Generic Key Recovery Attack on Feistel Scheme

Takanori Isobe and Kyoji Shibutani

Sony Corporation
1-7-1 Konan, Minato-ku, Tokyo 108-0075, Japan
{Takanori.Isobe,Kyoji.Shibutani}@jp.sony.com

Abstract. We propose new generic key recovery attacks on Feistel-type block ciphers. The proposed attack is based on the all subkeys recovery approach presented in SAC 2012, which determines all subkeys instead of the master key. This enables us to construct a key recovery attack without taking into account a key scheduling function. With our advanced techniques, we apply several key recovery attacks to Feistel-type block ciphers. For instance, we show 8-, 9- and 11-round key recovery attacks on n-bit Feistel ciphers with $2n$-bit key employing random keyed F-functions, random F-functions, and SP-type F-functions, respectively. Moreover, thanks to the meet-in-the-middle approach, our attack leads to *low-data complexity*. To demonstrate the usefulness of our approach, we show a key recovery attack on the 8-round reduced CAST-128, which is the best attack with respect to the number of attacked rounds. Since our approach derives the lower bounds on the numbers of rounds to be secure under the single secret key setting, it can be considered that we unveil the limitation of designing an efficient block cipher by a Feistel scheme such as a low-latency cipher.

Keywords: block cipher, key scheduling function, all-subkeys-recovery attack, meet-in-the-middle attack, key recovery attack, low-data complexity attack

1 Introduction

A block cipher is considered as an essential technology on modern cryptography, since it is one of the most widely used primitives. Moreover, studies on designing a secure and efficient block cipher are useful also for designing other symmetric primitives such as hash functions and stream ciphers. Since DES was developed in 1977 [19], a lot of progress has taken place in this area. Recently, with the large deployment of network devices requiring security, block ciphers satisfying new demands such as lightweight and low-latency have received a lot of attention. In fact, several block ciphers designed for a lightweight hardware implementation have been proposed such as PRESENT [9], KATAN/KTANTAN [16], LED [20] and Piccolo [32]. The concept of a low-latency encryption, which is used for an application requiring an instant response, was discussed in [24]. Since a low-latency encryption requires a quick response, the number of rounds must be reduced as much as possible compared to a general-purpose block cipher such as AES. In 2012, PRINCE was proposed as an instantiation of a low-latency cipher [12]. Note that PRINCE is not only a low-latency cipher, but also a lightweight block cipher even after supporting both encryption and decryption. Those features are considered to be important in practical use of the cipher, since its lightweightness directly leads to low power and energy consumption and supporting decryption function without much cost leading to this cipher being used more widely.

In general, an SPN cipher requires an inverse function when supporting decryption, and thus an SPN cipher with a decryption function needs additional gate areas. In spite of the fact that PRINCE is an SPN cipher, it is efficiently implemented even when implementing a decryption function due to its novel property called α-reflection. However, as pointed out by the designers, it has been known that α-reflection reduces the security of the cipher [12, 23, 33] and thus the cipher having α-reflection does not have optimal security. Meanwhile, it has been known that a Feistel cipher, another traditional structure of block cipher, is suitable for a lightweight block cipher especially when supporting both encryption and decryption, since it does not require an inverse

Table 1. Numbers of Attacked Rounds by Generic Attacks on Feistel Schemes

Single Secret Key Setting				
Attack Type	Feistel-1		Feistel-2	Feistel-3
Distinguisher	5 [28]		5 [28]	5 [28]
	5* [25]		5* [25]	5* [25]
	5* [11]		5* [11]	5* [11]
Key Recovery Attack ($k = 2n$)	7 [22]	**8** (Ours)	**9** (Ours)	**11** (Ours)
Key Recovery Attack ($k = 3n/2$)	5 [22]	**6** (Ours)	**7** (Ours)	**9** (Ours)
Key Recovery Attack ($k = n$)	3 [22]	**4** (Ours)	**5** (Ours)	**7** (Ours)
Known Key Setting				
Distinguisher	not given		7 [26]	11* [31]

* : Each F function is restricted to a permutation

function. Thus, a Feistel cipher is considered as a possible candidate of a low-latency cipher, if it has sufficiently small number of rounds. However, it has been still unknown how many rounds are sufficient for a Feistel cipher to be secure. Note that, for low-latency encryption, since the key scheduling function can be precomputed, it can be a heavy function. Thus, its performance with respect to low-latency is considered to mainly depend on the data processing part, namely its number of rounds. Hence, our question is "how many rounds can be reduced without loss of security requirements for Feistel schemes".

In this paper, we tackle the security evaluations of several Feistel schemes, assuming that the key scheduling function is an ideal function. We deal with key recovery attacks under the single secret key setting by extending the all subkeys recovery approach [22]. Since our approach derives the lower bounds on the numbers of rounds to be secure against a key recovery attack even if the underlying key scheduling function is an ideal function, our results show the limitation of designing a low-latency encryption by a Feistel scheme. We introduce several advanced techniques including *function reduction* and *key linearization*. Using those advanced techniques and with the help of the meet-in-the-middle approach [10, 21], we show several key recovery attacks on various Feistel ciphers. Table 1 summarizes the number of attacked rounds for Feistel schemes by both distinguishers and key recovery attacks under the single secret key and known-key settings. Compared to the previous results, some of our attacks are the first generic key recovery attacks and also the best for several Feistel schemes with respect to the number of attacked rounds, even if the attacker is allowed to use the known secret key. Moreover, our attack does not restrict the underlying F-function to a permutation, which is a limitation of some of the previous attacks. Furthermore, one of the advantages of our approach is its low data requirement thanks to the meet-in-the-middle approach, in contrast to the classical statistical attacks such as an impossible differential attack [6]. As an example for the practical impact of our work, we show the best attack on the reduced CAST-128 [1] even when its key scheduling function is ideal. Also, we show extremely low-data attacks on the reduced Camellia [5] with less than 60 data sets.

This paper is organized as follows: Section 2 gives notations and definitions used throughout this paper, and gives a brief review of the all subkeys recovery approach. We review the related work and show its improvement in Section 3. Our key recovery attacks on two types of Feistel ciphers and those applications to CAST-128 and Camellia are described in Sections 4 and 5. Section 6 discusses the usefulness of our attack. Finally, we conclude in Section 7.

2 Preliminary

In this section, we give notations used throughout this paper, then define our target Feistel ciphers. Finally, we briefly review the all subkeys recovery approach presented in [22].

2.1 Notation

The following notation will be used throughout this paper:

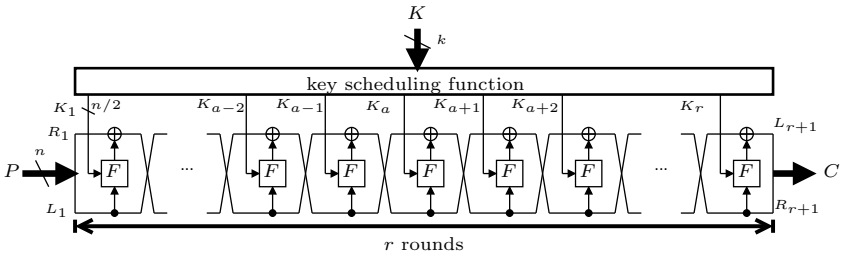


Fig. 1. Balanced Feistel Network (Feistel-1)

n	: block size.
k	: the size of the master key.
L_i, R_i	: left or right half of the i-th round input.
K_i	: the i-th round subkey ($n/2$ bits).
ℓ	: the size of an S-box.
m	: the number of S-boxes in an S-box layer.
X_i	: the i-th round state.
$X_{i,j}$	: the j-th S-box word (ℓ-bit data) of X_i.
X_{iL}, X_{iR}	: left or right half bits of X_i.
$a\|b$ or $(a\|b)$	: Concatenation.

2.2 Feistel Cipher

In this paper, we focus on balanced Feistel networks as illustrated in Fig. 1. An n-bit plaintext P is divided into two sub-blocks as $P = (L_1, R_1)$, where $L_i, R_i \in \{0,1\}^{n/2}$. Then the $(i+1)$-th round input state is calculated as follows:

$$(L_{i+1}, R_{i+1}) \leftarrow (R_i \oplus \mathcal{F}_i^{K_i}(L_i), L_i),$$

where $\mathcal{F}_i^{K_i} : \{0,1\}^{n/2} \rightarrow \{0,1\}^{n/2}$ is a keyed function in the i-th round using the i-th round $(n/2)$-bit subkey K_i. An n-bit ciphertext C for the r-round encryption function is derived as $C = (R_{r+1}, L_{r+1})$. Note that the last round of the Feistel cipher does not have a swap operation. Hereafter, the size of each subkey used in one round is assumed to be half of the block size (i.e., $K_i \in \{0,1\}^{n/2}$).

In this work, we deal with three types of Feistel block ciphers illustrated in Fig. 2. Feistel-1 denotes the Feistel cipher with random keyed F-functions. Each subkey is assumed to be randomly independent. Thus each keyed F-function is also independent from each other. In concrete ciphers, each subkey is usually XORed before an F-function. Feistel-2 reflects such ciphers. In other words, the output of the F-function $Y_i = \mathcal{F}_i^{K_i}(X_i)$ is represented as $Y_i = F_i(X_i \oplus K_i)$, where F_i is a fixed function in the i-th round (not limited to a permutation). Similarly, Feistel-3 is the Feistel-2 cipher whose F_i is limited to an SP-type F-function, where each F-function consists of a bijective S-box layer (S-layer) and a linear diffusion layer (P-layer), and an $n/2$-bit subkey is XORed before the S-box layer. Each S-box layer consists of m ℓ-bit S-boxes (i.e., $m \cdot \ell = n/2$), and each P-layer consists of an $m \times m$ linear matrix represented as M_i. Note that Feistel-1 includes Feistel-2 and Feistel-3, also Feistel-3 is a subset of Feistel-2. The size of the master key is denoted as Feistel-$[k]$. For example, Feistel-2$[n]$ is the Feistel cipher with fixed F-functions XORed by a subkey before the function whose master key size is the same as the block size (e.g., a 128-bit block cipher taking a 128-bit key).

