Improved Known-Key Distinguishers on Feistel-SP Ciphers and Application to Camellia

Yu Sasaki[1], Sareh Emami[2], Deukjo Hong[3], and Ashish Kumar[4]

[1] NTT Corporation, Japan
sasaki.yu@lab.ntt.co.jp
[2] Macquarie University, Australia
sareh.emami@mq.edu.au
[3] The Attached Institute of ETRI, Korea
hongdj@ensec.re.kr
[4] Indian Institute of Technology Kharagpur, India
kumarashish.iitkgp@gmail.com

Abstract. This paper revisits previous known-key distinguishers on generic Feistel-SP ciphers based on rebound attacks. In this paper first we propose a new 5-round inbound phase that requires 2^c computations, while the previous work requires 2^{2c} computations (c is a size of the S-box). The new method also improves the number of rounds which can be attacked. Then, we apply the new procedure to Camellia. After several optimizations for Camellia, it is shown that collisions are efficiently generated against 9 rounds out of 18 rounds of Camellia-128 including FL and whitening layers in the compression function modes such as MMO and Miyaguchi-Preneel modes. The attack on Camellia is verified by a machine experiment and the generated results are presented in the paper.

Keywords: block cipher, Feistel-SP, Camellia, known-key, rebound.

1 Introduction

Block-ciphers are often used as building blocks of secret-less primitives such as hash functions, hence recently cryptographers have started to evaluate the security of block-ciphers as hash functions. Known-key distinguishers proposed by Knudsen and Rijmen [1] are the evidence of this approach, whereas a block-cipher becomes a fixed permutation for a fixed key. The known-key distinguisher efficiently detects non-ideal properties of a random instantiation of a fixed permutation, while the same properties cannot be observed in a random permutation with the same complexity. Knudsen and Rijmen presented a known-key distinguisher on 7-round Feistel ciphers. They also pointed out that their attack detects collisions in hashing modes such as MMO and Miyaguchi-Preneel modes.

At FSE 2011, Sasaki and Yasuda presented another known-key distinguisher on Feistel ciphers [2] with the rebound attack proposed by Mendel *et al.* [3]. They showed that 11 rounds could be attacked if the round function consists of the subkey addition, S-box applications and the permutation layer. In the

Table 1. Complexities against Feistel-SP Ciphers. Notations are defined in Sect. 2.

Complexities of Previous Work [2]

N	n	c	r	5-round inbound	11-round distinguisher	11-round half-collision	9-round collision	9-round $(N-c)$-bit coll	7-round coll
128	64	8	8	2^{16}	2^{16}	2^{24}	2^{32}	–	–
128	64	4	16	2^{16}	2^{16}	2^{24}	2^{32}	–	–
64	32	8	4	2^{16}	Impossible	Impossible	Impossible	2^{24}	2^{24}
64	32	4	8	2^{8}	2^{8}	2^{12}	2^{16}	–	–

Complexities of Our Attacks

N	n	c	r	5-round inbound	11-round distinguisher	11-round half-collision	9-round collision	9-round $(N-c)$-bit coll	7-round coll
128	64	8	8	2^{16}	2^{16}	2^{16}	2^{24}	–	–
128	64	4	16	2^{8}	2^{8}	2^{16}	2^{24}	–	–
64	32	8	4	2^{16}	Impossible	Impossible	2^{24}	2^{16}	Impossible
64	32	4	8	2^{8}	2^{8}	2^{12}	2^{16}	–	–

hashing modes, the distinguishers are exploited to apply collision attacks or its variants. Examples of the known-key attack were provided in [4,5,6,2].

A chosen-key distinguisher was firstly proposed by Biryukov and Nikolić [7]. These distinguishers are able to choose the key value. Hash functions are suitable subjects to apply chosen-key distinguishing attacks, since the attacker can control the key values on the hash functions. Several papers have discussed chosen-key distinguishers on block ciphers [8,9].

In this paper, firstly we revisit the known-key distinguisher against generic Feistel-SP ciphers by [2]. One of the core techniques of [2] is a 5-round inbound phase that requires 2^{2c} computations, where c is the size of the S-box. We show that complexity of the 5-round inbound phase can be improved to 2^c computations (i.e. square root of the previous complexity). The new technique makes the attack possible for more number of rounds. Summary of our results in comparison to the previous attacks is shown in Table 1.

In the second part of this paper, we apply the proposed attack on generic Feistel-SP ciphers to Camellia [10]. Camellia is not a plain Feistel-SP cipher due to the P operation, FL, and whitening layers, so the attack needs several modifications. We evaluate Camellia using 128-bit keys including the FL and whitening layers. The best related work were the key-recovery attacks by Lu *et al.* [11] and Li *et al.* [12], which recover the key for 10 rounds of Camellia-128[1]. However, the key-recovery attack does not indicate a faster collision attack than the birthday attack. We show several attacks on hashing modes of Camellia-128 including collisions for 9 rounds. The results are shown in Table 2.

2 Preliminaries

We introduce the following notations. Recall that many of the block ciphers are equipped with 128-bit or 64-bit blocks and use 8-bit or 4-bit S-boxes.

[1] After the submission, 11-round key recovery attacks have been reported [13,14].

