given below, instead of guessing more subkey nibbles. The nibbles are given as a function of all the subkey nibbles needed to compute them and the f_i are non-linear functions, possibly involving internal states words, taking subkey words as parameters.

$$
\begin{aligned}
W_1[14] &= K_1[3] \oplus f_0(K_0[14]) \oplus f_1(K_0[7]) \oplus f_2(K_0[0]);\\
W_2[12] &= K_2[1] \oplus f_0(K_1[12], f_1(K_0[1])) \oplus f_2(K_1[5], f_3(K_0[2]), f_4(K_0[9]))\\
&\quad \oplus f_5(K_1[2], f_6(K_0[3]), f_7(K_0[4]), f_8(K_0[13]));\\
W_3[14] &= K_3[3] \oplus f_0(K_2[14], f_1(K_1[3]), f_2(K_0[0]), f_3(K_0[7]), f_4(K_0[14]))\\
&\quad \oplus f_5(K_2[7], f_6(K_1[11], f_7(K_0[7]), f_8(K_1[0], f_9(K_0[1]), f_{10}(K_0[6]), f_{11}(K_0[15]))))\\
&\quad \oplus f_{12}(K_2[0], f_{13}(K_1[15], f_{14}(K_0[0])), f_{15}(K_1[6], f_{16}(K_0[8]), f_{17}(K_0[3])),\\
&\quad f_{18}(K_1[1], f_{19}(K_0[2]), f_{20}(K_0[5]), f_{21}(K_0[12])))\\
W_4[12] &= K_4[1] \oplus f_0(K_3[12], f_1(K_2[1], f_2(K_1[12], f_3(K_0[1])),\\
&\quad f_4(K_1[5], f_5(K_0[9]), f_2(K_0[2]))f_6(K_1[2], f_7(K_0[3]), f_8(K_0[4]), f_9(K_0[13]))))\\
&\quad \oplus f_{10}(K_3[5], f_{11}(K_2[9], f_{12}(K_1[5], f_{13}(K_0[9]), f_{14}(K_0[2]))), f_{15}(K_2[2], f_{16}(K_1[13],\\
&\quad f_{17}(K_0[2])), f_{18}(K_1[4], f_{19}(K_0[10]), f_{20}(K_0[1])), f_{21}(K_1[3], f_{22}(K_0[0]), f_{23}(K_0[7]), f_{24}(K_0[14]))))\\
&\quad \oplus f_{25}(K_3[2], f_{26}(K_2[13], f_{27}(K_1[2], f_{28}(K_0[13]), f_{29}(K_0[4]), f_{30}(K_0[3])),\\
&\quad f_{31}(K_2[4], f_{32}(K_1[10], f_{33}(K_0[4])), f_{34}(K_1[1], f_{35}(K_0[12]), f_{36}(K_0[5]), f_{37}(K_0[2])),\\
&\quad f_{38}(K_2[3], f_{39}(K_1[0], f_{40}(K_0[15]), f_{41}(K_0[6]), f_{42}(K_0[1])), f_{43}(K_1[7], f_{44}(K_0[11]), f_{45}(K_0[0])),\\
&\quad f_{46}(K_1[14], f_{47}(K_0[3])))))\\
X_{20}[1] &= K_{20}[1, 9, 13] \oplus f_0(K_{21}[2, 10, 14]) \oplus f_1(K_{21}[5, 13])\\
X_{19}[3] &= K_{19}[3, 11, 15] \oplus f_0(K_{20}[0, 8, 12], f_1(K_{21}[1, 9, 13]), f_2(K_{21}[6, 14]), f_3(K_{21}[15]))\\
&\quad \oplus f_4(K_{20}[7, 15], f_5(K_{21}[11]), f_6(K_{21}[0, 8, 12])) \oplus f_7(K_{20}[14], f_8(K_{21}[3, 11, 15]))\\
X_{18}[1] &= K_{18}[1, 9, 13] \oplus f_0(K_{19}[2, 10, 14], f_1(K_{20}[13], f_2(K_{21}[2, 10, 14])), f_3(K_{20}[4, 12], f_4(K_{21}[10]),\\
&\quad f_5(K_{21}[1, 9, 13])), f_6(K_{20}[3, 11, 15], f_7(K_{21}[14]), f_8(K_{21}[7, 15]), f_9(K_{21}[0, 8, 12])))\\
&\quad \oplus f_{10}(K_{19}[5, 13], f_{11}(K_{20}[9], f_{12}(K_{21}[5, 13])), f_{13}(K_{20}[2, 10, 14], f_{14}(K_{21}[3, 11, 15]),\\
&\quad f_{15}(K_{21}[4, 12]), f_{16}[K_{21}[13]])\\
&\quad \oplus f_{17}(K_{19}[12], f_{18}(K_{20}[1, 9, 13], f_{19}(K_{21}[12]), f_{20}(K_{21}[5, 13]), f_{21}(K_{21}[2, 10, 14]).
\end{aligned}
$$