2.3 All Subkeys Recovery Approach [22]

The all subkeys recovery (ASR) attack was proposed by Isobe and Shibutani at SAC 2012 [22]. The ASR attack is considered as an extension of the meet-in-the-middle (MITM) attack, which mainly exploits a low key-dependency in the key scheduling function. The basic concept of the ASR attack is guessing all subkeys instead of the master key so that the attack can be constructed independently

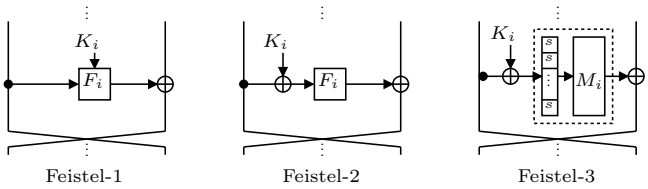


Fig. 2. Target Feistel Ciphers

from the structure of the key scheduling function, by regarding all subkeys as independent variables. Thus the attack can also be applied to a block cipher having a complex key scheduling function.

Let us briefly review the procedure of the ASR attack. In the ASR attack, an attacker first determines a t-bit matching state X, where $X \in \{0,1\}^t$. In the forward direction, the matching state derived from a plaintext P and a set of subkeys $\mathcal{K}_{(1)}$ by a function $\mathcal{F}_{(1)}$ is represented as $X = \mathcal{F}_{(1)}(P, \mathcal{K}_{(1)})$. Similarly, the state computed from a ciphertext C and another set of subkeys $\mathcal{K}_{(2)}$ by a function $\mathcal{F}_{(2)}$ in the backward direction is denoted as $X = \mathcal{F}^{-1}_{(2)}(C, \mathcal{K}_{(2)})$. $\mathcal{K}_{(3)}$ denotes a set of the remaining subkeys not required for computing X, i.e., $|\mathcal{K}_{(1)}|+|\mathcal{K}_{(2)}|+|\mathcal{K}_{(3)}| = r \cdot n/2$. The attacker guesses $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ in parallel, then checks if the equation $\mathcal{F}_{(1)}(P, \mathcal{K}_{(1)}) = \mathcal{F}^{-1}_{(2)}(C, \mathcal{K}_{(2)})$ holds. Note that the equation holds when the guessed subkey bits are correct. After this process, it is expected that there will be $2^{r \cdot n/2 - t}$ key candidates. Finally, the attacker exhaustively searches the correct key from the surviving key candidates. The required computations of the attack in total C_{comp} using N plaintext/ciphertext pairs is estimated as

$$C_{comp} = \max(2^{|\mathcal{K}_{(1)}|}, 2^{|\mathcal{K}_{(2)}|}) \times N + 2^{r \cdot n/2 - N \cdot t}. \tag{1}$$

The number of required plaintext/ciphertext pairs is $\max(N, \lceil (r \cdot n/2 - N \cdot t)/n \rceil)$. The required memory is about $\min(2^{|\mathcal{K}_{(1)}|}, 2^{|\mathcal{K}_{(2)}|}) \times N$ blocks, which is the cost of the table used for the matching. Clearly, the ASR attack works faster than the brute force attack when Eq.(1) is less than 2^k, which is the required computations for the brute force attack.

3 Generic Key Recovery Attack on Feistel-1

In this section, we first review key recovery attacks on balanced Feistel networks presented in [22] and generalize it to Feistel-1[n], -1[$\frac{3}{2}n$] and -1[$2n$]. After that, we show that the basic attack can be improved by using *splice and cut* [3] and *key linearization* techniques. By the improved attack, the numbers of attacked rounds for the Feistel-1 are increased by one round.

For a Feistel-1 cipher, an $(n/2)$-bit matching state X is computed from a plaintext P and a set of subkeys $\mathcal{K}_{(1)} \in \{K^{(1)}, K^{(2)}, ..., K^{(a-1)}\}$ as shown in Fig. 1 (i.e., $X = \mathcal{F}_{(1)}(P, \mathcal{K}_{(1)})$). Similarly, the matching state is obtained from a ciphertext C and another set of subkeys $\mathcal{K}_{(2)} \in \{K^{(a+1)}, K^{(a+2)}, ..., K^{(r)}\}$ as $X = \mathcal{F}^{-1}_{(2)}(C, \mathcal{K}_{(2)})$. Also, X is computed independently from an $(n/2)$-bit subkey $K^{(a)}$, i.e., $\mathcal{K}_{(3)} \in \{K^{(a)}\}$.

3.1 Basic Attack on Feistel-1 [22]

For Feistel-1[$2n$] (e.g., a 128-bit block cipher accepting a 256-bit key), 7 rounds of the cipher can be attacked in a straightforward manner, since $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$ are composed of 3 rounds of the cipher and thus the sizes of $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ are both $3 \cdot n/2$ bits. In this attack, the total time complexity C_{comp} using four plaintext/ciphertext pairs is estimated as

$$C_{comp} = \max(2^{3n/2}, 2^{3n/2}) \times 4 + 2^{7 \cdot n/2 - 4 \cdot n/2} \approx 2^{3n/2+2} \quad (= 2^{3k/4+2})$$

The required memory is about $4 \times 2^{3n/2}$ blocks. Since C_{comp} is less than $2^{2n} (= 2^k)$ when $(4 < n)$, the attack works faster than the exhaustive key search.

Similarly to this, for Feistel-1[$\frac{3}{2}n$] and Feistel-1[n] (e.g., a 128-bit block cipher accepting a 192-bit key or a 128-bit key), key recovery attacks of at least 5 and 3 rounds of the cipher are

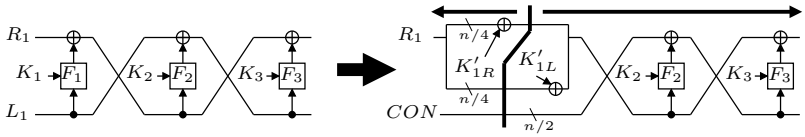


Fig. 3. Splice and Cut Technique for Feistel-1

constructed, respectively. For Feistel-1[$\frac{3}{2}n$], $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$ consist of 2 rounds of the cipher, and thus the sizes of $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ are both n bits. Therefore, the required time complexity using 3 plaintext/ciphertext pairs is estimated as $C_{comp} = \max(2^n, 2^n) \times 3 + 2^{5n/2-3n/2} \approx 2^{n+2}$, and the required memory is about 2^{n+2} blocks. For Feistel-1[n], a similar attack on 3 rounds requiring $2^{n/2+1}$ ($\approx 2^{n/2} \times 2 + 2^{n/2}$) computations and $(2 \times 2^{n/2})$ blocks memory is mounted by using 1 round of $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$. Roughly speaking, when Eq.(1) is less than 2^k, the ASR attack works faster than the brute force attack. Therefore, the necessary condition for the basic ASR attack is that each size of all subkeys in $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$ is less than the size of the master key.

3.2 Improved Attack on Feistel-1

We demonstrate that the basic attack on Feistel-1 presented in [22] is improved by controlling the value of plaintexts. It allows us to attack one more round on Feistel-1, e.g., an 8-round attack on Feistel-1[$2n$].

Suppose that an input $L_1(= R_2)$ is fixed to an arbitrary $(n/2)$-bit constant CON, then L_2 is expressed as $L_2 = R_1 \oplus K'_1$, where $K'_1 = F_1(K_1 \oplus CON)$. Since K'_1 depends only on K_1, it is regarded that a new $(n/2)$-bit subkey K'_1 is linearly inserted in the first round without an F-function, which is called key linearization.

As shown in Fig. 3, since K'_1 can be divided into two $(n/4)$-bit words K'_{1L} and K'_{1R}, the splice and cut technique in [4] enables us to separately use K'_{1L} and K'_{1R} in $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$, respectively. Note that, in the splice and cut technique, the MITM attack starts from multiple values of start states for parallel guesses of $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$, while the basic MITM attack starts from multiple plaintext/ciphertext pairs.

For Feistel-1[$2n$], an 8-round generic key recovery attack is mounted thanks to the splice and cut technique, while each cost (namely time, memory and data) for the attack is increased by $\mathcal{O}(2^{n/4})$ compared to the basic attack. The size of each key set $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ is increased by $(n/4)$ bits due to the splice and cut, and thus the size of each set $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ is $7n/4 (= 3 \cdot n/2 + n/4)$ bits long. In this attack, the total time complexity C_{comp} using five start states is estimated as

$$C_{comp} = \max(2^{7n/4}, 2^{7n/4}) \times 5 + 2^{8 \cdot n/2 - 5 \cdot n/2} \approx 2^{7n/4+3} \quad (= 2^{7k/8+3}).$$

The required memory is about $5 \times 2^{7n/4}$ blocks. Since $(n/4)$ bits of plaintexts are varied depending on $\mathcal{K}_{(2)}$ and the start states, the required data is $2^{n/4}$ chosen plaintexts when the other $3n/4$ bits of the start state are fixed.

