Differential Meet-in-the-Middle Attacks on Feistel Ciphers

Bastien Michel, Dounia M'foukh, María Naya-Plasencia

Inria, Paris, France
{bastien.michel,dounia.mfoukh,maria.naya_plasencia}@inria.fr

Abstract. Differential meet-in-the-middle attacks, introduced by Boura *et al.* in 2023, propose a new way of dealing with differential distinguishers. They allow, in particular, to combine differential attacks with initial structures, that were usually used exclusively for meet-in-the-middle attacks. Several applications of this new technique have been published, but so far the results on Feistel constructions have not improved much upon previous best known attacks. In this paper, we apply them to Feistel constructions including all the optimisations proposed so far, and we propose some additional new ideas to generically improve these kinds of attacks. We also propose an automated tool for optimising the attacks on Simon-like constructions. Our tool outputs a graphical representation of the attack that makes it very easy to verify. All this has allowed us to provide improved single-key key-recovery attacks on most of the variants of Simon and Simeck, that increase the highest number of rounds attacked by 1 or 2. In addition, we propose a new best attack on CLEFIA-256, reaching for the first time 16 rounds.

Keywords: symmetric cryptanalysis · differential meet-in-the-middle · Feistel · automatic cryptanalysis · full key-recovery · state-test · parallel partitioning · Simon · Simeck · CLEFIA

1 Introduction

Symmetric cryptanalysis is fundamental to assess the security margin of the constructions we use, and to be able to predict in advance when problems might appear. It is the base of the confidence we have in our constructions. Because of this, knowing how many rounds the best existent attacks cover over the most important constructions, and how far they are from reaching the full version is very important. Proposing new attacks remains a tedious task. Any effort towards generalising and automating this task is of great importance for improving the security and the security understanding of the used primitives.
To have a meaningful security margin, we need to evaluate the security of our constructions with respect to all known and newly proposed cryptanalysis families. In this context, a new technique was proposed in 2023 [8]: the differential meet-in-the-middle (MITM) attack. This technique, that mixes differential and meet-in-the middle attacks, has been applied to several constructions, such as SKINNY and Craft [2], providing the best-known attacks. Some improvements were proposed in [28], and some new applications in [12,13,20].

So far, its applications to Feistel constructions have not significantly improved on previous attacks. We believe this is because many improvements proposed in differential MITM attacks were not always taken into account.

Our contributions. In this paper, we have raised the question of the efficiency of differential MITM attacks on Feistel constructions. In particular, we have studied the security of the SIMON family of ciphers, proposed in 2013 [4] by the NSA. In 2018, it was standardized by the International Organization for Standardization (ISO) for radio frequency identification devices (RFID). Due in part to the lack of rationale in its proposal, that raises questions about the intention of the choices in the design, it has received an enormous amount of attention from the cryptanalysis community, like [27,1,24,10,18,11,13,23]. The best-known attacks up to now against this family seemed to be in general linear attacks. We will show how differential MITM attacks considerably outperform those. We have also analysed the cipher SIMECK [32], that was proposed in 2015 and has a SIMON-like structure. Our attacks also outperform most of the best previously known ones, that used also linear cryptanalysis [18]. To efficiently cryptanalyse all the variants of these two families, we have implemented an automated tool that finds the best differential MITM attacks possible when given an existing differential distinguisher. This tool, that outputs a graphic representation of the attack (rendering it particularly easy to verify), takes into account all the proposed improvements, namely the state-test technique, probabilistic key-recovery extension, and, for the first time to the best of our knowledge, parallel partitions or initial structures. Our tool is available here: https://github.com/Babmichel/MILP_Model_for_Diff_MITM_on_Simonlike_cipher/blob/main/README

We have also analyzed the cipher CLEFIA-256 [26], proposed by SONY in 2007 and an ISO standard since 2012 as a suitable block cipher for lightweight cryptographic implementations, alongside PRESENT and LEA. The highest number of attacked rounds has been 15 rounds since 2010 [31]. Though new attacks have been published with the same number of rounds [19,33], this has not increased in the last 15 years. We propose for the first time an attack reaching 16 rounds. In addition, our differential MITM attack recovers the full master key, while previous attacks recovered only partial information from the subkeys, without showing whether this could lead to master-key recovery.

Our new applications can be seen in Table 1, including the previous best attacks against all the variants. In most cases, we gain 1 or 2 rounds with respect to previous attacks, while having equivalent complexity. Even in the cases where these attacks are too close to exhaustive search (and was also the case of the previous best attacks on less rounds), we provide the complexities of our new attacks for the previously known best number of rounds, considerably improving the complexity well below exhaustive search. This shows that our attacks are the best-known ones for all these ciphers.

In addition to our improved applications on those ciphers and the fully inclusive tool for searching for differential MITM attacks on SIMON-like ciphers, we present in Sec 3.3 a new way for reducing the data complexity.

Cipher	Rounds	Type of attack	Time	Data	Memory	Succ.rate	Source
Simon 32/64	22/32	Differential	$2^{58.76}$	2^{32}	-	0.315	[24]
	24/32	Diff MITM	$2^{63.8}$	2^{32}	$2^{66.57}$	-	[13]
	24/32	Mutli-Linear	$2^{63.57}$	$2^{31.57}$	-	-	[3]
	24/32	Diff MITM	$2^{60.85}$	2^{32}	2^{56}	0.63	[21]
	25/32	Diff MITM	2^{63}	2^{32}	2^{57}	0.63	Sec 3.2
	25/32	Diff MITM	$2^{62.92}$	2^{31}	2^{57}	0.63	Sec 3.3
Simon 48/96	25/36	Diff MITM	$2^{94.32}$	2^{48}	$2^{57.946}$	-	[12]
	26/36	Differential	$2^{67.89}$	2^{47}	-	0.42	[23]
	27/36	Linear	$2^{89.05}$	2^{47}	-	0.66	[23]
	28/36	Diff MITM	$2^{93.5}$	2^{48}	2^{77}	0.63	[21]
	29/36	Diff MITM	$2^{93.5}$	2^{48}	2^{84}	0.63	[21]
Simon 64/128	32/44	Diff MITM	$2^{125.93}$	2^{63}	$2^{90.493}$	-	[12]
	32/44	Differential	$2^{123.89}$	2^{63}	-	0.41	[23]
	33/44	Linear	$2^{120.09}$	2^{63}	-	0.68	[23]
	34/44	Diff MITM	$2^{121.5}$	2^{60}	2^{109}	0.63	[21]
	35/44	Diff MITM	$2^{123.5}$	2^{62}	2^{110}	0.63	[21]
	36/44	Diff MITM	$2^{125.7}$	2^{63}	2^{122}	0.63	[21]
Simon 128/256	50/72	Diff MITM	$2^{251.74}$	2^{128}	2^{78}	-	[12]
	52/72	Differential	$2^{250.76}$	2^{127}	-	0.6	[23]
	56/72	Linear	2^{249}	2^{126}	2^{144}	0.63	[18]
	56/72	Diff MITM	$2^{248.6}$	2^{128}	2^{254}	0.63	[21]
CLEFIA-256	15/26	Improbable diff	$2^{247.5}$	$2^{127.4}$	-	0.99	[31]
	15/26	Truncated Diff	2^{203}	2^{100}	-	-	[19]
	15/26	Integral	$2^{230.7}$	2^{128}	$2^{96.3}$	-	[33]
	15/26	Multi Zero-cor	$2^{237.9}$	$2^{124.5}$	-	-	[33]
	15/26	Diff MITM	2^{192}	2^{127}	2^{168}	0.71	[21]
	16/26	Diff MITM	$2^{236.64}$	2^{127}	2^{232}	0.71	Sec 5.2
Simeck 32/64	21/32	Diff MITM	2^{59}	2^{32}	2^{21}	-	[12]
	22/32	Differential	$2^{57.9}$	2^{32}	-	0.471	[24]
	23/32	Linear	$2^{61.78}$	$2^{31.91}$	-	-	[25]
	24/32	Diff MITM	2^{63}	2^{31}	2^{47}	0.63	[21]
Simeck 48/96	29/36	Diff MITM	$2^{93.28}$	2^{48}	2^{37}	-	[12]
	30/36	Differential	$2^{85.5}$	2^{48}	2^{76}	0.63	[18]
	32/36	Linear	$2^{93.28}$	2^{48}	-	0.63	[18]
	33/36	Diff MITM	$2^{95.3}$	2^{48}	2^{71}	0.63	[21]
Simeck 64/128	35/44	Diff MITM	$2^{125.49}$	2^{63}	2^{39}	-	[12]
	40/44	Differential	$2^{123.9}$	2^{63}	2^{123}	0.63	[18]
	41/44	Diff MITM	$2^{126.2}$	2^{63}	2^{92}	0.63	[21]
	42/44	Linear	$2^{123.9}$	$2^{63.5}$	-	0.63	[18]

Table 1: A summary of the best-known attacks on Simon $n/2n$, with $n \in \{32, 48, 64, 128\}$, CLEFIA-256 and Simeck $n/2n$, with $n \in \{32, 48, 64\}$. All attacks are in the single key setting. In this table we consider the memory-free access model.

Cipher	Rounds	Comput.	Mem. Access	Data	Mem.	S.rate	Efficient	Source
SIMON 32/64	24	$\mathbf{2^{61.8}}$	$2^{61.36}$	2^{32}	2^{55}	0.63	✓	[21]
	25	2^{63}	$\mathbf{2^{66.3}}$	2^{31}	2^{57}	0.63	✗	3.3
SIMON 48/96	28	$\mathbf{2^{93.5}}$	$2^{84.85}$	2^{48}	2^{77}	0.63	✓	[21]
	29	$2^{93.6}$	$\mathbf{2^{94}}$	2^{48}	2^{82}	0.63	∼	[21]
SIMON 64/128	34	$\mathbf{2^{121.5}}$	$2^{118.76}$	2^{60}	2^{109}	0.63	✓	[21]
	35	$\mathbf{2^{123.5}}$	$2^{117.7}$	2^{62}	2^{110}	0.63	✓	[21]
	36	$2^{126.5}$	$\mathbf{2^{127.9}}$	2^{63}	2^{122}	0.63	✗	[21]
SIMON 128/256	56	$2^{248.6}$	$\mathbf{2^{263}}$	2^{128}	2^{254}	0.63	✗	[21]
	56	2^{250}	$\mathbf{2^{251}}$	2^{128}	2^{253}	0.63	✓	[21]
CLEFIA-256	15	2^{192}	$\mathbf{2^{196.72}}$	2^{127}	2^{168}	0.71	✓	[21]
	16	$\mathbf{2^{236.64}}$	$2^{229.37}$	2^{127}	2^{232}	0.71	✓	5.2
SIMECK 32/64	24	$\mathbf{2^{63}}$	$2^{53.5}$	2^{31}	2^{47}	0.63	✓	[21]
SIMECK 48/96	33	$\mathbf{2^{95.27}}$	$2^{78.14}$	2^{48}	2^{71}	0.63	✓	[21]
SIMECK 64/128	41	$\mathbf{2^{126.21}}$	$2^{99.5}$	2^{63}	2^{92}	0.63	✓	[21]

Table 2: A summary of the differential MITM attacks described in this and the long version of this paper [21], considering a non-free memory access model.

2 Preliminaries

2.1 Differential Meet-in-the-Middle

In this part, we recall how differential meet-in-the-middle (MITM) attacks are mounted by first giving the classical approach presented in [8], then we present how we can improve this framework by using the *state-test*, *probabilistic key-recovery* and *structure* techniques as presented in [8] and [2]. We will denote by E an n-bit block cipher with a k-bit master key K.

The attacker starts by considering a differential distinguisher: an r_d-round differential property that propagates from a difference $\delta_{in} \in (\mathbb{F}_2^n)$ to a difference $\delta_{out} \in (\mathbb{F}_2^n)$ with a probability $2^{-p} > 2^{-n}$.

In the basic approach, the attacker splits the cipher into three parts: E_U (Upper part), E_D (Differential distinguisher) and E_L (Lower part) of r_u, r_d and r_l rounds, respectively, such that $E = E_L \circ E_D \circ E_U$.

We denote by K_{in} the set of bits needed to compute, given a certain plaintext P, another plaintext $\tilde{P}$, such that the difference of the two plaintexts after r_u rounds, at the beginning of the differential distinguisher, is δ_{in}. The size of the set K_{in} is $2^{k_{in}}$. Similarly, we denote by K_{out} the set of bits needed to compute, given a certain ciphertext C, another ciphertext $\tilde{C}$ such that the difference of the two ciphertexts after r_l rounds (at the end of the distinguisher), is δ_{out}. The size of this set is $2^{k_{out}}$. We will denote as $k_\cup$ the cardinality of the set $K_{in} \cup K_{out}$ and as $k_\cap$ the cardinality of the set $K_{in} \cap K_{out}$.

The framework of the attack, illustrated in Fig.1, can be described as follows:

The attacker starts by asking for the encryption C of a plaintext P. Then for every possible key guess k_i of K_{in}, the attacker can compute the plaintext $\tilde{P}_i$ such that, after r_u rounds, $E_U(P) \oplus E_U(\tilde{P}) = \delta_{in}$. The attacker can save in a hash table

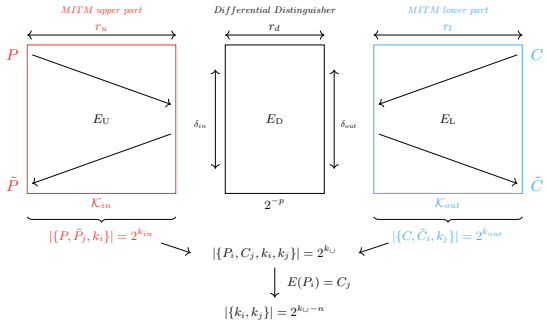


Fig. 1: High level description of the classic differential MITM technique.

$\mathcal{H}_{in}$ the $2^{k_{in}}$ "upper" triplets $\{P, \tilde{P}_i, k_i\}$. Similarly, the attacker can generate the $2^{k_{out}}$ "lower" triplets $\{C, \tilde{C}_j, k_j\}$ such that, for every possible key guess k_j of K_{out}, the attacker can generate a ciphertext $\tilde{C}_j$ verifying $E_L^{-1}(C) \oplus E_L^{-1}(\tilde{C}_j) = \delta_{out}$. As the lower triplets are generated, the attacker will match them on the fly with respect to the upper triplets saved in the hash table $\mathcal{H}_{in}$ by checking if $E(\tilde{P}_i) = \tilde{C}_j$ [1]. In the case of an equality, the attacker can save the key information k_i and k_j as a valid key candidate. Each key candidate is completed, if needed, then tested on extra data until the right key is recovered. There are $2^{k_\cup - n}$ key candidates to test, as we expect the equality between two states of size n to filter 2^{-n} elements. This full process needs to be repeated 2^p times to expect that one pair (P, C) verifies the differential distinguisher.

As one can start by guessing the common bits of K_{in} and K_{out}, the time complexity of this attack is:

$$\mathcal{T} = 2^{p+k_\cap}(2^{k_{in}-k_\cap} + 2^{k_{out}-k_\cap} + 2^{k_{in}+k_{out}-2\cdot k_\cap - n}). \tag{1}$$

In the previous framework, we stored the upper triplets arbitrarily, but one can also decide to stock the lower triplets. The memory complexity is then:

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

The naive data complexity is $\mathcal{D} = \min(2^n, 2^{p+\min(k_{in},k_{out})})$, as we have to check if $E(\tilde{P}_i) = \tilde{C}_j$ during the matching step, but in [8], the authors proposed to fix x bits of the plaintexts to force them to belong to some restricted portion of the codebook. This method could be used without modifying the time complexity. This reduces the data needed; however, the probability of the attack decreases by 2^{-x} as the values of $\tilde{P}$ need to verify also these conditions. Thus, the constraint on x is $p + x < n - x \Leftrightarrow x < \frac{n-p}{2}$ to expect at least one pair to follow the differential. The resulting data complexity is: $D = 2^{n-x}$.

