IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2016, No. 2, pp. 307–337. DOI:10.13154/tosc.v2016.i2.307-337

Meet-in-the-Middle Attacks on Classes of Contracting and Expanding Feistel Constructions

Jian Guo[1], Jérémy Jean[1,2], Ivica Nikolić[1] and Yu Sasaki[1,3]

[1] Nanyang Technological University, Singapore, Singapore
inikolic@ntu.edu.sg
[2] Agence nationale de la sécurité des systèmes d'information (ANSSI), Paris, France
[3] NTT Secure Platform Laboratories, Tokyo, Japan
sasaki.yu@lab.ntt.co.jp

Abstract. We show generic attacks on unbalanced Feistel ciphers based on the meet-in-the-middle technique. We analyze two general classes of unbalanced Feistel structures, namely contracting Feistels and expanding Feistels. In both of the cases, we consider the practical scenario where the round functions are keyless and known to the adversary. In the case of contracting Feistels with 4 branches, we show attacks on 16 rounds when the key length k (in bits) is as large as the block length n (in bits), and up to 24 rounds when $k = 2n$. In the case of expanding Feistels, we consider two scenarios: one, where different nonlinear functions without particular structures are used in the round function, and a more practical one, where a single nonlinear is used but different linear functions are introduced in the state update. In the former case, we propose generic attacks on 13 rounds when $k = n$, and up to 21 rounds when $k = 2n$. In the latter case, 16 rounds can be attacked for $k = n$, and 24 rounds for $k = 2n$.

Keywords: Unbalanced Feistel · Generic Attack · Key Recovery · MITM

1 Introduction

Block ciphers based on the Feistel construction are one of the most popular and well studied symmetric-key primitives. The former block cipher standard DES [Cop94] as well as a number of other ciphers are all based on the Feistel construction. Most of the Feistel ciphers are in fact balanced Feistels. In such constructions, the state of the cipher is divided into two parts of equal size, and it goes through several rounds, each with a round function that updates one part of the state.

The topic of our research, however, are ciphers that belong to the class of unbalanced Feistels. In this type of construction, the state is divided into more than two parts – usually four or eight equal parts, called branches. With such a division, there is a wide variety of round functions that can be considered. In particular, the size of the input and the output of the non-linear round update can vary. Furthermore, the round may end not only with a simple rounds twist (or cyclic shift of the parts), but also with an arbitrary permutation. These types of ciphers have been introduced by Nyberg [Nyb96] and are called generalized Feistel networks. In the networks, the size of the non-linear functions can be smaller than that in balanced Feistels, which is useful to design lightweight ciphers.

Compared to balanced Feistels, the family of unbalanced Feistels offers a wider range of designs, but the increased number of possible constructions requires larger effort for cryptanalysis. In particular, no generic attack on a significantly large number of rounds is

Received: 2016-09-01, Accepted: 2016-11-01, Published: 2017-02-03

applicable to all of the unbalanced Feistels. Thus, one usually divides this large family into several classes, each of which allows some type of a generic analysis. Among the classes, two of the most popular unbalanced Feistels are contracting and expanding Feistels, which are also called source-heavy and target-heavy Feistels [SK96, YI13].

A contracting Feistel uses a round function that takes all but one branch as inputs, while it outputs a single branch. On the other hand, expanding Feistels rely on a round function that takes as input a single branch and produces output of size equivalent to all but one branch. Sometimes, these ciphers are defined more generally, i.e. a contracting Feistel has round function with range smaller than domain, while in the expanding it is the opposite. Under such a wide definition, a number of Feistel schemes are considered contracting, e.g. RC2 [Riv98], SPEED [Zhe97], SMS4 [Lu08], SHACAL [HHN00], or expanding e.g. MARS [BCD+98]. The wider definition, however, does not allow to generalize the attacks on these Feistels. Thus, similarly to the previously published analysis, we focus on the classical definition.

In most of the current analysis of contracting and expanding Feistels, the round functions follow the Luby-Rackoff paradigm, i.e. the functions are chosen uniformly at random from a family of all functions (with such a domain and range). In a series of Asiacrypt papers, Patarin et al. [PNB06, PNB07, VNP10] have provided generic attacks that penetrate most rounds in both contracting and expanding Feistels. Yanagihara and Iwata [YI13, YI14] have found the required number of rounds to achieve full diffusion in the states of such ciphers. In addition, they have provided saturation and impossible differential characteristics that can be used in attacks, as well as upper bounds on the best differential and linear characteristics. Naor and Reingold [NR99] have studied the security of contracting Feistel schemes with pairwise independent permutations and have shown lower bounds for the security of such constructions. Lucks [Luc96] has given security results on contracting Feistel built upon hash functions. Yun et al. [YPL11] proved the birthday bound security for $2d-1$ rounds on contracting Feistel. Suzaki and Minematsu [SM10] evaluated generalized Feistel network with block shuffle structure with some particular attack approaches e.g. integral or impossible differential attacks. However, the contracting and expanding Feistel networks are not covered.

Our Contributions. We show generic chosen-plaintext attacks on contracting and expanding Feistels. However, instead of assuming the general Luby-Rackoff type, we analyze Feistels based on more practical round updates. A round of these ciphers is composed of a keyless non-linear round function, surrounded by an arbitrary number of subkey XORs. In such a framework, we show new generic attacks that penetrate a large number of rounds.

Our analysis is based on the meet-in-the-middle technique as proposed by Demirci and Selçuk [DS08] in the attacks on AES. We use Dunkelman et al. [DKS10a] improvements to this technique and follow their style of presentation. This relatively new cryptanalytic technique seems promising: first it has been applied to the SPN construction adopted in the AES by Dunkelman et al. [DKS10a], then it has been improved in [DFJ13, LJW14, LJ15], and more recently it has been used for analysis on several different concrete primitives: e.g., PRINCE in [DP15], TWINE in [BDP15], CLEFIA in [LJWD15], CAMELLIA in [LJWD15, DLJW15], and also on generic balanced Feistel constructions in [GJNS14].

We examine three classes of unbalanced Feistels: one class of contracting Feistels, and two separate classes of expanding Feistels. The success of our analysis depends on the number of branches d and on the key bit-lengths k of the n-bit Feistels: the higher d and k, the more rounds we can attack. This is because a bigger key and a bigger state size will allow to guess more state values (without reaching the time complexity bound limited by exhaustive key search and the data complexity bound limited by the code book), thus we can add more rounds in the middle. For the class of contracting Feistels, we show a meet-in-the-middle attack on $5d-4$ rounds. In the general case of expanding Feistels with arbitrary keyless round functions (we denote them as expanding Feistel-F), we show an attack on $4d-3$ rounds. We separately analyze more specific expanding Feistels, with linear dependency of the output branches in the round function (we denote it as expanding Feistel-FL). Such round functions are more lightweight and are already in use, for instance in ARIRANG [CHK+08]. For this type of Feistels, we give attacks on $5d-4$ rounds.

The numbers of attacked rounds given in the previous paragraph are valid for the case where the key length matches the block length, i.e. $k=n$. When the key is larger, our meet-in-the-middle attacks reach more rounds. The exact number of attacked rounds is $(3+2\frac{k}{n})d-4$ in the case of contracting and expanding Feistels-FL, and $(2+2\frac{k}{n})d-3$ rounds in the case of expanding Feistel-F. To get a sense of concrete values implied by these numbers, consider the case of a lightweight block cipher based on unbalanced Feistel with 64-bit state, 128-bit key and 16 branches, i.e. $(n,k,d)=(64,128,16)$. If the cipher is an expanding Feistel-F within our analyzed class, then our attack penetrates 93 rounds. On the other hand, if the cipher is a contracting Feistel, it reaches 108 rounds.

We stress out that our meet-in-the-middle attacks result in a full subkey recovery (and not only distinguishers). The precise approach used to achieve the recovery is given for example in [GJNS14], and due to space restrictions we omitted it from this paper. Our results are summarized in Table 1.

We also stress out that this paper is not a straightforward application of existing techniques to other constructions that have not been evaluated yet. For example, we introduce the classic meet-in-the-middle attack to enumerate valid differential characteristics, is a particular approach for expanding Feistels, i.e. cannot be seen in the previous work on balanced Feistels. All of our three attacks have been experimentally verified of small scale variants[1].

Table 1: The number of attacked rounds for unbalanced Feistel constructions. All of our attacks given in the table are more efficient than exhaustive search by a factor of $2^{n/2d}$. They are chosen-plaintext attacks, and recover all the subkeys. In comparison, the attacks from [PNB06, PNB07, VNP10] are applicable to wider classes of expanding and contracting Feistels but only yield distinguishers.($\dagger : r \geq 2$)

Type	**Bit Length of Key k** (d **branches**)	**#rounds**	
		Patarin et al.	**Ours**
Contracting Feistel (Section 3)	n	$2d-1$	$5d-4$
	$2n$	$2d-1$	$7d-4$
	$n+\frac{rn}{d}$	$2d-1$	$5d-4+2r$
Expanding Feistel-F (Section 4)	n	$3d-1$	$4d-3$
	$n+\frac{n}{d}$	$3d-1$	$4d$
	$2n$	$3d-1$	$6d-3$
	$n+\frac{rn}{d}$	$3d-1$	$4d-3+2r$ †
Expanding Feistel-FL (Section 5)	n	$3d-1$	$5d-4$
	$2n$	$3d-1$	$7d-4$
	$n+\frac{rn}{d}$	$3d-1$	$5d-4+2r$

[1]The source code of the attacks can be found at http://www1.spms.ntu.edu.sg/~syllab/attacks/attacks.zip

2 The Meet-in-the-Middle Attack

Our analysis is in line with the recent meet-in-the-middle (MITM) attacks on `AES` [DKS10b, DFJ13, LJW14], a technique pioneered by Demirci and Selçuk [DS08] and improved by Dunkelman et al [DKS10b]. Despite having the same name as the traditional MITM attacks (the attacks on double `DES` [DH77] and on a number of hash functions, e.g. [AS09, GLRW10]), this approach follows a different and a more complex strategy.[2] Further, we give a brief overview of the technique as seen by Dunkelman et al.

2.1 Overview of the Technique

Demirci-Selçuk MITM attack is a method for analysis of iterative block ciphers. It can be seen as a *technique that converts a differential on R rounds into a subkey recovery on $R' + R + R''$ rounds.* A trivial way of extending a differential to more rounds usually does not provide a sufficiently strong filter (filter=whether $R' + R + R''$ round data pairs follow the R round differential). On the other hand, Demirci-Selçuk technique leads to a very strong filter.

Not every differential can be used in Demirci-Selçuk MITM attack. Let us take a brief look at the details.

1. *The differential has a low number of differential characteristics.* An R-round differential $\Delta \xrightarrow{R} \nabla$ is specified only with two differences: an input difference Δ and an output difference ∇. If besides Δ and ∇, we specify the difference after each transformation in the R rounds, then we obtain a differential characteristic. The number of all possible characteristics (that have the same input and output differences as the differential) should be low – usually much lower than 2^k, where k is the length of the master key in bits.

2. *The characteristics provide partial state values.* By definition, a characteristic specifies both the input and the output differences for each transformation in the R rounds. Given these two differences $(\Delta_{in}, \nabla_{out})$ for some non-linear transformation $F(x)$ (used in the R rounds), we can actually find the input. That is, we can solve $F(x \oplus \Delta_{in}) \oplus F(x) = \nabla_{out}$ and find either the entire value of x or at least some bits of x (the equation is called differential equation and usually it is solved with a look-up table). Therefore, each characteristic allows the recover partial state values – the R rounds may have many non-linear transformations and each will allow to recover some bits of the state.

3. *The partial values are sufficient to compute a property called b-δ-sequence.*[3] Let $F^R(x)$ be the R rounds of the cipher covered by the differential (x is the input state; for now we ignore the subkeys). If a characteristic of the differential provides sufficient amount of state values, then we can compute a sequence of 2^b differences (called b-δ-sequence) defined as $F^R(v) \oplus F^R(v \oplus 1), F^R(v) \oplus F^R(v \oplus 2), \ldots, F^R(v) \oplus F^R(v \oplus 2^b)$. Each value of this sequence defines how a certain input difference propagates into an output difference after R rounds. For instance, the first value is propagation of the difference 1 to $F^R(v) \oplus F^R(v \oplus 1)$. To find the output difference, we have to go through all the transformations of the R rounds. The propagation through the linear transformations can be computed trivially. The non-linear transformation may introduce branching. However, the partial state values recovered at Step 2 may be sufficient to uniquely determine the propagation through the non-linear

[2]To avoid confusion, it might be in the best interest to use another name for the approach.

[3]The term "n-δ-set" was defined in [GSW+14] as δ-set with n active bytes. The term "b-delta-set" was defined in [GJNS14] as δ-set with b active bits, and "b-δ-sequences" was introduced as the corresponding difference sequences.

transformations (for instance, if in Step 2 we have found the value of x for the non-linear $F(x)$, then obviously, for any δ we can compute $F(x \oplus \delta) \oplus F(x)$). In those cases, we can compute the entire sequence. It is critical to understand that each characteristic will result in a different b-δ-sequence (because each characteristic provides different partial state values that are used to construct the b-δ-sequence).

The above analysis indicates that the differential $\Delta \xrightarrow{R} \nabla$ may be described as a set of different b-δ-sequences with cardinality less than 2^k. This set is stored in a look-up table T. The whole procedure consists in the **offline** phase of the attack (because it can be executed without querying the cipher).

To convert the R-round differential into a subkey recovery on $R' + R + R''$ rounds, we use the standard approach which basically relies on guessing parts of the subkeys used in the first R' and the last R'' rounds. *The wrong subkey guesses will be entirely discarded as we rely on a very strong filter – the b-δ-sequence.* This is the **online** phase of the attack.

