Meet-in-the-Middle Attacks on Generic Feistel Constructions

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

[1] Nanyang Technological University, Singapore

[2] NTT Secure Platform Laboratories, Tokyo, Japan

ntu.guo@gmail.com, {JJean,INikolic}@ntu.edu.sg, sasaki.yu@lab.ntt.co.jp

Abstract. We show key recovery attacks on generic balanced Feistel ciphers. The analysis is based on the meet-in-the-middle technique and exploits truncated differentials that are present in the ciphers due to the Feistel construction. Depending on the type of the round function, we differentiate and show attacks on two types of Feistels. For the first type, which is the most general Feistel and has no assumption on the round function, we show a 5-round distinguisher (based on a truncated differential), which allows to launch 6- and 10-round attacks, for single-key and double-key sizes, respectively. For the second type, we assume the round function follows the SPN structure with a linear layer P that has a maximal branch number, and based on a 7-round distinguisher, we show attacks that reach up to 14 rounds. Our attacks outperform all the known attacks for any key sizes, have been experimentally verified (implemented on a regular PC), and provide new lower bounds on the number of rounds required to achieve a practical and a secure Feistel.

Key words: Feistel, generic attack, key recovery, meet-in-the-middle.

1 Introduction

A Feistel network [13] is a scheme that builds n-bit permutations from smaller, usually $n/2$-bit permutations or functions. In ciphers based on the Feistel network, both of the encryption and decryption algorithms can be achieved with the use of a single scheme, thus such ciphers exhibit an obvious implementation advantage. The Feistel-based design approach is widely trusted and has a long history of usage in block ciphers. In particular, a number of current and former international or national block cipher standards such as DES [5], Triple-DES [18], Camellia [1], CAST [4] are Feistels. In addition to the standard block ciphers, Feistel is an attractive choice for many lightweight ciphers, for instance the recent NSA proposals SIMON and SPECK [2], LBlock [24], Piccolo [22], etc. The application of the Feistel construction is not limited only to ciphers, and is used in other crypto primitives: the hash function SHAvite-3 [3], the CAESAR proposal for authentication scheme LAC [25] and others.

The analysis of Feistel primitives and their provable security bounds depend on the type of the round function implemented by the Feistel. Luby and Rackoff [20] have shown that an n-bit pseudorandom permutation can be constructed from an $n/2$-bit pseudorandom function with 3-round Feistel network. In this construction, the round functions are chosen uniformly at random from a family of $2^{n/2 \cdot 2^{n/2}}$ functions – a set that can be enumerated with $n/2 \cdot 2^{n/2}$-bit key. Later, Knudsen [19] considered a practical model, in which the round functions are chosen from a family of 2^k functions and showed a generic attack on up to 6 rounds. Knudsen's construction was coined as *Feistel-1* by Isobe and Shibutani in [17] to reflect the fact that it is the most general type of Feistels. They further introduced the term *Feistel-2* to denote ciphers in which the round functions are composed of XOR of a subkey, followed by public function or a permutation. Generic attacks on Feistel-2 such as impossible differentials [19], all-subkey recovery [16,17], and integral-like [23] penetrate up to 6 rounds when the key size equals the state size, and up to 9 rounds when the key is twice larger than the block. Better attacks were published, but they are on so-called *Feistel-3* with round functions based on the well-known substitution-permutation network (SPN), i.e. the rounds start with an XOR of a subkey, followed by a layer of S-Boxes and a linear diffusion layer. The attacks on Feistel-3 presented in [17] reach up to 7 rounds in the case of equal key and state sizes, and 11 rounds for twice larger keys.

We present attacks on Feistel-2 and Feistel-3 ciphers based on the well-known meet-in-the-middle cryptanalytic technique. Its most basic form corresponds to the textbook case of Double-DES [21] and in the past few years, a few improvements have been proposed to more specific cases, for instance, Dinur et al. [11] generalized the attack on Double-DES when multiple encryption (more than two n-bit keys) is used. A notable application of the meet-in-the-middle technique and a line of research that has been

started by Dermirci and Selçuk [7] are the attacks on the Advanced Encryption Standard (AES). They presented cryptanalysis of AES-192 and AES-256 reduced to 8 rounds by improving the collision attack due to Gilbert and Minier [14] and with the use of the meet-in-the-middle technique. Later, their strategy has been revisited by Dunkelman, Keller and Shamir [12], and most recently further improved by Derbez, Fouque and Jean [10,9]. In this advanced form, the attack combines both the classical differential attack and the meet-in-the-middle strategy. In the differential attack, a high-probability differential is used to detect statistical biases to deduce information on the last subkey used in a block cipher. Here, the attacker detects correct subkey guesses by checking meet-in-the-middle equations during the encryption process. Namely, the attack starts by fully tabulating in T a distinguishing behavior particular to the targeted cipher, e.g. AES, in a precomputation phase, and later searches for messages verifying the distinguisher by checking the precomputed table T.

Our contributions. We show the best known generic attacks on Feistel-2 and Feistel-3 cipher constructions. Our analysis, and a preliminary step of the attacks, relies on a special differential behavior of several consecutive rounds that is inherited by the generic Feistel construction. This property can be seen as a distinguisher, and for Feistel-2 it extends to 5 rounds, while for Feistel-3 to 6-7 rounds. The attacks exploit the distinguishers, and by adding rounds before, in the middle, and after the distinguisher, they can penetrate higher number of rounds. The distinguisher allows the differential behavior of the Feistel rounds to be enumerated offline and without the knowledge of the actual subkeys. This in fact is the first step of our attacks: a precomputation phase used to create a large look-up table. The next step is the collection of a sufficient number of plaintext/ciphertext pairs, some of which will comply with the conditions of the distinguisher. Each such pair suggests candidates for the round subkeys, and the look-up table is used to filter the correct subkeys. This step consists of the meet-in-the-middle part of the attack.

In the case of the Feistel-2 construction, the number of rounds of our attacks depends on the ratio of key to state sizes k/n: the larger the ratio, the more rounds we can attack. Namely, $4s+2$ rounds can be attacked for $k/n = (s+1)/2$, which translates to 6 rounds when $k = n$, 8 rounds for $k = 3n/2$, 10 for $k = 2n$, etc. As long as the ratio is increasing, the number of attacked rounds will grow. This property comes from the meet-in-the-middle nature of the attacks, i.e. when we increase the key by bit size equivalent to one Feistel branch (and thus allow the complexity of the attack to increase by this amount), then we can add one round to the distinguisher in the offline phase, and prepend one round in the online phase. Since the attack is meet-in-the-middle, the complexity of these two phases is not multiplied but simply added, hence the accumulative complexity remains below the trivial exhaustive key search. In the analysis of Feistel-2, regardless of the number of attacked rounds, we make no assumptions on the round function: it can be any invertible or one-way function or a permutation, unique for each round.

For the Feistel-3 construction and a linear diffusion layer P with maximal branch number, we can attack up to 14 rounds of the ciphers when the key is twice as large as the state ($k = 2n$), while for smaller keys we have attacks on 12 and 10 rounds, for key sizes $k = 3n/2$ and $k = n$, respectively. The above generalization (the number of attacked rounds always increases when the key size increase) is no longer possible as the data complexity grows beyond the full codebook. To reach more rounds compared to Feistel-2, we use the SPN structure of the round function in both the offline and the online stages of the attack. The best such example given in the paper is the redefinition of the Feistel-3 by moving the linear layer from one round to the surrounding rounds – this allows to extend the attack for an additional round. Other natural improvements based on the SPN structure are better (in terms of number of rounds) distinguisher and key recovery. For the main Feistel-3 attacks, we assume that the P-layers of all rounds are the same, but in case they are different, we show that the attacks can be adapted on one round less only.

Our analysis results in a recovery of the whole values (not only partial values or bytes) of certain subkeys. This is the main advantage of the attack, and by repeating it a few times (as shown in Appendix C), we can recover one by one all the subkeys and thus be able to encrypt and decrypt without the knowledge of the initial master key. Hence, the key schedule plays no role in the analysis and the attacks are in fact a full-subkey recovery. We have also experimentally confirmed the validity

Table 1: Comparison of previous results and ours for n-bit block-length, k-bit key-length and c-bit S-Box length.

Target	Condition on round functions	#rounds, complexity k = n	k = 3n/2	k = 2n	Technique	Reference
Feistel-2	bijective	5, $O(2^{3n/4})$	6, $O(2^{n})$	7, $O(2^{3n/2})$	Impossible Differential	[19]
	-	3, $O(2^{n/2})$	5, $O(2^{n})$	7, $O(2^{3n/2})$	All Subkeys Recovery	[16]
	-	5, $O(2^{n/2})$	7, $O(2^{5n/4})$	9, $O(2^{3n/2})$	All Subkeys Recovery	[17]
	bijective, identical	6, $O(2^{n/2})$	-	-	Integral-like	[23]
	-	**6,** $O(2^{3n/4})$	**8,** $O(2^{4n/3})$	**10,** $O(2^{11n/6})$	**Meet-in-the-Middle**	**Section 3**
Feistel-3	-	7, $O(2^{3n/4+c})$	9, $O(2^{n+c})$	11, $O(2^{7n/4+c})$	All Subkeys Recovery	[17]
	-	**9,**$O(2^{n/2+4c})$	**11,**$O(2^{n+4c})$	**13,**$O(2^{3n/2+4c})$	**Meet-in-the-Middle**	**Section 4**
	identical	**10,** $O(2^{n/2+4c})$	**12,** $O(2^{n+4c})$	**14,** $O(2^{3n/2+4c})$	**Meet-in-the-Middle**	**Section 4**

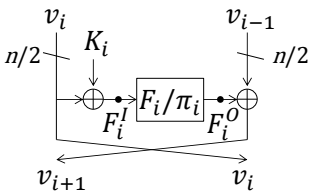

Figure 1: Feistel-2.

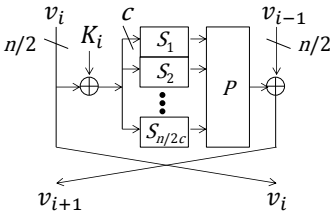

Figure 2: Feistel-3.

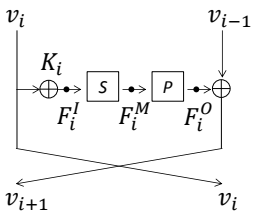

Figure 3: Simplified Feistel-3.

of our attacks in the case of small state Feistel-2[3]. The experiments ran on a regular PC supported the complexity evaluation and the correctness of the attacks. We refer the reader to Appendix D for more details. All of the results described in this paper are summarized in Table 1 and compared to the already-published generic analysis on Feistel-2 and Feistel-3.

2 Preliminaries

Throughout the paper, we assume that the block size is n bits and the Feistel is balanced, thus the branch size is $n/2$ bits. The internal state value (the branch) is denoted by v_i and the n-bit plaintext is assigned to $v_0 \| v_{-1}$. We count the rounds starting from 0, and at round i, v_{i+1} is computed as $v_{i+1} \leftarrow \texttt{RoundFunction}(v_i, v_{i-1}, K_i)$. The round function depends on the class defined further, i.e. it is either Feistel-2 or Feistel-3. In the final round, the network twist is not performed.

Generic Feistel-2 construction. A Feistel-2 round function consists of a subkey XOR and a subsequent public function as illustrated in Figure 1. Several classes of public functions can be considered. Typical classifications are bijective or non-bijective, invertible or non-invertible, and different functions for different rounds or an identical function for all rounds.

Generic Feistel-3 construction. A Feistel-3 round function consists of a subkey XOR, an S-layer, and a P-layer. The S-layer performs word-wise S-Boxes applications, while the P-layer performs a linear operation for mixing all words. Several classes of S-layers and P-layers can be considered. An example of the classification of the S-layer is different S-Boxes for different words or an identical S-Box for all words. The P-layers can be classified according to the branch number[4] of the linear transformation used in the layer. In our analysis, if c is the bit size of a word then the internal state value has $n/2c$ words, and we assume that the branch number of the linear operation in the P-layer is $n/2c+1$, i.e. it is maximal. For example, a multiplication by an MDS matrix has the maximal branch number of $n/2c+1$. The Feistel-3 construction is shown in Figure 2. We often use the simplified description given in Figure 3.

Number of solutions of differential equations. Once the input and the output differences to a particular (round) function are fixed, then we expect one value to follow such differential on average. It

[3] For Feistel-3, even when the S-Boxes have only 3 bits, the memory complexity required to launch the attacks is just above what we can afford on a regular PC.

[4] The branch number of a linear transformation is the minimum number of active/non-zero input and output words over all inputs with at least one active/non-zero word.

is important to notice that although one solution is expected, it does not mean that it can be found trivially. We repeatedly use this observation to the non-linear part of a round function (the S-Boxes).

