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 round function, we differentiate and show attacks on two types of Feistels. For the first type, which is the most general Feistel, we show a 5-round distinguisher (based on a truncated differential), which allows to launch 6-round 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.

Keywords: 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 the encryption and the 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 [6], Triple-DES [19], Camellia [2], and CAST [5] are Feistels. In addition to the standard block ciphers, the Feistel construction is an attractive choice for many lightweight ciphers, for instance the recent NSA proposal SIMON [3], LBlock [26], Piccolo [24], etc. The application of the Feistel construction is not limited only to ciphers, and has been used to design other crypto primitives: the hash function SHAvite-3 [4], the CAESAR proposal for authentication scheme LAC [27] and others.

The analysis of Feistel primitives and their provable security bounds depend on the type of the round function implemented. Luby and Rackoff [21] have

P. Sarkar and T. Iwata (Eds.): ASIACRYPT 2014, PART I, LNCS 8873, pp. 458–477, 2014.

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 keys. Later, Knudsen [20] 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 [18] 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 an XOR of a subkey followed by an application of a public function or permutation. Generic attacks on Feistel-2 such as impossible differentials [20], all-subkey recovery [17,18], and integral-like attacks [25] penetrate up to 6 rounds when the key size equals the state size, and up to 9 rounds when the key is twice as large as the block. Better attacks have been published, but they are on so-called *Feistel-3* that has round functions based on 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 [18] reach up to 7 rounds for 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 meet-in-the-middle cryptanalytic technique. Its most basic form corresponds to the textbook case of Double-DES [22] and in the past few years, a few improvements have been proposed to more specific cases, for instance, Dinur et al. [11] have generalized the attack on Double-DES when multiple encryption (more than two n-bit keys) is used. Besides the applications to preimage attacks on hash functions [1, 16, 23], a notable application of the meet-in-the-middle technique and a line of research that has been started by Demirci and Selçuk [8] 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 [9, 10]. 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. The attacker detects correct subkey guesses by checking meet-in-the-middle equations during the encryption process. Namely, the attack starts with a precomputation phase which is used to fully tabulate the distinguishing behavior particular to the targeted cipher, e.g. `AES`, and later in the online phase, the attacker searches for messages verifying the distinguisher by checking the precomputed table.

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 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 is indeed the meet-in-the-middle part of the attack.

In the case of the Feistel-2 construction, the number of rounds that our attacks can reach 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 relies on the meet-in-the-middle strategy, the complexities of these two phases are 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 type of the round functions: they can be any invertible or one-way functions or permutations, unique for each round. What we assume, however, is that the round functions have standard differential behavior. That is, given a large set of input-output differences of these functions (which can be seen as a set of differentials), on average for each differential there is one solution that conforms to it.

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 when key size is more than $2n$ bits. To reach more rounds compared to Feistel-2, we use the SPN structure of the round function in both the offline and 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 by an additional round. Other 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 only one round less.

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

Target	Round functions	#rounds and complexity						Reference
		k = n		**k = 3n/2**		**k = 2n**		
Feistel-2	bijective	5	$2^{3n/4}$	6	2^{n}	7	$2^{3n/2}$	[20]
	—	3	$2^{n/2}$	5	2^{n}	7	$2^{3n/2}$	[17]
	—	5	$2^{n/2}$	7	$2^{5n/4}$	9	$2^{3n/2}$	[18]
	bij., ident.	6	$2^{n/2}$	—	—	—	—	[25]
	—	**6**	$2^{3n/4}$	**8**	$2^{4n/3}$	**10**	$2^{11n/6}$	**Section 3**
Feistel-3	—	7	$2^{3n/4+c}$	9	2^{n+c}	11	$2^{7n/4+c}$	[18]
	—	**9**	$2^{n/2+4c}$	**11**	2^{n+4c}	**13**	$2^{3n/2+4c}$	**Section 4**
	identical	**10**	$2^{n/2+4c}$	**12**	2^{n+4c}	**14**	$2^{3n/2+4c}$	**Section 4**

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, 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 an all-subkey recovery. We have also experimentally confirmed the validity of our analysis on the case of small state Feistel-2[1]. The experiments ran on a regular PC supported the complexity evaluation and the correctness of the attacks. 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.

Due to space constraints, in the sequel, we present only our main ideas that result in 6-round attack on Feistel-2 and 10-round attack on Feistel-3. The full version of the paper, including additional attacks, the technique to recover all the subkeys and the experimental results can be found in [15].