Table 2. Summary of attack results on Camellia-128 with FL and whitening layers

Target Modes	#Rounds	Approach	Complexity	Target structure	Reference
Block-Cipher	10	Imp. Diff.	2^{118}		[11]
Hash Function	7	4-sum	2^{32}	⬚⬚■■■▯⬚⬚	Ours
Compression Function	9	Half-collision	2^{16}	⬚⬚⬚■■■▯⬚⬚⬚	Ours
Compression Function	9	4-sum	2^{40}	⬚⬚⬚■■■▯⬚⬚⬚	Ours
Compression Function	9	Collision	2^{48}	⬚▯⬚■■■■■▯⬚⬚	Ours

⬚', ■', and ▯' represent inbound round, outbound round, and FL layer, respectively.

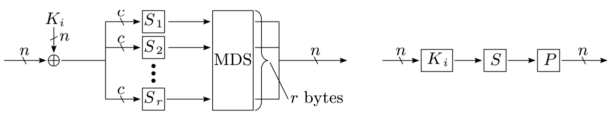


Fig. 1. Left: Detailed description of the SP round function **Right:** Simplified one

N: The block length of the cipher (in bits),
n: The word size in bits, equal to the size of the round function ($n = N/2$).
c: The size of an S-box in bits,
r: The number of S-box sequences, so that $r = n/c$.

SP Round Function. We denote ciphers with the Feistel network and an SP-round function by *Feistel-SP ciphers*, which is specified in Fig. 1.

Key XOR: The round-function input is XORed with a round key K_i.
S-box layer: Each input value is substituted with the output value by S-box. For simplicity, we assume that the S-box is designed to resist differential and linear cryptanalysis, like the one in AES [15,16].
Permutation layer: The linear diffusion is introduced to output sequences of the S-boxes. We make the assumption that the branch number of P is $r + 1$ e.g., a multiplication by an Maximum Distance Separable matrix.

Note that the assumptions on S-box and Permutation layers are not necessary. In fact, the branch number of the permutation layer of Camellia is not $r + 1$. In addition, Whirlpool [17] adopts a more biased S-box, however Lamberger *et al.* [18] showed that the rebound attack for Whirlpool can work similar to AES.

Hashing Modes. Preneel *et al.* [19] considered all possible configurations of a compression function built from a block cipher and proved that 12 modes are secure (providing the block cipher as a family of random permutations indexed by the key [20]). Given a block cipher E_K with a key K, the compression function for the so-called MMO mode computes H_i by $H_i = E_{H_{i-1}}(M_{i-1}) \oplus M_{i-1}$ for a message M_{i-1} and a previous chaining value H_{i-1}. While the Miyaguchi-Preneel mode computes H_i by $H_i = E_{H_{i-1}}(M_{i-1}) \oplus M_{i-1} \oplus H_{i-1}$. To build a hash function, the domain extender must be defined. Because our attack works on 1-block message, we only assume that the initial value H_0 is a fixed constant.

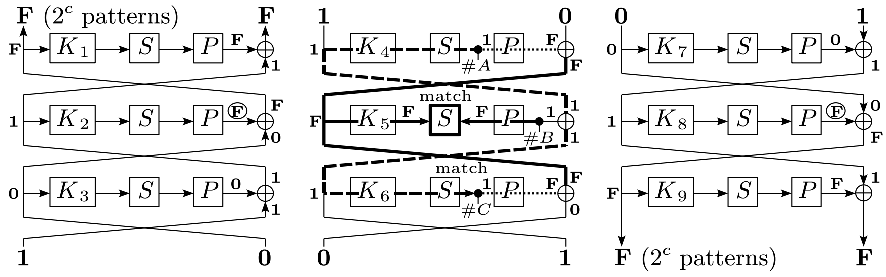


Fig. 2. A differential path for three round inbound and three round outbounds by [2]

3 Previous and Related Work

3.1 Previous Rebound Attack on Feistel-SP Ciphers

Similarly to Sasaki and Yasuda [2], we are going to use the following notations:

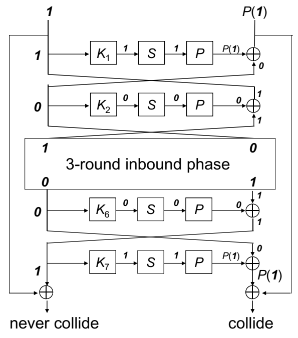
- **0**: A word where all the differences are equal to zero.
- **1**: A word where the difference on the j-th byte position is non-zero (we call it active) and the differences on the other byte positions are zeros,
- **F**: A word where all the differences are non-zeros.

The goal of the rebound attack is to find a pair of values that satisfies the truncated differential path. The rebound attack consists of two phases: *inbound* and *outbound*. In the *inbound* phase, the attacker computes differential path using the meet-in-the-middle strategy. In the *outbound* phase, the differential paths are extended outwards (to the input and output) using either deterministic or probabilistic arguments. A sample differential path including 3-round inbound and 3-round outbound phases is given in Fig. 2 (taken from [2]).

The inbound phase starts from the difference $(\mathbf{1}, \mathbf{0})$ and ends with $(\mathbf{0}, \mathbf{1})$. Sasaki and Yasuda [2] presented a procedure to find such a path through 3 and 5 rounds with a complexity of $r \cdot 2^{2c}$ and $r \cdot 2^{2c}$, respectively.

The outbound phase examines whether a solution of the inbound phase satisfies the outbound differential path or not. They showed that the difference $(\mathbf{1}, \mathbf{0})$ goes to the difference $(P(\mathbf{1}), \mathbf{F})$ after 3 rounds backward computation, with probability 1. Similarly, the difference $(\mathbf{0}, \mathbf{1})$ results the difference $(P(\mathbf{1}), \mathbf{F})$ after 3 rounds forward computation, with probability 1. The outbound phase sometimes only covers 2 rounds for each direction. In this case, with probability 1, the differences $(\mathbf{1}, \mathbf{0})$ and $(\mathbf{0}, \mathbf{1})$ on the inbound sides result differences $(\mathbf{1}, P(\mathbf{1}))$ and $(\mathbf{1}, P(\mathbf{1}))$ after 2 rounds backward and forward computations, respectively.