Constructing δ-sets. Our analysis uses the similar idea of δ-set introduced by Knudsen [6].

Definition 1 (δ-set). *A δ-set for byte-oriented cipher is a set of 2^8 state values that are all different in 1 byte and are all equal in the remaining bytes.*

However we do not need byte-oriented sets. For this purpose we introduce the following definition:

Definition 2 (b-δ-set). *A b-δ-set is a set of 2^b state values that are all different in b state bits (the active bits) and are all equal in the remaining state bits (the inactive bits).*

By this definition, the original Knudsen's δ-set can be seen as an 8-δ-set, since it takes all the values of a particular byte, which is an 8-bit value. To define b-δ-set, we have to specify not only the value of b, but also the position of the active bits. In some cases, however, the position is irrelevant and the analysis is applicable for any b active bits.

Given a state value v, we can construct a b-δ-set from v, by applying $2^b - 1$ differences to some b bits of the state v. Furthermore, we can take a function F, order all the possible $2^b - 1$ input differences, and obtain a sequence of output differences of F. An example of such sequence, when the active bits are the least significant bits, is $F(v) \oplus F(v \oplus 1), F(v) \oplus F(v \oplus 2), F(v) \oplus F(v \oplus 3), \ldots F(v) \oplus F(v \oplus 2^b - 1)$.

Demirci and Selçuk attacks. In [8], Demirci and Selçuk generalized the collision attack on 7-round `Rijndael` proposed by Gilbert and Minier in 2000 in [15]. In the latter, a 3-round distinguisher is observed and used to mount a collision attack on 4 rounds, that can be extended to 7 rounds. Demirci and Selçuk extended the internal distinguisher to 4 rounds and combined it to a meet-in-middle strategy to detect the correct subkey(s). The large meet-in-the-middle table consists of all the possible functions linking one byte of an internal round to another byte four rounds later in the encryption process. In the attacks described in this paper, we apply the same technique by also storing these functions, but targeting the case of Feistel networks. We show that the high-level strategy is the same and can cover a few more rounds due to the Feistel structure. The main drawback of the Demirci and Selçuk attack is the large memory requirements to store the table. In our case, the basic attack only requires $2^{n/2}$ units of memory, which can be leveraged to trade some time and/or data for memory. All in all, the memory complexity is the bottleneck of the Demirci and Selçuk attack, whereas here, this is not the case for the basic attack: that is why we can apply straightforward time/memory/data tradeoffs.

3 Key-recovery attacks against Feistel-2 construction

The attack setting in this paper is the standard model: we assume the attacker to have no direct information about the internal state values of the block cipher, and his goal is to recover a fixed and unknown secret key. All he has is an unrestricted oracle access to the encryption and decryption algorithms (considered as black boxes) using the fixed secret key.

We start the analysis with a key-recovery attack on 6 rounds for the case when $k = n$. We then extend the attack to $(4 + 2s)$-round key recovery for the case when $k = n(s + 1)/2$ for some $s \geq 1$, i.e. 8 rounds for $k = 3n/2$, 10 rounds for $k = 2n$, etc. In the attack, the round function can be either bijective or non-bijective, i.e. a permutation or a function, and it can even be one-way. The attack given further is for the case when the round function is one-way, and different functions are used in different rounds: we make weak assumptions on the round functions to make a generic analysis on the most general type of construction.

Notations. We use F_i to denote the round function at round i of the construction. To refer to the input (resp. output) of this function F_i, we write $F_i^{\mathcal{I}}$ (resp. $F_i^{\mathcal{O}}$). Similarly, the input difference (resp. output difference) of F_i is denoted by $\Delta F_i^{\mathcal{I}}$ (resp. $\Delta F_i^{\mathcal{O}}$). Recall that the two branches, as well as the subkeys K_i, have $n/2$ bits each.

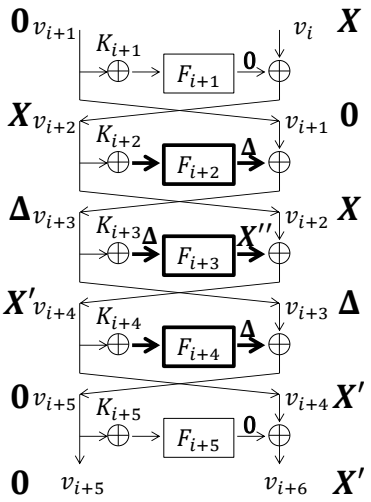

Figure 4: 5-round differential characteristic.

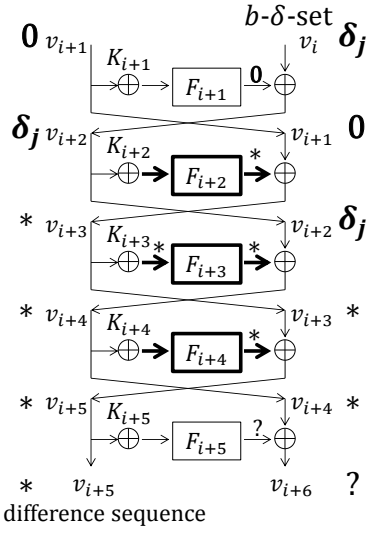

Figure 5: b-δ-set construction.

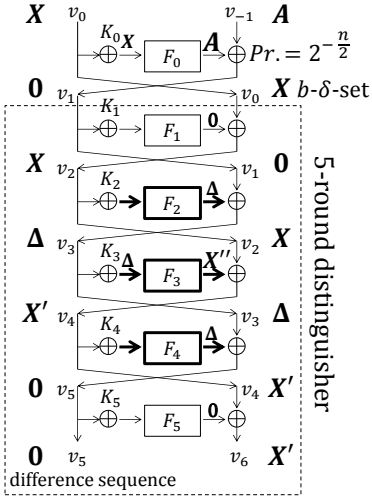

Figure 6: 6-round key-recovery.

3.1 6-round key-recovery attack for $k = n$.

The 6-round key-recovery attack is based on a non-ideal behavior of 5 rounds of Feistel-2. First, we present this 5-round distinguisher in the below lemma and proposition, and then we use it to launch a key-recovery attack on a 6-round Feistel-2 primitive where the 5 last rounds are the ones from the distinguisher. The last round of the distinguisher becomes the last round of the entire 6-round construction. Thus, we remove the last network twist in the last round of the distinguisher. Recall that in this section, we assume that the key is as large as the block size. In Lemma 1 below, for simplifying the analysis, we first assume the that each round function is differentially equally distributed. Namely, for any pair of input and output differences, the number of solutions is 1. Indeed, the average number of solutions is "1" when the input and output differences of the internal function F_i's are fixed. When the internal function F_i's are differentially strong, the maximum number of solutions can also be limited. For instance, if F_i is an optimal S-Box, the maximum is bounded by 2 or 4 solutions. After the proof of Lemma 1, more generic cases are discussed.

Lemma 1. *Let X and X' be a pair of fixed non-zero branch differences such that $X \neq X'$. If there exists an input pair (m, m') with difference $0\|X$ of 5-round Feistel-2 and output difference is $X'\|0$, then the number of possible internal state values corresponding to the input m for the middle 3 rounds is limited to $2^{n/2}$ on average.*

Proof. We note that there is $n/2$-bit round key added in each round, and hence the number of possible internal state values for the middle 3 rounds is naturally limited by $2^{3n/2}$. However, with the information on the existence of the special input/output difference, we show here the bound can be tighten to $2^{n/2}$ by analyzing the differential behavior round by round.

The 5-round differential characteristic from round $i+1$ to $i+5$ is depicted in Figure 4. The input difference $(0, X)$ becomes $(X, 0)$ after the first round with probability 1. Similarly, the output difference $(0, X')$ becomes $(0, X')$ after the inversion of the last round with probability 1. This makes $\Delta F^{\mathcal{O}}_{i+3}$ to be $X'' \leftarrow X \oplus X'$. Since $X \neq X'$, we deduce that $X'' \neq 0$ and therefore $\Delta F^{\mathcal{I}}_{i+3} \neq 0$ with probability 1. Let us denote with Δ the nonzero value of $\Delta F^{\mathcal{I}}_{i+3}$. It means that both $\Delta F^{\mathcal{O}}_{i+2}$ and $\Delta F^{\mathcal{O}}_{i+4}$ also have the value Δ. For each Δ, the input and output differences of the round function in rounds $i+2$, $i+3$, and $i+4$ are fixed, and we expect one state value on average to match that differential. As a consequence, the internal state values in these rounds are fixed. Since Δ can assume at most $2^{n/2}$ different values, the state values in rounds $i+2$, $i+3$, $i+4$ can assume $2^{n/2}$ possibilities. In Figure 4, the fixed value for each Δ is drawn by bold line. □

Remark 1. Suppose that each round function is not differentially equally distributed. More precisely, probabilities that randomly given input and output differences $(\Delta I, \Delta O)$ have solutions for each of three middle rounds are $\Pr[\Delta I \xrightarrow{F_2} \Delta O] = 1/p_1$, $\Pr[\Delta I \xrightarrow{F_3} \Delta O] = 1/p_2$, and $\Pr[\Delta I \xrightarrow{F_4} \Delta O] = 1/p_3$, where

$p_1 \cdot p_2 \cdot p_3 < 2^{n/2}$. Also suppose that, once the differential equation is satisfied, the number solutions for each round function is p_1, p_2, and p_3. Then, assume that each round function can be evaluated independently. This situation is very likely in practice, and Lemma 1 can still hold.

When $p_1 \cdot p_2 \cdot p_3$ values of Δ are chosen, one of them will satisfy the characteristic on average. Then, $p_1 \cdot p_2 \cdot p_3$ internal state values are obtained for the middle three rounds. After examining $2^{n/2}$ choices of Δ, $2^{n/2}/(p_1 \cdot p_2 \cdot p_3)$ choices have solutions and the number of solutions is $2^{n/2} \cdot (p_1 \cdot p_2 \cdot p_3)/(p_1 \cdot p_2 \cdot p_3) = 2^{n/2}$.

If the assumption of the independence of different rounds is removed, we need to evaluate $\Pr[X \xrightarrow{F_2} \Delta]$, $\Pr[\Delta \xrightarrow{F_3} X \oplus X']$, and $\Pr[X' \xrightarrow{F_4} \Delta]$, for a given fixed pair of (X, X'). When the round functions are extremely biased e.g. linear, the number of solutions can be 0 i.e. the characteristic is impossible. However, such a case is usually analyzed in a different method e.g. characteristic with probability 1 is generated for any number of rounds. Thus, we do not discuss in this direction.

For simplicity, hereafter we only describe the analysis for differentially equally distributed functions.

From Lemma 1, we obtain the following proposition.

Proposition 1. *There exists a message pair (m, m') conforms to the 5-round differential characteristic given in Figure 4. Let us denote the 5-round Feistel-2 as a function F, and a new function F^Δ as $F^\Delta(\delta_j) = Trunc_{n/2}(F(m) \oplus F(m \oplus (0\|\delta_j)))$ where $Trunc_{n/2}$ denotes a truncation of the first $n/2$ bits, i.e., the left branch of the output. Then, we show the number of such unique F^Δ is limited by $2^{n/2}$ on average, in other words the sequence of differences Δv_{i+5} obtained from the b-δ-set constructed from m in v_i can assume $2^{n/2}$ values on average.*

Remark 2. The b-δ-set assumes b bits out of n/2-bit input of F^Δ δ_j, and we leave the choice of bits later. From a theoretical point of view, the number of F^Δ from a set of 2^b elements to a set of $2^{n/2}$ elements amounts to $(2^{n/2})^{2^b} = 2^{2^b n/2}$. This is much larger than $2^{n/2}$ of the Feistel-2 case as long as $b \geq 1$.

Proof. Generally speaking, we are given the information that some message pair follows the 5-round differential characteristic. Then we define a function F^Δ with difference of v_i as input and difference of v_{i+5} as output, and tries to count how many such F^Δ there are.

From Lemma 1, the difference Δ assumes at most $2^{n/2}$ values, and for each of them the corresponding internal state values, i.e. the actual input and output values of $F_{i+2}, F_{i+3}, F_{i+4}$, are fixed. We use this fact to show further that for each fixed Δ and the corresponding internal state values, the output Δv_{i+5} can be computed uniquely from the input Δv_i (refer to Figure 5). In other words, when Δ is fixed, the function F^Δ from Δv_i to Δv_{i+5} is deterministic, hence the number of such function is limited by the number of possible Δ.