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 description of the attacks, we omit the network twist in the last round as it has not cryptographic significance.

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

[1] The interested reader can find the implementations of our attacks at `http://www1.spms.ntu.edu.sg/~syllab/attacks/F2-6rounds.tar.gz` and `http://www1.spms.ntu.edu.sg/~syllab/attacks/F2-8rounds.tar.gz`.

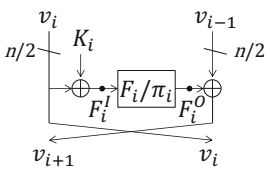

Fig. 1. Feistel-2

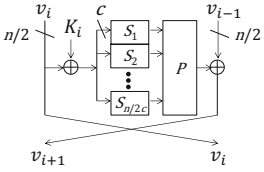

Fig. 2. Feistel-3

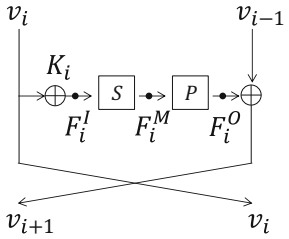

Fig. 3. Simplified Feistel-3

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[2] 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 produces 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.

Solutions of Differential Equations. In our analysis, we make the following assumption on the non-linear round functions F_i of the Feistel cipher. We assume that given a large set of fixed input and output differences of F_i, i.e. $(\Delta_{I_j}, \Delta_{O_j}), j = 1, 2, \ldots$, then on average there is one solution of each of the differential equations $F_i(X \oplus \Delta_{I_j}) \oplus F_i(X) = \Delta_{O_j}, j = 1, 2, \ldots$. That is, some of the equations may have many solutions and some none, however, we assume that on average (over a large set) the number of solution is one per equation. This requirement is sufficient for our analysis, as we solve the differential equations for a large number of (Δ_I, Δ_O), thus we can take the average case which is one solution per equation. Our computer simulations of the attacks confirmed this expectation and the complexity of the attacks was as predicted by our analysis, in part because the aforementioned assumption is true in the case of randomly chosen (Feistel-2 and Feistel-3) non-linear round functions. There are examples of round functions[3] where the assumption does not hold, for instance, linear

[2] 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.

[3] We do not claim attacks on Feistel-2 that have this type of round functions.

functions[4]. However, to the best of our knowledge, such round functions are either not used as building blocks of ciphers, or they can be attacked using other, more trivial attacks.

It is important to notice that although one solution is expected, it does not mean that it can be found trivially. To solve most of the equations, we use precomputation tables, i.e. we tabulate the functions, store their values, and later perform table lookups to solve the differential equations.

Definition 1 (δ-Set, [7]). *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.*

We introduce slightly modified definition (without byte-oriented sets).

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 from [7] 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 $F(v) \oplus F(v \oplus 1), F(v) \oplus F(v \oplus 2), \ldots, F(v) \oplus F(v \oplus 2^b - 1)$.

The Attack Model. The key-recovery attacks presented in the paper follow the standard attack model. That is, the key of the block cipher is secret and chosen uniformly at random. The attacker can query both the encryption and the decryption functions of the block cipher. His task is to recover the secret key (or the subkeys produced from the key schedule) based on the queries. We explicitly state that the attacker has no information about the internal state values of the block cipher.

3 Key-Recovery Attacks against Feistel-2 Construction

In this section, we present a key-recovery attack on 6-round Feistel-2 ciphers for the case when the key and the state sizes are equal, i.e. $k = n$. The extensions of the attack to 8 rounds for $k = 3n/2$, 10 rounds for $k = 2n$, and in general to $(4 + 2s)$ rounds for $k = n(s + 1)/2$, can be found in the full version of the paper [15]. In our attack, the round functions can be either bijective or non-bijective, i.e. permutations or functions, and they can even be one-way. To make

[4] For linear function, the probability that a solution exist depends on the size of the large set.