First, we construct two differentials: $\overline{\Delta} \xrightarrow{R'} \Delta$ for the first R' rounds, and $\nabla \xrightarrow{R''} \overline{\nabla}$ for the last R'' rounds (hence all $R' + R + R''$ rounds are covered with $\overline{\Delta} \xrightarrow{R'} \Delta \xrightarrow{R} \nabla \xrightarrow{R''} \overline{\nabla}$). Then, we query plaintext pairs $(p_0, p_0 \oplus \overline{\Delta})$ and check if corresponding ciphertext pairs (c_0, c_0^*) have the difference $\overline{\nabla}$. For each good pair, we guess the subkeys used in the first R' rounds.[4] According to the guess, we compute the value v of the state after R' rounds. Then, we compute plaintexts $p_1, p_2, \ldots, p_{2^b}$ that correspond to the values $v \oplus 1, v \oplus 2, \ldots, v \oplus 2^b$. For instance, p_1 can be computed by R'-round decryption (with the guessed subkeys) of the state $v \oplus 1$. We query all p_i and obtain corresponding c_i. Finally, we guess the subkeys used in the last R'' rounds. For each guess of these subkeys, we do a partial R'' round decryption of all c_i, which results in state values s_i. Namely, for each guess of subkeys, we obtain the following 2^b data transitions for $R' + R + R''$ rounds.

$$\begin{aligned} p_0 &\xrightarrow{R'} v \xrightarrow{R} s_0 \xrightarrow{R''} c_0, \\ p_1 &\xrightarrow{R'} v \oplus 1 \xrightarrow{R} s_1 \xrightarrow{R''} c_1, \\ p_2 &\xrightarrow{R'} v \oplus 2 \xrightarrow{R} s_2 \xrightarrow{R''} c_2, \\ &\cdots \\ p_{2^b} &\xrightarrow{R'} v \oplus 2^b \xrightarrow{R} s_{2^b} \xrightarrow{R''} c_{2^b}. \end{aligned}$$

Then, we create b-δ-sequence based on s_i, i.e. we compute the sequence of differences $s_0 \oplus s_1, s_0 \oplus s_2, \ldots, s_0 \oplus s_{2^b}$ and check if the sequence is in the table T. *Only correctly guessed subkeys used in R' and R'' can result in a match.* Therefore, we can entirely recover these subkeys.

Above, we have defined the b-δ-sequence as $F^R(v) \oplus F^R(v \oplus 1), F^R(v) \oplus F^R(v \oplus 2), \ldots, F^R(v) \oplus F^R(v \oplus 2^b)$, i.e. we have added the values $1, 2, \ldots, 2^b$ to the least significant bits of the input v. This is an oversimplification – in an actual analysis the values are added at the same positions as Δ (e.g. if the input difference Δ of the differential is added to the third byte of the state, then $1, 2, \ldots, 2^b$ are added to this byte as well). Furthermore, we have assumed that b-δ-sequence is defined in all bits of the output $F^R(v) \oplus F^R(v \oplus 1), \ldots$ In fact, the sequence can be defined only in t bits. Even when $t = 1$, the sequence will still provide a strong filter as long as the value of b is large (the filter is on $t \cdot 2^b$ bits).

2.2 Presentation of the Attacks

We will present our attacks by the following unified approach:

1. Offline phase (the construction of the table T)

[4]Essentially, we guess the subkeys in a way such that after the first R' rounds the state difference is Δ.

- **Differential**: we give the differential and show that its number of characteristics is smaller than 2^k.
- **The b-δ-sequence**: we show that for each characteristic, we can deduce partial state values, and based on these state values, we can compute the b-δ-sequence.
- (Optional) **Truncated differential**: we show that instead of a single differential, we can consider several differentials that belong to a particular truncated differential.

2. Online phase

- **Key recovery:** we define the number of added rounds (either before or after the differential). We show how to use structures to reduce the amount of plaintexts, and how to achieve the filtering of the ciphertext pairs.
- **Complexity analysis**: we give the estimates for the offline and online time, memory, and data complexities.

Contrary to several recent papers, e.g. [DF16, FWG+16, SHW+14], where automated tools have been used to find differential, impossible or integral characteristics, we do not exactly follow this approach in our work: we did implement the search for contracting Feistels, but not for expanding Feistels. The main reason is that in the expanding case, finding a long differential characteristic suitable for the meet-in-the-middle attack can be done by looking at sparse characteristics, which can usually be done easily by hand for (generalized) Feistel networks. Consequently, optimality of the resulting attacks cannot be proven without more in-depth analysis of differential characteristics, but it can probably be reached using automated tools. Finally, we have arbitrarily decided to turn the attacks into chosen-plaintext by extending (when possible) the differential characteristics towards the plaintext side rather the ciphertext side, but the attacks would reach the same complexities should one decide the other way.

3 Contracting Feistels

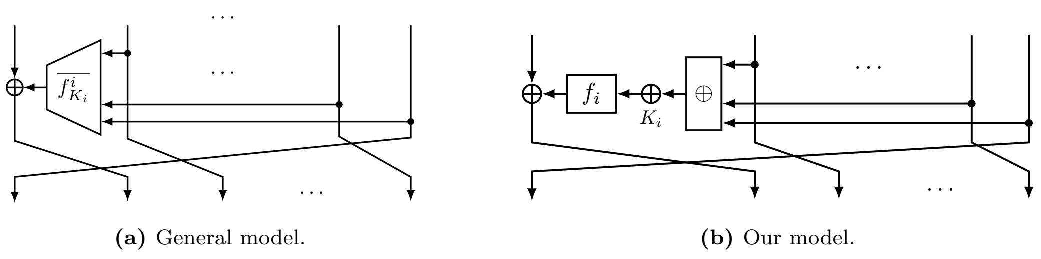
Contracting Feistels belong to the class of unbalanced Feistels. Their number of branches d can vary, but satisfies $d > 2$. In a single round, only one branch is updated (refer to Figure 1a). The Feistels are called contracting because the round update is achieved with a non-linear function that takes as inputs the values of $d-1$ branches (and the subkey), while it outputs a single branch.

Patarin et al. have analyzed in [PNB06] the most generic contracting Feistels as specified in Figure 1a. When $d > 3$, they have shown attacks on $2d-1$ rounds of such ciphers. Yanagihara and Iwata have investigated in [YI13] the resistance of similar constructions against a variety of attacks for specific values of d. Their results come in a form of attacks as well as security upper bounds. For instance, for $d = 4$, they show that there is an impossible differential on 6 rounds (attack), but there is no high-probability linear characteristic on 14 rounds (upper bound).

We focus on contracting Feistels with non-linear round functions defined as

$$\overline{f_i}(x_1, \ldots, x_{d-1}, K_i) = f_i(x_1 \oplus \ldots \oplus x_{d-1} \oplus K_i),$$

where f_i is an arbitrary *public* non-linear function, with one branch input and one branch output (refer to Figure 1b). We stress out that this type of functions are used, for instance, in the Chinese block cipher standard `SMS4` [Lu08]. Our MITM-based approach, in comparison to the distinguishers provided in [PNB06], yields chosen-plaintext key-recovery attacks on larger number of rounds. The precise formula for this number depends on the bit length of key: when the bit lengths of key and the state are equal, then we penetrate



(a) General model. **(b)** Our model.

Figure 1: Models of contracting Feistel with d branches.

through $5d-4$ rounds, while when the key is twice as large as the state, then $7d-4$ rounds. In fact, our generic formula is $(3+2\frac{k}{n})d-4$ rounds for contracting Feistel, with k-bit key (k is a multiple of $\frac{n}{d}$ and is greater than n), n-bit state, and an even number of branches $d \geq 4$. When $k = n + \frac{rn}{d}$, this becomes $5d - 4 + 2r$ rounds as shown in Table 1.[5]

Note that the analysis given in the sequel can be generalized even further: the round functions can be different depending on the round, the subkeys can be XORed after the application of the round functions, etc. The analysis of all these variants is similar to the original, thus is omitted. As an warm-up, we start with the case of a 4-branch contracting Feistels when $k = n$, and from there develop the analysis for the cases of larger number of branches and larger keys.

3.1 Case $k = n$ and $d = 4$

In this section, we construct a 16-round key-recovery attack for a contracting Feistel structure with $d = 4$ branches, when key and state share the same length in bits.

Differential. The 13-round differential used in the distinguisher is depicted in Figure 7 (in Appendix). For a given Δ_0 and Δ_1, its input difference is $(\Delta_0, 0, \Delta_0, \Delta_0)$ and its output difference is $(0, \Delta_1, \Delta_1, \Delta_1)$. We show further that this differential contains at most $2^{\frac{3n}{4}}$ distinct differential characteristics, which verifies the requirement discussed earlier (less than 2^k).

First, we note that in the first three rounds, the propagation is fully linear since the input differences to the functions f_1, f_2 and f_3 are zero. After three rounds, $(\Delta_0, 0, \Delta_0, \Delta_0)$ becomes $(0, \Delta_0, \Delta_0, \Delta_0)$ with probability one. Similarly, $(0, \Delta_1, \Delta_1, \Delta_1)$ is transformed to $(\Delta_1, 0, \Delta_1, \Delta_1)$ after three backward rounds.

Next, we observe that the slow diffusion in the d branches allows to linearly deduce differences further in many branches. Not all of them can be determined since non-linearity is introduced by the round functions. However, by enumerating all the output differences Δ_X of f_4, Δ_Y of f_5 and Δ_Z of f_6, it is possible to determine all the differences in all the branches. Note that we do not need to guess the output difference of f_7 as it depends only on Δ_0 and Δ_1. This holds because it is halfway between the input and the output difference.

Example. The output difference of f_{10} can be uniquely computed for each Δ_X, Δ_Y, and Δ_Z (and given Δ_0 and Δ_1.) From the output of round 10, the input difference of f_{10} is Δ_1. The leftmost input to round 10 is a copy (no update) of the second left branch of the input to round 7. So, we know the output difference of f_{10} is $\Delta_0 \oplus \Delta_Z$.

[5]For odd d, the construction from Figure 1b allows trivial differential distinguishers (with the same plaintext difference in all d branches), thus has no scientific interest.

Consequently, we enumerate all the possible differential characteristics of the differential $(\Delta_0, 0, \Delta_0, \Delta_0) \to (0, \Delta_1, \Delta_1, \Delta_1)$ by considering all the possible nonzero values for $(\Delta_X, \Delta_Y, \Delta_Z)$. Since this tuple can assume $(2^{\frac{n}{4}} - 1)^3$ values, we conclude that about $2^{\frac{3n}{4}}$ differential characteristics compose the differential.

Evaluation of the b-δ-sequence. Assume we have a differential characteristic with fixed internal differences in all of the 13 rounds (as mentioned previously, each such characteristic is defined with the three output differences $\Delta_X, \Delta_Y, \Delta_Z$). The functions $f_4, f_5, \ldots, f_{10}$ are active and we know their input and output differences. As a result, by solving a differential equation for each such function, we can determine the input and the output paired values, of these seven functions.

We now determine the b-δ-sequence based on the found values. If (v, v') denotes the initial pair of inputs with difference $(\Delta_0, 0, \Delta_0, \Delta_0)$, let $(v, v_\Delta), \Delta = 1, 2, \ldots, 2^b$, be the pair with the difference $(\Delta, 0, \Delta, \Delta)$. In the first three rounds, none of f_1, f_2 and f_3 is active, and the difference becomes $(0, \Delta, \Delta, \Delta)$. The input difference to f_4 is Δ. Since we already have the input value of f_4 (that we found previously when analyzing v and v'), it means we can compute the input value of f_4 that correspond to v_Δ. For instance, if the found value is a, then the new value would be $a \oplus \Delta$. From the input value, we can compute the output value of f_4 simply as $f_4(a \oplus \Delta)$, and thus we can obtain the output difference of f_4 as $f_4(a) \oplus f_4(a \oplus \Delta)$. By following the same procedure, we can compute the output difference of the remaining $f_5, \ldots, f_{10}$. Consequently, we can have the internal state difference before the last three rounds. In particular, we have the state difference of the second branch. This difference, after the remaining three rounds, becomes the state difference of the first branch. Therefore, the b-δ-sequence can be computed based on the first branch of the output and is determined uniquely without the knowledge of any of the subkeys used in these 13 rounds.

Truncated Differentials. A possible tradeoff consists in alleviating the fixed input and output differences of the differential, and in allowing $2^x < 2^{\frac{n}{4}}$ possible input differences Δ_0, and $2^y < 2^{\frac{n}{4}}$ possible output differences Δ_1. Although (Δ_0, Δ_1) can take more than one value, we still require the differential to be of the form: $(\Delta_0, 0, \Delta_0, \Delta_0) \to (0, \Delta_1, \Delta_1, \Delta_1)$. In that case, the complexities (in particular, the number of differential characteristics that compose the truncated differential) increase by a factor 2^{x+y} and become $2^{\frac{3n}{4}+x+y}$. In the subsequent analysis, we use the two additional parameters x and y to find their optimal values as to minimize the overall complexity of the complete attack.

Key Recovery. To launch the 16-round attack, we prepend three rounds to the differential as shown in Figure 2. As there are three round functions with nonzero input differences, we introduce three variables, namely δ_1, δ_2 and δ_3, that correspond to the output differences of f_1, f_2 and f_3, respectively.

In order to get plaintext pairs with a valid difference Δ_0 in the second branch, we use a structure of plaintexts. Namely, we consider all the $2^{\frac{3n}{4}}$ possible values in all the branches except the second branch where we take only 2^x values. Note that pairs with two plaintexts from this structure already conform to the truncated differential discussed earlier. We reach a data complexity of $2^{\frac{3n}{4}+x}$ chosen plaintexts, and we get up to $2^{\frac{3n}{2}+2x}$ pairs with one valid difference Δ_0 in the second branch.