For the sake of clarity, we rewrite the equations erasing the key bits we already know after the attack steps. We get the following equations:

- round 1: the equation is $K_1[3] \oplus \text{Known} = 0$; which give us the value of $K_1[3]$.
- round 2: $f_0(K_1[5]) \oplus f_1(K_1[2]) \oplus \text{Known} = 0$; from this we get $K_1[5]$ in function of $K_1[2]$.
- round 20: $f_0(K_1[2, 10, 14]) \oplus f_1(K_1[5]) \oplus \text{Known} = 0$, which gives us $K_1[10]$ in function of $K_1[2]$ and $K_1[14]$.

– round 3: $f_0(f_1(K_1[11]), f_2(K_1[0])) \oplus f_3(f_4(K_1[11]), f_5(K_1[6]), f_6(K_1[1])) \oplus$ Known $= 0$.
– round 19: $f_0(f_1(K_1[1,9]), f_2(K_1[6,14]), f_3(K_1[11])) \oplus f_4(f_5(K_1[11]), f_6(K_1[0])) \oplus$ Known $= 0$.
– round 4: the equation becomes

$$f_0(K_1[5], f_1(f_{12}(K_1[5]))) \oplus f_2(K_1[2], f_3(f_4(K_1[2]), f_5(f_6(K_1[10]) , f_7(K_1[1]), f_8(f_9(K_1[0]), f_{10}(K_1[7]), f_{11}(K_1[14]))))) \oplus \text{Known} = 0.$$

– round 18: the equation becomes

$$f_0(K_1[2,10,14], f_1(f_2(K_1[2,10,14])), f_3(f_4(K_1[10]), f_5(K_1[1,9])), f_6(f_7(K_1[14]) , f_8(K_1[7,11]), f_9(K_1[0]))) \oplus f_{10}(K_1[5], f_{11}(f_{12}(K_1[5]))) \oplus \text{Known} = 0.$$

F Details of the 23-round attack on SKINNY-64-192

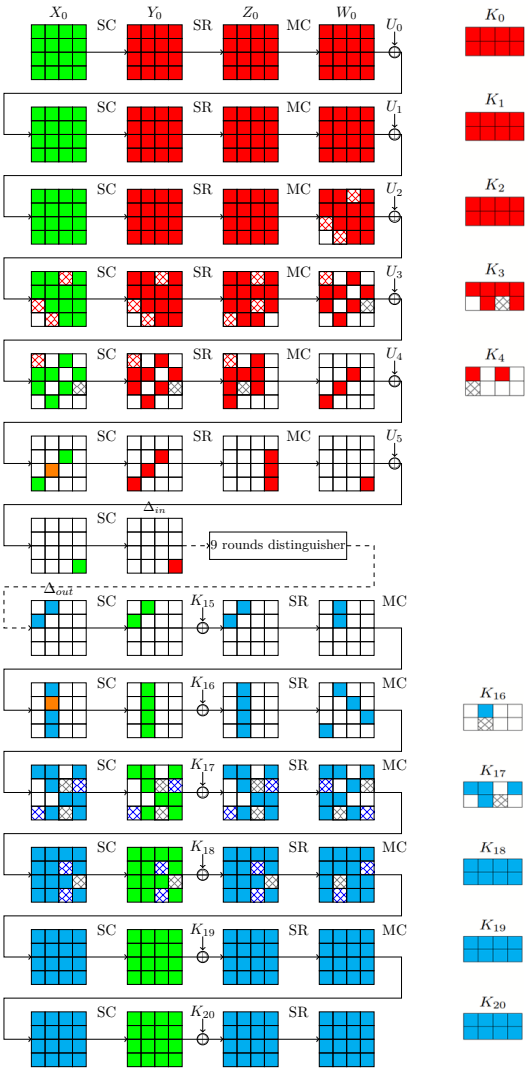

Fig. 14. The 21-round core part of the 23-round attack on SKINNY-64-192 from round 0 to round 20 included. Differential propagation in the upper (lower) part has been shown in red (blue). The stripped red (blue) nibbles are those non-active nibbles whose value is needed to compute the value of green ones in the upper (lower) part. The state-test nibble is shown in orange. The gray-striped nibbles are whose values are no longer required, thanks to the state-tests. The equivalent subkey $U_i = \texttt{MC}(\texttt{SR}(K_i))$.