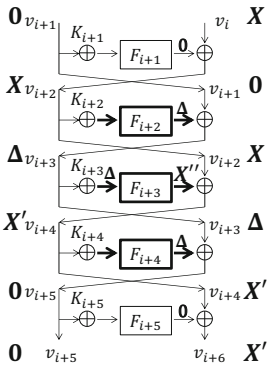

Fig. 4. 5-round differential characteristic

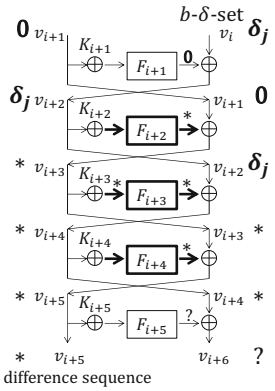

Fig. 5. b-δ-set construction

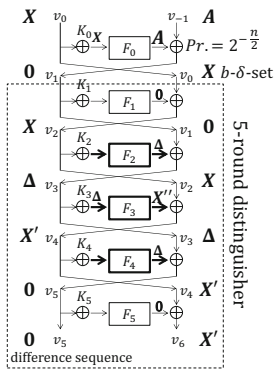

Fig. 6. 6-round key-recovery

the attack applicable to the most general type of constructions, in the sequel, we assume that the round functions are one-way and pairwise distinct.

We use F_i to denote the round function at round i of the construction. To refer to the input (resp. output) of 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.

The 6-round key-recovery attack is based on a non-ideal behavior of 5 rounds of Feistel-2, which is described by the lemma and the proposition that follow. In the 6-round attack (refer to Figure 6), the last five rounds are the rounds where this distinguisher is used.

Lemma 1. *Let X and X', where $X \neq X'$, be two non-zero branch differences. If a 5-round Feistel-2 encrypts a pair of plaintexts (m, m') with difference $0\|X$ to a pair of ciphertexts with difference $0\|X'$, then the number of possible internal state values of the three middle rounds that correspond to the plaintext m is limited to $2^{n/2}$ on average.*

Proof. Note that $n/2$-bit round keys are added in each round, and hence the number of possible internal state values for the three middle rounds is limited by its size, $2^{3n/2}$. We show, however, that the bound can be tightened to $2^{n/2}$.

A 5-round differential characteristic, with input difference $0\|X$ and output difference $0\|X'$ is depicted in Figure 4 (the rounds are denoted from $i+1$ to $i+5$ to make this part of the analysis generic). From the figure, we can see that after the first round, the input difference $(0, X)$ must become a state difference $(X, 0)$. Similarly, after the inversion of the last round the output difference $(0, X')$ becomes $(0, X')$. This makes $\Delta F_{i+3}^{\mathcal{O}}$ to be $X'' \leftarrow X \oplus X'$. Since $X \neq X'$, it follows that $X'' \neq 0$ and thus $\Delta F_{i+3}^{\mathcal{I}} \neq 0$ – let us denote this difference with Δ. It means that both $\Delta F_{i+2}^{\mathcal{O}}$ and $\Delta F_{i+4}^{\mathcal{O}}$ also have the difference Δ. To summarize, we get that for each fixed Δ, the input and output differences of the round functions at rounds $i+2$, $i+3$, and $i+4$ are fixed. Therefore, there exists one state value (one solution) that satisfies such input-output difference in each of the three rounds.

As Δ can take at most $2^{n/2}$ different values (one branch has $n/2$ bits), the states in rounds $i+2$, $i+3$, $i+4$ can assume only $2^{n/2}$ different values. In Figure 4, the fixed value for each Δ is drawn by bold line. □

We use Lemma 1 to prove the below proposition that will help us later to launch the attack on 6 rounds. To present the proposition, we need additional notations. Let $F : m \to F(m)$ be a 5-round Feistel-2 (we omit writing the key k as input) and let the function $F^{\Delta} : \{0,1\}^{\frac{3n}{2}} \to \{0,1\}^{\frac{n}{2}}$ be defined as $F^{\Delta}(m,\delta) = Trunc_{n/2}\Big(F(m) \oplus F(m \oplus (0\|\delta))\Big)$, where $Trunc_{n/2}$ denotes the truncation to the first $n/2$ bits. In other words, $F^{\Delta}(m,\delta)$ gives the output difference (of the left branch) in the pair of ciphertexts, produced by encryption of a pair of plaintexts $(m, m \oplus 0\|\delta)$ with the 5-round Feistel. Furthermore, instead of taking a single pair of plaintexts, let us create several pairs such that in each pair, the first element is always m, while the second is $m \oplus 0\|\delta_j$ where $\delta_j = 1, \ldots, 2^b - 1$ (the precise value of b is defined later in the section). In fact, we can see that the second elements of the pairs form a b-δ-sequence. The proposition given further claims that the sequence of differences in the ciphertexts pairs (that correspond to such plaintexts pairs) can take only $2^{n/2}$ values.