Among all the pairs, we expect a fraction $2^y/2^n = 2^{-n+y}$ to verify one of the possible output differences according to the 2^y values we allowed for Δ_1. To efficiently filter the remaining $2^{\frac{n}{2}+2x+y}$ pairs, we store each plaintext/ciphertext value in a hash table sorted by the 3-element value $(c_1, c_2 \oplus c_3, c_2 \oplus c_4)$ where the ciphertext equals (c_1, c_2, c_3, c_4). This way, two ciphertexts (c_1, c_2, c_3, c_4) and (c'_1, c'_2, c'_3, c'_4) are stored under the same key if and

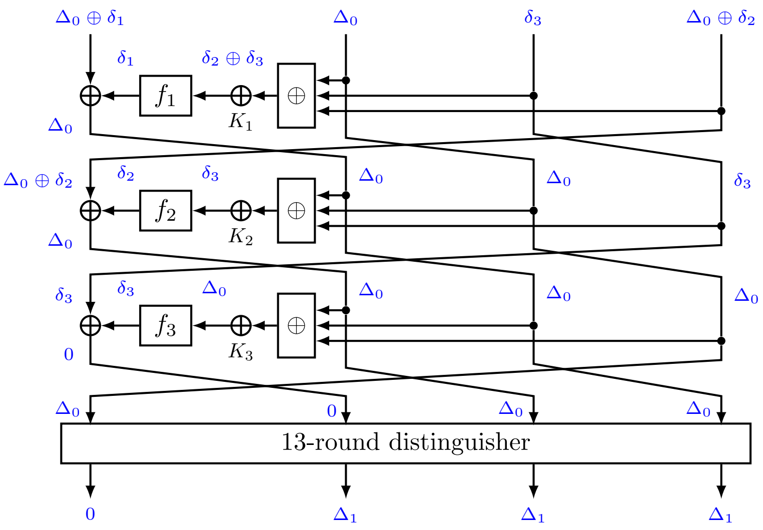


Figure 2: Key recovery for 16 rounds (case $k = n$). The attack makes use of the 13-round distinguisher from Figure 7.

only if $c_1 \oplus c_1' = 0$, $c_2 \oplus c_2' \oplus c_3 \oplus c_3' = 0$ and $c_2 \oplus c_2' \oplus c_4 \oplus c_4' = 0$, which subsequently lead to the corresponding Δ_1 to be $\Delta_1 = c_2 \oplus c_2'$.

Among the $2^{\frac{n}{2}+2x+y}$ remaining pairs, we need one pair that verifies the 13-round differential in the last rounds. We detect it by checking the precomputed table T. The probability that an input pair with known input difference yields one of the 2^x intermediate difference $(\Delta_0, 0, \Delta_0, \Delta_0)$ chosen during the precomputation phase is $2^{-3n/4}$ for each pair. Indeed, when the pair with plaintext difference $(\Delta P_1, \Delta P_2, \Delta P_3, \Delta P_4)$ is selected, we can compute the four differences Δ_0, δ_1, δ_2 and δ_3 as $\Delta_0 = \Delta P_2$, $\delta_1 = \Delta P_1 \oplus \Delta_0$, $\delta_2 = \Delta P_4 \oplus \Delta_0$ and $\delta_3 = \Delta P_3$, and from them, by solving differential equations, deduce the subkeys K_1, K_2 and K_3 used in the first three rounds. Once we have the subkey values, we can easily construct plaintexts that correspond to the b-δ-sequences. For instance, to get the plaintext that corresponds to $\Delta = 1$, we take the value of the state after three rounds that corresponds to the initial plaintext p, XOR the difference $(1, 0, 1, 1)$, go back three rounds (this is possible because we have the values of the subkeys), and obtain the new plaintext p_1. In a similar manner, we can produce the remaining plaintexts $p_2, \ldots, p_{2^b}$ and thus we can obtain the b-δ-sequence, $E(p) \oplus E(p_1), E(p) \oplus E(p_2), \ldots, E(p) \oplus E(p_{2^b})$, where $E(x)$ is the ciphertext for the plaintext x. Note, *we do not need to query the plaintexts* $p_1, \ldots, p_{2^b}$ as they have already been queried previously. Indeed, the initial structure of plaintexts already contains all the possible plaintext that can be obtained from any of the plaintexts by going three forward rounds, adding a difference of the type $(\Delta, 0, \Delta, \Delta)$ and then inverting the state for three rounds. That is, S is composed of plaintexts P_i such that for any i (and any K_1, K_2, K_3), it holds $F^{-3}(F^3(P_i) \oplus (\Delta, 0, \Delta, \Delta)) \in S$.

The probability to pass the first three rounds is $2^{-3n/4}$, and thus we need to collect as many as $2^{3n/4}$ pairs, that is: $2^{\frac{n}{2}+2x+y} \geq 2^{3n/4}$, which implies $8x + 4y \geq n$.

Complexity Analysis. The total data complexity required to conduct the attack amounts to $2^{\frac{3n}{4}+x}$ chosen plaintexts that are required to produce at least $2^{\frac{n}{2}+2x+y}$ pairs for the online phase. Therefore, the number of such pairs processed in the online phase amounts to $2^{\frac{n}{2}+2x+y}$, and each of them requires slightly less than one encryption. Finally, the memory requirements to store the precomputed table is $2^{\frac{3n}{4}+x+y}$ sequences of 2^b elements.

To find optimal values for x and y that minimize the overall complexity, we observe that $8x + 4y \geq n$ must hold in order to collect enough pairs for the online phase, and

$0 \le x + y < n/4$ so that the memory and data complexities stay below 2^n. As a result, for $x = n/8$ and $y = 0$, this attack leads to a data complexity of $2^{\frac{3n}{4}+x} = 2^{\frac{7n}{8}}$ chosen plaintexts, an online phase with time complexity of $2^{\frac{n}{2}+2x+y} = 2^{\frac{3n}{4}}$ operations, and a memory complexity of $2^{\frac{3n}{4}+x+y} = 2^{\frac{7n}{8}}$ of b-δ-sequences.

To find a good value for b, we evaluate the number of false positive. In the online phase on the attack, we perform $2^{\frac{3n}{4}}$ checks in the precomputed table T, which contains $2^{\frac{7n}{8}}$ sequences of 2^b elements of $\frac{n}{4}$ bits each. Consequently, one check in the table yields a false positive with probability $2^{\frac{7n}{8}}/2^{\frac{n}{4}\cdot 2^b} = 2^{\frac{n}{8}(7-2^{b+1})}$. We can ensure that no false positive occurs for all the $2^{\frac{3n}{4}}$ checks, if we select b such that: $\frac{n}{8}(7-2^{b+1}) + \frac{3n}{4} < 0$, that is $b > \log_2(13) - 1$. Hence, by fixing $b = 3$, we control the expected number of false positive while minimizing the size of the elements stored in T.

Finally, we point out that the precise complexity of our attacks obviously has to incorporate the value of b (so far we have omitted). It is easy to notice that the actual time complexity is $2^{\frac{3n}{4}} \cdot 2^b$ (rather than $2^{\frac{3n}{4}}$). We have omitted the value of b from the complexity estimates as: 1) it is a very small factor, 2) it makes the complexity formula bulkier, and 3) the time complexity is given in terms of full encryptions, which overestimate the actual operations that only evaluate a few rounds. For these reasons, we omit the value of b in the further analysis as well.

3.2 Generalization to More Branches

The 16-round attack can be generalized and extended to more rounds when the number of branches is higher. More precisely, when the contracting Feistel has d branches, we can attack $5d - 4$ rounds. To deduce this number, we focus on the two phases of our MITM on 16-round 4-branch Feistel and see how the number of rounds in each phase would change with the increase of the amount of branches.

Differential and Characteristics. For the 4-branch Feistel, the used 13-round differential can be decomposed as $13 = 3 + 7 + 3$, i.e. it has 3 rounds at the beginning and 3 rounds at the end (where the differential is determined uniquely, i.e. the probability is one), and 7 rounds in the middle. The 3 rounds on both sides of the differential, in a d-branch Feistel are equivalent to $d - 1$ rounds with input difference $(\Delta_0, 0, \Delta_0, \ldots, \Delta_0)$ and output difference $(0, \Delta_0, \Delta_0, \ldots, \Delta_0)$. The middle 7 rounds of the differential for the 4-branch Feistel, are equivalent to middle $2d - 1$ rounds for a d-branch Feistel. This comes from the fact that the outputs (in terms of branches) of the $2d - 1$ contracting functions in these rounds will have to satisfy a system of equations on d branches and the whole system can be solved by guessing $d - 1$ outputs (as a result, $d + d - 1 = 2d - 1$). Note, having an additional round in the middle would result in $2d$ outputs, thus it would require d branches to be guessed and would incidentally raise the complexity of the attack to 2^n. Thus, having $2d - 1$ middle rounds is optimal, and therefore the differential for d-branch Feistel is on $d - 1 + d - 1 + 2d - 1 = 4d - 3$ rounds.

The $(4d - 3)$-round differential is uniquely determined in the first and in the last $d - 1$ rounds. However, the differences in the middle $2d - 1$ rounds are not unique and this is where the characteristics that compose the differential differ. Although there are $2d - 1$ active functions in these rounds, as mentioned previously, by guessing $d - 1$ outputs, the remaining d outputs are determined uniquely. Thus, the total number of possible differential characteristics that compose the differential is upper bounded by $2^{(d-1)\cdot\frac{n}{d}} = 2^{\frac{d-1}{d}n}$.

From a single differential we can switch to a truncated differential by taking 2^x values for Δ_0. Subsequently, the number of differential characteristics becomes $2^{x+\frac{d-1}{d}n}$. Hence, this restricts x to $x < \frac{n}{d}$.

The b-δ-sequence. Evaluation of the b-δ-sequence through the $4d-3$ rounds is similar to the previous case of $d=4$. The input difference $(\Delta_0, 0, \Delta_0, \ldots, \Delta_0)$ is changed to $(\Delta, 0, \Delta, \ldots, \Delta)$. Then, the first $d-1$ rounds are passed deterministically. The middle $2d-1$ rounds are all active: hence, the values of all inputs and outputs of the $2d-1$ non-linear functions are known. Therefore, the input difference $(0, \Delta, \Delta, \ldots, \Delta)$ at the beginning of the middle rounds, can be propagated through all $2d-1$ rounds thus the difference in all d branches can be computed after the middle rounds. Finally, the difference in the second branch, when propagated through the remaining $d-1$ rounds, becomes a difference in the first branch, thus the b-δ-sequence can be computed at this branch.

Key Recovery and Complexity. In the online phase, we prepend $d-1$ rounds to the differential. The resulting input difference is $(*, \Delta_0, *, \ldots, *)$,[6] which after $d-1$ rounds will become $(\Delta_0, 0, \Delta_0, \ldots, \Delta_0)$ (with probability $2^{-(d-1)\frac{n}{d}}$), which is the input difference of the $(4d-3)$-round differential. The remaining analysis is the same as in the case of $d=4$. That is, we create a structure of plaintexts with 2^x values at the second branch, and all possible $2^{\frac{n}{d}}$ values in the remaining $d-1$ branches. In total, there are $2^{x+\frac{d-1}{d}n}$ plaintexts, which after querying result in around $2^{2x+2n\frac{d-1}{d}}$ pairs of ciphertexts. Among these pairs, we expect around $2^{2x+2n\frac{d-1}{d}-n}$ to survive the n-bit filter (the output difference of the differential is fully specified in all n bits). Note, the number of pairs is strictly less than 2^n, which is ensured by the restriction $x < \frac{n}{d}$. A portion $2^{\frac{d-1}{d}n}$ of these remaining pairs produce the required difference after the first $d-1$ rounds. Thus, to have at least one pair, it follows that $x = \frac{n}{2d}$.

The offline phase dominates the total complexity of the attack and requires $2^{\frac{n}{2d}+\frac{d-1}{d}n} = 2^{n-\frac{n}{2d}}$ time and memory. The time and data complexities of the online phase are $2^{n-\frac{n}{d}}$. As a result, we can attack $4d-3+d-1 = 5d-4$ rounds of a d-branch contracting Feistel.

3.3 Generalization to Larger Keys

With the increase of the bit length of the key, the number of attacked rounds raises and a d-branch contracting Feistel with k-bit key is susceptible to a MITM attack on $(3+2\frac{k}{n})d-4$ rounds. For instance, a 4-branch contracting Feistel with $k=2n$ bit key is vulnerable to 24-round attack. Each time the key is increased by a bit amount equivalent to one branch, we can attack two more rounds because both the offline and online phases of our previous attacks benefit from this increase.

We first focus on the offline phase, in particular on the construction of differential and enumeration of the characteristics. From the previous section, it follows that the differential covers $4d-3$ rounds when $k=n$. Assume that the key length in bits has increased for an additional $r \cdot \frac{n}{d}$ bits, i.e. for r branches. Then, we can construct a differential on $4d-3+r$ rounds that contains at most $2^{\frac{d-1}{d}n + r\cdot\frac{n}{d}} = 2^{(d-1+r)\frac{n}{d}}$ differential characteristics. This differential is constructed by adding r additional rounds in the middle of the previous differential. The output differences of the round functions in these r rounds are guessed and thus the number of differential characteristics increases $2^{r\frac{n}{d}}$ fold. In the sequel, we consider truncated differential that contains $2^{\frac{n}{2d}}$ input differences.