The 11-round distinguisher is built from a 5-round inbound part followed by 3-round outbound parts in both directions. It allows to find a pair of plaintext values whose difference is $(P(\mathbf{1}), \mathbf{F})$ and a pair of ciphertext values with the difference $(P(\mathbf{1}), \mathbf{F})$. The complexity of this attack is $r \cdot 2^{2c}$. On the other hand, the complexity of the generic attack is equal to the birthday attack on $n - c$ bits, which requires $2^{(n-c)/2}$ computations. Moreover this does not always work,



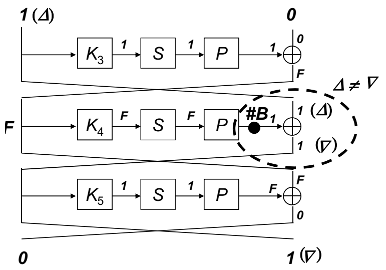


Fig. 3. Differential Path for 7R Collisions

Fig. 4. Inbound Phase with $\mathit{\Delta} \neq \nabla$

when $(N, c) = (128, 4)$, two bytes must be activated to increase the degree of freedom, and the complexity is higher, 2^{4c} in this specific case.

The 11-round distinguisher was exploited in [2] to attack the hashing modes. In the MMO and Miyaguchi-Preneel modes, the plaintext and ciphertext are XORed. Hence, if the left half of the differences in the 11-round distinguisher is cancelled, the half-state collision is obtained. Several other attacks were presented in [2]. The attack complexities are summarized in Table 1.

4 New Known-Key Distinguishers on Feistel-SP Ciphers

First of all, we explain the impossibility of the previous 7-round collision attack in Sect. 4.1. Then an improved 5-round inbound phase will be described in Sect. 4.2. Finally, in Sect. 4.3, we show that 4-sums can be detected even on the full state of hashing modes with 11-round Feistel-SP ciphers.

4.1 Flaw of Previous 7-Round Collisions for $(N, c) = (64, 8)$

It was claimed that collisions could be obtained for 7 rounds with parameters $(N, c) = (64, 8)$ in the MMO and Miyaguchi-Preneel modes [2, Sect.4.4]. Fig. 3 illustrates the attack with 3-round inbound phase and two (2-round) outbound phases. The attacker first generates a pair of values satisfying the differences of the inbound phase, namely $(\mathit{\Delta}, 0) \xrightarrow{\text{3R}} (0, \nabla)$. Through the outbound phase, the plaintext difference and ciphertext difference becomes $(\mathit{\Delta}, P(\mathit{\Delta}'))$ and $(\nabla, P(\nabla'))$. It was claimed that $\mathit{\Delta} = \nabla$ and $\mathit{\Delta}' = \nabla'$ are satisfied with probability 2^{-2c}. However, we prove that $\mathit{\Delta} = \nabla$ is an impossible event and always $\mathit{\Delta} \neq \nabla$.

Lemma 1. *Given a 7-round Feistel-SP cipher, the collision attack based on the rebound attack with 3-round inbound phase and 2-round outbound phases always fails.*

Proof. Inside the 3-round inbound phase in Fig. 4, to obtain $\Delta = \nabla$, the difference right after the P-layer in the middle round (denoted by $\#B$) must be $\mathbf{0}$. However, to perform the rebound attack, the difference at $\#B$ is always $\mathbf{1}$. Hence, $\Delta = \nabla$ is never satisfied. □

4.2 Improved 5-Round Inbound Phase with 2^c Computations

In this section, the improved inbound phase with 2^c computations is described. First we give an overview of the attack presented in [2].

First Inbound Phase: A pair of values which follow the differential path is obtained. 2^c solutions are generated with the workload of 2^c computations.

Second Inbound Phase: A pair of values for the 4th and 5th inbound rounds that follow the differential path is obtained. It costs 2^c computations.

Merge Inbound Phase: All possible solutions of the first and second inbound phases are combined in the third inbound round.

Validity Check Phase: Pairs of values generated by the merge inbound phase are computed from the third inbound round to the first and fifth inbound rounds, respectively. Then, 1-byte match is performed in both of the first and fifth rounds. The probability of each match is 2^{-c}, thus 2^{-2c} totally. Finally, by trying 2^{2c} combinations at the merge inbound phase, a pair which satisfies two matches is obtained.

In our improved attack, the differential path is developed using S-box differential profile. This approach leads to an attack with better complexity. The core of the improvement is the merging phase, in which we combine the results of the two inbound phases so that an n-bit match condition is always satisfied. This reduces the complexity of each step below by 2^c. In more details, we change the merge inbound phase so that the validity check is carried out in the third and fifth inbound rounds. We first choose a solution of the first inbound phase from 2^c candidates. Then, at the validity check in the third inbound round, we only choose solutions of the second inbound phase which satisfy the n-bit condition. Finally, the validity check at the fifth inbound round succeeds with probability 2^{-c}. By iterating this for 2^c candidates of the first inbound phase, we will succeed the validity check in the fifth round with a negligible cost.