Here, we show how Δv_{i+5} can be computed from Δv_i uniquely when Δ is fixed. Assume now we have a message m conforming to the 5-round characteristic, i.e., the output difference corresponding to the input pair $(m, m \oplus (0\|X))$ is $0\|X'$. Furthermore, let $t_{i+2}, t_{i+3}, t_{i+4}$ be the corresponding input values to $F_{i+2}, F_{i+3}, F_{i+4}$ for processing the message m, in which $t_{i+2}, t_{i+3}, t_{i+4}$ are determined depending on Δ. We now consider a new input value pair, $(m, m \oplus (0\|\delta_j))$, by introducing a difference δ_j to the right branch. Since the difference $\Delta F^{\mathcal{O}}_{i+1}$ is always 0, we obtain that $\Delta v_{i+2} = \Delta v_i = \delta_j$. In round $i+2$, the attacker knows the value of $F^{\mathcal{I}}_{i+2} = t_{i+2}$ and difference $\Delta F^{\mathcal{I}}_{i+2} = \delta_j$. Hence, the new paired values of $F^{\mathcal{I}}_{i+2}$ are t_{i+2} and $t_{i+2} \oplus \delta_j$. Therefore, the new $\Delta F^{\mathcal{O}}_{i+2}$ can be obtained as $\Delta F^{\mathcal{O}}_{i+2} \leftarrow F_{i+2}(t_{i+2}) \oplus F_{i+2}(t_{i+2} \oplus \delta_j)$. In Figure 5, we represent this type of unfixed but computable difference by the attacker as '$*$'. The new difference for $\Delta F^{\mathcal{O}}_{i+2}$ is propagated forward to v_{i+3} and the same reasoning as in round $i+2$ is applied to round $i+3$. The attacker knows the value of $F^{\mathcal{I}}_{i+3} = t_{i+3}$ and $\Delta F^{\mathcal{I}}_{i+3} = \Delta F^{\mathcal{O}}_{i+2}$. Hence $(t_{i+3}, t_{i+3} \oplus \Delta F^{\mathcal{O}}_{i+2})$ are the paired values. The new $\Delta F^{\mathcal{O}}_{i+3}$ can therefore be computed as $\Delta F^{\mathcal{O}}_{i+3} \leftarrow F_{i+3}(t_{i+3}) \oplus F_{i+3}(t_{i+3} \oplus \Delta F^{\mathcal{O}}_{i+2})$. The knowledge of $\Delta F^{\mathcal{O}}_{i+3}$ gives the difference for v_{i+4} for the next round, namely: $\Delta v_{i+4} \leftarrow \Delta F^{\mathcal{O}}_{i+3} \oplus \delta_j$. The analysis continues the same way for round $i+4$. From the knowledge of the value of $F^{\mathcal{O}}_{i+4} = t_{i+4}$ and the new difference $\Delta F^{\mathcal{O}}_{i+4} = \Delta v_{i+4}$, the output difference of the round function $\Delta F^{\mathcal{O}}_{i+4}$ is computed, and finally Δv_{i+5} is computed as $\Delta F^{\mathcal{O}}_{i+4} \oplus \Delta v_{i+3} = \Delta F^{\mathcal{O}}_{i+4} \oplus \Delta F^{\mathcal{O}}_{i+2}$.

In summary, for each of δ_j we can compute the output difference Δv_{i+5}, i.e., the mapping from δ_j to Δv_{i+5} becomes deterministic. Therefore for the ordered sequence of δ_j taking values of $1, 2, \ldots, 2^{n/2} - 1$, we can determine the sequence of corresponding differences Δv_{i+5}, i.e., the b-δ-set. It is important to note that the mapping depends only on values of $t_{i+2}, t_{i+3}, t_{i+4}$, which are determined from the value of

Δ, X and X', and acts independently from the value of m. Since Δ can take at most $2^{n/2}$ values, the number of such sequences of Δv_{i+5} is also limited to $2^{n/2}$. We will use the Δv_{i+5} sequences to deduce the value of Δ, hence $t_{i+2}, t_{i+3}, t_{i+4}$ and other intermediate state values later in our attacks. □

We note that the Proposition 1 cannot be extended to more rounds. Indeed, the original paired values in $F^{\mathcal{I}}_{i+5}$ are not known to the attacker, so that the corresponding $\Delta F^{\mathcal{O}}_{i+5}$ cannot be computed. In Figure 5, uncomputable differences are represented as '?'.

6-round key-recovery attack. We prepend one round to the 5-round distinguisher shown in Figure 4 and the resulting construction is illustrated in Figure 6. The attack consists of precomputation and online phases, and the online phase is further divided into collecting pair and key recovery phases. First, we fix X and choose multiple values of X'. For each pair of (X, X'), we precompute and store all sequences of Δv_5 based on Proposition 1 in a large table together with its corresponding internal state values. Next, we collect many pairs satisfying the differential characteristic. Finally, for each of obtained pairs, we compute Δv_5 sequences with guessing the first round key K_0. We then find a match of Δv_5 sequences between the precomputed table and the one computed online, with which we are able to determine the internal states and recover K_0. (The name of MitM derives from the fact that Δv_5 sequence is computed offline for the last 5 rounds and online for the first round, and the results are later matched efficiently. Offline and online computations take roles of subfunctions in the MitM framework.)

Precomputation. From Proposition 1, the number of possible sequences of Δv_5 is $2^{n/2}$ for a fixed X and a fixed X'. In fact, we can achieve a time/memory tradeoff by relaxing the $n/2$-bit constraint of a fixed X' and by allowing $2^{x'}$ different possible differences for X', where $0 \leq x' \leq n/2$. Without loss of generality, assume that the values of X' differ in the last x' bits and are the same in the remaining $n/2 - x'$ most significant bits. In the sequel, we will determine the optimal value for x' to reach the best time/data/memory complexities for the attack.

First, we show how we can compute all the $2^{x'} \cdot 2^{n/2} = 2^{x'+n/2}$ sequences of 2^b differences as an offline precomputation in time less than $2^{x'+n/2+b}$ encryptions, and memory for $2^{x'+n/2+b}$ blocks of $n/2$ bits. This offline precomputation results in a table T_δ, that contains all the sequences. Since the precomputation step is the same for all X' differences, further we show the procedure for a particular X' and assume that for the whole offline execution this procedure is repeated $2^{x'}$ times for the possible values of X' differences.

In rounds 2 and 4, the input differences to the round functions are fixed to X and X', respectively, while both of the output difference are Δ. To reduce the time complexity, we first tabulate completely the round functions F_2, F_3 and F_4 in order to have constant-time access to paired values for some input or output differences. Namely, we construct precomputation tables T_2 and T_4, which take the difference Δ as input and return the paired values conforming to the differentials $X \to \Delta$ and $X' \to \Delta$ through F_2 and F_4, respectively. The strategy consists simply in iterating over all possible inputs, and storing the results indexed by output difference as described in Algorithm 1.

Similarly, in round 3 we want to construct the table T_3 that gives in constant time a paired-value input to F_3 resulting in the fixed output difference X''. However, since the function F_3 is assumed to be one-way and we impose a constraint on its output, we cannot compute F_3^{-1} to construct T_3. Consequently, we first need to evaluate F_3 for all input values, store the values in a temporary table, and later consider the difference, as detailed in Algorithm 2. After this part of the precomputation phase, for an arbitrary fixed difference Δ (which is the difference $\Delta F^{\mathcal{O}}_2 = \Delta F^{\mathcal{I}}_3 = \Delta F^{\mathcal{O}}_4$), the corresponding state values in rounds 2, 3, and 4 can be looked up in tables T_2, T_3, and T_4 in constant time. Hence, we can compute the b-δ-set for all the $2^{n/2}$ possible choices of Δ and store the resulting sequences in the precomputation table T_δ, which later is used for the meet-in-the-middle check of the online phase. This step is described in Algorithm 3.

Finally, another table T_0 of size $2^{n/2}$ is generated to make more efficient the online phase and the recovery of the subkey K_0. That is, in round 0, for all values of $F^{\mathcal{I}}_0$, the corresponding $\Delta F^{\mathcal{O}}_0$ is computed. Namely, for $i = 0, 1, \ldots, 2^{n/2} - 1$, $F_0(i) \oplus F_0(i \oplus X)$ is computed and stored in T_0.

As stated previously, we repeat this procedure for $2^{x'}$ different choices of the difference X'. For the sake of simplicity, the resulting tables for each X' are all merged in the same table T_δ. For a fixed choice of X', to build T_0, T_2, T_3 and T_4 requires $2^{n/2}$ round function computations for each and hence to build

Algorithm 1: Construction of the tables T_2 and T_4.

for $i = 0, 1, \ldots, 2^{n/2} - 1$ **do**
  Compute $\Delta F_2^{\mathcal{O}} \leftarrow F_2(i) \oplus F_2(i \oplus X)$.
  Store $(i, \Delta F_2^{\mathcal{O}})$ in T_2 indexed by $\Delta F_2^{\mathcal{O}}$.
  Compute $\Delta F_4^{\mathcal{O}} \leftarrow F_4(i) \oplus F_4(i \oplus X')$
  Store $(i, \Delta F_4^{\mathcal{O}})$ in T_4 indexed by $\Delta F_4^{\mathcal{O}}$.

Algorithm 2: Construction of the table T_3.

for $i = 0, 1, \ldots, 2^{n/2} - 1$ **do**
  Store $(i, F_3(i))$ in a temporal table `tmp` indexed by $F_3(i)$.
for $i = 0, 1, \ldots, 2^{n/2} - 1$ **do**
  Compute $F_3(i) \oplus X''$.
  Look up `tmp` to obtain j such that $F_3(j) = F_3(i) \oplus X''$.
  Store $(i, i \oplus j)$ in T_3 indexed by $i \oplus j$.

Algorithm 3: Construction of the sequences of Δv_5.

for $\Delta = 1, \ldots, 2^{n/2} - 1$ **do**
  Obtain internal state values $F_2^{\mathcal{I}}$, $F_3^{\mathcal{I}}$ and $F_4^{\mathcal{I}}$ by looking up T_2, T_3 and T_4, respectively.
  for all b active bits of the b-δ-set **do**
    Modify Δv_0, and compute the corresponding Δv_5.
  Compute the sequence of Δv_5 and add it to T_δ.

Algorithm 4: Data collection phase of the 6-round attack.

Choose $2^{x'}$ differences X' so that the $n/2 - x'$ most significant bits of X' are 0 for all X'.
Choose a difference X such that $X \neq X'$.
for $2^{n/2-x'}$ different values of v_0 **do**
  for all $2^{n/2}$ choices of v_{-1} **do**
    Query (v_0, v_{-1}) and store it in a list L_0 sorted by the ciphertext value.
    Query $(v_0 \oplus X, v_{-1})$ and store it in a list L_1 sorted by the ciphertext value.
    Pick up the elements of $L_0 \times L_1$ whose ciphertexts match in the $n - x'$ most significant bits.

T_δ requires less than $2^b \cdot 2^{n/2}$ encryptions. The entire analysis is iterated over $2^{x'}$ choices of X' so that the computational cost is less than $2^{x'+b+n/2}$ encryptions. The memory requirement to construct T_0, T_2, T_3 and T_4 is $2^{n/2}$ blocks of $n/2$ bits, and is constant as we can reuse the memory across different X'. The size of T_δ increases with the iteration of $2^{x'}$ choices of X', namely, the memory requirement for the precomputation phase amounts to $2^b \cdot 2^{x'+n/2} = 2^{x'+n/2+b}$ blocks of $n/2$ bits.

Collecting pairs. In the data collection phase, we query the encryption oracle with chosen plaintexts to get enough pairs such that one conforms to the whole 6-round differential characteristic. To do so, we start by constructing a structure of $2^{n/2+1}$ plaintexts that consists of two lists of sizes $2^{n/2}$. All the elements of the first list are constant to a fixed random value v_0 on their left half (branch), while the right halves are pairwise distinct. The second list is constructed similarly, except that the left half is fixed to $v_0 \oplus X$. As a result, we can construct 2^n pairs of plaintexts such that the difference in the left branch equals X and the right one is nonzero.

For a single structure, the data complexity therefore corresponds to the encryption of $2^{n/2+1}$ chosen plaintexts, which can subsequently be sorted by their ciphertext values to detect the pairs that match on their left half ($n/2$ bits) and $n/2 - x'$ right most significant bits. Consequently, we expect one structure of plaintexts to provide $2^n / 2^{n/2+n/2-x'} = 2^{x'}$ pairs conforming to the truncated output difference, i.e. such that only the x' less significant bits of the right half are nonzero. To complete the attack, we need $2^{n/2}$ pairs, as the difference cancellation at the output of the first round holds with probability $2^{-n/2}$. Hence by repeating the data collection for $2^{n/2-x'}$ different values of v_0, we can expect one pair among the $2^{n/2}$ to follow the whole characteristic. Therefore, the data complexity amounts to $2^{n/2-x'} \times 2^{n/2+1} = 2^{n-x'+1}$ chosen plaintexts, requires the same amount of memory access as time complexity to be generated, and can be stored using only $2^{n/2}$ elements by using a hash table for the pairs that verify the truncated output difference. The whole procedure is described in Algorithm 4.

