# Improvements of the differential meet-in-the-middle attack

Dounia M'Foukh

# Internship report

Improvements of the differential meet-in-the-middle attack

M'foukh Dounia

**Internship carried out from 01/03/2023 to 31/08/2023**

**Internship tutor :** Mme. Naya-Plasencia María

**Éducational institution :** Université Paris-Saclay UVSQ - M2 Algèbre Appliquée/ Université Paris-Saclay

# Contents

# 1 Introduction

The main goal of cryptography is to protect information. There exist two types of cryptography : asymmetric and symmetric cryptography. In symmetric-key cryptography, both users know a secret key and thus they are the only one who are supposed to be able to encrypt or decrypt messages. In asymmetric cryptography both users have their own secret key and its associated public key, thus everyone can encrypt a message but only the user with the right secret key can decipher the message. The security of asymmetric systems is based on some mathematical problem that is considered difficult to solve. For example, reconstructing the private key of the RSA algorithm from the public key is done through factorization, which is considered to be a hard problem. While the security of asymmetric cryptography is based on mathematical problems with high computational cost, symmetric-key cryptography security is defined by the best generic attack, namely an attack that doesn't depends on the specifications of the cryptographic primitive. In order to ensure the actual security meets this ideal or generic one, we need to perform cryptanalysis which is the study of the possible flaws in cryptographic primitives through the elaboration of techniques and tools. However, public-key primitives need more computational power to be used and are much less efficient, so they are usually used to share the secret key of a symmetric-key primitive that will be used to encrypt the data.

Moreover in the last decades, the number of connected devices has considerably increased and all types of everyday devices are now connected. Thus there is a need to come up with efficient and secure cryptographic algorithms well adapted for this new situation. Moreover, since the devices keep getting smaller and smaller then the primitives need to be well adapted to this constrained environment. The lightweight block ciphers are symmetric-key cryptosystems that have been designed to perform well in this restricted environment. In the last few years many lightweight block ciphers have been proposed for different constraint and with different advantages and flaws, the trade-off between a high level of security and an efficient implementation is interesting. However the confidence of these algorithms to be secure against an attacker is based on the cryptanalysis of the researchers who try to find flaws in the algorithm. Thus cryptanalysis research is essential to provide secure systems to the growing number of connected devices.

During my internship at Inria, Paris under the supervision of María Naya-Plasencia, I have studied a new family of attack in symmetric cryptanalysis, and provided several improvements. Thus I have first studied the article [8] which introduces the differential meet-in-the-middle attack. Then we worked on finding improvements of this new attack, by using another type of differential characteristic or by improving the key-recovery part of the attack using a probabilistic path during the extension of the differential characteristic, using the state-test technique introduced in [9] and by generalizing the improvement using structures given in the differential MITM article [8]. Finally, we worked with Zahra Ahmadian, Akram Khalesi and Hossein Moghimi of Shahid beheshti University of Tehran in Iran who optimize an automated search tool, Mixed Integer Linear Programming (MILP), to find the optimum truncated differential characteristic. And we used these attacks to study the security of some lightweight block ciphers such as SKINNY [5] and CRAFT [7], providing the best known attacks in both case. While searching for an application for the truncated differential MITM attack, we worked on the block cipher QARMA [3]. And even though we didn't found a satisfying attack with the truncated differential MITM technique, we instead found a truncated differential attack on 10 rounds of QARMA which is the best known single-key attack on the cipher. And we submitted this attack on the journal Transactions on Symmetric Cryptology 2023. However, I will not present this result in this report.

# 2 Differential Meet-In-The-Middle

The differential Meet-In-The-Middle (differential MITM) attack was introduced by Christina Boura, Nicolas David, Patrick Derbez, Gregor Leander and María Naya-Plasencia in [8]. It is a new cryptanalysis technique against symmetric primitives. This attack aims at combining two of the most important families of symmetric cryptanalysis attack, the meet-in-the-middle (MITM) attacks together with differential attack and can be seen as an extension of classical MITM attacks but can also be interpreted as a new key-recovery method to apply in differential cryptanalysis.

In [8], they applied this new technique in the single-tweakey setting on the block cipher SKINNY, and they show how to break 25 out of the 56 rounds of SKINNY-128-384, the 128-bit block variant employing a 384-bit key. This application show the efficiency of the attack since it improve by two rounds the previous best known attack on the cipher. Then, they have also provided an application to AES-256 by breaking 12 rounds of the cipher in the related-key model. Their attack uses only 2 related keys, whereas the best previous attacks with this number of related keys could reach at most 10 rounds.

In this section, I will give a description of how the differential MITM attack is built. I will also briefly describe a techniques for [8] that combined with the differential MITM attack allow to add one round mostly for free using a parallel treatment of data partitions and a technique to reduce the data complexity.

## 2.1 Preliminaries on differential characteristics

First we present some result on differential distinguisher. We 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 $\Delta X$ and $\Delta Y$, respectively.

**Definition 1** (Differential Probability)**.** For a block cipher $E$, a key $K$ and a $n$-bit block $x$ the differential probability of the concrete input difference $\alpha \in F_2^n$ and output difference $\beta \in F_2^n$ is defined as:

$$P(\alpha \xrightarrow{E} \beta) = P(\Delta Y = \beta | \Delta X = \alpha) = P[E_K(x) \oplus E_K(x \oplus \alpha) = \beta] \tag{1}$$

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 block cipher $E$, the truncated differential probability with input truncated difference $\Delta_{in} \subseteq F_2^n$, and output truncated difference $\Delta_{out} \subseteq F_2^n$, is defined as:

$$\begin{aligned} P(\Delta_{in} \xrightarrow{E} \Delta_{out}) &= P(\Delta Y \subseteq \Delta_{out} | \Delta X \subseteq \Delta_{in}) \\ &= P[E_K(x) \oplus E_K(x \oplus \alpha) \in \Delta_{out} | \alpha \in \Delta_{in}] \end{aligned} \tag{2}$$

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

$$P(\Delta_{in} \xrightarrow{E} \Delta_{out}) > P(\Delta_{in} \xrightarrow{PRP} \Delta_{out}) = \frac{|\Delta_{out}|}{2^n}. \tag{3}$$

**Preposition 1.** *For block cipher $E$ with concrete input-output differential pair $(\alpha, \beta)$, it holds that:*

$$P(\alpha \xrightarrow{E} \beta) = P(\beta \xrightarrow{E^{-1}} \alpha) \tag{4}$$

**Lemma 1.** *For block cipher $E$ with truncated input-output differential pair $(\Delta_{in}, \Delta_{out})$ it holds that:*

$$P(\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in}) = P(\Delta_{in} \xrightarrow{E} \Delta_{out}) \frac{|\Delta_{in}|}{|\Delta_{out}|} \tag{5}$$

*Proof.* Using Bayse theorem,

$$\begin{aligned} P(\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in}) &= P(\Delta X \subseteq \Delta_{in} | \Delta Y \subseteq \Delta_{out}) \\ &= P(\Delta Y \subseteq \Delta_{out} | \Delta X \subseteq \Delta_{in}) \frac{Pr(\Delta X \subseteq \Delta_{in})}{P(\Delta Y \subseteq \Delta_{out})} \\ &= P(\Delta_{in} \xrightarrow{E} \Delta_{out}) \frac{|\Delta_{in}|}{|\Delta_{out}|} \end{aligned} \tag{6}$$

where the last equality holds by assuming uniform distributions for $\Delta X$ and $\Delta Y$. □

**Preposition 2.** *The truncated differential* $(\Delta_{in} \xrightarrow{E} \Delta_{out})$ *is efficient, iff* $(\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in})$ *is efficient.*

*Proof.*

$$\begin{aligned} (\Delta_{in} \xrightarrow{E} \Delta_{out}) \text{ is efficient} &\iff Pr(\Delta_{in} \xrightarrow{E} \Delta_{out}) > \frac{|\Delta_{out}|}{2^n} \\ &\overset{(5)}{\iff} Pr(\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in}) > \frac{|\Delta_{in}|}{2^n} \\ &\iff (\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in}) \text{ is efficient.} \end{aligned} \tag{7}$$

□

**Preposition 3.** *The differential* $(\Delta_{in} \xrightarrow{E} \Delta_{out})$ *is the optimum (highest probability) efficient truncated differential for* $E$*, iff* $(\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in})$ *is the optimum efficient truncated differential for* $E^{-1}$*.*

*Proof.* This preposition can be easily proved by contradiction.

Indeed, the efficient equivalence is treated in Preposition 2 and thanks to Lemma 1, if $(\Delta_{in} \xrightarrow{E} \Delta_{out})$ is the optimum efficient truncated differential for $E$ then $(\Delta_{out} \xrightarrow{E^{-1}} \Delta_{in})$ will be the optimum efficient truncated differential for $E^{-1}$. □

## 2.2 Framework of the differential MITM attack

Consider a $n$-bit cipher $E$ decomposed into three sub-ciphers: $E_{out} \circ E_m \circ E_{in}$, as depicted in Figure 1. Let the number of rounds of $E_{in}$, $E_m$ and $E_{out}$ be $r_{in}$, $r_m$ and $r_{out}$ respectively. Finally, let $\Delta_{in}$ be the input difference to the middle part $E_m$, $\Delta_{out}$ the output difference of $E_m$ and suppose that the differential $\Delta_{in} \to \Delta_{out}$, covering the $r_m$ middle rounds, has probability $2^{-p}$. The main idea of the attack is to use a MITM approach, i.e. from a pair of plaintext/ciphertext $(P, C)$ we compute the candidate plaintexts and ciphertexts $\tilde{P}$ and $\tilde{C}$ respectively from the plaintext $P$ and the difference $\Delta_{in}$ for each possible value of the associated key $k_{in}$ and from the ciphertext $C$ and the difference $\Delta_{out}$ for each possible value of the associated key $k_{out}$ such that the pair follows the differential characteristic as summarized on figure 1.