Parallel Check of Differences in the First Inbound Phase. The goal of the first inbound phase is to find the differential path for the first two inbound rounds which is shown in Fig. 5. The original attack procedure is as follows [2]:

1. Search for several pairs of differences $\Delta\#A^j$ and $\Delta\#B^j$ such that $P(\Delta\#A^j)$ and $P^{-1}(\Delta\#B^j)$ have solutions for all S-boxes in the second inbound round (bold line in Fig. 5).
2. For each pair, exhaustively try 2^c values of $\#A^j$ which is denoted by x, and check if $S^{-1}(x) \oplus S^{-1}(x \oplus \Delta\#A^j)$ can cancel $\Delta\#B^j$ (broken line in Fig. 5).

Step 1 and Step 2 are independently performed. Thus the possibility of the 1-byte match at Step 2 is never considered during Step 1.

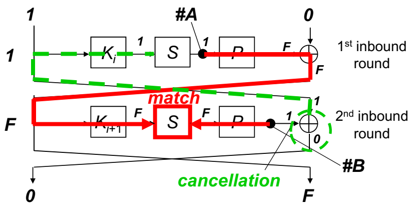


Fig. 5. Parallel check of the match of differences

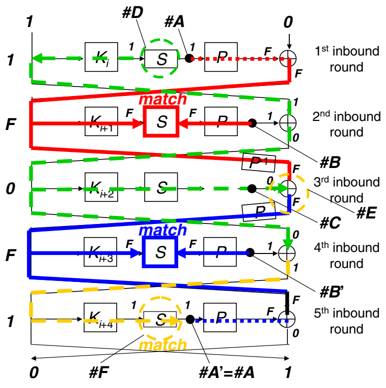


Fig. 6. Improved 5-round inbound phase

We notice that fixing the differences $\Delta\#A^j$ and $\Delta\#B^j$ for Step 1 immediately fixes the input and output differences of the active S-box for Step 2. This indicates that $(\Delta\#A^j, \Delta\#B^j)$ determined at Step 1 may not have any solution for $\Delta\#A^j \xrightarrow{S^{-1}} \Delta\#B^j$ at Step 2. Obviously, spending 2^c computations for such $(\Delta\#A^j, \Delta\#B^j)$ at Step 2 is meaningless. In other words, by only choosing $(\Delta\#A^j, \Delta\#B^j)$ which is known to have solutions at Step 1, the probability of the match at Step 2 is increased.

For any input and output differences of the S-box $(\Delta I, \Delta O)$ which is known to have solutions, $Pr\big[\big(S(x) \oplus S(x \oplus \Delta I)\big) = \Delta O\big] \geq 2^{-c+1}$, where x is a randomly chosen value. Hence, the probability increases from 2^{-c} to 2^{-c+1}. So the parallel check is applied to the match in the fourth and fifth inbound rounds (second inbound phase). Therefore, the entire probability increases from 2^{-2c} to 2^{-2c+2}.

Attack Procedure. For simplicity, we first assume that all S-boxes are identical and r and c satisfy $c \geq r + 1$ (It only happens for $(N, c) = (64, 8)$). The assumption is going to be weakened later on. In the following, we explain the attack procedure as illustrated in Fig. 6. In Fig. 6, the equivalent transformation is applied to the third inbound round. The first inbound phase is denoted by red lines. Similarly, the second inbound phase, the merge inbound phase, and the validity check are denoted by blue, green and yellow lines, respectively.

First Inbound Phase: Choose a difference at $\#A$ (i.e. $\Delta\#A$), and compute $P(\Delta\#A)$, which is an input to the S-layer in the second inbound round. Then, choose all differences at $\#B$ (i.e. $\Delta\#B$) such that the differential propagation through the active S-box in the first inbound round can have solutions, namely, $\exists x : S(x) \oplus S(x \oplus \Delta\#B) = \Delta\#A$. The number of such $\Delta\#B$ is approximately 2^{c-1}. For 2^{c-1} choices of $\Delta\#B$, compute $P^{-1}(\Delta\#B)$, which is an output of the S-layer in the second inbound round. Check if all S-boxes in the second inbound round have solutions. If the check succeeds, for each of the possible solutions, store the corresponding pair of values at state $\#B$ by computing P in the second inbound round and pair of values at state $\#E$ by computing the subkey XOR and then $P^{-1}(\cdot)$. Let T_1 be the table in which these values are stored. T_1 is expected to have 2^{c-1} entries.

Second Inbound Phase: Set $\Delta\#A' \leftarrow \Delta\#A$. Similar to the first inbound phase, compute 2^{c-1} solutions of the last two inbound rounds and store the paired values at $\#E$ in table T_2.

Merge Inbound Phase: For 2^{c-1} solutions of the first two inbound rounds stored in T_1 and all solutions (2 solutions on average) of the active S-box in the first inbound round (at state $\#D$), do the followings: Regarding only the active byte, compute the value up to the state $\#C$. If the computed value at state $\#E$ matches one of the entries in T_2, fix the solution for the last two rounds to this value. Go to the validity check with this value.

Validity Check Phase: Regarding only the active byte, compute the value up to the output of the active S-box in the fifth inbound round ($\#F$). If the computed difference matches $\Delta\#A'$, the paired values are the valid solutions. Otherwise, go back to the merge inbound phase.