Recovery of K_0. The previous phase results in $2^{n/2}$ candidate pairs with a plaintext difference $(X, \Delta v_{-1})$ and an appropriate ciphertext difference. For each pair, we match against the precomputed

table T_0 to find the corresponding value of $F_0^{\mathcal{I}}$, and thus determine uniquely a subkey candidate for K_0 by $K_0 \leftarrow v_0 \oplus F_0^{\mathcal{I}}$.

However, among these $2^{n/2}$ candidates for K_0, only one is correct while the remaining are false positives. To find the correct subkey, we use the results of Proposition 1 and the precomputation table T_δ, i.e. we construct a b-δ-set by modifying the active bits of v_0. For each modified plaintext, with the knowledge of K_0, we compute the corresponding $F_0^{\mathcal{O}}$ and modify v_{-1} so that the value of v_1 stays unchanged. Then, we query the plaintexts and observe the left half of the corresponding ciphertexts. Hence, we can compute the sequence of Δv_5 and if the sequence is included in the precomputation table T_δ, K_0 is the correct guess with high probability, otherwise it is wrong. We note that this does not increase the data complexity, since the structures of plaintexts already includes the plaintexts for the b-δ-set evaluation.

Complexity analysis. In the online phase of the attack, we perform $2^{n/2}$ checks in the precomputed table T_δ that contains all the possible stored sequences of differences. If we do not store enough information in this table (if b is too small), many checks will wrongly yield to valid subkey candidates K_0. On the other hand, if we store too much information (if b is too large), the table will require higher time and memory complexity to be constructed. Thus, we need to select an optimal value of b. One check yields a false positive with probability $2^{n/2}/2^{n2^b/2} = 2^{n(1-2^b)/2}$ as there are $2^{n/2}$ valid sequences of 2^b elements among the $2^{n2^b/2}$ theoretically possible ones. Therefore, we want $n(1-2^b)/2 + n/2 < 0$ so that among all the $2^{n/2}$ checks, only the correct K_0 results in a stored element, and thus $b \geq 2$.

In terms of tradeoff, adjusting the value x' balances the data, time and memory complexities. Indeed, the data complexity corresponds to the encryption of $2^{n-x'+1}$ chosen plaintexts, the time complexity is $2^{x'+n/2}$ encryptions to construct T_δ and $2^{n-x'+1}$ memory access to query the encryption oracle. The memory complexity is also $2^{x'+n/2}$ blocks of $n/2$ bits required to store T_δ. Consequently, by choosing $x' = n/4$ makes the data complexity to become about $2^{3n/4}$ chosen plaintexts, the time complexity equivalent to about $2^{3n/4}$ encryptions and the memory also $2^{3n/4}$ blocks of $n/2$ bits to store all the sequences of $b = 2$ differences in T_δ.

3.2 Extension of the key-recovery attack on Feistel-2.

In this section, we extend the previous attack on 6 rounds to $4 + 2s$ rounds, when the key size k is related to the block size n by $k = n(s+1)/2$. Namely, we construct a $(4+s)$-round distinguisher that covers the last rounds of the construction, and we prepend an additional s rounds at the beginning that we control probabilistically in the online phase. As the case $s = 1$ has been treated before, we assume further that $s \geq 2$.

The distinguisher on $4 + s$ rounds works exactly the same as the one presented before for 5 rounds when $s = 1$, expect that we add additional rounds right in the middle of the characteristic, and we guess completely the fully active $n/2$-bit extra differences. See Figure 7 for the illustration. Recursively, the distinguisher for $4 + s + 1$ rounds uses the same strategy as the one on $4 + s$ rounds, except that the differential characteristic includes on more fully active round in the middle, and we guess the corresponding $n/2$-bit difference. Consequently, the precomputation phase for $4 + s + 1$ rounds requires $\mathcal{T}_{4+s+1} = 2^{n/2} \times \mathcal{T}_{4+s}$ computations, where $\mathcal{T}_{4+s}$ is the time complexity for the $(4+s)$-round distinguisher (recall that in previous section $\mathcal{T}_5 = 2^{n/2+x'}$). The memory complexity follows a similar formula as the time complexity. Therefore, the precomputation phase constructs the meet-in-the-middle table T_δ in time (number of encryptions) and memory (blocks of $n/2$ bits) complexities $\mathcal{T}_{4+s+1} = 2^{ns/2+x'}$, where $x' \in [0, n/2]$ still represents the number of nonzero bits at the output difference X' of the distinguisher.

In the online phase, we prepend s rounds to the differential characteristic, which destroy the partially known input difference in the plaintext pairs (in the previous attack, the left halves of the plaintext pairs had a difference fixed to X). Indeed, now, by going backwards one (or more) round, the plaintext difference becomes completely unknown. Therefore, we use a fully truncated input plaintext difference in both halves, and not only in the right half. The main difference compared to the case $s = 1$ is that now we need to guess the $s - 2$ differences in the left halves of the first s rounds (the remaining can be determined from the plaintext difference and from the starting difference of the distinguisher) to be able to construct the b-δ-set when $s \geq 2$. See Figure 8 for the illustration.

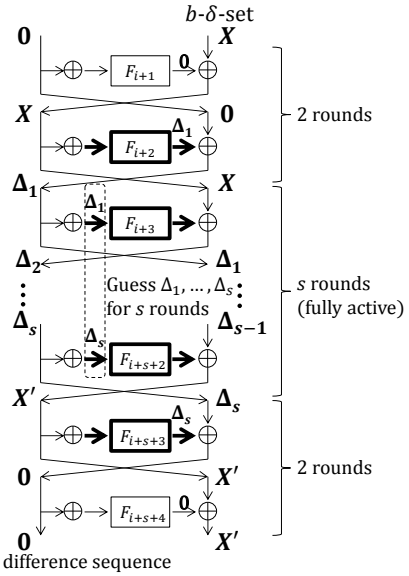

Figure 7: $4 + s$ round differential characteristic.

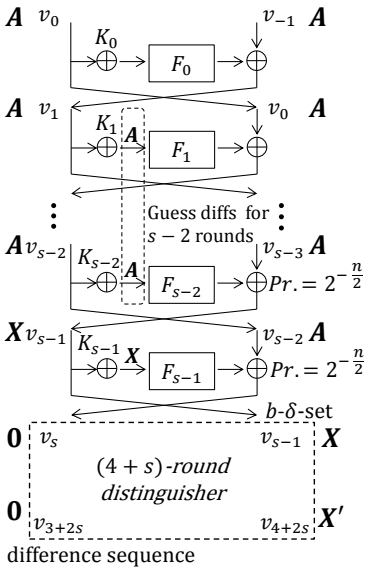

Figure 8: $4 + 2s$ round key recovery.

The data collection procedures rely on the same arguments as before: we construct a set of chosen-plaintext pairs such that one verifies the full $(4 + 2s)$-round characteristic (s rounds at the beginning, and $4 + s$ rounds in the distinguisher). To do so, we simply choose 2^d random n-bit plaintexts, $0 < d < n$, and query their encryption to the oracle. We then sort the corresponding ciphertexts by the $n - x'$ inactive bits, as in the previous attack, and finally construct $2^{2d-n+x'}$ pairs that have the correct output difference. The value d depends on the number of pairs needed to finish the attack. In the s first rounds, the probability that a random plaintext pair is transformed into $(0, X)$ after s rounds is 2^{-n}, which is the probability that a zero difference occurs in the s-th round and the difference X occurs in the $(s-1)$-th round. The attacker therefore needs 2^n pairs, which means that d has to verify $d = n - x'/2$. The data complexity therefore corresponds to the encryption of $2^{n-x'/2}$ chosen plaintexts.

For each of the 2^n pairs, and for each $2^{(s-2)n/2}$ value of the $s-2$ guessed differences, we get one subkey candidate for each of the first s subkeys. As in the previous attack (construction of the table T_0), the precomputation phase can include small computations to tabulate the differential behavior of the s first round functions so that the subkey candidates are obtained in constant time. Then, with the knowledge of the subkeys $K_0, \ldots, K_{s-1}$ and the paired input values to each $F_0, \ldots, F_{s-1}$, the attacker constructs the b-δ-set at the end of the s-th round to check the pair and the key candidates for the distinguisher using T_δ.

Complexity analysis. In total, the attack performs $2^n \times 2^{(s-2)n/2} = 2^{ns/2}$ checks in the table. Let us evaluate the number of false positive to assign a value to b. As the table contains $2^{(2s+1)n/4}$ sequences, we want $ns/2 + (2s+1)n/4 - n2^b/2 < 0$ so that only the correct subkey candidates passes the check in the table. This condition is equivalent to $b > \log_2(4s+1) - 1$, b being an integer. For instance, for $s = 1$, we find $b \geq 2$ again, for $s = 2$ or $s = 3$, $b \geq 3$, etc.

We now discuss the tradeoff parameter $x' \in [0, n/2]$ and select it to balance the time/memory/data complexities. The time complexity is equivalent to $2^{ns/2+x'} + 2^{n-x'/2} + 2^{ns/2}$ for the number of encryptions required in the precomputation, the memory access for the data generation and the number of encryptions for the online phase, respectively. The memory complexity consists of the table T_δ and the storage of the plaintext pairs, that is $2^{x'+ns/2} + 2^{ns/2}$. Finally, the data complexity corresponds to $2^{n-x'/2}$ chosen plaintexts. Selecting $x' = 0$ would result in encrypting the full codebook and $x' = n/2$ corresponds to the exhaustive key search, so we choose x' in between that minimizes the overall complexity. Since $k = (s+1)n/2$, we have the time (number of encryptions) and memory (blocks of $n/2$ bits) complexities equal to $2^{k-(n/2-x')}$ and the data complexity is $2^{n-x'/2}$ chosen plaintext. So we want $n/2 - x' = x'/2$, which gives $x' = n/3$. The time complexity therefore becomes equivalent to $2^{k-n/6}$ encryptions, the data complexity is $2^{n-n/6} = 2^{5n/6}$ chosen plaintexts, and the memory complexity amounts to $2^{k-n/6}$, $s \geq 2$.

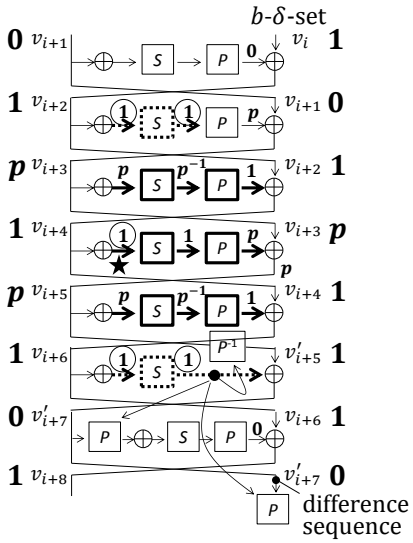

Figure 9: 7-round differential.

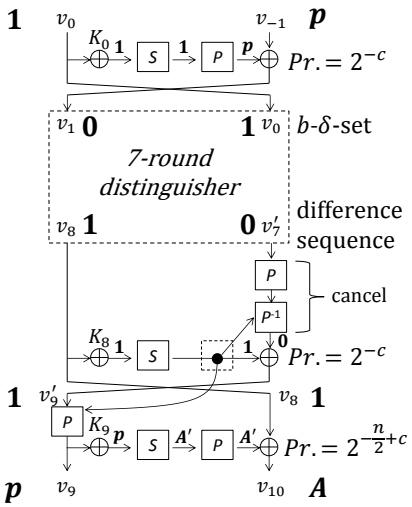

Figure 10: 10-round key-recovery for $k = n$.

4 Key-recovery attacks against Feistel-3 construction

In this section, we first present a 10-round key-recovery attack on the Feistel-3 construction with $k = n$. Then, we extend the attack to 12 and 14 rounds for key sizes of $k = 3n/2$ and $k = 2n$, respectively. Due to space constraints, these extended attacks are reported in Appendix A. We assume that different S-Boxes are used for different words in a given round, but we consider they are the same among all rounds. Recall that all the S-Boxes operate on c-bit words, and thus there are $\frac{n}{2c}$ words per branch. We consider that the P-layer is identical for all rounds and it has the maximal branch number of $\frac{n}{2c} + 1$. The analysis extends to a class of P-layers that not necessarily have a maximal branch number, as shown in Appendix B.