In the online phase, we add in total $d-1+r$ rounds to the differential: $d-1$ rounds are added before the differential (similarly to the previous attacks), and r rounds are added after the differential. We create the same structure of $2^{n-\frac{n}{d}}$ plaintexts and obtain the corresponding ciphertexts. Then, we *guess the subkeys of the last r rounds.* For each such guess, we obtain $2^{n-\frac{n}{d}}$ values of the state after the differential by inverting the last r rounds. These values are sorted according to the previous approach and the pairs of states that have the required difference are checked additionally on b-δ-sequences. That is,

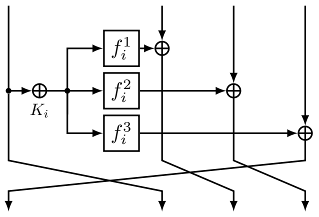
[6] A star $(*)$ denotes any nonzero difference.

for the plaintext that corresponds to the first state of the pair, we get the corresponding b-δ-sequence ciphertexts, invert them through r rounds, and finally check if the sequence of values of the states coincide with some of the sequence from the precomputed table T. It is critical to understand that we do *not* query additional plaintexts as all of them have already been queried at the beginning of the online phase.

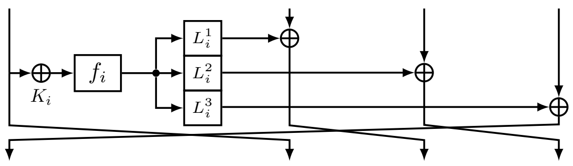
The complexity of the offline phase is $2^{(d-\frac{1}{2}+r)\frac{n}{d}} = 2^{n+r\frac{n}{d}-\frac{n}{2d}}$ in time and memory. In the online phase, we query $2^{n-\frac{n}{d}}$ chosen plaintexts, while the processing of the ciphertexts and the subkey guessing requires $2^{n-\frac{n}{d}} \cdot 2^{r\frac{n}{d}} = 2^{n+r\frac{n}{d}-\frac{n}{d}}$ operations. As the key length in bits is $k = n + r\frac{n}{d}$, our attack outperforms the exhaustive key search by a factor $2^{\frac{n}{2d}}$. In total, we are able to attack $(4d-3+r)+(d-1+r) = (3+2\frac{k}{n})d-4$ rounds.

4 Expanding Feistels with $d-1$ Functions in a Round

Expanding Feistels are another class of unbalanced Feistels. Similarly to contracting Feistels, the number of branches d can vary with $d > 2$. In one round, $d-1$ branches are updated with input from a single branch, as depicted in Figure 3a. The round starts with an XOR of a subkey, followed by keyless update function composed of $d-1$ branch-to-branch non-linear functions. While it is natural to consider $d-1$ independent functions (see the core function of CRUNCH [GIJ+08]), the combination of a single nonlinear function followed by $d-1$ linear transformations (refer to Figure 3b) is more friendly for practical implementations, as used in the hash function design ARIRANG [CHK+08]. In this paper, we consider both cases. The case of $d-1$ independent functions is presented in this section, and the case of $d-1$ linear transformations is given in Section 5.



(a) General case (Expanding Feistel-F).



(b) Particular case (Expanding Feistel-FL).

Figure 3: Round function for expanding Feistels with $d-1$ functions in the general case (a) and in a more practical scenario using a single non-linear function and $d-1$ linear functions (b).

Patarin et al. have provided in [PNB07, VNP10] security analysis of expanding Feistels. They mainly focused on the case of $d-1$ independent functions. Under the most general assumption that each of these functions is chosen uniformly at random from all the functions depending on the key, they have shown a distinguishing attack on at most $3d-1$ rounds.

A more recent application of expanding Feistels pertains to a long-lasting open problem in symmetric-key cryptography, which consists in finding a so-called white-box implementation of the Advanced Encryption Standard (AES), which resists to a powerful attacker that can see all the intermediate computations. As of today, all the propositions have been broken [BGE04, MWP10, MRP12, LRM+13]. In a recent paper [BI15], Bogdanov and Isobe solve the (much) easier problem of designing a block cipher allowing efficient white-box implementations. They achieve this using expanding Feistels and rely on keyed permutations (that is, a block cipher) as F-functions, where both input and output sizes can be adapted to offer various security/efficiency tradeoffs. The attacks presented in this paper rely on publicly known F-functions and consequently does not currently apply in our framework. We thus do not discuss further details of [BI15].

In the remaining of this section, we show how the MITM framework can be applied to expanding Feistels with $d-1$ independent but *public* functions, and provide a generic all-subkey-recovery attack on $4d-3$ rounds. We emphasize that the genericity of the attack would make it work for any key scheduling algorithm. Similarly to Section 3, we begin our description with the case when the number of branches $d=4$ and the key length in bits $k=n$, and then we generalize it to more branches and larger key lengths.

4.1 Case $k = n$ and $d = 4$

We construct a 10-round differential and append three rounds for the key recovery, which makes a 13-round key-recovery attack.

Differential. For a given pair of differences (Δ_I, Δ_O), the difference $(0, \Delta_I, 0, 0)$ becomes $(\Delta_I, 0, 0, 0)$ with probability one after 3-round encryption, and the difference $(\Delta_O, 0, 0, 0)$ becomes $(0, \Delta_O, 0, 0)$ with probability one after 3-round decryption. Theoretically, we can inject six rounds in the middle, and obtain a differential on $3+6+3=12$ rounds. However, as we explain later, the b-δ-sequence cannot be computed in the last two rounds. Therefore, we use a differential composed of only $3+6+1=10$ rounds, which starts with $(0, \Delta_I, 0, 0)$ and ends with $(0, 0, \Delta_O, 0)$. The differential is shown in Figure 8 (in Appendix). We denote the three $\frac{n}{d}$-bit output differences produced by three functions in Rounds 4, 5 and 6 by $(\Delta_{X_1}, \Delta_{X_2}, \Delta_{X_3})$, $(\Delta_{Y_1}, \Delta_{Y_2}, \Delta_{Y_3})$ and $(\Delta_{Z_1}, \Delta_{Z_2}, \Delta_{Z_3})$, respectively. Similarly, let $(\Delta_{U_1}, \Delta_{U_2}, \Delta_{U_3})$, $(\Delta_{V_1}, \Delta_{V_2}, \Delta_{V_3})$ and $(\Delta_{W_1}, \Delta_{W_2}, \Delta_{W_3})$ be the output differences of functions in Rounds 9, 8 and 7, respectively.

Let us enumerate all the differential characteristics. For this purpose, we need to identify all the valid intermediate differences $\Delta_{X_\ell}, \Delta_{Y_\ell}, \Delta_{Z_\ell}, \Delta_{W_\ell}, \Delta_{V_\ell}, \Delta_{U_\ell}$, where $\ell \in \{1,2,3\}$. When these differences are fixed, the corresponding internal state values can be obtained easily. There are $2^{\frac{n}{4}}$ choices for each difference, thus $2^{\frac{9n}{2}}$ choices for all of them. Exhaustively trying all the possibilities exceeds the cost of exhaustive key search. Hence, we need a more efficient method.

Observe that, for example, in Round 4, if one of $\Delta_{X_1}, \Delta_{X_2}$ and Δ_{X_3} is fixed, the internal state value is fixed to one choice on average, and then the corresponding output differences for the other two functions can be computed. Namely, we can enumerate $2^{\frac{n}{4}}$ choices of Δ_{X_1} and then compute the corresponding Δ_{X_2} and Δ_{X_3}. The same procedure can be applied to all six middle rounds, which will reduce the number of all differences to $2^{\frac{3n}{2}}$, however, this is still more than the cost of an exhaustive key search.

To cope with this, we use the classical meet-in-the-middle approach and find the valid differences in the middle six rounds.[7] First, in the forward direction, we enumerate $2^{\frac{3n}{4}}$ choices of $(\Delta_{X_1}, \Delta_{Y_1}, \Delta_{Z_1})$ and compute the corresponding differences after Round 6, which we store in a list L. Second, in the backward direction, we enumerate $2^{\frac{3n}{4}}$ choices of $(\Delta_{U_1}, \Delta_{V_1}, \Delta_{W_1})$ and compute the difference before Round 7, which are matched to the entries in L. A total of $2^{\frac{3n}{4}} \cdot 2^{\frac{3n}{4}} = 2^{\frac{3n}{2}}$ pairs exist and the probability of each match is 2^{-n}. As a result, in $2^{\frac{3n}{4}}$ operations, we obtain all $2^{\frac{n}{2}}$ valid differential characteristics for the middle six rounds.[8]

Evaluation of the b-δ-sequence and Truncated Differential. For each of the $2^{\frac{n}{2}}$ differential characteristics, we modify the input difference $(0, \Delta_I, 0, 0)$ to $(0, \Delta, 0, 0)$, and propagate the impact of the change. The analysis is similar to Section 3. In the first three rounds, the difference of the active branch becomes Δ. For the next six rounds, the input

[7]The usage of the classic meet-in-the-middle to find valid differences is a particular approach for expanding Feistels, which cannot be seen in the previous work on balanced Feistels. The analysis in this part is quite new.

[8]Here, we can match $\frac{n}{4}$ bits after computing two rounds in the backward direction. However, as far as we know, this does not improve the attack.

value to each function in the original pair and the modified difference allow us to compute the modified difference. This makes the state differences known in all of the four branches after nine rounds. In the 10th round, the input difference in the leftmost branch is simply copied to the second leftmost branch in the output. For the other three branches, no information is available to the attacker, thus the computation must stop here.

In summary, for each of the $2^{\frac{n}{2}}$ differential characteristics, we choose 2^b input differences $(0, \Delta, 0, 0)$, compute the corresponding differences in the second leftmost branch of Round 10, and store the sequence of differences in the precomputation table T. Note that as we explain later, the value of b is very small, i.e. $b = 3$, and thus it has only a constant impact on the attack complexity.

Similarly to Section 3, we consider $2^x < 2^n$ input differences Δ_I and $2^y < 2^n$ input differences Δ_O. This way, the complexities increase by a factor 2^{x+y} and become $2^{\frac{3n}{4}+x+y}$. A total of $2^{\frac{n}{2}+x+y}$ sequences are stored in T.

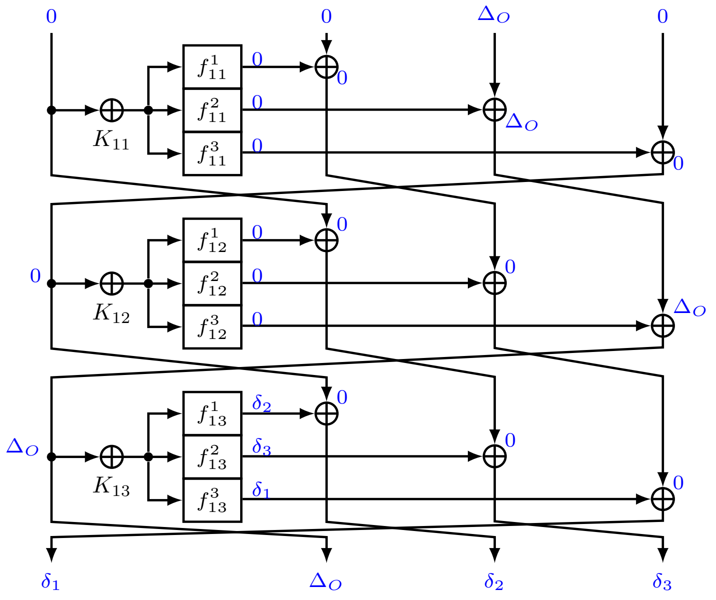


Figure 4: Key recovery for 13 rounds (case $k = n$).

Key Recovery. We append three rounds to the 10-round truncated differential, as shown in Figure 4. We first aim to obtain pairs of plaintexts that satisfy the truncated differential. We choose the plaintexts with the required input difference and then filter out wrong ciphertext pairs based on the output difference of the differential. Based on this difference, there is an obvious filter for $\frac{n}{4} - y$ bits in the second leftmost branch whose difference is limited to 2^y choices. Furthermore, we can obtain stronger filter by focusing on the truncated differential. Namely, due to the sparse propagation, the output difference of the truncated differential after Round 12 is limited to $(\Delta_O, 0, 0, 0)$, which indicates that $(\delta_1, \delta_2, \delta_3)$ in the ciphertext difference must be canceled during the decryption of Round 13. In other words, the output differences $\Delta f^1_{13}, \Delta f^2_{13}, \Delta f^3_{13}$ of three functions are determined by the ciphertext difference. For each of the 2^y choices of Δ_O, the number of all the possible differences for $(\Delta f^1_{13}, \Delta f^2_{13}, \Delta f^3_{13})$ is limited to $2^{\frac{n}{4}}$. If those limitations are exploited efficiently, more wrong pairs can be filtered out. However, because $2^{\frac{n}{4}}$ choices of $(\Delta f^1_{13}, \Delta f^2_{13}, \Delta f^3_{13})$ are dependent on Δ_O, there is no obvious way to efficiently use these filters.

Our advanced strategy uses a two-level filtering, by which the number of pairs is firstly reduced by the initial $\frac{n}{4} - y$ bits filter on Δ_O, and then the remaining pairs are further filtered by the $2^{\frac{n}{4}}$ possibilities of $(\Delta f^1_{13}, \Delta f^2_{13}, \Delta f^3_{13})$. More precisely, we perform the procedure described in Algorithm 1. At Step 8 of the procedure, the number of possible pairs is 2^{2x} and the number of bits for the filter is $\frac{n}{4} - y$, thus $2^{2x-\frac{n}{4}+y}$ pairs will remain. At Step 9, the probability that each pair remains is $2^{\frac{n}{4}}/2^{\frac{3n}{4}} = 2^{-\frac{n}{2}}$, thus $2^{2x-\frac{3n}{4}+y}$ pairs will remain. This is iterated $2^{\frac{3n}{4}}$ times by the for loop specified at Step 5. As a result, we obtain 2^{2x+y} pairs satisfying the truncated differential. For each such pair, the probabilities that the differential propagation backwards in the last three rounds follows the one given in Figure 4 are $2^{-\frac{n}{4}}$, 1 and 1 for Rounds 13, 12 and 11, respectively. Thus when $2^{2x+y} = 2^{\frac{n}{4}}$, collecting $2^{\frac{n}{4}}$ pairs will assure that there is at least one pair sufficient to recover the key.