Attack Evaluation. The complexity of the first inbound phase is 2^c 1-round computations in time and 2^{c-1} state in memory. The number of solutions over the P-layer in the second middle round is 2^{c-1}, and for each of them, 2 solutions are obtained for the active S-box in the first inbound round. Overall, the first inbound phase produces 2^c solutions with a cost of 2^c. The evaluation for the second inbound phase is the same. In the merge inbound phase, the number of trials is 2^c. The match at state #E succeeds with probability 2^{-c}. Since 2^c solutions are stored in T_2, we expect to find one match for each trial. Then, the matched result is computed up to state $\#F$ for the validity check. The success probability is 2^{-c}. Since the merge inbound phase is iterated 2^c times, we expect to find one solution of the validity check. The merge inbound phase and the validity check require 2^c 5-round computations. Finally, one solution of the inbound phase is computed with 2^c 1-round $+ 2^c$ 1-round $+ 2^c$ 5-round $=$ 2^c 7-round computations and 2^c state in memory.

Remarks for Other Parameters. The above attack can also be applied to other parameters with several minor changes. First of all we explain the case for $c = r$. $(N, c) = (128, 8)$ is included in this parameter. In this case, the freedom degrees will be slightly short in the first inbound phase (Step 5) because only $2^{c-1} = 2^{r-1}$ pairs can be examined. This problem is solved by running the first inbound phase for two different $\Delta\#A$. Hence, the complexity does not change from 2^c, and our procedure can also be applied to the parameter $c = r$.

The attack is also applied to the cases $(N, c) = (128, 4)$ and $(64, 4)$. We regard a group of two S-boxes as a big S-box with the size of $2c$ bits. This gives enough freedom degrees to find the match of differences over the S-layer. The attack becomes the same as $(N, c) = (128, 8)$ and $(64, 8)$.

Impact of the Improvement. Due to the improvement of the complexity of the 5-round inbound phase, many attacks presented in [2] are improved. The updated results are summarized in Table 1.

4.3 4-Sums on Compression Function Modes

The notion of *4-sum* represents 4 different inputs where the XOR-sum of the corresponding outputs is 0. The 4-sum is known to have many applications such as an attack on a random oracle instantiation [21] and on a signature scheme [22]. In the 11-round attack, the right halves of the plaintext and ciphertext do not have any property. Thus, the previous attack could only detect a non-ideal property on the left-half of the state. We explain that 4-sums can be obtained on the full state in compression function modes e.g. MMO and Miyaguchi-Preneel modes.

Each solution of the inbound phase produces a pair of outputs whose difference is the form $(P(\mathbf{1}), \mathbf{F})$. Let t_1 and t_2 be the paired solutions of the inbound phase, and $O(t_1)$ and $O(t_2)$ be corresponding outputs. Then, $O(t_1) \oplus O(t_2) = (P(\mathbf{1}), \mathbf{F})$. Assume that you have two pairs (t_1, t_2) and (t'_1, t'_2). The second-order difference $(O(t_1) \oplus O(t_2)) \oplus (O(t'_1) \oplus O(t'_2))$ becomes 0 if $O(t_1) \oplus O(t_2) = O(t'_1) \oplus O(t'_2)$. Because $O(t_1) \oplus O(t_2)$ takes 2^{c+n} possibilities, 4-sums can be generated with $2^{(c+n)/2}$ solutions of the inbound phase due to the birthday paradox.

5 Applications to Camellia and Its Hashing Modes

5.1 Specification of Camellia

Camellia was jointly designed by NTT and Mitsubishi Electric Corporation. It is widely standardized such as ISO [23], NESSIE [24], and CRYPTREC [25]. This paper attacks Camellia-128, where both of the key and block sizes are 128 bits.

Let M and K be 128-bit plaintext and secret key, respectively. Eighteen 64-bit round keys $K_1, \ldots, K_{18}$, four 64-bit whitening keys $kw_1, \ldots, kw_4$, and four 64-bit subkeys for the FL layer $kl_1, \ldots, kl_4$ are generated from K. Let L_r and R_r $(0 \le r \le 18)$ be left and right 64-bits of the internal state in each round. The plaintext is loaded into $L_0 \| R_0$ after the whitening operation, i.e. $L_0 \| R_0 \leftarrow M \oplus (kw_1 \| kw_2)$. $(L_{18} \| R_{18})$ is computed by $L_r = R_{r-1} \oplus F(L_{r-1}, K_r), R_r = L_{r-1}$ for $1 \le r \le 18$. Note that the FL and FL^{-1} functions are applied to L_r and R_r for $r = 6$ and 12. Finally, $(L_{18} \| R_{18}) \oplus (kw_3 \| kw_4)$ is the ciphertext.

Key Schedule. The key schedule takes 128-bit key K as input and firstly produces another 128-bit value K_A. In our known-key attacks, subkeys are randomly given values, thus the key schedule function is irrelevant. In chosen-key attacks, we only need to control the values of kl_1, kl_2, kl_3, and kl_4. kl_1 is the left 64-bits of $(K_A \lll 30)$ and kl_2 is the right 64-bits of $(K_A \lll 30)$. kl_3 is the left 64-bits of $(K_L \lll 77)$ and kl_4 is the right 64-bits of $(K_L \lll 77)$.