Notations. As in the previous section, $F_i^{\mathcal{I}}$ and $\Delta F_i^{\mathcal{I}}$ denote the input value and input difference of the i-th round, respectively, that is the input to the S-layer in F_i. Similarly, $F_i^{\mathcal{M}}$ and $\Delta F_i^{\mathcal{M}}$ refer to the state value and state difference after the S-layer, that is between the S-layer and P-layer of F_i, and $F_i^{\mathcal{O}}$ and $\Delta F_i^{\mathcal{O}}$ denote the output value and output difference of the P-layer in F_i, respectively. For the branch-wise difference, we use $\mathbf{0}$ to refer to branch with no active words, $\mathbf{1}$ to the case when only a single pre-specified word is active, and $\mathcal{P}$ and $\mathcal{P}^{-1}$ for branch-wise differences obtained after $\mathbf{1}$ has been processed by P and P^{-1}, respectively. Lastly, $X[\mathbf{1}]$ and $\Delta X[\mathbf{1}]$ respectively denote the pre-specified active-word value and difference of a branch-wise variable X.

4.1 10-round key-recovery attack for identical round functions

The 10-round key-recovery attack is based on a non-ideal behavior of 7 rounds of Feistel-3. We first present the 7-round distinguisher in the proposition below, and then use it to launch a key-recovery attack on a 10-round Feistel-3 primitive where the inner rounds are the ones from the distinguisher. Recall that in this section, we assume the key to be as large as the block size, i.e. $k = n$, and all the round functions to be equal. Considering that several rounds are later appended after the distinguisher, the last network twist is included in the 7-round distinguisher.

To construct the distinguisher, we first apply an equivalent transformation to the 7-round primitive, as shown in Figure 9. Namely, the P-layer of round $i + 6$ is removed from this round, and linear transformations are added to three different positions in order to obtain a primitive that is computationally equivalent to the original. Hereafter, v'_{i+7} represents the value of $P^{-1}(v_{i+7})$. We use the non-ideal behavior of the new representation to mount the 10-round key recovery attack by extending the 7-round differential by one round at the beginning and two rounds at the end. The newly-introduced P after v_{i+7} is later addressed in the key-recovery part.

Proposition 2. *If a message m belongs to a pair of states conforming to the 7-round differential $(\boldsymbol{0}, \boldsymbol{1}) \xrightarrow{7R} (\boldsymbol{1}, \boldsymbol{0})$ shown in Figure 9, then the sequence of differences $\Delta v'_{i+7}[\boldsymbol{1}]$ obtained from the b-δ-set, $b < c$, constructed from m in v_i can take only $2^{n/2+4c}$ values.*

Proof. We show here that the number of internal state values for pairs satisfying the 7-round differential in Figure 9 is at most $2^{n/2+4c}$. Namely, we show they can be parameterized by five nonzero differences in five c-bit words (marked by circles in Figure 9), and by the values of $n/2-c$ inactive bits of $F^{\mathcal{I}}_{i+4}$ (marked by a star '★' in Figure 9).

We first assume that the five word differences circled in Figure 9 are fixed, that is: $\Delta F^{\mathcal{I}}_{i+2}$, $\Delta F^{\mathcal{M}}_{i+2}$, $\Delta F^{\mathcal{I}}_{i+4}$, $\Delta F^{\mathcal{I}}_{i+6}$ and $\Delta F^{\mathcal{M}}_{i+6}$ are fixed to random nonzero values. When $\Delta F^{\mathcal{I}}_{i+2}$ and $\Delta F^{\mathcal{M}}_{i+2}$ are fixed, we expect one value on average to be determined for $F^{\mathcal{I}}_{i+2}[\mathbf{1}]$. In Figure 9, the state in which the value is fixed only in one word is represented by dotted lines. Then, the corresponding $\Delta F^{\mathcal{O}}_{i+2} = \Delta v_{i+3} = \Delta F^{\mathcal{I}}_{i+3}$ can be fully computed linearly by $P(\Delta F^{\mathcal{M}}_{i+2})$. Since the branch number of P is $n/2c+1$, $P(\Delta F^{\mathcal{M}}_{i+2})$ is fully active. Similarly, when $\Delta F^{\mathcal{I}}_{i+6}$ and $\Delta F^{\mathcal{M}}_{i+6}$ are fixed, one value on average can be determined for $F^{\mathcal{I}}_{i+6}[\mathbf{1}]$, and the corresponding fully active difference $\Delta v_{i+5} = \Delta F^{\mathcal{I}}_{i+5}$ can also be computed linearly by $P(\Delta F^{\mathcal{M}}_{i+6})$. Then, $\Delta F^{\mathcal{O}}_{i+4}$ is computed by $\Delta v_{i+3} \oplus \Delta v_{i+5}$, where both Δv_{i+3} and Δv_{i+5} are of type $\mathcal{P}$. Since P is linear, $\Delta F^{\mathcal{O}}_{i+4}$ also has the form $\mathcal{P}$, which implies that the form of $\Delta F^{\mathcal{M}}_{i+4}$ is $P^{-1}(\mathcal{P}) = \mathbf{1}$. Then, the middle difference $\Delta F^{\mathcal{I}}_{i+4}$ is considered fixed. When $\Delta F^{\mathcal{I}}_{i+4} \neq \Delta F^{\mathcal{I}}_{i+2}$ and $\Delta F^{\mathcal{I}}_{i+4} \neq \Delta F^{\mathcal{I}}_{i+6}$, the corresponding differences $\Delta F^{\mathcal{O}}_{i+3}$ and $\Delta F^{\mathcal{O}}_{i+5}$ are computed by simply taking their XOR. Thus, both $\Delta F^{\mathcal{O}}_{i+3}$ and $\Delta F^{\mathcal{O}}_{i+5}$ are of type $\mathbf{1}$, which makes $\Delta F^{\mathcal{M}}_{i+3}$ and $\Delta F^{\mathcal{M}}_{i+5}$ fully active (denoted by $\mathcal{P}^{-1}$). Then, the values of $F^{\mathcal{I}}_{i+3}, F^{\mathcal{M}}_{i+3}, F^{\mathcal{O}}_{i+3}$ and $F^{\mathcal{I}}_{i+5}, F^{\mathcal{M}}_{i+5}, F^{\mathcal{O}}_{i+5}$ are uniquely determined, as well as the values for $F^{\mathcal{I}}_{i+4}[\mathbf{1}], F^{\mathcal{M}}_{i+4}[\mathbf{1}]$.

Finally, when we additionally consider the $n/2-c$ inactive bits of $F^{\mathcal{I}}_{i+4}$ marked by a star in Figure 9 being fixed, along with the already-fixed c bits of the active word $\mathbf{1}$, the full $n/2$-bit values of $F^{\mathcal{M}}_{i+4}$ and $F^{\mathcal{O}}_{i+4}$ are determined. In summary, for each value of the five c-bit active differences circled in Figure 9 and the $n/2-c$ inactive bits of $F^{\mathcal{I}}_{i+4}$, all the differences of the differential as well as one word values in rounds $i+2$, $i+6$, and all state values in rounds $i+3$, $i+4$, $i+5$ are uniquely fixed.

For each of $5c + n/2 - c = n/2 + 4c$ word parameters, we can partially evaluate a b-δ-set v_i up to $\Delta v'_{i+7}[\mathbf{1}]$. Namely, for one member of the pairs, $v_i[\mathbf{1}]$ is modified so that $\Delta v_i[\mathbf{1}]$ becomes δ_j. The modification changes the difference in subsequent rounds, but we still can compute the corresponding difference $\Delta v'_{i+7}[\mathbf{1}]$ without requiring the knowledge of the subkey bits.

Indeed, in round $i+1$, $\Delta F^{\mathcal{O}}_{i+1} = 0$, $\Delta v_{i+2} = \Delta F^{\mathcal{I}}_{i+2} = \delta_j$. In round $i+2$, from the original active word value of $F^{\mathcal{I}}_{i+2}$ and updated difference $\Delta F^{\mathcal{I}}_{i+2} = \delta_j$, the updated $\Delta F^{\mathcal{O}}_{i+2}$ can be computed as $P \circ S(F^{\mathcal{I}}_{i+2}) \oplus P \circ S(F^{\mathcal{I}}_{i+2} \oplus \delta_j)$. This also derives the updated differences Δv_{i+3} and $\Delta F^{\mathcal{I}}_{i+3}$. Then, in round $i+3$ to $i+5$, from the original value and the updated difference of $F^{\mathcal{I}}_x$, the updated difference $\Delta F^{\mathcal{O}}_x$, and moreover the updated differences Δv_{x+1} and $\Delta F^{\mathcal{I}}_{x+1}$ can be computed for $x = i+3, i+4, i+5$. Note that, in round $i+4$, $\Delta F^{\mathcal{I}}_{i+4}$ originally has only one active word, while the updated difference is fully active. Because $n/2-c$ inactive bits of $F^{\mathcal{I}}_{i+4}$ are parameters, and thus known to the attacker, $\Delta F^{\mathcal{M}}_{i+4}$ can be computed in all words. Finally, in round $i+6$, the updated difference Δv_{i+6} is known in all words while the original value is known only in one active word. Since the position of the P-layer is moved, the attacker can still compute the 1-word updated difference $\Delta v'_{i+7}[\mathbf{1}]$.

To conclude, for each of the $2^{n/2+4c}$ possible values of the parameters, the sequence of $\Delta v'_{i+7}[\mathbf{1}]$ is uniquely obtained by computing $\Delta v'_{i+7}[\mathbf{1}]$ for all δ_j in $\Delta v_i[\mathbf{1}]$, which concludes the proof. □

10-round key-recovery attack. We now describe the 10-round key-recovery attack using this 7-round distinguisher. Namely, as shown in Figure 10, we extend the 7-round differential of the distinguisher by one round at the beginning and two rounds at the end (the analysis and complexity will be similar if we extend by two rounds at the beginning and one at the end). Recall that the P computation after v'_7 introduced by the distinguisher has to be addressed in the key-recovery part. We also note that the active word $\mathbf{1}$ in the branches can be located in any position, but to be able to conduct the attack, the position has to be fixed beforehand. The P-layer in round 8 is moved to two different positions as shown in Figure 10. The newly-introduced P^{-1} transformation and the P transformation after v'_7 generated by the distinguisher cancel each other, we therefore ignore them. Similarly to the analysis for Feistel-2, the

Algorithm 5: Construction of the difference sequences of $\Delta v_7'[\mathbf{1}]$ (precomputation).

1: **for** all $2^{n/2+4c}$ values of the parameters **do**
2: Derive all differences of the differential.
3: Derive 1-word state values in rounds 2 and 6, and all state values in rounds 3, 4 and 5.
4: **for** 2^b different differences in v_0 **do**
5: Modify $\Delta v_0[\mathbf{1}]$, and update the corresponding sequence of $\Delta v_7'[\mathbf{1}]$.
6: Insert the sequence of $\Delta v_7'[\mathbf{1}]$ in the table T_δ.

Algorithm 6: Data collection for the 10-round attack.

1: Fix the $n/2 - c$ inactive bits of v_0 and v_{-1}.
2: **for** all 2^{2c} choices (v_0, v_{-1}) **do**
3: Query (v_0, v_{-1}) to obtain (v_9, v_{10}).
4: Store (v_9, v_{10}) in a hash table indexed by the wanted inactive bits in $P^{-1}(v_9)$.
5: Construct about $2^{4c}/2^{n/2-c} = 2^{-n/2+5c}$ pairs verifying the truncated ciphertext difference.
6: Iterate the above analysis 2^{n-4c} times by changing the the inactive-bit value of v_0 and v_t.

attack consists of three parts: the precomputation phase, followed by the data collection and finally the meet-in-the-middle check to detect correct subkey candidates.

Precomputation. Given the proof of Proposition 2, the precomputation phase is straightforward. For each of the $2^{n/2+4c}$ values of the parameters, and for any value of δ_j constructed at v_0, the corresponding $\Delta v_7'[\mathbf{1}]$ can be computed easily as shown in Algorithm 5. As in the attack on Feistel-2, in this phase we construct the meet-in-the-middle table T_δ that contains all the sequences of differences in $\Delta v_7'[\mathbf{1}]$ for $2^b < 2^c$ nonzero differences δ_j in v_0. The computational cost is about $2^{n/2+4c}$ encryptions as the b parameter is relatively small and we consider only a small fraction of all the rounds. The memory requirement for T_δ is $2c/n \times 2^{n/2+4c+b}$ blocks of $n/2$ bits, as the sequences contains 2^b elements of c bits.