A similar framework is proposed in [2] in the case of considering a *truncated differential* instead of a single one. In this case, we have an input set of differences to the distinguisher δ_{in} of size $2^{\Delta_{in}}$, an output set of differences δ_{out} of size $2^{\Delta_{out}}$

[1] In this work, similar to previous ones, we will consider being in a Random Access Machine model, where one can query the memory in unit time as explained in [22].

and a probability of going from δ_{in} to δ_{out} of 2^{-p}. The time complexity in this case is:

$$\mathcal{T} = 2^{p-\Delta_{in}+k_\cap}(2^{\Delta_{in}}2^{k_{in}-k_\cap} + 2^{\Delta_{out}}2^{k_{out}-k_\cap} + 2^{\Delta_{in}+\Delta_{out}+k_{in}+k_{out}-2k_\cap-n}). \quad (3)$$

The memory and data complexities are:

$$\mathcal{M} = 2^{min(\Delta_{in}+k_{in}-k_\cap,\ \Delta_{out}+k_{out}-k_\cap)}, \quad \mathcal{D} = \min(2^n, 2^{p-\Delta_{in}+\min(k_{in}+\Delta_{in},k_{out}+\Delta_{out})}). \quad (4)$$

State-test technique. The state-test technique introduced in [10],[15] in the context of impossible differential and MITM attacks, respectively, was then adapted in [2] for differential MITM attacks, and an in-depth study of it was performed in [20]. It reduces time complexity by guessing some state bits instead of all involved key bits, decreasing the total number of guesses needed to compute $\tilde{P}_i$ and $\tilde{C}_j$. We will denote, respectively, by S_{in} and S_{out} the sets of state-test guesses in the upper and lower parts, with $s_{in} = |S_{in}|$ and $s_{out} = |S_{out}|$. The resulting time complexity is:

$$\mathcal{T} = 2^{p+k_\cap}(2^{k_{in}-k_\cap+s_{in}} + 2^{k_{out}-k_\cap+s_{out}} + 2^{k_{in}+k_{out}+s_{in}+s_{out}-2k_\cap-n}). \quad (5)$$

The resulting memory complexity is:

$$\mathcal{M} = 2^{min(k_{in}+s_{in}-k_\cap,k_{out}+s_{out}-k_\cap)}. \quad (6)$$

The data complexity is not impacted by this method.

Each state-test also gives an equation over the unknown key bits. If the state-test is made in the upper part, the equation is linking at least two key elements that have not been guessed by the upper part and some bits of the plaintext. If the state-test is made in the lower part, the equation links at least two key elements that have not been guessed by the lower part with bits of the ciphertext. As those equations contain information about the key, it is interesting to exploit them to recover the full key or as a filter during the matching step. In [20], the authors explain how the state-test technique can be applied to improve different types of differential attacks.

Probabilistic key-recovery. With the similar goal of decreasing the number of key guesses needed to compute the upper and lower parts, the authors of [2] (and of [9] and [29] in another contexts) introduced the probabilistic key-recovery technique. The main idea is to consider the cancellation of some differences in both the lower and upper parts to stop the diffusion of the differences. We denote by $2^{-p_{in}}$ and $2^{-p_{out}}$ the probability of all the cancellation in the upper and lower parts of the path extension, respectively. By doing so, the overall probability of a pair to follow the differential path is now $2^{-p-p_{in}-p_{out}}$. Therefore, the number of repetitions of the attack needs to be increased by a factor of $2^{p_{in}+p_{out}}$. However, despite the fact that the common factor of the time complexity is increased by this amount, the three terms related to the upper, the lower and the matching parts of the time complexity are only increased by $2^{p_{out}}$, $2^{p_{in}}$ and nothing respectively. This is due to the fact that in the upper case we now keep only a factor of $2^{-p_{in}}$ of the previous kept triplets, and the same goes for the lower part with $2^{-p_{out}}$. The matching term compensates the increased common factor with these two reductions and stays the same.

Parallel partitioning. With parallel partitioning, the attacker tries to add some rounds to his attack by generating the inputs at the start of the upper or lower parts after some rounds. This technique is originally used and well known in Meet-in-the-Middle attacks. Automated tools for finding MITM attacks were proposed, incorporating parallel partitioning as in [14]. However, it could not be used in differential attacks before the differential MITM. The framework of parallel partitioning in differential MITM was first presented in [8] then improved in [2]. (An illustration of this framework is described in Fig. 2).

The attacker now splits the cipher into four parts, adding to the previous separation a new part E_S (Structure) before the upper part or after the lower part. Such structures cover r_s rounds. In the following description of the attack framework, we will consider that the attacker placed it before the upper part, and we denote by X the state at the end of the structure (and so at the beginning of the upper part). The new framework is as follows:

The attacker starts by fixing $F \leq n$ bits anywhere in the structure. The F fixed bits are chosen such that, for any guess of K_{in}, the attacker can compute F bits of X. Similarly, the F fixed bits are chosen such that, for any guess of K_{out}, the attacker can compute F bits of P. The attacker can consider all the possible values on the remaining bits of X and P and generate 2^{n-F} values for the full state X_f and for $C_f = E(P_f)$. The index f is used to denote that the messages X_f and P_f were generated from the specific set of values f of the set of fixed bits F. By considering each X_f, the attacker can compute $2^{n-F+k_{in}}$ triplets $\{X, \tilde{X}_i, k_i\}_f$ as in the previous framework. Similarly, the attacker can compute $2^{n-F+k_{out}}$ triplets $\{C, \tilde{C}_j, k_j\}_f$ and ask for the encryption of the $C_{f,i}$. As in the previous framework, we match the triplets in order to extract a list of key candidates that will be tested on extra data. For each considered pair $P_{f,j}$ and $X_{f,i}$, the attacker can verify if the messages are consistent through the F fixed bits, sieving 2^{-F} pairs of triplets. Additionally, sieving can be done by studying in more detail the non-fixed relations in the structure, but for now there is no generic method, and in previous work it was computed in an ad-hoc way. We will consider 2^{add} as the quantity of pairs of triplets that can be filtered from the non-fixed equations in the structure. We also underline that these equations can be used to retrieve information about the key but this also needs a specific study to be exploited.

For a specific f, we generate 2^{n-F} pairs (X_f, P_f). As we want to ensure that at least one pair will follow the differential path, we need to repeat the previous framework $2^{p-(n-F)}$ times. The resulting time complexity of this attack is:

$$\mathcal{T} = 2^{p-(n-F)}(2^{n-F}2^{k_{in}} + 2^{n-F}2^{k_{out}} + 2^{k_{in}+k_{out}-k_{\cap}-F-add}). \tag{7}$$

We note that adding a structure is a priori not changing the overall complexity of generating the upper and lower triplets, the real point of interest is the filtering part. To have an effective structure, we need to have the filter 2^{-F-add} be as close to 2^{-n} as possible. If it is, the structure adds some rounds to the attack without increasing its time complexity. The search for such an optimised structure can be complex by hand, and its generalisation and automatisation remained an open

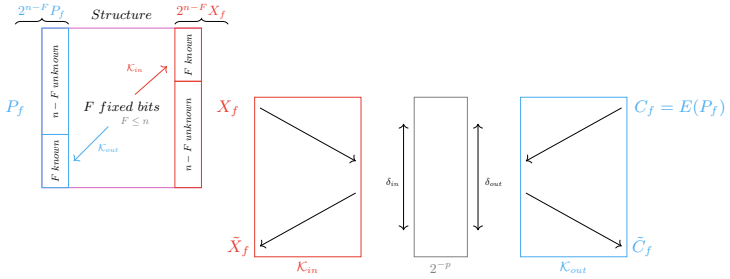


Fig. 2: High level description of the classic differential MITM technique with parallel partitions.

problem. The memory complexity of the attack is:

$$\mathcal{M} = 2^{min((n-F)+k_{in}-k_{\cap},(n-F)+k_{out}-k_{\cap})}. \tag{8}$$

In the case where $F = n$, the memory complexity is the same as in the classic framework, in other cases, it is higher.

The naive data complexity is the whole codebook. As before, fixing some word in the plaintext could decrease the area of the message. The structure makes this work more complex, as choosing the fixed bits inside the structure (including the plaintext) is the cornerstone of its description.

The resulting complexities of the differential MITM attack improved with previous techniques are as follows:

$$\begin{aligned}\mathcal{T}_E = 2^p \cdot (&2^{k_{in}+s_{in}+p_{out}} C_U + 2^{k_{out}+s_{out}+p_{in}+\Delta_{out}-\Delta_{in}} C_L \\ &+2^{k_{out}+k_{in}-k_{\cap}+s_{in}+s_{out}+\Delta_{out}+n-2F-add} C_M) C_E\end{aligned} \tag{9}$$

encryptions, where C_U, C_L and C_M are respectively the costs of the computations done in the upper, lower or middle parts of the attack, normalized by the cost of one encryption (C_E).

$$\mathcal{M} = 2^{min(n-F+k_{in}+s_{in}-k_{\cap},n-F+k_{out}+s_{out}-k_{\cap})}, \tag{10}$$

$$\mathcal{D} = 2^{n-a}, a < \frac{n-(p+p_{in}+p_{out})}{2}. \tag{11}$$

A lower bound of the time complexity of this attack, derived in [8], is:

$$\mathcal{T} \geq 2^{p+k-n}. \tag{12}$$

Memory access model. So far, for time complexity we have considered the memory-free-access model, where access to a hash table is negligible compared to one encryption. Since the tables used in our attacks can be large, memory access costs may not be negligible compared to encryption. We will also consider here the model where C_M and C_R correspond respectively to the cost of writing and the cost of reading an element in a hash table of size $\mathcal{M}$. The time complexity to store and access the elements in a hash table in this model is as follows:

$$\begin{aligned}\mathcal{T}_M = 2^p \cdot (&2^{min(k_{in}+s_{in}+p_{out},k_{out}+s_{out}+p_{in})} C_W \\ &+2^{max(k_{in}+s_{in}+p_{out},k_{out}+s_{out}+p_{in})} C_R).\end{aligned} \tag{13}$$

For the sake of simplicity, we will consider both C_W and C_R to be similar and call them C_A for cost of a memory access.

Thus in this model, the time complexity of the attack is $\mathcal{T}_E$ encryptions $+\mathcal{T}_M$ memory accesses, and we will present the costs of our attacks in this model this way. We have nevertheless considered the free-access memory model in Table 1 for fairness of comparison with previous results. In Table 2, we point out that most of our attack applications remain strictly valid attacks when considering non-free memory access model.

In our setting, both the upper and lower parts of our attacks are highly parallelizable, and therefore the throughput scales linearly with the size of the database ($\log \mathcal{M}$), and we will consider this cost for memory accesses.

In most of the applications of this paper, the attack is successful if one of the considered pairs of data (P, C) verifies the differential middle transitions when both input and output differences are added to it at the correct states. This is also the usual case for the differential MITM framework. In most cases, the attacker tests $P = 2^{p'} = 2^{p+p_{in}+p_{out}}$ pairs (P, C) of data for each candidate key, generating the same amount of differential pairs per key. The attack is successful if, one of these pairs verifies the middle differential for the secret key, which happens with probability $1-P_{n\text{-}d}$, with $P_{n\text{-}d}$ being the non-detection probability. From [7], we know that $1 - P_{n\text{-}d} = 1 - (1 - P^{-1})^P \approx 1 - e^{-1} = 0.63$. We can increase this probability when possible, by considering $a \cdot 2^{p'}$ pairs of data. The success probability is then $1 - P_{n\text{-}d} = 1 - (1 - P^{-1})^{aP} \approx 1 - e^{-a}$.

3 Differential MITM Attacks Against Simon

In this section, we present our new attacks on four instances of the Simon cipher. They are all in the single-key setting. By considering the differential MITM attacks along with all the techniques for optimising those attacks (state-test, probabilistic key-extension, modified data reduction technique and initial structure) we are able to increase by one, two or three rounds the best-known attacks on these ciphers, as can be seen in Table 1. The efficiency of the attacks relies, in part, in the 3-round initial structure which follows the description from 2.1 but are more complex than the ones previously used and cover up to three rounds, while the previous ones used in differential MITM attacks only covered two.

To find those attacks, we have implemented a tool, described in Sec. 4 that can be adapted for Simon-like ciphers, like Simeck as shown in the next section. We carefully verified all attacks by hand. In this section, we describe in detail the first known attack on 25 rounds on Simon 32/64. For the others, we provide the parameters and the figure in the long version of this paper [21], but the procedure remains the same.

3.1 Brief description of the Simon family of block ciphers

Simon n/k, a family of Feistel block ciphers presented in [5], operates on blocks of size $n \in \{32, 48, 64, 96, 128\}$ and with a key of size $k \in \{n, \frac{3n}{2}, 2n\}$ divided

into m keywords. The number of rounds r depends on n and k. There are 10 instances of SIMON with parameters given in [21].

The state is divided into two blocks L_i, R_i of size $\frac{n}{2}$ corresponding to the left and the right part of the input of round i of a Feistel network. The round function is: $L_{i+1} = R_i \oplus K_i \oplus f(L_i)$ and, $R_{i+1} = L_i$, where $f(x) = ((x \lll 8) \wedge (x \lll 1)) \oplus (x \lll 2)$, K_i is the subkey of round i. The subkey K_i is given by the following relation:

$$K_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus K_i \oplus (K_{i+1} \ggg 3) \oplus (K_{i+1} \ggg 4) \text{ if } m = 2 \\ c \oplus (z_j)_i \oplus K_i \oplus (K_{i+2} \ggg 3) \oplus (K_{i+2} \ggg 4) \text{ if } m = 3 \\ c \oplus (z_j)_i \oplus K_i \oplus (K_{i+3} \ggg 3) \oplus (K_{i+3} \ggg 4) \oplus K_{i+1} \oplus (K_{i+1} \ggg 1) \text{ if } m = 4, \end{cases}$$

for $0 \leq i \leq r - m$, $c = 2^{\frac{n}{2}} - 4$ and $(z_j)_i$ is the i-th bit of a constant sequence.

In the attacks, we will use the equivalent subkey technique, introduced in [16] and described in [21] for our case.

3.2 Differential MITM attack on 25-round SIMON 32/64

Differential distinguisher. We propose a new single-key differential Meet-in-the-Middle attack on SIMON 32/64 covering 25 rounds, which is up-to-date the best-known attack against this cipher, increasing for the first time the number of rounds since 2015. We also propose an improved attack on the 24-round version, whose details are given in [21].

Our attack exploits the 14-round differential distinguisher published in [17] :

$$(0000, 0008) \xrightarrow[14r]{2^{-30.81}} (0800, 0000).$$

The validity of this distinguisher for all the keys represents an open problem for SIMON. In [6] the authors study the validity of a characteristic for six rounds and show that the characteristic is valid for only a quarter of the keys and the probability of the distinguisher is 2^2 higher than expected. As in previous cryptanalysis on SIMON, we assume this distinguisher is valid for the full set of keys (as for the other distinguishers we use for attacks on SIMON).