For Feistel-1[$\frac{3}{2}n$] and Feistel-1[n], by using the splice and cut technique, key recovery attacks of at least 6 and 4 rounds of the cipher are constructed, respectively. For Feistel-1[$\frac{3}{2}n$], the sizes of $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ are $5n/4$ bits each. Therefore, the required time complexity with four start states is estimated as $C_{comp} = \max(2^{5n/4}, 2^{5n/4}) \times 4 + 2^{6n/2-4n/2} \approx 2^{5n/4+2}$, and the required memory is about $2^{5n/4+2}$ blocks. For Feistel-1[n], a similar attack requiring $2^{3n/4+2}$ ($\approx 2^{3n/4} \times 3 + 2^{n/2}$) computations and $(3 \times 2^{3n/4})$ blocks memory is mounted. These attacks also require $2^{n/4}$ chosen plaintexts. Those results are summarized in Table 2.

4 Key Recovery Attack on Feistel-2

This section shows generic key recovery attacks on Feistel-2 ciphers. In contrast to Feistel-1 ciphers, key injections of Feistel-2 ciphers are restricted to XOR operations. This allows an attacker to equivalently transform subkeys, then more rounds can be attacked. To begin with, we introduce an advanced technique called *function reduction*, which enables us to reduce the number of

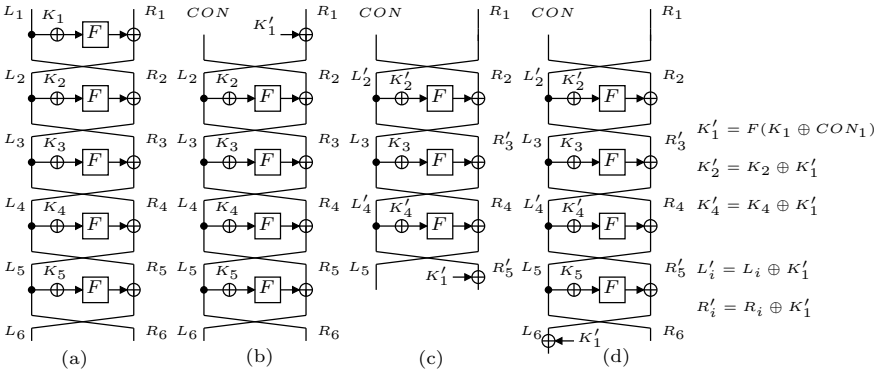


Fig. 4. Function Reduction Technique

involved subkey bits by exploiting degrees of freedom of a plaintext/ciphertext pair. Combining it with a (multi-)collision technique, 5, 7 and 9 rounds attacks on Feistel-2[n], -2[$\frac{3}{2}n$] and -2[$2n$] are demonstrated, respectively. The overview of the function reduction is depicted in Fig. 4. The required complexities for those attacks are summarized in Table 2, and the overview of the attacks are illustrated in Fig. 5. Note that the key additions of Feistel-2 are limited to XOR operations, however, similar idea may be applied to other key additions such as modular additions. Moreover, as an application of our approach on Feistel-2, we show a key recovery attack on the reduced CAST-128 [1, 2]. The structure of CAST-128 is similar to Feistel-2, however, the size of each round key of CAST-128 is larger than that of Feistel-2 and the key additions are not only XOR operations but also modular additions and subtractions. Since the larger round key generally requires more computations to guess, it seems to be hard to directly mount an attack on CAST-128. We use the improved function reduction technique to make an attack feasible, then show a key recovery attack on the 8-round reduced CAST-128, which is the best attack known in literature.

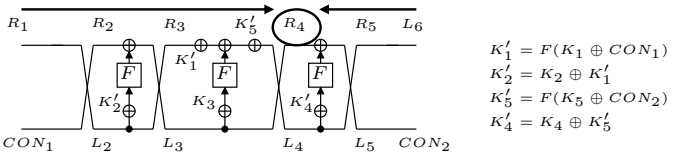
4.1 Function Reduction Technique

Suppose that the half outputs of the r-round Feistel-2 cipher L_{r+1} and R_{r+1} are represented by functions $\mathcal{F}_{L,r}$ and $\mathcal{F}_{R,r}$ as $L_{r+1} = \mathcal{F}_{L,r}(\mathcal{K}_L, L_1|R_1)$ and $R_{r+1} = \mathcal{F}_{R,r}(\mathcal{K}_R, L_1|R_1)$, where $\mathcal{K}_L$ and $\mathcal{K}_R$ denote sets of subkeys used in $\mathcal{F}_{L,r}$ and $\mathcal{F}_{R,r}$, respectively. In general, after sufficient number of round operations, all subkeys are required to compute L_{r+1}, i.e., $|\mathcal{K}_L| = n/2 \cdot r$, while R_{r+1} is derived independently from the last subkey K_r, i.e., $|\mathcal{K}_R| = n/2 \cdot (r-1)$. For the Feistel-2 cipher, fixing half bits of inputs, one more round of subkey data can be reduced as follows:

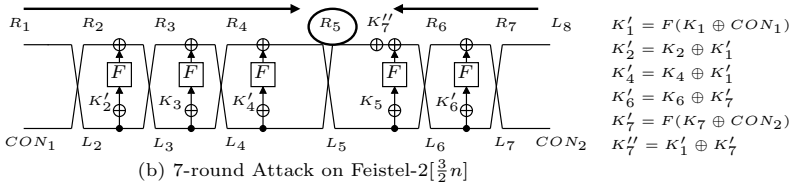
Theorem 1 (Function Reduction). For the Feistel-2 cipher, if L_1 is fixed, $\mathcal{K}_L$ and $\mathcal{K}_R$ used in $\mathcal{F}_{L,r}$ and $\mathcal{F}_{R,r}$ contain at most $(n/2 \cdot r)$ and $(n/2 \cdot (r-2))$ subkey bits when r is odd, and contain at most $(n/2 \cdot (r-1))$ and $(n/2 \cdot (r-1))$ subkey bits when r is even, respectively.

Proof. By using the key linearization, L_2 is considered to be linearly affected by the subkey K'_1 as follows. Assuming that L_1 is an arbitrary $(n/2)$-bit constant CON, L_2 and R_2 are expressed as $L_2 = R_1 \oplus K'_1$ and $R_2 = CON$, where $K'_1 = F(K_1 \oplus CON)^\star$. Since K'_1 depends only on K_1, it can be regarded as a new subkey instead of K_1 (see Fig. 4-(b)). By using an equivalent transform, K'_1 is moved to the end of the cipher as shown in Figs. 4-(c) and (d). After the transform, each subkey introduced in even round is XORed with K'_1, and thus it can be redefined as $K'_p = K_p \oplus K'_1$ (p is even). When r is even, K'_1 is linearly affecting to R_{r+1} in the last as shown in Fig. 4-(c). Therefore, both L_{r+1} and R_{r+1} contain at most $(n/2 \cdot (r-1))$ bits of subkeys. When r is odd, K'_1 is linearly affecting to L_{r+1} in the last as shown in Fig. 4-(d). Consequently, R_{r+1} contains at most $(n/2 \cdot (r-2))$ bits of subkeys, while the amount of subkey bits required for computing L_{r+1} is not reduced (i.e., $|\mathcal{K}_L| = n/2 \cdot r$). □

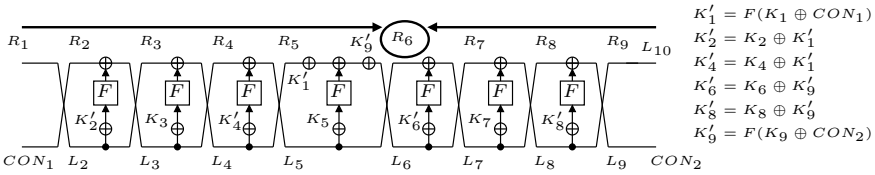
* For simplicity, we assume that all F-functions are identical. However, our attack works even if each F-function is distinct from each other.



(a) 5-round Attack on Feistel-2[n]



(b) 7-round Attack on Feistel-2[$\frac{3}{2}n$]



(c) 9-round Attack on Feistel-2[$2n$]

Fig. 5. Key Recovery Attacks on Feistel-2 Ciphers

The function reduction technique, which consists of equivalent transforms of round keys and the key linearization, is related to the complementation properties of Feistel networks in which the round keys of even (or odd) rounds are complemented by some fixed values. It essentially exploits the property of Feistel network that an input of a keyed F-function in the i-th round (L_i) linearly affects an input of a keyed F-function in the $(i+2)$-th round (L_{i+2}). In other words, the relation of L_i and L_{i+2} is expressed as $L_{i+2} = L_i \oplus X_{i+1}$, where X_{i+1} is an output of an F-function of (L_{i+1}). We exploit it in the line of a MITM attack to reduce the subkey data for the computation of the intermediate values, while the previous attacks are used for differential attacks [15, 8] and speeding up keysearches using equivalent keys [7, 18].

4.2 Key Recovery Attack on 5-round Feistel-2[n]

In order to apply the function reduction to both the forward and backward computations, we prepare plaintext/ciphertext pairs in the form of $L_1 = CON_1$ and $R_6 = CON_2$, where CON_1 and CON_2 denote arbitrary $(n/2)$-bit constants.