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

$$\begin{cases} E_{in}(P) \oplus E_{in}(\tilde{P}) &= \Delta_{in} \\ E_{out}^{-1}(C) \oplus E_{out}^{-1}(\tilde{C}) &= \Delta_{out}. \end{cases} \tag{8}$$

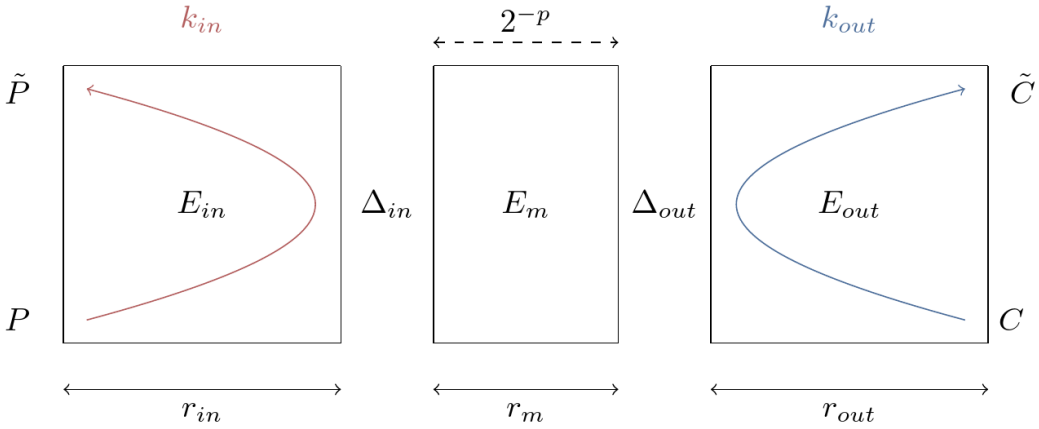


Figure 1: Framework of the differential meet-in-the-middle attack.

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

- *Upper part :* From $P$, $\Delta_{in}$ and some key information, we want to compute $\tilde{P}$ such that $E_{in}(P) \oplus E_{in}(\tilde{P}) = \Delta_{in}$ if the key guess is correct. Thus, we want to minimize the number of key information, noted $k_{in}$, 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 we store them in a hash table.

- *Lower part :* Similarly, given $C$, $\Delta_{out}$ and a minimized number of key information, noted $k_{out}$, we can compute $\tilde{C}^j$ such that $E_{out}(C) \oplus E_{out}(\tilde{C}^j) = \Delta_{out}$ 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. Then the match is performed on the relation $\hat{C}^i = \tilde{C}^j$.

Furthermore, since the transition $\Delta_{in} \to \Delta_{out}$, has probability $2^{-p}$, we will have to repeat the procedure $2^p$ times with $2^p$ 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 differential characteristic. When this is the case, we will get a collision between a $\hat{C}^i$ of the upper part and a $\tilde{C}^j$ of the lower part. Each collision $(i, j)$ will give us a candidate key.

For each $P_l$, we have $2^{|k_{in}|+|k_{out}|}$ candidate pairs $(\hat{C}_l^i, i)$ and $(\tilde{C}_l^j, j)$ for a collision. After matching through the relation $\hat{C}_l^i = \tilde{C}_l^j$, we still have $2^{|k_{in}|+|k_{out}|-n}$ candidates.

**Complexity** The time complexity for the computation done in the upper and lower part of the procedure is $2^{|k_{in}|+p}$ and $2^{|k_{out}|+p}$ respectively. And as explained above, the number of expected candidates key is $2^{|k_{in}|+|k_{out}|-n-|k_{in}\cap k_{out}|+p}$. Thus the time complexity is :

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

With this, we recover $k_{in} \cup k_{out}$ , so if we expect fewer key candidates than the whole set $k_{in} \cup k_{out}$ 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 an 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 $\mathscr{T}$ . In the case we need to guess the remaining bits of the master, namely if $|k_{in}| + |k_{out}| - n - |k_{in} \cap k_{out}| + p > 0$ then the total time complexity is thus :

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

The data complexity is

$$\mathscr{D} = \min(2^n, 2^{p+\min(|k_{in}, k_{out}|)}).$$

And the memory complexity is

$$\mathscr{M} = 2^{\min(|k_{in}|-|k_{in} \cap k_{out}|, |k_{out}|-|k_{in} \cap k_{out}|)},$$

by guessing the common key bit before running the attack.

## 2.3 Improvement : Parallel partitions for layers with partial subkeys

In [8], they give two ways to improve the basic differential MITM attack. One way to add an extra round at the beginning or the end the attack in some particular cases and the second way that allows to reduce the needed data.

In the case the round key addition only affect a part of the state as in SKINNY [5] or Gift [4], then the differential MITM attack can be extended by one round. Let's denote by $m < n$, the number of bits of the state affected by the round key addition. Moreover, if $p > m$ then the time complexity does not increase. The framework of the improvement is given in Figure 2.

The main idea of the improvement is to only keep the ciphertexts that satisfy the following condition : we fix the $n - m$ bits which are not affected by the key addition in the states before and after the round key addition, $X$ and $Y$, it will also fix some bits of the ciphertext. Thus now, we will repeat the procedure $2^{p-m}$ times.

Then from the $2^m$ possible $X$, we can proceed with the lower part of differential MITM attack. We get $2^{|k_{out}|+m}$ possible candidates. In parallel, we do the same for each of the $2^m$ possible $Y$, and we proceed with the upper part of differential MITM attack and we get $2^{|k_{in}|+m}$ possible candidates. But since we have to match $X$ and $Y$ and their associated pairs $\tilde{X}$ and $\tilde{Y}$ through the relation $X \oplus Y = \tilde{X} \oplus \tilde{Y}$ then the number of expected collision does not increase.

In [8], 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.

## 2.4 Reducing data with imposed conditions

The idea is to fix $x$ bits of the plaintext $P$ and of the associated plaintext $\tilde{P}$ to an unique value, using only $2^{n-x}$ plaintexts instead of the whole codebook. We can choose the plaintext $P$ as wanted but the probability to get a $\tilde{P}$ with the fixed $x$ bits is $2^{-x}$. Then the probability of the attack decreases of $2^x$ as we have to repeat the procedure $2^x$ times to get a pair that satisfy the differential characteristic. But it also means that, during the upper and lower part of the attack, we will discard $2^x$ 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 $2^x$.

Moreover, we can combined this techniques with the differential MITM attack which derive the two following inequalities on the number $x$ of bits we fix :

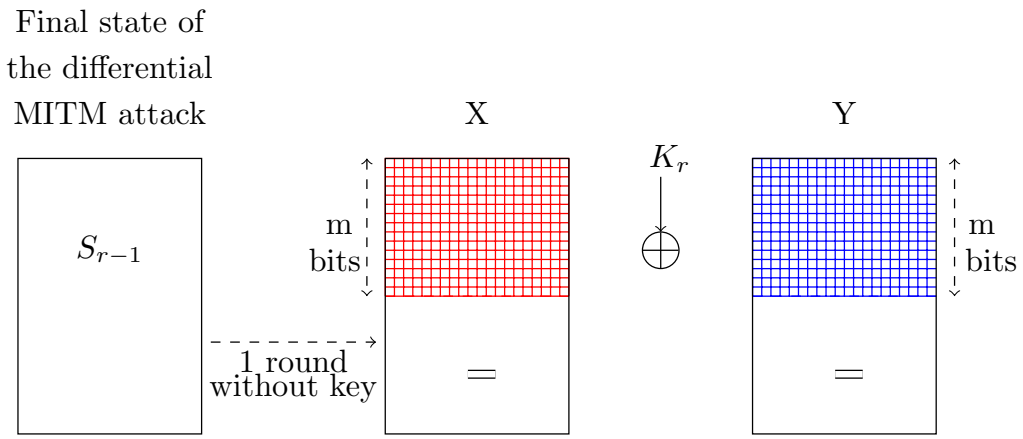


Figure 2: Framework of the one rounds extension improvement.

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

This technique particularly apply when the whole code book is needed before we fix the $x$ bits which is the case in the application on SKINNY-128-384 of the differential MITM attack presented in [8].

# 3 Improvements of the Differential MITM attack

In this section, I present the main results of my internship. Indeed, truncated differential cryptanalysis was introduced by Knudsen at FSE in 1994 [14] 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 [16, 2]. 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 us that since the probability of a truncated differential characteristic is not the same in both direction then it is not clear how to properly take this propriety into account in the truncated differential MITM extension.

Another idea we proposed is to improve the key recovery part of the attack. For this, we introduce a probabilistic part in the extended rounds before and after the differential distinguisher which will sometimes allow to reduce the overall time complexity of the attack. We will also apply the state-test technique introduced in [9], which allows to guess fewer key-bits of information and thus decrease the time complexity.

In the previous section, we have also seen a possible improvements of the differential MITM attack, that allows to extend one round without any time complexity cost. We generalized the partial partition improvement to extend the attack for one or more rounds and for ciphers for which the key is added to the whole state with little to no cost.

## 3.1 Truncated differential MITM

The main idea of the truncated differential MITM improvement is to use, instead of a differential path, a truncated differential path which only considers if a word has a difference on it or not, but not which one is the concrete difference. An advantage of truncated differential path is that during the key recovery part, we don't need to know the exact values of the state just before the characteristic and thus we might need to guess

less subkey words which can allow us to reach more rounds of a cipher. Also, for some ciphers truncated differential paths might reach more rounds than the differential paths.

**Framework of the truncated differential MITM attack** Similarly to the differential MITM attack, consider a $n$-bit cipher $E$ decomposed into three sub-ciphers: $E_{out} \circ E_m \circ E_{in}$, as depicted in Figure 3. Let the number of rounds of $E_{in}$, $E_m$ and $E_{out}$ be $r_{in}$, $r_m$ and $r_{out}$ respectively. Finally, let $\Delta_{in}$ be the input difference to the middle part $E_m$, $2^{\delta_{in}}$ be the number of possible differences $\Delta_{in}$, $\Delta_{out}$ the output difference of $E_m$, $2^{\delta_{out}}$ be the number of possible differences $\Delta_{out}$ and suppose that the differential $\Delta_{in} \rightarrow \Delta_{out}$, covering the $r_m$ middle rounds, has probability $2^{-p}$.