Outline of the attack. The detailed 25-round attack is given in Fig. 3. It can be separated in four parts: the first 3 rounds, that will be covered by an initial structure, 4 key-recovery rounds in the upper part where the guesses and known state bits are represented in red, 14 rounds of differential distinguisher and 4 key-recovery rounds in the lower part with guesses and known state bits represented in blue. In the upper key-recovery part of the attack, we consider a probabilistic key-extension with probability of 2^{-1} marked with a P in Fig. 3. The parameters of the attack are as follows:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
25	3	4	14	4	30.81	25	24	3	3	1	0	2

In the following subsections, we will detail the bits that need to be guessed, but we provide in Table 3 a summary of the guessed key relations, and of the determined key bits once a match for the structure is found.

Initial structure on 3 rounds. We propose here the first 3-round initial structure to be used in a differential MITM attack, which was made possible because of the Feistel configuration of SIMON.
We will build an initial structure of size 2^{30}, and for this, we need for each structure to fix two bits of the second round: $L_1[0,6]$. For the lower part, we guess the key bits $K_0[0,6]$ and for the upper part, we guess $K_2[0,6]$. From the plaintext side, we will consider a structure of size 2^{32} that takes all the possible values in all the bits but in $R_0[0]$ and in $R_0[6]$. These two bits will be determined by the key bits guesses and the fixed values, as the following relations hold:

$$L_1[0] \oplus K_0[0] = L_0[2] \oplus (L_0[1] \cdot L_0[8]) \oplus R_0[0] \tag{14}$$

$$L_1[6] \oplus K_0[6] = L_0[8] \oplus (L_0[7] \cdot L_0[14]) \oplus R_0[6] \tag{15}$$

The attacker generates all the possible values of $L_0|R_0$ verifying these two equations, leading to 2^{30} possible plaintexts. Similarly, as $K_2[0,6]$ is known, the attacker can generate 2^{30} $L_3|R_3$ states from the two fixed bits.

Filtering through the structure. During the matching step of the attack, each of the 2^{30} pairs (L_0, R_0) will be associated to its generated pairs $(\tilde{L_0}, \tilde{R_0})$, and each of the 2^{30} (L_3, R_3) will be associated to its $(\tilde{L_3}, \tilde{R_3})$ pairs. The attacker can verify, with the guessed key bits, if the values of $\tilde{L_1}[0,6]$ are a match, leading to a 2^{-2} bit filter. Then the attacker, by guessing some additional key bits, can also verify if the following relations between the upper and the lower triplets are consistent in some bits (with $\Delta L_i = L_i \oplus \tilde{L_i}$ and $\Delta R_i = R_i \oplus \tilde{R_i}$):

$$\Delta L_1 = \Delta R_0 \oplus f(L_0) \oplus f(\tilde{L_0}) = \Delta L_3 \oplus f(R_3) \oplus f(\tilde{R_3}) \tag{16}$$

$$\Delta L_0 = \Delta R_3 \oplus f(L_3 \oplus K_2 \oplus f(R_3)) \oplus f(\tilde{L_3} \oplus K_2 \oplus f(\tilde{R_3})) \tag{17}$$

$$\Delta R_3 = \Delta L_0 \oplus f(R_0 \oplus K_0 \oplus f(L_0)) \oplus f(\tilde{R_0} \oplus K_0 \oplus f(\tilde{L_0})) \tag{18}$$

$$with\ f(X)[i] = X[i-2] \oplus (X[i-8] \cdot X[i-1]) \tag{19}$$

Eq. 16 is independent of the key, therefore the attacker can filter out the upper and lower pairs on the differences of the 14 non-fixed bits of L_1 (as for the two remaining bits, we have already filtered regarding the 2 values of $\tilde{L_1}$ corresponding to the initially fixed bits in L_1). This adds a filter of 2^{-14} to the matching step.

Regarding the structure, there are potentially 16 additional filtering bits. For this, the attacker will consider the known differences in L_0 and R_3. The 16-bit equations Eq. 17 and Eq. 18, show that to filter a 1-bit difference we need to guess two key bits (the ones appearing in the non-linear operation, as the others will cancel out), and some key bits can be used for two equations. To optimise our attack, we have chosen to guess 7 bits from K_0: $K_0[0,2,4,6,9,11,13]$ in the lower part, and 11 bits from K_2: $K_2[0,1,3,5,6,7,8,10,12,14,15]$ in the upper part. Please note that these two sets already include the 2 bits that need to be guessed from each side to build the 2^{30} structures, so it is the total number of

bits added for each side to K_{in} and K_{out}. It is easy to verify that, by guessing these bits, we can filter 6 bits regarding equation Eq. 18 (the ones at positions $\Delta R_3[1,3,5,8,10,12]$), and 10 bits regarding equation Eq. 17 (the ones at positions $\Delta L_0[0,2,4,6,7,9,11,13,14,15]$). This adds a filtering of $10+6=16$ bits, and the overall filtering of the states will be of $2^{-2-14-16}=2^{-32}$. As we will see later, when recovering a candidate match, the remaining $16\cdot 3-18=30$ bits of the key in the three initial rounds will also become fully determined. During each match, there will be an additional filtering regarding the degrees of freedom in the key, that we will detail later.

State-tests filtering. In order to reduce the amount of key bits guessed in the attack, 3 state-tests are used in the upper part and 3 in the lower part. In previous uses of the state-test technique in differential MITM attacks, the considered ciphers were SPN. The non-linear equations derived involved Sboxes and were more complex than our case. In SIMON, as the non-linear function used is the bit-wise AND, in some specific setups the associated state-test equation can become linear, if some involved variables are known.

We consider the example of $S = K_A \oplus (K_B \cdot\ C)$ where K_A and K_B denote key elements, C a known state and S the value of the state needed to perform the attack. As C is known, instead of guessing 2 key bits, the attacker can guess the state S with the state-test technique, and the time complexity decreases. When performing the match, the attacker can use the known value of C, indeed if $C=0$ then we can deduce that $S=K_A$, otherwise $S=K_A\oplus K_B$. In both cases, the attacker can recover one direct bit of information on the key.

The six state-test equations used in this attack are as follows (unknown elements are represented in bold, while the others can be computed with the guessed elements from each side):

$$
\begin{aligned}
S_1^{up} =& \mathbf{L_5}[3] \oplus \mathbf{K'_5}[3] = \mathbf{K'_5}[3] \oplus R_4[3] \oplus L_4[5] \oplus ((L_4[11] \oplus K'_4[11]) \cdot (L_4[4] \oplus \mathbf{K'_4}[4]))\\
S_2^{up} =& \mathbf{L_6}[3] \oplus \mathbf{K'_6}[3] = \mathbf{K'_6}[3] \oplus R_5[3] \oplus L_5[5] \oplus ((L_5[11] \oplus K'_5[11]) \cdot (L_5[4] \oplus \mathbf{K'_5}[4]))\\
S_3^{up} =& \mathbf{L_6}[5] \oplus \mathbf{K'_6}[5] = \mathbf{K'_6}[5] \oplus R_5[5] \oplus R_4[7] \oplus L_4[9] \oplus ((L_4[8] \oplus K'_4[8]) \cdot (L_4[15] \oplus K'_4[15]))\\
& \oplus ((R_4[6] \oplus L_4[8] \oplus ((L_4[7] \oplus \mathbf{K'_4}[7]) \cdot (L_4[14] \oplus K'_4[14])) \oplus \mathbf{K'_5}[6]) \cdot (L_5[13] \oplus K'_5[13])\\
S_1^{down} =& \mathbf{L_{22}}[11] \oplus \mathbf{K'_{22}}[11]\\
=& \mathbf{K'_{22}}[11] \oplus R_{24}[13] \oplus L_{24}[11] \oplus ((L_{23}[12] \oplus K'_{23}[12]) \cdot (L_{23}[3] \oplus \mathbf{K'_{23}}[3]))\\
S_2^{down} =& \mathbf{L_{21}}[11] \oplus \mathbf{K'_{21}}[11]\\
=& \mathbf{K'_{21}}[11] \oplus R_{24}[15] \oplus R_{24}[11] \oplus L_{24}[13] \oplus (L_{22}[12] \oplus K'_{22}[12]) \cdot (L_{22}[3] \oplus K'_{22}[3])\\
& \oplus ((L_{23}[5] \oplus K'_{23}[5]) \cdot (L_{23}[14] \oplus \mathbf{K'_{23}}[14]))\\
S_3^{down} =& \mathbf{L_{21}}[13] \oplus \mathbf{K'_{21}}[13]\\
=& \mathbf{K'_{21}}[13] \oplus R_{24}[13] \oplus R_{24}[1] \oplus L_{24}[15] \oplus ((L_{23}[7] \oplus K'_{23}[7]) \cdot (L_{23}[0] \oplus \mathbf{K'_{23}}[0]))\\
& \oplus (L_{24}[14] \oplus R_{24}[0] \oplus (L_{23}[15] \oplus \mathbf{K'_{23}}[15]) \cdot (L_{23}[6] \oplus K'_{23}[6])\\
& \oplus \mathbf{K'_{22}}[14]) \cdot (L_{22}[5] \oplus K'_{22}[5])
\end{aligned}
$$

Those equations can be used to recover $3+3=6$ additional bits of direct information on the key.

Filtering Relations from the Key-Schedule. To efficiently filter a key-information bit k^{f}, the attacker can rewrite it as $k^{\mathrm{f}} = l(k^{\mathrm{up}}) \oplus l(k^{\mathrm{down}}) \oplus l(k^{\mathrm{uk}})$, where l is a linear function and k^{uk} is some key-information bit known to neither

the upper nor the lower part. We assume that k^{f} is known to the upper part through the state test equations. During the match, the upper part can compute $l(k^{\mathrm{up}}) \oplus k^{\mathrm{f}}$, and the lower part can compute $l(k^{\mathrm{down}})$. If $l(k^{\mathrm{uk}}) = 0$, the attacker can filter this equation directly. If $l(k^{\mathrm{uk}}) \neq 0$, the equation cannot be filtered alone. Nevertheless, by collecting at least k' equations of this form (with k' the dimension of still unknown master-key space), the attacker can write and solve a linear system that yields the full master key and, by extension, $l(k^{\mathrm{uk}})$ (assuming the equations form a system of rank k'). Any additional key equation can then be filtered directly.

In our attack, we consider $K_{\mathrm{in}} + K_{\mathrm{out}} + 14 + 16 = 79$ equations, given by Table 3, forming a rank-64 system. Thus, during the matching step, the attacker has an additional $2^{-79+64} = 2^{-15}$ filtering factor.

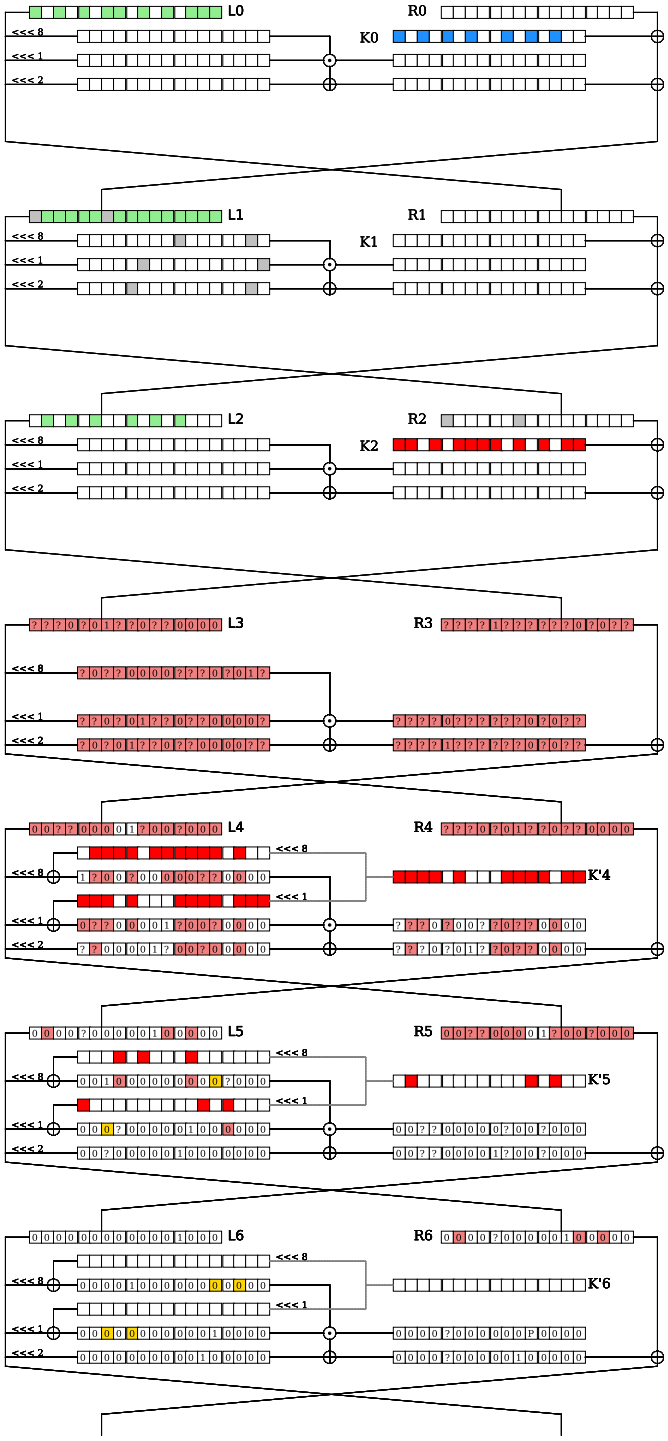
This factor can be improved by incorporating the $s_{\mathrm{in}} + s_{\mathrm{out}} = 6$ key bits extracted from the state-test equations. However, these 6 bits are combinations of key elements that vary with the messages chosen at the boundary of each part. In this case, k^{f} has two possible expressions (k_1^{f} and k_2^{f}), selected directly by the message values. For example, the key information contained in S_1^{up} can be either $K_5'[3] \oplus K_4'[4]$ or $K_5'[3]$. The upper part knows which expression of k^{f} is computed and can readily evaluate $l(k^{\mathrm{up}})$. The lower part, however, does not know this expression, since it depends only on values held by the upper part; it must therefore compute the value of both expressions ($l(k_1^{\mathrm{down}})$ and $l(k_2^{\mathrm{down}})$), increasing its size by 2^1. Since a single key bit is verified, this yields no net filtering (the 2^{-1} filter is cancelled by the 2^1 increase). Moreover, such a bit cannot be used to build and solve the rank-k linear system, for the same reason.

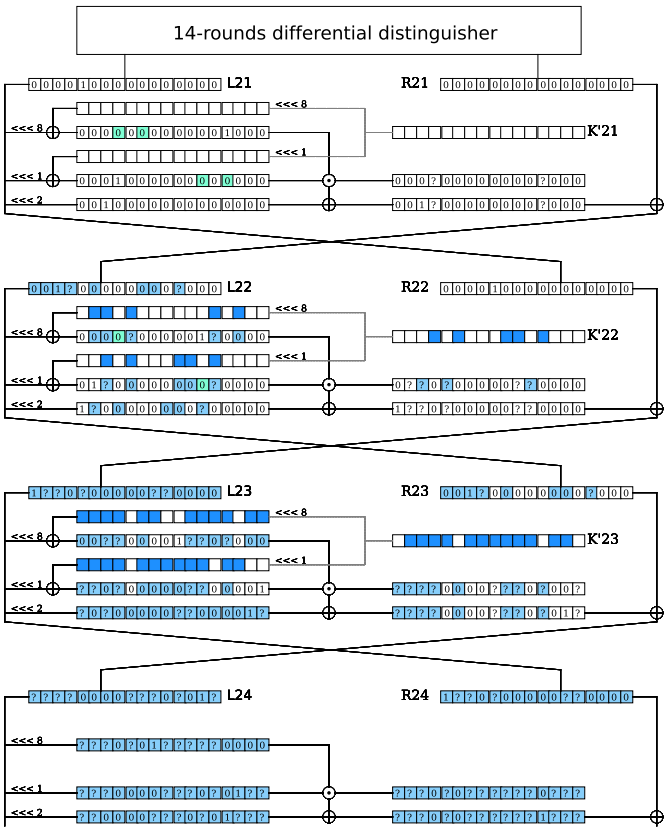
If, the attacker can build a rank-k linear system, there is no need to consider both expressions of k^{f} in the lower part: all key elements are known, so the lower part reads the required key bits directly from the linear system, with no additional guesses, giving a 2^{-1} filter.