Let R_4 be an $(n/2)$-bit matching state. From Theorem 1, in the forward computation, R_4 can be computed by an $(n/2)$-bit subkey $K'_2 (= K_2 \oplus K'_1)$, where $K'_1 = F(K_1 \oplus CON_1)$. In the backward computation, since R_4 can be regarded as an output of the even round ($r = 2$), R_4 can also be computed by an $(n/2)$-bit subkey $K'_4 (= K_4 \oplus K'_5)$, where $K'_5 = F(K_5 \oplus CON_2)$, i.e., $\mathcal{K}_{(1)} \in K'_2$ and $\mathcal{K}_{(2)} \in K'_4$. Since $|\mathcal{K}_{(1)}| = |\mathcal{K}_{(2)}| = n/2$ and the size of the matching state is also $n/2$, two plaintext/ciphertext pairs are sufficient to determine $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$. In order to obtain such two pairs that have the form of $L_1 = CON_1$ and $R_6 = CON_2$, we use $2^{n/4}$ chosen plaintexts by randomly changing R_1 as $P = (CON_1|R_1)$. After this process, we have $2^{n/4}$ corresponding ciphertexts, and thus there will exist $(n/2)$ bits colliding R_6 with high probability due to the birthday paradox.

The time complexity of determining K'_2 and K'_4 by the MITM approach is estimated as $C_{comp} = \max(2^{n/2}, 2^{n/2}) \times 2 = 2^{n/2+1}$. In order to determine all subkeys, we use the following equation: $F(R_4 \oplus K_3) = R_1 \oplus K'_1 \oplus L_6 \oplus K_5 = R_1 \oplus L_6 \oplus K''_1$, where $K''_1 = K'_1 \oplus K_5$. Since R_4 can be computed from K'_2 or K'_4, we can recursively mount the MITM approach to determine K_3 and K''_1 with complexity of $2^{n/2+1} (= \max(2^{n/2}, 2^{n/2}) \times 2)$. After exhaustively guessing K_1 with a time complexity of $2^{n/2}$, all subkeys K_i $(1 \le i \le 5)$ are determined from the previously obtained subkeys K'_2, K'_4 and K''_1. Therefore, the whole time complexity is estimated as $2^{n/2+2} (\approx 2^{n/2+1} + 2^{n/2+1} + 2^{n/2})$. Due to $k = n$, the time complexity $2^{n/2+2} = 2^{k/2+2}$ is less than 2^k which is required computations

Table 2. Details of Our Attacks

Target	key size	Round	Time	Memory	Data	Reference
Feistel-1	n	4	$2^{3n/4+2}$	$2^{3n/4+2}$	$2^{n/4}$	Sect. 3
	$\frac{3}{2}n$	6	$2^{5n/4+2}$	$2^{5n/4+2}$	$2^{n/4}$	Sect. 3
	$2n$	8	$2^{7n/4+3}$	$2^{7n/4+3}$	$2^{n/4}$	Sect. 3
Feistel-2	n	4	$2^{n/2+2}$	$2^{n/2+1}$	2	Sect. 4.2
		5	$2^{n/2+2}$	$2^{n/2+1}$	$2^{n/4}$	Sect. 4.2
	$\frac{3}{2}n$	6	$2^{5n/4+4}$	2^{n+4}	9	Sect. 4.4
		7	$2^{5n/4+4}$	2^{n+4}	$9!^{1/9} \cdot (2^{n/2})^{8/9}$	Sect. 4.4
	$2n$	8	$2^{3n/2+3}$	$2^{3n/2+3}$	6	Sect. 4.3
		9	$2^{3n/2+3}$	$2^{3n/2+3}$	$6!^{1/6} \cdot (2^{n/2})^{5/6}$	Sect. 4.3
Feistel-3	n	7	$2^{3n/4+\ell} \cdot N_1$	$2^{3n/4+\ell} \cdot N_1$	N_1	Sect. 5.3
	$\frac{3}{2}n$	8	$2^{n+\ell} \cdot N_2$	$2^{n+\ell} \cdot N_2$	N_2	Sect. 5.5
		9	$2^{n+\ell} \cdot N_2$	$2^{n+\ell} \cdot N_2$	$N!^{1/N_2} \cdot (2^{n/2})^{(N_2-1)/N_2}$	Sect. 5.5
	$2n$	11	$2^{7n/4+\ell} \cdot N_3$	$2^{7n/4+\ell} \cdot N_3$	N_3	Sect. 5.4

$N_1 = (3n/2 + 2\ell)/\ell$, $N_2 = (2n + 2\ell)/\ell$, $N_3 = (7n/2 + 2\ell)/\ell$

for the brute force attack. The required data is $2^{n/4}$ chosen plaintext, and the required memory is about $2^{n/2+1}$ words. If the function reduction technique is used only in the forward computation, a 4-round attack is constructed with less data (see Fig. 5-(a) and Table 2).

4.3 Key Recovery Attack on 9-round Feistel-2[2n]

A key recovery attack on a 9-round Feistel-2[2n] is constructed in a similar way to the 5-round attack on Feistel-2[n]. In this attack, we can add 2 more rounds in each direction, and a 6-multicollision is required to obtain desired plaintext/ciphertext pairs unlike the attack on Feistel-2[n]. It has been known that an n-bit t-multicollision is found in $t! \cdot 2^{n \cdot (t-1)/t}$ random data with high probability [34]. Thus, the six plaintext/ciphertext pairs whose form are $P = (CON_1|R_1)$ and $C = (CON_2|L_{10})$ could be found from $6!^{1/6} \cdot (2^{n/2})^{5/6} \approx 3 \cdot (2^{n/2})^{5/6}$ chosen plaintexts. More precisely, after querying $3 \cdot (2^{n/2})^{5/6}$ chosen plaintexts with distinct R_1, there will exist a 6-collision of R_{10} in corresponding ciphertexts with high probability (see Fig. 5-(c) and Table 2).

4.4 Key Recovery Attack on 7-round Feistel-2[$\frac{3}{2}n$]

In this attack, R_5 is used as the matching state. From Theorem 1, in the forward computation, R_5 can be computed from $3 \cdot n/2$ bits subkeys K'_2, K_3 and K'_1, where $K'_2 = K_2 \oplus K'_1$ and $K'_1 = F(K_1 \oplus CON_1)$. In the backward computation, R_5 can be computed from $3 \cdot n/2$ bits subkeys K'_6, K_5 and K'_7, where $K'_6 = K_6 \oplus K'_7$ and $K'_7 = F(K_7 \oplus CON_2)$. Since R_5 is expressed as $K'_1 \oplus L_4$ and $K'_7 \oplus (F(R_6 \oplus K_5) \oplus L_6)$, if only $(n/4)$ bits of $K'_1 \oplus K'_7$ are guessed, $(n/4)$-bit matching is feasible. It is regarded that $K''_7 (= K'_1 \oplus K'_7)$ is included in the backward computation (see Fig. 5-(b)).

Then, since $|\mathcal{K}_{(1)}| = n/4$, $|\mathcal{K}_{(2)}| = 5n/4$, and the size of the matching state is $n/4$, nine plaintext/ciphertext pairs are required to determine $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ due to the relation $(n+5n/4)/(n/4) = 9$. Such nine plaintext/ciphertext pairs whose form are $P = (CON_1|R_1)$ and $C = (CON_2|L_8)$ can be found from $9!^{1/9} \cdot (2^{n/2})^{8/9} \approx 4.2 \cdot (2^{n/2})^{8/9}$ chosen plaintexts. The other complexities required for this attack and the low data attack on 6-round Feistel-2[$\frac{3}{2}n$] are described in Table 2.

4.5 Application to 8-Round Reduced CAST-128

In order to demonstrate the practical impact of our work on Feistel-2, we apply it to CAST-128 block cipher. Using the improved function reduction techniques, we show an attack on the 8-round reduced CAST-128 having more than 118 bits key, which is the best attack with respect to the number of attacked rounds in literature even when its key scheduling is an ideal function.

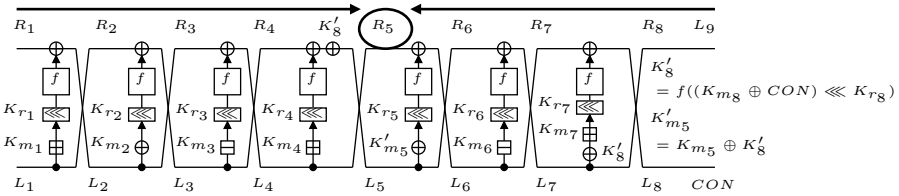


Fig. 6. Key Recovery Attack on 8-Round CAST-128

Description of CAST-128. CAST-128 [1, 2] is a 64-bit Feistel block cipher accepting a variable key size from 40 up to 128 bits (but only in 8-bit increments). The number of rounds is 16 when the key size is longer than 80 bits. First, the algorithm divides the 64-bit plaintext into two 32-bit words L_0 and R_0, then the i-th round function outputs two 32-bit data L_{i+1} and R_{i+1} as follows:

$$L_{i+1} = R_i \oplus F_i(L_i, K_i^{rnd}), R_{i+1} = L_i,$$

where F_i denotes the i-th round function and K_i^{rnd} is the i-th round key consisting of a 32-bit masking key K_{m_i} and a 5-bit rotation key K_{r_i}. The detail of F_i is expressed as

$$F_i = f((L_i \bigcirc_i K_{m_i}) \lll K_{r_i}),$$

where f consists of four 8 to 32-bit S-boxes, $\lll K_{r_i}$ denotes a K_{r_i}-bit left rotation, and $\bigcirc_i$ denotes addition, XOR or subtraction depending on the round number i, i.e., $\bigcirc_i$ denotes addition for $i \in \{1, 4, 7, 10, 13\}$, XOR for $i \in \{2, 5, 8, 11, 14\}$ and subtraction for $i \in \{3, 6, 9, 12, 15\}$. We omit the details of f, since, in our analysis, it is regarded as the random function that outputs a 32-bit random value from a 32-bit input.