Proposition 1. *Let (m, m') be a pair of plaintexts that conforms to the 5-round differential characteristic given in Figure 4 and let $\delta_j = 1, \ldots, 2^b - 1, b \geq 1$ forms b-δ-sequence. Then, the sequence $F^{\Delta}(m, \delta_j)$, $\delta_j = 1, \ldots, 2^b - 1$ can assume only $2^{n/2}$ possible values.*

Remark 1. We note that the sequence can be constructed from any of the two plaintexts m or m' given in Proposition 1, as long as the pair (m, m') conforms to the differential characteristic.

Remark 2. From a theoretical point of view, Proposition 1 yields a distinguisher since the number of functions reached by the 5-round Feistel-2 construction is much less than the theoretical number of functions from a set of 2^b elements to a set of $2^{n/2}$ elements when $b \geq 1$. Indeed, for a fixed m, the latter equals $(2^{n/2})^{2^b} = 2^{2^b n/2}$, whereas it is only $2^{n/2}$ in the case of the 5-round Feistel-2 construction.

Proof. The initial pair of plaintexts (m, m') is only used to compute the state values of the three middle rounds that correspond to the plaintext m. We have seen from Lemma 1 that these three states can take only $2^{n/2}$ possible values (each of them corresponds to one of the values of Δ). We will show that if the values of these three states are fixed, then we can change the right half of the plaintext (instead of m, we take $m \oplus 0\|\delta_j$) and still be able to compute the output difference in the left half of the ciphertexts. In fact, we can change the value of the plaintext many times (i.e. we can produce many pairs of the form $(m, m \oplus 0\|\delta_j)$), and for each of them, we can easily compute the output difference in the right halves of the ciphertext. The number of plaintexts pairs adds no complexity in predicting the ciphertext difference – once the three middle states are fixed (and they can have only $2^{n/2}$ different values), the sequence of differences in the ciphertext pairs is uniquely determined.

Assume the difference Δ is fixed[5], and thus are fixed the three internal state values. Let $t_{i+2}, t_{i+3}, t_{i+4}$ be the input values to $F_{i+2}, F_{i+3}, F_{i+4}$ that correspond to the plaintext m, in which $t_{i+2}, t_{i+3}, t_{i+4}$ are determined depending on Δ. Let v_i be the values of the states that correspond to the plaintext m as shown in Fig. 5. Let us consider a new pair of plaintexts, $(m, m \oplus (0 \| \delta_j))$, i.e. we introduce a difference δ_j to the right branch, i.e. $\Delta v_i = \delta_j$. Since the difference $\Delta F^{\mathcal{O}}_{i+1}$ is always zero, 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 the 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 computable difference with '$*$'. 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 $i+3$. As we know the value of $F^{\mathcal{I}}_{i+3} = t_{i+3}$ and $\Delta F^{\mathcal{I}}_{i+3} = \Delta F^{\mathcal{O}}_{i+2}$, it follows that $(t_{i+3}, t_{i+3} \oplus \Delta F^{\mathcal{O}}_{i+2})$ 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 an arbitrary δ_j, we can compute the output difference Δv_{i+5}, i.e., the mapping from δ_j to Δv_{i+5} becomes deterministic (as long as Δ is fixed). Therefore, for the ordered sequence of δ_j that takes the values $1, 2, \ldots, 2^{n/2} - 1$, we can determine the sequence of corresponding differences Δv_{i+5} (which indeed is the difference in the left half of the ciphertext). We emphasize that the mapping depends only on values of $t_{i+2}, t_{i+3}, t_{i+4}$, which in turn are determined from the value of Δ, X and X', and acts independently of the value of m. Since Δ takes at most $2^{n/2}$ values, the number of sequences of Δv_{i+5} is limited to $2^{n/2}$. □

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. The online phase is further divided into collecting pair and key recovery phases. In the precomputation phase, we choose many pairs (X, X'), where X is fixed while X' takes multiple values, and for each pair, we find all possible $2^{n/2}$ sequences of Δv_5 based on Proposition 1. We store all the sequences in a large table along with its corresponding internal state values. Next, in the online phase, we collect many pairs that satisfy one of the differential characteristics $(X, 0) \rightarrow (X', 0)$. Finally, for each of the obtained pairs, we compute Δv_5 sequences by 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 – this allows us to determine the internal states and to recover K_0. The meet-in-the-middle nature of our attack comes from the fact that the Δv_5 sequence is computed offline for the last

[5] Recall that this difference corresponds to an internal state difference for the plaintext pair (m, m').

five rounds and online for the first round, and the results are later matched in a meet-in-the-middle-like fashion.