For each of the $2^{\frac{n}{4}}$ obtained pairs, we construct the b-δ-sequence according to the framework introduced in Section 2. We start with a partial decryption from the corresponding ciphertexts. In the last round, the internal state value after the subkey can be recovered with $T_{filter}^{\Delta_O}$. This allows to uniquely recover the last subkey K_{13}. We then exhaustively guess $\frac{n}{2}$ bits of (K_{12}, K_{11}), which is sufficient to carry out the partial decryption from the ciphertext to the target difference at the state between Rounds 10 and 11, and thus we can compute b-δ-sequence and compare to the stored values of the table T.

Complexity Analysis. The complexities required to enumerate all the differences in the precomputation phase are $2^{\frac{3n}{4}+x+y}$ time and memory. The offline preparation in Algorithm 1 requires only $2^{y+\frac{n}{4}}$ time and memory, which is negligible compared to the enumeration of the differences. The data complexity for the $\frac{3n}{4}$ bits for inactive branches (Step 5) and x bits for active branch (Step 6) amounts to $2^{\frac{3n}{4}+x}$ chosen plaintexts. The time complexity for the filtering is $2^{\frac{3n}{4}} \cdot 2^x$ for Step 8 and $2^{\frac{3n}{4}} \cdot 2^{2x-\frac{n}{4}+y}$ for Step 9, or in total $2^{\frac{3n}{4}+x} + 2^{\frac{n}{2}+2x+y}$. As explained before, we have the condition $2^{2x+y} = 2^{\frac{n}{4}}$. Then, the complexity for the filtering phase becomes $2^{\frac{3n}{4}+x}$ for a sufficiently large x. After obtaining $2^{\frac{n}{4}}$ pairs satisfying the 13-round truncated differential, the time complexity for recovering K_{13}, K_{12} and K_{11} is $2^{\frac{n}{2}}$ for each pair due to the exhaustive guess of (K_{11}, K_{12}). Thus, the time complexity for the key recovery phase is $2^{\frac{n}{4}+\frac{n}{2}} = 2^{\frac{3n}{4}}$ partial decryptions.

We use $x = \frac{n}{8}$ and $y = 0$ to balance the complexities under the condition $2^{2x+y} = 2^{\frac{n}{4}}$, which makes the time, memory and data complexities equal to $2^{\frac{7n}{8}}$. Finally, let us evaluate the value of b. There are $2^{\frac{7n}{8}}$ b-δ-sequences stored in T and $2^{\frac{3n}{4}}$ b-δ-sequences are generated online. Matching each differences reduces the number of correct key candidates by a factor $2^{-\frac{n}{4}}$. Hence, setting $b = 3$ to provide $2^{(-\frac{n}{4})\cdot 2^3} = 2^{-2n}$ matching condition suffices to detect the correct key.

Algorithm 1: Two-level filter for data collection.

```
Offline Preparation
 1: for $2^y$ choices of $\Delta_O$ do
 2:   for $i = 0, 1, \ldots, 2^{\frac{n}{4}} - 1$ do
 3:     Compute $(\delta_1, \delta_2, \delta_3) = \big(f_{13}^3(i) \oplus f_{13}^3(i \oplus \Delta_O), f_{13}^1(i) \oplus f_{13}^1(i \oplus \Delta_O),$
        $f_{13}^2(i) \oplus f_{13}^2(i \oplus \Delta_O)\big)$. Store the result along with the value $i$ in $T_{filter}^{\Delta_O}$.
 4:   Hash $T_{filter}^{\Delta_O}$ for a later usage as a filter.
Online Filtering
 5: for $2^{\frac{3n}{4}}$ values of three inactive branches in the plaintext do
 6:   for $2^x$ values of the active branch in the plaintext do
 7:     Query the chosen plaintexts. Store the corresponding ciphertexts in $T_{ct}$ indexed
        by the value of the second leftmost branch.
 8:   Pick up all the pairs in $T_{ct}$ satisfying any of $2^y$ differences for $\Delta_O$.
 9:   For the picked up pairs, check if $(\delta_1, \delta_2, \delta_3)$ can be produced by $\Delta_O$,
      i.e. check the match of $(\delta_1, \delta_2, \delta_3)$ and $T_{filter}^{\Delta_O}$.
10:   The pairs with successful match are regarded as passing the filter: they are
      used for the key recovery.
```

4.2 Generalization to More Branches

The 13-round attack for $d = 4$ consists of 10-round differential and 3-round key recovery. The 10-round differential can be further divided into three rounds with a probability-one differential, six rounds covered by the classical MITM approach and a final round allowing the simple copy from the input to the output.

This can be generalized to a d-branch case as follows. The differential starts with $d-1$ rounds that hold with probability one. Then, the number of middle rounds is $2(d-1)$ so that all the differential characteristics can be enumerated with the meet-in-the-middle approach with a complexity below 2^n. At the end, one additional round is appended (it corresponds to the simple copy from the input to the output). The key recovery part will also have $d-1$ rounds such that the sparse differential continues for $d-2$ rounds followed by an additional round. In summary, $(d-1)+2(d-1)+1+(d-1) = 4d-3$ rounds can be attacked under the condition $k = n$. The attack complexity is faster than the exhaustive search by a factor of $2^{\frac{n}{2d}}$, and it amounts to $2^{n-\frac{n}{2d}}$.

Compared to the previous $(3d-1)$-round attack by Patarin et al. in [PNB07, VNP10], our MITM attack penetrates significantly more rounds. However, we remind the reader that the round function considered in this paper are less general than the Luby-Rackoff-like structure tackled by Patarin et al. We indeed study the more practical Feistel-2-like structure. Moreover, we note that the analysis of the most general case conducted by Patarin et al. yields distinguishers, while the more specific framework examined here allows to reach all-subkey-recovery attacks.

4.3 Generalization to Larger Key Lengths

When the key length in bits k increases, so does the number of attacked rounds. The most straightforward attack for $k = n + \frac{rn}{d}$ for $r \geq 1$ is obtained by iterating the same attack as for $k = n$ and by exhaustively guessing the additional $\frac{rn}{d}$-bit subkeys. This strategy works for r additional rounds compared to $k = n$.

We can penetrate larger number of rounds by exploiting the property that the MITM attack matches results computed independently in offline phase and online phases. In each of these phases, we can additionally spend $2^{\frac{rn}{d}}$ cost by exhaustively guessing subkeys for r more rounds, thus $2r$ additional rounds can be attacked than in the case of $k = n$. We need additional filter for the increased $2^{\frac{rn}{d}} \cdot 2^{\frac{rn}{d}} = 2^{\frac{2rn}{d}}$ bits when the results from offline and online phases are matched. This can be achieved by merely increasing b in the b-δ-sequence by one bit, which requires a negligibly small additional cost. Recall that $4d-3$ rounds are attacked for $k = n$. In the case of larger keys, $4d-3+2r$ rounds are attacked for $k = n + \frac{rn}{d}$. For example, for $d = 4$ and $k = 2n$, 21 rounds can be attacked. The improved factor of the attack complexity stays unchanged, and is $\frac{n}{2d}$ bits faster than the exhaustive search, i.e. $2^{n+\frac{rn}{d}-\frac{n}{2d}}$.

When $r = 1$, i.e. $k = n + \frac{n}{d}$, we can further improve the attack by exploiting the fact that the enumeration of the characteristics in the offline phase exploits the classical meet-in-the-middle approach, thus can be further divided into two independent computations. In the characteristics enumeration, $2^{(d-1)\frac{n}{d}}$ possible differential characteristics are obtained both in forward and backward computations and the n-bit match is examined in the middle, which provides $2^{2(d-1)\frac{n}{d}-n} = 2^{(d-2)\frac{n}{d}}$ valid characteristics and those are stored in a table T.[9] Here, the computational cost of $2^{\frac{(d-1)n}{d}}$ is not balanced with the memory requirement for T which is $2^{\frac{(d-2)n}{d}}$. In the particular case of $k = n + \frac{n}{d}$, we can set up the attack so that those two complexities are indeed balanced. In the characteristics enumeration, we inject two more rounds in the middle (when $r = 1$). Each direction enumerates 2^n

[9]For $d = 4$ in Section 4.1, $2^{\frac{3n}{4}}$ characteristics were enumerated in both directions and $2^{\frac{3n}{4}+\frac{3n}{4}-n} = 2^{\frac{n}{2}}$ valid combinations were obtained.

characteristics with time complexity 2^n and $2^{n+n-n} = 2^n$ valid characteristics are provided after matching n bits in the middle, which requires 2^n memory. Thus, the two complexities are balanced and the attack is still faster than the exhaustive search of $(n + \frac{n}{d})$-bit key. In summary, for $r = 1$, two rounds are added in the offline phase and one round in the online phase and as a result a total of $4d - 3 + 3 = 4d$ rounds are attacked. At the end, we note that adding two (or more) rounds in the offline phase cannot be achieved when $r > 1$. This is because the number of matched (filtering) bits in the classical meet-in-the-middle approach will not increase beyond n even when r increases. The increase of the key length by $\frac{n}{d}$ bits allows the attacker to spend an additional factor of $2^{\frac{n}{d}}$, however, by adding two rounds in the offline phase, the number of valid differential characteristics stored in the table increases by a factor of $2^{\frac{2n}{d}}$, thus the factor is not sufficient to cover the higher number of characteristics.

5 Expanding Feistels with $d - 1$ Linear Transformations

In this section, we apply the MITM attack to expanding Feistels with a round update composed of a non-linear function followed by $d - 1$ linear transformations. The general construction of this subclass of expanding Feistels is depicted in Figure 3b. The round key is XORed into one branch, followed by the application of the non-linear updating function F. Then, the output is passed through distinct linear transformations $L_1, L_2, \dots, L_{d-1}$ to update each of the other $d - 1$ branches. One can view this subclass as a special case of the expanding Feistels considered in Section 4, i.e., each updating function comprises the linear transformation and the non-linear function. Hence, all results presented in Section 4 apply here naturally. On top of that, we show how to utilize the properties of these linear transformations and how to attack $5d - 4$ rounds (cf. to $4d - 3$ rounds before) when $k = n$, and a larger number of rounds when the key length increases. Note, the updating functions and linear transformations can be different between or within each round, and this is irrelevant to the presented attacks.

In the same manner, we begin our analysis for the case of key length $k = n$ and $d = 4$ branches in Section 5.1, which is one of the most commonly used setting in real block cipher designs. Later, we extend our attacks to larger numbers of branches and key length in Section 5.2.

5.1 Case $k = n$ and $d = 4$

Differential. We use a differential on 13 rounds (depicted in Figure 9 in Appendix), with an input difference $(0, \Delta_I, 0, 0)$ and an output difference $(\Delta_O, 0, 0, 0)$, where Δ_I and Δ_O are arbitrary nonzero values. The input and output differences are chosen such that the differential penetrates the maximal number of rounds. The input difference $(0, \Delta_I, 0, 0)$ propagates in three rounds to $(\Delta_I, 0, 0, 0)$ and with probability one, since no difference passes through the F functions in the first three rounds. Afterwards, the difference goes through three more rounds and is determined by output differences Δ_X, Δ_Y and Δ_Z of F at Rounds 4, 5, and 6. The number of possible differences is increased by a factor of $2^{\frac{n}{4}}$ at each of these rounds, resulting in $2^{\frac{3n}{4}}$ possibilities at the end of Round 6 of the differential. Similarly, at the other end of the differential, the output difference $(\Delta_O, 0, 0, 0)$ becomes $(0, \Delta_O, 0, 0)$ at the beginning of Round 11 with probability one. As a result, there are four rounds left in between. We show that for each pair of difference at the 6-th and the 11-th rounds, there is a unique $(\Delta_M, \Delta_W, \Delta_V, \Delta_U)$ that connect them and hence the output differences of Rounds 7, 8, 9, and 10 are uniquely determined. Overall, there are $2^{\frac{3n}{4}}$ differential characteristics for each fixed input and output difference of the differential, thus much less than that of a random permutation with n-bit key.

Further, we explain how the differences in the middle four rounds can be computed. Let us denote the input difference at the i-th round (in our case, $i = 7$) as $(\Delta_A^1, \Delta_A^2, \Delta_A^3, \Delta_A^4)$, and the output difference of the $(i + 3)$-th round as $(\Delta_B^1, \Delta_B^2, \Delta_B^3, \Delta_B^4)$. We can express the input/output differences and the four intermediate differences as the following system of four linear equations:

$$\begin{cases} \Delta_A^1 + & & L_{i+2}^1(\Delta_W) + & L_{i+3}^2(\Delta_V) + & L_{i+4}^3(\Delta_U) & = \Delta_B^1 \\ \Delta_A^2 + & L_{i+1}^1(\Delta_M) + & L_{i+2}^2(\Delta_W) + & L_{i+3}^3(\Delta_V) & & = \Delta_B^2 \\ \Delta_A^3 + & L_{i+1}^2(\Delta_M) + & L_{i+2}^3(\Delta_W) + & & L_{i+4}^1(\Delta_U) & = \Delta_B^3 \\ \Delta_A^4 + & L_{i+1}^3(\Delta_M) + & & L_{i+3}^1(\Delta_V) + & L_{i+4}^2(\Delta_U) & = \Delta_B^4. \end{cases} \tag{1}$$

This system can be rewritten in the matrix form $(\Delta_M, \Delta_W, \Delta_V, \Delta_U)^T = \mathbf{M_4}^{-1} \times (\Delta_B^1 - \Delta_A^1, \Delta_B^2 - \Delta_A^2, \Delta_B^3 - \Delta_A^3, \Delta_B^4 - \Delta_A^4)^T$, where $\mathbf{M_4}$ is a 4×4 matrix with entries of L's or zero.