Here the attacker considers 80 linear key relations; the 6 state-test key bits can then be filtered efficiently, for a total filtering factor of $2^{-(79-64)-6} = 2^{-21}$.

Attack procedure. We now describe the steps of the attack:

1. Fix the set $\mathcal{F}$ composed of the bits $L_1[0, 6]$,
2. *Upper part*: For i from 0 to 2^{29} (guesses of (K_{in}, S_{in})),
   - Generate the 2^{30} possible $L_3|R_3$, as previously described, verifying the values of $L_1[0, 6]$.
     - For each $L_3|R_3$ and each value i for (K_{in}, S_{in}), generate the triplet $\{L_3|R_3, \tilde{L}_3|\tilde{R}_3, i\}$, such that after 4 rounds, $L_7|R_7$ and $\tilde{L}_7|\tilde{R}_7$ verify the input difference of the distinguisher. Due to the probabilistic extension that we force, we only keep half of the $\tilde{L}_7|\tilde{R}_7$. Store the triplets in a hash table $\mathcal{H}$ of size $2^{30+28-1}$.
3. *Lower part*: For j from 0 to 2^{27} (guesses of (K_{out}, S_{out})),
   - Generate the 2^{30} possible $(L_0|R_0)$ as previously described, verifying the values of $L_1[0, 6]$.


L0
R0
K0
<<< 8
<<< 1
<<< 2
L1
R1
K1
L2
R2
K2
L3
R3
L4
R4
K'4
L5
R5
K'5
L6
R6
K'6



■ : This key bit is guessed by the upper part of the attack
■ : This state bit can be computed by the upper part of the attack
■ : This state bit is guessed by the upper part of the attack,
it is used to sieve the candidates during the match

■ : This key bit is guessed by the lower part of the attack
■ : This state bit can be computed by the lower part of the attack
■ : This state bit is guessed by the lower part of the attack,
it is used to sieve the candidates during the match

0 : The difference on this bit is 0
1 : The difference on this bit is 1
? : The difference on this bit can be 0 or 1
P : The difference on this bit is considered 0 by probabilistic propagation
■ : The difference on this bit can be computed by the upper
and lower part of the attack
■ : The value of this bit is fixed

Fig. 3: Path of the 25 rounds attack against SIMON 32/64.

Part of the attack	Key	total
K_{in}	$K_2[0,1,3,5,6,7,8,10,12,14,15]$, $K'_4[0,1,2,3,5,9,10,11,12,14,15]$, $K'_5[1,11,13]$	25
K_{out}	$K_0[0,2,4,6,9,11,13]$, $K'_{22}[3,5,9,10,12]$, $K'_{23}[1,2,3,4,6,7,8,9,10,11,13,14]$	24
S_{in}	$K'_5[3] \oplus K'_4[4]$ or $K'_5[3]$ $K'_6[3] \oplus K'_5[4]$ or $K'_6[3]$ Combination of $K'_6[5], K'_5[6], K'_4[7]$	3
S_{out}	$K'_{22}[3] \oplus K'_{23}[4]$ or $K'_{22}[3]$ $K'_{21}[11] \oplus K'_{23}[14]$ or $K'_{21}[11]$ Combination of $K'_{21}[13], K'_{22}[14], K'_{23}[0], K'_{23}[15]$	3
determined matching in L_1 from $(\tilde{L}_0, \tilde{R}_0)$ and $(\tilde{L}_3, \tilde{R}_3)$	$K_0[1,3,5,7,8,10,12,14,15]$ $K_2[2,4,9,11,13]$	14
determined matching in R_1 from $(\tilde{L}_0, \tilde{R}_0)$ and $(\tilde{L}_3, \tilde{R}_3)$	K_1[0-16]	16

Table 3: Total number of key bits guessed and determined in the attack.

- For each $L_{25}|R_{25} = E_{\text{SIMON}_{25}}(L_0|R_0)$ and each value j for (K_{out}, S_{out}), generate the triplet $\{L_{25}|R_{25}, \tilde{L}_{25}|\tilde{R}_{25}, j\}$, such that 4 rounds before, $L_{21}|R_{21}$ and $\tilde{L}_{21}|\tilde{R}_{21}$ verify the output difference of the distinguisher. Store the triplets $\{L_0|R_0 = E^{-1}_{\text{SIMON}_{25}}(L_{25}|R_{25})$, $\tilde{L}_0|\tilde{R}_0 = E^{-1}_{\text{SIMON}_{25}}(\tilde{L}_{25}|\tilde{R}_{25}), j\}$ in a hash table $\mathcal{L}$ of size 2^{30+27}.

4. *Match*: For each triplet in $\mathcal{H}$ (though this could be done on the fly without the need for storing this list) verify with which elements in $\mathcal{L}$ there is a collision on the $2 + 30 = 32$ filtering bits from the structure and on the 22-bit redundant relations from the key-schedule. Computing these 54 values from the upper and lower parts, and using them to index the hash tables, this check can be efficiently done, and the bottleneck of the matching step will be the number of solutions.
5. Test the remaining key candidates (i, j) on extra data.
6. Repeat the whole procedure for $2^2 > 2^{1.81}$ different values of the set $\mathcal{F}$ composed of the bits $L_1[0, 6]$ (from step 1.), to ensure that at least one pair verifies the differential distinguisher and the probabilistic key-recovery path.

The time complexity of this attack in the free-access memory model is:

$$\mathcal{T}_E = 2^2(C_U 2^{30} 2^{25+3} 2^{-1} + C_L 2^{30} 2^{24+3} + C_M 2^{57+57-32-21}) \approx 2^{63} \text{ encryptions.} \quad (20)$$

If we consider memory accesses cost:

$$\mathcal{T}_M = 2^2(2^{57} + 2^{57})C_A. \quad (21)$$

Here C_U, C_L and C_M correspond respectively to the normalised cost of a partial encryption in the upper part, the cost of a partial encryption in the lower part and the cost of a full encryption of this reduced form of SIMON and C_A corresponds to the normalised cost of a memory access in a hash table of

size $\mathcal{M} = 2^{30+24+3} = 2^{57}$. Then, the cost is $\mathcal{T}_M = 2^{60.5}2^{5.8} = 2^{66.3}$. As one SIMON encryption takes less than 1 nanosecond, this term would be higher than exhaustive search, and the attack would not be valid in the model with such memory accesses cost.

As the probability of the distinguisher including the probabilistic key extension is $2^{-31.82}$, we cannot apply directly the trick presented in [8] for reducing the data complexity, and the data complexity of the attack, as above described, requires the whole codebook. We will now explain how this data complexity can be reduced by using multiple differentials, as was done for instance with impossible differential attacks in [9].

3.3 Reducing the data complexity using multiple differentials

In this section, we show how the previous attack can be adapted in order to reduce its data complexity from the whole codebook to 2^{31}. This idea can be considered in general in differential MITM attacks as a generic way of reducing the data complexity, as we explain at the end of this section. Please note that here, for the sake of simplicity, we consider one fixed structure and multiple distinguishers that are rotated versions of the original to artificially increase the data available. We could also consider rotated versions of the structure if more pairs were needed in a similar way.

In the previous attack, we needed to repeat the overall procedure $2^2 > 2^{1.81}$ times for each structure of size 2^{30} for being able to expect a pair that would verify the distinguisher and the probabilistic extension. Therefore, we cannot force x plaintext bits to 0 as $p = 31.81$, $n = 32$, and as we need that $p + x < n - x \Leftrightarrow x < \frac{n-p}{2}$, x could only be equal to zero.

In order to counter this problem, we will only build two structures of size 2^{30}, where one of the two fixed bits, instead of being in L_1, will be in R_0, so that we can restrict the data needed to 2^{31}. In order to be able to do this, the upper part will have to guess in addition one bit already guessed by the lower part: $K_0[0]$ for instance. This implies the lower term of the complexity inside the parenthesis to be reduced by a factor 2, as this bit is now common to both parts, and an additional external factor of 2^1 corresponding to this commonly guessed bit.

For the sake of simplicity, we can fix this bit to the value 0, and compute and store all the possible associated ciphertexts in a hash table $\mathcal{H}$. This is the only data that we will use in our attack.

When performing the lower part of the attack now, each time we compute a value for $\tilde{L}_{25}|\tilde{R}_{25}$, we check if it appears in $\mathcal{H}$. If it is not, which happens with probability 2^{-1}, we can discard the triplet as it will not verify the data condition.

The probability now of having a triplet verifying the distinguisher, the probabilistic extension and the one fixed plaintext bit for $\tilde{P}$ is of $2^{-30.82-1-1} = 2^{-32.82}$.

For generating enough triplets, we will consider similar rotated variants of the distinguisher: it is easy to see that each one of the 16 rotated versions of the original differential distinguisher will have a similar probability, and will just involve different keys in it (a rotated version of the original positions).

We will therefore consider that we generate one structure of 2^{31}, and from it we will check the triplets that we will keep when performing the attack for each of 7 out of the 16 possible rotated versions of the distinguisher (7 is enough to ensure sufficient sampling). The amount of data for each is smaller than the $2^{32.8}$ originally expected, but as we check the 2^{31} initial structure with respect to $7 \approx 2^{2.8}$ different distinguishers, we can expect one to verify the differential path, the probabilistic transition and the fixed bit in the plaintext.

The complexity of this attack, where we only consider the free-acces memory model would be:

$$\begin{aligned}\mathcal{T}_E &= 7 \times 2^1(2^{30}2^{25+3}2^{-1} + 2^{30}2^{24+3}2^{-2} + 2^{57+55-32-21}) \\ &= 2^{61.8}C_U + 2^{58.8}C_L + 2^{62.8}C_M = 2^{62.92}.\end{aligned} \tag{22}$$

Here C_U, C_L and C_M are, respectively, the cost of building one upper triplet, one lower triplet, and of obtaining one match. We do not want to go into further detail of these costs, but it is easy to see that the overall cost of the attack is smaller than 2^{63} 25-round encryptions, while the exhaustive search cost is 2^{64} 25-round encryptions.

Generalising the idea.

As for impossible differential attacks, we consider 2^m different equivalent distinguishers (up to a rotation) to reduce the data needed. We consider, without loss of generality, that the structure is added in the first rounds.

If we aim at having a data complexity of 2^{n-x}, we need to verify that $x \leq F$. The next condition is that $2^{p+p_{in}+p_{out}-s+x} \leq 2^m$ so that we have enough sampling for detecting one pair following one of the differentials (potentially with probabilistic extension) and also verifying the x fixed values on the data.

Next, we still need to verify that the knowledge of the key allows to fix x bits from the plaintext (or ciphertext) in the structure, so that we only pay this probability once. For this, we might have a rearranging of guessed bits to do, but potentially, we do not need to respect the previous limitations for reducing the data complexity. In the case of most structures, and definitely the ones used in all the SIMON attacks, if the bits guessed for the lower part that are involved in the structure (the key bits from K_0) are guessed in common also for the upper part of the attack, the fixed bits can be placed in the plaintext instead of a middle state.

We will choose $2^m = 2^{p+p_{in}+p_{out}-s+\mathbf{x}}$ equivalent distinguishers as this will be enough. With all of this, the final time complexity will be given by:

$$\begin{aligned}\mathcal{T}_E &= 2^m 2^{\mathbf{x}}(2^s 2^{k_{in}+s_{in}-p_{in}} + 2^s 2^{k_{out}+s_{out}-p_{out}-\mathbf{2x}} \\ &\qquad + 2^{2s}2^{k_{in}+s_{in}-p_{in}+k_{out}+s_{out}-p_{out}-\mathbf{2x}-n-k_\cap}) \\ &= 2^p(2^{p_{out}+\mathbf{2x}+k_{in}+s_{in}} + 2^{p_{in}+k_{out}+s_{out}} + 2^{k_{in}+s_{in}+k_{out}+s_{out}-F-k_\cap}).\end{aligned} \tag{23}$$

It is easy to see that the first term of the time complexity, associated to the upper key-recovery part, is the only one affected, with an additional factor of 2^{2x}, while the two other terms, the one associated to the lower part and the one

on the match, remain the same as the one without applying this data reduction technique. While this increased term is not the bottleneck and remains below the two others, the time complexity will not change (or will change and be determined by this term).

We can see that, if in the previous attack we considered $x = 2$, the overall time complexity would become $\mathcal{T}_E \approx 2^{64}$. We could claim that this is slightly smaller than exhaustive search when taking into account the cost of each operation when compared to a full encryption, but this was very close, and we preferred to detail the attack for $x = 1$. The data complexity is now $\mathcal{D} = 2^{n-x}$.

4 Tool for SIMON-like ciphers

4.1 State of the art

The first automated tool for finding full differential MITM attacks was proposed in [2]. This tool was dedicated to the SPN ciphers SKINNY and CRAFT and was used to find the current best attacks on SKINNY 64-192 and SKINNY 128-384. This tool searched for truncated differential distinguishers, the upper and the lower parts of the attack at the same time, including the state-test and the probabilistic key-recovery techniques in the search. The tool did not take into account initial structures with parallel partitioning, and the authors had to add this by hand. Other tools have been proposed, like the one in [12], that, like ours, is dedicated to SIMON-like ciphers. This tool searches for the differential trail, the upper and lower key-recovery parts of a differential MITM attack, using the selective key guessing technique, but does not consider state-tests, probabilistic key-recovery or parallel partitioning techniques.

In this section, we will present our tool, which is also dedicated to SIMON-like ciphers, built from scratch using classic MILP modelling techniques in symmetric cryptanalysis. By taking as input a differential distinguisher and the parameter size of each part of the attack, the model outputs the trail of the structure, the upper and the lower parts of the attack with all the previously described techniques (except the reduction of data complexity with multiple differentials). The graphical output helps to easily verify the proposed attacks.

4.2 Our MILP model

Our tool takes as input an existing differential distinguisher and finds an efficient way of extending it into a differential MITM attack. We have created a MILP model that does this, taking into account all the existent optimisation techniques, such as state-test, probabilistic key-extension, and, for the first time in differential MITM attacks, parallel partitioning techniques. Though the final aim is to build a tool that will look for the distinguisher and the optimal attack at the same time, this is for now out of reach, and our tool has already allowed us to find much-improved attacks with respect to the best-known ones (our full MILP model is available in the link given at the introduction).

A new feature proposed by our tool is that it provides a graphical output describing the attack that significantly simplifies verification and understanding of the attack. For instance, all the figures included in this paper, used to describe the SIMON and SIMECK attacks, have been generated automatically by the tool. All the figures in the paper can also be found in the repository. We will now describe our model, the input parameters, the propagation rules we used, how the parallel partitioning was taken into account, and the objective function.

Input parameters The inputs of the model are:

1. The size n of the block and the size k of the key.
2. The shift of each branch (as an example for SIMON it's $8, 1, 2$).
3. The size of each part of the attack: r_S, r_U, r_D, r_L.
4. The probability p of the differential distinguisher.
5. The input and output of the distinguisher: δ_{out} and δ_{in}.
6. A boolean variable describing if the key-schedule is linear.