Key Recovery Attack on 8-round CAST-128. The structure and the parameter of CAST-128 having sufficiently large key are similar to Feistel-2[2n]. However, for CAST-128, a 37(= 32+5)-bit subkey is inserted into each F_i, i.e., a 32-bit subkey is used in $\bigcirc_i$ and the remaining 5-bit subkey is used in a key dependent rotation, while a 32-bit subkey is inserted in each round for Feistel-2[2n] with $n = 32$. Thus, the 9-round attack on Feistel-2[2n] is not directly applicable to CAST-128. However, the improved function reduction technique allows us to construct an 8-round attack on CAST-128.

Let R_5 be an $(n/2)$-bit matching state. In the backward computation, R_9 is fixed as CON, and $K'_8 = f((CON \oplus K_{m_8}) \lll K_{r_8})$ is moved to L_5 and an input of the 7-th round function, by converting K_{m_5} into $K'_{m_5} = K'_8 \oplus K_{m_5}$, as shown in Fig. 6. Then, the input of f in the 7-th round is expressed as $(L_9 \oplus K'_8) + K_{m_7}$. If the lower b bits of L_9, which are controllable by the ciphertext, are fixed to 0, the lower b bits of this computation are expressed as $K'_8 + K_{m_7}$. Thus, $(K'_8 + K_{m_7})$ is regarded as a new b-bit subkey $K'_{m_7} = (K'_8 + K_{m_7})$, while the upper $(n/2 - b)$ bits remain $(L_9 \oplus K'_8) + K_{m_7}$. In the backward computation of R_5, $|\mathcal{K}_{(2)}| = 37 \times 2 + (b + (n/2 - b) \times 2 + 5)$ bits of the key are involved.

Evaluation. Since $|\mathcal{K}_{(1)}| = 111$, $|\mathcal{K}_{(2)}| = 114$ ($b = 29$) and the size of the matching state is 32 bits, eight plaintext/ciphertext pairs are required to determine $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ due to the relation $(111 + 114)/32 < 8 (= 2^{32-29})$. The required time complexity to determine subkeys K_1^{rnd}, K_2^{rnd}, K_3^{rnd}, K_6^{rnd}, K'_{m_5}, K_{r_5}, the lower 29 bits of K'_{m_7}, the upper 3 bits of K'_8, and K_{m_7} is estimated as $C_{comp} = \max(2^{111}, 2^{114}) \times 10 \approx 2^{118}$. The remaining K_{r_4} and K'_8 are exhaustively searched with the time complexity of 2^{64}. Then, all subkeys are obtained by using the relations of $K'_{m_7} = K'_8 + K_{m_7}$, $K'_{m_5} = K'_8 \oplus K'_{m_5}$ and $K'_8 = f((CON \oplus K_{m_8}) \lll K_{r_8})$. The required data is eight chosen ciphertexts, and the required memory is 2^{111} words. Therefore, when the key size is more than 118 bits long, our attack works faster than the brute force attack.

5 Key Recovery Attack on Feistel-3

This section presents generic key recovery attacks on Feistel-3 ciphers. Feistel-3 ciphers are the Feistel-2 ciphers whose F-functions are restricted to be SP-type F-functions, which consist of an

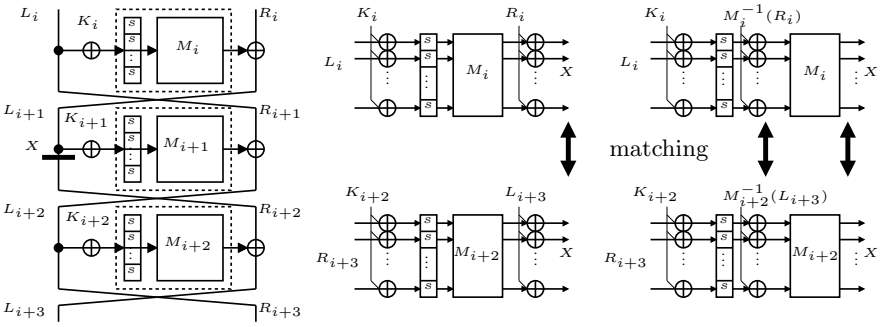


Fig. 7. Matching without Matrix

S-box layer followed by a linear matrix operation. This allows an attacker to exploit a linearity of a matrix computation, and thus the number of attacked rounds can be increased. To begin with, we review two techniques which exploit a linearity of a matrix computation. We refer those two techniques as *matching without matrix* and *matrix separation* to make our explanation simple. However, those techniques have already been introduced, for example, in [27, 30]. Combining them with a (multi-)collision technique and *function reduction*, 7, 9 and 11 rounds attacks on Feistel-3[n], -3[$\frac{3}{2}n$] and -3[$2n$] are demonstrated, respectively. Furthermore, as an application of our approach on Feistel-3, we show several key recovery attacks on the reduced Camellia [5]. Since Camellia is a Feistel cipher with SP-type F-functions, our attack on Feistel-3 can be directly applied to it even if its key scheduling function is ideal. Besides, the number of attacked rounds by our attack is further increased by one round for Camellia due to its non-MDS matrix. Consequently, we present generic key recovery attacks requiring extremely low data on the 8-, 10-, 12-round reduced Camellia without FL/FL^{-1} functions and key whitenings.

5.1 Matching without Matrix [30]

Let us consider three consecutive rounds of the Feistel-3 cipher whose input and output are represented as $(L_i|R_i)$ and $(L_{i+3}|R_{i+3})$ as shown in Fig. 7. Assuming that an attacker knows those input and output variables, the following equation holds:

$$F_i(L_i \oplus K_i) \oplus R_i = F_{i+2}(R_{i+3} \oplus K_{i+2}) \oplus L_{i+3}. \tag{2}$$

In order to check if the equation holds, we need to guess $2{\cdot}n/2$ bits subkeys K_i and K_{i+2}, while K_{i+1} is not needed to be guessed. However, if F-functions are SP-type F-functions (i.e., $F_i = M_i \circ S_i$, where M_i and S_i denote an $m \times m$ matrix and an S-box layer consisting of m ℓ-bit S-boxes, respectively), the size of guessing subkey bits can be reduced by exploiting the linearity of the matrix operation. Since M_i is a linear function, Eq.(2) is redescribed as:

$$M_i(S_i(L_i \oplus K_i)) \oplus R_i = M_{i+2}(S_{i+2}(R_{i+3} \oplus K_{i+2})) \oplus L_{i+3},$$
$$M_i(S_i(L_i \oplus K_i) \oplus M_i^{-1}(R_i)) = M_{i+2}(S_{i+2}(R_{i+3} \oplus K_{i+2}) \oplus M_{i+2}^{-1}(L_{i+3})).$$

When $M_i = M_{i+2}$, we have

$$S_i(L_i \oplus K_i) \oplus M_i^{-1}(R_i) = S_{i+2}(R_{i+3} \oplus K_{i+2}) \oplus M_{i+2}^{-1}(L_{i+3}). \tag{3}$$

Unlike Eq.(2), we can separately check if Eq.(3) holds by the size of the S-box ℓ. Therefore, this technique enables us to reduce the number of subkey bits to be guessed for the 3-round matching from $2^{2\cdot n/2}$ to $2^{2\ell}$. When $M_i \neq M_{i+2}$, the matching technique called matching through matrix presented in [29] is utilized. In this case, more than $m \cdot \ell$ bits subkeys are required to be guessed. For simplicity, from now on, we assume that $M_i = M_{i+2}$.

In the function reduction, the modified subkey K_1' affects L_{2t} and R_{2t-1} $(t = 1, 2, ...)$. Also, in the matching without matrix, we utilize the relation of L_{i+1} as the matching state. This implies that if $(i + 1)$ is even (i.e., $(i + 1) = 2t$), L_{i+1} is affected by K_1' and it cannot be used as the matching state. Therefore, if the matching without matrix is used with the function reduction, the starting round of the matching i must be even (i.e., $(i + 1)$ must be odd).

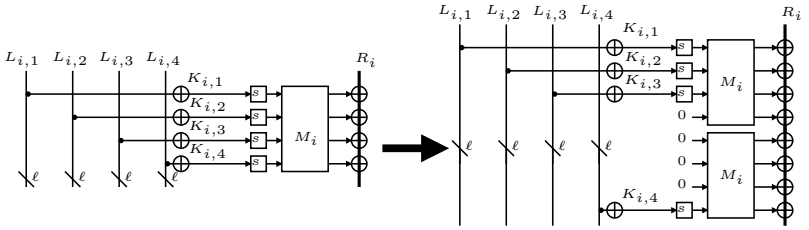


Fig. 8. Matrix Separation

5.2 Matrix Separation [27]

In general, for the function reduction technique, all inputs of an F-function are needed to be fixed. However, in the Feistel-3 ciphers, the (partial) function reduction is constructed by fixing only a part of inputs due to the linearity of the matrix. This technique referred as matrix separation in this paper gives more degrees of freedom to the inputs.