**Upper part.** We randomly pick a pair of plaintext/ciphertext $(P, C)$, and try to generate appropriate candidates for $(\tilde{P}, \tilde{C})$ such that the difference of these two data at round $r_{in}$ ensures the truncated difference $\Delta_{in}$ i.e. $E_{in}(P) \oplus E_{in}(\tilde{P}) \in \Delta_{in}$. To generate $\tilde{P}$, it is necessary to guess some key material, which is denoted by $k_{in}$. For each $P$, and each key guess $i$ of $k_{in}$, there are $2^{\delta_{in}}$ distinct candidates $\tilde{P}_l^i, l = 1, \ldots, 2^{\delta_{in}}$, corresponding to input difference $l \in \Delta_{in}$.

All the $2^{|k_{in}|} \times 2^{\delta_{in}}$ ciphertexts $\tilde{C}_l^i = E_K(\tilde{P}_l^i)$, along with its associated key material $i$ are stored in a hash table.

**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 $(\Delta_{in} \xrightarrow{E_d} \Delta_{out})$ will happen with probability $2^{-p}$. So, a number of $2^p$ pairs of data should be tested to find the correct key with high probability. For each $P$, and a fixed key material $i$, we can provide $2^{\delta_{in}}$ differential pairs $(P, \tilde{P})$. So, it is required to repeat the upper and lower parts of the attack for a number of $2^{p-\delta_{in}}$ plaintext $P$.

On the other hand, despite the concrete differential-MITM attack, the output difference $\Delta_{out}$ does not have a specific value, but it belongs to a set of size $2^{\delta_{out}}$.

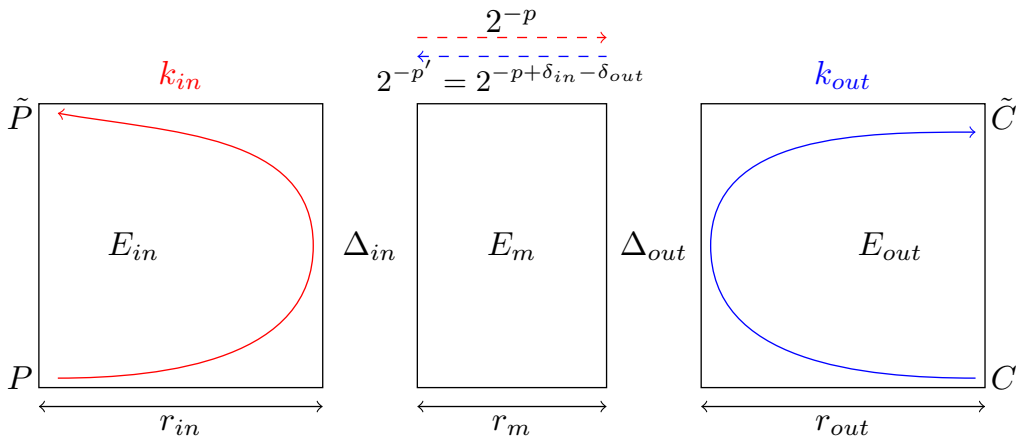


Figure 3: Framework of the truncated differential meet-in-the-middle attack. The probability $2^{-p}$ is the probability of the distinguisher in the forward direction and the probability $2^{-p'}$ is the probability in the backward direction.

**Time complexity** As done in the differential MITM attack, we first guess the possible linear relations between $k_{in}$ and $k_{out}$. During the upper part of the attack, for each guess of $k_{in}$, we get $2^{\delta_{in}}$ candidates $\tilde{P}_l^i$. Thus we have $2^{|k_{in}|+\delta_{in}}$ candidates for $(P, \tilde{P}, i)$. Similarly, in

the lower part, we have $2^{|k_{out}+\delta_{out}}$ candidates triplets for $(C, \tilde{C}, j)$. Let's note $\hat{C} = E(\tilde{(P)})$. Then, we match the $\hat{C}_l^i$ and the $\tilde{C}_m^j$, which leave us with $2^{|k_{in}|+\delta_{in}+|k_{out}|+\delta_{out}-n}$ candidates. Thus we now have $2^{|k_{in}|+\delta_{in}+|k_{out}|+\delta_{out}-|k_{in}\cap k_{out}|-n}$ candidates. Moreover, 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 :

$$\begin{aligned}\mathscr{T} &= 2^{p-\delta_{in}} \times 2^{|k_{in}\cap k_{out}|}(2^{|k_{in}|+\delta_{in}-|k_{in}\cap k_{out}|} + 2^{|k_{out}|+\delta_{out}-|k_{in}\cap k_{out}|}) \\ &+ 2^{p-\delta_{in}} \times 2^{|k_{in}\cap k_{out}|}(2^{|k_{in}|+\delta_{in}+|k_{out}|+\delta_{out}-2|k_{in}\cap k_{out}|-n}) \\ &+ 2^{p-\delta_{in}}(2^{k-n+\delta_{in}+\delta_{out}}) \end{aligned} \tag{9}$$

The data and memory complexities are similar of the differential MITM ones. Indeed, we have that :

The data complexity is

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

and the memory complexity is

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

## 3.2 Improving the structures for round extension

In [8] the authors proposed a way of extending the attack of one round in the cases where the round key addition affect only a part of the state of the cipher, as it was the case for SKINNY [5] or Gift [4]. They managed to reach one more round with out increasing the complexity in their attacks on SKINNY-128 thanks to this.

In this section we will explain how this round extension using structures can be generalized in the case of SPN ciphers that add the key to the whole state as will be shown in our application on CRAFT on section 4, and that it can potentially add more than one round as we will show how to cover two rounds with this technique in the attack on SKINNY-64 on sections 4.

*The general idea.* The state is formed of $W$ words of size $s$ bits. The idea is to fix $F$ words of the internal state that will imply, possibly thanks to the already known keybits, $F$ fixed words in the left and in the right parts, so that at the start of both the backward and forward parts we only have $2^{(W-F)s}$ possibilities for each. Remark that we can fix the $F$ words at different points of the round functions. On figure 4 we have represented the situation when the additional rounds are added at the end (they could also be added to the beginning, with out loss of generality).

From the $F$ fixed words of internal state between the states $X$ and $Y$ in figure 4, we compute the $F$ fixed word in $X$ by possibly guessing a few subkey bits increasing $|k_{out}|$ and similarly we compute the $F$ fixed words in $Y$ by possibly guessing some subkey bits increasing $|k_{in}|$. Note that if the fixed words are at one extreme, we only need to guess additional keybits at the other parallel computation, so many trade-off are possible. Thus we have $2^{(W-F)s}$ states at the input $X$ of the structure and $2^{(W-F)s}$ at the output $Y$. While the size of the structure is smaller than the number of repetitions needed for verifying the probability, namely while $(W-F)s \leq p$, we can perform the MITM attack independently for each $2^{(W-F)s}$ possible starting point with out any additional time, and in the end try to merge both partial list of solutions with respect to linear equations we might have remaining between the unknown words, the red part of $X$ and the blue part of $Y$ in figure 4. We have to ensure that the sieving that we have left is enough to discard enough candidates to retain a number well below exhaustive search. To do this we have to match the $X$ and $Y$, as well as their associated pairs $\tilde{X}$ and $\tilde{Y}$ that remain coherent.

In order to profit of the whole sieving potential, we would need to find $2(W-F)s$ linear relations between the pairs $X$ and $Y$, and $\tilde{X}$ and $\tilde{Y}$. We consider that we have $L$ linear relations available to sieve, that are upper bounded by $2(W-F)s$, and will depend on the particular cases. We can sieve in addition $2^{-Fs}$ as we can fully test that $\tilde{X}$ and $\tilde{Y}$ verify the relations at the $F$ positions.

We will need to repeat the procedure of fixing $F$ words with different values a number of times given by $2^{p-(W-F)s}$ to ensure we get a pair that satisfy the differential.

Now in the upper and lower parts of the attack we obtain $2^{|k_{in}|+(W-F)s}$ and $2^{|k_{out}|+(W-F)s}$ candidates respectively.

The time complexity is then :

$$\begin{aligned}\mathscr{T} &= 2^{p-(W-F)s} \times 2^{|k_{in} \cap k_{out}|}(2^{(W-F)s} \times 2^{|k_{in}|-|k_{in} \cap k_{out}|} + 2^{(W-F)s} \times 2^{|k_{out}|-|k_{in} \cap k_{out}|} \\ &+ 2^{|k_{in}|+|k_{out}|+2(W-F)s-Fs-L-2|k_{in} \cap k_{out}|}).\end{aligned}$$

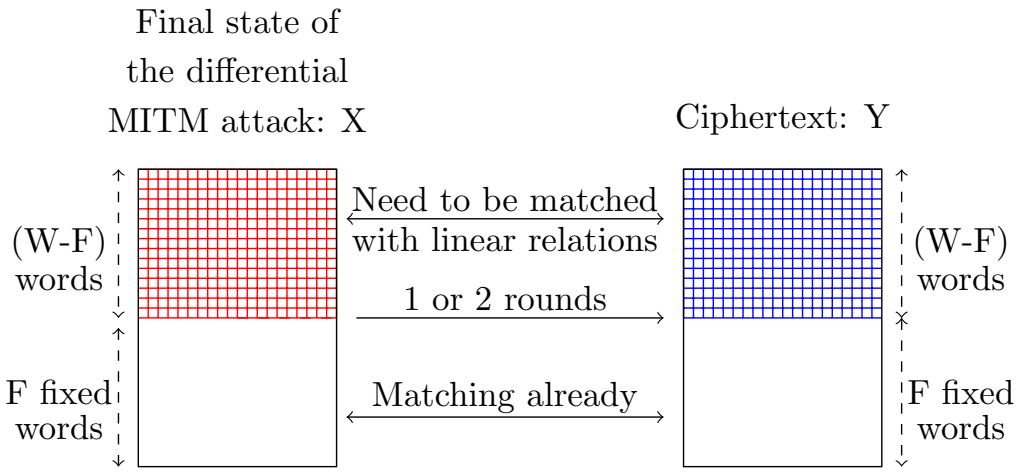


Figure 4: High level representation of the proposed structures when added at the end of the attack. They could also be added at the input without loss of generality.