Model Variables Each considered bit of the cipher is represented through three variables in the MILP model:

1. The *state variable* that takes values in 0, 1, 2 where 0 represent an unknown state, 1 a known state and 2 a state-tested state.
2. The *difference variable* that takes values in 0, 1, 2, 3 where 0 represent a known difference of value 0 on this bit, 1 a known difference of value 1 on this bit, 2 an unknown difference on this bit and 3 for a difference that should be unknown but that is cancel with some probability.
3. The *linear property variable* that takes values in 0, 1, 2, i.e the linear property of the equation to compute this bit the beginning of either the lower or upper part. 0 represent a linear equation, 1 a pseudo-linear equation and 2 a non-linear equation.

The next sections describe the modelling of each part of the attack, starting with the modelling of the propagation of differences, including the probabilistic key-recovery technique. Next, the modelling of the state propagation with the state-test technique and the filtering of the almost linear state-test equations. Then, we describe the modelling of the structure construction, followed by the description of the MILP objective function, which makes the link between all the parts of the attack in order to output the best possible external rounds for the given distinguisher.

Differences propagation rules Considering one bit of one word, the difference on this bit can take four different values as described in Fig. 4:

The possible transitions through the XOR operator are described in Fig. 5. The possible transitions through the AND operator are described in Fig. 6, but require the knowledge of some state values. Therefore, we first introduce the representation of state values.

Value propagation rules. Considering one bit of one word, the value of this bit can take three different values as described in Fig. 7:
The propagation of a state value through the XOR and AND operations is trivial. In both cases, $A \oplus B = C$ and $A \cdot B = C$. In order to know the value of C, either

$\boxed{0}$: The difference on this bit is null, correspond to a *difference variable* value of 0

$\boxed{1}$: The difference on this bit is active, correspond to a *difference variable* value of 1

$\boxed{?}$: The difference on this bit can be null or active, correspond to a *difference variable* value of 2

$\overset{p}{\boxed{0}}$: The difference on this bit is null with probability p, correspond to a *difference variable* value of 3

Fig. 4: Four difference possibilities in the MILP Model.

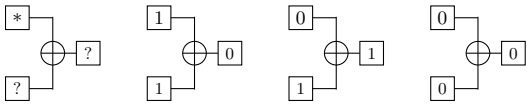


Fig. 5: Rules for the propagation of differences through the XOR operator ($*$ means any possible differential).

we guess its value or the values of both A and B are needed. We can model them as in Fig. 8:
The necessity of knowing the value of a bit in the key-recovery is directly given by the differential propagation through the non-linear layer, here the AND operation. In our case, to control the propagation of the difference through the AND, the minimum needed values for the possible difference propagation can be designed as described in Fig. 6.

The model considers that the start of the upper and lower parts are known bits. state-tests can be done only on the bits where the difference is known (0 or 1). Key addition follows the same rules as the XOR of two states, the key bits are free variables, i.e., the model can activate them without constraints. For the probabilistic cancellation, the model is limited: the total probability of the attack cannot exceed the size of the block.

Almost linear state-test equations. As explained in 3.2, some state-test equations may be almost linear and used to filter the candidates during the matching step. To ensure that the model makes the difference between the almost linear and the non-linear state-test equations, we add a variable to each bit of the model characterising that (cf Fig. 9).

We build some propagation rules such that when a state is guessed, we know if the equation it involves is pseudo-linear or not. Those propagation rules are described in Fig. 10:

Parallel partitioning. To build the structure, the model considers the values and the differences of the bits at the start of the structure known by the lower part and the value of the bits at the end of the structure known by the upper part. Then, with the same propagation rules for values and differences, the model checks if the value of a bit is known by both the upper and lower parts. In this case, this bit will be a fixed one amongst the F fixed bits of the structure. The model also checks if some differences are known by both the upper and lower parts on the non-fixed bits. In this case, we consider that the differences D on this bit recover one bit of key information during the matching step. As we repeat the attack $2^{p-(n-F)}$ times, we add the following constraint: $p - (n - F) > 0$

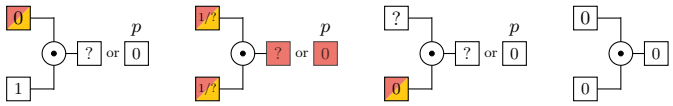


Fig. 6: Rules for the propagation of differences through the AND operator and the values they require.

: The value of this state is not needed, correspond to a *state variable* value of 0

: This state is known by propagation of the plaintext/ciphertext with the key guesses k_{in}/k_{out}, correspond to a *state variable* value of 1

: This state is guessed, correspond to a *state variable* value of 2

: This state is known by propagation or state-tested

Fig. 7: Representation of state value.

to ensure at least one repetition. During the optimisation process, the model will search for the optimal attack, including the structure. It will select the key bits such that the structure can be built during the attack, while minimising the cost of key guesses required to ensure an efficient filter during the match step. Due to the design of SIMON, for a given amount of k_{in} and k_{out}, many different structures exist (all corresponding to a rotation of another) with the same filtering properties; the model will arbitrarily choose one of them.

Objective function. The model computes the value of the following variables through each part of the attack: the key bits guessed in each part: K_{in} and K_{out}, the state-test guessed in each part: s_{in} and s_{out}, the pseudo-linear state-test guessed in each part: s_{in_PL} and s_{out_PL}, the probabilistic transitions in each part: p_{in} and p_{out}, the fixed bits in the structure: F and the matched differences in the structure: D.

The objective function we are using is the time complexity described in equation Eq. 9 without the 2^p term (as we are not searching for the distinguisher, p is fixed and so is not impacting the objective function). We consider two separate cases for the *add* parameter denoting the filtering regarding the non-fixed part of the structure. If the key-schedule is linear, $add = k - k_{in} - k_{out} - s_{in_AL} - s_{out_AL} - D$ as each pseudo-linear state-test gives one bit of information about the key and each matching difference in the structure gives one bit of information about the key. If the total quantity of key bits recovered exceeds the size of the key, we can use this to filter the candidates as previously explained. Considering that $n = 2k$, the match complexity can be rewritten as equation Eq.24:

$$\mathcal{T}_M = 2^{p+3n-2F-D+(s_{in}-s_{in_AL})+(s_{out}-s_{out_AL})} \tag{24}$$

We note that the lower bound is achieved if we can make $F = n$ (implying $D = 0$), s_{in}=s_{in_AL} and s_{out}=s_{out_AL}.

If we consider a non-linear key-schedule, the excessive key information cannot be sieved during the match, we consider that $add = 0$.

Memory model As demonstrated before, considering a non-free memory access model can impact the overall time complexity of the attack. When considering

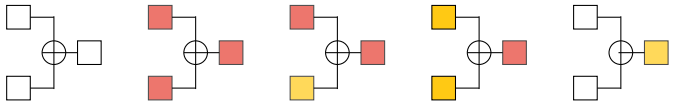


Fig. 8: Rules of propagation for values through the XOR operation, the states before the XOR can be either state bits or key bits.

$\boxed{L}$: The equation involved in this bit is linear considering the plaintext/ciphertext and k_{in}/k_{out}, correspond to a *linear property variable* of 0

$\boxed{A}$: The equation involved in this bit is almost linear considering the plaintext/ciphertext and k_{in}/k_{out}, correspond to a *linear property variable* of 1

$\boxed{N}$: The equation involved in this bit is non-linear considering the plaintext/ciphertext and k_{in}/k_{out}, correspond to a *linear property variable* of 2

Fig. 9: The different possible linearity properties of a bit.

that in the model, we add to the time complexity the term corresponding to the memory accesses :

$$\mathcal{T}_E = (2^{\mathcal{M}_{up}} + 2^{\mathcal{M}_{down}})2^{\lceil probability \rceil + pin + pout - (n-F)} \cdot \mathcal{M}. \tag{25}$$

By using an objective function of the type $f = 2^x + 2^y + 2^z$, we are limited by the computer's 'int' representation of the values of x, y and z below 128. This limits the search for attacks where $n < 128$. In the case $n \geq 128$, we must adapt the objective function to $f = x + y + z$, which is no longer optimal and can be very far from the optimal result. The search for attacks on SIMON 128/256 could not be optimal; still, we found a better attack than the existing differential ones, but not the linear ones. However, for other instances of this cipher, we expect to find better differential MITM attacks than the current linear ones and, in particular, a better differential MITM attack using this differential distinguisher to exist, since our search is not optimal. We present this attack in [21].

5 Differential Meet-in-the-Middle attacks on CLEFIA-256

The best previous attacks on CLEFIA-256 have reached up to 15 rounds [31] since 2010. Other attacks like [19,33] have been cited reaching the same number of rounds, but none increases the number of attacked rounds. These best-known attacks reach 15 rounds out of 26 and recover 256 bits of information on the subkeys in less time than brute force. However, the authors do not say how to recover efficiently the master key from the subkeys information. As the number of candidates is often quite high and the bits in the subkeys should be completed in order to compute the whitening keys, there is no obvious way for recovering the full master key from that information in previous attacks. In this section, we propose the first attack on 16 rounds that recovers the full master key. In [21], we propose an improved attack on 15 rounds.

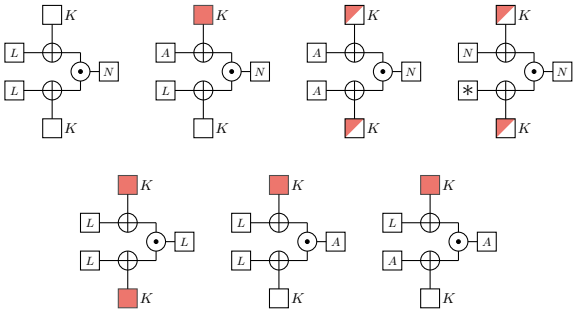


Fig. 10: The different possible propagation of linearity properties, $\square_K$ represents a key bit, filled in red means a known key bit, in white an unknown key bit. $*$ mean this bit can be either L, A or N.

5.1 Brief description of CLEFIA

CLEFIA [26] is a family of 128-bit block ciphers using a key of size $k \in \{128, 192, 256\}$ and is iterated $r \in \{18, 22, 26\}$ times depending on the size of the key. The cipher has a structure of a 4-branch Feistel network and for $0 \leq i < r$ we denote by LL_i, LR_i, RL_i, RR_i the 4 32-bit blocks forming the state of the i-th round and by kL_i and kR_i the round subkeys of the i-th round. We denote by WK_0, WK_1, WK_2, WK_3, the four whitening keys added to LR_0, RR_0, LR_r and RR_r respectively. Then, the encryption process is as follows:

$$LL_{i+1} = F_0(LL_i, kL_i) \oplus LR_i, \quad LR_{i+1} = RL_i, \tag{26}$$
$$RL_{i+1} = F_1(RL_i, kR_i) \oplus RR_i, \quad RR_{i+1} = LL_i, \tag{27}$$

Here F_0 and F_1 are the two round functions of CLEFIA and have the same structure. First, we XOR the round's subkey then we apply a layer composed of four 8×8 sboxes (S_0 or S_1) and finally we mix the bytes by applying a 4×4 Hadamard-type matrix, M_0 for F_0 and M_1 for F_1 defined as follows:

$$M_0 = \begin{pmatrix} 0x01 & 0x02 & 0x04 & 0x06 \\ 0x02 & 0x01 & 0x06 & 0x04 \\ 0x04 & 0x06 & 0x01 & 0x02 \\ 0x06 & 0x04 & 0x02 & 0x01 \end{pmatrix} \text{ and } M_1 = \begin{pmatrix} 0x01 & 0x08 & 0x02 & 0x0a \\ 0x08 & 0x01 & 0x0a & 0x02 \\ 0x02 & 0x0a & 0x01 & 0x08 \\ 0x0a & 0x02 & 0x08 & 0x01 \end{pmatrix}.$$

A more detailed description of CLEFIA is given in [26].

5.2 Attack on 16 rounds of CLEFIA-256

In this section, we present a single-key truncated differential Meet-in-the-Middle attack on CLEFIA-256 covering 16 rounds, which is the best-known attack on this instance of CLEFIA. In our attack, we additionally recover the whole master key, unlike previous attacks on 15 rounds. The non-linearity of the key schedule of CLEFIA and the addition of the whitening-key, which is the XOR of the two parts of the master key, make it difficult to compute the whole master key from partial knowledge of bits of the subkeys, besides completing the unknown bits

and computing the full key-schedule. But for this we will have to try all the recovered candidates for the bits of information, which is usually high. In order to solve this problem, we use a differential MITM attack with a counter, as proposed in [20] to reduce the number of candidates remaining after the attack, such that we can guess some missing key bits that will be used to recover the whole master key through the key-schedule. This attack is the first one to recover the full master key and not just 256 bits of information on the subkeys in less time than brute force. In addition, it covers one more round than the previously best-known attacks. At the end of this section, we will also propose some other attacks when the full key is not recovered, that also improve upon previous results. In our attack, we consider the distinguisher established in [19]:

$$(\texttt{0000}, \texttt{0000}, \texttt{0000}, ?\texttt{000}) \xrightarrow[10r]{2^{-104}} (\texttt{M}_0(?\texttt{000}), \texttt{0000}, \texttt{0000}, ?\texttt{000}).$$

This distinguisher is found by appending one round with probability 1 after the following 9-round truncated differential distinguisher:

$$(\texttt{0000}, ?\texttt{000}, \texttt{0000}, \texttt{0000}) \xrightarrow[9r]{2^{-104}} (?\texttt{000}, \texttt{0000}, \texttt{0000}, \texttt{0000}).$$

We can observe that the difference $M_0(?000)$ in LL_{14}, as shown in Fig. 11, depends only on the output difference of the sbox S_0 with the input difference RR_{14}. Moreover, on average, for a fixed input difference, there are $2^{6.64}$ possible output differences. Thus, the size $2^{\delta_{out}}$ of the possible output difference of the distinguisher is $2^{14.64}$. Our attack is divided into four parts: the first round covered by an initial structure using the parallel partitioning technique, three key-recovery rounds in the upper part, a ten round distinguisher and two key-recovery rounds in the lower part. Here, unlike the attacks on SIMON, we can only build a 1-round structure. Indeed, for SIMON -like ciphers, the bit-wise nature of their operations, coupled with the linearity of their key schedule, allows us to build longer structures, as we can match the upper and lower parts with minimal key guesses, 18 bits for 3 rounds and a filtering of 54 bits, and we can divide the bits needed in the structure between the lower and upper parts. For CLEFIA , we could not add more rounds and match 128 bits between both parts without guessing 64 bits of the subkeys making the time complexity of the lower or upper part too high. Since we are using counters, we repeat the attack $2^a = 2^6$ times with new data to improve the signal-to-noise ratio and reduce therefore the number of triplet candidates. The detailed path of the attack is given in Fig. 11 and the parameters are as follows:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	Δ_{in}	Δ_{out}	F	a
16	1	3	10	2	104	72	128	40	0	0	0	8	14.64	128	6