Since M_i is a linear operation, each operation can be divided by ℓ bits. For instance, we show the case of $m = 4$ as an example (see Fig. 8). Suppose that $K_i = (K_{i,1}|K_{i,2}|K_{i,3}|K_{i,4})$, $K_{i,j} \in \{0,1\}^\ell$ and $L_i = (L_{i,1}|L_{i,2}|L_{i,3}|L_{i,4})$, $L_{i,j} \in \{0,1\}^\ell$. If three input words $L_{i,1}$, $L_{i,2}$ and $L_{i,3}$ are fixed, only $3/4 \times n/2$ bits of K_i are linearly inserted into the $(i+1)$-round by regarding $T = M(S'((L_{i,1} \oplus K_{i,1})|(L_{i,2} \oplus K_{i,2})|(L_{i,3} \oplus K_{i,3}))|\mathbf{0}^\ell)$ as new subkey bits, where S' consists of three S-boxes and $\mathbf{0}^\ell$ denotes ℓ bits of 0. Note that T is an $(n/2)$-bit data, however, it is determined by $(3/4 \cdot n/2)$ bits subkeys $K_{i,1}$, $K_{i,2}$ and $K_{i,3}$. Since $L_{i,4}$ is not fixed, $M(\mathbf{0}^{3/4 \cdot n/2}|s(L_{i,4} \oplus K_{i,4}))$ is non-linearly inserted into the $(i+1)$-th round.

5.3 Key Recovery Attack on 7-round Feistel-3[n]

For the 7-round Feistel-3[n], it seems that the function reduction is applied to both directions and the matching without matrix is used in the rounds 3 to 5. However, this approach does not work due to the restriction of the combination of the matching without matrix and the function reduction. To overcome this problem, we utilize the partial function reduction in conjunction with the matching without matrix.

At first, L_1 is fixed as CON_1, and $K'_1 = F(K_1 \oplus CON_1)$ is moved to R_5 by converting K_2 and K_4 into $K_2 \oplus K'_1$ and $K_4 \oplus K'_1$, respectively. In addition, R_{1L}, which is the left half of R_1 ($n/4$ bits), is also fixed as an $n/4$-bit constant CON_L. Using the matrix separation technique, the partial function reduction technique is applicable to the left half of K'_2 represented as K'_{2L}. Specifically, let an $n/2$-bit variable K''_2 be $K''_2 = M(S'(K'_{2L} \oplus CON_L)|\mathbf{0}^{n/4})$, where S' consisting of $m/2$ S-boxes and $\mathbf{0}^{n/4}$ denotes $n/4$ bits of 0. Since K''_2 is linearly inserted in round 2 by the matrix separation, it is possible to move to L_7 (see Fig. 9-(a)).

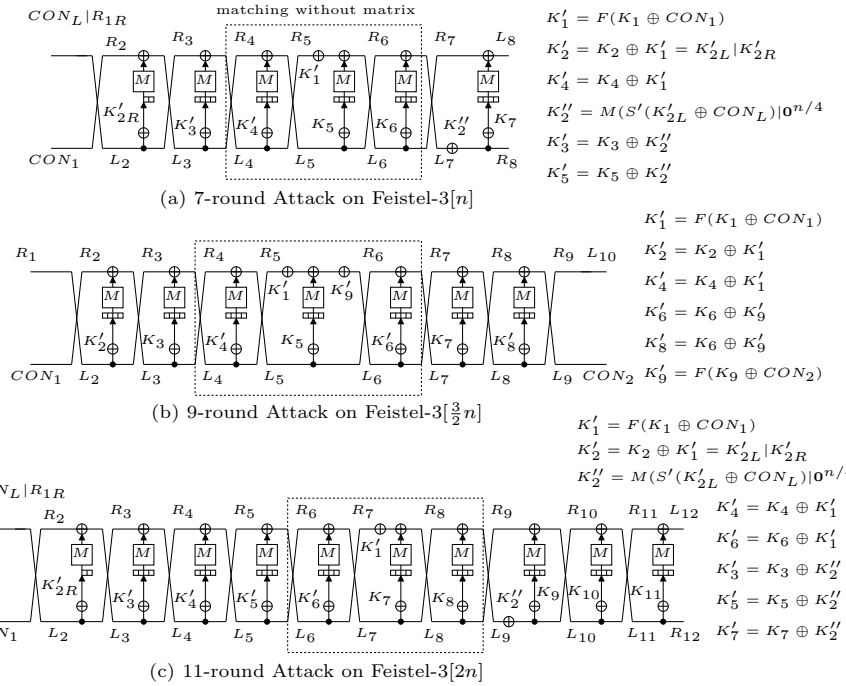
The matching without matrix technique is applied to the three consecutive rounds from rounds 4 to 6. In the forward and backward computations, $(L_4|R_4)$ and $(L_7|R_7)$ are computable from (K'_{2R}, K'_3) and (K'_{2L}, K'_7), respectively. Then, if ℓ bits of K'_4 and K_6 are guessed, an ℓ-bit matching is feasible, i.e., $\mathcal{K}_{(1)} \in \{K'_{2R}, K'_3, K'_{4,a}\}$ and $\mathcal{K}_{(2)} \in \{K'_{2L}, K'_7, K_{6,a}\}$, where $(1 \le a \le m)$, and $K'_{4,a}$ and $K_{6,a}$ denote arbitrary ℓ bits data of K'_4 and K_6, respectively.

Since $|\mathcal{K}_{(1)}| = |\mathcal{K}_{(2)}| = 3/2 \cdot n/2 + \ell$ and the matching size is ℓ bits, $N_1 = (3n/2 + 2\ell)/\ell$ plaintext/ciphertext pairs are required to determine $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$. The complexity of determining $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ is estimated as $C_{comp} = \max(2^{3n/4+\ell}, 2^{3n/4+\ell}) \times N_1$. After that, we are able to determine the other bits for finding all subkey bits by using a simple MITM attack on the remaining K'_4 and K_6, and K'_1 and K_5, respectively.

Therefore, the whole time complexity is estimated as $2^{3n/4+\ell} \times N_1$. Due to $k = n$, the required complexity $2^{3k/4+\ell} \cdot N_1$ is less than 2^k. The required data is $N_1 = (3n/2 + 2\ell)/\ell$ chosen plaintexts, and the memory is $2^{3n/4+\ell} \cdot N_1$ words.

5.4 Key Recovery Attack on 11-round Feistel-3[$2n$]

Similarly to the attack on the 7-round Feistel-3[n], chosen plaintexts in the form of $P = (L_1|R_{1L}|R_{1R})$ $= (CON|CON_L|R_{1R})$ are used. Then two more rounds can be added to both forward and back-



(a) 7-round Attack on Feistel-3[n]

(b) 9-round Attack on Feistel-3[$\frac{3}{2}n$]

(c) 11-round Attack on Feistel-3[$2n$]

Fig. 9. Key Recovery Attacks on Feistel-3 Ciphers

ward directions due to increasing the master key size. Thus, an 11-round attack is constructed. For the detailed parameters, see Table 2 and Fig. 9.

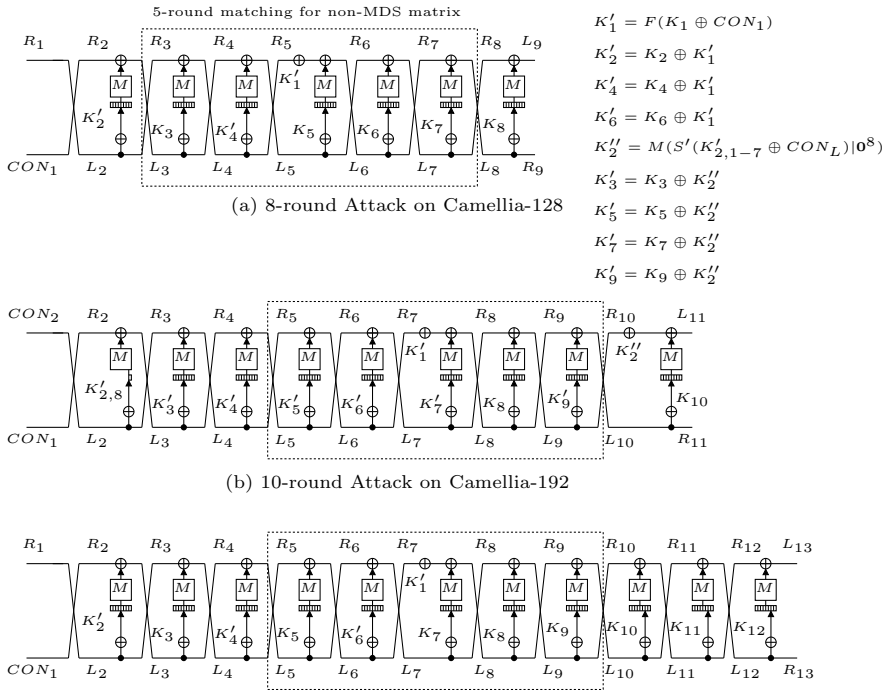
5.5 Key Recovery Attack on 9-round Feistel-3[$\frac{3}{2}n$]

As shown in Fig. 9-(b), for the 9-round Feistel-3[$\frac{3}{2}n$], the function reduction is applied to both directions combined with the matching without matrix to the rounds 4 to 6, since the middle of the matching is odd indexed round. Thus, a key recovery attack is constructed in a straightforward way, unlike the attacks on Feistel-3[n] and -3[$2n$].