Let us show two particular examples of how this can be done for more than one round, or even when the key is added to the whole state.

**The case of SKINNY.** In section 4, we add two rounds at the end of our truncated differential MITM attack on SKINNY-64-192 as shown in figure 9. For the cipher SKINNY, knowing the first key row of the first added key permits to check all the linear equations needed to profit of the whole sieving potential. In our attack, we construct structures of size $2^{40}$, thus to profit of the whole sieving potential, we want to find 10 linear relations between the pairs $X$ and $Y$, and $\tilde{X}$ and $\tilde{Y}$. In our case, we guessed 3 subkey nibbles of the first key row of the first key which give 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.

**The case of Craft.** In our attack of CRAFT, 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 $MC(X) \oplus P^{-1}(SB^{-1}(Y)) = MC(\tilde{X}) \oplus P^{-1}(SB^{-1}(\tilde{Y}))$ for all the non-fixed words, which erases the unknown key bits. Since we fixed 6 words in $X$ and $Y$, we get 10 linear relations on 4 bits each.

### 3.3 Key guessing improvement

Given a pair of plaintext/ciphertext ($P$,$C$) the parallel key guessing step aims at obtaining $\tilde{P}$ (respectively $\tilde{C}$) such that $P$ and $\tilde{P}$ (resp. $C$ and $\tilde{C}$) generate the input (resp. output) differential for each possible value of $k_{in}$. As said previously, we want to minimize the information guessed during this part as it has a direct impact in the complexity. The two following improvements allow us to guess fewer key information than $k_{in}$ and $k_{out}$ in the classical approach. In the first technique, we guess a word of the state instead of the key materials thus reducing the the total amount of bits we need to guess and the second technique introduces a probability on the key-guessing part to reduce the number of active words in the external part of the path, and therefore of the keybits involved.

**Applying the state-test technique** The state-test technique was introduced in [9] in the context of impossible differential attacks 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, then 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 section 4.

**Key guessing with probability** Usually in the key recovery part of a differential attack, we extend with probability one, the differential characteristic for some rounds in both sides.

The idea is to force one or more transition to have no difference paying some probability. Thus the number of active words will decrease and fewer key bits will be needed.

Suppose we are in the case of a truncated differential attack. In the classical case, we extend $\Delta_{in}$ and $\Delta_{out}$ for $r_{in}$ and $r_{out}$ rounds respectively and we generate pairs of plaintexts/ciphertexts that verify a difference $\Delta_{in}$ after $r_{in}$ rounds and a difference $\Delta_{out}$ after $r_{out}$ rounds with probability 1. And then test those pairs to see if they verify the differential characteristic when the key guess is correct.

Here instead of extending $\Delta_{in}$ and $\Delta_{out}$ for $r_{in}$ and $r_{out}$ rounds with probability 1, we wait for some transition that happen with probability $2^{-p_{in}}$ and $2^{-p_{out}}$, that means that given a pair ($P$, $\tilde{P}$) the probability of obtaining $\Delta_{in}$ after $r_{in}$ rounds is $2^{-p_{in}}$ and similarly for $C$,$\tilde{C}$, $\Delta_{out}$ and $2^{-p_{out}}$, to reach a difference $\Delta_{in}$ and $\Delta_{out}$ after $r_{in}$ and $r_{out}$ rounds respectively. Thus the overall probability for a random pair to follow the differential path is now $2^{-p-p_{in}-p_{out}}$. Therefore we now have to repeat the attack $2^{p_{in}+p_{out}}$ more times. However the number of pairs we keep decreases of $2^{p_{in}+p_{out}}$ so the time complexity stays unchanged.

We will show an application of this improvement in the case of the cipher CRAFT in section 4.

## 4 Applications of the improvements

In this section, we provide two applications of our newly introduced techniques and improvements. We will use our truncated differential MITM technique to attack 23 rounds of SKINNY-64 matching the best known attacks with a trade-off allowing a slightly better

time and data complexities and 22 rounds of CRAFT improving by one round the best known attack [12]. The structure improvement has allowed us to extend by two rounds the attack on SKINNY and by one round the attack on CRAFT.

## 4.1 Attack on SKINNY-64-192

SKINNY [5] introduced in 2016 is a popular cipher that has generated a lot of interest from cryptanalysts. In this subsection, we will first give a description of the cipher SKINNY and then provide the scheme of our attack.

The best known attack on SKINNY-64-192 [6] published at Crypto 2022 reaches 23 rounds. While our attack reaches the same number of rounds, the time and data complexities are lower.

**Description of SKINNY** SKINNY is a family of tweakable lightweight block ciphers designed by Beierle et al. [5], with block size 64 bits or 128 bits and a tweakey ([13] key plus tweak) of size $n$, $2n$ or $3n$ bits where $n$ is the size of the block. The exact specification of the cipher depends on the bloc and key sizes. Here we consider SKINNY-64-192, i.e. with a block size 64-bits and a key size 192-bits. The ciphers follows a SPN structure with a very compact S-box, a linear layer based on a sparse non-MDS binary matrix and a lightweight key schedule. The state is seen as a $4 \times 4$ 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 is as follows.

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

The round function of SKINNY is depicted on Figure 5, and this function is iterated 40 times in our case for the full version. One round of SKINNY is composed of five operations applied in the following order: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR) and MixColumns (MC).

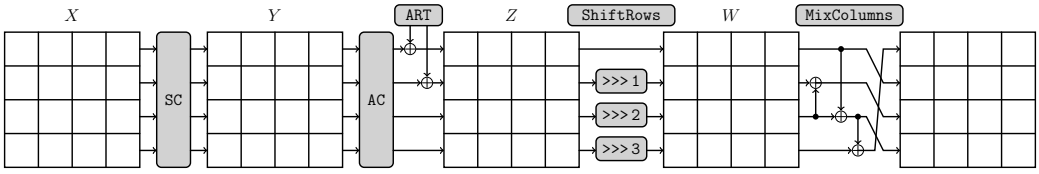


Figure 5: Round function of SKINNY.

1. SubCells. A 4-bit Sbox is applied to every cell of the cipher 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 |
| $S^{-1}(x)$ | 3 | 4 | 6 | 8 | c | a | 1 | e | 9 | 2 | 5 | 7 | 0 | b | d | f |

2. AddRoundTweakey (ART). In our case, when the key is of size $3n$, the first and second rows of the 3 tweakey arrays **TK1**, **TK2** and **TK3** are extracted from the tweakey with respect the the key schedule describe in paragraph 4.1 and XORed to the first two rows the internal state, respecting the bit positions inside the arrays.

3. ShiftRows (SR). The rows of the cipher state cell 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.

4. 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):

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

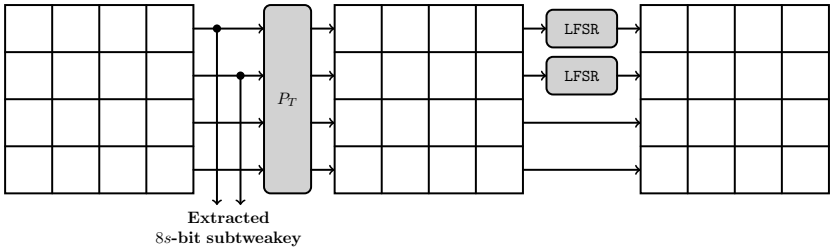


Figure 6: The tweakey schedule in SKINNY. Each tweakey word $TK1$, $TK2$ and $TK3$ follows a similar transformation update, except that no LFSR is applied to $TK1$.

**SKINNY key schedule** The key schedule is illustrated on figure 6, is as follow :

First, a permutation $P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ is applied on the cells positions of all the key arrays, i.e. for all $0 \leq i \leq 15$ we have $TK1_i \leftarrow TK1_{P_T(i)}$, $TK2_i \leftarrow TK2_{P_T(i)}$ and $TK3_i \leftarrow TK3_{P_T(i)}$. Moreover, every cell of the first and second rows of $TK2$ and $TK3$ are individually updated with an LFSR given by Table 1.

Table 1: The LFSRs used to generate the round constants of SKINNY-64-192.

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

Since SKINNY key schedule is linear, it is more efficient to guess some subkey bits and to 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 3 master key nibble, one for each of the **TK** and they all have the same index.

**An attack on SKINNY-64 without structures** First we provide the scheme of our attack on 21 rounds of SKINNY-64-192 and then we will show how to extend the attack for two rounds. The main idea of our attack is to extend a 9-rounds truncated differential characteristic showed in Figure 7, found by Zahra Ahmadian, Akram Khalesi and Hossein Moghimi using a MILP program, for six rounds on both sides. The scheme of the attack is shown on Figure 8. We are able to extend the attack by adding two rounds at the end using the structure procedure of section 3.2. The attack parameters are the following:

$$
\begin{aligned}
p &= 52 \\
\delta_{in} &= 4 \\
\delta_{out} &= 8 \\
|k_{in}| &= 128 \\
|k_{out}| &= 116
\end{aligned}
$$

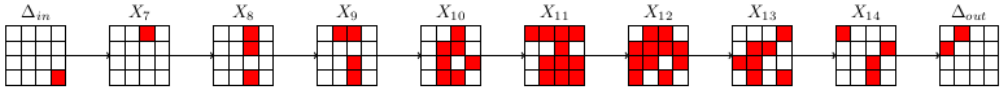


Figure 7: 9-round truncated differential characteristic of probability $2^{52}$ for Skinny-64.