Equivalent subkeys. Instead of guessing the whitening keys separately, we guess the needed XOR values of the whitening key and the round-key in the even rounds of the upper and lower parts of the attack.
State-test technique. We use the state-test technique to reduce k_{in} by guessing 5 bytes of the state instead of 11 bytes of the round keys. The 5 bytes of state-test

give us the following non-linear equations over the key bits:

$$
\begin{aligned}
X_1 =& \mathbf{kL_2[0]} \oplus \mathbf{LL_2[0]} \\
=& \mathbf{kL_2[0]} \oplus LR_1[0] \oplus (0x01 S_0(LL_1[0] \oplus kL_1[0]) \\
&+ 0x02 S_0(LL_1[1] \oplus \mathbf{kL_1[1]}) + 0x04 \mathbf{S_0}(LL_1[2] \oplus \mathbf{kL_1[2]}) + 0x01 \mathbf{S_0}(LL_1[3] \oplus \mathbf{kL_1[3]})) \\
X_2 =& \mathbf{kL_2[1]} \oplus \mathbf{LL_2[1]} \\
=& \mathbf{kL_2[1]} \oplus LR_1[1] \oplus (0x02 S_0(LL_1[0] \oplus kL_1[0]) \\
&+ 0x01 S_0(LL_1[1] \oplus \mathbf{kL_1[1]}) + 0x06 \mathbf{S_0}(LL_1[2] \oplus \mathbf{kL_1[2]}) + 0x01 \mathbf{S_0}(LL_1[3] \oplus \mathbf{kL_1[3]})) \\
X_3 =& \mathbf{kL_2[2]} \oplus \mathbf{LL_2[2]} \\
=& \mathbf{kL_2[2]} \oplus LR_1[2] \oplus (0x04 S_0(LL_1[0] \oplus kL_1[0]) \\
&+ 0x06 S_0(LL_1[1] \oplus \mathbf{kL_1[1]}) + 0x01 \mathbf{S_0}(LL_1[2] \oplus \mathbf{kL_1[2]}) + 0x01 \mathbf{S_0}(LL_1[3] \oplus \mathbf{kL_1[3]})) \\
X_4 =& \mathbf{kL_2[3]} \oplus \mathbf{LL_2[3]} \\
=& \mathbf{kL_2[3]} \oplus LR_1[3] \oplus (0x06 S_0(LL_1[0] \oplus kL_1[0]) \\
&+ 0x04 S_0(LL_1[1] \oplus \mathbf{kL_1[1]}) + 0x02 \mathbf{S_0}(LL_1[2] \oplus \mathbf{kL_1[2]}) + 0x01 \mathbf{S_0}(LL_1[3] \oplus \mathbf{kL_1[3]})) \\
X_5 =& \mathbf{kR_3[0]} \oplus \mathbf{RL_3[0]} \\
=& \mathbf{kR_3[0]} \oplus RR_2[0] \oplus (0x01 \mathbf{S_1}(RL_2[0] \oplus \mathbf{kR_2[0]}) \\
&+ 0x08 \mathbf{S_1}(RL_2[1] \oplus \mathbf{kR_2[1]}) + 0x02 \mathbf{S_1}(RL_2[2] \oplus \mathbf{kR_2[2]}) + 0x0a \mathbf{S_1}(RL_2[3] \oplus \mathbf{kR_2[3]}))
\end{aligned}
$$

Since we are using counters to reduce the number of candidates for the right key, we want to take into account in the counter some of those state-test equations. As proposed in [20], we will fix some bits of LL_1 to a chosen value for the whole attack to create a partition of the key so that the value of X_1, X_2, X_3 and X_4 can be taken into account in the counters. However, the number of bits we can fix is upper bounded by the difference between the data available and the amount of data we need to perform the attack, namely $128 - (p - \Delta_{in} + p_{in} + p_{out} + a) = 128 - 104 + 8 - 6 = 26$. Thus, we can fix three bytes of LL_1 of size 24 (and we therefore still have two free bits that will allow us to fix one additional bit at the end to reduce the data complexity – this is explained in the end of the attack). We chose to fix the bytes $LL_1[1]$, $LL_1[2]$ and $LL_1[3]$ as shown in Fig.11 to a certain value for the whole attack (0 for instance).

Steps to find the candidate keys.

We fix the bytes $LL_1[1]$, $LL_1[2]$ and $LL_1[3]$ to the fixed value 0. Next,

1. Build the structures with the parallel partitioning technique by fixing the set $\mathcal{F}$ composed of the bytes of RL_0, RR_0 and RR_1 and the byte $LL_1[0]$.
2. *Upper part*: Generation of the $\{LL_1|LR_1|RL_1|RR_1,\ \tilde{LL_1}|\tilde{LR_1}|\tilde{RL_1}|\tilde{RR_1}, i\}_{\mathcal{F}}$ triplets.
   (a) Guess the 72 bits of K_{in} and the 40 bits of S_{in}.
   (b) For each guess i of K_{in}, compute the value of $LL_1|LR_1|RL_1|RR_1$.
   (c) For each guess i of (K_{in}, S_{in}) and for each value of δ_{in} compute $\tilde{LL_1}|\tilde{LR_1}|\tilde{RL_1}|\tilde{RR_1}$ such that after 3 rounds, $LL_1|LR_1|RL_1|RR_1$ and $\tilde{LL_1}|\tilde{LR_1}|\tilde{RL_1}|\tilde{RR_1}$ verify the constraints at the start of the distinguisher. Store the triplets in a hash table $\mathcal{H}$ of size $2^{72+40+8} = 2^{120}$.
3. *Lower part*: Generation of the $\{LL_0|LR_0|RL_0|RR_0,\ \tilde{LL_0}|\tilde{LR_0}|\tilde{RL_0}|\tilde{RR_0}, j\}_{\mathcal{F}}$ triplets.
   (a) Guess the 128 bits of K_{out}.
   (b) For each guess j of K_{out} compute $LL_0^j|LR_0^j|RL_0^j|RR_0^j$.
   (c) Request the encryption of $P = LL_0^j|LR_0^j|RL_0^j|RR_0^j$, obtaining $C = LL_{16}^j|LR_{16}^j|RL_{16}^j|RR_{16}^j$.

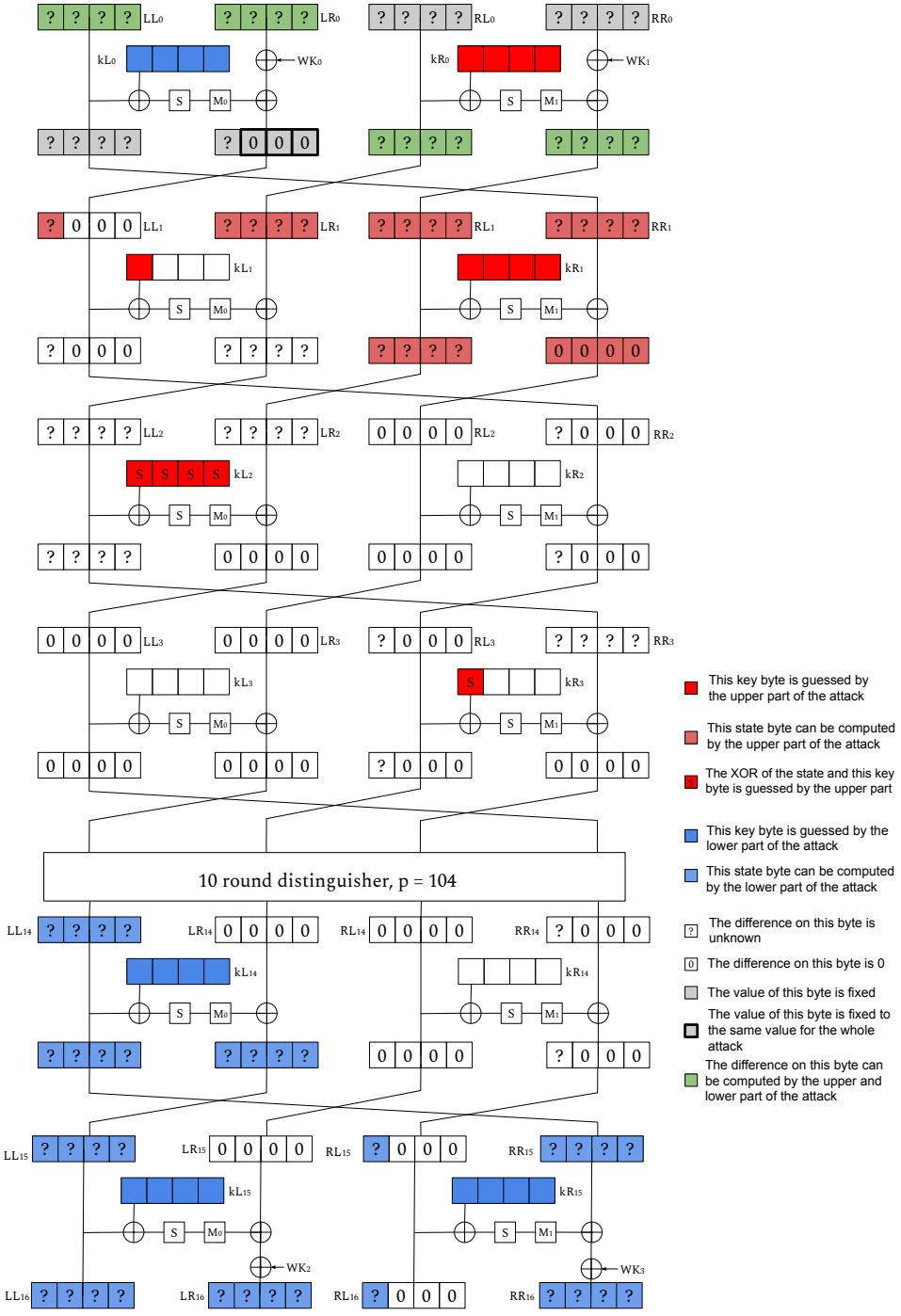


Fig. 11: Path of the attack on 16 rounds of CLEFIA-256.

(d) For each guess j of K_{out} and each value of δ_{out} compute $\tilde{LL}_{16}^{j}|\tilde{LR}_{16}^{j}|\tilde{RL}_{16}^{j}|\tilde{RR}_{16}^{j}$ such that two rounds before, $LL_{16}^{j}|LR_{16}^{j}|RL_{16}^{j}|RR_{16}^{j}$ and $\tilde{LL}_{16}^{j}|\tilde{LR}_{16}^{j}|\tilde{RL}_{16}^{j}|\tilde{RR}_{16}^{j}$ verify the constraints at the end of the distinguisher.

(e) For $l \in \{1, 2, 3\}$, verify that $F_0(\tilde{LL}_0 \oplus kL_0)[l] \oplus F_0(LL_0 \oplus kL_0)[l] \oplus \tilde{LR}_0[l] \oplus LR_0[l] = 0$: pre-sieve 2^{-24} candidates.

4. *Match*: As each of the $2^{128+14.64-24} = 2^{118.64}$ lower triplets are generated, verify the consistency of each lower triplets with all the upper triplets stored in $\mathcal{H}$. The match is performed on the following differences:
   (a) $LL_0 \oplus \tilde{LL}_0$ are equal to $RR_1 \oplus \tilde{RR}_1$: a 32 bit filter.
   (b) $RL_0 \oplus \tilde{RL}_0$ are equal to $LR_1 \oplus \tilde{LR}_1$ respectively: a 32 bit filter.
   (c) $F_0(\tilde{LL}_0 \oplus kL_0)[0] \oplus F_0(LL_0 \oplus kL_0)[0] \oplus \tilde{LR}_0[0] \oplus LR_0[0]$ is equal to $\tilde{LL}_1[0] \oplus LL_1[0]$: a 8 bits filter.
   (d) $F_0(\tilde{LR}_1 \oplus kR_0) \oplus F_0(LR_1 \oplus kR_0) \oplus \tilde{RL}_1 \oplus RL_1$ are equal to $RR_0 \oplus \tilde{RR}_0$: a 32 bit filter.
5. Repeat from step 1, $2^a \cdot 2^{96} = 2^6 \cdot 2^{96}$ times.

The time complexity at the end of those 5 steps is:

$$\begin{aligned}\mathcal{T}_E &= 2^6 2^{104-8} \cdot (2^8 2^{72+40} + 2^{14.64} 2^{128} 2^{-24} + 2^{120+118.64-104}) \\ &= 2^{222} + 2^{220.64} + 2^{236.64} \approx 2^{236.64} \text{ encryptions.} \quad (28)\end{aligned}$$

When considering the cost of memory accesses with a hash table of size 2^{121}:

$$\mathcal{T}_M = (2^{222} + 2^{220.64}) C_A, \quad (29)$$

for a total time complexity of $\mathcal{T} = \mathcal{T}_E + \mathcal{T}_M$. In order to verify if this complexity is smaller than exhaustive search, we consider a cost $C_A \approx 121 = 2^{6.9}$. Then $\mathcal{T}_M \approx 2^{222.47} 2^{6.9} = 2^{229.37}$ equivalent encryptions, which is lower than exhaustive search and not the bottleneck of the attack.

We recover $2^{236.64}$ candidates for 232 bits of information on the key and 8 bits of state-test with X_5. We consider a counter of size 2^{232} for the key candidates, not taking into account the state-test equation from X_5. For each possible value of the 232 key bits, we try in total $N = 2^{6+104-8+8+14.64+8} = 2^{132.64}$ data [2], and the probability that a given pair of data is retained in the counter is of 2^{-128}, as it corresponds to the probability of performing a match between $\tilde{P}$ and $\tilde{X}$. For each wrong key, its associated counter will follow a binomial distribution $\mathcal{B}(2^{132.64}, 2^{-128})$ while the counter for the right key follows the binomial distribution, $\mathcal{B}(2^{110}, 2^{-104})$. Thus we can distinguish the right key from the wrong keys, as we show in the next paragraph. We had chosen $a = 6$ to have good advantage in the remaining part of the attack.

Filtering phase. Given the previous binomial distributions, we choose to keep the key candidates that appear at least $t = 60$ times. With this parameter, the probability of non-detection of the right key becomes 0.29, which implies a

[2] We can verify that $2^{132.64+232-128} = 2^{236.64}$, which is the number of candidates we had in the previous step

success probability for the attack higher than 71%. The probability for a wrong key to be kept in this first filtering is of $P_t = 2^{-28.96}$, and we therefore keep $2^{232-28.96} = 2^{203.04}$ key candidates. We will now use the remaining state-test equation X_5 on 8 bits to filter further. They depend non-linearly on the bits of kR_2 and linearly on $kR_3[0]$. Since each remaining candidate key appears at least 60 times, then we have, for each candidate, 60 times the 8-bit equation given by X_5 with different data depending on the same 40 unknown key bits. To facilitate the study of these equations, we guess the 32 bits of kR_2.

For each of the $2^{203.04+32} = 2^{235.04}$ candidates, we have 60 linear equations, denoted by X_5^l for $1 \leq l \leq 60$. For the right key, a big number of these equations will be consistent with each other. We choose $t' = 34$, asking the remaining candidates to be consistent in at least 36 out of the 60 kept equations. This will happen for the right key. For the wrong key, instead, it will occur with probability that can be approximated by $2^{-8\cdot(t'-1)}\binom{t}{t'} = 2^{-272}2^{55} = 2^{-217}$. The probability of verifying t' equations randomly is now $2^{-8\cdot(t'-1)}$ for a fixed set of t' equations, as the first one will determine the value of $kR_3[0]$, and for the remaining values all is fixed, and we want to verify an 8-bit equation each time. Thus, we expect at this point to keep only $2^{235.04-217} = 2^{18.04}$ candidates for the 232 bits of the key we had in the beginning, plus the 40 bits from kR_2 and $kR_3[0]$ that are now fully determined (so 272 bits of information on the key in total). However, considering the non-linearity of the key schedule and the addition of the whitening keys, we have not recovered enough subkey bits yet to reverse the key schedule and recover the master key.