In this attack, since $|\mathcal{K}_{(1)}| = |\mathcal{K}_{(2)}| = 2n/2 + \ell$ and the matching size is ℓ bits from Fig. 9-(b), $N_2 = (2n + 2\ell)/\ell$ plaintext/ciphertext pairs are required to determine $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$. Such pairs in the form of $L_1 = CON_1$ and $R_{10} = CON_2$ are found from $N_2!^{1/N_2} \cdot (2^{n/2})^{N_2 - 1/N_2}$ chosen plaintext/ciphertext pairs. Note that, if the number of required chosen plaintext/ciphertext pairs, which depends on the parameter n and ℓ, is more than $n/2$, the partial function reduction technique can be applied to L_1 and R_{10}. Otherwise, another attack approach is required for this variant. Moreover, if the function reduction is used only in the forward direction, an 8-round attack with extremely low data complexity is derived (see Table 2).

5.6 Application to Reduced Camellia

In order to demonstrate the usefulness and versatility of our approach on Feistel-3, we apply our attack to the reduced version of Camellia block cipher [5], which is Camellia without FL/FL^{-1} functions and key whitenings. Camellia is a Feistel block cipher whose F-function is the SP-type F-function consisting of eight 8-bit S-boxes followed by an 8×8 matrix operation. Thus, our attacks on the Feistel-3 cipher presented in the previous section are directly applicable to the 7/9/11-round reduced Camellia-128/192/256. Note that since our attack does not depend on the key scheduling function, the attack works on any key scheduling function even ideal. Furthermore, by exploiting the low diffusion property on the matrix used in Camellia, we develop the advanced five round matching technique. Then we present low-data complexity attacks requiring less than 60 plaintext/ciphertext pairs on the 8/10/12-round reduced Camellia-128/192/256 without FL/FL^{-1} and key whitenings.



(a) 8-round Attack on Camellia-128

(b) 10-round Attack on Camellia-192

(c) 12-round Attack on Camellia-256

Fig. 10. Key Recovery Attacks on Reduced Camellia-128/192/256

Five Round Matching for Non-MDS Matrix. Let us consider five consecutive rounds of the Camellia whose input and output are represented as $(L_i|R_i)$ and $(L_{i+5}|R_{i+5})$, respectively. By using the three-round matching without matrix technique in the middle, the following equation holds.

$$S(L_{i+1} \oplus K_{i+1}) \oplus M^{-1}(L_i) = S(R_{i+4} \oplus K_{i+3}) \oplus M^{-1}(R_{i+5}).$$

Since the S-box layer consists of eight 8-bit S-boxes, by guessing two bytes of subkeys with the same byte position $K_{i+1,j}$ and $K_{i+3,j}$, the 8-bit matching is possible if the same indexed 8 bits data $L_{i+1,j}$ and $R_{i+4,j}$ are also known. Since $L_{i+1} = M(S(L_i \oplus K_i)) \oplus R_i$ and $R_{i+4} = M(S(L_{i+5} \oplus K_{i+4})) \oplus R_{i+5}$, all bits of K_i and K_{i+4} are required to be guessed to obtain any byte of L_{i+1} and R_{i+4} if the underlying matrix M is optimal (i.e., MDS matrix). However, for Camellia, the 8 bits data $L_{i+1,j}$ and $R_{i+4,j}$ are derived by guessing corresponding $40 (= 8 \times 5)$ bits of K_i and K_{i+4} when $(5 \leq j \leq 8)$, since Camellia utilizes non-MDS matrix (See [5] for the details of the matrix used in Camellia). For example, $L_{i+1,5}$ and $R_{i+4,5}$ are derived from $K_{i,p} (p \in \{1, 2, 6, 7, 8\})$ and $K_{i+4,q} (q \in \{1, 2, 6, 7, 8\}$, respectively. Therefore, the number of key bits to be guessed for the 5-round matching in each direction is reduced from 128 bits $(= 64 \times 2)$ to 48 bits $(= 8 + 40)$.

Key Recovery Attack on 8-Round Reduced Camellia-128. Let us consider the 8-round reduced Camellia-128. In order to use the function reduction technique in the forward process, we collect chosen plaintexts in the form of $L_1 = CON_1$.

The five round matching for non-MDS matrix technique is used from rounds 3 to 7. In the forward and backward computations, $(L_3|R_3)$ and $(L_8|R_8)$ are computable by using $K'_2 (= K_2 \oplus K'_1)$ and K_8, respectively. Then, for the 8-bit matching, 8 bits subkey $K'_{4,a}$ and the corresponding 40 bits of subkey K_3 in the forward computation are required to be guessed, where $K'_4 = K_4 \oplus K'_1$. Similarly, we need to guess 8 bits subkey $K_{6,a}$ and the corresponding 40 bits of subkey K_7 in the backward computation. In other words, $\mathcal{K}_{(1)} \in \{K'_2, K'_{4,a}, 40 \text{ bits of } K_3\}$ and $\mathcal{K}_{(2)} \in \{K_8, K_{6,a}, 40 \text{ bits of } K_7\}$, where $5 \leq a \leq 8$.

Since $|\mathcal{K}_{(1)}| = |\mathcal{K}_{(2)}| = 112$ and the matching size is 8 bits, $28 (= (112 + 112)/8)$ plaintext/ciphertexts are sufficient to determine $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$. The complexity of determining $\mathcal{K}_{(1)}$ and $\mathcal{K}_{(2)}$ is estimated as $C_{comp} = \max(2^{112}, 2^{112}) \times 28 \approx 2^{117}$. After that, we are able to determine

Table 3. Summary of Key Recovery Attacks on Reduced Camellia-128/192/256 without FL/FL^{-1} Functions and Key Whitenings

Target	# Attacked Rounds	Attack Type	Time	Memory	Data	Reference
Camellia-128	8	Meet-in-the-Middle	2^{117}	2^{117}	**28**	Sect. 5.6
Camellia-192	10	Meet-in-the-Middle	2^{190}	2^{174}	**44**	Sect. 5.6
Camellia-256	12	Meet-in-the-Middle	2^{246}	2^{246}	**60**	Sect. 5.6

the other bits for finding all subkeys by using the simple MITM attack on the remaining 24 bits of K_3 and K_7, and 56 bits of K'_4 and K_6 in the forward and backward computations, respectively. Therefore, the whole complexity is estimated as $2^{117}(\approx 2^{117} + 2^{80})$. The required memory is 2^{117} words, and the required data is only 28 chosen plaintext/ciphertext pairs (see Fig. 10-(a)).

Key Recovery Attack on 12-Round Reduced Camellia-256. Similarly to the attack on the reduced Camellia-128, for the reduced Camellia-256, the five round matching for non-MDS matrix technique is used. Since two more rounds can be appended to each direction, a 12-round attack is constructed (see Fig. 10-(c) and Table 3).

Key Recovery Attack on 10-Round Reduced Camellia-192. In this attack, in order to utilize the function reduction technique in conjunction with the matrix separation technique, we collect chosen plaintexts in the form of $L_1 = CON_1$ and $R_{1,1-7} = CON_L$, where $R_{1,1-7}$ denotes the left 56 bits of R_1 and CON_L is a 56-bit constant. Then $K'_1 = F(CON_1 \oplus K_1)$ is moved to R_7 by redefining $K'_p = K_p \oplus K'_1 (p = 2, 4, 6)$. In addition, the left 56 bits of K'_2 defined as $K'_{2,1-7}$ is also moved to R_{10} by using the partial function reduction technique. Namely, we assume that $K''_2 = M(S'(K'_{2,1-7} \oplus CON_L)|\mathbf{0}^8)$ is linearly inserted in round 2 and the remaining 8-bit subkey $K'_{2,8}$ is non-linearly inserted in round 2, where S' consists of seven 8-bit S-boxes.

The five round matching for non-MDS matrix technique is used from rounds 5 to 9. Here, $(L_5|R_5)$ and (L_{10}, R_{10}) are computed from $(K'_{2,8}, K'_3 (= K_3 \oplus K''_2), K'_4 (= K_4 \oplus K'_{1L}))$ and $(K'_{2,1-7}, K_9)$, respectively. For the 8-bit matching, $K'_{6,8}$ and the corresponding 40 bits of $K'_5 (= K_5 \oplus K''_2)$ are required to be guessed in the forward computation, where $K'_6 = K_6 \oplus K'_1$. Similarly, in the backward computation, $K_{8,8}$ and the corresponding 40 bits of $K'_9 (= K_9 \oplus K''_2)$ are required to be guessed. Namely, $\mathcal{K}_{(1)} \in \{K'_{2,8}, K'_3, K'_4, K'_{6,8}$, 40 bits of $K'_5\}$ and $\mathcal{K}_{(2)} \in \{K'_{2,1-7}, K_9, K_{8,8}$, 40 bits of $K'_9\}$. In this attack, the whole complexity to determine all subkey bits is estimated as $2^{190}(\approx 2^{190} + 2^{80})$. The required memory is $2^{174}(\approx 2^{168} \times 44)$ words, and the required data is only 44 chosen plaintext/ciphertext pairs (see Fig. 10-(b) and Table 3).

6 Discussion

In order to compare the numbers of attacked rounds by our attacks with the previous results, we consider key recovery attacks from a 5-round impossible differential distinguisher or a 5-round zero-correlation linear distinguisher on the Feistel ciphers employing bijective F-functions [6, 11]. Note that those distinguishers depend only on the structure of the cipher unlike the other distinguishers such as a differential and a linear distinguisher. When $k = n$, guessing $n/2$ bits subkey involved in the 6-th round, it is possible to construct a 6-round key recovery attack from the 5-round distinguishers. Similarly, for $k = 3n/2$ and $k = 2n$, a 7 and an 8-round key recovery attacks are constructed by additionally guessing $n/2$ and n bits subkeys, respectively. Compared to those results, our attacks are the best attacks with respect to the number of attacked rounds for Feistel-2[$2n$], -3[n], -3[$\frac{3}{2}n$] and -3[$2n$] as described in Table 1. Also, for Feistel-1[$2n$] and Feistel-2[$\frac{3}{2}n$], the same numbers of rounds are attacked by our approach. Especially, the attack on the 11-round Feistel-3[$2n$] greatly exceeds the number of attacked rounds given by the distinguisher based attacks. More importantly, Feistel-3[$2n$] structure is well used in concrete block ciphers such as a 128-bit block cipher taking a 256-bit key, e.g., Camellia-256.