$\mathbf{TK}_i$ Nibble	k_{in}	k_{out}	# Equations
0	$K_0[0]$, $K_2[2]$	$K_{18}[2]$, $K_{20}[4]$	1
1	$K_0[1]$, $K_2[0]$, $K_4[2]$	$K_{16}[1]$, $K_{18}[0]$, $K_{20}[2]$	3
2	$K_0[2]$, $K_2[4]$	$K_{18}[4]$, $K_{20}[6]$	1
3	$K_0[3]$, $\mathbf{K_2[7]}$	$K_{18}[7]$, $K_{20}[1]$	1
4	$K_0[4]$, $K_2[6]$	$\mathbf{K_{18}[6]}$, $K_{20}[5]$	1
5	$K_0[5]$, $K_2[3]$	$K_{18}[3]$, $K_{20}[7]$	1
6	$K_0[6]$, $K_2[5]$	$K_{18}[5]$, $K_{20}[3]$	1
7	$K_0[7]$, $K_2[1]$, $K_4[0]$	$K_{18}[1]$, $K_{20}[0]$	2
8	$K_1[2]$	$K_{19}[4]$	0
9	$K_1[0]$, $\mathbf{K_3[2]}$	$K_{17}[0]$, $K_{19}[2]$	1
10	$K_1[4]$	$K_{19}[6]$	0
11	$K_1[7]$, $K_3[1]$	$K_{19}[1]$	0
12	$K_1[6]$, $\mathbf{K_3[5]}$	$K_{19}[5]$	0
13	$K_1[3]$, $K_3[7]$	$K_{17}[3]$, $K_{19}[7]$	1
14	$K_1[5]$, $K_3[3]$	$\mathbf{K_{17}[5]}$, $K_{19}[3]$	1
15	$K_1[1]$, $K_3[0]$	$K_{17}[1]$, $K_{19}[0]$	1

Table 9. Subkey nibbles involved in the 21-round attack of `SKINNY-64-192`. The bold symbols correspond to the key words not involved in state-test relations anymore.

$\mathbf{TK}_i$ Nibble	k_{in}	k_{out}	# Equations
0	$K_0[0]$, $K_2[2]$, $K_{22}[6]$	$K_{18}[2]$, $K_{20}[4]$	2
1	$K_0[1]$, $K_2[0]$, $K_4[2]$	$K_{16}[1]$,$K_{18}[0]$, $K_{20}[2]$	3
2	$K_0[2]$, $K_2[4]$	$K_{18}[4]$, $K_{20}[6]$	1
3	$K_0[3]$, $K_2[7]$	$K_{18}[7]$, $K_{20}[1]$	1
4	$K_0[4]$, $K_2[6]$	$K_{18}[6]$, $K_{20}[5]$	1
5	$K_0[5]$, $K_2[3]$	$K_{18}[3]$, $K_{20}[7]$	1
6	$K_0[6]$, $K_2[5]$	$K_{18}[5]$, $K_{20}[3]$	1
7	$K_0[7]$, $K_2[1]$, $K_4[0]$	$K_{18}[1]$, $K_{20}[0]$	2
8	$K_1[2]$	$K_{19}[4]$	0
9	$K_1[0]$, $K_3[2]$	$K_{17}[0]$, $K_{19}[2]$	1
10	$K_1[4]$	$K_{19}[6]$	0
11	$K_1[7]$, $K_3[1]$	$K_{19}[1]$, $K_{21}[0]$	1
12	$K_1[6]$, $K_3[5]$	$K_{19}[5]$, $K_{21}[3]$	1
13	$K_1[3]$, $K_3[7]$	$K_{17}[3]$, $K_{19}[7]$, $K_{21}[1]$	2
14	$K_1[5]$, $K_3[3]$	$K_{19}[3]$, $K_{17}[5]$	1
15	$K_1[1]$, $K_3[0]$	$K_{17}[1]$, $K_{19}[0]$	1

Table 10. Subkey nibbles involved in the 23-round attack of `SKINNY-64-192`.