Recovering the whole master key. In the first round of the attack, we have recovered, for each kept candidate, the value of the subkeys kL_0 and kR_0 which are equal to the first 64 bits of the subkey L. Thus, we can exhaustively search for the 64 remaining bits of L and the 128 bits of R such that we can reverse the key schedule and find the master key KL and KR. We can then check that the master key found is consistent with the 208 bits of round keys that we guessed during the attack which leads us to one candidate key. The time complexity of this step is less than $\mathcal{T}_{exh} = 2^{18.04} \times 2^{192} = 2^{210.04}$ CLEFIA encryptions, as we only kept $2^{18.04}$ candidates in the previous step.

The time complexity of the whole attack is dominated by the time complexity of the matching step. Thus $\mathcal{T} = \mathcal{T}_E = 2^{236.64}$. The memory complexity is of $\mathcal{M} = 2^{232}$ to store the counters. And by using the data improvement technique from [8], we can choose $x = 1$ as we have two remaining free bits in the plaintext, as previously mentioned, leading to a data complexity of $\mathcal{D} = 2^{128-1} = 2^{127}$.

6 Conclusion

We have proposed new applications of differential MITM attacks on Feistel constructions that improve upon previously known results. We believe that Feistel constructions might be a particularly good target due to the fact that they naturally can lead to longer initial structures. We have proposed the first tool that builds differential MITM attacks on some ciphers, including all the previously proposed optimisations.

So far, we did not analyse the Feistel ciphers SPECK, LEA and HIGHT because of the modular addition used in the round function, complicating the analysis and the modelling. We leave this as an interesting open problem. Similarly, the analysis of Camellia could be more complex because of the use of specific non-linear functions. WARP and ShangMi4 are not good candidates for differential Meet-in-the-Middle attacks, as their instances consider a key size equal to the size of the block. For many other Feistel ciphers such as SAND, KASUMI or Blowfish, the lack of existing differential distinguishers has not allowed to easily build improved differential MITM attacks. We searched for attacks on TWINE without success, but we expect that searching for new truncated differential distinguishers at the same time as the key-recovery extensions will lead to better attacks than the existing ones. Building a tool that optimises both parts simultaneously would be a panacea to look forward, but several technical issues and open problems are yet to be solved before reaching it.

Acknowledgements . This research is funded by the European Union (ERC2023-COG, SoBaSyC, 101125450). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. The authors thank the anonymous reviewers for their insightful comments, which helped improve the quality of this work.

References

1. Abdelraheem, M.A., Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P.: Improved linear cryptanalysis of reduced-round SIMON-32 and SIMON-48. In: Biryukov, A., Goyal, V. (eds.) Progress in Cryptology - INDOCRYPT 2015 - 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9462, pp. 153–179. Springer (2015). https://doi.org/10.1007/978-3-319-26617-6_9, https://doi.org/10.1007/978-3-319-26617-6_9
2. Ahmadian, Z., Khalesi, A., M'foukh, D., Moghimi, H., Naya-Plasencia, M.: Improved differential meet-in-the-middle cryptanalysis. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 280–309. Springer (2024)
3. Ashur, T.: Improved linear trails for the block cipher simon. IACR Cryptol. ePrint Arch. p. 285 (2015), http://eprint.iacr.org/2015/285
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch. p. 404 (2013), http://eprint.iacr.org/2013/404
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015. pp. 175:1–175:6. ACM (2015)
6. Beyne, T., Rijmen, V.: Differential cryptanalysis in the fixed-key model. 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 III. Lecture Notes in Computer Science, vol. 13509, pp. 687–716. Springer (2022)
7. Blondeau, C., Gérard, B., Tillich, J.: Accurate estimates of the data complexity and success probability for various cryptanalyses. Des. Codes Cryptogr. **59**(1-3), 3–34 (2011). https://doi.org/10.1007/S10623-010-9452-2, https://doi.org/10.1007/s10623-010-9452-2
8. 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)
9. Boura, C., Lallemand, V., Naya-Plasencia, M., Suder, V.: Making the impossible possible. J. Cryptol. **31**(1), 101–133 (2018). https://doi.org/10.1007/S00145-016-9251-7, https://doi.org/10.1007/s00145-016-9251-7
10. 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)
11. Chakraborty, D., Hadipour, H., Nguyen, P.H., Eichlseder, M.: Finding complete impossible differential attacks on andrx ciphers and efficient distinguishers for ARX designs. IACR Trans. Symmetric Cryptol. **2024**(3), 84–176

(2024). https://doi.org/10.46586/TOSC.V2024.I3.84-176, https://doi.org/10.46586/tosc.v2024.i3.84-176

12. Chakraborty, D., Sahoo, S., Nguyen, P.H., Sarkar, S.: An automated model for differential meet in the middle attacks on andrx ciphers. Scientific Reports **16**(1), 14773 (2026)
13. Deng, W., Zhang, J., Wang, H.: Improved differential meet-in-the-middle cryptanalysis on SIMON and piccolo. In: Susilo, W., Pieprzyk, J. (eds.) Information Security and Privacy - 30th Australasian Conference, ACISP 2025, Wollongong, NSW, Australia, July 14-16, 2025, Proceedings, Part I. Lecture Notes in Computer Science, vol. 15658, pp. 78–97. Springer (2025). https://doi.org/10.1007/978-981-96-9095-4_5, https://doi.org/10.1007/978-981-96-9095-4_5
14. 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). https://doi.org/10.1007/978-3-030-84252-9_10, https://doi.org/10.1007/978-3-030-84252-9_10
15. 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)
16. Isobe, T., Shibutani, K.: Generic key recovery attack on feistel scheme. IACR Cryptol. ePrint Arch. p. 526 (2015)
17. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 161–185. Springer (2015). https://doi.org/10.1007/978-3-662-47989-6_8, https://doi.org/10.1007/978-3-662-47989-6_8
18. Leurent, G., Pernot, C., Schrottenloher, A.: Clustering effect in simon and simeck. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13090, pp. 272–302. Springer (2021). https://doi.org/10.1007/978-3-030-92062-3_10, https://doi.org/10.1007/978-3-030-92062-3_10
19. Li, L., Jia, K., Wang, X., Dong, X.: Meet-in-the-middle technique for truncated differential and its applications to CLEFIA and camellia. In: Leander, G. (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9054, pp. 48–70. Springer (2015)
20. M'foukh, D., Naya-Plasencia, M., Neumann, P.: The state-test technique on differential attacks: a 26-round attack on CRAFT and other applications. IACR Cryptol. ePrint Arch. p. 1597 (2025), https://eprint.iacr.org/2025/1597
21. Michel, B., M'foukh, D., Naya-Plasencia, M.: Differential meet-in-the-middle attacks on feistel ciphers. IACR Cryptol. ePrint Arch. **2025**, 1911 (2025), https://eprint.iacr.org/2025/1911

22. MINAUD, B.: Meet-in-the-Middle Cryptanalysis, chap. 7, pp. 89–100. John Wiley & Sons, Ltd (2024). https://doi.org/https://doi.org/10.1002/9781394256327.ch7, https://onlinelibrary.wiley.com/doi/abs/10.1002/9781394256327.ch7
23. Niu, C., Li, M., Zhang, J., Wang, M.: Improved differential and linear cryptanalysis on round-reduced SIMON. Des. Codes Cryptogr. **94**(5), 107 (2026). https://doi.org/10.1007/S10623-026-01827-9, https://doi.org/10.1007/s10623-026-01827-9
24. Qiao, K., Hu, L., Sun, S.: Differential analysis on simeck and SIMON with dynamic key-guessing techniques. In: Camp, O., Furnell, S., Mori, P. (eds.) Information Systems Security and Privacy - Second International Conference, ICISSP 2016, Rome, Italy, February 19-21, 2016, Revised Selected Papers. Communications in Computer and Information Science, vol. 691, pp. 64–85. Springer (2016). https://doi.org/10.1007/978-3-319-54433-5_5, https://doi.org/10.1007/978-3-319-54433-5_5
25. Qin, L., Chen, H., Wang, X.: Linear hull attack on round-reduced simeck with dynamic key-guessing techniques. In: Liu, J.K., Steinfeld, R. (eds.) Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9723, pp. 409–424. Springer (2016). https://doi.org/10.1007/978-3-319-40367-0_26, https://doi.org/10.1007/978-3-319-40367-0_26
26. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (extended abstract). In: Biryukov, A. (ed.) Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4593, pp. 181–195. Springer (2007). https://doi.org/10.1007/978-3-540-74619-5_12, https://doi.org/10.1007/978-3-540-74619-5_12
27. Song, L., Hu, L., Ma, B., Shi, D.: Match box meet-in-the-middle attacks on the SIMON family of block ciphers. In: Eisenbarth, T., Öztürk, E. (eds.) Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8898, pp. 140–151. Springer (2014). https://doi.org/10.1007/978-3-319-16363-5_9, https://doi.org/10.1007/978-3-319-16363-5_9
28. Song, L., Liu, H., Yang, Q., Chen, Y., Hu, L., Weng, J.: Generic differential key recovery attacks and beyond. In: Chung, K., Sasaki, Y. (eds.) Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part VII. Lecture Notes in Computer Science, vol. 15490, pp. 361–391. Springer (2024). https://doi.org/10.1007/978-981-96-0941-3_12, https://doi.org/10.1007/978-981-96-0941-3_12
29. Song, L., Yang, Q., Chen, Y., Hu, L., Weng, J.: Probabilistic extensions: A one-step framework for finding rectangle attacks and beyond. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 339–367. Springer (2024)
30. Sun, L., Fu, K., Wang, M.: Improved zero-correlation cryptanalysis on SIMON. In: Lin, D., Wang, X., Yung, M. (eds.) Information Security and Cryptology - 11th International Conference, Inscrypt 2015, Beijing, China, November 1-3, 2015,

Revised Selected Papers. Lecture Notes in Computer Science, vol. 9589, pp. 125–143. Springer (2015)
31. Tezcan, C.: The improbable differential attack: Cryptanalysis of reduced round CLEFIA. In: Gong, G., Gupta, K.C. (eds.) Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6498, pp. 197–209. Springer (2010)
32. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The simeck family of lightweight block ciphers. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9293, pp. 307–329. Springer (2015)
33. Yi, W., Chen, S.: Improved integral and zero-correlation linear cryptanalysis of reduced-round CLEFIA block cipher. IACR Cryptol. ePrint Arch. p. 149 (2016)

Supplementary Material

A Round function and parameters of Simon

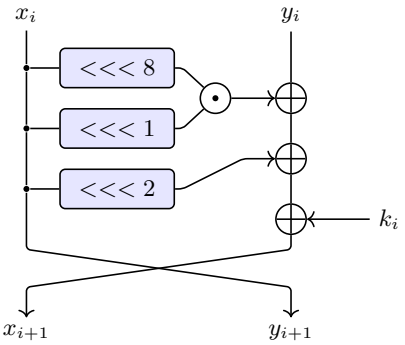


Fig. 12: Description of one round of Simon.

n	32	48		64		96		128		
k	64	72	96	96	128	96	144	128	192	256
m	4	3	4	3	4	2	3	2	3	4
r	32	36	36	42	44	52	54	68	69	72

Table 4: Parameters of the different instances of Simon.

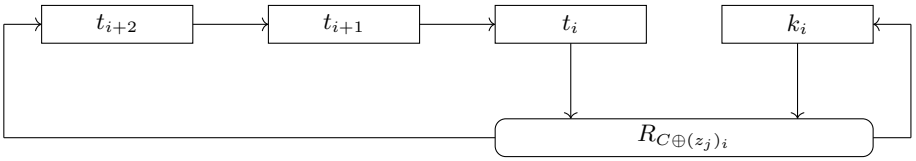


Fig. 13: SIMECK key schedule.

B Equivalent subkey technique

A strategy to reduce the number of subkey bits guessed during the key-recovery steps of an attack is to move the step of adding the key in the round function to optimise the number of subkey bits we need to know. This transformation defines an equivalent subkey. This technique has been introduced by Isobe et al. in [16] for Feistel networks and has been adapted by Sun et al. in [30] for zero-correlation attacks on SIMON.
For the SIMON family of block cipher, the idea is to move the subkey of the i-th round to the $(i+1)$-th round, for $0 \leq i \leq r_U - 1$ in the upper part and to move the subkey of the i-th round to the $(i-1)$-th round, for $(r_U + r_D) \leq i \leq (r_U + r_D + r_L - 1)$ in the lower part as shown in fig. 14. The new equivalent subkeys are defined as follows:

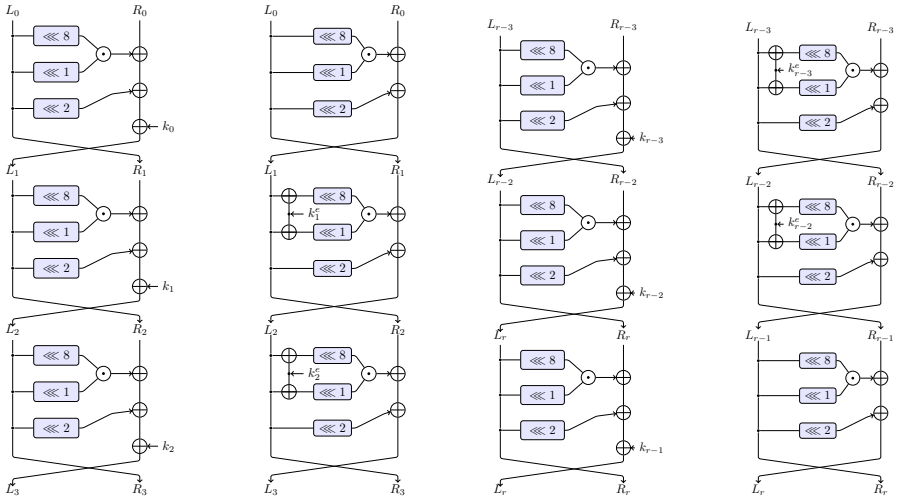
$$\begin{cases} k_1^e = k_0 \\ k_2^e = (k_1^e \lll 2) \oplus k_1 \\ k_3^e = k_1^e \oplus (k_2^e \lll 2) \oplus k_2 \\ \vdots \\ k_i^e = k_{i-2}^e \oplus (k_{i-1}^e \lll 2) \oplus k_{i-1} \\ \vdots \\ k_{r_U}^e = k_{r_U-2}^e \oplus (k_{r_U-1}^e \lll 2) \oplus k_{r_U-1} \end{cases} \quad \begin{cases} k_{r-2}^e = k_{r-1} \\ k_{r-3}^e = (k_{r-2}^e \lll 2) \oplus k_{r-2} \\ k_{r-4}^e = k_{r-2}^e \oplus (k_{r-3}^e \lll 2) \oplus k_{r-3} \\ \vdots \\ k_{r-i}^e = k_{r-(i-2)}^e \\ \qquad \oplus (k_{r-(i-1)}^e \lll 2) \oplus k_{r-(i-1)} \\ \vdots \\ k_{r_U+r_D-1}^e = k_{r_U+r_D+1}^e \oplus k_{r_U+r_D} \\ \qquad \oplus (k_{r_U+r_D}^e \lll 2) \end{cases}$$

The exact same technique can be used on SIMECK.

C CLEFIA's key schedule description

Key schedule The key schedule of CLEFIA uses 92 blocks of 32-bit constants CON_i and a function called the Double Swap $\Sigma : \{0,1\}^{128} \to \{0,1\}^{128}$ defined as: $X_{128} \mapsto X[7-63]|X[121-127]|X[0-6]|X[64-120]$.
For CLEFIA with a 256 bits master key $K = K_L|K_R$, the key schedule is as follows:

1. Apply 10 rounds of the round function of CLEFIA to K, with the constant CON_i, $0 \leq i < 40$ as keys and obtain two 128 bits subkeys L and R.
2. $WK_0, WK_1, WK_2, WK_3 \leftarrow K_L \oplus K_R$ are four whitening keys.