Precomputation. From Proposition 1, the number of possible sequences of Δv_5 is $2^{n/2}$ for a fixed X and a fixed X'. We can achieve a time/memory tradeoff by relaxing the $n/2$-bit constraint of a fixed X' and allow $2^{x'}$ 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 (MSBs). 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 to compute all $2^{x'} \cdot 2^{n/2} = 2^{x'+n/2}$ sequences of 2^b differences as an offline precomputation in $2^{x'+n/2+b}$ time (encryptions), and $2^{x'+n/2+b}$ memory (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 and thus we will have a 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 in the attack we need to invert it, we cannot compute F_3^{-1} to construct T_3. Thus, we first 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 $\Delta F_2^{\mathcal{O}} = \Delta F_3^{\mathcal{I}} = \Delta F_4^{\mathcal{O}}$), 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 $F_0^{\mathcal{I}}$, the corresponding $\Delta F_0^{\mathcal{O}}$ 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', building T_0, T_2, T_3 and T_4 requires $2^{n/2}$ round function computations each. Hence, constructing T_δ requires less[6] 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 build 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 construct 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 fixed to a constant random value v_0 on their left half, 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 have 2^n pairs of plaintexts such that the difference in the left half equals X and the right half is nonzero.

For a single structure, the data complexity corresponds to 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 $n/2 - x'$ most significant bits of the right half. 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 with the use of 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

[6] Less, as one evaluation of the round functions costs less than one encryption query.

Algorithm 1. Construction of the tables T_2 and T_4

1: **for** $i = 0, 1, \ldots, 2^{n/2} - 1$ **do**
2: Compute $\Delta F_2^{\mathcal{O}} \leftarrow F_2(i) \oplus F_2(i \oplus X)$.
3: Store $(i, \Delta F_2^{\mathcal{O}})$ in T_2 indexed by $\Delta F_2^{\mathcal{O}}$.
4: Compute $\Delta F_4^{\mathcal{O}} \leftarrow F_4(i) \oplus F_4(i \oplus X')$
5: 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

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

Algorithm 3. Construction of the sequences of Δv_5

1: **for** $\Delta = 1, \ldots, 2^{n/2} - 1$ **do**
2: 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.
3: **for** all b active bits of the b-δ-set **do**
4: Modify Δv_0, and compute the corresponding Δv_5.
5: Compute the sequence of Δv_5 and add it to T_δ.

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

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

corresponding ciphertexts. Hence, we can compute the sequence of Δv_5. If this sequence is included in the precomputation table T_δ, K_0 is a 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. The data complexity is $2^{n-x'+1}$ 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, the choice of $x' = n/4$ 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 complexity to $2^{3n/4}$ blocks of $n/2$ bits.

4 Key-Recovery Attacks against Feistel-3 Construction

In this section, we present a 10-round key-recovery attack on the Feistel-3 construction with $k = n$. In the attack, we assume that different S-Boxes are used for different words in a given round, but we consider they are the same across all of the 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 extensions of the attack to 12 and 14 rounds for key sizes of $k = 3n/2$ and $k = 2n$, respectively, and the analysis of a class of P-layers that not necessarily has a maximal branch number are given in the full version of the paper [15].

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. To construct the distinguisher, we first apply an equivalent transformation to the 7-round primitive, as shown in Figure 7. 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 one. 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.

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.