Collecting pairs. To be able to launch the attack, we need a pair that satisfies the 7-round differential characteristic in Figure 9, i.e. the plaintext difference $(\mathbf{1}, \mathcal{P})$ should propagate to the ciphertext difference $(\mathcal{P}, A)$, where A is a truncated difference. The probability that the plaintext difference $(\mathbf{1}, \mathcal{P})$ after the first round becomes $(\mathbf{0}, \mathbf{1})$ is 2^{-c}, while the probability that the ciphertext difference $(\mathcal{P}, A)$ after inversion of the last round becomes $(\mathbf{1}, \mathbf{1})$ is $2^{-n/2+c}$, and to become $(\mathbf{1}, \mathbf{0})$ after another inverse round is 2^{-c}. Therefore, a random pair verifying a plaintext difference $(\mathbf{1}, \mathcal{P})$ conforms to the inner 7-round differential with probability $2^{-n/2-c}$. Hence, we need to collect $2^{n/2+c}$ pairs satisfying the differential $(\mathbf{1}, \mathcal{P}) \overset{10R}{\rightarrow} (\mathcal{P}, A)$. Among all of them, one is expected to satisfy $(\Delta v_1, \Delta v_0) = (\mathbf{0}, \mathbf{1})$ and $(\Delta v_8, \Delta v_7) = (\mathbf{1}, \mathbf{0})$. The procedure is given in Algorithm 6.

For fixed values of the inactive bits in v_0 and v_{-1}, about 2^{4c} pairs can be generated, and we expect approximately $2^{4c} \cdot 2^{-n/2+c} = 2^{-n/2+5c}$ of them to verify the ciphertext truncated difference $(\mathcal{P}, A)$. By iterating the procedure for 2^{n-4c} different values, we obtain $2^{n-4c-n/2+5c} = 2^{n/2+c}$ pairs satisfying the desired $(\Delta v_0, \Delta v_{-1})$ and $(\Delta v_9, \Delta v_{10})$. The data complexity required to generate the $2^{n/2+c}$ pairs amounts to approximately $2^{2c+n-4c} = 2^{n-2c}$ chosen plaintexts, the computational cost is equivalent to 2^{n-2c} memory accesses, and the memory requirement is about $2^{n/2+c}$ blocks of $n/2$ bits.

Detecting subkeys. For each of the $2^{n/2+c}$ obtained pairs, we derive 2^c candidates for $n/2 + 2c$ bits of key material, namely $K_0[\mathbf{1}]$, $K_8[\mathbf{1}]$, and K_9. For each pair, we first guess the 1-word difference of $\Delta v_8[\mathbf{1}]$. Then, we assume that the differential is satisfied, i.e. $\Delta v_1 = \mathbf{0}$, $\Delta v_7' = \mathbf{0}$, and $\Delta v_8 = \mathbf{1}$. This fixes the input and output differences for the active words in rounds 0 and 8, and for all words in round 9. Then, the input values for these S-Boxes are fixed to one value on average, and the corresponding subkey values $K_0[\mathbf{1}]$, $K_8[\mathbf{1}]$ and K_9 can be calculated.

Finally, we construct the b-δ-set by modifying $v_0[\mathbf{1}]$. For each modified plaintext, with the knowledge of $K_0[\mathbf{1}]$, we modify v_{-1} such that v_1 remains unchanged. From the corresponding ciphertexts, with the knowledge of K_9 and $K_8[\mathbf{1}]$, we compute the sequence of 2^b differences $\Delta v_7'[\mathbf{1}]$, and if it matches one of the entries in the precomputed table T_δ, then the guessed subkeys $K_0[\mathbf{1}]$, $K_8[\mathbf{1}]$, and K_9 are correct, otherwise they are wrong. When the values of c and n are in a particular range (see below), only the right guess will remain, thus the subkeys are recovered.

The computational cost of the key-recovery phase is the one for computing $\Delta v_7'[\mathbf{1}]$ for $2^{n/2+c}$ pairs, 2^c guesses for $\Delta v_8[\mathbf{1}]$, and 2^b choices of δ_j in the b-δ-set, which is upper bounded by $2^{n/2+3c}$ encryptions.

Complexity analysis and constraints on (n, c). As shown above, the data complexity requires 2^{n-2c} chosen plaintexts, the time complexity is equivalent to $2^{n-2c} + 2^{n/2+5c}$ encryptions and the memory complexity is $2^{n/2+5c}$ blocks of $n/2$ bits. We note that the overall complexity is balanced when $n/2c = 7$, i.e. when a branch includes 7 S-Boxes. On top on the attack, it is also possible to use a simple tradeoff where only a fraction $1/2^c$ of all the sequences are stored in T_δ, which decreases the memory complexity to $2^{n/2+4c}$ blocks of $n/2$ bits, but in turn makes the data complexity and the time complexity of the online phase increased by a factor 2^c as we have decreased the chance to hit one element in T_δ. With this tradeoff, the data complexity becomes 2^{n-c} chosen plaintexts, and the time complexity becomes about $2^{n-c} + 2^{n/2+4c}$ encryptions, which is balanced for $n/2c = 5$ S-Boxes per branch.

Moreover, to launch the attack, a branch must have at least 5 S-Boxes so that $n/2 + 4c < n$. Additionally, in the subkey detection phase, the number of remaining key candidates should be 1 or small enough. The number of sequences in T_δ is $2^{n/2+4c}$ and the number of candidates derived online is $2^{n/2+2c}$. Thus in total, 2^{n+6c} matches are examined, whether or not we use the tradeoff. In theory, there exists $2^{c \cdot 2^b}$ sequences from $b < c$ bits to c bits. Hence, the condition to extract only the correct subkey is $n + 6c - c \cdot 2^b < 0$, which gives $b > \log_2(6 + n/c)$. Since $2^b < 2^c$, by combining the two conditions, the valid range for (n, c) is $10c \leq n < c(2^c - 6)$. For example, 128-bit block ciphers with 8-bit S-Boxes and 80-bit block ciphers with 5-bit S-Boxes can be attacked.

Another possible tradeoff is the one used to achieve the best attacks on reduced variants of the `AES` in [10]. We show that adding a second active word at the beginning of the differential allows to reduce the data complexity while keeping the same overall complexity. This tradeoff is possible as long as there are at least 7 words per branch, i.e. $n/2c \geq 7$. The main advantage of adding an active word is to increase the size of the structures of plaintext from 2^{2c} to 2^{4c}, which allows to construct about 2^{8c} input pairs already verifying the input difference. The precomputation requires $2^{n/2+6c}$ encryptions and a memory of $2c/n \times 2^{n/2+6c+b}$ blocks of $n/2$ bits, the online phase requires more pairs, namely $2^{n/2+2c}$, but this is achieved with less data: only 2^{n-3c} chosen plaintexts. Therefore, the final time complexity is $2^{n-3c} + 2^{n/2+6c}$ for both the encryption of the data and the precomputation. This yields an attack as long as $n/2 + 6c < n$, which is true for $n/2c \geq 7$ S-Boxes. For example, with 8 S-Boxes per branch, the attack without tradeoff gives an attack with $2^{14n/16}$ chosen plaintexts, $2^{14n/16}$ encryptions and the memory for about $2^{12n/16}$ blocks of $n/2$ bits so that the overall complexity is $2^{14n/16}$. For the same primitive, applying the tradeoff gives the same overall complexity while reducing the data complexity to $2^{13n/16}$ chosen plaintexts.

4.2 9-round key-recovery attack for different S and P in different rounds

The previous 10-round attack assumes an identical P-layer in all the rounds. This assumption helps to cancel P and P^{-1} introduced by the linear transformations in rounds 6 and 8, respectively. If different P-layers are used in those rounds, i.e., P_6 and P_8, then P_6 and P_8^{-1} do not cancel each other, which forces us to guess all $n/2$ bits of K_8, and the attack complexity exceeds the exhaustive search bound 2^k. However, even if P-layers are different in all rounds, the meet-in-the-middle attack still works up to 9 rounds, which improves by two more rounds the best known results.

The strategy of the 9-round attack is similar to the 10-round attack described in Section 4.1. The main differences are that the number of rounds for the differential characteristic is 6, which is reduced by one round, and the linear transformation to move the position of P in round 6 is not applied (see Figure 11). We note that the linear transformation in the penultimate round is still performed.

Distinguisher. Consider the 6-round differential characteristic in Figure 11 from round $i+1$ to $i+6$. For the key-recovery part later, we apply the linear transformation in round $i+7$, which introduces P_{i+7}^{-1} after v_{i+6}. We deal with this P_{i+7}^{-1} inside the characteristic. In Figure 11, internal state values are parameterized by $n/2 + 4c$ bits; namely: c bits for $\Delta F_{i+2}^{\mathcal{I}}$, $\Delta F_{i+2}^{\mathcal{M}}$, $\Delta F_{i+6}^{\mathcal{I}}$, $\Delta F_{i+6}^{\mathcal{M}}$ and $n/2$ bits of $\Delta F_{i+4}^{\mathcal{I}}$. When the $4c$ bits corresponding to $\Delta F_{i+2}^{\mathcal{I}}$, $\Delta F_{i+2}^{\mathcal{M}}$, $\Delta F_{i+6}^{\mathcal{I}}$ and $\Delta F_{i+6}^{\mathcal{M}}$ are fixed, the associated $\mathcal{P}_{i+2}$ and $\mathcal{P}_{i+6}$ are computed. At round $i+4$, $\Delta F_{i+4}^{\mathcal{M}}$ is computed as $p_{i+4}^{-1}(\mathcal{P}_{i+2} \oplus \mathcal{P}_{i+6})$. Since the P-layers are

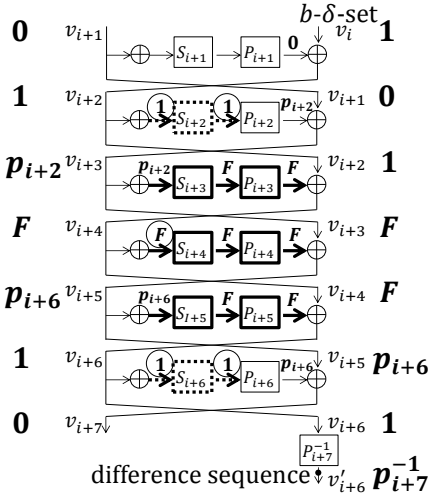

Figure 11: 6-round characteristic for different P-layers.

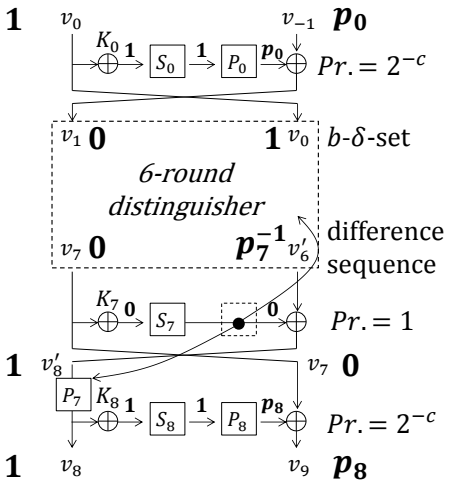

Figure 12: 9-round key-recovery for different P-layers.

different in each round, this is not transformed to type **1** but to type **F** at $F^{\mathcal{M}}_{i+4}$. Then, if the $n/2$-bit difference $F^{\mathcal{I}}_{i+4}$ is fixed, all the input and output differences for the three middle rounds are fixed, and thus we can obtain the corresponding values.

For each value of the $n/2 + 4c$ bit-parameters, we modify Δv_i to construct the b-δ-set, computes the sequence of $\Delta v'_{i+6}[\mathbf{1}]$ and stores it in T_δ. The procedure is essentially the same as in the 10-round attack, so we omit the full details here. The full difference Δv_{i+6} is computed, so that $\Delta v'_{i+6}$ after the newly-introduced P^{-1}_{i+7} is also fully computed. However, the key-recovery phase only uses the information for one word. Consequently, we store only the one-word value of $\Delta v'_{i+6}[\mathbf{1}]$ in the table T_δ. Both the computational cost (number of encryptions) and the memory requirement (blocks of $n/2$ bits) to compute T_δ are $2^{n/2+4c}$. Here the factor of 2^b is ignored as $2^b < 2^c$.

Key recovery. For the 9-round key recovery, one round and two rounds are injected before and after the 6-round distinguisher as shown in Figure 12. To collect pairs satisfying the characteristic, we construct 2^d structures, each consisting of 2^{2c} chosen plaintexts. Each structure allows to generate 2^{4c} pairs. The probability of satisfying the ciphertext difference is 2^{-n+2c}, thus we expect about 2^{d-n+6c} pairs to remain. The probability of the differential propagation in the key recovery part is 2^{-c} for round 0 and 2^{-c} for round 8. Thus, we need about 2^{2c} pairs during the online phase. The data complexity therefore equals 2^{n-2c} chosen plaintexts to obtain 2^{2c} pairs for the key-recovery part.