(a) Original subkeys of the upper part.

(b) Equivalent subkeys of the upper part.

(c) Original subkeys of the lower part.

(d) Equivalent subkeys of the lower part.

Fig. 14: The equivalent subkey technique applied to SIMON.

3. The round keys kL_i and kR_i, $0 \leq i \leq 26$ are defined as follows for $i = 0$ to 12:
   - if $i \bmod 4 = 0$ or 1:
     $T \leftarrow L \oplus (CON_{40+4i}|CON_{40+4i+1}|CON_{40+4i+2}|CON_{40+4i+3})$
     $L \leftarrow \Sigma(L)$
     if i is odd: $T \leftarrow T \oplus K_R$
   - else:
     $T \leftarrow R \oplus (CON_{40+4i}|CON_{40+4i+1}|CON_{40+4i+2}|CON_{40+4i+3})$
     $R \leftarrow \Sigma(R)$
     if i is odd: $T \leftarrow T \oplus K_L$
   - $kL_i|kR_i|kL_{i+1}|kR_{i+1} \leftarrow T$.

D Additional attacks against SIMON n/2n

In this section, we briefly describe the parameters and attacks on the other variants; all the figures are provided in the GitHub repository. All these attacks work in a similar way as the one on SIMON 32/64 we described in Section 3.2. All these attacks have been found with the tool described in section 4.

D.1 24-round attack against Simon 32/64

The following attack against 24 rounds of Simon 32/64 harnesses the following differential distinguisher presented in [17]:

$$(\mathtt{0000}, \mathtt{0040}) \xrightarrow[13r]{2^{-28.79}} (\mathtt{4000}, \mathtt{0000}).$$

The path of the attack is given in the GitHub repository. The parameters of the attacks are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
24	3	4	13	4	28.79	21	22	3	4	1	2	2

As all the state-test equations imply almost linear equations, the data and memory complexities of this attack are:

$$\mathcal{M} = 2^{56} \quad \mathcal{D} = 2^{32}.$$

The time complexity of the attack is composed of $\mathcal{T}_E = 2^{60.8}$ encryptions. To compare to exhaustive search, and following the memory access model we mentioned in the preliminaries, we consider $C_A = 56$. This leads to $\mathcal{T}_M \approx (2^{56} + 2^{57}) \cdot 56 = 2^{63.3}$ encryptions, leading to a total time complexity of $2^{63.3}$. This attack was not optimised considering the memory access complexity. We can build an other one, more optimal in the non free memory access model. The parameters of the attack are :

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
24	3	4	13	4	28.79	20	22	3	3	1	2	2

All the state-test of the upper part are filtered but only 2 on 3 in the lower part. The complexity of this attack are :

$$\mathcal{T} = 2^{61.8}, \mathcal{T}_{\mathcal{M}} = 2^{61.27}, \mathcal{M} = 2^{55}, \mathcal{D} = 2^{32}$$

D.2 28-round attack against Simon 48/96

The path of the attack is given in the GitHub repository. The parameters of the attack are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
28	3	4	17	4	45.49	25	26	5	5	0	2	2

As all state-test equations imply almost linear equations, the data and memory complexities of this attack are:

$$\mathcal{M} = 2^{77} \quad \mathcal{D} = 2^{48}.$$

The time complexity of the attack is composed of $\mathcal{T}_E = 2^{93.5}$ equivalent encryptions and $\mathcal{T}_M = (2^{77} + 2^{78})log2(77) = 2^{84.852}$ (negligible).

We think that a better Differential Meet-in-the-Middle attack can be found on 28 rounds of SIMON by taking a distinguisher on 16 rounds with a lower probability and increase the size of the upper part by 1 as in the attack on 29 rounds. By taking a smaller distinguisher, we decrease the lower bound of Diff MITM: 2^{p+k-n}, as this lower bound is achieved in the previous attack we expect to find a better time complexity with a more probable distinguisher. As searching for differential trails on SIMON is a complex task, we consider it out of the scope of this paper.

D.3 Attack on 29 rounds of SIMON 48/96

We have been able to improve by two rounds the best-known attacks on this variant. To mount our attack, we used the 17-round differential distinguisher presented in [23]:

$$(\texttt{000000}, \texttt{800000}) \overset{2^{-45.49}}{\underset{17r}{\longrightarrow}} (\texttt{800000}, \texttt{000000}).$$

The path of the attack is given in the GitHub repository. And the parameters are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
29	3	5	17	4	45.49	31	32	10	6	2	0	2

As all state-test equations imply almost linear equations, and we are able to apply the data reduction technique with $x = 2$, the data and memory complexities of this attack are:

$$\mathcal{M} = 2^{86} \quad \mathcal{D} = 2^{48}.$$

The time complexity of the attack is composed of $\mathcal{T}_E = 2^{93.5}$ equivalent encryptions and $\mathcal{T}_M = (2^{86} + 2^{87})C_A \approx 2^{94}$.

D.4 34-round attack against SIMON 64/128

The path of the attack is given in the GitHub repository.

As all state-test equations imply almost linear equations, the data and time complexities of this attack are:

$$\mathcal{M} = 2^{109} \quad \mathcal{D} = 2^{60}.$$

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
34	3	4	23	4	61.50	48	48	1	2	0	0	4

The time complexity of the attack is composed of $\mathcal{T}_E = 2^{121.5}$ equivalent encryptions and $\mathcal{T}_M = 2^{110}2^2 \cdot 109 \approx 2^{118.76}$.

For the same reason explained in the attack against 28 rounds of SIMON 48/96, we expect to find better Differential MITM attack by taking a distinguisher on fewer rounds.

D.5 35-round attack against SIMON 64/128

The path of the attack is given in the GitHub repository. The parameters of the attack are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
35	3	4	23	5	61.50	43	42	5	8	0	2	2

As all state-test implies almost linear equations, the complexities of this attack are:

$$\mathcal{M} = 2^{110} \quad \mathcal{D} = 2^{62}$$

The time complexity of the attack is composed of $\mathcal{T}_E = 2^{123.5}$ equivalent encryptions and $\mathcal{T}_M = 2^{111}2^2C_A \approx 2^{118.8}$.

For the same reason as explain in the attack against 28 rounds of SIMON48/96, we expect better to found better Differential Meet in the middle attacks by taking a distinguisher on fewer rounds.

D.6 Attack on 36 rounds of SIMON 64/128

We have been able to improve by 3 rounds the best-known attacks on this variant. For this, we used the 23-round differential distinguisher presented in [23]:

$$(\texttt{8800 0000}, \texttt{0200 0000}) \xrightarrow[23r]{2^{-61.5}} (\texttt{0200 0000}, \texttt{8800 0000}).$$

The path of the attack is given in the GitHub repository. The parameters of the attack are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
36	3	5	23	5	61.5	48	48	13	10	2	0	2

As all state-test equations imply almost linear equations, the data and memory complexities of this attack are:

$$\mathcal{M} = 2^{122} \quad \mathcal{D} = 2^{64}.$$

The time complexity of the attack is composed of $\mathcal{T}_E = 2^{125.7}$ equivalent encryptions and $\mathcal{T}_M \approx 2^{122} + 2^{123}2^2 C_A \approx 2^{132.9}$, leading to a time complexity over than brute force.
Here, if we use the data reduction technique with $x = 1$, the bottleneck becomes the time complexity of the upper key-recovery part.
And the time complexity would become:

$$\mathcal{T} = (10/36) \cdot 2^{125.5+2x} + (10/36) \cdot 2^{124.5} + 2^{125.5} \approx 2^{126.5}.$$

And the final data and memory complexities of the attack are:

$$\mathcal{M} = 2^{123} \quad \mathcal{D} = 2^{63}.$$

The time complexity of the attack is $\mathcal{T}_E + \mathcal{T}_M \approx 2^{132.9}$ equivalent encryptions.

D.7 56-round attack against SIMON128/256

The attack against 56 rounds of SIMON128/256 harnesses the following differential distinguisher presented in [23]:

$$(0200\ 0000\ 0000\ 0000, 8880\ 0000\ 0000\ 0000) \xrightarrow[41r]{2^{-113}} (8880\ 0000\ 0000\ 0000, 0200\ 0000\ 0000\ 0000).$$

The path of the attack is given in the GitHub repository. The parameters of the attack are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
56	3	6	41	6	113	99	102	27	25	7	8	8

In the upper part, 17 state-test on 23 are almost linear. For the lower part, 22 out of 25 are almost linear. The resulting complexities are:

$$\mathcal{M} = 2^{254} \quad \mathcal{D} = 2^{128}.$$

And the time complexity of the attack is composed of $\mathcal{T}_E = 2^{248.6}$ equivalent encryptions and $\mathcal{T}_M = 2^{255} \cdot 254 \approx 2^{263}$, that becomes the bottleneck in time.

This attack was found without considering the non-free memory access model. In the non-free access setup we can find another attack with a worse $\mathcal{T}_E$ complexity but an overall time complexity better than brute force. The parameters of this attack are :

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
56	3	6	41	6	113	94	101	27	21	26	13	4

In the upper part, 26 state-test on 27 are almost linear. For the lower part, 13 out of 21 are almost linear. The resulting complexities are:

$$\mathcal{T}_E = 2^{250} \quad \mathcal{T}_M = 2^{251} \quad \mathcal{M} = 2^{523} \quad \mathcal{D} = 2^{128}.$$

The resulting time complexity is $2^{251.4}$ which is better than brute force.

For the reasons explained in Section 4, we think the key recovery of these attacks are suboptimal and better differential MITM attack can be found with this distinguisher.

E Attacks against Simeck n/2n

In this section, we present the found attacks on different variants of Simeck found by our tool.

E.1 Description of Simeck

Simeck $n/2n$ is a family of Feistel block ciphers, derived from Simon, presented in [32]. It operates on blocks of size $n \in \{32, 64, 128\}$. The round function is similar to Simon, only the value of the circular shift changes: $f(x) = ((x \lll 0) \wedge (x \lll 5)) \oplus (x \lll 1)$.
Unlike Simon, the key-schedule of Simeck is non-linear. The master key is separated into four parts $K = t_2|t_1|t_0|k_0$ and the round keys are generated following the framework of Fig. 13

E.2 Attacks against Simeck

Attack on 24 rounds of Simeck 32/64

As Simeck's design is very close to Simon's design, the attacks have been found with the same tool, described in sec. 4. The major difference comes from the key-schedule, as it is not linear in Simeck, we cannot sieve the excessive key information during the match. Covering more than k bits of information about the key is now not optimal.

The differential distinguisher used to mount our attack against Simeck 32/64 is presented in [18] and is as follows:

$$(\texttt{000000}, \texttt{000001}) \xrightarrow[14r]{2^{-30.06}} (\texttt{000001}, \texttt{000000}).$$

The full path of the attack is given in the GitHub repository. The parameters of the attack are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
24	2	4	14	4	30.06	17	16	1	1	0	1	2

By applying the complexity formulas Eq. 9, Eq. 10 and the data reduction technique, the attack data and memory complexities are:

$$\mathcal{D} = 2^{31}$$
$$\mathcal{M} = 2^{47}.$$

The time complexity of the attack is composed of $\mathcal{T}_E = 2^{30.06}(2^{17+1+1} + 2^{16+1} + 2^{17+1+16+1+32-4-30}) = 2^{63}$ equivalent encryptions and $\mathcal{T}_M = 2^{49.32}$ memory accesses to a hash table of size 2^{47}, that can be approximated by $2^{5.5}$, which is negligible in comparison.

Recover the full key With this framework, the attacker recovers $k_{in} + k_{out} + s_{in} + 32 = 64$ bits of information about the key in less than brute force, which represent an advantage. However, recovering the full master key from the information of the attack can be a complex work because of the non-linearity of the key-schedule.

33-round attack against Simeck 48/96 The attack against Simeck 48/96 exploit the 23 rounds differential distinguisher:

$$(\texttt{000000}, \texttt{000001}) \xrightarrow[23r]{2^{-47.27}} (\texttt{000001}, \texttt{000000}).$$

presented in [18].

The full path of the attack is given in the GitHub repository. The parameters of the attack are:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
33	2	4	23	4	47.27	24	23	1	1	0	0	1

The data and memory complexities of this attack are:

$$\mathcal{M} = 2^{71} \quad \mathcal{D} = 2^{48}.$$

And the time complexity of the attack is composed of $\mathcal{T}_E = 2^{95.27}$ equivalent encryptions and $\mathcal{T}_M = 2^{72}$ memory accesses to a hash table of size 2^{71}, which is also negligible in comparison.

41-round attack against Simeck 64/128

The path of the attack is given in the GitHub repository. The parameters of the attack are:

The complexities of this attack are:

$$\mathcal{M} = 2^{92} \quad \mathcal{D} = 2^{63}.$$

And the time complexity of the attack is composed of $\mathcal{T}_E = 2^{126.21}$ equivalent encryptions and $\mathcal{T}_M = 2^{93}$ memory accesses to a hash table of size 2^{92}, which is negligible in comparison.

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	F
41	2	4	31	4	62.21	35	32	0	0	0	2	3

F Attack on 15 rounds of CLEFIA-256

The method used for the attack on 16 rounds to find the master key is also applicable for the attacks that reach fewer rounds. Indeed, if we keep the same differential path as in 11 but with only one round in the lower part then, we can recover the whole master key for CLEFIA-256 reduced to 15 rounds with a better time complexity as the previous best-known attacks. The parameters of the attack are as follows:

r	r_S	r_U	r_D	r_L	p	k_{in}	k_{out}	s_{in}	s_{out}	p_{in}	p_{out}	Δ_{in}	Δ_{out}	F	a
15	1	3	10	1	104	40	96	40	0	0	0	8	14.64	128	2^6

Following the same procedure as in the 16-round attack, we get $2^{172.64}$ candidates for 168 bits of information on the key and 8 bits of state-test, with a time complexity of:

$$\begin{aligned}\mathcal{T}_E &= 2^6 2^{104-8} \cdot (2^8 2^{40+40} + 2^{14.64} 2^{96} 2^{-24} + 2^{88+86.64-64-40}) \\ &= 2^{190} + 2^{188.64} + 2^{172.64} \approx 2^{190} \text{ encryptions.}\end{aligned}$$

When considering the cost of memory accesses with a hash table of size 2^{87}:

$$\mathcal{T}_M = 2^{190} C_A + 2^{188} C_W \text{ memory accesses,}$$

for a total time complexity of $\mathcal{T} = \mathcal{T}_E + \mathcal{T}_M$. In order to verify if this complexity is smaller than exhaustive search, we consider $\mathcal{T}_M \approx 2^{190.32} 2^{6.4} = 2^{196.72}$ equivalent encryptions, which is lower than exhaustive search.
We then keep the candidate keys that appear at least $t = 60$ times thus now, we have $2^{143.68}$ candidates. Then we guess the 32 bits of kR_2 and for all the $2^{143.68+32} = 2^{175.68}$ candidates, we verify that at least 36 equations of X_5 are consistent so that only one candidate remain.

As before, we have recovered 64 bits of L and we guess the 192 remaining bits of L and R to revert the key schedule and recover the master key KL and KR. We can check if the master key is consistent with the subkeys we guessed, which lead to 2^{44} candidates that we can test with another pair of data.
The time, data, and memory complexities are as follows:

$$\mathcal{T} = 2^{192},\ \mathcal{D} = 2^{116}\ , \mathcal{M} = 2^{168}.$$