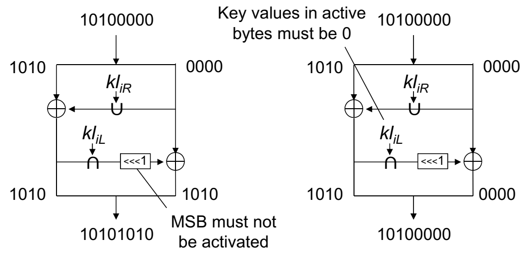


Fig. 7. Differential propagation through FL^{-1} for known-key (left) and chosen-key (right)

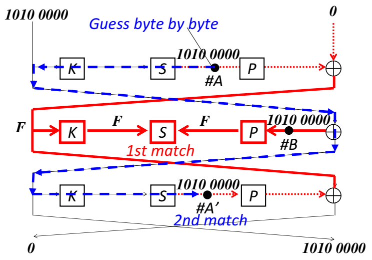


Fig. 8. 3-round inbound phase with 2 active bytes

Round Function. The round function consists of the subkey addition, S-layer and P-layer. For the S-layer, 4 S-boxes are defined. The DDTs for these S-boxes have the same property as the one for AES. Let $(z_1\|z_2\|\cdots\|z_8)$ be 64-bit values input to the P-layer. The output $(z'_1\|z'_2\|\cdots\|z'_8)$ is computed as follows. Here, $z[s,t,u,\cdots]$ means $z_s \oplus z_t \oplus z_u \oplus \cdots$. The branch number of P is only 5.

$$\begin{aligned} z'_1 &= z[1,3,4,6,7,8], & z'_3 &= z[1,2,3,5,6,8], & z'_5 &= z[1,2,6,7,8], & z'_7 &= z[3,4,5,6,8],\\ z'_2 &= z[1,2,4,5,7,8], & z'_4 &= z[2,3,4,5,6,7], & z'_6 &= z[2,3,5,7,8], & z'_8 &= z[1,4,5,6,7]. \end{aligned}$$

***FL* and *FL*$^{-1}$ Functions.** The FL function takes a 64-bit value $(X_L\|X_R)$ and a 64-bit subkey $(kl_L\|kl_R)$ as input and produces a 64-bit value $(Y_L\|Y_R)$ by computing $Y_R = \big((X_L \cap kl_L) \lll 1\big) \oplus X_R$ and $Y_L = (Y_R \cup kl_R) \oplus X_L$, where $\cap$, $\cup$, and $\lll 1$ are the logical AND, OR, and a left cyclic shift by 1 bit, respectively.

5.2 Applications to Camellia Hashing Modes

A Small Branch Number in the *P*-Layer. The attack in Sect. 4 assumes that the branch number of the P function is $r+1$, i.e. 9 for Camellia. Otherwise, $P(\mathbf{1})$ and $P^{-1}(\mathbf{1})$ may not become full active in the first inbound phase. However, the branch number of the P function in Camellia is 5.

We avoid this problem by activating more bytes at state $\#A$ and $\#B$ in Fig. 6. Since increasing the number of active bytes makes the attack inefficient, we need to choose active byte positions carefully. The conditions of active byte positions are as follows; 1) Active byte positions for $\#A$ and $\#B$ must be identical. 2) The active byte positions of $P(\Delta\#A)$ and $P^{-1}(\Delta\#B)$ must be identical.

We first search for a single active byte position satisfying these conditions. Since the value of r is 8, we have 8 possibilities. Unfortunately, any of the 8 cases cannot satisfy the conditions. We then search to find the positions of two active bytes. There are $\binom{8}{2} = 28$ possibilities. The result is shown in Table 3. We found 8 solutions for two-active byte differences. Hereafter, 5th and 7th bytes are activated. We also use the notation `10100000` to represent that only 5th and 7th bytes have differences in some states.

Table 3. Active-byte positions

	0th	1st	2nd	3rd	4th	5th	6th	7th
0th		×	✓	×	×	×	×	×
1st			×	✓	×	×	×	×
2nd				×	×	×	×	×
3rd					×	×	×	×
4th						✓	✓	✓
5th							✓	✓
6th								✓

Table 4. Sample solutions of the improved 5-round inbound phase for Camellia-128 (in hexadecimal)

	L	$\triangle_L$	R	$\triangle_R$
Input	3C0EF35D6F89AE61	0F00A20000000000	33F035B96274F068	0000000000000000
Output	32848AF48CE3D7EB	0000000000000000	A383FB0B17307503	2200810000000000
Input	1447452393AF07BF	D5006B0000000000	D7A0E862AF8343B9	0000000000000000
Output	FD6901A999C97E6C	0000000000000000	CE1249DCA3LC251B	C400D40000000000
Input	AB131279BE6E5342	3E00780000000000	6EF4FD4CA00881EB	0000000000000000
Output	798EB0992C1C6160	0000000000000000	301DBF0D730C71AD	0600780000000000
Input	62B5DB720A58E01D	33002D0000000000	95AAF7AD94613A12	0000000000000000
Output	8B24C8FAA65E8E33	0000000000000000	D2F68668ADE68225	C6001B0000000000

***FL* and $\boldsymbol{FL^{-1}}$ Functions.** These functions mix the difference of the half state. If they are inserted inside the inbound rounds, the differential path would be broken. Hence, we choose the starting round of our attacks carefully. As shown in Table 2, the FL-layer is located immediately after the inbound phase. This avoids inserting the FL-layer in the middle of the inbound phase.