In the key-recovery part, we recover $n/2 + 2c$ subkey bits: $K_0[\mathbf{1}]$, $K_7[\mathbf{1}]$ and K_8. For each of the obtained 2^{2c} pairs, suppose that the 6-round characteristic is satisfied. Then, the plaintext difference derives the one-word value of $F^{\mathcal{I}}_0[\mathbf{1}]$, and further deduces a candidate for $K_0[\mathbf{1}]$. Similarly, the ciphertext difference derives the one-word value of $F^{\mathcal{I}}_8[\mathbf{1}]$, and further derives a candidate for $K_8[\mathbf{1}]$. We then guess the value of $n/2 - c$ inactive bits in K_8 and c bits of $K_7[\mathbf{1}]$. For each key guess, we modify the plaintext so that a b-δ-set is constructed at v_0. Then, with the knowledge of $K_0[\mathbf{1}]$, he modifies v_{-1} so that v_1 remains unaltered. From the corresponding ciphertexts and the subkeys K_8 and $K_7[\mathbf{1}]$, the sequence of $\Delta v'_6[\mathbf{1}]$ can be computed. If a matching sequence is found in T_δ, the guessed subkey bits are correct. The computational cost of the online phase is equivalent to approximately $2^{n/2+2c}$ encryptions. As before, we ignore the small 2^b factor.

Finally, we evaluate the optimal value of b. The precomputation phase generates $2^{n/2+4c}$ sequences of $\Delta v'_6[\mathbf{1}]$ in T_δ. The online phase generates $2^{n/2+2c}$ sequences of $\Delta v'_6[\mathbf{1}]$. Thus, the total number of checks performed in T_δ is 2^{n+6c}. This is the same as the 10-round attack in Section 4.1, which gives $b > \log_2(6 + n/c)$. Note that tradeoffs and extensions can be introduced in the very same way as for the 10-round attack. The extensions to longer key sizes are explained in Appendix A.3.

5 Conclusion

With the use of the meet-in-the-middle technique, we have shown the best known generic attacks on balanced Feistel ciphers. As we imposed no particular restrictions on the round functions, our attacks are applicable to practically all balanced Feistels. Such ciphers, with an arbitrary round function and a double key are insecure on up to 10 rounds. In the case when the round function is SPN, for a large class of linear P-layers the attacks penetrate 14 rounds and recover all the subkeys. To confirm the correctness of our reasoning, we have produced experimental verification of the attacks.

Our results give insights on the lower bound on the number of rounds a secure Feistel have. They suggest that this number in the case of SPN round functions should be surprisingly high. Furthermore, from the attacks on Feistel-2 and their extension to more rounds, we can see that as long as the ratio of key to state size is increasing, the number of rounds attacked will grow, while the data complexity will always stay below the full codebook. Thus, we have shown that *a block cipher designer cannot fix a priori the number of rounds in a balanced Feistel and allow any (or very large) key size*, as for each increment of the key by amount of bits equivalent to the state size, we can attack four more rounds.

We have analyzed generic constructions and as such, we could not make any assumptions about the particular details of the ciphers, e.g. the key schedule, the permutation layer, etc. However, the attacks on the `AES` have shown that it is possible to take advantage of the cipher details in order to penetrate more rounds. Thus, we believe that our analysis can be used as a beginning step for attacks on larger number of rounds of specific Feistel ciphers: that is, we expect the attacks to be improved once the details are taken into account.

Finally, we would like to stress out the importance of the meet-in-the-middle technique. Although relatively new, it has already shown a lot of potential and its application has resulted in the best attacks on some ciphers, especially the `AES`. We leave as an open problem the application of this technique to some other classes of ciphers, for instance, non-balanced Feistels or ciphers with round functions based on addition-rotation-xor.

References

1. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis. In Stinson, D.R., Tavares, S.E., eds.: Selected Areas in Cryptography. Volume 2012 of LNCS., Springer (2000) 39–56
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013)
3. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function. Submission to NIST (Round 2) (2009)
4. Communications Security Establishment Canada: Cryptographic algorithms approved for Canadian government use. (2012)
5. Coppersmith, D.: The Data Encryption Standard (DES) and its Strength Against Attacks. IBM Journal of Research and Development. **38**(3) (1994) 243–250
6. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher Square. In Biham, E., ed.: FSE. Volume 1267 of Lecture Notes in Computer Science., Springer (1997) 149–165
7. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round `AES`. In Nyberg, K., ed.: FSE. Volume 5086 of LNCS., Springer (2008) 116–126
8. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In Nyberg, K., ed.: FSE. Volume 5086 of LNCS., Springer (2008) 116–126
9. Derbez, P., Fouque, P.A., Jean, J.: Improved key recovery attacks on reduced-round aes in the single-key setting. IACR Cryptology ePrint Archive **2012** (2012) 477
10. Derbez, P., Fouque, P.A., Jean, J.: Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In Johansson, T., Nguyen, P.Q., eds.: EUROCRYPT. Volume 7881 of LNCS., Springer (2013) 371–387
11. Dinur, I., Dunkelman, O., Keller, N., Shamir, A.: Efficient Dissection of Composite Problems, with Applications to Cryptanalysis, Knapsacks, and Combinatorial Search Problems. In Safavi-Naini, R., Canetti, R., eds.: CRYPTO. Volume 7417 of LNCS., Springer (2012) 719–740
12. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-round AES-192 and AES-256. In Abe, M., ed.: ASIACRYPT. Volume 6477 of LNCS. Springer (2010) 158–176
13. Feistel, H., Notz, W., Smith, J.: Some Cryptographic Techniques for Machine-to-Machine Data Communications. Proceedings of IEEE **63**(11) (1975) 15545–1554
14. Gilbert, H., Minier, M.: A Collision Attack on 7 Rounds of Rijndael. In: AES Candidate Conference. (2000) 230–241
15. Gilbert, H., Minier, M.: A Collision Attack on 7 Rounds Rijndael. In: Third AES Candidate Conference (AES3), Springer (2000) 230–241
16. Isobe, T., Shibutani, K.: All Subkeys Recovery Attack on Block Ciphers: Extending Meet-in-the-Middle Approach. In Knudsen, L.R., Wu, H., eds.: Selected Areas in Cryptography. Volume 7707 of LNCS., Springer (2012) 202–221

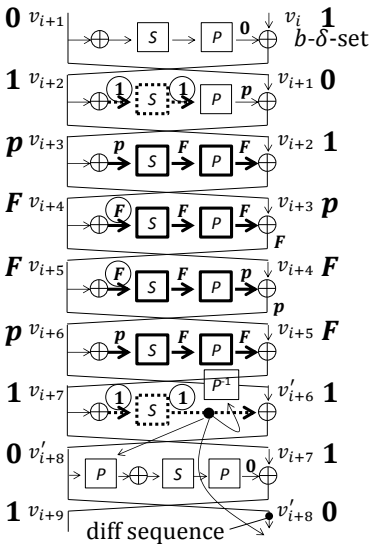

Figure 13: 8-round distinguisher for $k = 3n/2$.

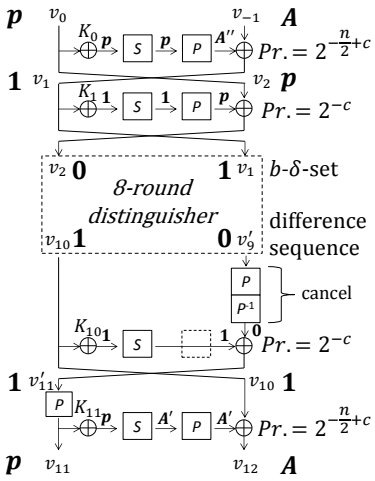

Figure 14: 12-round key recovery for $k = 3n/2$.

17. Isobe, T., Shibutani, K.: Generic Key Recovery Attack on Feistel Scheme. In Sako, K., Sarkar, P., eds.: ASIACRYPT (1). Volume 8269 of LNCS., Springer (2013) 464–485
18. ISO/IEC 18033-3:2010: Information technology–Security techniques–Encryption Algorithms–Part 3: Block ciphers. (2010)
19. Knudsen, L.R.: The Security of Feistel Ciphers with Six Rounds or Less. J. Cryptology **15**(3) (2002) 207–222
20. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. **17**(2) (1988) 373–386
21. Merkle, R.C., Hellman, M.E.: On the Security of Multiple Encryption. Commun. ACM **24**(7) (1981) 465–467
22. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In Preneel, B., Takagi, T., eds.: CHES. Volume 6917 of Lecture Notes in Computer Science., Springer (2011) 342–357
23. Todo, Y.: Upper Bounds for the Security of Several Feistel Networks. In Boyd, C., Simpson, L., eds.: ACISP. Volume 7959 of LNCS., Springer (2013) 302–317
24. Wu, W., Zhang, L.: LBlock: A Lightweight Block Cipher. In Lopez, J., Tsudik, G., eds.: ACNS. Volume 6715 of LNCS., Springer (2011) 327–344
25. Zhang, L., Wu, W., Wang, Y., Wu, S., Zhang, J.: LAC: A Lightweight Authenticated Encryption Cipher. Submitted to the CAESAR competition (March 2014)

A Extension of the attacks on Feistel-3 to 12 and 14 rounds

A.1 12-round key-recovery attack for $k = 3n/2$

We extend here the previous 10-round result on Feistel-3 to 12 rounds, for the case with key size $k = 3n/2$. The main difference compared to the previous attack is: first, the distinguisher has one more round (i.e., 8 rounds), and second, we prepend an additional round at the beginning (that is, two rounds before and after the 8-round distinguisher). In the sequel, we use $\mathbf{F}$ to denote a fully active difference of $n/2$-bit branch. Note that there are $2^{n/2} - 1$ valid $\mathbf{F}$ whereas there only $2^c - 1$ valid $\mathbf{p}$ for a fix $\mathbf{1}$ position. The differential characteristic for the distinguisher is the natural extension where one full active round is added in the middle. The truncated characteristic therefore becomes (also see Figure 13):

$$(0,\mathbf{1}) \xrightarrow{1R} (\mathbf{1},0) \xrightarrow{1R} (\mathbf{p},\mathbf{1}) \xrightarrow{1R} (\mathbf{F},\mathbf{p}) \xrightarrow{1R} (\mathbf{F},\mathbf{F}) \xrightarrow{1R} (\mathbf{p},\mathbf{F}) \xrightarrow{1R} (\mathbf{1},\mathbf{p}) \xrightarrow{1R} (0,\mathbf{1}) \xrightarrow{1R} (\mathbf{1},0).$$

The precomputation phase (construction of T_δ) works exactly as before, except that we need to include the full active difference $\mathbf{F}$ in the parameters. Therefore, the table size is increased by a factor of $2^{n/2}$, and thus the memory complexity to store T_δ is 2^{n+4c+b} blocks of $n/2$ bits.

To collect the data, we use 2^d structures containing $2^{n/2+c}$ plaintexts (see Figure 14), so that $2^{d+n+2c}/2^{n/2-c} = 2^{d+n/2+3c}$ are expected to verify the output differences, which is constrained on $n/2 - c$ bits. In the online phase, we need 2^n pairs since the differential characteristic in the first two rounds hold with probability $2^{-(n/2-c)-c} = 2^{-n/2}$, and the same is true for the last two rounds. Hence, the number of structures should be $2^d = 2^{n/2-3c}$, which makes the data complexity equals 2^{n-2c} chosen plaintexts.

Finally, in the online phase, for each of the 2^n pairs, we need to guess the $\mathbf{1}$ difference at the input of the second and the $\mathbf{1}$ difference at the input of the penultimate round. In total, for each 2^{n+2c} combinations, we get a suggestion for $n+2c$ bits of key material: K_0, $K_1[\mathbf{1}]$, $K_{10}[\mathbf{1}]$ and K_{11}. Then, the remaining of the attack is similar to the previous one: with that knowledge of the key material, we can partially encrypt/decrypt the plaintext/ciphertext pairs to construct the b-δ-set and check for a match in the table T_δ.

Complexity analysis. Again, we analyze the number of elements in the sequences so that false positives are avoided, which gives $b > \log_2(6+2n/c)$ for this attack. In total, the data complexity is 2^{n-2c} chosen plaintexts. The time complexity is equivalent to $2^{n+4c+b} + 2^{n-2c} + 2^{n+2c}$ encryptions, which is dominated by the precomputation step of 2^{n+4c+b} encryptions. The memory complexity is also the one from the precomputation, with 2^{n+4c+b} blocks of $n/2$ bits.

As a tradeoff to balance the complexities on the precomputation and the online phases, we may choose to store a fraction $1/2^c$ of the sequences in T_δ, which in turn imposes to replay the online attack with 2^c more pairs. We can for instance achieve this by fixing the input difference X to the distinguisher to a random nonzero value. This result in a data complexity of 2^{n-c} chosen plaintexts, 2^{n+3c+b} encryptions, and memory of 2^{n+3c} blocks of $n/2$ bits.

Similarly to the 10-round attack from Section 4.1, we can add one active word in the distinguisher characteristic to increase the size of the structure for reducing the data complexity. However, in the present case where $k = 3n/2$ doing so would increase the overall complexity as the bottleneck complexity would also increase. Although this would yield an attack faster than the exhaustive search bound, we do not give the full details on this tradeoff.