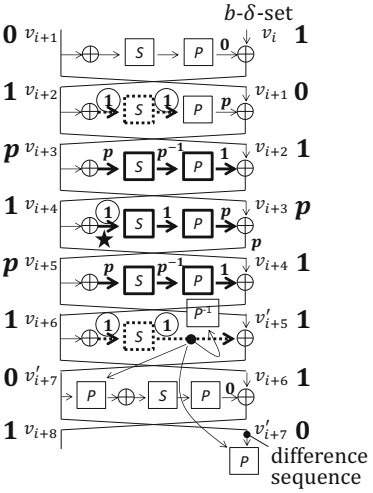

Fig. 7. 7-round differential

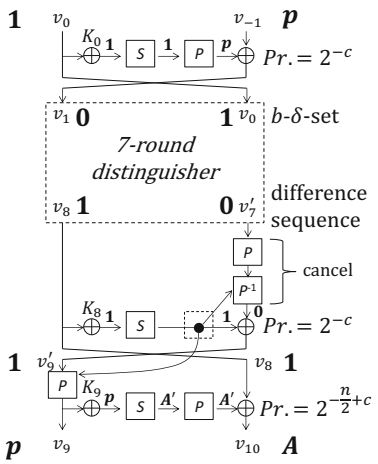

Fig. 8. 10-round key-recovery for $k = n$

Finally, $X[\mathbf{1}]$ and $\Delta X[\mathbf{1}]$, respectively, denote the pre-specified active-word value and difference of a branch-wise variable X.

The technique used to construct the 7-round distinguisher (described in the proposition below) is very similar to the technique we have used in the distinguisher on five rounds of Feistel-2. In other words, first we show that if a pair (m, m') of plaintexts follows a particular differential characteristic, then the number of possible internal state values that correspond to m is limited. Based on this, we can introduce a difference in the plaintext and predict the output difference in the ciphertext. Again, we introduce many pairs of plaintexts where each right half differs on δ_j (and thus get a b-δ-sequence) and observe that the pairs of ciphertexts have predictable difference. Unlike the proposition for Feistel-2 where we observed the difference in the left half of the ciphertext, for Feistel-3, we check the difference in one word of the right half in the ciphertext pairs (the position of this particular word plays no role in the analysis). That is why we have to redefine $F^{\Delta}(m, \delta_j)$. To avoid bulky notations, we define it informally as one-word difference in the right half of the ciphertext pair that are produced from the encryption of a plaintext pair $(m, m \oplus 0\|\delta_j)$ through 7-round Feistel-3. In Figure 7, this is the ciphertext difference in the word v'_{i+7}.

Proposition 2. *Let (m, m') be a pair of plaintexts that conforms to the 7-round differential $(\mathbf{0}, \mathbf{1}) \xrightarrow{7R} (\mathbf{1}, \mathbf{0})$ shown in Figure 7 and let $\delta_j = 1, 2, \ldots 2^b - 1$ forms a b-δ-sequence. Then, the sequence $F^{\Delta}(m, \delta_j)$, $\delta_j = 1, \ldots, 2^b - 1$ can assume only $2^{n/2+4c}$ possible values.*

Proof. We show here that the number of internal state values for pairs satisfying the 7-round differential in Figure 7 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 7), and by the values of $n/2-c$ inactive bits of $F^{\mathcal{I}}_{i+4}$ (marked by a star '★' in Figure 7).

We first assume that the five word differences circled in Figure 7 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 7, 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 7 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 7 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 can still 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. Let us describe the 10-round key-recovery attack that uses the 7-round distinguisher. As shown in Figure 8, we extend the 7-round differential characteristic of the distinguisher by one round at the beginning and two rounds at the end (the analysis and complexity would be similar if we extend by two rounds at the beginning and one at the end). Recall that the additional P-layer 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 the position has to be fixed beforehand to be able to conduct the attack. The P-layer in round 8 is moved to two different positions as shown in Figure 8. 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 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. Storing T_δ requires $2c/n \times 2^{n/2+4c+b}$ blocks of $n/2$ bits, as the sequences contains 2^b elements of c bits.

Collecting Pairs. To launch the attack, we need a pair that satisfies the 7-round differential characteristic in Figure 7, 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,

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.
4: Derive all state values in rounds 3, 4 and 5.
5: **for** 2^b different differences in v_0 **do**
6: Modify $\Delta v_0[\mathbf{1}]$, and update the corresponding sequence of $\Delta v_7'[\mathbf{1}]$.
7: 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 analysis 2^{n-4c} times by changing the the inactive-bit value of v_0 and v_t.

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 the differential characteristic 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 possible inputs for each of these S-Boxes can be reduced to a single value, and the corresponding subkeys $K_0[\mathbf{1}]$, $K_8[\mathbf{1}]$ 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 with high probability, 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. It is possible to achieve 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 one 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]. If we add a second active word at the beginning of the differential characteristic, it 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 the second active word requires $2^{14n/16}$ chosen plaintexts, $2^{14n/16}$ encryptions and the memory of about $2^{12n/16}$ blocks of $n/2$ bits, hence the overall complexity is $2^{14n/16}$. For the same primitive, but with an additional active word, the tradeoff gives an attack that requires the same overall time complexity while the data complexity is reduced to $2^{13n/16}$ chosen plaintexts.