Special Matching. When the difference pairs are three or less rounds apart, the number of intermediate differences reduces, resulting in an *over-defined* system of linear equations with four equations but less unknown variables. In such cases, not all input/output difference pairs have solutions. Given the input and output, we further show how we can check whether the two states can be connected, on an example of state values instead of differences. Following the above notations, we denote the state value as (A_1, A_2, A_3, A_4) at round i and (B_1, B_2, B_3, B_4) at round $i + 3$, and use (M, W, V) for the output of the F functions at rounds $i + 1$, $i + 2$ and $i + 3$. The system of equations reduces to:

$$\begin{cases} A_1 + & & L_{i+2}^1(W) + & L_{i+3}^2(V) & = B_4 \\ A_2 + & L_{i+1}^1(M) + & L_{i+2}^2(W) + & L_{i+3}^3(V) & = B_1 \\ A_3 + & L_{i+1}^2(M) + & L_{i+2}^3(W) + & & = B_2 \\ A_4 + & L_{i+1}^3(M) + & & L_{i+3}^1(V) & = B_3. \end{cases} \tag{2}$$

Similarly, the system can be rewritten in the matrix form: $(B_4 - A_1, B_1 - A_2, B_2 - A_3, B_3 - A_4)^T = \mathbf{M}_3 \times (M, W, V)^T$, where $\mathbf{M}_3$ is a 4×3 matrix with entries of L's and zeros. Let us denote $\mathbf{M_3^U}$ a new 3×3 matrix duplicating the first three rows of $\mathbf{M_3}$, and $\mathbf{M_3^B}$ as the last row of $\mathbf{M_3}$. Then $(M, W, V)^T = \mathbf{M_3^U}^{-1}(B_4 - A_1, B_1 - A_2, B_2 - A_3)^T$ and $B_3 - A_4 = \mathbf{M_3^B}(M, W, V)^T = \mathbf{M_3^B}\mathbf{M_3^U}^{-1}(B_4 - A_1, B_1 - A_2, B_2 - A_3)^T$. Hence:

$$\mathbf{M_3^B}\mathbf{M_3^U}^{-1}(B_4, B_1, B_2)^T - B_3 = \mathbf{M_3^B}\mathbf{M_3^U}^{-1}(A_1, A_2, A_3)^T - A_4. \tag{3}$$

Given two lists of states A and B, we can compute the two sides of the above equation independently and store one side in a hash table, then match the other side by looking up the hash table. Note, when the two states are less than three rounds apart, we will obtain more equations. In general, x such equations for $4 - x$ rounds achieve a filtering power of $\frac{xn}{d}$ bits.

The same argument applies when one matches the state difference, instead of values. In fact, the same matrices and formulae as above can be used. Lastly, when the input difference is given, one can also sort the output values (instead of differences), then filter out those output pairs which are possible to be connected with the input difference.

Truncated Differential and b-δ-sequence. We can choose to use multiple Δ_I and Δ_O to balance different aspects of the attack complexities. We use here a single choice of Δ_I and $2^{\frac{n}{8}}$ choices of Δ_O. Hence, we need to compute and store the b-δ-sequence for a total of $2^{\frac{n}{8}+\frac{3n}{4}} = 2^{\frac{7n}{8}}$ differential characteristics.

In contrast to the previous analysis, the b-δ-sequence cannot be computed in any of the branches after 13 rounds. It can be computed trivially in all the branches after 10 rounds. However, only one branch will remain computable after the 11-th round, and none for more rounds, since the internal values and differences of F functions are unknown for Rounds 11 to 13. Hence, we compute the b-δ-sequence on the entire state only after the 10-th round and store them in the table T. We show later how these values can be used for online matching.

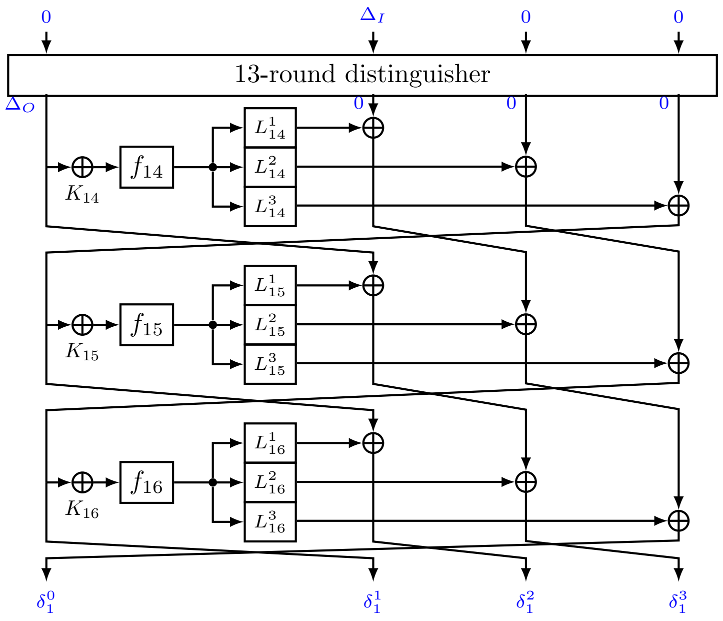


Figure 5: Key recovery for 16 rounds in the case $k = n$. The attack makes use of a 13-round differential.

Data Collection, Key Recovery and Complexity. As depicted in Figure 5, the overall attack penetrates 16 rounds: we append three rounds to the 13-round differential. We collect sufficiently many ciphertext pairs by querying plaintext pairs with input difference $(0, \Delta_I, 0, 0)$. To ensure the specific form of the output difference at the 13-th round can be achieved, we need $2^{n-\frac{n}{8}} = 2^{\frac{7n}{8}}$ plaintext pairs. As an intermediate step, for each of the $2^{\frac{n}{8}}$ output differences that belong to the truncated differential, we compute all possible state differences that can be achieved after three rounds. This results in around $2^{\frac{n}{8}} \cdot 2^{\frac{3n}{4}} = 2^{\frac{7n}{8}}$ output differences (at Round 16) that we store in a hash table T_O. Each entry of this table contains as well the corresponding input difference and the three output differences from the round functions. The table is indexed by the output differences and is used in the online phase to filter out wrong ciphertext pairs. Out of all $2^{\frac{7n}{8}}$ ciphertext pairs, only $2^{\frac{7n}{8}+\frac{7n}{8}-n} = 2^{\frac{3n}{4}}$ are expected to match against the entries of T_O. In addition, for each of these pairs, from T_O we obtain the output differences at Rounds 14, 15, and 16 (and thus subkey suggestions) as well as the state difference after Round 13.

Further filtration makes use of the b-δ-sequence and the table T. For each of the remaining $2^{\frac{3n}{4}}$ pairs, we construct the b-δ-sequence of plaintexts and obtain a sequence of ciphertexts. We decrypt the sequence for three backward rounds (with the use of three suggested subkeys for each pair), and obtain a sequence of state values at the end of Round 13. We turn the sequence of state values into a sequence of state differences by computing the difference between the sequence of values with the value of the state that corresponds to the original pair (from the initial pair of plaintexts). By performing this task for all of the filtered pair, we end up with $2^{\frac{3n}{4}}$ sequences of state differences at Round 10. On the other hand, in the table T we have another $2^{\frac{7n}{8}}$ b-δ-sequence differences that correspond to the state differences at Round 13. The final step is to find a match between these two sets. As we have only three rounds in between, the matching can be performed with the use of (3) – obviously the right and the left side of the equation are independent, thus in a meet-in-the-middle fashion, the match can be found. As the sequences are composed of several differences, the match will be unique (i.e. even though we are matching on $\frac{n}{4}$ bits per equation, there are b such equations, thus we can find the unique solution).

The overall complexities of the attack are $2^{\frac{7n}{8}}$ for time, memory and data. Finally, the required value of b can be estimated as follows. There are $2^{\frac{3n}{4}}$ sequences at Round 13, and $2^{\frac{7n}{8}}$ sequences at Round 10, and each value in the b-δ-sequence carries a filtering power of $\frac{n}{4}$ bits. We therefore need $2^b \cdot \frac{n}{4} > \frac{3n}{4} + \frac{7n}{8}$, hence $b = 3$.

5.2 Generalization to More Branches and Larger Key Lengths

For a general expanding Feistels with d branches, the differential covers $4d - 3$ rounds. Similarly to the previously considered cases, the input difference passes $d - 1$ rounds with probability one, and $d - 1$ additional rounds, each with a guess of a branch difference, resulting in $2^{\frac{(d-1)n}{d}}$ possible differences after $2d - 2$ rounds. The output difference, in the backward direction, also passes $d - 1$ rounds with probability one, while the intermediate d rounds can be determined completely. Hence, overall $(d-1)+(d-1)+d+(d-1) = 4d-3$ round compose the differential, with $2^{\frac{(d-1)n}{d}}$ entries in the precomputed table T. To balance the attack complexity, we use $2^{\frac{n}{2d}}$ choices for the input difference, resulting in about $2^{\frac{(2d-1)n}{2d}}$ entries in T. In the key recovery phase, we append $d-1$ rounds at the end of the differential. The remaining of the attack is similar to the attack of the previous section. Note that in this case, we have a linear system of d equations, with $d - 1$ unknowns, but a similar equation as (3) will still be able to provide a match between the sequences from the table T and from the ciphertexts pairs. Overall, we can attack $(4d - 3) + (d - 1) = 5d - 4$ rounds with time, memory, and data complexities of $2^{\frac{(2d-1)n}{2d}}$.

For a key length in bits $k = n + \frac{rn}{d}$, we can extend the differential by r additional rounds and add r rounds in the online phase of the attack. The analysis is as in the previous sections and we omit it. We only note that the complexities are increased by a factor of $2^{\frac{rn}{d}}$, resulting in overall complexities of $2^{n+\frac{(2r-1)n}{2}d}$ time, memory and data for an attack on $5d - 4 + 2r$ rounds.

6 Discussions: Comparison with Other Work

6.1 Comparison with Extended Generalized Feistel Network

At SAC 2013, Berger et al. proposed a new class of Feistel network called *Extended Generalized Feistel Network (EGFN)* [BMT13]. EGFN enables a synthetic representation of various generalized Feistel network including applications of non-linear functions from one branch to several branches or from several to one. Besides, it also considers applications of linear functions. For example, the network in Figure 6 with non-linear functions F_1, F_2 and linear functions L_1, L_2 can be represented using the following matrix.

$$\begin{pmatrix} L_1 & F_1 & 1 & 0 \\ F_2 & L_2 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Expanding Feistels with $d - 1$ functions that we discussed in 3a in Section 4 is exactly

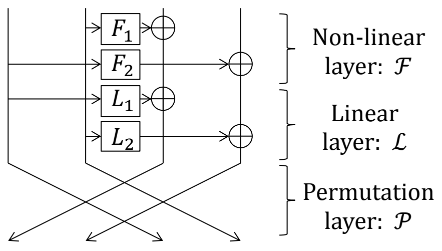

F_1
F_2
L_1
L_2
Non-linear layer: $\mathcal{F}$
Linear layer: $\mathcal{L}$
Permutation layer: $\mathcal{P}$

Figure 6: Extended Generalized Feistel Network.

an instantiation of EGFN, whose network can be represented by the following matrix.

$$\begin{pmatrix} f^3 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ f^1 & 1 & 0 & 0 \\ f^2 & 0 & 1 & 0 \end{pmatrix}$$

Therefore, our attacks can be directly applied to this instantiation of EGFN.

On the other hand, EGFN cannot represent network that applies a non-linear function to the result of linear combination of branches or applies several different linear functions after a non-linear function is applied. Thus, there is a gap between EGFN and the contracting Feistel and expanding Feistels with $d-1$ linear transformations discussed in this paper. We think that filling this gap is a good future research direction. One approach is (possible automatically) extending our attacks to any arbitrary matrix choice of EGFN. Another approach is proposing yet another type of generalized Feistel that can deal with the constructions discussed in this paper.

6.2 Detailed Comparison about SMS4

Because our attacks on contracting Feistel models the round function of SMS4, the attack with 4-branches, $k = n$ can be directly used to recover the key of SMS4. If applied, 16 rounds can be attacked with complexity of $2^b \times 2^{3n/4}$, where $b = 3$ and $n = 128$, i.e. the attack complexity is 2^{99} operations. We note that this is a direct application of our generic attack without any dedicated optimization. It will be interesting to compare the number of attacked rounds between our MitM attack and other approaches. Summary of existing attacks is given in Table 2.

The current best attacks are differential cryptanalysis and multiple linear cryptanalysis reaching 23 rounds out of 32 rounds. In general, differential and linear cryptanalysis and their variant such as boomerang or rectangle attacks well work for SMS4. Meanwhile, other attacks such as integral, impossible differential, algebraic attacks are less effective than differential and linear cryptanalysis. The efficiency of the MitM attack against SMS4 belongs to the latter case. However, it is still interesting to observe that the straightforward application of our generic MitM attack is better than several approaches with a lot of optimizations such as using relations of subkeys.

7 Conclusion

We have presented new generic attacks on classes of contracting and expanding Feistels. Our results show that when we switch from Luby-Rackoff round function as analyzed by Patarin et al. to more implementation-friendly round functions (such as keyless SP, SPS, ARX, etc., surrounded by subkey additions), then contracting and expanding Feistels

Table 2: Comparison of existing attacks against `SMS4`.