In addition, thanks to the MITM approach, most of our attacks require an extremely small data complexity, in contrast to the classical statistical attacks such as the impossible differential

and zero correlation linear attacks that generally require huge amount of data. This implies that our attacks may work even if the number of queries to the encryption oracle is restricted. In fact, the similar approach, which is the low-data complexity attacks on AES, has already been studied in [13, 14]. Thus, our work is also regarded as the first evaluation results on the low-data complexity attacks on the Feistel schemes.

7 Conclusion

This paper has shown the improved generic key recovery attacks on Feistel schemes independent of the key scheduling function. The proposed approach is based on the all subkeys recovery attack. With several advanced techniques such as function reduction and key linearization, which basically reduce the number of involved subkey bits, we presented several new key recovery attacks on the Feistel schemes.

To demonstrate the usefulness and the versatility of our approach, we showed several attacks on the concrete block ciphers including CAST-128 and Camellia. Among them, we would like to stress that the presented attack on the 8-round reduced CAST-128 having more than 118 bits key is the best attack with respect to the number of attacked rounds. Since our approach is generic, it is expected to be applied to other Feistel-type block ciphers. We believe that our results are useful not only for a deeper understanding the security of the Feistel schemes, but also for designing an efficient block cipher such as a low-latency cipher. Moreover, we expect that our attacks could be improved by combining with the recent attack called sieve-in-the-middle attack [17].

References

1. C. Adams, "The CAST-128 encryption algorithm." RFC-2144, May 1997.
2. C. Adams, "Constructing symmetric ciphers using the CAST design procedure." *Des. Codes Cryptography*, vol. 12, no. 3, pp. 283–316, 1997.
3. K. Aoki and Y. Sasaki, "Preimage attacks on one-block MD4, 63-step MD5 and more." in *SAC* (R. Avanzi, L. Keliher, and F. Sica, eds.), vol. 5381 of *LNCS*, pp. 103–119, Springer, 2008.
4. K. Aoki, J. Guo, K. Matusiewicz, Y. Sasaki, and L. Wang, "Preimages for step-reduced SHA-2." in *ASIACRYPT* (M. Matsui, ed.), vol. 5912 of *LNCS*, pp. 578–597, Springer, 2009.
5. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis." in *Selected Areas in Cryptography* (D. R. Stinson and S. E. Tavares, eds.), vol. 2012 of *LNCS*, pp. 39–56, Springer, 2000.
6. E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials." in *EUROCRYPT* (J. Stern, ed.), vol. 1592 of *LNCS*, pp. 12–23, Springer, 1999.
7. E. Biham and A. Shamir, "Differential cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer." in *CRYPTO* (J. Feigenbaum, ed.), vol. 576 of *LNCS*, pp. 156–171, Springer, 1991.
8. A. Biryukov and I. Nikolic, "Complementing Feistel ciphers." in *FSE* (S. Moriai, ed.), vol. 8424 of *LNCS*, pp. 3–18, Springer, 2014.
9. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher." in *CHES* (P. Paillier and I. Verbauwhede, eds.), vol. 4727 of *LNCS*, pp. 450–466, Springer, 2007.
10. A. Bogdanov and C. Rechberger, "A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN." in *Selected Areas in Cryptography* (A. Biryukov, G. Gong, and D. R. Stinson, eds.), vol. 6544 of *LNCS*, pp. 229–240, Springer, 2010.
11. A. Bogdanov and V. Rijmen, "Linear hulls with correlation zero and linear cryptanalysis of block ciphers." *Des. Codes Cryptography*, vol. 70, no. 3, pp. 369–383, 2014.
12. J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçin, "PRINCE - a low-latency block cipher for pervasive computing applications - extended abstract." in *ASIACRYPT* (X. Wang and K. Sako, eds.), vol. 7658 of *LNCS*, pp. 208–225, Springer, 2012.
13. C. Bouillaguet, P. Derbez, O. Dunkelman, N. Keller, V. Rijmen, and P.-A. Fouque, "Low data complexity attacks on AES." *IEEE Transactions on Information Theory*, vol. 58, no. 11, pp. 7002–7017, 2012.
14. C. Bouillaguet, P. Derbez, and P.-A. Fouque, "Automatic search of attacks on round-reduced AES and applications." in *CRYPTO* (P. Rogaway, ed.), vol. 6841 of *LNCS*, pp. 169–187, Springer, 2011.

15. C. Bouillaguet, O. Dunkelman, G. Leurent, and P.-A. Fouque, "Another look at complementation properties." in *FSE* (S. Hong and T. Iwata, eds.), vol. 6147 of *LNCS*, pp. 347–364, Springer, 2010.
16. C. D. Cannière, O. Dunkelman, and M. Knežević, "KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers." in *CHES* (C. Clavier and K. Gaj, eds.), vol. 5747 of *LNCS*, pp. 272–288, Springer, 2009.
17. A. Canteaut, M. Naya-Plasencia, and B. Vayssière, "Sieve-in-the-middle: Improved MITM attacks." in *CRYPTO (1)* (R. Canetti and J. A. Garay, eds.), vol. 8042 of *LNCS*, pp. 222–240, Springer, 2013.
18. I. Dinur, O. Dunkelman, and A. Shamir, "Improved attacks on full GOST." in *FSE* (A. Canteaut, ed.), vol. 7549 of *LNCS*, pp. 9–28, Springer, 2012.
19. FIPS, "Data Encryption Standard." Federal Information Processing Standards Publication 46.
20. J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw, "The LED block cipher." in *CHES* (B. Preneel and T. Takagi, eds.), vol. 6917 of *LNCS*, pp. 326–341, Springer, 2011.
21. T. Isobe, "A single-key attack on the full GOST block cipher." *J. Cryptology*, vol. 26, no. 1, pp. 172–189, 2013.
22. T. Isobe and K. Shibutani, "All subkeys recovery attack on block ciphers: Extending meet-in-the-middle approach." in *Selected Areas in Cryptography* (L. R. Knudsen and H. Wu, eds.), vol. 7707 of *LNCS*, pp. 202–221, Springer, 2012.
23. J. Jean, I. Nikolic, T. Peyrin, L. Wang, and S. Wu, "Security analysis of PRINCE." in *FSE* (S. Moriai, ed.), vol. 8424 of *LNCS*, pp. 92–111, Springer, 2014.
24. M. Knezevic, V. Nikov, and P. Rombouts, "Low-latency encryption - is "lightweight = light + wait"?." in *CHES* (E. Prouff and P. Schaumont, eds.), vol. 7428 of *LNCS*, pp. 426–446, Springer, 2012.
25. L. R. Knudsen, "DEAL - a 128-bit block cipher." Technical Report 151, University of Bergen, Department of Informatics, Norway, Feb. 1998.
26. L. R. Knudsen and V. Rijmen, "Known-key distinguishers for some block ciphers." in *ASIACRYPT* (K. Kurosawa, ed.), vol. 4833 of *LNCS*, pp. 315–324, Springer, 2007.
27. C. Ohtahara, K. Okada, Y. Sasaki, and T. Shimoyama, "Preimage attacks on full-ARIRANG: Analysis of DM-mode with middle feed-forward." in *WISA* (S. Jung and M. Yung, eds.), vol. 7115 of *LNCS*, pp. 40–54, Springer, 2011.
28. J. Patarin, "Security of random Feistel schemes with 5 or more rounds." in *CRYPTO* (M. K. Franklin, ed.), vol. 3152 of *LNCS*, pp. 106–122, Springer, 2004.
29. Y. Sasaki, "Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool." in *FSE* (A. Joux, ed.), vol. 6733 of *LNCS*, pp. 378–396, Springer, 2011.
30. Y. Sasaki, "Preimage attacks on Feistel-SP functions: Impact of omitting the last network twist." in *ACNS* (M. J. Jacobson Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, eds.), vol. 7954 of *LNCS*, pp. 170–185, Springer, 2013.
31. Y. Sasaki and K. Yasuda, "Known-key distinguishers on 11-round Feistel and collision attacks on its hashing modes." in *FSE* (A. Joux, ed.), vol. 6733 of *LNCS*, pp. 397–415, Springer, 2011.
32. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An ultra-lightweight blockcipher." in *CHES* (B. Preneel and T. Takagi, eds.), vol. 6917 of *LNCS*, pp. 342–357, Springer, 2011.
33. H. Soleimany, C. Blondeau, X. Yu, W. Wu, K. Nyberg, H. Zhang, L. Zhang, and Y. Wang, "Reflection cryptanalysis of PRINCE-like ciphers." in *FSE* (S. Moriai, ed.), vol. 8424 of *LNCS*, pp. 71–91, Springer, 2014.
34. K. Suzuki, D. Tonien, K. Kurosawa, and K. Toyota, "Birthday paradox for multi-collisions." in *ICISC* (M. S. Rhee and B. Lee, eds.), vol. 4296 of *LNCS*, pp. 29–40, Springer, 2006.