In the following, we analyze the differential propagation through FL and FL^{-1}. Assume that the FL-layer is used only once immediately after the inbound phase. For the FL-layer, the form of the input differences is always $(\mathbf{0}, \texttt{10100000})$. It is obvious that $\mathbf{0}$ results in $\mathbf{0}$. Therefore, we only need to consider the right half of the state, where `10100000` is input to the FL^{-1} function. The differential propagation through FL^{-1} is described in Fig. 7. Distinguishers want to keep the number of active bytes low. Hence, in the known-key setting, distinguishers avoid activating the most significant bit (MSB) in each byte in order to prevent the difference from propagating through $\lll 1$ operation. This reduces the degrees of freedom while selecting differences. However, there is enough freedom for the attack to proceed as expected. Note that if the MSB in each active byte of the key kl_{iL} is 1, distinguishers can activate the MSB in each byte. Such weak keys exist with probability 2^{-2}, and the number of weak keys is $2^{128-2} = 2^{126}$. In the chosen key scenario, distinguishers choose the key value. Choosing one subkey value is trivially done with complexity 1 for any key value. This is because kl_i is a part of K or K_A. If it is a part of K, distinguishers directly choose the value. If it is a part of K_A, distinguishers firstly choose K_A and then invert it to K through the key schedule.

As shown in Table 2, distinguishers need to control two FL-layers in the chosen-key attacks with 5-round inbound phase. The FL-layer is inserted between the first and the second rounds of the backward outbound. According to Fig. 2, the form of the input differences to the inverse of the FL-layer is $(\mathbf{0}, \texttt{10100000})$. Therefore, we need to analyze the differential propagation of $(FL^{-1})^{-1}(\texttt{10100000})$, which is equivalent to $FL(\texttt{10100000})$. The analysis is the same as the previous one, and we omit it. As a result, if distinguishers can choose the values of 2 active bytes of the subkey, the form of the output difference is unchanged from `10100000`. Finally, distinguishers need to control two bytes in K_A and two bytes in K. We search for such keys by the brute force manner. The success probability of this event is 2^{-16}.

3-Round Inbound Phase and Its Application. The procedure of the 3-round inbound phase is heavily based on the one in previous work [2]. Our attack activates 2-bytes, however, we can keep the attack complexity unchanged with some optimization. The procedure is described in Fig. 8. The attack first chooses the differences of state #A and #B so that the differences can match over the S-layer in the second inbound round. If they match, one solution for all bytes are chosen and the corresponding paired values denoted by red in Fig. 8 are computed. Then, the attack searches for the value of each active byte at state $\#A$ that can satisfy the difference of the same byte position at state $\#A'$. Finally, the 3-round inbound phase is carried out with 2^8 computations, and once it is satisfied, up to 2^{40} solutions can be generated for free.

As listed in Table 2, the 3-round inbound phase has 3 applications. Due to the page limitation, we only explain 4-sums on the 7-round hash function. Hereafter we denote by **2** the difference form of `10100000`, and by **4** the difference form of `10101010`. The differential path that we use is $(\mathbf{2}, P(\mathbf{2})) \xrightarrow{2^{\mathrm{nd}}\mathrm{R}} (\mathbf{0}, \mathbf{2}) \xrightarrow{3^{\mathrm{rd}}\mathrm{R}} (\mathbf{2}, \mathbf{0})$ for the first outbound rounds, $(\mathbf{2}, \mathbf{0}) \xrightarrow{4^{\mathrm{th}}\mathrm{R}} (\mathbf{F}, \mathbf{2}) \xrightarrow{5^{\mathrm{th}}\mathrm{R}} (\mathbf{2}, \mathbf{F}) \xrightarrow{6^{\mathrm{th}}\mathrm{R}} (\mathbf{0}, \mathbf{2})$ for the inbound rounds, and $(\mathbf{0}, \mathbf{2}) \xrightarrow{FL} (\mathbf{0}, \mathbf{4}) \xrightarrow{7^{\mathrm{th}}\mathrm{R}} (\mathbf{4}, \mathbf{0}) \xrightarrow{8^{\mathrm{th}}\mathrm{R}}$ for the second outbound rounds. The number of active bytes increases by the FL-layer. After the feed-forward, the output should be $\Big(\mathbf{2}\oplus\mathbf{4}, P(2)\oplus P(4)\Big)$, which is in the space of $(\mathbf{4}, P(4))$. With the technique in Sect. 4.3, if we generate $2^{8*4} = 2^{32}$ pairs, differences on two pairs collide, and form a 4-sum. Since 3-round inbound phase generates up to 2^{40} solutions for free, our attack requires 2^{32} computations, which is equivalent to the information theoretic bound to generate a 4-sum $(2^{128/4})$, and faster than the generalized birthday attack.

5-Round Inbound Phase and Its Application. The procedure for the 5-round inbound phase is basically the same as the generic case while activating two S-boxes in Camellia is the same as activating 1 big S-box of the size $2c = 16$ bits. Hence, one solution of the 5-round inbound phase is obtained with 2^{16} in both time and memory. This leads to collisions on the 9-round compression function. The differential path that we use is $(\mathbf{2}, P(\mathbf{2})) \xrightarrow{6^{\mathrm{th}}\mathrm{R}} (\mathbf{0}, \mathbf{2}) \xrightarrow{FL} (\mathbf{0}, \mathbf{2}) \xrightarrow{7^{\mathrm{th}}\mathrm{R}} (\mathbf{2}, \mathbf{0})$ for the first outbound rounds, $(\mathbf{2}, \mathbf{0}) \xrightarrow{8^{\mathrm{th}}\mathrm{R}} (\mathbf{F}, \mathbf{2}) \xrightarrow{9^{\mathrm{th}}\mathrm{R}} (\mathbf{0}, \mathbf{F}) \xrightarrow{10^{\mathrm{th}}\mathrm{R}} (\mathbf{F}, \mathbf{0}) \xrightarrow{11^{\mathrm{th}}\mathrm{R}} (\mathbf{2}, \mathbf{F}) \xrightarrow{12^{\mathrm{th}}\mathrm{R}} (\mathbf{0}, \mathbf{2})$ for the inbound rounds, and $(\mathbf{0}, \mathbf{2}) \xrightarrow{FL} (\mathbf{0}, \mathbf{2}) \xrightarrow{13^{\mathrm{th}}\mathrm{R}} (\mathbf{2}, \mathbf{0}) \xrightarrow{14^{\mathrm{th}}\mathrm{R}} (\mathbf{2}, P(\mathbf{2}))$ for the second outbound rounds. The difference of the output of the compression function has the form $(\mathbf{2}, P(\mathbf{2}))$. Hence, if we generate such pairs 2^{32} times, the difference is **0** in one pair. With the improved 5-round inbound phase in Sect. 4.2, 1 solution of the inbound phase is generated with 2^{16} computations. Hence, 2^{32} solutions are generated with 2^{48} computations, which is faster than the birthday attack on a 128-bit value.