Attack	**Rounds**	**Data**	**Time**	**Reference**
Differential	21	2^{118}	$2^{126.6}$	[ZZW08]
Differential	22	2^{118}	$2^{125.7}$	[TKS08]
Differential	22	2^{117}	$2^{112.3}$	[ZWFS09]
Differential	23	2^{115}	$2^{124.3}$	[SWZ10]
Linear	22	2^{117}	$2^{120.4}$	[TKS08]
Linear	22	2^{117}	$2^{112.3}$	[ER08]
Multiple linear	22	2^{112}	$2^{124.2}$	[LGZ09]
Multiple linear	23	$2^{126.6}$	$2^{127.4}$	[CN11]
Boomerang	18	2^{120}	$2^{116.8}$	[TKS08]
Rectangle	14	$2^{121.8}$	$2^{116.7}$	[Lu07]
Rectangle	14	$2^{107.9}$	$2^{107.9}$	[TD08]
Rectangle	16	2^{125}	2^{116}	[ZZW08]
Rectangle	18	2^{124}	$2^{112.8}$	[TKS08]
Rectangle	18	2^{127}	$2^{103.8}$	[KWX13]
Integral	13	2^{16}	2^{114}	[LJH+07]
Impossible differential	16	2^{105} †	2^{107} †	[Lu07]
Impossible differential	16	$2^{117.1}$	2^{132} ‡	[TD08]
Impossible differential	17	$2^{117.1}$	2^{132} ‡	[Wan10]
Impossible differential	18	$2^{117.1}$	2^{132} ‡	[SWX12]
Algebraic	7	–	–	[EDC09]
MitM	16	2^{99}	2^{99}	**Ours**

†: It was pointed out by [TD08] that those complexities are underestimated.
‡: Assuming that 2^{132} memory access is faster than 2^{128} encryptions.

are susceptible to MITM attacks on a much larger number of rounds. For example, in the case of 4-branch contracting Feistels that have the same key and block lengths, this results in an increase from 7 rounds to 16 rounds, while in the case of expanding Feistels, from 11 rounds to 16 rounds. Moreover, our attacks penetrate even larger number of rounds with the increase of the number of branches and with the increase of the key length. For instance, `SMS4`-like cipher with a double key is vulnerable to a 24-round full subkey recovery attack. In general, we have shown attacks on d-branch unbalanced Feistels with k-bit keys on: $(3 + 2\frac{k}{n})d - 4$ rounds for contracting Feistels and expanding Feistels with linear expansion, and $(2 + 2\frac{k}{n})d - 3$ rounds for the general case of expanding Feistels.

Although we have analyzed generic constructions without exploiting particular properties of the round functions, some of our attacks, in terms of penetrated rounds, are on par with attacks on dedicated designs. For instance, several results on the block cipher `SMS4`, including integral, rectangle and impossible differentials penetrate 13 to 16 rounds [Lu07, TD08, ZZW08]. Our generic attack on contracting Feistels and thus on `SMS4` reaches 16 rounds and fully recovers all the subkeys. In other words, in the same chosen-plaintext framework, we can launch stronger attacks, without considering the round function and the key schedule.

Additionally, the number of rounds required to achieve full diffusion is not correlated to the number of rounds penetrated with our MITM attacks. Yanagihara and Iwata have found in [YI13] that four rounds are sufficient to achieve full diffusion in the case of contracting and expanding Feistels with the number of branches $d = 4$. Even twice this number is not sufficient to prevent the MITM attacks on these Feistels. Hence, the designers should not estimate the resistance of the cipher based solely on the diffusion property. Furthermore, the resistance cannot be proven even if the designers focus on some of the differential properties of the Feistels. For instance, Yanagihara and Iwata show that no differential characteristic can exist for seven rounds of contracting Feistels with $d = 4$. Our 16-round MITM attack indicates that the security margin should be much larger.

As MITM have been the most successful attacks against balanced Feistels [GJNS14] as well as against the class of unbalanced Feistels analyzed in this paper, it would be interesting to understand their importance against other classes of Feistels, both generic and dedicated. We leave as an open problem the analysis of Type 1, 2, 1.x Feistels [YI14], more generalized class of unbalanced Feistels [BMT13] and designs based on dedicated Feistels (such as `CRUNCH` [GIJ+08] and `ARIRANG` [CHK+08]).

Acknowledgement

We would like to thank Patrick Derbez for the help during the shepherding phase. Ivica Nikolić is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06).

References

[Abe10] Masayuki Abe, editor. *Advances in Cryptology - ASIACRYPT 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*. Springer, 2010.

[AS09] Kazumaro Aoki and Yu Sasaki. Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 70–89. Springer, 2009.

[BCD+98] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M Matyas Jr, Luke O'Connor, Mohammad Peyravian, David Safford, et al. MARS-a candidate cipher for AES, 1998.

[BDP15] Alex Biryukov, Patrick Derbez, and Léo Perrin. Differential Analysis and Meet-in-the-Middle Attack Against Round-Reduced TWINE. In Leander [Lea15], pages 3–27.

[BGE04] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2004.

[BI15] Andrey Bogdanov and Takanori Isobe. White-box cryptography revisited: Space-hard ciphers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1058–1069. ACM, 2015.

[BMT13] Thierry P. Berger, Marine Minier, and Gaël Thomas. Extended Generalized Feistel Networks Using Matrix Representation. In Lange et al. [LLL14], pages 289–305.

[CHK$^+$08] Donghoon Chang, Seokhie Hong, Changheon Kang, Jinkeon Kang, Jongsung Kim, Changhoon Lee, Jesang Lee, Jongtae Lee, Sangjin Lee, Yuseop Lee, Jongin Lim, and Jaechul Sung. ARIRANG. Submission to SHA-3 Competition, organized by NIST, 2008.

[CN11] Joo Yeon Cho and Kaisa Nyberg. Improved linear cryptanalysis of sms4 block cipher. Workshop Record of SKEW 2011, 2011.

[Cop94] Don Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. *IBM journal of research and development*, 38(3):243–250, 1994.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic Search of Meet-in-the-Middle and Impossible Differential Attacks. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 157–184. Springer, 2016.

[DFJ13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 371–387. Springer, 2013.

[DH77] W. Diffie and M.E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10(6):74–84, 1977.

[DKS10a] Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved Single-Key Attacks on 8-round AES-192 and AES-256. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 158–176. Springer, 2010.

[DKS10b] Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In Abe [Abe10], pages 158–176.

[DLJW15] Xiaoyang Dong, Leibo Li, Keting Jia, and Xiaoyun Wang. Improved Attacks on Reduced-Round Camellia-128/192/256. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, volume 9048 of *Lecture Notes in Computer Science*, pages 59–83. Springer, 2015.

[DP15] Patrick Derbez and Léo Perrin. Meet-in-the-Middle Attacks and Structural Analysis of Round-Reduced PRINCE. In Leander [Lea15], pages 190–216.

[DS08] Hüseyin Demirci and Ali Aydin Selçuk. A Meet-in-the-Middle Attack on 8-Round `AES`. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 116–126. Springer, 2008.

[EDC09] Jeremy Erickson, Jintai Ding, and Chris Christensen. Algebraic cryptanalysis of SMS4: gröbner basis attack and SAT attack compared. In Dong Hoon Lee and Seokhie Hong, editors, *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*, volume 5984 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2009.

[ER08] Jonathan Etrog and Matthew J. B. Robshaw. The cryptanalysis of reduced-round SMS4. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC*

2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers, volume 5381 of *Lecture Notes in Computer Science*, pages 51–65. Springer, 2008.

[FWG+16] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2016.

[GIJ+08] Louis Goubin, Mickael Ivascot, William Jalby, Olivier Ly, Valerie Nachef, Jacques Patarin, Joana Treger, and Emmanuel Volte. CRUNCH. Submission to NIST, 2008.

[GJNS14] Jian Guo, Jérémy Jean, Ivica Nikolić, and Yu Sasaki. Meet-in-the-Middle Attacks on Generic Feistel Constructions. In Sarkar and Iwata [SI14], pages 458–477.

[GLRW10] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In Abe [Abe10], pages 56–75.

[Gol96] Dieter Gollmann, editor. *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*. Springer, 1996.

[GSW+14] Jian Guo, Yu Sasaki, Lei Wang, Meiqin Wang, and Long Wen. Equivalent key recovery attacks against HMAC and NMAC with whirlpool reduced to 7 rounds. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 571–590. Springer, 2014.

[HHN00] Helena Handschuh, H Helena, and David Naccache. SHACAL (-Submission to NESSIE-), 2000.

[KWX13] Xianglong Kong, Wei Wang, and Qiuliang Xu. Improved rectangle attack on SMS4 reduced to 18 rounds. In *Ninth International Conference on Computational Intelligence and Security, CIS 2013, Emei Mountain, Sichan Province, China, December 14-15, 2013*, pages 575–578. IEEE Computer Society, 2013.

[Lea15] Gregor Leander, editor. *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*. Springer, 2015.

[LGZ09] Zhiqiang Liu, Dawu Gu, and Jing Zhang. Multiple linear cryptanalysis of reduced-round sms4 block cipher. Cryptology ePrint Archive, Report 2009/256, 2009. http://eprint.iacr.org/2009/256.

[LJ15] Rongjia Li and Chenhui Jin. Meet-in-the-Middle Attacks on 10-Round AES-256. *Designs, Codes and Cryptography*, pages 1–13, 2015.

[LJH+07] Fen Liu, Wen Ji, Lei Hu, Jintai Ding, Shuwang Lv, Andrei Pyshkin, and Ralf-Philipp Weinmann. Analysis of the SMS4 block cipher. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, volume 4586 of *Lecture Notes in Computer Science*, pages 158–170. Springer, 2007.

[LJW14] Leibo Li, Keting Jia, and Xiaoyun Wang. Improved Single-Key Attacks on 9-Round AES-192/256. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 2014.

[LJWD15] Leibo Li, Keting Jia, Xiaoyun Wang, and Xiaoyang Dong. Meet-in-the-Middle Technique for Truncated Differential and Its Applications to CLEFIA and Camellia. In Leander [Lea15], pages 48–70.

[LLL14] Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors. *Selected Areas in Cryptography - SAC 2013*, volume 8282 of *Lecture Notes in Computer Science*. Springer, 2014.

[LRM$^+$13] Tancrède Lepoint, Matthieu Rivain, Yoni De Mulder, Peter Roelse, and Bart Preneel. Two Attacks on a White-Box AES Implementation. In Lange et al. [LLL14], pages 265–285.

[Lu07] Jiqiang Lu. Attacking Reduced-Round Versions of the SMS4 Block Cipher in the Chinese WAPI Standard. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS 2007*, volume 4861 of *Lecture Notes in Computer Science*, pages 306–318. Springer, 2007.

[Lu08] Shu-Wang Lu. SMS4 Encryption Algorithm for Wireless Networks. Cryptology ePrint Archive, Report 2008/329, 2008. Translated from Chinese by Whitfield Diffie and George Ledin.

[Luc96] Stefan Lucks. Faster Luby-Rackoff Ciphers. In Gollmann [Gol96], pages 189–203.

[MRP12] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the Xiao - Lai White-Box AES Implementation. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2012.

[MWP10] Yoni De Mulder, Brecht Wyseur, and Bart Preneel. Cryptanalysis of a Perturbated White-Box AES Implementation. In Guang Gong and Kishan Chand Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*, volume 6498 of *Lecture Notes in Computer Science*, pages 292–310. Springer, 2010.

[NR99] Moni Naor and Omer Reingold. On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited. *J. Cryptology*, 12(1):29–66, 1999.

[Nyb96] Kaisa Nyberg. Generalized Feistel Networks. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT 1996*, volume 1163 of *Lecture Notes in Computer Science*, pages 91–104. Springer, 1996.

[PNB06] Jacques Patarin, Valérie Nachef, and Côme Berbain. Generic Attacks on Unbalanced Feistel Schemes with Contracting Functions. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 396–411. Springer, 2006.

[PNB07] Jacques Patarin, Valérie Nachef, and Côme Berbain. Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 325–341. Springer, 2007.

[Riv98] Ron Rivest. *A Description of the RC2 Encryption Algorithm.* Internet Engineering Task Force (IETF), RFC 2268, 1998. http://www.ietf.org/rfc/rfc2268.txt.

[SHW$^+$14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In Sarkar and Iwata [SI14], pages 158–178.

[SI14] Palash Sarkar and Tetsu Iwata, editors. *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*. Springer, 2014.

[SK96] Bruce Schneier and John Kelsey. Unbalanced Feistel Networks and Block Cipher Design. In Gollmann [Gol96], pages 121–144.

[SM10] Tomoyasu Suzaki and Kazuhiko Minematsu. Improving the generalized feistel. In Seokhie Hong and Tetsu Iwata, editors, *FSE 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 19–39. Springer, 2010.

[SWX12] Tao Shi, Wei Wang, and Qiuliang Xu. Improved impossible differential cryptanalysis of SMS4. In *Eighth International Conference on Computational Intelligence and Security, CIS 2012, Guangzhou, China, November 17-18, 2012*, pages 492–496. IEEE Computer Society, 2012.

[SWZ10] Bozhan Su, Wenling Wu, and Wentao Zhang. Differential cryptanalysis of sms4 block cipher. Cryptology ePrint Archive, Report 2010/062, 2010. http://eprint.iacr.org/2010/062.

[TD08] Deniz Toz and Orr Dunkelman. Analysis of two attacks on reduced-round versions of the SMS4. In Liqun Chen, Mark Dermot Ryan, and Guilin Wang, editors, *ICICS 2008*, volume 5308 of *Lecture Notes in Computer Science*, pages 141–156. Springer, 2008.

[TKS08] Seokhie Hong Taehyun Kim, Jongsung Kim and Jaechul Sung. Linear and differential cryptanalysis of reduced sms4 block cipher. Cryptology ePrint Archive, Report 2008/281, 2008. http://eprint.iacr.org/2008/281.

[VNP10] Emmanuel Volte, Valérie Nachef, and Jacques Patarin. Improved Generic Attacks on Unbalanced Feistel Schemes with Expanding Functions. In Abe [Abe10], pages 94–111.

[Wan10] Gaoli Wang. Improved impossible differential cryptanalysis on sms4. In *Communications and Intelligence Information Security (ICCIIS), 2010 International Conference on*, pages 105–108, Oct 2010.