A.2 14-round key-recovery attack for $k = 2n$

Now, we show how to attack two more rounds of Feistel-3 for $k = 2n$, that is 14 rounds, by increasing the distinguisher by one round to consider 9 rounds, and by prepending an extra round at the beginning. The truncated differential characteristic used in the distinguisher is given by

$$(0,\mathbf{1}) \xrightarrow{1R} (\mathbf{1},0) \xrightarrow{1R} (\mathbf{p},\mathbf{1}) \xrightarrow{1R} (\mathbf{F},\mathbf{p}) \xrightarrow{1R} (\mathbf{F},\mathbf{F}) \xrightarrow{1R} (\mathbf{F},\mathbf{F}) \xrightarrow{1R} (\mathbf{p},\mathbf{F}) \xrightarrow{1R} (\mathbf{1},\mathbf{p}) \xrightarrow{1R} (0,\mathbf{1}) \xrightarrow{1R} (\mathbf{1},0),$$

where the additional round is the fully active one placed in the middle.

The extension from 12 to 14 rounds works essentially the same as the one from 10 to 12 rounds. Namely, the table T_δ now contains $2^{3n/2+4c}$ sequences of 2^b elements, and is constructed in $2^{3n/2+4c+b}$ computations simpler than full encryptions, and requires the same number of $n/2$-bit units to be stored. As for the data collection, the structures of plaintexts are not constrained anymore as we have added one round at the beginning, which destroyed the $(n/2-c)$-bit constraint on the left half. The number of pairs required by the online phase is increased by a factor of $2^{n/2-c}$ as we need to verify a $\mathbf{p}$ difference in the second round. Therefore, 2^d plaintexts in a structure lead to $2^{2d}/2^{n/2-c} = 2^{2d-n/2+c}$ pairs with a valid output difference, and we want $2^{n+n/2-c}$ pairs, hence the data complexity corresponds to 2^{n-c} chosen plaintexts. Finally, in the online phase, we analyze $2^{3n/2-c}$ pairs, we guess three words to perform the partial encryption/decryption, which give a suggestion for each of the $3n/2+2c$ bits of key material: K_0, K_1, $K_2[\mathbf{1}]$, $K_{12}[\mathbf{1}]$ and K_{13}. The remaining of the attack is exactly the same as before.

Complexity analysis. The number of elements in the sequences so that false positives are avoided, result in the condition $b > \log_2(6+3n/c)$. In total, the data complexity is 2^{n-c} chosen plaintexts. The time complexity is equivalent to $2^{3n/2+4c+b} + 2^{n-c} + 2^{3n/2+2c}$ encryptions, and it is dominated by the precomputation step that requires $2^{3n/2+4c+b}$ encryption queries. The memory complexity is also the one from the precomputation, with $2^{3n/2+4c+b}$ blocks of $n/2$ bits.

As the data complexity of this attack is very close to full codebook, only tradeoffs not increasing the data complexity could apply. However, even the previous tradeoff adding an extra active word in the distinguisher characteristic does not apply here. Indeed, the plaintexts cannot profit from any particular structure as both halves are fully active with no specific restrictions. Therefore adding the extra word would make a lower probability for the online phase, requiring more data, which is not wanted.

A.3 Attacks against Feistel-3 with different S and P for $k = 3n/2$ and $k = 2n$

The previous 9-round attack against Feistel-3 with different S- and P-layers for $k = n$ shown in Section 4.2 can be extended to 11 rounds and 13 rounds for $k = 3n/2$ and $k = 2n$, respectively. The extension to larger key sizes can be done similarly to Feistel-3 with an identical S and P shown in Section A.1 and Section A.2. Namely, the differential characteristic will contain 1 or 2 fully active rounds in the middle. When we construct the distinguisher, those differences are guessed as part of parameters by spending additional $2^{n/2}$ or 2^n computational cost and memory requirement. To avoid the redundancy, the detailed explanation is omitted. As for $k = n$, the attack complexity against 11 rounds for $k = 3n/2$ and 13 rounds for $k = 2n$ are the same as the 12-round attack in Section A.1 and the 14-round attack in Section A.2, respectively.

B Attacks on Feistel-3 for a class of P-layers

Let us confirm that the 10-round attack on Feistel-3 (and the subsequent extensions) is valid for some cases when the P-layer does not have a maximal branch number. The relaxed requirement on the P-layer can be seen as a full diffusion. Again, the attack is based on a 7-round truncated differential. Let $\mathbf{1}$ and $\mathbf{1}'$ be two one-word truncated differences that do not necessarily have the active word at the same position. Let $\mathbf{p}$ and $\mathbf{p}'$ be the truncated differences $P(\mathbf{1})$ and $P(\mathbf{1}')$, respectively, and they are active in all words (if not then the time complexity of the attack will increase). Then the differential has the form:

$$(0, \mathbf{1}) \xrightarrow{1R} (\mathbf{1}, 0) \xrightarrow{1R} (\mathbf{p}, \mathbf{1}) \xrightarrow{1R} (\mathbf{1} \oplus \mathbf{1}', \mathbf{p}) \xrightarrow{1R} (\mathbf{p}', \mathbf{1} \oplus \mathbf{1}') \xrightarrow{1R} (\mathbf{1}', \mathbf{p}) \xrightarrow{1R} (0, \mathbf{1}') \xrightarrow{1R} (\mathbf{1}', 0).$$

To avoid possible contradictions in the differential, the P-layer has to satisfy certain conditions. A careful investigation shows that there are two conditions imposed on the P-layer: $P(\mathbf{p}) = \mathbf{1}'$ and $P(\mathbf{p}') = \mathbf{1}$, which reduces to $P(P(\mathbf{1})) = \mathbf{1}', P(P(\mathbf{1}')) = \mathbf{1}$. Thus, the assumption of a maximal branch number can be replaced with a somewhat weaker requirement, which broaden the class of P-layers that can be attacked, as shown further.

We can proceed as in the original 10-round attack on Feistel-3 (assume P-layer of round 6 has been moved). To adapt Proposition 2 for this case, we find all possible differences that follow this differential. We fix the input/output difference of the single active words in rounds 2 and 6, the two input differences at round 4 (or one if $\mathbf{1} = \mathbf{1}'$), and the output difference of the S-layer at round 3, i.e. $\Delta F^{\mathcal{M}}_{i+3}$, or in total $2^{6c+n/2}$ differences. This fixes the input and the output differences of all active words to the S-layers in all 7 rounds, and reveals the values of two active words at rounds 2 and 6, two words at round 4, and all words of states at rounds 3 and 5. To construct the b-δ sequences for each fixed differences, we first guess the remaining inactive words of round 4 (in total $n/2 - 2c$), and proceed further as in the original attack. Thus, the total time and memory complexity of the offline phase required to construct T_δ is $2^{n/2+4c}$ i.e. exactly the same complexities as before. We note that T_δ still has $2^{n/2+4c}$ elements because each word guess corresponds to a different entry in the table.

In the data collection phase, the probabilities of the three additional rounds remain the same (the input to the tenth round of the S-Boxes is active in several words, so the output can be any) and thus we need $2^{n/2+c}$ pairs to confirm the differential behavior of the middle 7 rounds. Similarly, the key-recovery phase is exactly the same except that we can recover all words of K_9 except one (the non-active word). As a result, the whole attack has precisely the same time, data and memory complexities as the original attack on 10-round Feistel-3, where the P-layer has a maximal branch number.

C Recovering all Subkeys

In Sections 3 and 4, we have shown attacks that recover some subkeys (full values and words). Since our attack is generic and nothing has been assumed on the key schedule, there is a possibility that the knowledge of a subkey does not result in easily computable reduction of the master key space, for instance the case of non-invertible key schedule. An alternative is to recover all the subkeys of the cipher, which also allows the attacker to encrypt or decrypt any plaintext or ciphertext of his/her choice. In this section, we show how all the subkeys can be recovered.

Feistel-2. It is important to note that in all our attacks on Feistel-2, the first round subkey K_0 is recovered. With the knowledge of K_0, we can either repeat the same procedure or re-use other existing methods to attack a sub-cipher E_1 which is exactly the entire cipher but with the first round removed. Since there is a bijective mapping between (v_0, v_{-1}) and (v_1, v_0), and it is easily computable in both directions with the knowledge of K_0, i.e., it is easy to compute the plaintext (v_1, v_0) of E_1 from the plaintext (v_0, v_{-1}) of the original Feistel-2 cipher and vice versa, one will not have problem choosing the plaintext to E_1.

To attack the cipher with one less round, the attack complexities can be significantly lower, which eventually becomes negligible compared with that for the original attacks. For example, in our 6-round key-recovery attack for Feistel-2 with $k = n$ as in Section 3, besides other choices, the attacker can re-use the results in [17] on 5-round Feistel-2 to recover $K_1, K_2, \ldots, K_5$ in time $2^{n/2}$, compared with $2^{3n/4}$ for 6-round attack to recover K_0. Hence, complexities for recovering all subkeys remain unchanged.

For the extended attacks to larger key sizes, one can always prepend or append less rounds to the distinguisher and result in faster attacks when K_0 is recovered, using similar attack procedure, for the cases where reduced cipher is of more than 6 rounds.

Feistel-3. In the case of Feistel-3, the same method can be applied, except now it is the last round subkey which is recovered instead of the K_0. So here, instead of the first round, we remove the last round, and with the knowledge of the subkey of the last round, there is an easily computable bijective mapping between the ciphertext of the original cipher and ciphertext of the reduced cipher.

D Experimental verification of the attacks on Feistel-2

To check the correctness of our attacks we have implemented and applied them to small state ciphers. The ciphers were designed to follow the generic constructions and the state size was chosen to make the attacks feasible in real time. The codes were written in `C++` and complied with `g++` on Linux. The source code for the two experiments are available here for 6 rounds: http://www1.spms.ntu.edu.sg/~syllab/attacks/F2-6rounds.tar.gz, and there for 8 rounds: http://www1.spms.ntu.edu.sg/~syllab/attacks/F2-8rounds.tar.gz.

We first implemented the 6-round key recovery attack on Feistel-2. The outputs of the round functions were chosen as random values produced from a pseudo random number generator, and saved in tabular form. We tested the attack on Feistels with key and block sizes of 24 bits, i.e. $k = n = 24$, while the value of x' (as suggested by the attack) was set to 6, i.e. $x' = n/4 = 6$. We ran 1000 experiments for this cipher, and in each of them the subkeys were chosen at random. The program produced the following output:

```
$ ./Feistel_2_key_recovery_6_rounds
#Experiments: 1000
Values for ( n , xp , b ):  ( 24 , 6 , 2 )
[+] Precomputation phase:
    |T_delta| expected =   262144 = 2^18
    |T_delta| real     =   260772 = 2^18.0
[+] Collecting pairs:
    |Pairs  | expected =     4096 = 2^12
    |Pairs  | real     =     3996 = 2^12.0
[+] Recovery of K_0
    Percentage of recovered keys : 59
    Percentage of correct keys   : 100
```

Obviously the size of T_δ and the number of the collected pairs matches the estimates predicted by the attack. Furthermore, in 59% of the cases the collected pairs were able to reveal the secret key, and in such cases, the attack succeeded with 100% rate. For the courtesy, in our second series of 1000 experiment we took twice larger data set, and in this case around 90% of the keys were recovered, again with 100% success rate. Finally, when the data was four times larger than the initially suggested, the former percentage increased to 97%. Thus we can conclude that the 6-round key recovery is correct.

We then tested the extension of the above attack to 8 rounds as described in the paper. The round function was chosen similarly, the state size was set to 18 bits, and the key size to 27 bits, i.e. $n = 18, k = 27$. As the suggested value of x' is $n/3$, we took $x' = 6$. We ran 100 experiments, and the resulting output was as follow:

```
./Feistel_2_key_recovery_8_rounds
#Experiments: 100
Values for ( n , xp , b ):  ( 18 , 6 , 3 )
[+] Precomputation phase:
    |T_delta| expected = 16777216 = 2^24
    |T_delta| real     = 16331824 = 2^24.0
[+] Collecting pairs:
    |Pairs  | expected =   262144 = 2^18
    |Pairs  | real     =   130962 = 2^17.0
[+] Recovery of K_0
    Percentage of recovered keys : 52
    Percentage of correct keys   : 100
```

The results were similar as in the case above. The small deviation in the number of predicted pairs comes from the fact that 2^d data results in around 2^{2d-1} pairs, thus instead of 2^{18} pairs, in practice we have obtained 2^{17}, which in turn explains the smaller percentage of recovered keys. Once we took twice larger data set, the percentage increased to 81%. Thus the attack works as predicted.