On Table 2, we describe all the subkey nibbles of $k_{in}$ and $k_{out}$ needed in the 21-round attack of SKINNY-64 that can be derived from Figure 8. And we also give, for each nibble of the master key, which subkey nibble depends on them. We also give the number of relation we have between $k_{in}$ and $k_{out}$ for each nibble of the master key.

Table 2: Subkey nibbles involved in the 21-round attack of SKINNY-64-192.

| $TK_i$ 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 | $K_0[3]$ | $K_{18}[7]$, $K_{20}[1]$ | 0 |
| 4 | $K_0[4]$, $K_2[6]$ | , $K_{20}[5]$ | 0 |
| 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 | $K_1[2]$, $K_3[4]$ | $K_{19}[4]$ | 0 |
| 9 | $K_1[0]$ | $K_{17}[0]$, $K_{19}[2]$ | 0 |
| 10 | $K_1[4]$ | $K_{19}[6]$ | 0 |
| 11 | $K_1[7]$, $K_3[1]$ | $K_{19}[1]$ | 0 |
| 12 | $K_1[6]$ | $K_{19}[5]$ | 0 |
| 13 | $K_1[3]$ | $K_{17}[3]$, $K_{19}[7]$ | 0 |
| 14 | $K_1[5]$, $K_3[3]$ | $K_{19}[3]$ | 0 |
| 15 | $K_1[1]$, $K_3[0]$ | $K_{17}[1]$, $K_{19}[0]$ | 1 |

**State-test technique.** We use the state-test technique to reduce $|k_{in}|$ and $|k_{out}|$ by testing the 5 nibbles of Table 3, instead of guessing the subkeys nibbles described on the table. Thanks to this technique, the number of bits in $k_{in}$ decreases of 12 bits and the number of bits in $k_{out}$ decreases of 8 bits.

**Attack steps** We describe now the core attack on 21 rounds of SKINNY-64-192. The guesses needed for this attack are given in Table 2 and 3.

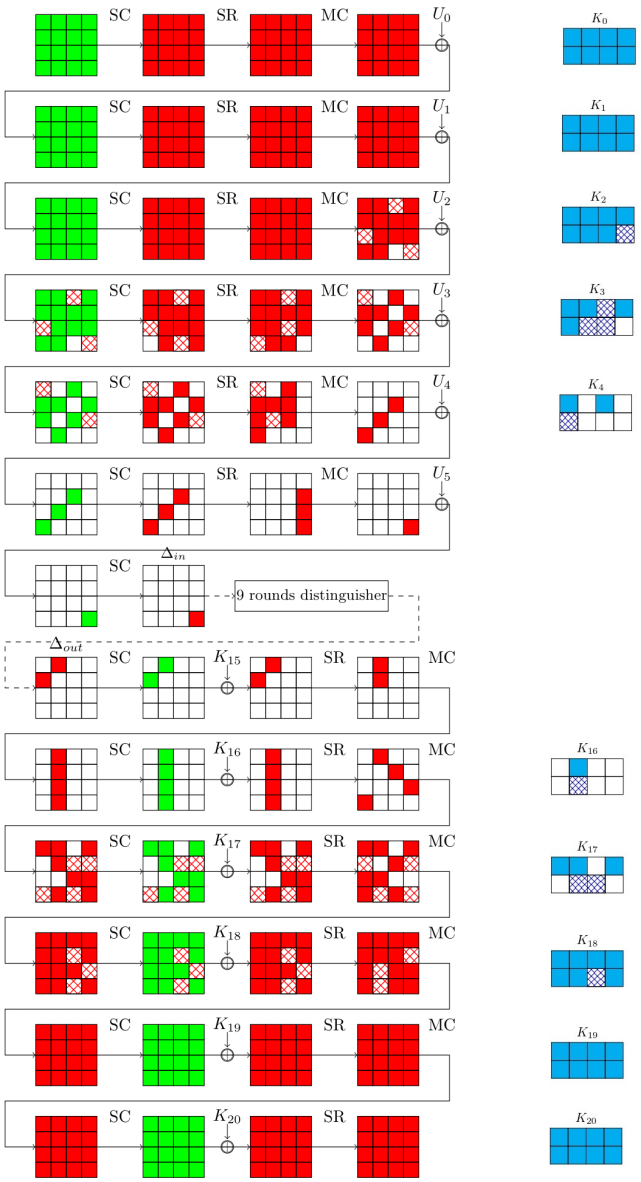


Figure 8: Core attack against 21 rounds of SKINNY-64/192 from round 0 to round 20 included. Knowledge of blue key nibbles allows to compute values of green ones and thus to propagate differences. Two blue stripped key nibbles are needed but are not guessed as we test the state nibbles instead as in Table 3. No difference in both white and stripped red nibbles, but the red ones are required to compute green nibbles. The equivalent subkeys $U_i$ are computed as $\mathrm{MC}(\mathrm{SR}(K_i))$, from the original subkeys $K_i$.

Table 3: Linear relations used in the state-test technique on the 21-round and 23-round attack of SKINNY-64-192.

| Cell wanted | RoundKey byte involved | Bytes needed from the precedent state |
|---|---|---|
| $x_4^2$ | $K_3^2$, $K_2^7$ | $K_3^2 \oplus SC(x_3^2) \oplus SC(x_3^{15}) \oplus SC(K_2^7 \oplus SC(x_2^7) \oplus SC(x_2^{10}))$ |
| $x_4^10$ | $K_3^5$, $K_2^7$ | $K_3^5 \oplus SC(x_3^5) \oplus SC(K_2^7 \oplus SC(x_2^7) \oplus SC(x_2^{10}))$ |
| $x_5^9$ | $K_4^4$, $K_3^6$ | $K_4^4 \oplus SC(x_4^4) \oplus SC(K_3^6 \oplus SC(x_3^6) \oplus SC(x_3^9))$ |
| $x_{17}^5$ | $K_{17}^5$,$K_{18}^6$ | $K_{17}^5 \oplus SC^{-1}(x_{18}^{10}) \oplus SC^{-1}(x_{18}^{14}) \oplus SC^{-1}(K_{18}^6 \oplus SC^{-1}(x_{19}^7) \oplus SC^{-1}(x_{19}^{11}) \oplus SC^{-1}(x_{19}^{15}))$ |
| $x_{16}^5$ | $K_{16}^5$,$K_{17}^6$ | $K_{16}^5 \oplus SC^{-1}(x_{17}^{10}) \oplus SC^{-1}(x_{17}^{14}) \oplus SC^{-1}(K_{17}^6 \oplus SC^{-1}(x_{18}^7) \oplus SC^{-1}(x_{18}^{11}) \oplus SC^{-1}(x_{18}^{15}))$ |

1. Ask for the encryption of the whole codebook (we will explain later how to apply the data reduction of subsection 2.4 to this case).
2. Pick one plaintext/ciphertext pair ($P$, $C$).
3. First we guess the 12 bits of the three subkey words that are entirely determined for both $k_{in}$ and $k_{out}$ related to $TK[1]$ as can be seen on Table 2. We also guess the 40 subkey bit common relations shared between $k_{in}$ and $k_{out}$ given by Table 2 and 3.
4. Compute all possible tuples $(P, \tilde{P}, i)$ for each of value $i$ of the remaining 76 bits of $k_{in}$ such that there is a difference after the 6th S-box layer on the active nibble. At the end of this step we have $2^{76+4}$ possible candidates.
5. For all $\tilde{P}$, computed $E(\tilde{P}) = \hat{C}$ and store them in a hash table.
6. Similarly, for each value $j$ of the remaining 64 bits of $k_{out}$, compute all possible tuples $(C, \tilde{C}, j)$ 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 $2^{64+8}$ possible candidates for the tuples $(C, \tilde{C}, j)$ .
7. 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.
8. Repeat from Step 1 until the right key is retrieved.

**Attack on 23-round of SKINNY-64-192** Now we will explain how to extend for two rounds the 21-round attack thanks to the improvement with the structures. On figure 9, we give the scheme of the two last rounds of the attack. We represent in blue the internal state and the subkey words that are necessary to build the structure and compute the upper part of the attack. And similarly we represent in blue the internal state and the subkey words needed to build the 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.

**Procedure of the attack on 23-round of SKINNY-64**

1. We fix the values of the words $F$, $A \oplus B$ and $C \oplus D$ of $W_{21}$ in Figure 9, and of the words $a \oplus d$ and $b \oplus d$ of $Y_{22}$ in Figure 9.
2. We guess the 12 bits of the three subkey words that are entirely determined for both $k_{in}$ and $k_{out}$ and we also guess the 56 subkey bits relations shared between $k_{in}$ and $k_{out}$ given in table 4 and 3.