[YI13] Shingo Yanagihara and Tetsu Iwata. Improving the Permutation Layer of Type 1, Type 3, Source-Heavy, and Target-Heavy Generalized Feistel Structures. *IEICE Transactions*, 96-A(1):2–14, 2013.

[YI14] Shingo Yanagihara and Tetsu Iwata. Type 1.x generalized Feistel structures. *IEICE Transactions*, 97-A(4):952–963, 2014.

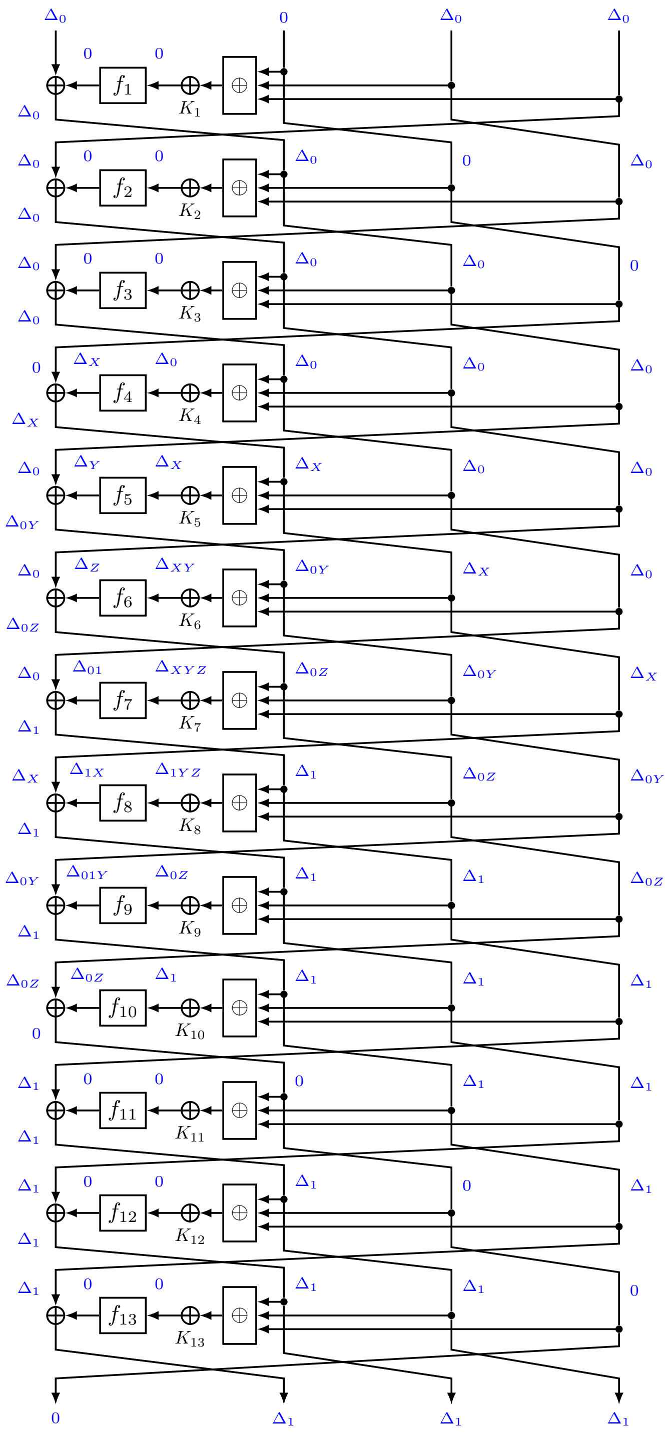
[YPL11] Aaram Yun, Je Hong Park, and Jooyoung Lee. On lai-massey and quasi-feistel ciphers. *Des. Codes Cryptography*, 58(1):45–72, 2011.

[Zhe97] Yuliang Zheng. The SPEED cipher. In Rafael Hirschfeld, editor, *Financial Cryptography 1997*, volume 1318 of *Lecture Notes in Computer Science*, pages 71–90. Springer, 1997.

[ZWFS09] Wentao Zhang, Wenling Wu, Dengguo Feng, and Bozhan Su. Some new observations on the SMS4 block cipher in the chinese WAPI standard. In Feng Bao, Hui Li, and Guilin Wang, editors, *Information Security Practice and Experience, 5th International Conference, ISPEC 2009, Xi'an, China, April 13-15, 2009, Proceedings*, volume 5451 of *Lecture Notes in Computer Science*, pages 324–335. Springer, 2009.

[ZZW08] Lei Zhang, Wentao Zhang, and Wenling Wu. Cryptanalysis of reduced-round SMS4 block cipher. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 2008*, volume 5107 of *Lecture Notes in Computer Science*, pages 216–229. Springer, 2008.

A Differential Characteristics



Δ_0 0 Δ_0 Δ_0
0 0 f_1 K_1 Δ_0
Δ_0 0 0 f_2 K_2 Δ_0 0 Δ_0 Δ_0
Δ_0 0 0 f_3 K_3 Δ_0 Δ_0 0 Δ_0
0 Δ_X Δ_0 f_4 K_4 Δ_0 Δ_0 Δ_0 Δ_X
Δ_0 Δ_Y Δ_X f_5 K_5 Δ_X Δ_0 Δ_0 Δ_{0Y}
Δ_0 Δ_Z Δ_{XY} f_6 K_6 Δ_{0Y} Δ_X Δ_0 Δ_{0Z}
Δ_0 Δ_{01} Δ_{XYZ} f_7 K_7 Δ_{0Z} Δ_{0Y} Δ_X Δ_1
Δ_X Δ_{1X} Δ_{1YZ} f_8 K_8 Δ_1 Δ_{0Z} Δ_{0Y} Δ_1
Δ_{0Y} Δ_{01Y} Δ_{0Z} f_9 K_9 Δ_1 Δ_1 Δ_{0Z} Δ_1
Δ_{0Z} Δ_{0Z} Δ_1 f_{10} K_{10} Δ_1 Δ_1 Δ_1 0
Δ_1 0 0 f_{11} K_{11} 0 Δ_1 Δ_1 Δ_1
Δ_1 0 0 f_{12} K_{12} Δ_1 0 Δ_1 Δ_1
Δ_1 0 0 f_{13} K_{13} Δ_1 Δ_1 0
0 Δ_1 Δ_1 Δ_1

Figure 7: Differential on 13 rounds used in the 16-round attack. The XOR of two differences Δ_A and Δ_B is denoted by Δ_{AB}, e.g. Δ_{0Y} in round 5 represents $\Delta_0 \oplus \Delta_Y$. This differential can be decomposed in about $2^{3n/4}$ distinct differential characteristics, parameterized by $(\Delta_X, \Delta_Y, \Delta_Z)$.

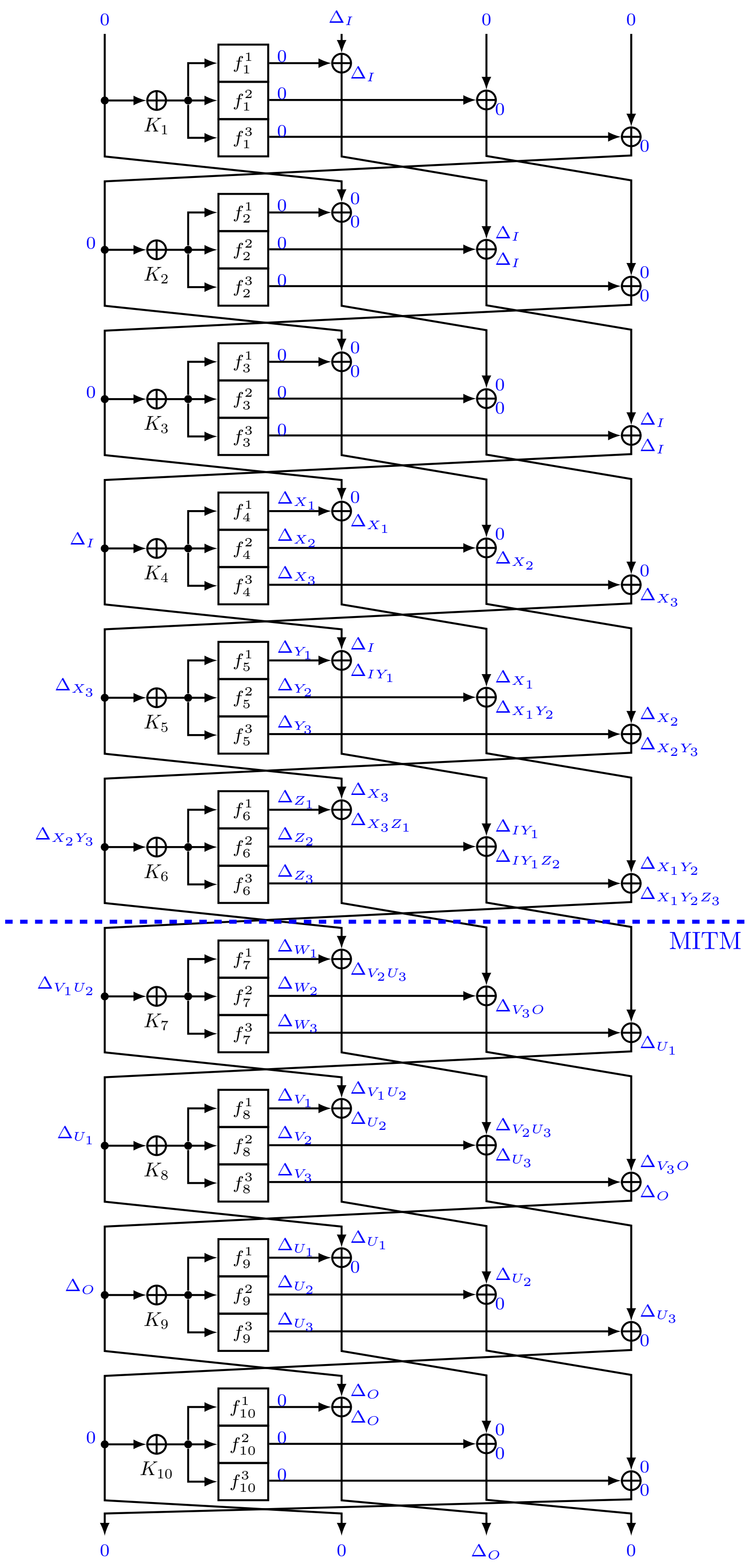


Figure 8: Differential on 10 rounds used in the 13-round attack. This differential can be decomposed in about $2^{n/2}$ distinct differential characteristics, which can be identified by the (classical) MITM approach.

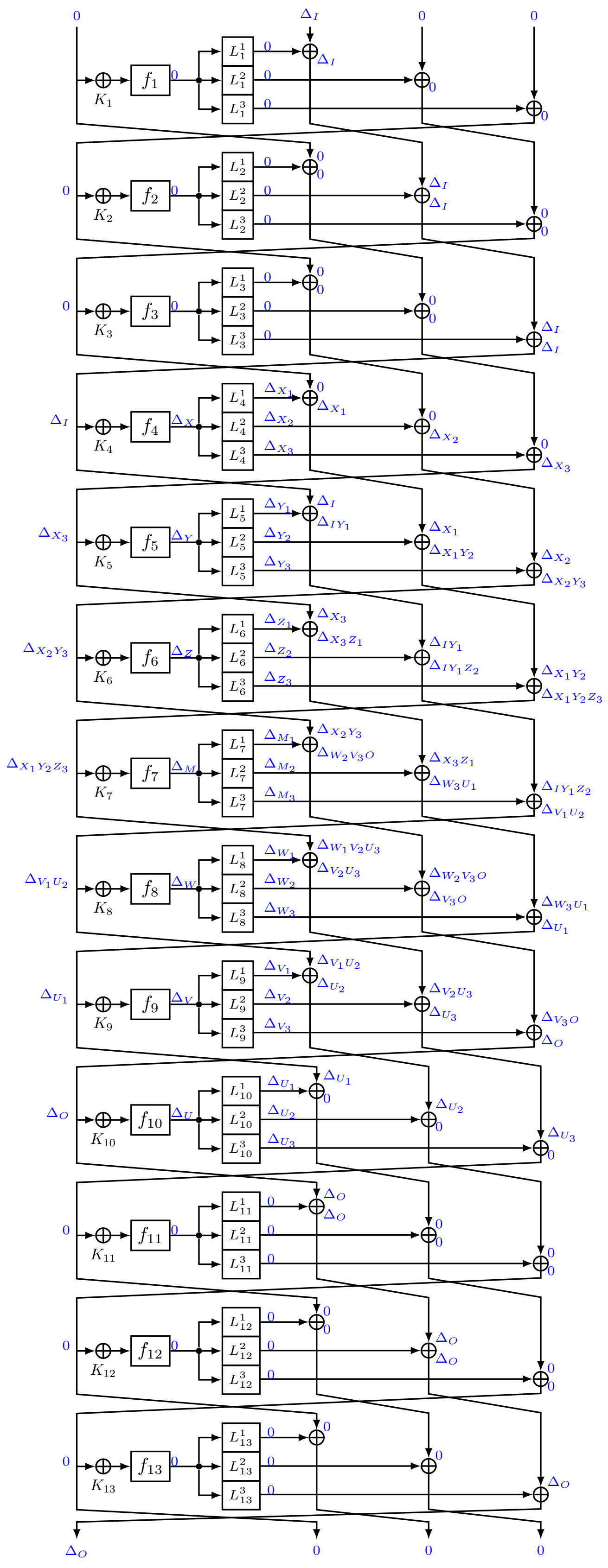


Figure 9: Differential on 13 rounds used in the 16-round attack. This differential can be decomposed in about $2^{3n/4}$ distinct differential characteristics, parameterized by $(\Delta_X, \Delta_Y, \Delta_Z)$.