5.3 Experiments and Generated Data

To verify the attacks, we implemented the chosen-key 5-round inbound phase. First of all, a valid key is chosen to bypass the FL^{-1} layer after round 6. We find these keys in the expected time of about 2^{16} computations in average. Then, the chosen keys are used to find solutions of the new 5-round inbound phase. Table 4 shows some of the found solutions. Most of the solutions are found in less than 2^8 computations in our experiments. Therefore the experimental complexity is less than predicated in theory (i.e. 2^{16} computations).

6 Concluding Remarks

In this paper, we revisited the known-key attacks on generic Feistel-SP ciphers. Our main contribution is a new 5-round inbound phase which results in an improved complexity and works for a high number of rounds. Then with several modifications, the framework was applied to Camellia-128. Our results have been confirmed by computer simulations. We have presented several new attacks on the hashing modes of Camellia-128 including a collision attack on 9 rounds.

References

1. Knudsen, L.R., Rijmen, V.: Known-Key Distinguishers for Some Block Ciphers. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 315–324. Springer, Heidelberg (2007)
2. Sasaki, Y., Yasuda, K.: Known-Key Distinguishers on 11-Round Feistel and Collision Attacks on Its Hashing Modes. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 397–415. Springer, Heidelberg (2011)
3. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
4. Mendel, F., Peyrin, T., Rechberger, C., Schläffer, M.: Improved Cryptanalysis of the Reduced Grøstl Compression Function, ECHO Permutation and AES Block Cipher. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 16–35. Springer, Heidelberg (2009)
5. Minier, M., Phan, R.C.-W., Pousse, B.: Distinguishers for Ciphers and Known Key Attack against Rijndael with Large Blocks. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 60–76. Springer, Heidelberg (2009)
6. Sasaki, Y.: Known-Key Attacks on Rijndael with Large Blocks and Strengthening *ShiftRow* Parameter. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) IWSEC 2010. LNCS, vol. 6434, pp. 301–315. Springer, Heidelberg (2010)
7. Biryukov, A., Nikolić, I.: A new security analysis of AES-128. Rump session of CRYPTO 2009 (2009), http://rump2009.cr.yp.to/
8. Biryukov, A., Nikolić, I.: Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 322–344. Springer, Heidelberg (2010)
9. Nikolić, I., Pieprzyk, J., Sokołowski, P., Steinfeld, R.: Known and Chosen Key Differential Distinguishers for Block Ciphers. In: Rhee, K.-H., Nyang, D. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 29–48. Springer, Heidelberg (2011)
10. 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)

11. Lu, J., Wei, Y., Kim, J., Fouque, P.A.: Cryptanalysis of reduced versions of the Camellia block cipher. In: Selected Areas in Cryptography 2011 (2011), http://sac2011.ryerson.ca/SAC2011/LWKF.pdf
12. Li, L., Chen, J., Jia, K.: New Impossible Differential Cryptanalysis of Reduced-Round Camellia. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 26–39. Springer, Heidelberg (2011)
13. Liu, Y., Li, L., Gu, D., Wang, X., Liu, Z., Chen, J., Li, W.: New observations on impossible differential cryptanalysis of reduced-round Camellia. In: Fast Software Encryption 2012. Springer (to appear, 2012)
14. Bai, D., Li, L.: New Impossible Differential Attacks on Camellia. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 80–96. Springer, Heidelberg (2012)
15. Daemen, J., Rijmen, V.: AES Proposal: Rijndael (1998)
16. U.S. Department of Commerce, National Institute of Standards and Technology: Specification for the ADVANCED ENCRYPTION STANDARD (AES) (Federal Information Processing Standards Publication 197) (2001)
17. Rijmen, V., Barreto, P.S.L.M.: The WHIRLPOOL hashing function. Submitted to NISSIE (2000)
18. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound Distinguishers: Results on the Full Whirlpool Compression Function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009)
19. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 368–378. Springer, Heidelberg (1994)
20. Black, J.A., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 320–335. Springer, Heidelberg (2002)
21. Leurent, G., Nguyen, P.Q.: How Risky Is the Random-Oracle Model? In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 445–464. Springer, Heidelberg (2009)
22. Wagner, D.: A Generalized Birthday Problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)
23. International Organization for Standardization: ISO/IEC 18033-3, Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers (2005)
24. NESSIE-New European Schemes for Signatures, Integrity, and Encryption: Final report of European project IST-1999-12324 (1999), https://www.cosic.esat.kuleuven.be/nessie/Bookv015.pdf
25. CRYPTREC-C-Cryptography Research and Evaluation Committees: e-Government recommended ciphers list (2003), http://www.cryptrec.go.jp/english/index.html