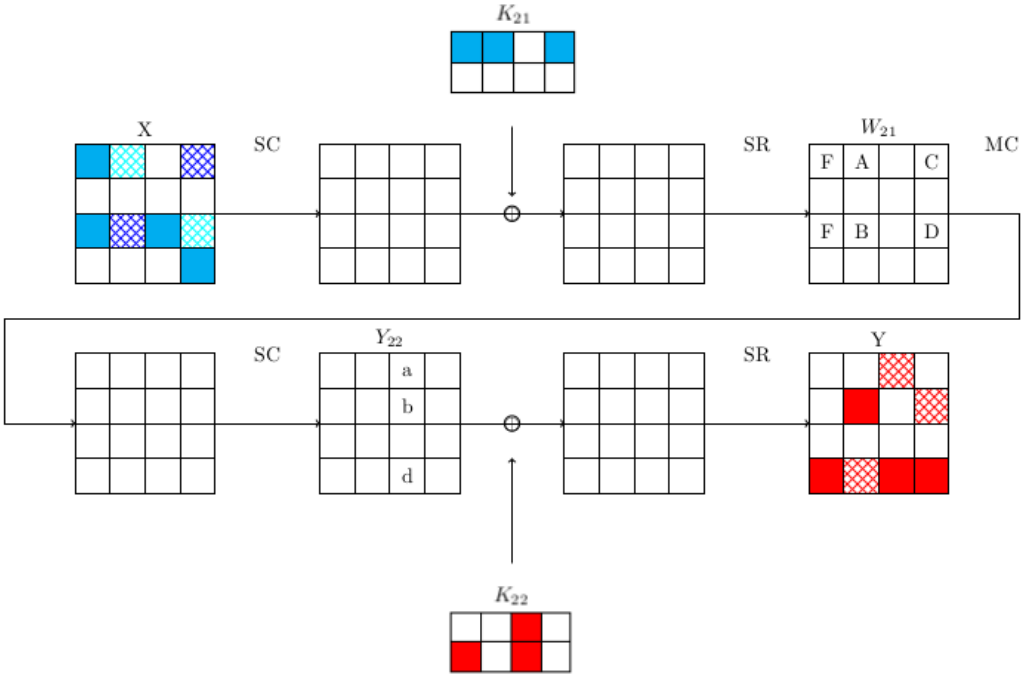


Figure 9: The two last rounds of the 23-round attack on SKINNY-64. The XOR of the following nibbles $X[1] \oplus X[11]$, $X[3] \oplus X[9]$, $Y[2] \oplus Y[13]$ and $Y[7] \oplus Y[13]$ are fixed.

3. Then for each value $i$ of the remaining 64 bits of $k_{in}$, we can compute from the values of step 1, the fixed nibbles $Y[4, 12, 14, 15]$, $Y[2] \oplus Y[13]$ and $Y[7] \oplus Y[13]$ as shown in Figure 9. Then we have $2^{40}$ possible states for $Y$ as we have imposed the values of 6 nibbles thus there are 10 nibbles that can take all possible values. For all the possible $Y$, we compute the ciphertext and we get the corresponding plaintext $P$.

4. Compute all possible tuples $(P, \tilde{P}, i)$ for each value $i$ of $k_{in}$ and each $P$ from the structure defined at the previous step such that there is a difference after the 6th S-box layer on the active nibble. At the end of this step we have $2^{40} \times 2^{64+4} = 2^{108}$ possible candidates.

5. Store them in a hash table.

6. Similarly, for each value $j$ of $k_{out}$, we can compute from the values of step 1, the fixed values $X[0, 8, 10, 15]$, $X[1] \oplus X[11]$ and $X[3] \oplus X[9]$, in red in 9. We then pick all the $2^{40}$ possible states of $X$ and we compute the $2^{40}$ possible $S_{20} = MC^{-1}(X)$.

7. For each value $j$ of $k_{out}$ and each state $S_{20}$, compute all possible tuples $(S_{20}, \tilde{S}_{20}, j)$ 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 $2^{40} \times 2^{60+8-8} = 2^{100}$ possible candidates for the tuples $(S_{20}, \tilde{S}_{20}, j)$.

8. Check for possible matches on the hash table. The match is performed on two quantities:

   – the values of the nibbles the values $\tilde{W}_{21}[0]$, $\tilde{W}_{21}[8]$, $\tilde{W}_{21}[1] \oplus \tilde{W}_{21}[9]$, $\tilde{W}_{21}[3] \oplus \tilde{W}_{21}[11]$, $\tilde{Y}_{22}[2] \oplus \tilde{Y}_{22}[14]$ and $\tilde{Y}_{22}[6] \oplus \tilde{Y}_{22}[14]$ need to verify the same relations as the originally fixed nibbles that we can check : a $6 \times 4 = 24$ bit filter;

Table 4: Subkey nibbles involved in the 23-round attack of SKINNY-64-192.

| $TK_i$ 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 | $K_0[3]$ | $K_{18}[7]$, $K_{20}[1]$ | 0 |
| 4 | $K_0[4]$, $K_2[6]$ | , $K_{20}[5]$ | 0 |
| 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 | $K_1[2]$, $K_3[4]$ | $K_{19}[4]$ | 0 |
| 9 | $K_1[0]$ | $K_{17}[0]$, $K_{19}[2]$ | 0 |
| 10 | $K_1[4]$ | $K_{19}[6]$ | 0 |
| 11 | $K_1[7]$, $K_3[1]$ | $K_{19}[1]$, $K_{21}[0]$ | 1 |
| 12 | $K_1[6]$ | $K_{19}[5]$, $K_{21}[3]$ | 0 |
| 13 | $K_1[3]$ | $K_{17}[3]$, $K_{19}[7]$, $K_{21}[1]$ | 1 |
| 14 | $K_1[5]$, $K_3[3]$ | $K_{19}[3]$ | 0 |
| 15 | $K_1[1]$, $K_3[0]$ | $K_{17}[1]$, $K_{19}[0]$ | 1 |

  - the linear relations between $X$ and $W_{22}$ and between $\tilde{X}$ and $\tilde{W}_{22}$ : a 80-bit filter ($10 \times 2$ equations on 4 bits each).

9. Repeat from Step 1 with different values for the fixed nibbles until the right key is retrieved.

**Linear relations to match with the structures improvement.** At the end of the attack procedure we have to match $X_{21}$ and the ciphertext and their associated pair $\hat{X}_{21}$ and $\tilde{C}$. After two rounds of SKINNY-64, if we know the 4 first subkey nibbles then the relations between the words of $X$ and $Y$ and $\tilde{X}$ and $\tilde{Y}$, of Figure 9, are linear. In our application, we have guessed the subkey nibbles $K^0_{21}$,$K^1_{21}$ and $K^3_{21}$. We don't know the subkey nibble $K^2_{21}$ but we have guessed the subkey nibbles $K^2_{22}$ and $K^6_{22}$ and thus we obtain the linear relations of the 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 relation (given for the pair ($X$,$Y$) but are also true for $(\tilde{X}, \tilde{Y})$) to match are as follows:

$$\begin{cases} SB(SB(X^0_{21}) \oplus K^0_{21} \oplus SB(X^{10}_{21}) \oplus SB(X^{13}_{21})) & = W^0_{22} \oplus K^0_{22} \\ SB(X^7_{21}) \oplus K^7_{21} \oplus SB(X^{10}_{21}) & = SB^{-1}(W^8_{22}) \end{cases} \tag{10}$$

$$\begin{cases} SB(SB(X^1_{21}) \oplus K^1_{21} \oplus SB(X^{11}_{21}) \oplus SB(X^{14}_{21})) & = W^1_{22} \oplus K^1_{22} \\ SB(SB(X^1_{21}) \oplus K^1_{21}) & = W^5_{22} \oplus K^5_{22} \\ SB(X^4_{21}) \oplus K^4_{21} \oplus SB(X^{11}_{21}) & = SB^{-1}(W^9_{22}) \end{cases} \tag{11}$$

$$\begin{cases} SB(X^2_{21}) \oplus K^2_{21} & = SB^{-1}(W^6_{22} \oplus K^6_{22}) \\ SB(X^5_{21}) \oplus K^5_{21} \oplus SB(X^8_{21}) & = SB^{-1}(W^{10}_{22}) \end{cases} \tag{12}$$

$$\begin{cases} SB(SB(X_{21}^3) \oplus K_{21}^3 \oplus SB(X_{21}^9) \oplus SB(X_{21}^{12})) & = W_{22}^3 \oplus K_{22}^3 \\ SB(SB(X_{21}^3) \oplus K_{21}^3) & = W_{22}^7 \oplus K_{22}^7 \\ SB(X_{21}^6) \oplus K_{21}^6 \oplus SB(X_{21}^9) & = SB^{-1}(W_{22}^{11}) \end{cases} \quad (13)$$

Since the probability of the truncated differential characteristic is $p = 52$ and $\delta_{in} = 4$ then we have to build $2^{p-\delta_{in}} = 2^{48}$ pairs to find a good pair satisfying the differential. With a structure, we obtain $2^{40}$ pairs to test, thus we have to repeat the attack steps $2^{48-40} = 2^8$ times.

The time complexity is $2^{40} \times 2^{68} C_{in} = 2^{108} C_{in}$ for the upper part, $2^{40} \times 2^{68} C_{out} = 2^{108} C_{out}$ for the lower part and the time complexity of the sieving part is $2^{108+108-24-40\times 2} C_{sieve} = 2^{112} C_{sieve}$, where $C_{in}, C_{out}$ and $C_{sieve}$ are the time complexities of the upper part procedure,the lower part of the procedure and the sieving steps.

To reduce the data complexity of our attack, we use the improvement of [8] to reduce data with imposed conditions. Thus we fix the nibbles $C[6]$ and $C[1] \oplus C[13]$ and $\tilde{C}[6]$ and $\tilde{C}[1] \oplus \tilde{C}[13]$, which is a 8 bits condition. Thus we compute and build a stored table with the data we need to perform the attack to avoid doing the encryption or decryption during the upper part of the attack. Finally the data complexity of this attack is $\mathscr{D} = 2^{64-8} = 2^{56}$. The memory complexity is determined by Step 6 in which $2^{108}$ words of $64 + 64 = 128$ bits each are stored.

**In-depth analysis of the complexity :** We estimate $C_{in}, C_{out}$ and $C_{sieve}$ by following the common practice of counting the number of Sbox applications computed in the bottleneck part of the attack compared to the number of Sbox applications in the full cipher. Here we can count on parallel the number of active Sbox of each round. Thus $C_{in} = (2 + \frac{34}{368})$ but we don't have to consider the cost of the two encryption during the upper part since we use a stored table to get the decryption and encryption. The complexity of the lower part is $2^{12}(\frac{8}{368} + (2^{40} \times 2^{40+8} \times \frac{41}{368})$ since we compute the partial encryption of the added rounds and the lower part in parallel and the complexity of the sieving part is $C_{sieve} = (\frac{14}{368} + 2^{-4} \times (\frac{3}{368} + 2^{-4}))$ since there are 14 Sboxs to test if a key follows some transition of the differential path at round 11.