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 very small restrictions on the round functions, our attacks are applicable to almost 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. We have produced experimental verification of the attacks supporting our claims.

Our results give insights on the lower bound on the number of rounds a secure Feistel should have. They suggest that this number in the case of SPN round functions should be surprisingly high. Furthermore, from the attacks on Feistel-2, we show that as long as the ratio of key to state size is increasing, the number of rounds that can be 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.

Acknowledgments. The authors would like to thank the ASIACRYPT 2014 reviewers for their valuable comments. Jian Guo, Jérémy Jean and Ivica Nikolić are supported by the Singapore National Research Foundation Fellowship 2012 NRF-NRFF2012-06.

References

1. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for Step-Reduced SHA-2. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 578–597. Springer, Heidelberg (2009)
2. 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. (eds.) SAC 2000. LNCS, vol. 2012, pp. 39–56. Springer, Heidelberg (2001)
3. 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)
4. Biham, E., Dunkelman, O.: The SHAvite-3 Hash Function. Submission to NIST, Round 2 (2009)
5. Communications Security Establishment Canada: Cryptographic algorithms approved for Canadian government use (2012)
6. Coppersmith, D.: The Data Encryption Standard (DES) and its Strength Against Attacks. IBM Journal of Research and Development 38(3), 243–250 (1994)
7. Daemen, J., Knudsen, L.R., Rijmen, V.: The Block Cipher SQUARE. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
8. Demirci, H., Selçuk, A.A.: A Meet-in-the-Middle Attack on 8-Round AES. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 116–126. Springer, Heidelberg (2008)
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, 477 (2012)

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 2013. LNCS, vol. 7881, pp. 371–387. Springer, Heidelberg (2013)
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 2012. LNCS, vol. 7417, pp. 719–740. Springer, Heidelberg (2012)
12. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 158–176. Springer, Heidelberg (2010)
13. Feistel, H., Notz, W., Smith, J.: Some Cryptographic Techniques for Machine-to-Machine Data Communications. Proceedings of IEEE 63(11), 15545–1554 (1975)
14. Gilbert, H., Minier, M.: A Collision Attack on 7 Rounds of Rijndael. In: AES Candidate Conference, pp. 230–241 (2000)
15. Guo, J., Jean, J., Nikolić, I., Sasaki, Y.: Meet-in-the-Middle Attacks on Generic Feistel Constructions - Extended Abstract. Cryptology ePrint Archive, Temporary version (to appear, 2014), http://www1.spms.ntu.edu.sg/~syllab/attacks/FeistelMitM.pdf
16. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010)
17. Isobe, T., Shibutani, K.: All Subkeys Recovery Attack on Block Ciphers: Extending Meet-in-the-Middle Approach. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 202–221. Springer, Heidelberg (2013)
18. Isobe, T., Shibutani, K.: Generic Key Recovery Attack on Feistel Scheme. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part I. LNCS, vol. 8269, pp. 464–485. Springer, Heidelberg (2013)
19. ISO/IEC 18033-3:2010: Information technology–Security techniques–Encryption Algorithms–Part 3: Block ciphers (2010)
20. Knudsen, L.R.: The Security of Feistel Ciphers with Six Rounds or Less. J. Cryptology 15(3), 207–222 (2002)
21. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM J. Comput. 17(2), 373–386 (1988)
22. Merkle, R.C., Hellman, M.E.: On the Security of Multiple Encryption. Commun. ACM 24(7), 465–467 (1981)
23. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
24. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: *Piccolo*: An Ultra-Lightweight Blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)
25. Todo, Y.: Upper Bounds for the Security of Several Feistel Networks. In: Boyd, C., Simpson, L. (eds.) ACISP. LNCS, vol. 7959, pp. 302–317. Springer, Heidelberg (2013)
26. Wu, W., Zhang, L.: LBlock: A Lightweight Block Cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)
27. Zhang, L., Wu, W., Wang, Y., Wu, S., Zhang, J.: LAC: A Lightweight Authenticated Encryption Cipher. Submitted to the CAESAR competition (March 2014)