Internal state computed from X_{21}	Internal state computed from C
$\mathtt{SB}(\mathtt{SB}(X_{21}[0]) \oplus K_{21}[0] \oplus \mathtt{SB}(X_{21}[10]) \oplus \mathtt{SB}(X_{21}[13]))$	$C[0] \oplus K_{22}[[0]]$
$\mathtt{SB}(X_{21}[7]) \oplus K_{21}[7] \oplus \mathtt{SB}(X_{21}[10])$	$\mathtt{SB}^{-1}(C[8])$
$\mathtt{SB}(\mathtt{SB}(X_{21}[1]) \oplus K_{21}[1] \oplus \mathtt{SB}(X_{21}[11]) \oplus \mathtt{SB}(X_{21}[14]))$	$C[1] \oplus K_{22}[1]$
$\mathtt{SB}(\mathtt{SB}(X_{21}[1]) \oplus K_{21}[1])$	$C[5] \oplus K_{22}[5]$
$\mathtt{SB}(X_{21}[4]) \oplus K_{21}[4] \oplus \mathtt{SB}(X_{21}[11])$	$\mathtt{SB}^{-1}(C[9])$
$\mathtt{SB}(X_{21}[2]) \oplus K_{21}[2]$	$\mathtt{SB}^{-1}(C[6] \oplus K_{22}[6])$
$\mathtt{SB}(X_{21}[5]) \oplus K_{21}[5] \oplus \mathtt{SB}(X_{21}[8])$	$\mathtt{SB}^{-1}(C[10])$
$\mathtt{SB}(\mathtt{SB}(X_{21}[3]) \oplus K_{21}[3] \oplus \mathtt{SB}(X_{21}[9]) \oplus \mathtt{SB}(X_{21}[12]))$	$C[3] \oplus K_{22}[3]$
$\mathtt{SB}(\mathtt{SB}(X_{21}[3]) \oplus K_{21}[3])$	$C[7] \oplus K_{22}[7]$
$\mathtt{SB}(X_{21}[6]) \oplus K_{21}[6] \oplus \mathtt{SB}(X_{21}[9])$	$\mathtt{SB}^{-1}(C[11])$

Table 11. Linear relation to match between (X_{21}, C) and between $(\tilde{X}_{21}, \tilde{C})$. Two sides of each row are equal.

G Figures of the added rounds in two 25-round attacks on SKINNY-128-384

In Sec. 7.2, we explained how to mount an attack on 25-round of SKINNY-128-384 using our improvement of the parallel partitioning of Sec. 4.1. Here we give the figure representing the two last rounds of the attacks.

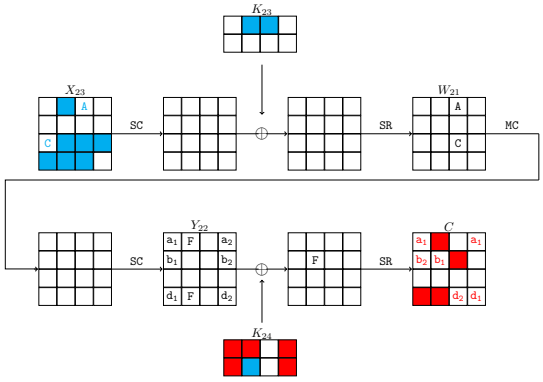

Fig. 15. Parallel partitioning applied to the last two rounds of the 25-round attack on SKINNY-128-384 using the 23-round path given in [12].

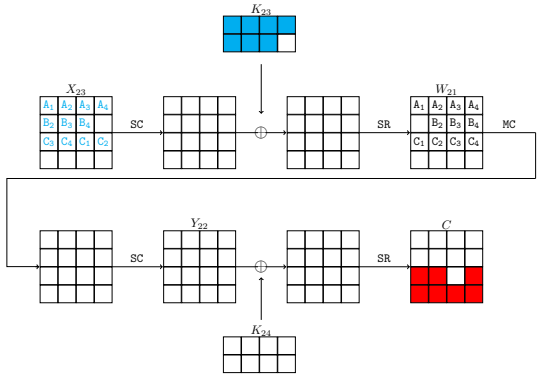

Fig. 16. Parallel partitioning applied to the last two rounds of the 25-round attack on SKINNY-128-384 using the 24-round path minus the last round given in [12].