The time complexity is:

$$\begin{aligned} \mathscr{T} &= 2^8 \times 2^{12+56}(2^{104.57} + 2^{104.56} + 2^{107.32}) \\ &= 2^{180.57} + 2^{180.56} + 2^{183.32}, \end{aligned} \quad (14)$$

thus $\mathscr{T} = 2^{183.69}$.

Thanks to the truncated differential, we manage to extend the characteristic for more rounds than if it was a fixed differential characteristic since the subkey around the characteristic are not needed in the attack. Moreover in the case of SKINNY, we can use, with little cost, the structure improvement to reach two more rounds and thus improving the time and data complexities of the previous best known attack.

## 4.2 Attack on CRAFT

Using truncated differentials, structures for full key addition ciphers, probabilistic key guessing and state-test techniques we managed to provide the best attack on CRAFT [7],

**Description of CRAFT** CRAFT is a lightweight tweakable block cipher made out of involutory building blocks designed in [7]. It consists of a 64-bit block, a 128-bit key, and a 64-bit tweak. The state is seen as a $4 \times 4$ matrix of 4-bits cells and is denoted $I$. Row

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

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.

The round function of CRAFT is depicted on Figure 10. By initializing the state with the plaintext, the cipher iterates 31 round functions ($R_i$, $0 \leq i \leq 30$) and appends one more linear round $R'_{31}$) to compute the ciphertext. Each round function $R_i$ applies the following five involutory round operations: SubBox, MixColumn, PermuteNibbles, AddConstant$_i$ and AddTweakey$_i$, while $R'_{31}$ only applies the MixColumn, AddConstant$_i$ and AddTweakey$_i$ operations. The round operations are defined as follows :

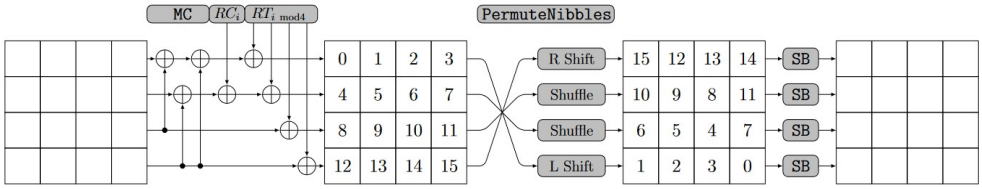


Figure 10: Craft round function.

1. MixColumn (MC): The following involutory binary matrix M is multiplied to each column of the state:

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

2. AddTweakey$_i$ ($RT_i$): Using a permutation $Q$ on the nibbles of the given tweak, the cipher derives four 64-bit tweakeys $TK0$, $TK1$, $TK2$ and $TK3$from the tweak T and the key $(K_0||K_1)$ as $TK_0 = K_0 \oplus T$ , $TK_1 = K_1 \oplus T$ , $TK_2 = K_0 \oplus Q(T)$ , $TK_3 = K_1 \oplus Q(T)$. Thereby, $Q(T)$ applies the permutation

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

on the nibbles of the tweak $T$ where for all $0 \leq i \leq 15$, $T_i$ is replaced by $T_{Q(i)}$. Then in each round $i$, without any key update, the tweakey $TK_i$ mod 4 is XOR-ed to the cipher state.

3. PermuteNibbles (P): An involutory permutation P is applied on the nibble positions of the state. In particular, for all $0 \leq i \leq 15$, $I_i$ is replaced by $I_{P(i)}$, where

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

4. SubBox (SB): The 4-bit involutory Sbox S is applied 16 each nibble of the state. Its table is given in the tabe below :

| $x$ | 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 |

**Security claim** In [7], 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 known attack on CRAFT [12] is a 21-round Impossible differential attack with time complexity of $2^{106.53}$, data complexity of $2^{60.99}$ and memory complexity of $2^{100}$.

**An attack on 22 rounds of CRAFT** For our attack, we use the 13-round truncated differential characteristic given in [11].

We extend the 13-round truncated differential characteristic for four rounds on both sides. The scheme of the attack is shown on figure 11. We then extend the attack by one round at the end using the structures procedure of section 4.1. Then the attack parameters are following :

$$\begin{aligned} p &= 50.28 \\ p_{in} &= 4 \\ p_{out} &= 4 \\ \delta_{in} &= 12 \\ \delta_{out} &= 12 \\ |k_{in}| &= |k_{out}| = 68. \end{aligned}$$

Since the matrix $M$ used in the MixColumn operation is involutory then the propagation of active cells in the upper and lower parts is almost the same. Thus, as shown in Table 5, the subkey words needed on both parts have many words in common thanks to the key schedule and the position of the odd and even subkeys, which allows us to efficiently construct structures.

For the key guessing steps of this attack, we will use the probabilistic key recovery rounds of section 4.2. For this we will impose that the difference during the MixColumn transition, on the word $X_1[0]$ for the upper part and on the word $Y_{18}[0]$ for the lower part is null which decrease the probability of getting a pair that verifies the truncated differential characteristics of $2^{-8}$ but the number of active words also decreases. Now the probability is $2^{-(50.28+8-12)} = 2^{-46.28}$.

Table 5: Subkey nibbles involved in the 22-round attack of CRAFT.

| Subkey | $k_{in}$ | $k_{out}$ | # equation |
|---|---|---|---|
| Even | $K_0[1,3,4,5,8,10,12,13,15]$ | $K_0[4,5,8,10,12,13,15]$ | 7 |
| Odd | $K_1[2,6,9,10,12,14]$ | $K_1[2,6,9,10,12,14]$ | 6 |

**State-test technique** We use the state-test technique to test the 6 nibbles of the state given below, instead of guessing more subkeys nibbles.

$$x_2^{12} = SB(K_1^1 \oplus x_1^1 \oplus x_1^9 \oplus x_1^{13})$$
$$x_2^{14} = SB(K_1^3 \oplus x_1^3 \oplus x_1^{11} \oplus x_1^{15})$$
$$x_{20}^1 = K_{20}^1 \oplus K_{20}^9 \oplus SB^{-1}(C^{12}) \oplus SB^{-1}(C^5) \oplus SB^{-1}(C^2) \oplus K_{20}^{13}$$
$$x_{20}^3 = K_{20}^3 \oplus K_{20}^{11} \oplus SB^{-1}(C^{14}) \oplus SB^{-1}(C^7) \oplus SB^{-1}(C^0) \oplus K_{20}^{15}$$
$$x_{19}^1 = K_{19}^1 \oplus K_{19}^{13} \oplus SB^{-1}(x_{20}^{12}) \oplus SB^{-1}(x_{20}^5) \oplus SB^{-1}(x_{20}^2) \oplus K_{19}^9$$
$$x_{18}^3 = K_{18}^3 \oplus K_{18}^{11} \oplus K_{18}^{15} \oplus SB^{-1}(x_{19}^{14}) \oplus SB^{-1}(x_{19}^7) \oplus SB^{-1}(x_{19}^0)$$

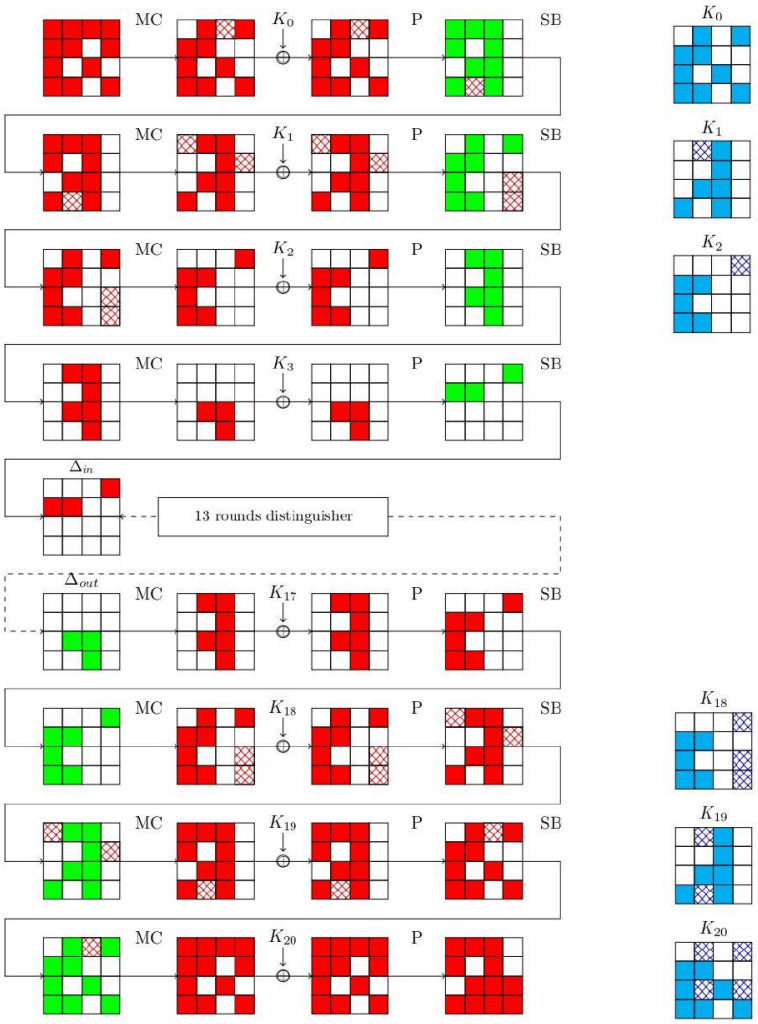


Figure 11: Core attack against 22 rounds of CRAFT. Knowledge of blue key nibbles allows to compute values of green ones and thus to propagate differences. The blue stripped key nibbles are needed to guess one green nibbles but are not directly guessed. No difference in both white and stripped red nibbles, but the stripped red ones are required to compute green nibbles.

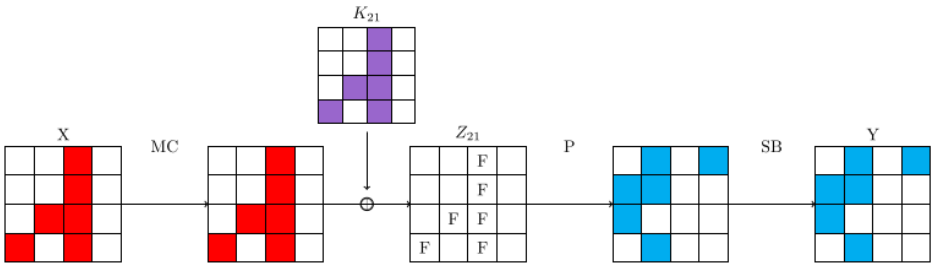


Figure 12: The last round of the 22-round attack on CRAFT. The blue nibbles are needed in upper part of the attack and the red nibbles are needed in the lower part of the attack. The purple subkey nibbles are known in both the upper and lower parts.

**Attack steps**

In Figure 11, we give the truncated differential path used in our attack. We need to know the active nibbles before the Sboxs, in green in figure 11.

1. We first guess the $13 \times 4 = 52$ subkey bit relations in the intersection of $k_{in}$ and $k_{out}$ given in Table 5 and in 4.2. We then fix the nibbles $Z_{21}[2, 6, 9, 10, 12, 14]$, the words $F$ in Figure 12 and since we know the subkey nibbles $K_{21}[2, 6, 9, 10, 12, 14]$ for both the upper and lower parts then we also have the fixed values of $X[2, 6, 9, 10, 12, 14]$ and $Y[2, 6, 9, 10, 12, 14]$.

2. Then we pick the $2^{40}$ possible $X$ and $Y$. We decrypt the ciphertext $Y$ and we obtain the associated plaintext.

3. For each value $i$ of $k_{in}$ and each $P$ from the structure defined at the previous step, compute all possible tuples $(P, \hat{P}, i)$ such that there is a difference after the 4th S-box layer on the three actives nibbles. At the end of this step we have $2^{40} \times 2^{16} \times 2^8$ candidates.

4. Store them in a hash table.

5. Similarly, for each value $j$ of $k_{out}$ and each state $X$, compute all possible tuples $(X, \tilde{X}, j)$ such that there is a difference on the state before the 17th S-box layer on the three actives nibbles. At the end of this step we have $2^{40} \times 2^{16} \times 2^8$ candidates for the tuples $(X, \tilde{X}, j)$.

6. Check for possible matches on the hash table. The match is performed on two quantities:

   – the filter on the values of $\tilde{Y}[1, 3, 4, 5, 8, 13]$ and $\tilde{X}[2, 6, 9, 10, 12, 14]$ that can be entirely computed: a 24-bits filter;

   – the linear relations between $MC(X)$ and $Z_{21}$ and between $MC(\tilde{X})$ and $\tilde{Z}_{21}$, i.e. $MC(X) \oplus Z_{21} = MC(\tilde{X}) \oplus \tilde{Z}_{21}$ : a 40-bit filter (10 equations on 4 bits each);

7. Repeat from Step 1 until the right key is retrieved.

Since the probability of getting a pair that verifies the characteristic is $2^{p+p_{in}+p_{out}-\delta_{in}}$, which is $2^{50.28+4+4-12} = 2^{-46.28}$, and each procedure gives us a structure of size $2^{40}$ then we have to repeat the procedure $2^{6.28}$ times. The data complexity of the attack is $2^{64}$ as we ask the whole codebook to the oracle. The memory complexity is determined by step 3 in which $2^{64}$ words of $64 + 16 + 8 = 88$ bits each is stored in the hash table.

The time complexity is $2^{52}$ for step 1, $2^{64}$ for computing the hash table, $2^{64}$ for performing Step 5. And we have $2^{64} \times 2^{64}$ pairs of tuples $(P, \hat{P}, i)$, $(S_{20}, \tilde{S}_{20}, j)$ that we match through the $6 \times 4$ bits of fixed words and the 40 bits of the linear relation $Y_{21} \oplus Z_{21} = \tilde{Y}_{21} \oplus \hat{Z}_{21}$ thus the time complexity of step 6 is $2^{64}$. Finally, the attack has to be repeated $2^{6.28}$ times in order to construct one right differential pair. Hence, the overall complexity of our attack is :

$$\begin{aligned} \mathscr{T} &= 2^{6.28} \times 2^{52}(2^{64} + 2^{64} + 2^{64+64-24-40}) \\ &= 2^{122.28} + 2^{122.28} + 2^{122.28} \\ &= 2^{122.28} \times 3, \end{aligned} \tag{15}$$

thus $\mathscr{T} = 2^{123.87}$.

To decrease the data complexity, we can apply generic improvement for the data reduction of [8], thus if we impose 6 bits of the ciphertext $C$ to be fixed and we expect the same from $\tilde{C}$ then the time complexity will not increase but the data complexity will become $2^{64-6} = 2^{58}$.

# 5 Conclusion

During this internship, I have studied different techniques of symmetric cryptanalysis such as the truncated differential attacks and the differential MITM attacks which is a combination of two important families of symmetric attacks. This has allowed me to get a better understanding of symmetric cryptanalysis, its importance and how to apply those techniques.

Moreover, during my internship, we found new techniques for improving the differential MITM attacks. The truncated differential allow, in some cases where the truncated paths are more efficient, to reach more rounds than the simple differential MITM attack. Furthermore, some of the improvements presented might be useful for others application such as the improvements on the key guessing part of the attacks. Indeed the extension of attacks through structures can be applied to any SPN cipher if we know or guess some key materials of the states we add and is and does not increase the complexity while covering more rounds when we can find linear relations between the last state of the lower part of the attack and the ciphertext. We also found that while decreasing the probability, adding a probability on the external paths of a differential characteristic allows to reduce the amount of key material we need to guess to construct a pair that follow the differential characteristic and can in some case reduce the overall time complexity.

The truncated differential MITM attack combined with the structure improvement on two rounds allowed us to mount an attack on 23 SKINNY-64-192 reaching 23 rounds with a lower time and data complexity than the last best known attack. And the application of the truncated differential MITM attack combined with the structure and the probabilistic improvement allowed us to mount the best know attack on CRAFT, reaching 22 rounds.

During my last week of internship, we got new truncated differential path to work on and we are confident that we have found an attack on 25 rounds of CRAFT with the same techniques used for our 22 rounds attack. And thus we will submit our results to the Eurocrypt conference in October. After this internship, I will continue, as a PHD student, to study the generalization of new cryptanalysis tools and the possible combination of existing cryptanalysis families. I will also try to understand the underlying algorithmic nature of the cryptographic problems and propose an algorithmic approach to those problems which would allow to optimize and improve the best known attacks.

## References


[1] Amirhossein Ebrahimi, M. and Ahmadian, Z.: New Automatic Search Method for Truncated-Differential Characteristics Application to Midori, SKINNY and CRAFT. COMPUTER JOURNAL 63, no. 12 (2020): 1813-1825.

[2] Ahmadian, Z., Rasoolzadeh, S., Salmasizadeh, M. and Reza Aref, M.: An improved truncated differential cryptanalysis of klein. Tatra Mountains Mathematical Publications, 67(1):135–147, 2016

[3] Avanzi, R.: "he QARMA block cipher family. Almost MDS matrices over rings with zero divisors, nearly symmetric Even-Mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Transactions on Symmetric Cryptology (2017): 4-44.

[4] 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.) CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer (2017)

[5] 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.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 123–153. Springer (2016)

[6] Bao, Z., Guo, J., Shi, D. and Tu, Y. : Superposition meet-in-the-middle attacks: updates on fundamental security of AES-like hashing. Annual International Cryptology Conference, 2022 Springer.

[7] Beierle, C., Leander, G., Moradi, A. and Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Transactions on Symmetric Cryptology. Ruhr-Universität Bochum, 2019.

[8] Boura, C. and David, N., Derbez, P., Leander, G. and Naya-Plasencia, M.: Differential meet-in-the-middle cryptanalysis. CRYPTO 2023.

[9] Boura, C. and Naya-Plasencia, M. and Suder, V.: Scrutinizing and improving impossible differential attacks: applications to CLEFIA, Camellia, LBlock and Simon. International Conference on the Theory and Application of Cryptology and Information Security, Springer. 2014.

[10] Dong, X., Hua, J., Sun, S., Li, Z., Wang, X., Hu, L. (2021). Meet-in-the-Middle Attacks Revisited: Key-Recovery, Collision, and Preimage Attacks. In: Malkin, T., Peikert, C. (eds) Advances in Cryptology – CRYPTO 2021. CRYPTO 2021. Lecture Notes in Computer Science(), vol 12827. Springer,

[11] Guo, H. and Zhang, Z. and Yang, Q. and Hu, L. and Luo, Y.: A New Method To Find All The High-Probability Word-Oriented Truncated Differentials: Application To Midori, SKINNY And CRAFT . The Computer Journal, 2023 Oxford University Press.

[12] Hadipour, H., Sadeghi, S., Eichlseder, M. (2023). 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. EUROCRYPT 2023. Lecture Notes in Computer Science, vol 14007. Springer,

[13] Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 274–288. Springer (2014)

[14] Knudsen, L.: Truncated and Higher Order Differentials,Fast Software Encryption Workshop,1994

[15] Tolba, M., Abdelkhalek, A., Youssef, A.M. (2017). Impossible Differential Cryptanalysis of Reduced-Round SKINNY. In: Joye, M., Nitaj, A. (eds) Progress in Cryptology - AFRICACRYPT 2017. AFRICACRYPT 2017. Lecture Notes in Computer Science(), vol 10239. Springer,

[16] Lallemand, V. and Naya-Plasencia, M.: Cryptanalysis of klein. In International Workshop on Fast Software Encryption, pages 451–470. Springer, 2014.